# TRIBHUVAN UNIVERSITY

## INSTITUTE OF SCIENCE AND TECHNOLOGY

# A Project Report On

# Emotion Detection System Using Convolutional Neural Network

In partial fulfillment of the requirements for the Bachelor's Degree in

Computer Science and Information Technology

## Under the Supervision of

Santosh Dhungana

Asian School of Management and Technology

Gongabu, Kathmandu

## Submitted By

Bhuwan Joshi (5-2-37-12-2018)

Sampurna Giri (5-2-1181-118-2019)

Sangharsha Pyakurel (5-2-1181-119-2019)

## Submitted To

Tribhuwan University

Institute of Science and Technology

April  2024

# ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. A special gratitude we give to our project supervisor, **Mr. Santosh Dhungana,** in whose contribution stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. Furthermore, we would also like to acknowledge with much appreciation the crucial role of the staff of Asian School of Management & Technology, who gave the permission to use all required equipment and the necessary materials to complete the task. We are thankful and fortunate enough to get constant support from our seniors and every teaching staff of B.Sc. CSIT Department which helped us successfully complete our project. We would also like to extend our regards to all the nonteaching staff of CSIT department for their timely support. We have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices. Our appreciations also go to each and every one of our colleagues for their encouragement and support in developing the project. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere appreciation to all of them.

With respect,
Bhuwan Joshi (24094/076)
Sampurna Giri (24120/076)
Sangharsha Pyakurel (24121/076)

# ABSTRACT

This project proposes an innovative Emotion Detection System that leverages Convolutional Neural Networks (CNN) for live face detection and emotion analysis. The system aims to enhance user experience by accurately recognizing and classifying emotions such as happiness, sadness, anger, and more from facial expressions in real-time. The CNN model used in this system was trained on a diverse dataset of facial expressions, ensuring accurate emotion classification. During the training phase, the model achieved an impressive accuracy of 65%, indicating its capability to effectively recognize and classify emotions from facial expressions. Users can detect their face through a webcam, and the system utilizes a diverse dataset of facial expressions to ensure accurate emotion classification. Frontend was created using HTML, CSS and JavaScript, for backend Python was used. The workflow begins with real-time face detection, followed by emotion classification using the trained CNN model. This system can be utilized by anyone interested in understanding and analyzing emotions through facial expressions.

**KEYWORDS:** *Convolutional Neural Network, Live Face Detection, computer vision, real-time emotion analysis*

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CNN: | Convolutional Neural Network |
| CPU: | Central Processing Unit |
| CSS: | Cascading Style Sheets |
| CV: | Computer Vision |
| ED: | Emotion Detection |
| FER: | Facial Expression Detection |
| HTML: | Hyper Text Markup Language |
| HTTP: | Hyper Text Transfer Protocol |
| RAM: | Random Access Memory |
| UI: | User Interaction |
| UML: | Unified Modelling Language |
| URL: | Uniform Resource Locator |

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1 INTRODUCTION

## 1.1 Introduction

Recent research has highlighted the importance of emotion detection systems, which use human facial expressions to determine emotional states. These systems are critical for understanding how people respond to different stimuli and situations. Emotion detection systems provide vital insights into human emotions and behavior by utilizing advanced technologies like Convolutional Neural Networks (CNNs) and facial recognition algorithms.

Human connection comprises a variety of communication channels, including facial expressions, which provide valuable emotional indicators. Emotion detection systems take use of this by automatically studying facial expressions to accurately predict emotional states. Such systems have a wide range of applications, including surveillance, image analysis, security, and human-computer interaction.

Individual interaction may be an important component of existence. It shows precise details and a large amount of data in humans, whether in the form of body language, speech, facial expression, or emotions. Emotion detection is now widely employed in a wide range of applications, including smart card applications, surveillance, image database investigation, criminal, video indexing, civilian applications, security, and adaptive human-computer interaction with multimedia environments. Real-time emotion recognition is frequently facilitated by using CNN models trained on large datasets of facial expressions. These models can reliably describe emotions like happiness, sadness, and rage, allowing for a better understanding of human emotion and behavior.

The goal of this research is to create an emotion detection system that uses CNN to detect faces in real time and analyze emotions. The system seeks to enhance the user experiences by giving precise and fast insights into emotional states, which can have many applications.

Implementation of facial emotion detection is performed using Convolutional Neural Network which gives approximately 65% of accuracy.

Here real time face detection is done by developing a CNN (Convolutional Neural Network) model. This Project uses HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets), and JavaScript for the frontend and the backend of the project is handled using Python.

The models are created using the Python language in Jupyter notebook.

## 1.2 Problem Statement

In everyday computer use, machines often struggle to understand our emotions, relying primarily on our commands or predefined programming. This can lead to interactions that feel impersonal and distant. However, recent advancements in technology are changing this dynamic. With new capabilities, computers can now analyze our facial expressions, listen to our tone of voice, and even interpret the words we type to better gauge our emotional states. By harnessing these tools, computers can respond in a more human-like manner, adapting to our emotions in real-time. This means they can offer more empathetic and personalized interactions, enhancing the overall user experience. As a result, computer interactions become warmer and more engaging, bridging the gap between man and machine. This shift towards emotionally intelligent systems marks a significant step forward in human-computer interaction, promising a future where technology truly understands and connects with us on a deeper level.

## 1.3 Objectives

The Project aims to meet the following objectives:

- To create a platform that integrates the emotion detection system into various platforms.
- To establish a model that can classify seven different emotions.

## 1.4 Scope and Limitation

The Emotion Detection System aims to create a robust CNN-based model capable of properly identifying and classifying facial expressions in real time. This technology uses subtle changes in face features to properly predict the user's emotional state. Furthermore, it aims to create a consistent user experience by incorporating emotion recognition technology into a variety of applications, including music players, video streaming platforms, and communication tools.

The limitation of the project may include:

- Accuracy may be compromised if faces are partially obscured or expression are subtle.
- Proper lighting conditions may be necessary for effective face detection as detection in dark environment is challenging.

## 1.5 Development Methodology

The project adopts the Incremental Model for development. This model involves breaking down a complex project into smaller, self-contained modules or increments. Each increment represents a partial system with added functionality, facilitating gradual development. The iterative nature of this model allows for the sequential development, testing, and integration of increments into the evolving system.

**1. Iteration 1:**

- Integrate a Convolutional Neural Network (CNN) model for real-time emotion analysis.
- Train the model on a diverse dataset to accurately classify facial expressions.
- Implement a simple code for webcam integration.

**2. Iteration 1:**

- Develop and integrate accuracy rich face detection algorithm.
- Verify the system's ability to recognize and locate faces in real time video.
- Implement a simple user interface for capturing the emotion through webcam.

# CHAPTER 2 BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

The exploration of technology for detecting human emotions spans various disciplines, including psychology, neuroscience, and computer science. Facial expressions are deemed among the most reliable indicators of human emotions, making the detection of facial expressions using machine learning algorithms a subject of extensive research.

The Haar Cascade Classifier, introduced by Viola and Jones in 2001, has emerged as a prominent technique for facial expression detection. This classifier employs a machine learning-based object identification approach, training a classifier to identify objects in images using sets of positive and negative training images. In recent years, the Haar Cascade Classifier has found widespread application, including facial expression detection for emotions such as happiness, sadness, anger, surprise, fear. The integration of facial expression for emotion detection represents a novel approach that has garnered significant interest.

Machine learning tools like the Haar Cascade Classifier have been crucial in making facial expression detection better. This helps a lot in figuring out how people are feeling. The idea behind our emotion detection system is to use this technology to accurately spot and categorize emotions in people's faces.

This kind of system has a lot of potential uses. It could help keep an eye on people's mental health, make computers easier to use, give companies better info on what people like, make learning more effective, and even make places safer. By looking at facial expressions, these systems can give us a better understanding of how people feel and act. This means we can offer personalized help, make interactions more personal, make smarter choices, and improve safety in lots of different areas.

## 2.2 Literature Review

There has also been research done on the Emotion Detection System. According to one such research [1], In this study, a convolutional neural network (CNN) was developed to identify emotions from grayscale facial images. The CNN utilized a Conv-ReLU-Pool architecture with softmax loss function. L2 regularization were employed, with ADAM optimizer and learning rate decay used for optimization. The final validation accuracy achieved was 60.7%. Challenges included small image sizes and difficulty in discerning emotions, addressed partially through the use of saliency maps to highlight important image regions. Despite achieving a relatively high accuracy, the authors suggest that increasing network depth and filters could further enhance performance.

The development of emotion classification and detection software has garnered significant interest in recent years, particularly in the realm of affective computing and artificial intelligence. In a study by [2], the authors proposed an innovative approach to facial emotion recognition through artificial intelligence techniques. Their research focused on testing the software with images to detect a range of emotions, including sadness, fear, anger, surprise, disgust, and happiness. Furthermore, the authors conducted real-time tests using multimedia content to assess the accuracy of emotion detection. This study contributes to the growing body of literature exploring the efficacy of emotion recognition systems in diverse contexts, laying the foundation for further advancements in this field.

In a particular system [3], the network was trained with varying numbers of CNN layers and epochs. The best accuracy, was achieved with five layers and 300 epochs. Precision, Recall, F1-score, and Support metrics were analyzed for seven emotion types, showing superior performance of the proposed model. Comparison with state-of-the-art models like AlexNet, VGG, GoogleNet, and ResNet revealed the proposed CNN models' higher accuracy, despite the small and challenging dataset. The focus remained on designing a system with strong performance under limited data conditions.

Another system [4] algorithm's performance was initially tested on the extended Cohn–Kanade expression dataset, achieving a maximum accuracy of 45% due to its limited size. To improve efficiency, additional datasets and the author's own images were included, resulting in increased accuracy as dataset size grew. The CNN architecture was optimized for both the number of layers and filters, with optimal accuracy achieved at four layers and four filters per layer. Despite challenges with grayscale images affecting skin tone detection and orientation variations, the algorithm demonstrated successful emotion detection, achieving a high accuracy of 73%. However, limitations such as high computational demands and issues with facial hair were noted. Future work may address these limitations while exploring the algorithm's performance with larger datasets.

In another system [5] two CNN models were developed to recognize emotions from facial expressions using the FER 2013 dataset. Model 1 achieved 70.0% accuracy with four emotions and 62.2% with five emotions, while Model 2 reached 73.5% and 70.5% accuracy, respectively. Happy emotion had the highest precision, recall, and F-score, while neutral emotion performed the worst. The proposed models outperformed other approaches in the literature.

# CHAPTER 3 SYSTEM ANALYSIS

## 3.1 System Analysis

System analysis is the process of examining a system or organization to better understand its components, how they interact, and how they might be improved. It is a holistic approach that examines the system as a whole and determines the relationships between its components. The purpose of systems analysis is to identify issues and inefficiencies in the existing system and provide ideas for improvement.

### 3.1.1 Requirement Analysis

Requirement analysis is a vital part of software development that involves obtaining, analyzing, and documenting the project's needs and limitations. The requirement analysis approach for the Emotion Detection Project sought to identify the system's core features and capabilities, as well as any limits or limitations that would need to be addressed throughout development.

#### 3.1.1.1 Functional Requirements

The system should be able to handle a large number of images of user expressions, with minimal processing time. The system should provide a user-friendly interface for displaying the results. Following is the main functional requirement our system has:

**Face Detection**

Objective: Implement algorithms to recognize emotional state.

Specifications: Develop models or techniques capable of accurately identifying emotions (like happiness, sadness, anger, excitement).

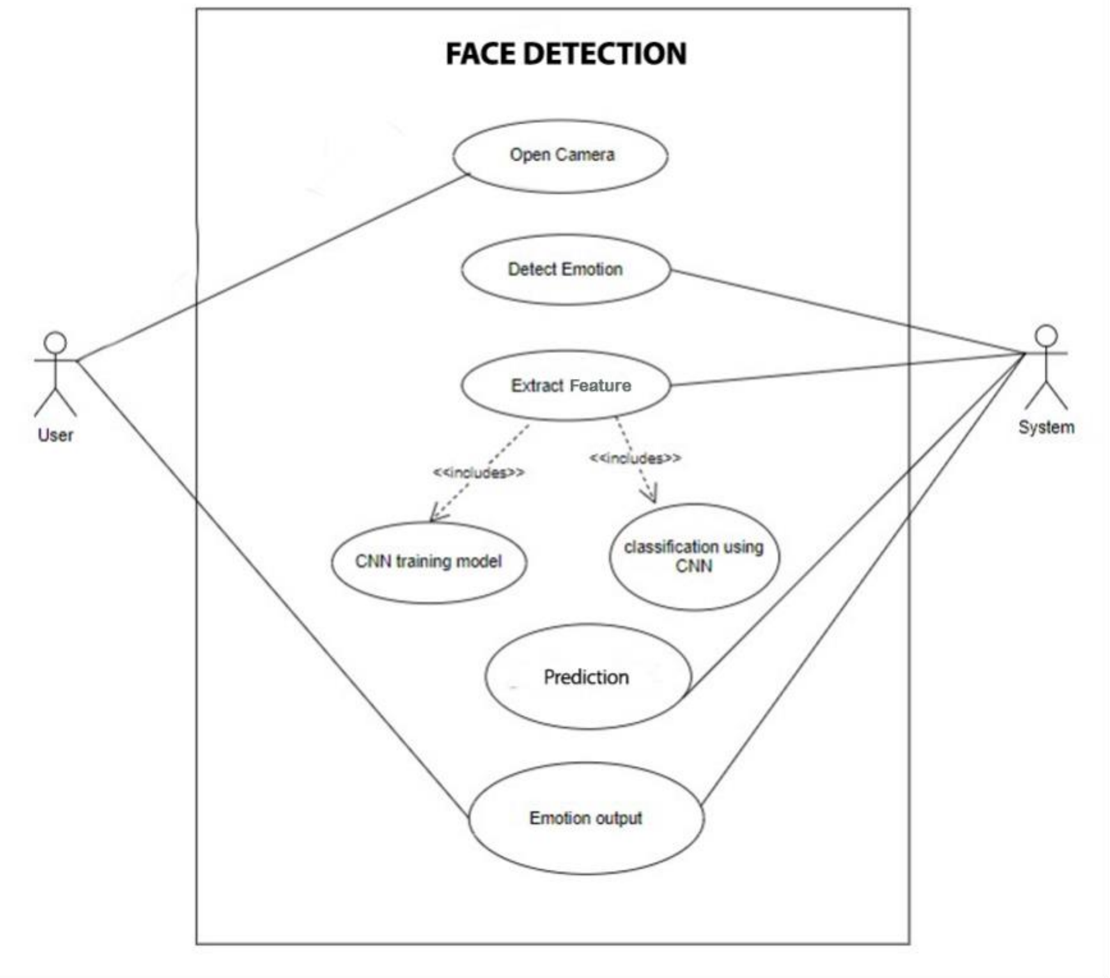


**Figure 3.1: Use case diagram of live face detection**

Above case diagram 3.1 gives a high-level picture of how users interact with a system and the important features it delivers. In the context of the Emotion Detection System, a use case diagram would depict the many actions that users can take and how the system responds. It comprises actors, use cases (Input, real-time prediction, user profile management and relationships.

#### 3.1.1.2 Non-Functional Requirements

Non-functional requirements are aspects of a system that describe how it should behave, rather than specific behaviors. These requirements often define the overall qualities and characteristics that the system must possess. Following are some non-functional requirement our system has:

**Performance**

Specify the expected response times for the system to recognize facial expressions and generate emotion. For example, the system should provide emotions within a few seconds of recognizing the user's facial expression.

**Scalability**

The system's capacity to manage a growing user base and the corresponding rise in data and processing load. It could involve defining the highest quantity of concurrent users that the system is capable of accommodating.

**Usability**

The system will be simple and build in a user-friendly interface such that anyone can understand and use it.

**Availability**

The system should be available to the user whenever they want to access.

### 3.1.2 Feasibility Study

Feasibility analysis, in simple words is an analysis and evaluation of a proposed project to ensure if it is technically, economically and operationally feasible. As the name suggests, a feasibility analysis is a study of the viability of an idea. It focuses on answering the essential question of "should this proposed project idea be proceeded?"

#### 3.1.2.1 Technical Feasibility

The technical feasibility involves the evaluation of the hardware, software, and other technology requirements for the construction of the project.

**Hardware Requirements:**

To run this project efficiently, a powerful CPU (Central Processing Unit) is employed to manage model execution, handle input data, and expedite training and inference processes. Furthermore, the project's smooth operation requires a system with a functional webcam.

**Software Requirements:**

This project requires a programming language like Python to construct Emotion Detection systems and image categorization algorithms, as well as machine learning libraries like Jupyter notebook for model training. Similarly, particular computer vision libraries, such as OpenCV (Open-Source Computer Vision Library), are required for activities like picture preprocessing and real-time video capture. The project's frontend is built with HTML, CSS, Bootstrap, and JavaScript, with Python serving as the backend language.

#### 3.1.2.2 Operational Feasibility

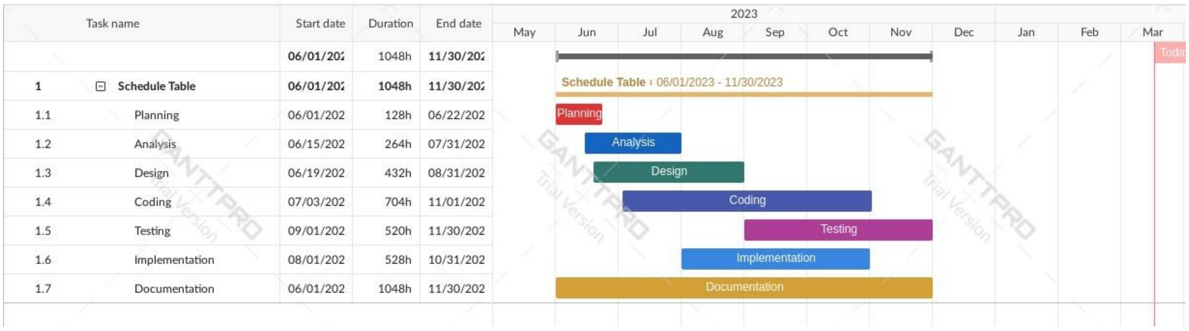
This system demonstrates operational feasibility by seamlessly integrating into user's routines with a user-friendly interface. It requires minimal additional resources, provides comprehensive documentation for easy adoption, and prioritizes privacy and security.

#### 3.1.2.3 Economic Feasibility

The system will be constructed at a minimum price. Almost every resource required for the system will be acquired from the internet. We will be able to do this project using our understanding of available languages and technologies. The dataset required for this project is available on internet. Once the dataset is collected, we will be able to train and test the model.

#### 3.1.2.4 Schedule Feasibility

The time given for the completion of this project was a whole semester. So, the project has enough time for completion for this project. Since, the people working on this project have some experience with web development strategies the project will not be so challenging. Hence, the project has been developed according to the following time schedule.



| | Task name | Start date | Duration | End date |
|---|---|---|---|---|
| | | 06/01/202 | 1048h | 11/30/202 |
| 1 | ⊟ Schedule Table | 06/01/202 | 1048h | 11/30/202 |
| 1.1 | Planning | 06/01/202 | 128h | 06/22/202 |
| 1.2 | Analysis | 06/15/202 | 264h | 07/31/202 |
| 1.3 | Design | 06/19/202 | 432h | 08/31/202 |
| 1.4 | Coding | 07/03/202 | 704h | 11/01/202 |
| 1.5 | Testing | 09/01/202 | 520h | 11/30/202 |
| 1.6 | Implementation | 08/01/202 | 528h | 10/31/202 |
| 1.7 | Documentation | 06/01/202 | 1048h | 11/30/202 |

**Figure 3.2: Gantt chart**

## 3.1.3 Analysis

### 3.1.3.1 Dataset

This Project used a special set of pictures called the FER-2013 Dataset. This set has 35,887 black-and-white pictures of people's faces showing different emotions like anger, disgust, fear, happiness, sadness, surprise, and calmness. The pictures are small, each only 48 by 48 pixels. These pictures are into two groups: 24,716 (68.9%) for training the computer program to recognize emotions and 7,178(31.1%) to test how well it learned. This dataset helps the project to understand human emotions by looking at faces.

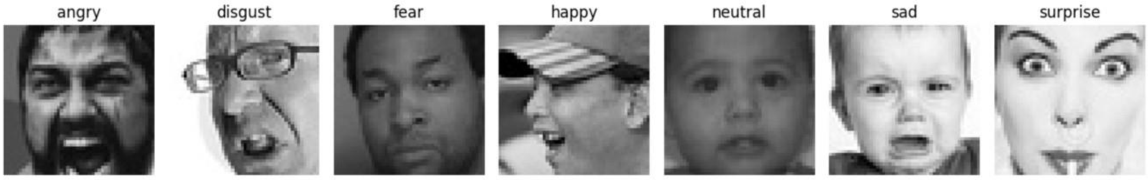


**Figure 3.3: Visualizing images**

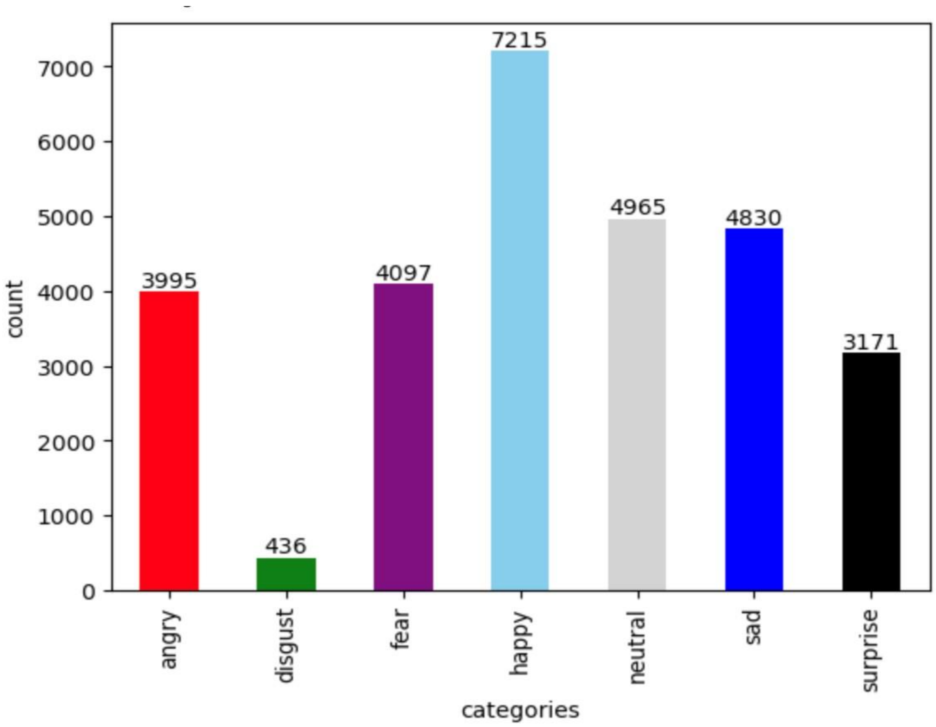


**Figure 3.4: Bar graph showing training data**

Above figure 3.4 shows how many pictures we have for each category. The x-axis is 'categories,' and the y-axis is 'count'. Looking at the graph, it's clear that 'happy' has the most pictures, and 'disgust' has the fewest. The number of images for other emotions is between 3000 and 4000.

### 3.1.3.2 Object Modeling using Class diagram

A class diagram in UML is a static structural diagram that illustrates a system's structure, including classes, properties, operations, and object relationships.

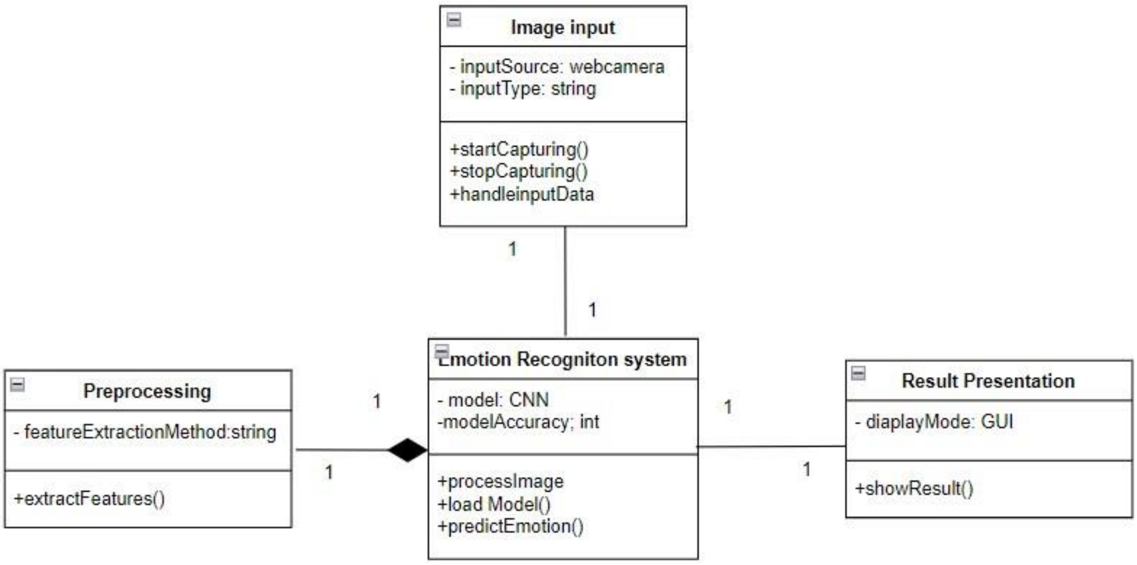


**Figure 3.5: Class diagram for Emotion Detection System**

Above class diagram 3.5 has four classes: Emotion Recognition System, Image Input, Preprocessing, and Result Presentation. The Image Input class enables users to send image input to the system. The preprocessing class is in charge of preparing images in order to extract features. The Result Presentation class is used to present the expected emotion on the UI. The Emotion Recognition System centralizes the operation and uses CNN for emotion recognition. It recognizes the emotion depicted in the image.

### 3.1.3.3 Object Modeling using Sequence diagram

Sequence diagrams are a type of UML (Unified Modeling Language) diagram used to visualize interactions between objects or components in a system over time.

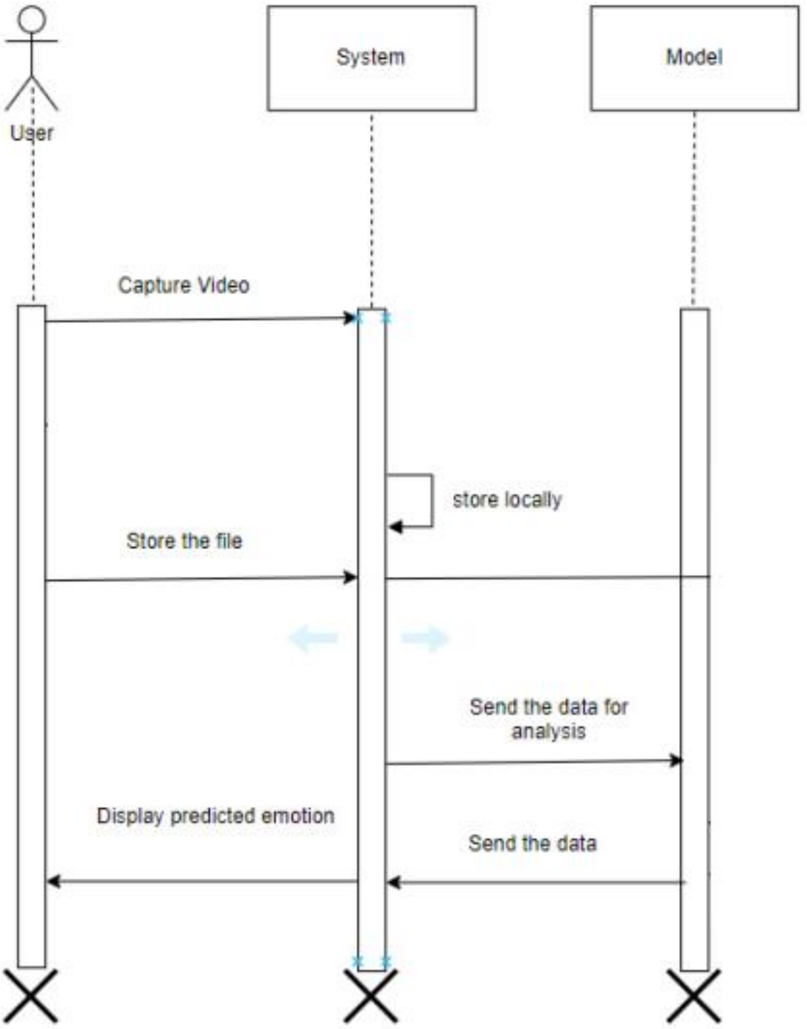


**Figure 3.6: Sequential diagram for Emotion Detection System**

### 3.1.3.4 Process Modeling using Activity Diagram

An activity diagram is a type of UML diagram that depicts the flow of control or a series of events in a system visually. It shows step-by-step work processes that support choice, iteration, and concurrency.

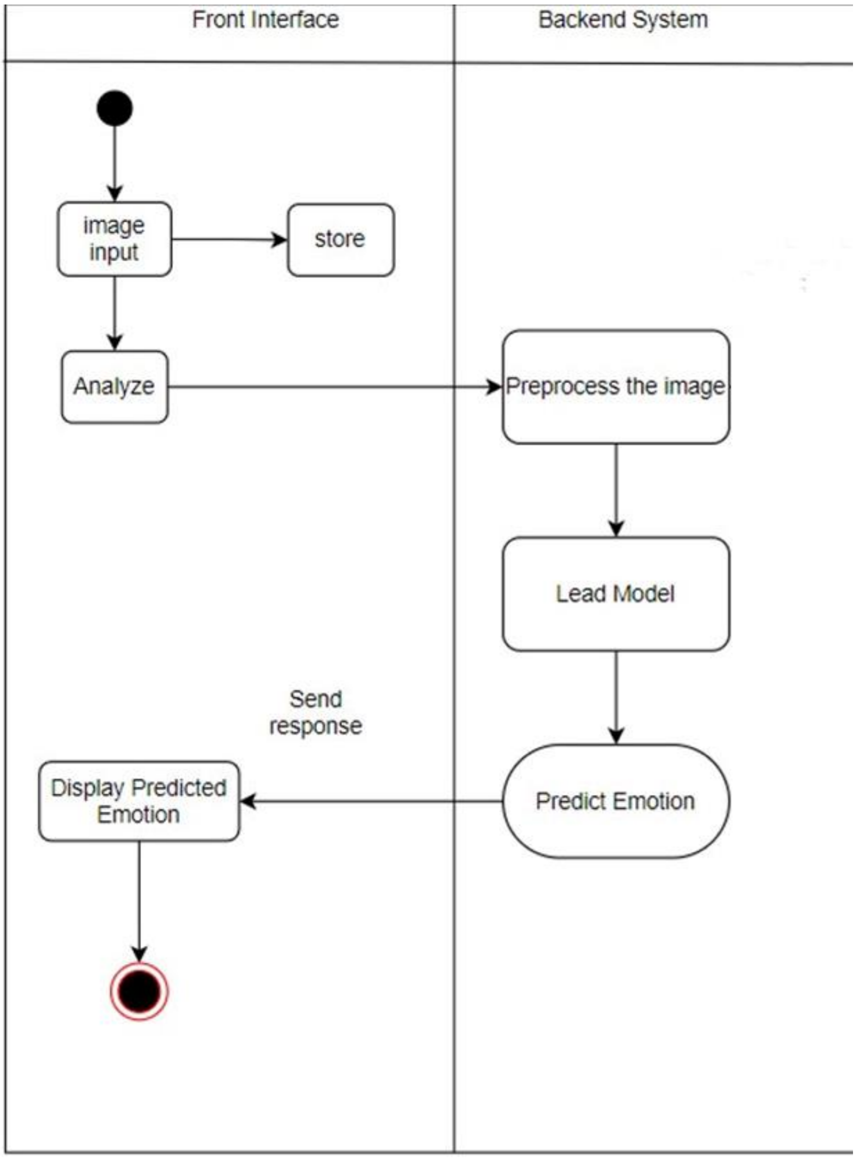


**Figure 3.7: Activity diagram for Emotion Detection System**

# CHAPTER 4 SYSTEM DESIGN

## 4.1 Workflow of system

The Emotion Detection System operates within a user-centric workflow, starting with the user's interaction. Utilizing facial expression recognition technology, the system captures live photos from the webcam to analyze the user's emotions. Initially, the system preprocesses the images from the dataset, enhancing their quality and standardizing their format. Following preprocessing, the system extracts features from these images, capturing key facial characteristics that are indicative of different emotional states. These features serve as inputs to a trained Convolutional Neural Network (CNN) algorithm, which has been developed specifically for the task of emotion detection. Through the CNN algorithm, the system classifies the extracted features and accurately predicts the user's emotional state. This comprehensive process ensures that the Emotion Detection System delivers timely and precise results, enhancing user experience.

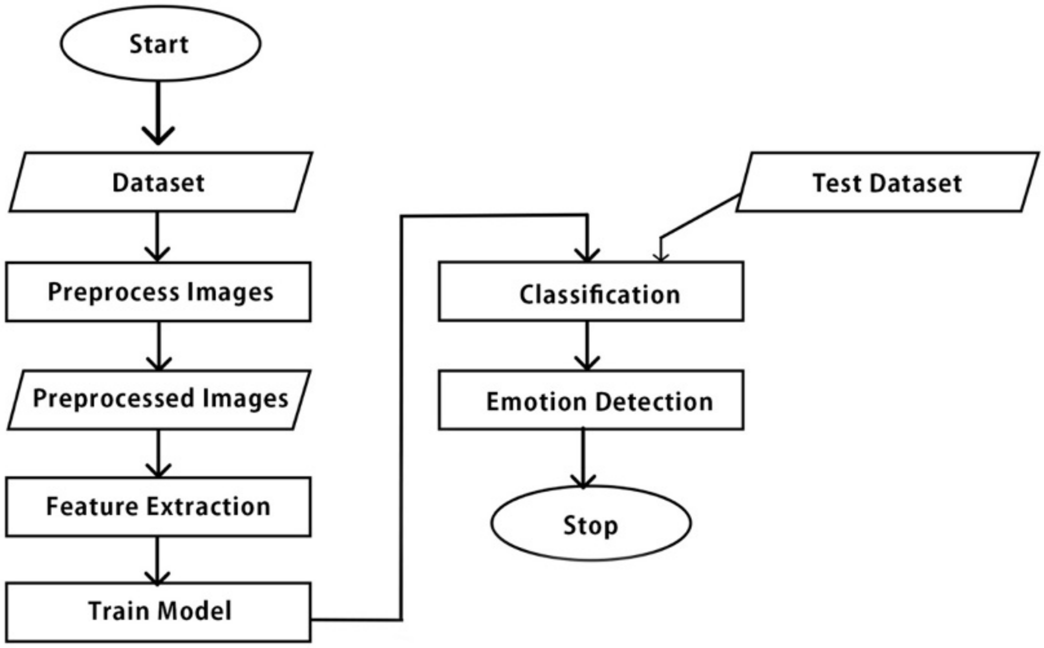


**Figure 4.1: Workflow architecture for Emotion Detection System**

### 4.1.2 Component Diagram

A component diagram is a sort of UML diagram that depicts the structural interactions and interdependence among the components of a software system. It demonstrates how software components are linked and interact with one another inside a system. Individual modules, libraries, executables, and other system pieces can all be represented as components.

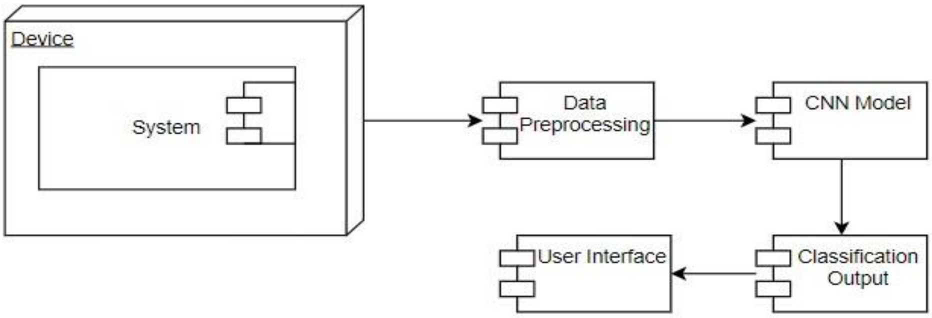


**Figure 4.2: Component Diagram**

### 4.1.3 Deployment Diagram

Deployment diagrams are UML structural diagrams that show the relationships between hardware and software components in the system, as well as the physical distribution of processing. In other words, deployment diagrams are used to visualize the topology of the physical components of the system where software components are deployed.

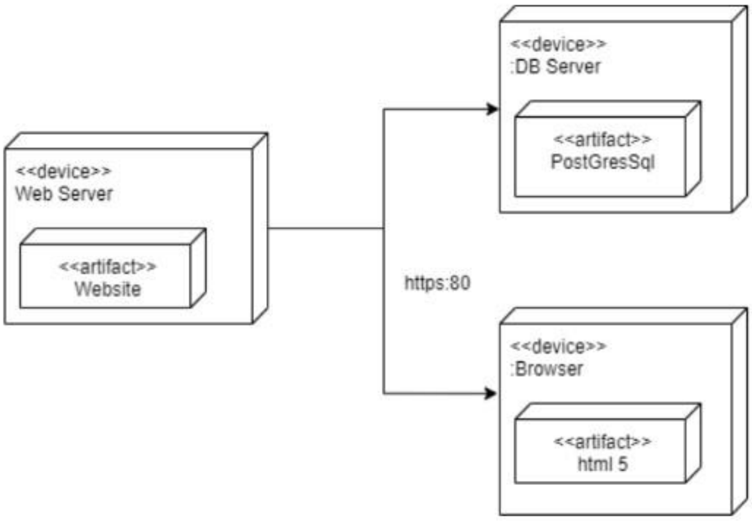


**Figure 4.3: Deployment Diagram**

## 4.2 Algorithm Details

The project implements the following algorithm:

**Convolution Neural Network**

One of the most common types of neural networks used for image categorization and recognition is convolutional neural networks. Convolutional neural networks are frequently utilized in various applications such as scene labeling, object identification, and face recognition. CNN takes an image as input, which is classified and process under a certain category. An image is viewed by the computer as an array of pixels, and its resolution is determined by this. It will see as h * w * d, where h = height, w = width, and d = dimension, depending on the image resolution. For example, an RGB image is a 6 * 6 * 3 matrix array, while a grayscale image is a 4 * 4 * 1 matrix array. Each input image in CNN will go through a series of convolution layers, pooling, fully linked layers, and filters (also known as kernels). After that, the Soft-max function is used to classify an object with probabilistic values between 0 and 1.

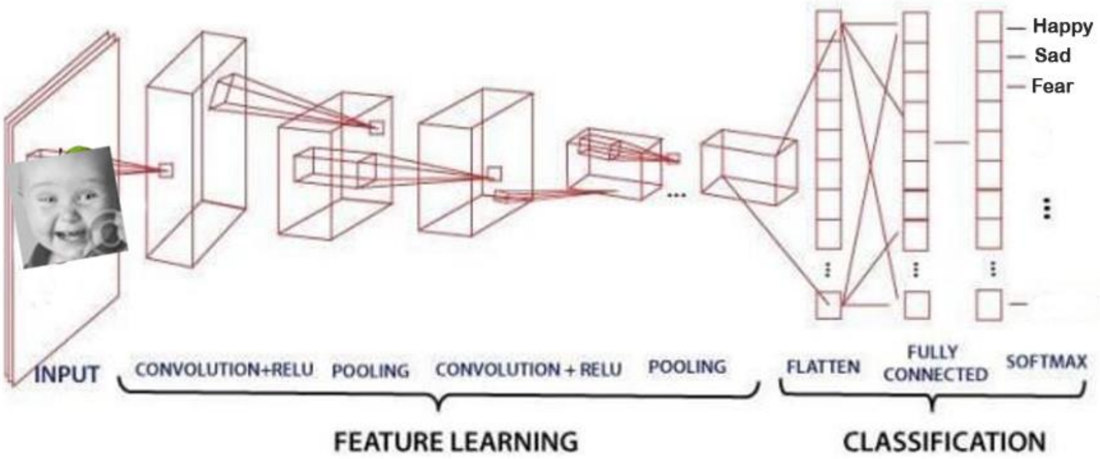


**Figure 4.4: Architecture of convolutional neural layers**

**Convolution layer:**

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

The image matrix represents the input image. It has three dimensions:

Height (h): The number of rows in the image.

Width (w): The number of columns in the image.

Depth (d): The number of channels in the image (our images has only1 channel).

The filter also has three dimensions:

Filter height (fh): The number of rows in the filter.

Filter width (fw): The number of columns in the filter.

Depth (d): The depth of the filter matches the depth of the input image.

After applying the convolution operation, the output feature map is generated. The dimension of the output depends on the size of the input image and the filter used.

The formula to calculate the output dimension is:

Output height = (input height - filter height + 1)

Output width = (input width - filter width + 1)

So, the dimension of the output feature map is (h - fh + 1) x (w - fw + 1) x 1.

Considering a 5*5 image whose pixel values are 0, 1, and filter matrix 3*3 as:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

5 × 5 − Image Matrix     3 × 3 − Filter Matrix

The convolution of 5*5 image matrix multiplies with 3*3 filter matrix is called "Features Map" and show as an output.

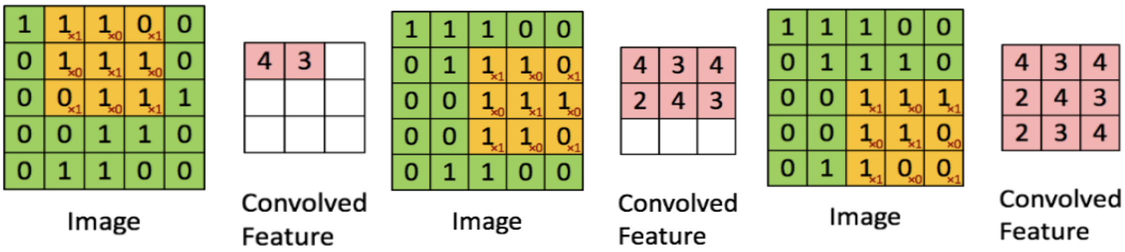


**Figure 4.5: Kernel Traverse**

Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

**Strides**

Stride is the number of pixels which are shift over the input matrix. When the stride is equaled to 1, then  move the filters to 1 pixel at a time and similarly, if the stride is equaled to 2, then  move the filters to 2 pixels at a time. The following figure shows that the convolution would work with a stride of 2.

In below figure the stride length is chosen as 1 so the kernel shifts nine times, each time performing a matrix multiplication of the kernel and the portion of the image under it.

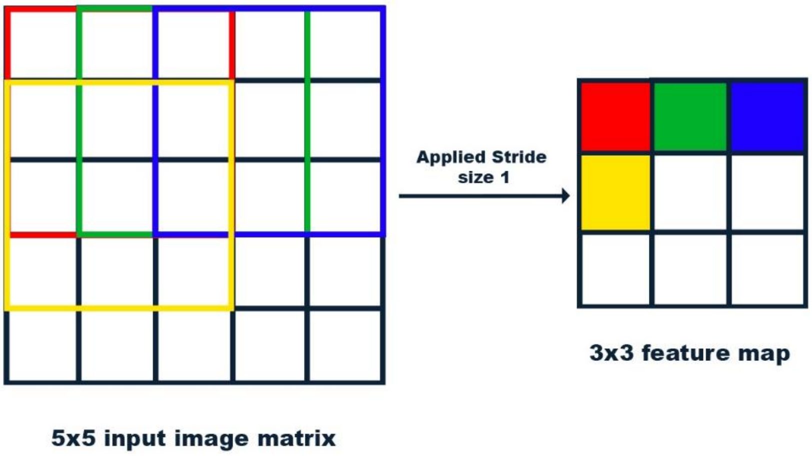


**Figure 4.6: Working of 3*3 size stride**

### Pooling Layer

Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "downscaling" of the image obtained from the previous layers. Spatial pooling is also called down sampling or subsampling, which reduces the dimensionality of each map but retains the important information. There are two types of pooling, namely max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel, while average pooling returns the average of the corresponding values.

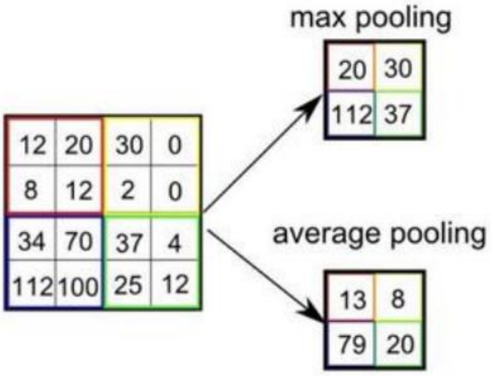


**Figure 4.7: Max pooling and Average pooling**

### Fully Connected Layer

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.

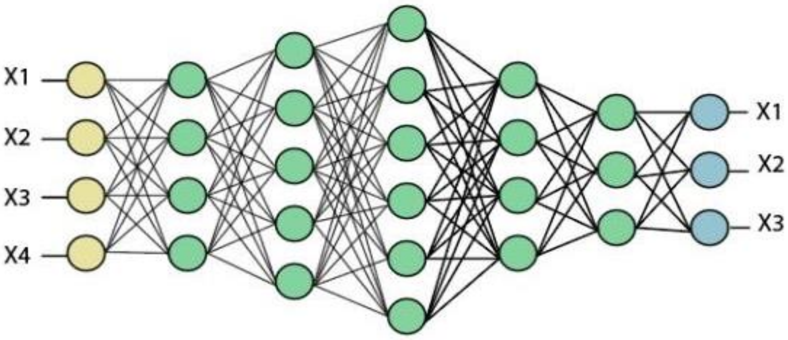


**Figure 4.8: Fully connected layer**

In the above diagram, the feature map matrix will be converted into the vector such as x1, x2, x3... xn which is called flattening. Combined features to create a model and apply the activation function such as soft-max or sigmoid to classify the outputs as a happy, sad, angry, fear etc.

**Dropout**

Dropout is a technique utilized in machine learning to combat overfitting, a phenomenon where a model performs exceptionally well on training data but poorly on unseen test data. It involves temporarily deactivating random neurons during the training process, effectively creating a smaller, less complex network for a particular forward or backward pass. The dropout rate is the probability of training a given node in a layer, where 1.0 means no dropout and 0.0 means all outputs from the layer are ignored.

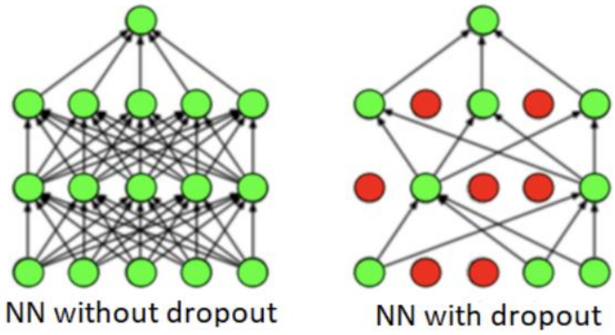


**Figure 4.9: Before and after dropdown in neurons**

## 4.3 Evaluation Metrices

**Accuracy:**

Accuracy measures the proportion of correctly classified instances of all instances. It is calculated as the ratio of the number of correctly predicted instances to the total number of correctly predicted instances to the total number of instances.

Mathematically,

$$\text{Accuracy} = \frac{Number\ of\ Corrected\ Predictions}{Total\ Number\ of\ Predictions}$$

Accuracy provides a general assessment of how well the model performs across all classes.

**Precision:**

Precision measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives + false positives). It focuses on the quality of positive predictions and indicates how many of the predicted positive instances are actually positive.

Mathematically,

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

**Recall:**

Recall measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives + false negatives). It focuses on the ability of the model to capture all positive instances.

Mathematically,

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

**F1 Score:**

F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is often used as a single metric to evaluate model performance. F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Mathematically,

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

# CHAPTER 5 IMPLEMENTATION AND TESTING

## 5.1 Implementation

### 5.1.1 Tools Used

#### Software Development Tools

Python is the primary programming language used in this project, and it makes use of a robust ecosystem of libraries to do a variety of tasks. For jobs involving numbers and data manipulation, NumPy and Pandas are indispensable because they offer effective structures for array operations and data analysis, respectively. OpenCV improves ability to interpret images and videos. Finally, HTML, CSS, Bootstrap and JavaScript are used for this project's frontend.

#### Hardware Tools

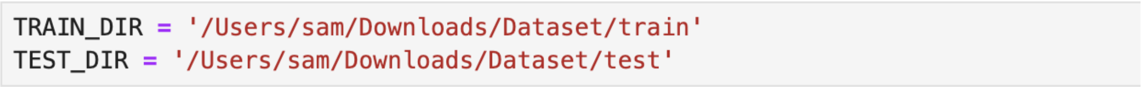
A strong CPU (central processing unit) is used to process input data, control model execution, and speed training and inference operations in order to execute the project efficiently.

Furthermore, a minimum of 8 GB of RAM (Random Access Memory) is required.

#### Diagram Tools

In this project, draw.io, Asana serves as a versatile and user-friendly diagramming tool for creating a variety of visual representations crucial to the software development process. Utilizing draw.io, Use case diagrams are designed to depict the interactions between system components and users, class diagrams to model the structure of software, activity diagrams for visualizing workflow and process dynamics.

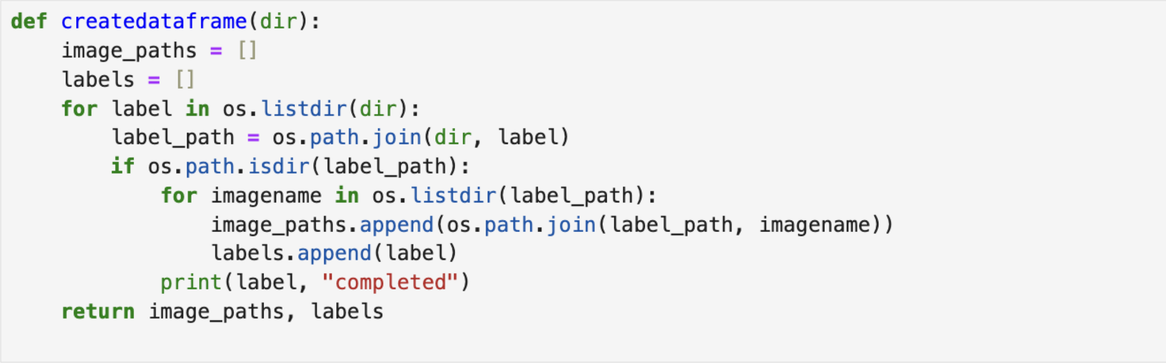
### 5.1.2 Implementation Details of Modules



```
TRAIN_DIR = '/Users/sam/Downloads/Dataset/train'
TEST_DIR = '/Users/sam/Downloads/Dataset/test'
```

**Figure 5.1: Data Training and Testing**

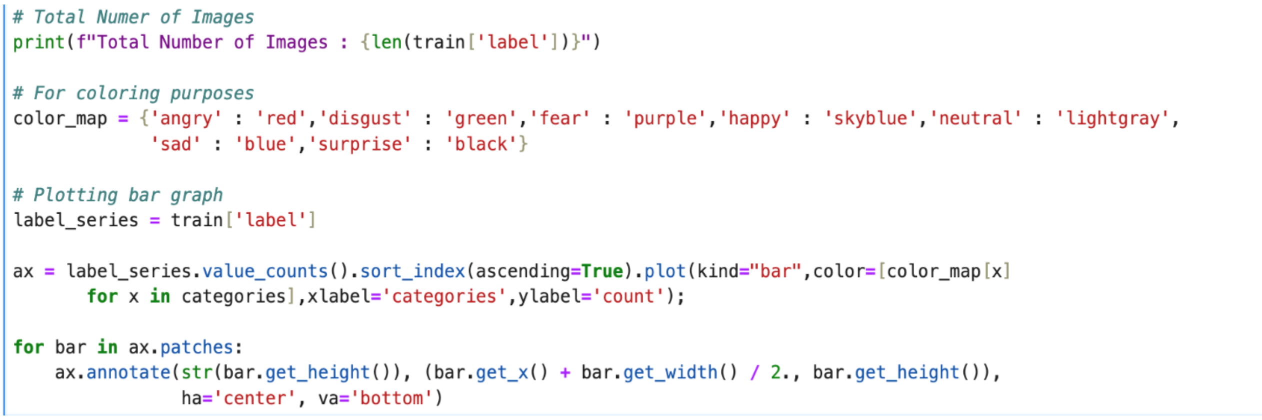
Above figure 5.1 shows the directory for training and testing datasets used in this system.



```
def createdataframe(dir):
    image_paths = []
    labels = []
    for label in os.listdir(dir):
        label_path = os.path.join(dir, label)
        if os.path.isdir(label_path):
            for imagename in os.listdir(label_path):
                image_paths.append(os.path.join(label_path, imagename))
                labels.append(label)
            print(label, "completed")
    return image_paths, labels
```

**Figure 5.2: Collecting images from each dataset and saving it into a numpy array**

Above figure 5.2 shows the python function, createdataframe(dir), is designed to create a dataframe containing image paths and their corresponding labels.



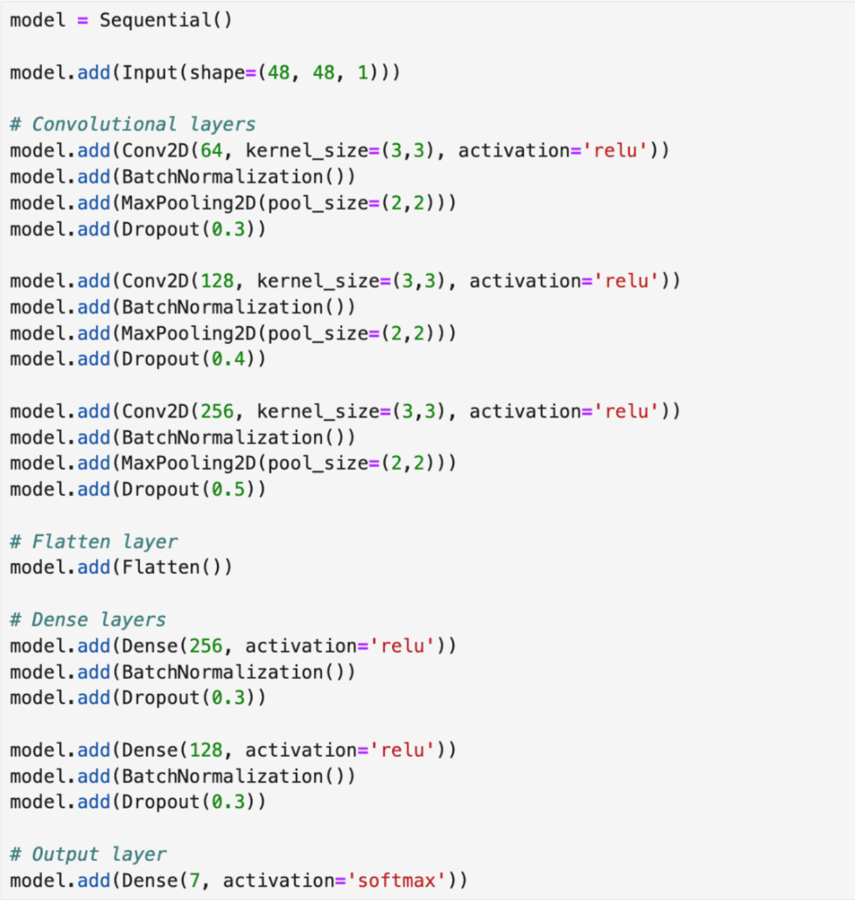
```
# Total Numer of Images
print(f"Total Number of Images : {len(train['label'])}")

# For coloring purposes
color_map = {'angry' : 'red','disgust' : 'green','fear' : 'purple','happy' : 'skyblue','neutral' : 'lightgray',
             'sad' : 'blue','surprise' : 'black'}

# Plotting bar graph
label_series = train['label']

ax = label_series.value_counts().sort_index(ascending=True).plot(kind="bar",color=[color_map[x]
       for x in categories],xlabel='categories',ylabel='count');

for bar in ax.patches:
    ax.annotate(str(bar.get_height()), (bar.get_x() + bar.get_width() / 2., bar.get_height()),
                ha='center', va='bottom')
```

**Figure 5.3: Visualization of image source code**

Above figure 5.3 is the visualization of image source code. It maps color for different type of emotions such as Red for angry, Green for disgust, Purple for fear, Skyblue for happy, Lightgray for neutral, Blue for sad, and Black for surprise.

### Training Module

This convolutional neural network (CNN) is structured to analyze grayscale images of facial expressions, each sized at 48x48 pixels. The model begins with convolutional layers, each identifying different features in the input image like edges or textures, followed by max-pooling layers to condense and preserve important information. Dropout layers are strategically placed after each max-pooling layer to prevent overfitting by randomly deactivating neurons during training, ensuring the network learns robust features. The flattened feature maps are then fed into densely connected layers, refining the extracted features to make accurate predictions. The final dense layer, employing a softmax activation function, outputs probabilities for seven possible facial expressions, including happiness, sadness, and anger. This architecture effectively balances feature extraction and classification, making it suitable for facial expression recognition tasks.



```python
model = Sequential()

model.add(Input(shape=(48, 48, 1)))

# Convolutional layers
model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.3))

model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

# Flatten layer
model.add(Flatten())

# Dense layers
model.add(Dense(256, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.3))

# Output layer
model.add(Dense(7, activation='softmax'))
```

**Figure 5.4: Training Module**

## 5.2 Testing

Testing is the process of reviewing and validating whether or not the generated software or application works properly working or not, that is, whether the actual results match the predicted ones. Testing occurs during the software development process.

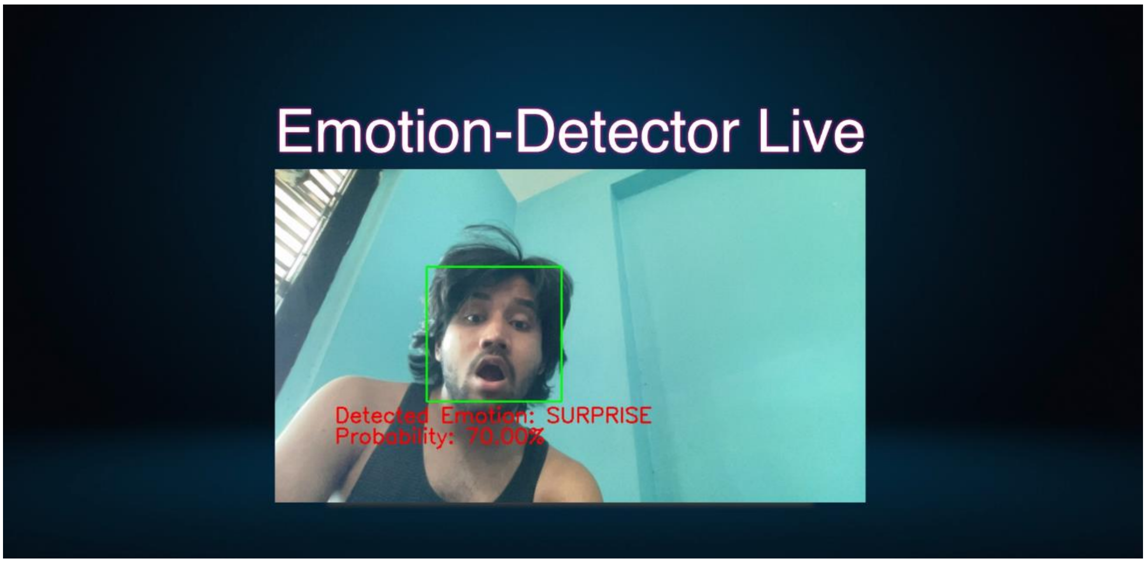
### 5.2.1 Unit Testing

Unit testing is a component of the testing process that covers testing both individual software modules and the components that comprise the complete software. The goal is to validate each unit of software code so that it works as intended.

| Test ID | Test Case | Test Steps | Expected Outcome | Remarks |
|---|---|---|---|---|
| 1 | Valid Facial Expression | User operating system under a good lightening. | The system accurately recognizes and interprets facial expressions. | Pass |
| 2 | Multiple Faces | Multiple user in a webcam. | The system handles multiple faces and accurately detects expressions. | Pass |
| 3 | No Face Detected | No clear facial features | The system recognizes the absence of an observable face. | Fail |
| 4 | Different Lighting Conditions | Use images with varying lighting conditions. | System performs consistently across different lighting scenarios | Fail |

**Table 5.1: Test Cases for Face Emotion detection**

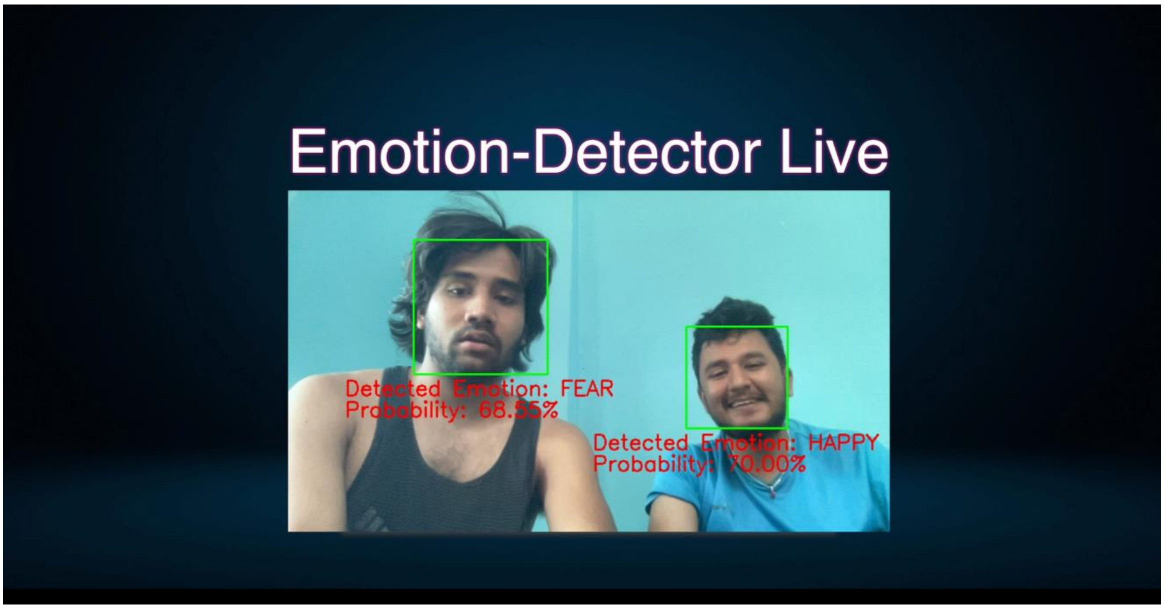
Above table 5.1 outlines test cases for evaluating the face detection component of the system. Tests include scenarios like detecting valid expressions, handling multiple faces, recognizing the absence of a face, and performing consistently under different lighting conditions. Each test assesses the system's ability to accurately detect and interpret facial expressions in various situations.



Emotion-Detector Live

Detected Emotion: SURPRISE
Probability: 70.00%

**Figure 5.5: Test Case for Valid Facial Expression**



Emotion-Detector Live

Detected Emotion: FEAR
Probability: 68.55%

Detected Emotion: HAPPY
Probability: 70.00%

**Figure 5.6: Test Case for Multiple Facial Expression**

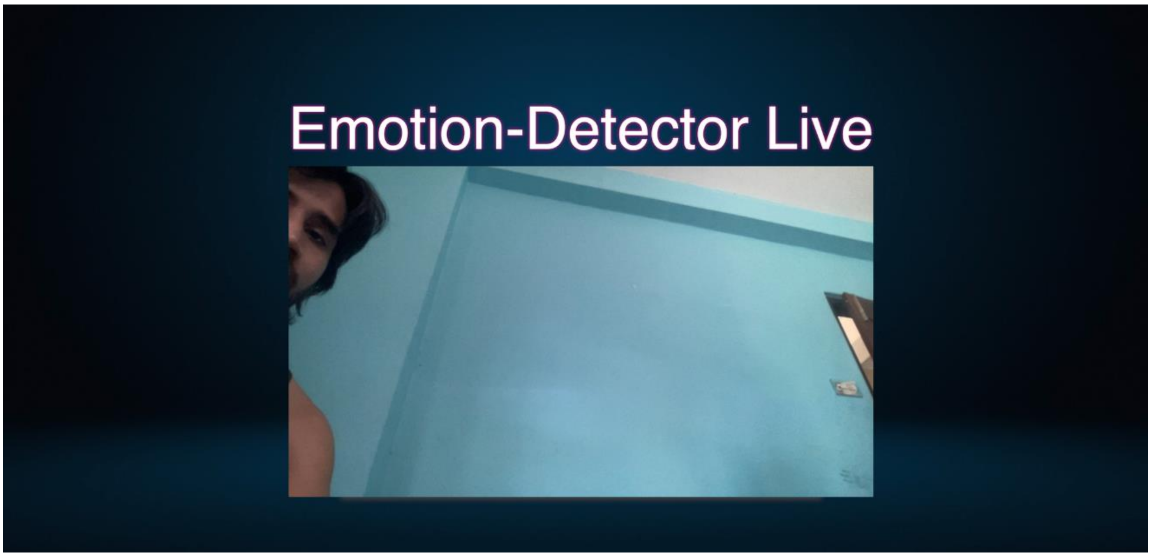
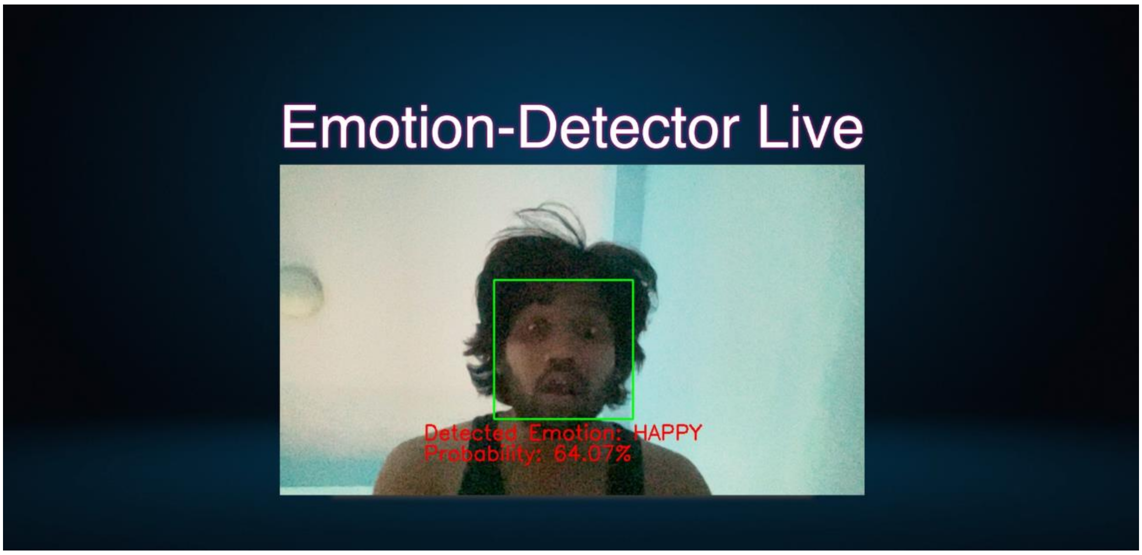


**Figure 5.7: Test Case for No Face Detected Expression**



Detected Emotion: HAPPY
Probability: 64.07%

**Figure 5.8: Test Case for Low Light**

### 5.2.2 Test cases for system Testing

System testing is a thorough assessment of an entire software system to ensure it functions as intended. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users.

| Test ID | Test Case | Test Steps | Expected Outcome | Remarks |
|---|---|---|---|---|
| 1 | End to end functionality | User operating system under a good lightening. | System accurately detects and interprets the emotion | Pass |
| 2 | Handling of multiple faces | Multiple user in a webcam. | System correctly identifies expressions for each face | Pass |
| 3 | Handling of poor lighting conditions | Use image with varying lighting conditions | System maintains consistent accuracy across different lighting scenarios | Fail |

**Table 5.2: Test case for Face Emotion Detection -2**

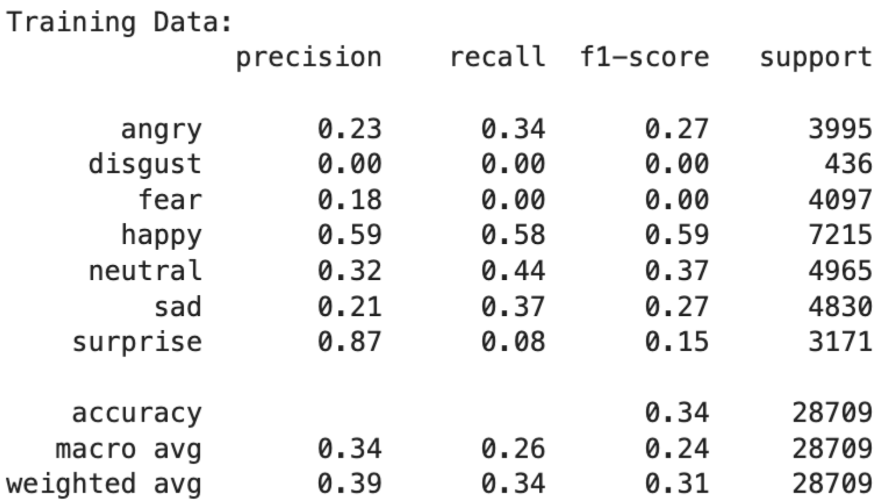
Above table 5.2 focuses on testing the face emotion detection system, covering aspects such as accuracy, handling multiple faces, performance under varying lighting conditions and robustness against noise or distortions.

## 5.3 Result Analysis

This project went through unit testing as well as system testing, demonstrating its effectiveness in performing designated tasks. To optimize the project's impact and enhance inclusivity within the community, a comprehensive report is generated, summarizing the performance metrics of various models used in this project. This analysis forms the basis for continuous development, ensuring ongoing enhancements to better serve the community. The goal is to continually improve the project, making it more beneficial and responsive to face detection.

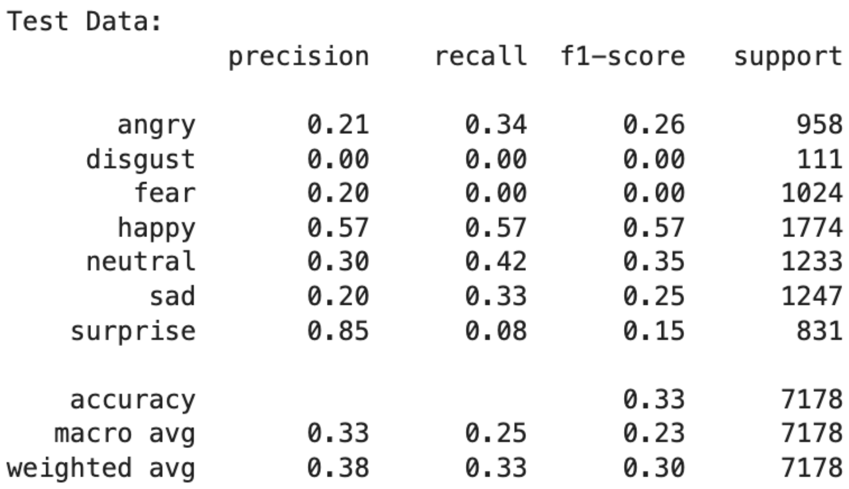
### 5.3.1 Performance metrics

Following classification report is a summary of the performance metrics of model.



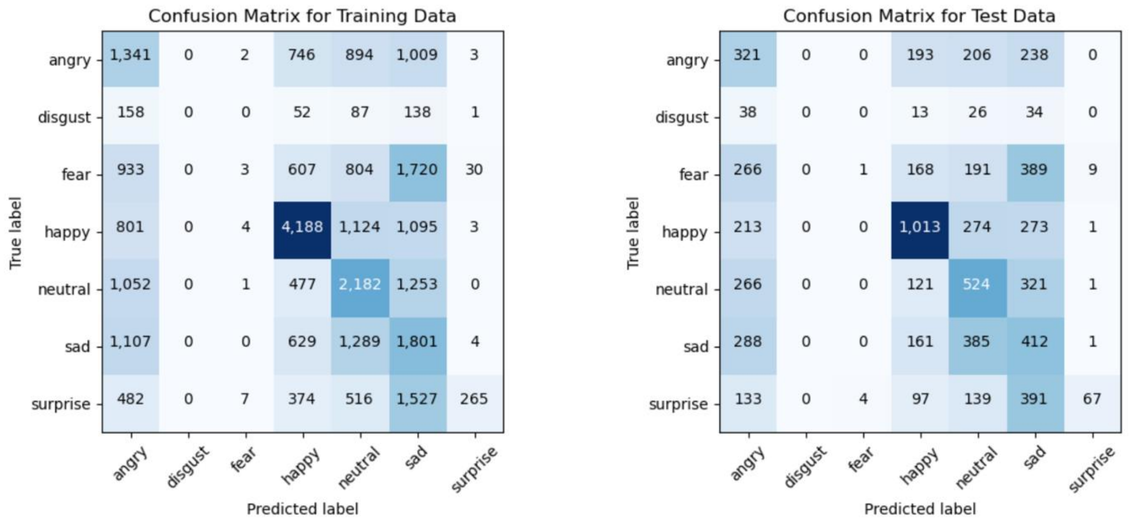
```
Training Data:
              precision    recall  f1-score   support

       angry       0.23      0.34      0.27      3995
     disgust       0.00      0.00      0.00       436
        fear       0.18      0.00      0.00      4097
       happy       0.59      0.58      0.59      7215
     neutral       0.32      0.44      0.37      4965
         sad       0.21      0.37      0.27      4830
    surprise       0.87      0.08      0.15      3171

    accuracy                           0.34     28709
   macro avg       0.34      0.26      0.24     28709
weighted avg       0.39      0.34      0.31     28709
```

**Figure 5.9: Classification report of train data**



```
Test Data:
              precision    recall  f1-score   support

       angry       0.21      0.34      0.26       958
     disgust       0.00      0.00      0.00       111
        fear       0.20      0.00      0.00      1024
       happy       0.57      0.57      0.57      1774
     neutral       0.30      0.42      0.35      1233
         sad       0.20      0.33      0.25      1247
    surprise       0.85      0.08      0.15       831

    accuracy                           0.33      7178
   macro avg       0.33      0.25      0.23      7178
weighted avg       0.38      0.33      0.30      7178
```

**Figure 5.10: Classification report of test data**



Confusion Matrix for Training Data

| | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 1,341 | 0 | 2 | 746 | 894 | 1,009 | 3 |
| disgust | 158 | 0 | 0 | 52 | 87 | 138 | 1 |
| fear | 933 | 0 | 3 | 607 | 804 | 1,720 | 30 |
| happy | 801 | 0 | 4 | 4,188 | 1,124 | 1,095 | 3 |
| neutral | 1,052 | 0 | 1 | 477 | 2,182 | 1,253 | 0 |
| sad | 1,107 | 0 | 0 | 629 | 1,289 | 1,801 | 4 |
| surprise | 482 | 0 | 7 | 374 | 516 | 1,527 | 265 |

Confusion Matrix for Test Data

| | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 321 | 0 | 0 | 193 | 206 | 238 | 0 |
| disgust | 38 | 0 | 0 | 13 | 26 | 34 | 0 |
| fear | 266 | 0 | 1 | 168 | 191 | 389 | 9 |
| happy | 213 | 0 | 0 | 1,013 | 274 | 273 | 1 |
| neutral | 266 | 0 | 0 | 121 | 524 | 321 | 1 |
| sad | 288 | 0 | 0 | 161 | 385 | 412 | 1 |
| surprise | 133 | 0 | 4 | 97 | 139 | 391 | 67 |

**Figure 5.11: Confusion matrix of train and test data**

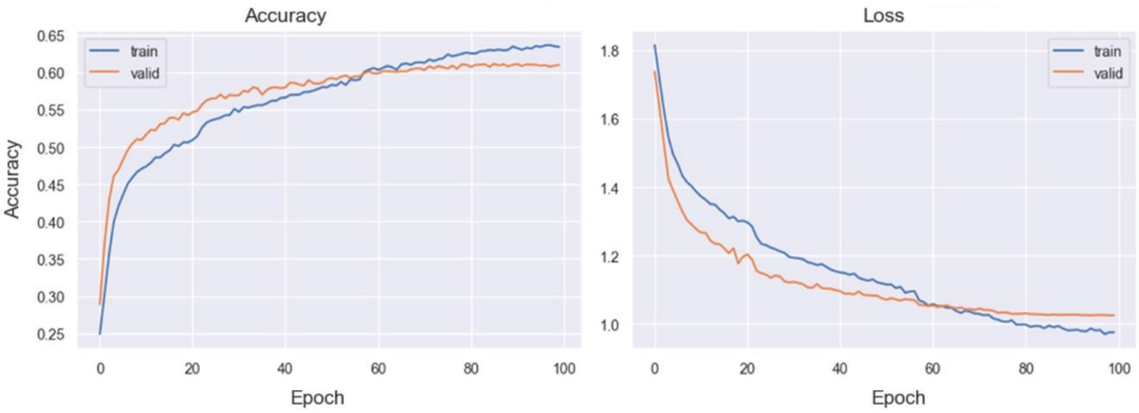
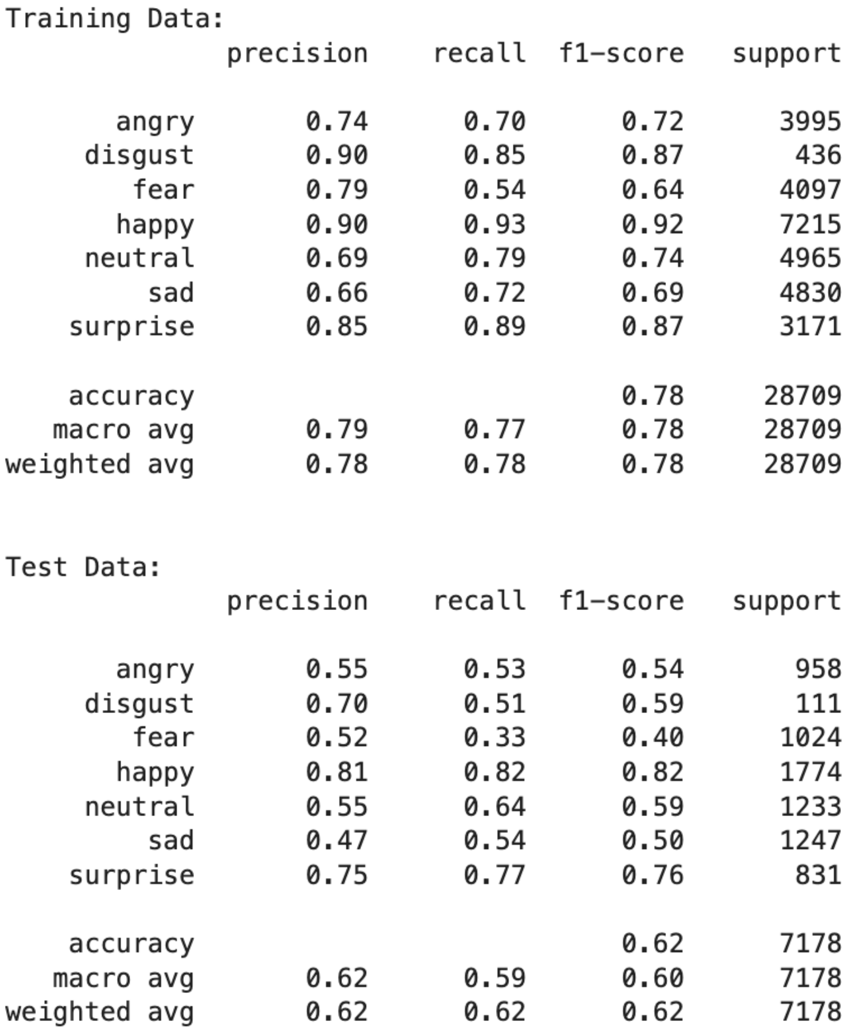


**Figure 5.12: Model accuracy graph**

### 5.3.2 Performance metrics of other model

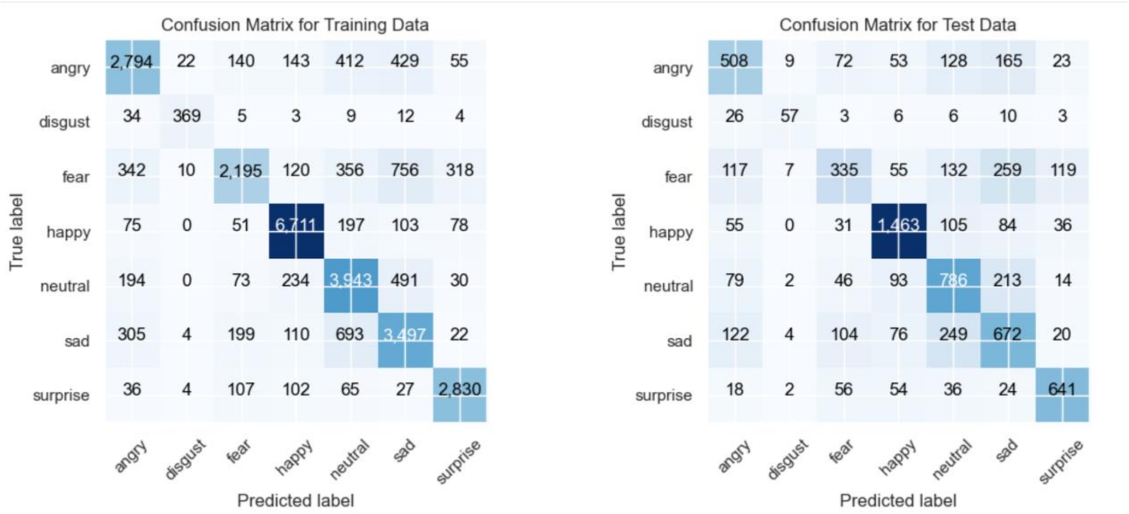
Following classification report is a summary of the performance metrics of other model we trained under different epoch and batch size.

**Epoch 50 Batch Size 16**



```
Training Data:
              precision    recall  f1-score   support

       angry       0.74      0.70      0.72      3995
     disgust       0.90      0.85      0.87       436
        fear       0.79      0.54      0.64      4097
       happy       0.90      0.93      0.92      7215
     neutral       0.69      0.79      0.74      4965
         sad       0.66      0.72      0.69      4830
    surprise       0.85      0.89      0.87      3171

    accuracy                           0.78     28709
   macro avg       0.79      0.77      0.78     28709
weighted avg       0.78      0.78      0.78     28709


Test Data:
              precision    recall  f1-score   support

       angry       0.55      0.53      0.54       958
     disgust       0.70      0.51      0.59       111
        fear       0.52      0.33      0.40      1024
       happy       0.81      0.82      0.82      1774
     neutral       0.55      0.64      0.59      1233
         sad       0.47      0.54      0.50      1247
    surprise       0.75      0.77      0.76       831

    accuracy                           0.62      7178
   macro avg       0.62      0.59      0.60      7178
weighted avg       0.62      0.62      0.62      7178
```

**Figure 5.13: Classification report of train data and Test data of Epoch 50 Batch Size 16**



Confusion Matrix for Training Data

| True label \ Predicted label | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 2,794 | 22 | 140 | 143 | 412 | 429 | 55 |
| disgust | 34 | 369 | 5 | 3 | 9 | 12 | 4 |
| fear | 342 | 10 | 2,195 | 120 | 356 | 756 | 318 |
| happy | 75 | 0 | 51 | 6,711 | 197 | 103 | 78 |
| neutral | 194 | 0 | 73 | 234 | 3,943 | 491 | 30 |
| sad | 305 | 4 | 199 | 110 | 693 | 3,497 | 22 |
| surprise | 36 | 4 | 107 | 102 | 65 | 27 | 2,830 |

Confusion Matrix for Test Data

| True label \ Predicted label | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 508 | 9 | 72 | 53 | 128 | 165 | 23 |
| disgust | 26 | 57 | 3 | 6 | 6 | 10 | 3 |
| fear | 117 | 7 | 335 | 55 | 132 | 259 | 119 |
| happy | 55 | 0 | 31 | 1,463 | 105 | 84 | 36 |
| neutral | 79 | 2 | 46 | 93 | 786 | 213 | 14 |
| sad | 122 | 4 | 104 | 76 | 249 | 672 | 20 |
| surprise | 18 | 2 | 56 | 54 | 36 | 24 | 641 |

**Figure 5.14: Confusion matrix of train and test data of Epoch 50 Batch Size 16**

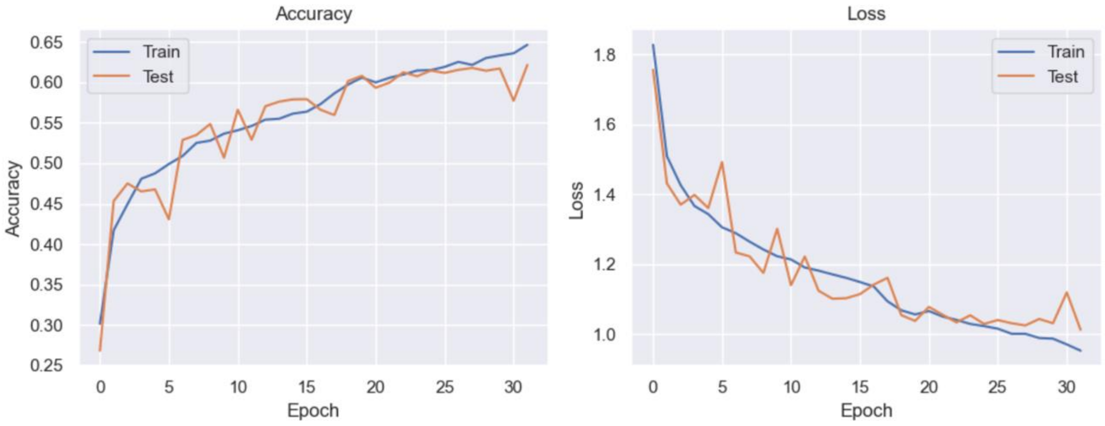
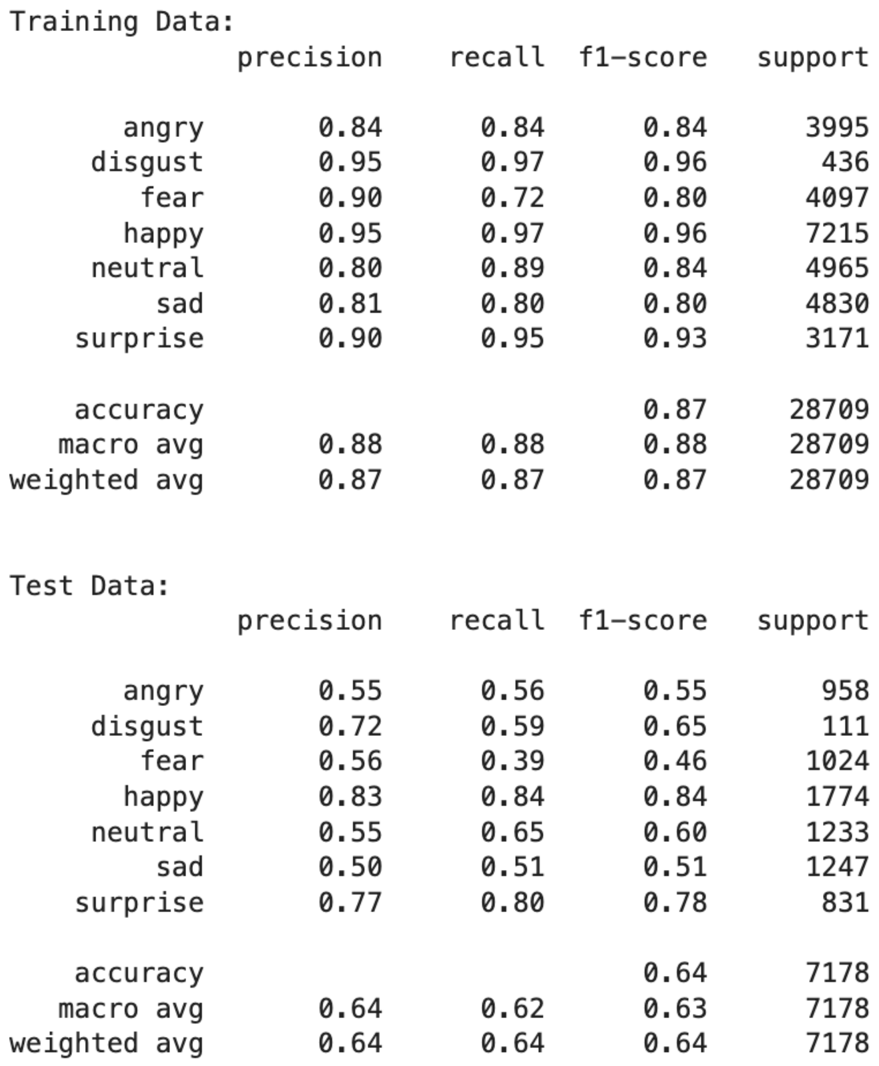


**Figure 5.15: Model accuracy graph of Epoch 50 Batch Size 16**

**Epoch 32 Batch Size 50**



```
Training Data:
              precision    recall  f1-score   support

       angry       0.84      0.84      0.84      3995
     disgust       0.95      0.97      0.96       436
        fear       0.90      0.72      0.80      4097
       happy       0.95      0.97      0.96      7215
     neutral       0.80      0.89      0.84      4965
         sad       0.81      0.80      0.80      4830
    surprise       0.90      0.95      0.93      3171

    accuracy                           0.87     28709
   macro avg       0.88      0.88      0.88     28709
weighted avg       0.87      0.87      0.87     28709


Test Data:
              precision    recall  f1-score   support

       angry       0.55      0.56      0.55       958
     disgust       0.72      0.59      0.65       111
        fear       0.56      0.39      0.46      1024
       happy       0.83      0.84      0.84      1774
     neutral       0.55      0.65      0.60      1233
         sad       0.50      0.51      0.51      1247
    surprise       0.77      0.80      0.78       831

    accuracy                           0.64      7178
   macro avg       0.64      0.62      0.63      7178
weighted avg       0.64      0.64      0.64      7178
```

**Figure 5.16: Classification report of train data and Test data of Epoch 32 Batch Size 50**

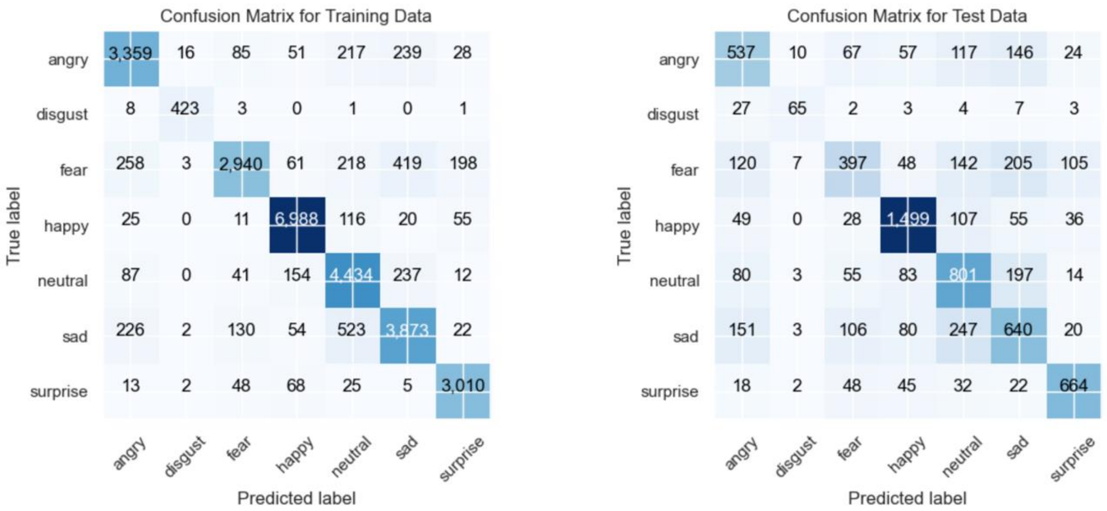


**Figure 5.17: Confusion matrix of train and test data of Epoch 32 Batch Size 50**

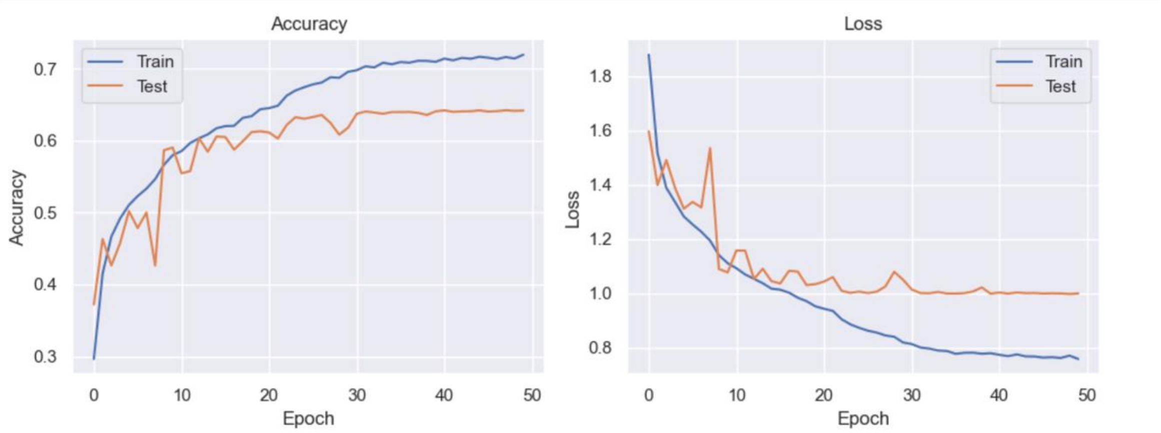
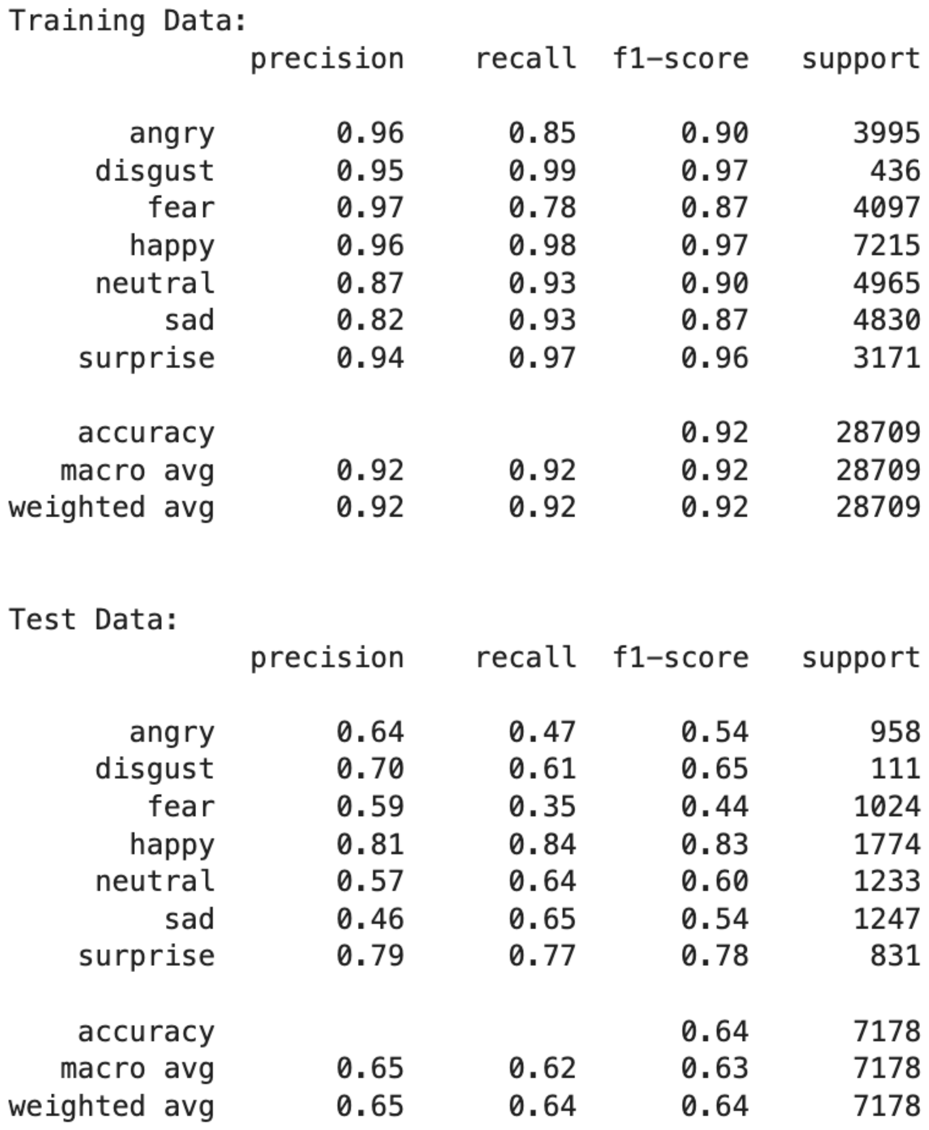


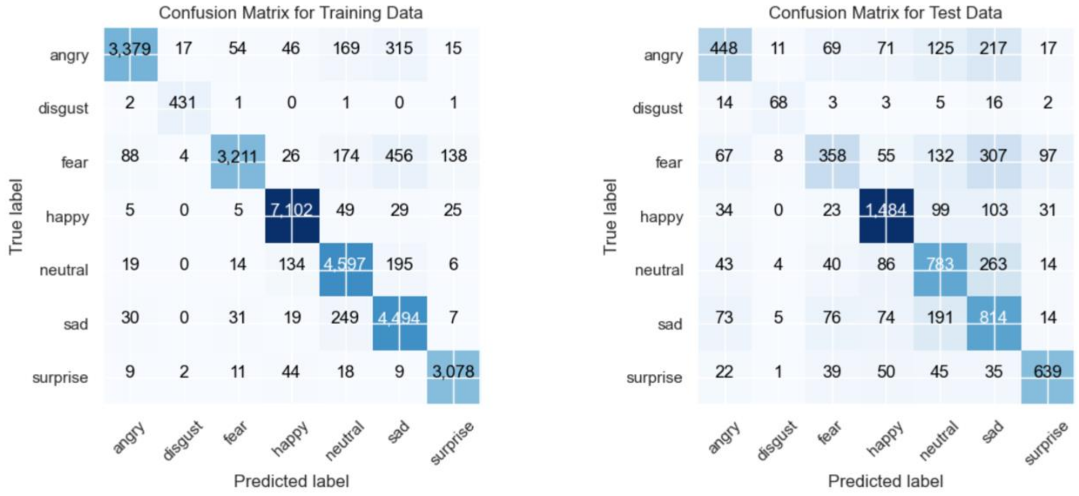
**Figure 5.18: Model accuracy graph of Epoch 32 Batch Size 50**

**Epoch 64 Batch Size 35**



```
Training Data:
              precision    recall  f1-score   support

       angry       0.96      0.85      0.90      3995
     disgust       0.95      0.99      0.97       436
        fear       0.97      0.78      0.87      4097
       happy       0.96      0.98      0.97      7215
     neutral       0.87      0.93      0.90      4965
         sad       0.82      0.93      0.87      4830
    surprise       0.94      0.97      0.96      3171

    accuracy                           0.92     28709
   macro avg       0.92      0.92      0.92     28709
weighted avg       0.92      0.92      0.92     28709


Test Data:
              precision    recall  f1-score   support

       angry       0.64      0.47      0.54       958
     disgust       0.70      0.61      0.65       111
        fear       0.59      0.35      0.44      1024
       happy       0.81      0.84      0.83      1774
     neutral       0.57      0.64      0.60      1233
         sad       0.46      0.65      0.54      1247
    surprise       0.79      0.77      0.78       831

    accuracy                           0.64      7178
   macro avg       0.65      0.62      0.63      7178
weighted avg       0.65      0.64      0.64      7178
```

**Figure 5.19: Classification report of train data and Test data of Epoch 42 Batch Size 55**



Confusion Matrix for Training Data

| True label \ Predicted label | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 3,379 | 17 | 54 | 46 | 169 | 315 | 15 |
| disgust | 2 | 431 | 1 | 0 | 1 | 0 | 1 |
| fear | 88 | 4 | 3,211 | 26 | 174 | 456 | 138 |
| happy | 5 | 0 | 5 | 7,102 | 49 | 29 | 25 |
| neutral | 19 | 0 | 14 | 134 | 4,597 | 195 | 6 |
| sad | 30 | 0 | 31 | 19 | 249 | 4,494 | 7 |
| surprise | 9 | 2 | 11 | 44 | 18 | 9 | 3,078 |

Confusion Matrix for Test Data

| True label \ Predicted label | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 448 | 11 | 69 | 71 | 125 | 217 | 17 |
| disgust | 14 | 68 | 3 | 3 | 5 | 16 | 2 |
| fear | 67 | 8 | 358 | 55 | 132 | 307 | 97 |
| happy | 34 | 0 | 23 | 1,484 | 99 | 103 | 31 |
| neutral | 43 | 4 | 40 | 86 | 783 | 263 | 14 |
| sad | 73 | 5 | 76 | 74 | 191 | 814 | 14 |
| surprise | 22 | 1 | 39 | 50 | 45 | 35 | 639 |

**Figure 5.20: Confusion matrix of train and test data of Epoch 42 Batch Size 55**

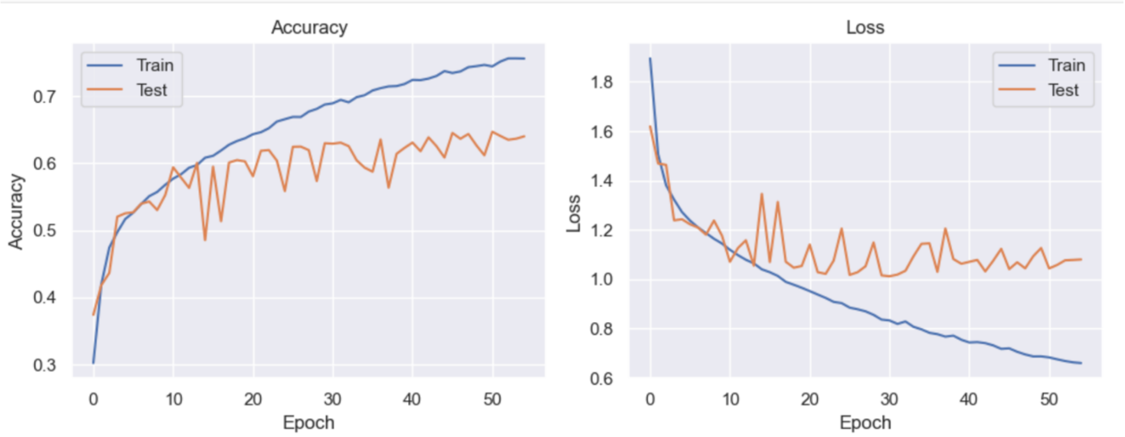


**Figure 5.21: Model accuracy graph of Epoch 42 Batch Size 55**

# CHAPTER 6 CONCLUSION AND FUTURE RECOMMENDATION

## 6.1 Conclusion

In conclusion, the Emotion Detection System effectively recognizes facial expressions and infers corresponding emotions, achieving an accuracy of 65%. Among the various trained models, CNN stands out for its superior performance. Leveraging the CNN paradigm, the system accurately classifies facial expressions, including anger, sad, happiness, surprise, and fear. Moving forward, the Emotion Detection System with Live Face Detection Using CNN aims to provide users with enhanced emotion detection capabilities, facilitating various applications such as emotion-aware user interfaces, personalized content recommendations, and interactive gaming experiences. By continuously refining the model and incorporating user feedback, the system strives to improve its accuracy and broaden its utility across diverse domains, ultimately enhancing user experiences and engagement.

## 6.2 Future Recommendations:

Consider developing a mobile application to extend the accessibility of the system. A mobile app would allow users to experience the better emotion detection system on their smartphones, providing flexibility and convenience. Deep learning based techniques require a large labeled dataset, significant memory, and lengthy training and testing cycles, making them challenging to execute on mobile and other devices with low resources. Simple solutions with minimal data and memory requirements are recommended. Furthermore, integrating user feedback methods within the app is critical for increasing user engagement and improving recommendation algorithms. By gathering and evaluating user feedback, the system can continuously improve the system and its functionalities.

# REFERENCES

[1] H. C. Chang and E. Dupont, CS231N project: Facial Expression Recognition - Stanford University, https://cs231n.stanford.edu/reports/2016/pdfs/009_Report.pdf (accessed Apr. 25, 2024).

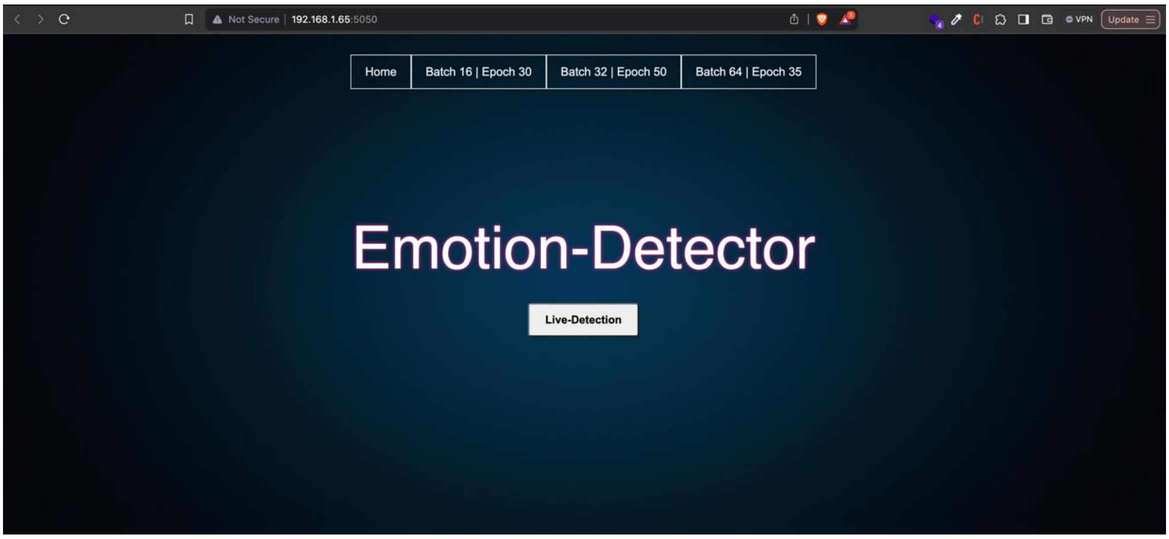
[2] J. A. Ballesteros, G. M. Ramírez V., F. Moreira, A. Solano, and C. A. Pelaez, “Facial emotion recognition through artificial intelligence,” *Frontiers in Computer Science*, vol. 6, Jan. 2024. doi:10.3389/fcomp.2024.1359471

[3] F. Ghaffar and K. Ali, A Robust System for Facial Emotions Recognition Using Convolutional Neural Network, https://arxiv.org/ftp/arxiv/papers/2001/2001.01456.pdf (accessed Apr. 25, 2024).

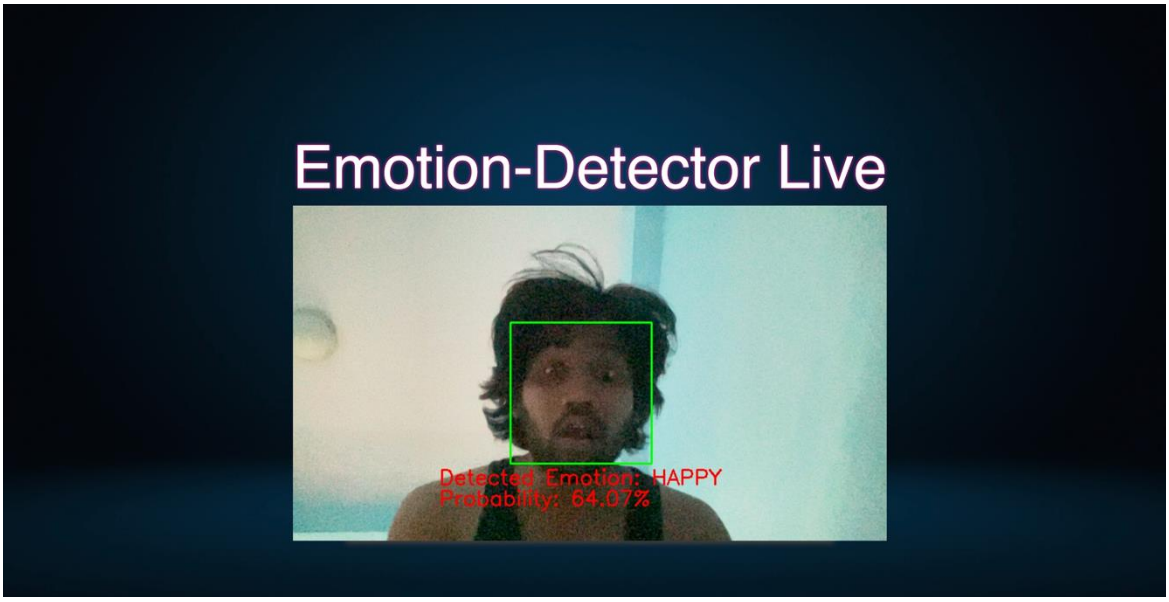
[4] N. Mehendale, “Facial emotion recognition using convolutional neural networks (FERC),” *SN Applied Sciences*, vol. 2, no. 3, Feb. 2020. doi:10.1007/s42452-020-2234-1

[5] S. A. Inigo and V. R. Kumar, Real-Time Emotion Recognition System using Facial Expressions and Soft Computing methodologies, https://philarchive.org/archive/ARURER (accessed Apr. 25, 2024).

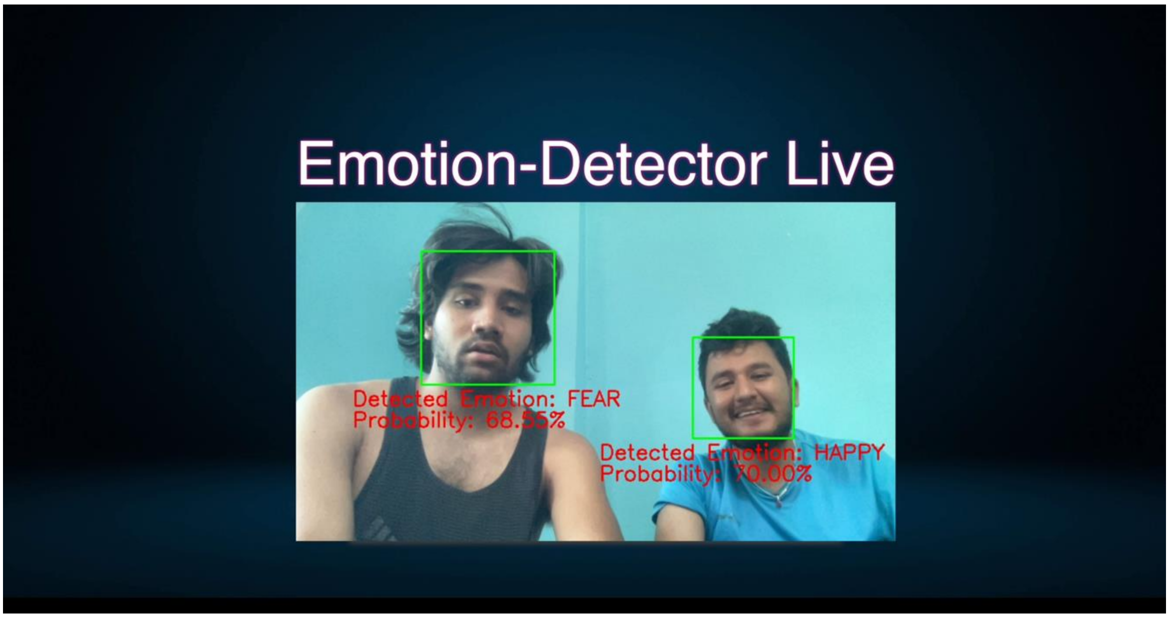
# APPENDICES



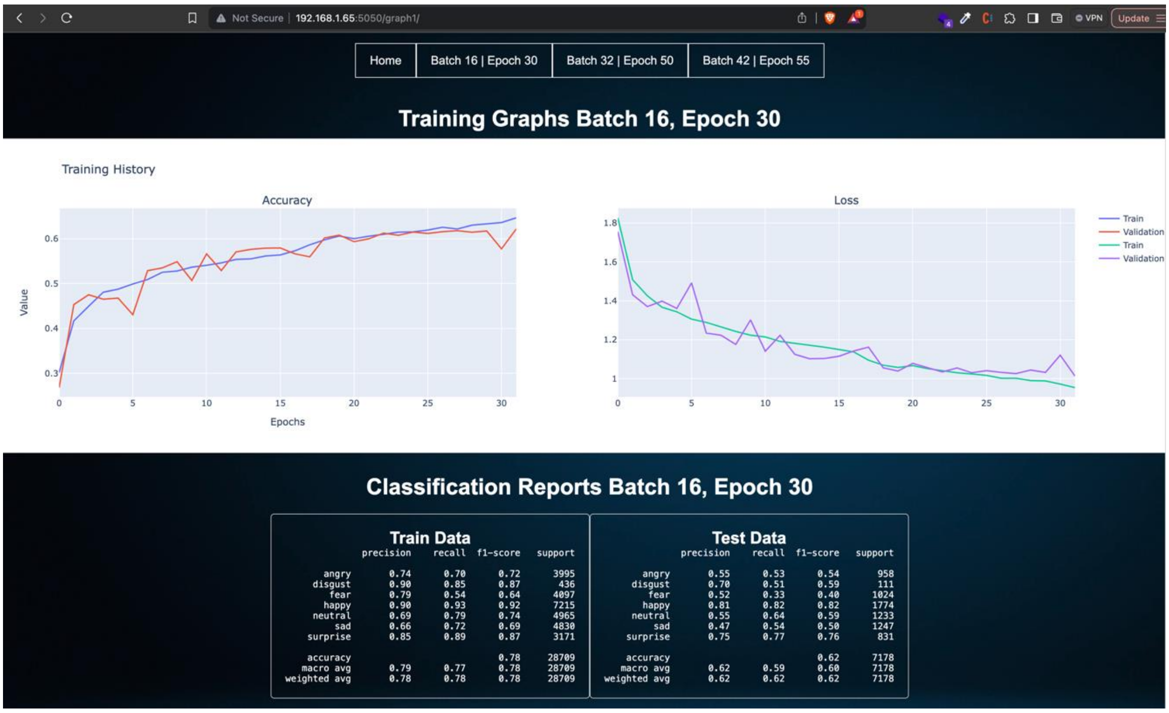
**User Interface**



Detected Emotion: HAPPY
Probability: 64.07%



**Live Emotion Detection**



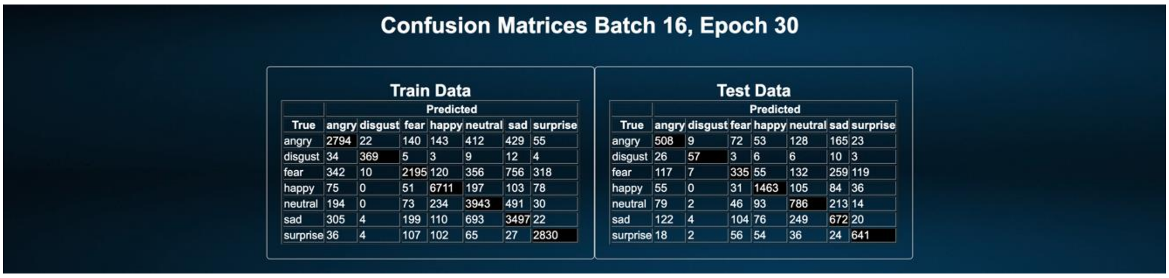
Classification Reports Batch 16, Epoch 30

Train Data

```
              precision    recall  f1-score   support

       angry       0.74      0.70      0.72      3995
     disgust       0.90      0.85      0.87       436
        fear       0.79      0.54      0.64      4097
       happy       0.90      0.93      0.92      7215
     neutral       0.69      0.79      0.74      4965
         sad       0.66      0.72      0.69      4830
    surprise       0.85      0.89      0.87      3171

    accuracy                           0.78     28709
   macro avg       0.79      0.77      0.78     28709
weighted avg       0.78      0.78      0.78     28709
```

Test Data

```
              precision    recall  f1-score   support

       angry       0.55      0.53      0.54       958
     disgust       0.70      0.51      0.59       111
        fear       0.52      0.33      0.40      1024
       happy       0.81      0.82      0.82      1774
     neutral       0.55      0.64      0.59      1233
         sad       0.47      0.54      0.50      1247
    surprise       0.75      0.77      0.76       831

    accuracy                           0.62      7178
   macro avg       0.62      0.59      0.60      7178
weighted avg       0.62      0.62      0.62      7178
```



Confusion Matrices Batch 16, Epoch 30

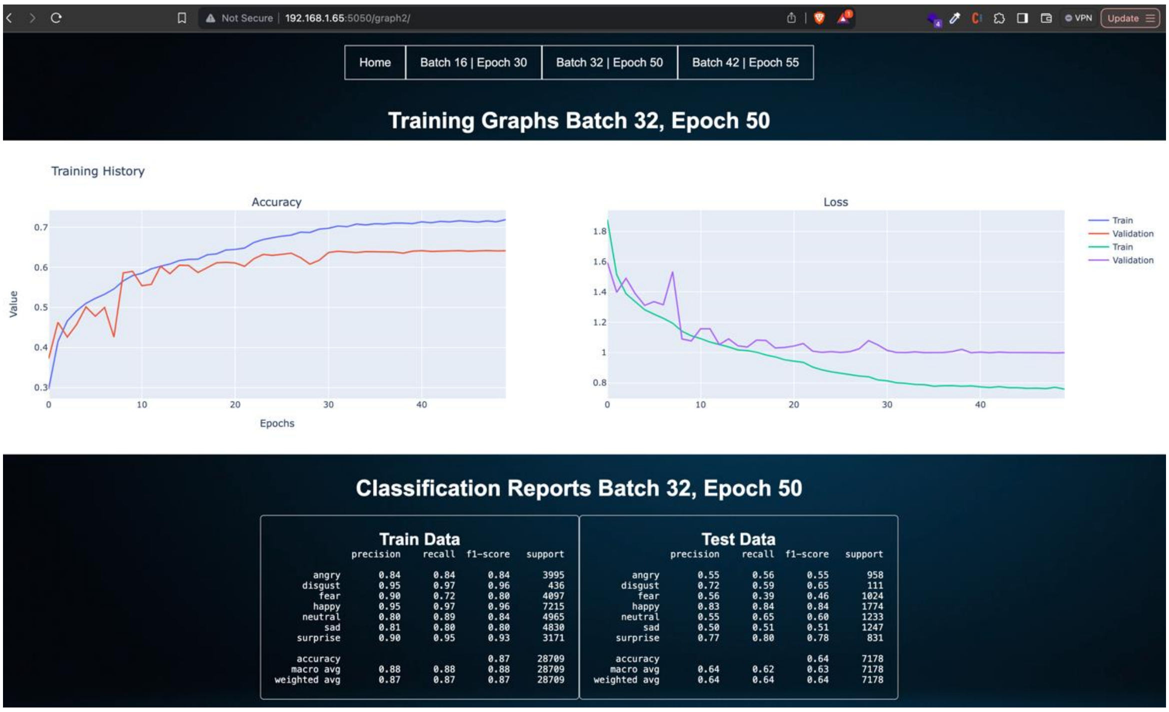
Train Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 2794 | 22 | 140 | 143 | 412 | 429 | 55 |
| disgust | 34 | 369 | 5 | 3 | 9 | 12 | 4 |
| fear | 342 | 10 | 2195 | 120 | 356 | 756 | 318 |
| happy | 75 | 0 | 51 | 6711 | 197 | 103 | 78 |
| neutral | 194 | 0 | 73 | 234 | 3943 | 491 | 30 |
| sad | 305 | 4 | 199 | 110 | 693 | 3497 | 22 |
| surprise | 36 | 4 | 107 | 102 | 65 | 27 | 2830 |

Test Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 508 | 9 | 72 | 53 | 128 | 165 | 23 |
| disgust | 26 | 57 | 3 | 6 | 6 | 10 | 3 |
| fear | 117 | 7 | 335 | 55 | 132 | 259 | 119 |
| happy | 55 | 0 | 31 | 1463 | 105 | 84 | 36 |
| neutral | 79 | 2 | 46 | 93 | 786 | 213 | 14 |
| sad | 122 | 4 | 104 | 76 | 249 | 672 | 20 |
| surprise | 18 | 2 | 56 | 54 | 36 | 24 | 641 |

**Interface showing performance metrics of Epoch 30, Batch size 16**



Training Graphs Batch 32, Epoch 50

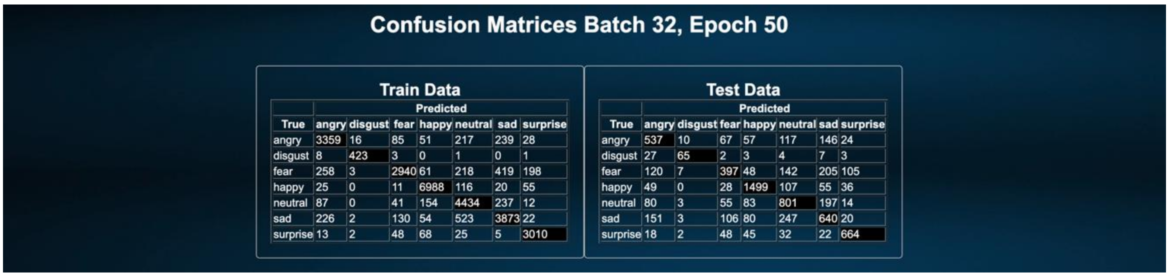
Classification Reports Batch 32, Epoch 50

Train Data

```
              precision    recall  f1-score   support

       angry       0.84      0.84      0.84      3995
     disgust       0.95      0.97      0.96       436
        fear       0.90      0.72      0.80      4097
       happy       0.95      0.97      0.96      7215
     neutral       0.80      0.89      0.84      4965
         sad       0.81      0.80      0.80      4830
    surprise       0.90      0.95      0.93      3171

    accuracy                           0.87     28709
   macro avg       0.88      0.88      0.88     28709
weighted avg       0.87      0.87      0.87     28709
```

Test Data

```
              precision    recall  f1-score   support

       angry       0.55      0.56      0.55       958
     disgust       0.72      0.59      0.65       111
        fear       0.56      0.39      0.46      1024
       happy       0.83      0.84      0.84      1774
     neutral       0.55      0.65      0.60      1233
         sad       0.50      0.51      0.51      1247
    surprise       0.77      0.80      0.78       831

    accuracy                           0.64      7178
   macro avg       0.64      0.62      0.63      7178
weighted avg       0.64      0.64      0.64      7178
```



Confusion Matrices Batch 32, Epoch 50

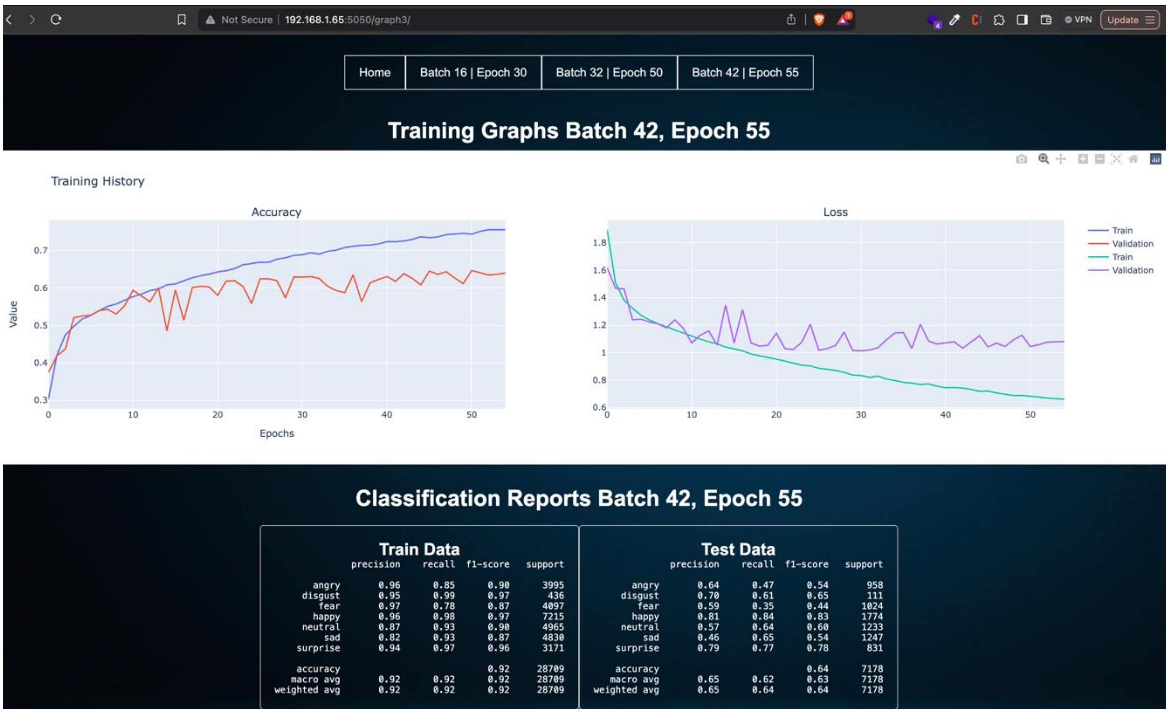
Train Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 3359 | 16 | 85 | 51 | 217 | 239 | 28 |
| disgust | 8 | 423 | 3 | 0 | 1 | 0 | 1 |
| fear | 258 | 3 | 2940 | 61 | 218 | 419 | 198 |
| happy | 25 | 0 | 11 | 6988 | 116 | 20 | 55 |
| neutral | 87 | 0 | 41 | 154 | 4434 | 237 | 12 |
| sad | 226 | 2 | 130 | 54 | 523 | 3873 | 22 |
| surprise | 13 | 2 | 48 | 68 | 25 | 5 | 3010 |

Test Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 537 | 10 | 67 | 57 | 117 | 146 | 24 |
| disgust | 27 | 65 | 2 | 3 | 4 | 7 | 3 |
| fear | 120 | 7 | 397 | 48 | 142 | 205 | 105 |
| happy | 49 | 0 | 28 | 1499 | 107 | 55 | 36 |
| neutral | 80 | 3 | 55 | 83 | 801 | 197 | 14 |
| sad | 151 | 3 | 106 | 80 | 247 | 640 | 20 |
| surprise | 18 | 2 | 48 | 45 | 32 | 22 | 664 |

**Interface showing performance metrics of Epoch 32, Batch size 50**



Training Graphs Batch 42, Epoch 55

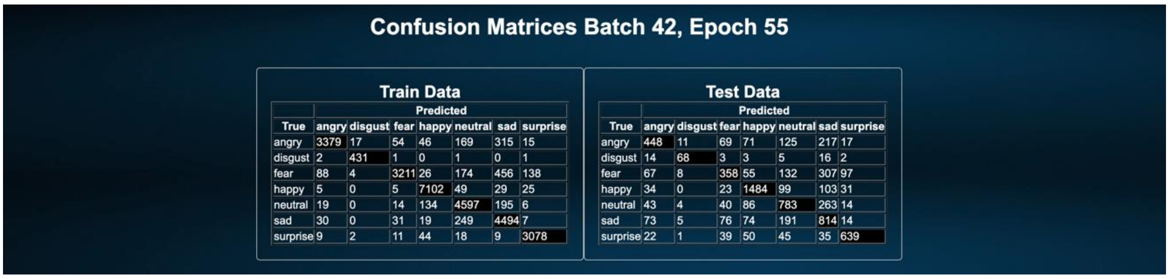
Classification Reports Batch 42, Epoch 55

Train Data

```
              precision    recall  f1-score   support

       angry       0.96      0.85      0.90      3995
     disgust       0.95      0.99      0.97       436
        fear       0.97      0.78      0.87      4097
       happy       0.96      0.98      0.97      7215
     neutral       0.87      0.93      0.90      4965
         sad       0.82      0.93      0.87      4830
    surprise       0.94      0.97      0.96      3171

    accuracy                           0.92     28709
   macro avg       0.92      0.92      0.92     28709
weighted avg       0.92      0.92      0.92     28709
```

Test Data

```
              precision    recall  f1-score   support

       angry       0.64      0.47      0.54       958
     disgust       0.70      0.61      0.65       111
        fear       0.59      0.35      0.44      1024
       happy       0.81      0.84      0.83      1774
     neutral       0.57      0.64      0.60      1233
         sad       0.46      0.65      0.54      1247
    surprise       0.79      0.77      0.78       831

    accuracy                           0.64      7178
   macro avg       0.65      0.62      0.63      7178
weighted avg       0.65      0.64      0.64      7178
```



Confusion Matrices Batch 42, Epoch 55

Train Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 3379 | 17 | 54 | 46 | 169 | 315 | 15 |
| disgust | 2 | 431 | 1 | 0 | 1 | 0 | 1 |
| fear | 88 | 4 | 3211 | 26 | 174 | 456 | 138 |
| happy | 5 | 0 | 5 | 7102 | 49 | 29 | 25 |
| neutral | 19 | 0 | 14 | 134 | 4597 | 195 | 6 |
| sad | 30 | 0 | 31 | 19 | 249 | 4494 | 7 |
| surprise | 9 | 2 | 11 | 44 | 18 | 9 | 3078 |

Test Data

| | Predicted | | | | | | |
|---|---|---|---|---|---|---|---|
| True | angry | disgust | fear | happy | neutral | sad | surprise |
| angry | 448 | 11 | 69 | 71 | 125 | 217 | 17 |
| disgust | 14 | 68 | 3 | 3 | 5 | 16 | 2 |
| fear | 67 | 8 | 358 | 55 | 132 | 307 | 97 |
| happy | 34 | 0 | 23 | 1484 | 99 | 103 | 31 |
| neutral | 43 | 4 | 40 | 86 | 783 | 263 | 14 |
| sad | 73 | 5 | 76 | 74 | 191 | 814 | 14 |
| surprise | 22 | 1 | 39 | 50 | 45 | 35 | 639 |

**Interface showing performance metrics of Epoch 42, Batch size 55**