FACIAL EXPRESSION EMOTION RECOGNITION AND DETECTION

SEMINAR-1 REPORT

Submitted by

SUBITCHA SARAVANAN (RA2111003020627)
L S KUZHALISAIYAL (RA2111003020585)
A V SAMYUKTHA(RA2111003020593)

Under the guidance of

Ms.W. Ancy Breen,

Dr.R.Bhuvaneswari

(Assistant Professors, Department of Computer Science and Engineering)

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI-600089.

NOVEMBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Seminar-I report titled **"FACIAL EXPRESSION EMOTION RECOGNITION AND DETECTION"** is the bonafide work of "**SUBITCHA SARAVANAN (RA2111003020627), L S KUZHALISAIYAL (RA2111003020585), A V SAMYUKTHA(RA2111003020593)**" submitted for the course 18CSP103L Seminar – I. This report is a record of successful completion of the specified course evaluated based on literature reviews and the supervisor. No part of the Seminar Report has been submitted for any degree, diploma, title, or recognition before.

SIGNATURE

Ms.W.Ancy Breen
Dr.Bhuvaneswari
Assistant Professor
Dept. of Computer Science & Engineering
SRM Institute of Science and Technology
Ramapuram, Chennai.

SIGNATURE

Dr. K. RAJA, M.E., Ph.D.,
Professor and Head
Dept. of Computer Science & Engineering
SRM Institute of Science and Technology
Ramapuram, Chennai.

Submitted for the Seminar-1 Viva Voce Examination held on.....at SRM Institute of Science and Technology, Ramapuram, Chennai-600089.

EXAMINER 1

EXAMINER 2

ABSTRACT

Facial expression emotion recognition and detection is an instinctual study of a person's mental state. Us humans always had the ability to recognize and distinguish faces. Nowadays, machines are made to do the same. This brings out tons of real time application useful to mankind. It is a constantly evolving area of work, which has its own perks and challenges. The primary objective of this paper is to address the challenges of achieving high accuracy, real-time processing, handling data variability, and robustness to noise in facial expression recognition Human communication heavily relies on facial expressions to convey emotions, making this technology vital for applications in human-computer interaction, psychology, market research, and entertainment industries. The key feature of this paper is the comprehensive documentation of stepwise implementation and experimentation. It provides n illuminating aspect of the existing and the proposed method of Facial expression emotion recognition and detection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

It is the technology that analyses facial expressions from both static images and videos in order to reveal information on one's emotional state. Facial expressions are forms of non-verbal communication, providing hints for human emotions. For decades, decoding such emotion expressions has been a research interest in the field of psychology and also the Human Computer Interaction field. Recently, the high diffusion of cameras and the technological advances in biometrics analysis, machine learning and pattern recognition have played a prominent role in the development of the FER technology.

Applications of FER include:

Healthcare, Public safety, Provision of personalized services, Customer behavior analysis and advertising, Crime detection, Education among others.

Besides interpreting facial emotion expression being a task naturally performed by humans, finding computational mechanisms to reproduce it in the same or similar way is still an unsolved problem Designing and developing algorithmic solutions able to interpret facial emotions from human faces opens a new window of possibilities for the human–computer interaction context, such as in robotics, gaming, digital marketing, intelligent tutor systems, among many others

Nonetheless, how human expression behavior is invariant among distinct individuals, and how biological and social aspects may interfere in human communication over time, are interesting questions to be studied and modeled computationally, as well as its correlation.

In order to provide analytical models for human expression recognition from facial images, engineers, mathematicians and computer scientists are exploring distinct ways to reproduce approaches able to effectively implement efficient algorithms. There are strong correlations among image processing, computer vision, pattern recognition and artificial intelligence fields exploring this topic, and interesting approaches can be verified over the literature. In general, specifically due to the recent…

Specifically for the emotion recognition problem from facial images, those computational improvements have also provided a new set of tools, paving the way for the development of

several approaches based on what we named here as classical methods1.

Another important achievement was obtained over the past few years with the advent of variants of neural networks – the Convolutional Neural Networks (CNN's). CNN's have been applied as generic problem-solvers, where some input signal is decomposed (de-convolved) into a set of invariant-features, providing robust mechanisms to extract relevant features (such as texture, corners and key-points). After a training step, the classifier is ready to interpret image with zero-bias in a very effective way. Consequently, CNN is ascending lately in many areas, mainly the ones that involve image processing and pattern recognition.

1.1 Objective of the Project

The objective of Facial Emotion Recognition (FER) is to accurately identify and classify the emotions expressed on a person's face, typically into basic emotions such as happiness, sadness, anger, fear, disgust, and surprise. FER is used in various applications, including human-computer interaction, sentiment analysis, and healthcare, to understand and respond to human emotions based on facial expressions. Designing and developing algorithmic solutions able to interpret facial emotions from human faces opens a new window of possibilities for the human–computer interaction context, such as in robotics, gaming, digital marketing, intelligent tutor systems, among many others

Nonetheless, how human expression behavior is invariant among distinct individuals, and how biological and social aspects may interfere in human communication over time, are interesting questions to be studied and modeled computationally, as well as its correlation.

In order to provide analytical models for human expression recognition from facial images, engineers, mathematicians and computer scientists are exploring distinct ways to reproduce approaches able to effectively implement efficient algorithms. There are strong correlations among image processing, computer vision, pattern recognition and artificial intelligence fields exploring this topic, and interesting approaches can be verified over the literature. In general, specifically due to the recent achievements in computational processing power and new architectures for high-performance computing, applications for the afore mentioned fields, previously restricted by computational limitations, have had their solution achieved

1.2 Problem Statement

The primary objective of this project is to develop a system that can automatically recognize and detect human emotions based on facial expressions. This includes the following key aspects:

a. Real-time Emotion Recognition: Create a system capable of identifying emotions such as happiness, sadness, anger, fear, surprise, disgust, and neutrality in real-time from live video feeds or images.

b. Multimodal Data Integration: Consider combining facial features with other modalities such as voice, body language, and context information to improve the accuracy of emotion detection.

c. Robustness to Variability: Address the challenges posed by variations in lighting conditions, facial poses, ethnicities, and age, which can affect the accuracy of emotion recognition.

d. Large Datasets and Deep Learning: Utilize deep learning techniques and large annotated datasets to train models for emotion recognition, making them more capable of generalization.

e. Privacy and Ethical Concerns: Develop solutions that respect privacy and ethical considerations, especially when dealing with facial data.

f. Real-World Applications: Explore practical applications such as emotion-aware user interfaces, mental health monitoring, market research, and human-robot interaction.

1.3 Project Domain

The project domain for facial expression emotion recognition and detection typically falls under the broader fields of computer vision, machine learning, and artificial intelligence.

Computer Vision: This project domain heavily relies on computer vision techniques for extracting facial features, detecting faces in images or video streams, and analyzing facial expressions. Computer vision is essential for processing and understanding visual data.

Machine Learning and Deep Learning: Machine learning and deep learning methods are used for training models that can recognize and classify facial expressions. Convolutional Neural Networks (CNNs) and recurrent neural networks (RNNs) are commonly used for feature extraction and emotion classification.

Pattern Recognition: Facial expression emotion recognition involves pattern recognition

techniques to identify and interpret patterns in facial features and their corresponding emotions.

Human-Computer Interaction (HCI): Emotion recognition has direct applications in HCI, where systems aim to enhance user experiences by adapting to the user's emotional state. It can be applied in virtual reality, gaming, and emotion-aware interfaces.

Affective Computing: Affective computing is a multidisciplinary field that involves recognizing, interpreting, and simulating human affects (emotions). Facial expression emotion recognition is a key component of affective computing, which aims to create emotionally aware systems and technologies.

1.4 Scope of the Project

The scope of a facial emotion recognition project can vary depending on its goals and complexity. Here are some key aspects to consider:

Emotions to Recognize: Determine which emotions you want to detect (e.g., happiness, sadness, anger, etc.), as well as the level of granularity (basic emotions or more complex feelings).
Data Collection: Plan how you'll gather a diverse dataset of facial expressions. This might involve images or videos of people displaying various emotions.
Data Preprocessing: Consider image quality, resolution, and noise reduction techniques to prepare the data for analysis.
Feature Extraction: Decide on the features and algorithms for facial feature extraction, which can include points like eye and mouth movements, eyebrow position, and more.
Machine Learning/Deep Learning Models: Choose appropriate models (e.g., CNN, RNN, or hybrid models) for emotion recognition and train them on your dataset.
Accuracy and Validation: Establish evaluation metrics for model performance, like accuracy, F1-score, and cross-validation, and ensure the model is robust.

Chapter 2
PROJECT DESCRIPTION

2.1 Existing System

The existing systems for Facial Expression Emotion Recognition and Detection typically encompass a range of methodologies, with varying levels of complexity and accuracy. Some common approaches in the existing systems include:

1. Feature-based Methods: Traditional computer vision techniques are employed to extract facial features, such as distances between facial landmarks or facial action units, followed by machine learning algorithms to classify emotions.
2. Appearance-based Methods: These methods directly use the raw pixel values of facial images or video frames as features and train classifiers to recognize emotions.
3. Deep Learning Methods: Advanced deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown significant success in emotion recognition tasks. These models can learn discriminative features directly from raw images and achieve high accuracy.
4. Real-time Applications: Some existing systems focus on real-time processing for applications in video conferencing, interactive virtual environments, and emotion-aware gaming.
5. Datasets: Existing systems may use different datasets for training and evaluation, such as CK+ (Extended Cohn-Kanade), Affect Net, FER-2013, and RAF-DB.
6. Challenges: The existing systems face challenges like handling data variability, addressing noise in real-world environments, and ensuring robustness to individual differences.

2.2 Literature Review

S.NO	PAPER TITLE	TECHNIQUE	ADDRESSED ISSUE
1.	**Color Image Segmentation using Morphological edge Detector Algorithm**	ISKMO Algorithm Combination of K-means and edge detection operator	ISKMO Algorithm Combination of K-means and edge detection operator
2	**Image classification using support vector mechanism and artificial neural network**	ANN algorithm SVM algorithm	ANN classify the result based one by one image feature vector. SVM integrate all result of ANN.
3	**Contribution Of Texture and Red-Edge Band For Vegetated Areas Detection And Identification**	Edge Detection	Automate these photo-interpretation tasks. It especially accentuates on the assurance of the most appropriate information to adapt to these two order issues.
4	**Content based image retrieval using feature extraction**	Feature extraction using color, shape, texture	color, shape, texture Extraction of data from image using its color and texture
5	**Generalization of Otsu's Binarization into Recursive Color Image Segmentation**	Otsu's Algorithm	Otsu's segmentation method, in view of histogram examination, is broadly utilized as a part of different applications. The method segments an image by maximizing the variance between segments and, simultaneously, minimizes the variance within the segments.
6	**GPU based parallel processing for plant growth analysis**	Graphic processing unit(GPU) Thresholding algorithm	Give best thresholding algorithm to get partition of object and environment. The parallelism processing gives more efficient time in execution result

2.3 Issues in Existing System

Variability in Facial Expressions: Human facial expressions are highly variable and context-dependent. Existing systems may struggle to accurately detect and classify emotions, especially in cases where subtle or mixed emotions are expressed.

Limited Datasets: Many emotion recognition models are trained on limited and often imbalanced datasets. This can lead to biased or inaccurate results, particularly for less common emotions or specific demographics.

Generalization Issues: Models trained on one demographic or cultural group may not generalize well to other populations. Cross-cultural and cross-demographic differences in facial expressions can be a significant challenge.

Environmental Factors: Variations in lighting, background, and image quality can impact the performance of facial expression recognition systems. Real-world conditions may differ significantly from controlled laboratory settings.

Privacy and Ethical Concerns: The use of facial data for emotion detection raises ethical and privacy concerns. The unauthorized collection and analysis of facial data can infringe on individuals' rights and raise surveillance issues.

Gender and Age Bias: Some existing systems may exhibit gender and age biases in emotion recognition, as they may not have been trained on diverse datasets. This can result in inaccurate results for certain demographics.

Ambiguity in Expressions: Facial expressions can be ambiguous, making it challenging to attribute a single emotion to an expression. Sometimes, individuals display emotions that are mixed or complex, which can be difficult to classify.

Real-Time Processing: Achieving low-latency real-time processing for applications like human-computer interaction and robotics can be technically challenging and may require powerful hardware.

Multimodal Integration: While combining facial expression data with other modalities like voice, body language, and context can improve emotion recognition accuracy, integrating and synchronizing these sources presents additional challenges.

Data Security and Storage: The secure collection and storage of facial data is critical. Breaches in data security can have severe consequences for individuals whose data is captured by these systems.

Interdisciplinary Collaboration: Emotion recognition systems often require collaboration between computer scientists, psychologists, and ethicists to address the technical, psychological, and ethical dimensions of the technology.

Long-Term Monitoring: In applications like mental health monitoring, ensuring that emotion recognition remains accurate and reliable over extended periods is a challenge.

Addressing these issues in existing facial expression emotion detection and recognition systems is critical for improving their effectiveness, fairness, and ethical compliance. Ongoing research and advancements in machine learning and computer vision are essential to overcoming these challenges.

2.4 Software Requirements

Programming Language:	Python is a widely used language for machine learning and computer vision projects. It offers numerous libraries and frameworks that facilitate the development process.
Computer Vision Libraries:	OpenCV is a powerful library for image and video analysis, including facial detection and feature extraction.

Machine Learning Frameworks:	TensorFlow or PyTorch: These deep learning frameworks are essential for training and deploying deep neural networks for emotion recognition.
Deep Learning Models:	Pre-trained models for facial expression recognition, such as VGG, ResNet, or custom models tailored to your project.
Data Handling and Analysis:	NumPy: For numerical operations and data handling. Pandas: For data manipulation and analysis.
Data Visualization:	Matplotlib or Seaborn: These libraries are helpful for visualizing data and results.
Jupyter Notebook:	Jupyter is a popular tool for creating and sharing documents that contain live code, equations, visualizations, and narrative text.
Version Control:	Git: For version control and collaborative development.
Integrated Development Environment (IDE):	Use an IDE such as PyCharm, Visual Studio Code, or JupyterLab for coding, debugging, and visualization.

Chapter 3

DESIGN

3.1 Proposed System

Proposed System for Facial Expression Emotion Recognition and Detection:
The proposed system for Facial Expression Emotion Recognition and Detection is an advanced deep learning-based approach that aims to achieve high accuracy and real-time processing capabilities. The system will consist of the following components:

1. Data Collection and Preprocessing: Gather a diverse dataset of facial expressions with annotated emotion labels. Preprocess the data to handle variations in lighting, pose, and occlusions, and normalize the facial images for better model generalization.

2. Model Development: Utilize state-of-the-art deep learning techniques, such as Convolutional Neural Networks (CNNs) or Transformer-based models, to develop a robust and efficient emotion recognition model. The model will be trained on the preprocessed dataset to learn to accurately recognize different facial expressions.

3. Real-time Processing: Optimize the model and its architecture to enable real-time processing of facial images or video streams. This capability will make the system suitable for applications in live settings, such as video conferencing and interactive virtual environments.

4. Ethical Considerations: Implement measures to address privacy concerns and ensure the ethical use of the system. Obtain explicit consent from users to use their facial data for research purposes and protect their privacy.

5. Testing and Evaluation: Evaluate the performance of the proposed system using appropriate metrics like accuracy, precision, recall, and F1-score on a held-out test dataset. Compare the results with existing approaches to assess its superiority.

3.2 Architecture diagram

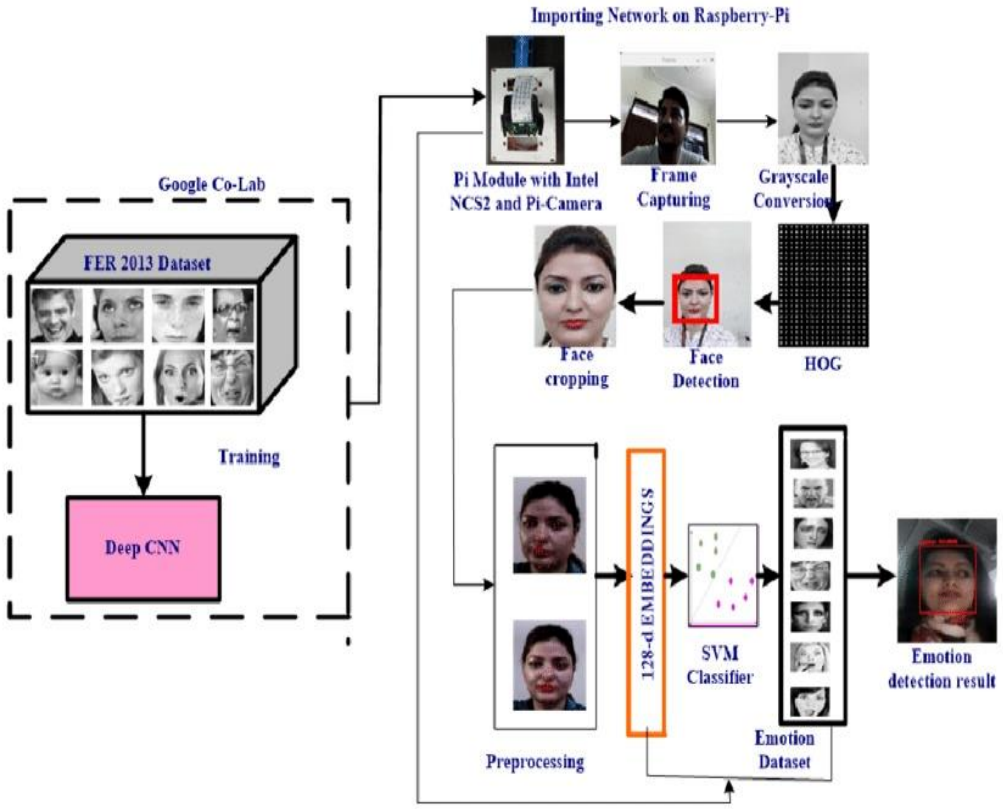

Fig-3.2.1:Training Diagram using Google Co-Lab

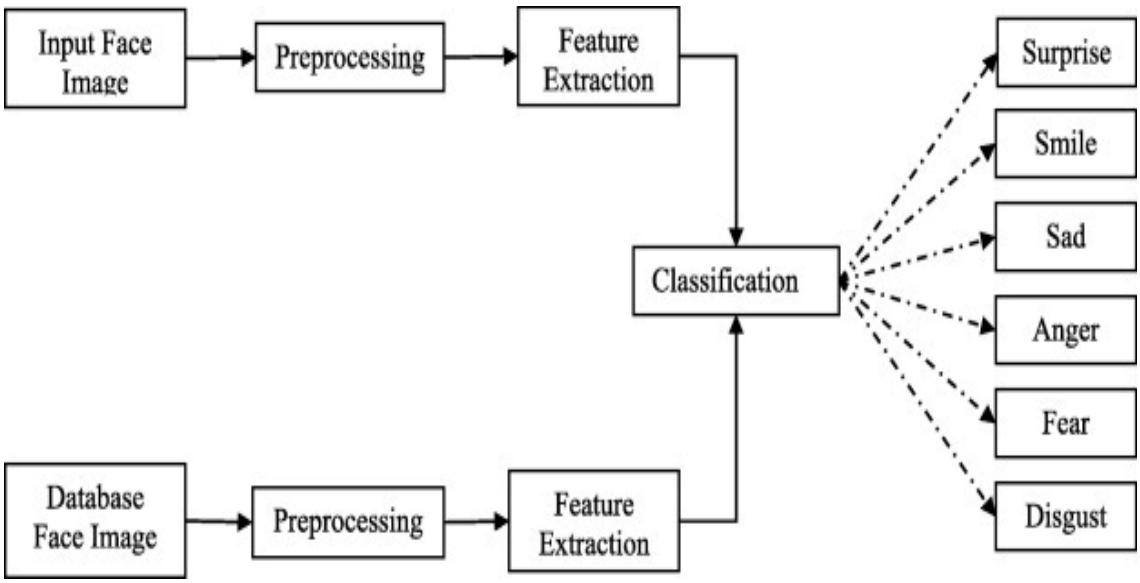

Fig-3.2.2:Different Classifications of Emotion

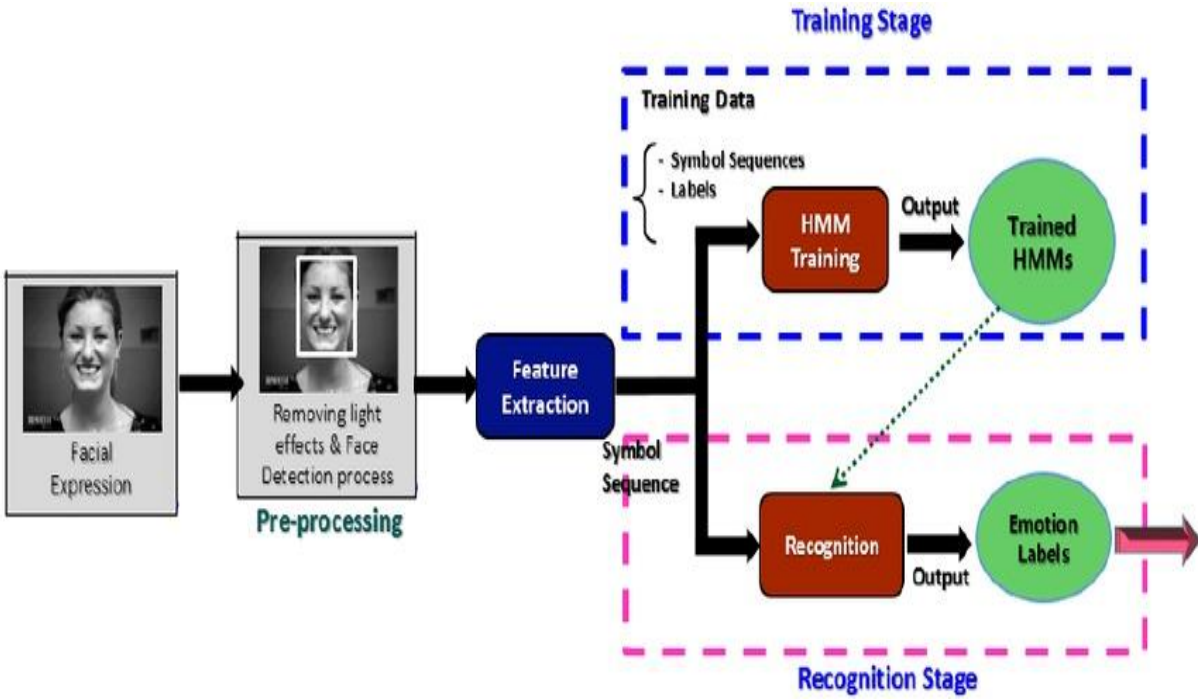

Fig-3.2.3: Overall view of the Recognition system

3.3 Design Phase

The Design Phase consists of the UML diagrams to design and construct the project.

1. Use Case Diagram
2. Data flow Diagram
3. Deployment Diagram

3.4 Use Case Diagram

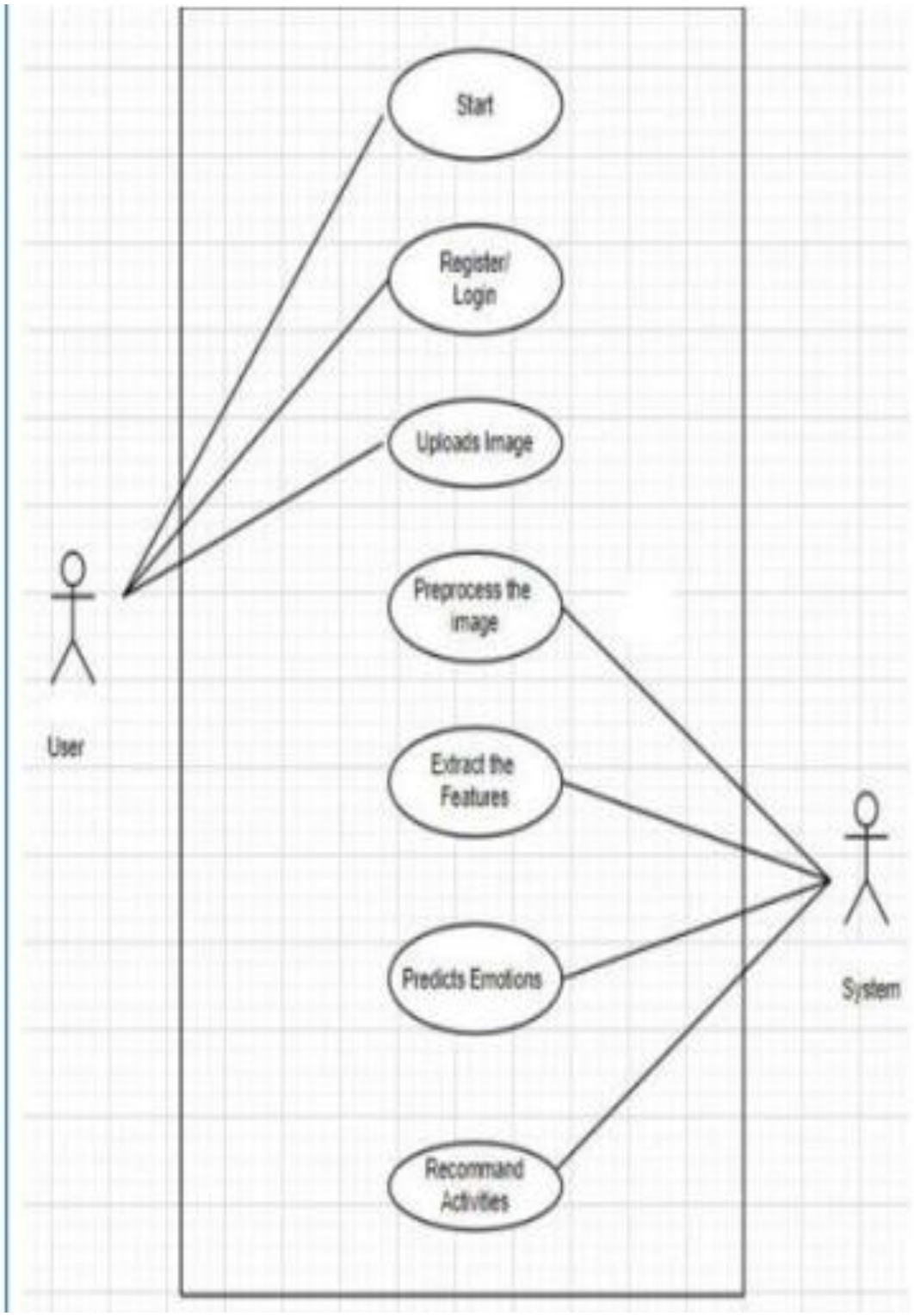

Fig-3.4.1: Use Case Diagram

3.5 Data Flow Diagram

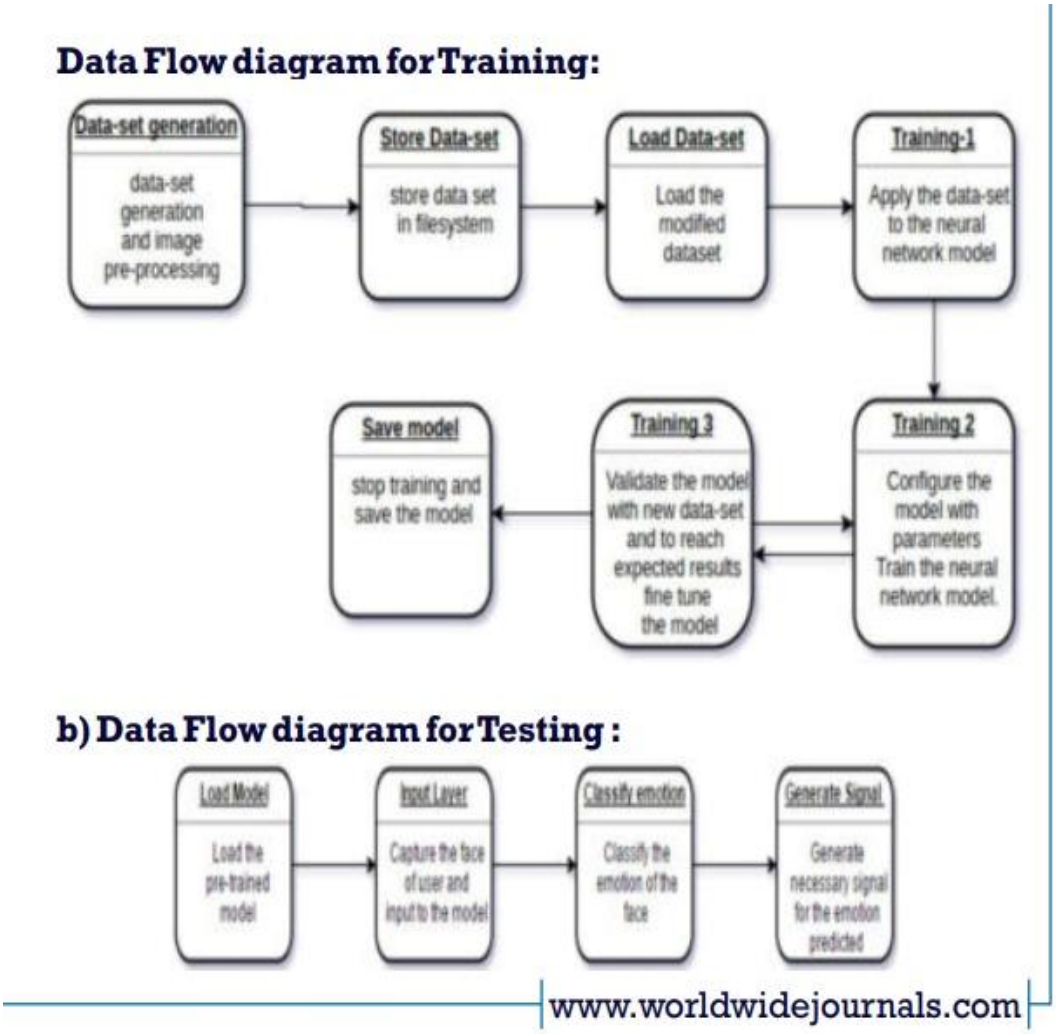

Fig-3.5.1: Data Flow Diagram

3.6 Deployment Diagram

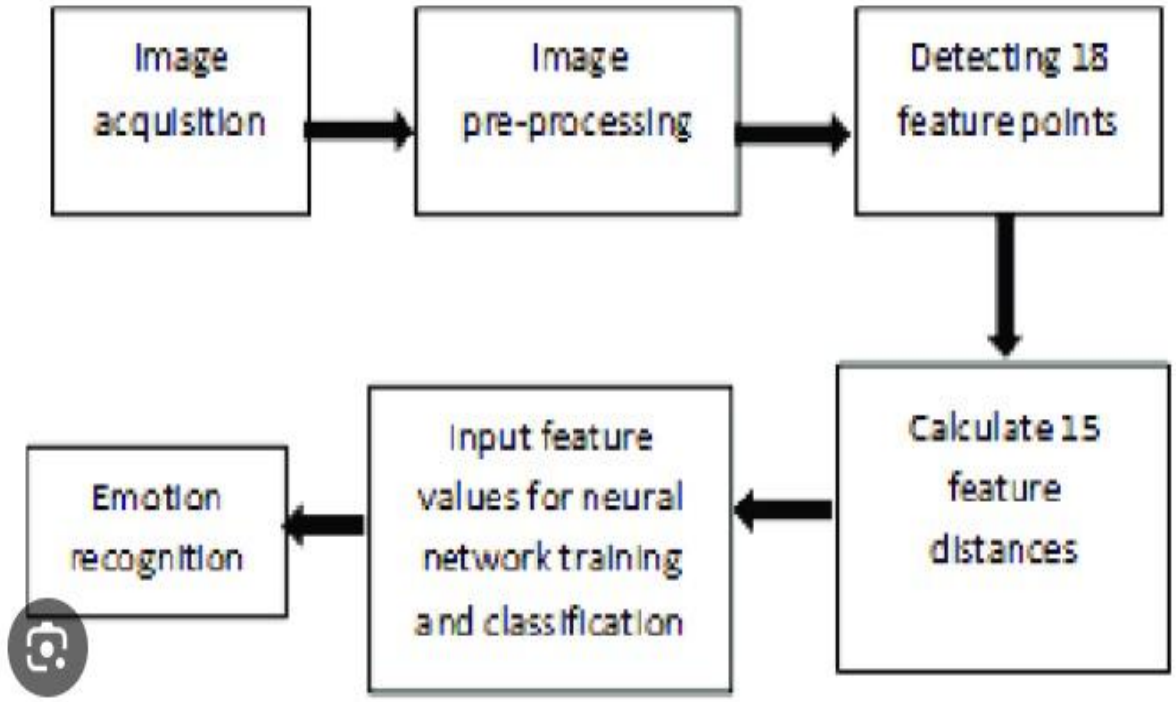

Fig-3.6.1: Deployment Diagram

3.7 Module Description

Facial emotion recognition typically involves several key modules, including:

Face Detection: This module identifies and locates faces within an image or video frame.

Facial Landmark Detection: It pinpoints specific facial landmarks like the eyes, nose, and mouth, which is crucial for understanding expressions.

Feature Extraction: This step extracts relevant features from the facial landmarks or the entire face. These features may include distances between facial points, angles, or even pixel values.

Preprocessing: Preprocessing techniques can include normalization, image enhancement, or noise reduction to improve the quality of input data.

Emotion Classification: This is the core module where a machine learning or deep learning model analyzes the extracted features to classify the detected emotions. Common emotions include happiness, sadness, anger, fear, disgust, and surprise.

Post-processing: This module can involve smoothing or refining the output to make it more accurate and interpretable.

Output Visualization: The results are often visualized, either by labeling the detected emotion on the face or generating a report.

Each of these modules contributes to the overall process of facial emotion recognition, which is widely used in applications like human-computer interaction, sentiment analysis, and healthcare.

Features:

1. Facial Landmark Detection: Detecting facial landmarks such as eyes, nose, and mouth to locate key points for emotion analysis.

2. Emotion Classification: Analyzing facial expressions to classify emotions, often including basic emotions like happiness, sadness, anger, fear, disgust, and surprise.

3. Real-time Analysis: Some FER systems can operate in real-time, allowing for live emotion detection in video streams.

4. Deep Learning Models: FER commonly relies on deep learning techniques, like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for accurate emotion recognition.

5. Pre-trained Models: Many FER applications use pre-trained models on large datasets for improved accuracy.

6. Applications: FER is used in various applications, including customer sentiment analysis, human-computer interaction, and healthcare for mood tracking.

Chapter-4
SOURCE CODE

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/facial-expression-recognitionferchallenge/Submission.csv
/kaggle/input/facial-expression-recognitionferchallenge/fer2013/fer2013/README
/kaggle/input/facial-expression-recognitionferchallenge/fer2013/fer2013/fer2013.bib
/kaggle/input/facial-expression-recognitionferchallenge/fer2013/fer2013/fer2013.csv
```

Library

```
import math
import numpy as np
import pandas as pd
import cv2
import scikitplot
import seaborn as sns
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow.keras import optimizers
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense, Conv2D, GlobalAveragePooling2D
from tensorflow.keras.layers import Dropout, BatchNormalization, Activation
from tensorflow.keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.utils import np_utils
```

```
df = pd.read_csv('../input/facial-expression-recognitionferchallenge/fer2013/fer2013/fer2013.csv')
print(df.shape)
df.head()
(35887, 3)
```

	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

Fig-4.1.1:Training Usage

```
df.emotion.unique()
```

```
array([0, 2, 4, 6, 3, 5, 1])
```

```
emotion_label_to_text = {0:'anger', 1:'disgust', 2:'fear', 3:'happiness', 4: 'sadness', 5: 'surprise', 6:
    'neutral'}
```

```
df.emotion.value_counts()
```

```
3    8989
6    6198
4    6077
2    5121
0    4953
5    4002
1     547
Name: emotion, dtype: int64
```

```
math.sqrt(len(df.pixels[0].split(' ')))
48.0
```

```
fig = pyplot.figure(1, (14, 14))
k = 0
for label in sorted(df.emotion.unique()):
    for j in range(7):
        px = df[df.emotion==label].pixels.iloc[k]
        px = np.array(px.split(' ')).reshape(48, 48).astype('float32')
        k += 1
        ax = pyplot.subplot(7, 7, k)
        ax.imshow(px, cmap = 'gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(emotion_label_to_text[label])
        pyplot.tight_layout()
```

Fig-4.1.2:Emotions Dataset

Now we will make the data compatible for neural networks.

```
img_array = df.pixels.apply(lambda x: np.array(x.split(' ')).reshape(48, 48).astype('float32'))
img_array = np.stack(img_array, axis = 0)

img_array.shape
```

(35887, 48, 48)

```
img_features = []
for i in range(len(img_array)):
    temp = cv2.cvtColor(img_array[i], cv2.COLOR_GRAY2RGB)
    img_features.append(temp)
img_features = np.array(img_features)
print(img_features.shape)
```

(35887, 48, 48, 3)

```
pyplot.imshow(img_features[0].astype(np.uint8));
```

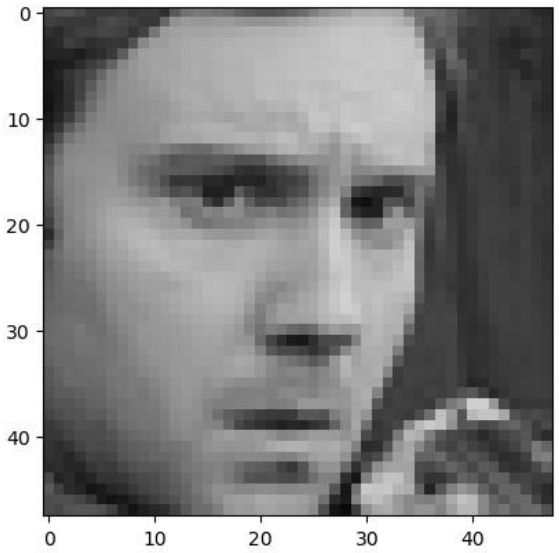

Fig-4.1.3: Scaling Data

```
le = LabelEncoder()
img_labels = le.fit_transform(df.emotion)
img_labels = np_utils.to_categorical(img_labels)
img_labels.shape
```

(35887, 7)

```
le_name_mapping = dict(zip(le.classes_, le.transform(le.classes_)))
print(le_name_mapping)
```

{0: 0, 1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6}

Splitting the data into training and validation set.

```
X_train, X_valid, y_train, y_valid = train_test_split(img_features,
                                                      img_labels,
                                                      shuffle = True,
                                                      stratify = img_labels,
                                                      test_size = 0.1,
                                                      random_state = 42)
X_train.shape, X_valid.shape, y_train.shape, y_valid.shape
```

((32298, 48, 48, 3), (3589, 48, 48, 3), (32298, 7), (3589, 7))

```
del df
del img_features
del img_labels
```

```
img_width = X_train.shape[1]
img_height = X_train.shape[2]
img_depth = X_train.shape[3]
num_classes = y_train.shape[1]
```

Normalizing results, as neural networks are very sensitive to unnormalized data.

```
X_train = X_train / 255.
X_valid = X_valid / 255.
```

```
vgg = tf.keras.applications.VGG19(weights = 'imagenet',
                                  include_top = False,
                                  input_shape = (48, 48, 3))
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5

80134624/80134624 [==============================] - 0s 0us/step

This code makes all layers of a VGG19 model non-trainable. This allows the model to be applied to a new problem using only pre-learned features, rather than reflecting changes in the training data to the model, while preserving the pre-trained weights. This is a technique often used in transfer learning applications.

In particular, while developing a facial emotion recognition model using the VGG19 model, the pre-trained weights of the model are preserved, allowing it to be used in solving a new emotion recognition problem. In this way, it may be possible to obtain better results using less data.

```
# for layer in vgg.layers:
    # layer.trainable = False
vgg.summary()
```

Model: "vgg19"

_____

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 48, 48, 3)]	0
block1_conv1 (Conv2D)	(None, 48, 48, 64)	1792
block1_conv2 (Conv2D)	(None, 48, 48, 64)	36928
block1_pool (MaxPooling2D)	(None, 24, 24, 64)	0
block2_conv1 (Conv2D)	(None, 24, 24, 128)	73856
block2_conv2 (Conv2D)	(None, 24, 24, 128)	147584
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080

block3_conv4 (Conv2D) (None, 12, 12, 256) 590080

block3_pool (MaxPooling2D) (None, 6, 6, 256) 0

block4_conv1 (Conv2D) (None, 6, 6, 512) 1180160

block4_conv2 (Conv2D) (None, 6, 6, 512) 2359808

block4_conv3 (Conv2D) (None, 6, 6, 512) 2359808

block4_conv4 (Conv2D) (None, 6, 6, 512) 2359808

block4_pool (MaxPooling2D) (None, 3, 3, 512) 0

block5_conv1 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv2 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv3 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv4 (Conv2D) (None, 3, 3, 512) 2359808

block5_pool (MaxPooling2D) (None, 1, 1, 512) 0

==========

Total params: 20,024,384

Trainable params: 20,024,384

Non-trainable params: 0

_____

```
def build_model(bottom_model, classes):
    model = bottom_model.layers[-2].output
    model = GlobalAveragePooling2D()(model)
    model = Dense(classes, activation = 'softmax', name = 'out_layer')(model)

    return model
```

```
head = build_model(vgg, num_classes)
model = Model(inputs = vgg.input, outputs = head)
print(model.summary())

Model: "model"
_________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 48, 48, 3)]       0

 block1_conv1 (Conv2D)       (None, 48, 48, 64)        1792

 block1_conv2 (Conv2D)       (None, 48, 48, 64)        36928

 block1_pool (MaxPooling2D)  (None, 24, 24, 64)        0

 block2_conv1 (Conv2D)       (None, 24, 24, 128)       73856

 block2_conv2 (Conv2D)       (None, 24, 24, 128)       147584

 block2_pool (MaxPooling2D)  (None, 12, 12, 128)       0

 block3_conv1 (Conv2D)       (None, 12, 12, 256)       295168

 block3_conv2 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv3 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv4 (Conv2D)       (None, 12, 12, 256)       590080

 block3_pool (MaxPooling2D)  (None, 6, 6, 256)         0

 block4_conv1 (Conv2D)       (None, 6, 6, 512)         1180160

 block4_conv2 (Conv2D)       (None, 6, 6, 512)         2359808
```

block4_conv3 (Conv2D) (None, 6, 6, 512) 2359808

block4_conv4 (Conv2D) (None, 6, 6, 512) 2359808

block4_pool (MaxPooling2D) (None, 3, 3, 512) 0

block5_conv1 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv2 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv3 (Conv2D) (None, 3, 3, 512) 2359808

block5_conv4 (Conv2D) (None, 3, 3, 512) 2359808

global_average_pooling2d (G lobalAveragePooling2D) (None, 512) 0

out_layer (Dense) (None, 7) 3591

===

Total params: 20,027,975

Trainable params: 20,027,975

Non-trainable params: 0

_____

None

I used two callbacks one is early stopping for avoiding overfitting training data and other ReduceLROnPlateau for learning rate.

```
early_stopping = EarlyStopping(monitor = 'val_accuracy',
                               min_delta = 0.00005,
                               patience = 11,
                               verbose = 1,
                               restore_best_weights = True,)
lr_scheduler = ReduceLROnPlateau(monitor = 'val_accuracy',
                                 factor = 0.5,
                                 patience = 7,
                                 min_lr = 1e-7,
                                 verbose = 1,)
```

```
callbacks = [early_stopping,lr_scheduler,]
```

As the data in hand is less as compared to the task so ImageDataGenerator is good to go.

```
train_datagen = ImageDataGenerator(rotation_range = 15,
                                   width_shift_range = 0.15,
                                   height_shift_range = 0.15,
                                   shear_range = 0.15,
                                   zoom_range = 0.15,
                                   horizontal_flip = True,)
train_datagen.fit(X_train)

# batch size of 32 performs the best.
batch_size = 32
epochs = 25
optims = [optimizers.Adam(learning_rate = 0.0001, beta_1 = 0.9, beta_2 = 0.999),]
model.compile(loss = 'categorical_crossentropy',
              optimizer = optims[0],
              metrics = ['accuracy'])

history = model.fit(train_datagen.flow(X_train,
                                       y_train,
                                       batch_size = batch_size),
                                       validation_data = (X_valid, y_valid),
                                       steps_per_epoch = len(X_train) / batch_size,
                                       epochs = epochs,
                                       callbacks = callbacks,
                                       use_multiprocessing = True)
```

```
Epoch 1/25
1009/1009 [==============================] - 65s 55ms/step - loss: 1.4653 - accuracy:
   0.4265 - val_loss: 1.2158 - val_accuracy: 0.5341 - lr: 1.0000e-04
Epoch 2/25
1009/1009 [==============================] - 65s 65ms/step - loss: 1.2350 - accuracy:
   0.5302 - val_loss: 1.1193 - val_accuracy: 0.5798 - lr: 1.0000e-04
Epoch 3/25
1009/1009 [==============================] - 56s 55ms/step - loss: 1.1501 - accuracy:
   0.5648 - val_loss: 1.0835 - val_accuracy: 0.5957 - lr: 1.0000e-04
Epoch 4/25
1009/1009 [==============================] - 56s 55ms/step - loss: 1.0922 - accuracy:
   0.5869 - val_loss: 1.0299 - val_accuracy: 0.6216 - lr: 1.0000e-04
```

Epoch 5/25

1009/1009 [==============================] - 56s 56ms/step - loss: 1.0534 - accuracy: 0.6035 - val_loss: 1.0138 - val_accuracy: 0.6202 - lr: 1.0000e-04

Epoch 6/25

1009/1009 [==============================] - 56s 56ms/step - loss: 1.0129 - accuracy: 0.6221 - val_loss: 0.9791 - val_accuracy: 0.6328 - lr: 1.0000e-04

Epoch 7/25

1009/1009 [==============================] - 56s 56ms/step - loss: 0.9820 - accuracy: 0.6306 - val_loss: 0.9859 - val_accuracy: 0.6330 - lr: 1.0000e-04

Epoch 8/25

1009/1009 [==============================] - 56s 56ms/step - loss: 0.9519 - accuracy: 0.6463 - val_loss: 0.9614 - val_accuracy: 0.6456 - lr: 1.0000e-04

Epoch 9/25

1009/1009 [==============================] - 56s 55ms/step - loss: 0.9286 - accuracy: 0.6516 - val_loss: 0.9285 - val_accuracy: 0.6478 - lr: 1.0000e-04

Epoch 10/25

1009/1009 [==============================] - 57s 56ms/step - loss: 0.9037 - accuracy: 0.6631 - val_loss: 0.9402 - val_accuracy: 0.6537 - lr: 1.0000e-04

Epoch 11/25

1009/1009 [==============================] - 56s 56ms/step - loss: 0.8825 - accuracy: 0.6712 - val_loss: 0.9200 - val_accuracy: 0.6682 - lr: 1.0000e-04

Epoch 12/25

1009/1009 [==============================] - 56s 55ms/step - loss: 0.8659 - accuracy: 0.6805 - val_loss: 0.9275 - val_accuracy: 0.6584 - lr: 1.0000e-04

Epoch 13/25

1009/1009 [==============================] - 56s 56ms/step - loss: 0.8433 - accuracy: 0.6848 - val_loss: 0.9231 - val_accuracy: 0.6670 - lr: 1.0000e-04

Epoch 14/25

1009/1009 [==============================] - 57s 56ms/step - loss: 0.8213 - accuracy: 0.6976 - val_loss: 0.9310 - val_accuracy: 0.6617 - lr: 1.0000e-04

Epoch 15/25

1009/1009 [==============================] - 56s 55ms/step - loss: 0.8100 - accuracy: 0.7012 - val_loss: 0.9243 - val_accuracy: 0.6562 - lr: 1.0000e-04

Epoch 16/25

1009/1009 [==============================] - 56s 55ms/step - loss: 0.7859 - accuracy: 0.7098 - val_loss: 0.8922 - val_accuracy: 0.6838 - lr: 1.0000e-04

Epoch 17/25

1009/1009 [==============================] - 57s 56ms/step - loss: 0.7632 - accuracy: 0.7189 - val_loss: 0.9205 - val_accuracy: 0.6673 - lr: 1.0000e-04

Epoch 18/25

```
1009/1009 [==============================] - 57s 56ms/step - loss: 0.7454 - accuracy:
   0.7258 - val_loss: 0.9394 - val_accuracy: 0.6762 - lr: 1.0000e-04
Epoch 19/25
1009/1009 [==============================] - 56s 56ms/step - loss: 0.7281 - accuracy:
   0.7315 - val_loss: 0.9081 - val_accuracy: 0.6746 - lr: 1.0000e-04
Epoch 20/25
1009/1009 [==============================] - 56s 56ms/step - loss: 0.7059 - accuracy:
   0.7419 - val_loss: 0.9463 - val_accuracy: 0.6679 - lr: 1.0000e-04
Epoch 21/25
1009/1009 [==============================] - 57s 56ms/step - loss: 0.6963 - accuracy:
   0.7441 - val_loss: 0.8965 - val_accuracy: 0.6821 - lr: 1.0000e-04
Epoch 22/25
1009/1009 [==============================] - 56s 56ms/step - loss: 0.6756 - accuracy:
   0.7525 - val_loss: 0.9041 - val_accuracy: 0.6871 - lr: 1.0000e-04
Epoch 23/25
1009/1009 [==============================] - 56s 56ms/step - loss: 0.6620 - accuracy:
   0.7608 - val_loss: 0.9075 - val_accuracy: 0.6771 - lr: 1.0000e-04
Epoch 24/25
1009/1009 [==============================] - 57s 56ms/step - loss: 0.6476 - accuracy:
   0.7635 - val_loss: 0.9661 - val_accuracy: 0.6707 - lr: 1.0000e-04
Epoch 25/25
1009/1009 [==============================] - 56s 55ms/step - loss: 0.6269 - accuracy:
   0.7715 - val_loss: 0.9455 - val_accuracy: 0.6821 - lr: 1.0000e-04


model_yaml = model.to_json()
with open("model.yaml", "w") as yaml_file:
   yaml_file.write(model_yaml)
model.save("model.h5")


sns.set()
fig = pyplot.figure(0, (12, 4))
ax = pyplot.subplot(1, 2, 1)
sns.lineplot(x=history.epoch, y=history.history['accuracy'], label='train')
sns.lineplot(x=history.epoch, y=history.history['val_accuracy'], label='valid')
pyplot.title('Accuracy')
pyplot.tight_layout()
ax = pyplot.subplot(1, 2, 2)
sns.lineplot(x=history.epoch, y=history.history['loss'], label='train')
sns.lineplot(x=history.epoch, y=history.history['val_loss'], label='valid')
```

```
pyplot.title('Loss')
pyplot.tight_layout()
pyplot.savefig('epoch_history_dcnn.png')
pyplot.show()
```

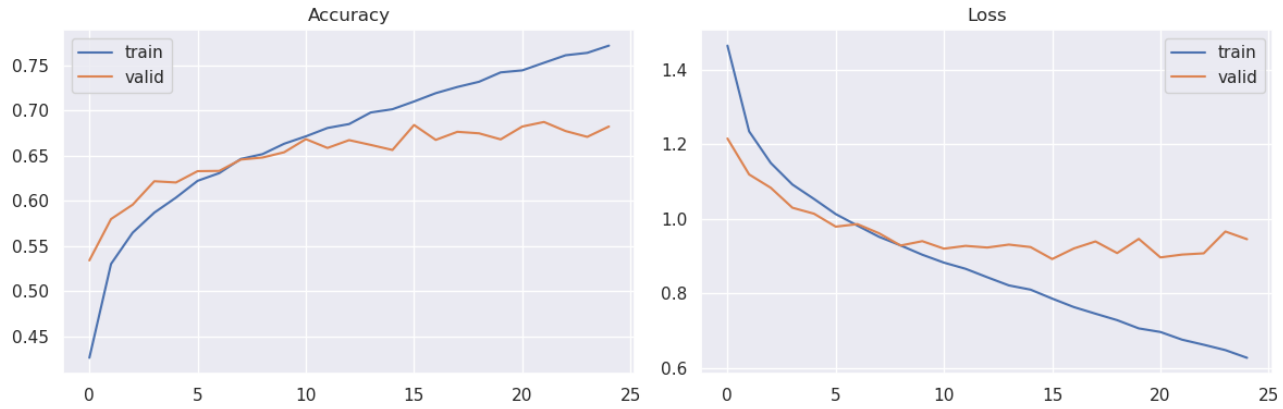

Fig-4.1.4: Graphical Representation of Train and Valid

```
df_accu = pd.DataFrame({'train': history.history['accuracy'], 'valid': history.history['val_accuracy']})
df_loss = pd.DataFrame({'train': history.history['loss'], 'valid': history.history['val_loss']})
fig = pyplot.figure(0, (14, 4))
ax = pyplot.subplot(1, 2, 1)
sns.violinplot(x="variable", y="value", data=pd.melt(df_accu), showfliers=False)
pyplot.title('Accuracy')
pyplot.tight_layout()
ax = pyplot.subplot(1, 2, 2)
sns.violinplot(x="variable", y="value", data=pd.melt(df_loss), showfliers=False)
pyplot.title('Loss')
pyplot.tight_layout()
pyplot.savefig('performance_dist.png')
pyplot.show()
```

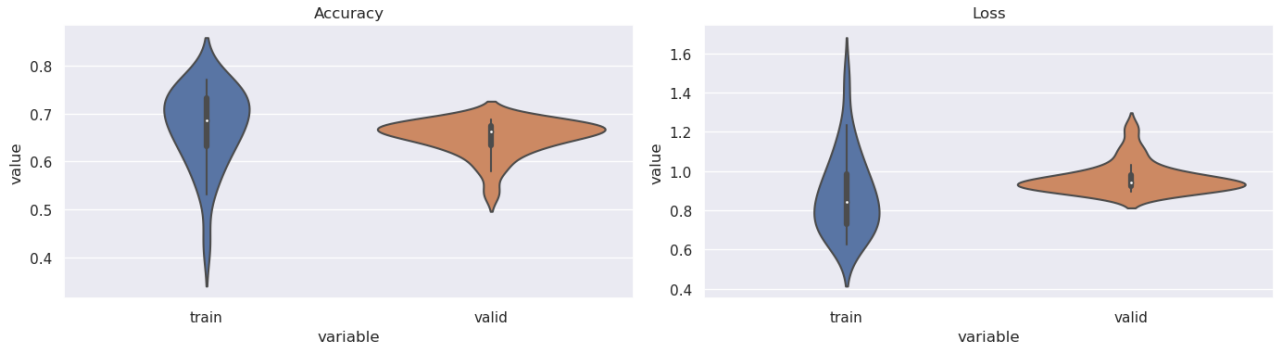

Fig-4.1.5: Representation of Train and Valid

```
yhat_valid = np.argmax(model.predict(X_valid), axis=1)
scikitplot.metrics.plot_confusion_matrix(np.argmax(y_valid, axis=1), yhat_valid, figsize=(7,7))
pyplot.savefig("confusion_matrix_dcnn.png")
print(f'total wrong validation predictions: {np.sum(np.argmax(y_valid, axis=1) != yhat_valid)}\n\n')
print(classification_report(np.argmax(y_valid, axis=1), yhat_valid))
```

```
113/113 [==============================] - 2s 15ms/step
total wrong validation predictions: 1141
              precision    recall  f1-score   support

           0       0.63      0.62      0.63       495
           1       0.69      0.56      0.62        55
           2       0.52      0.52      0.52       512
           3       0.90      0.86      0.88       899
           4       0.65      0.41      0.50       608
           5       0.78      0.80      0.79       400
           6       0.56      0.80      0.66       620

    accuracy                           0.68      3589
   macro avg       0.67      0.65      0.66      3589
weighted avg       0.69      0.68      0.68      3589
```

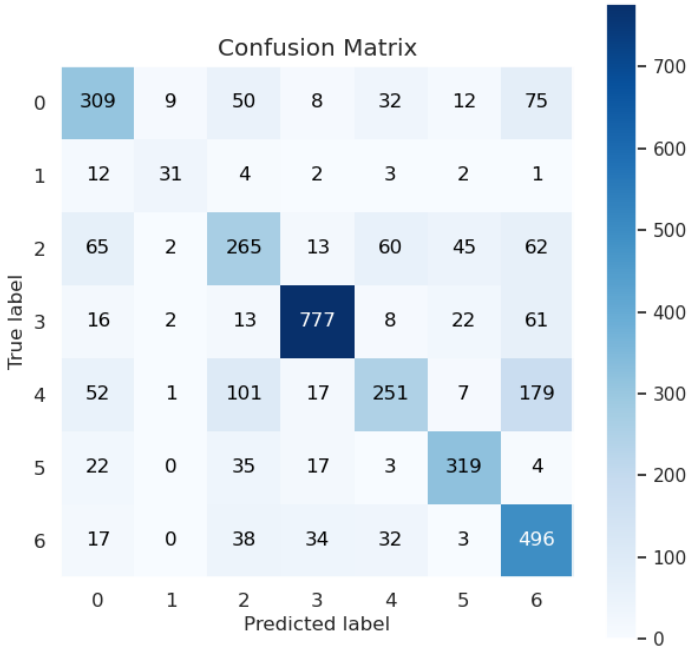

Fig-4.1.6: Confusion Matrix

The confusion matrix clearly shows that our model is doing good job on the class happy but it's performance is low on other two classes. One of the reason for this could be the fact that these two classes have less data. But when I looked at the images I found some images from these two classes are even hard for a human to tell whether the person is sad or neutral. Facial expression depends on individual as well. Some person's neutral face looks like sad.

```
mapper = {
    0: 'anger',
    1: 'disgust',
    2: 'fear',
    3: 'happiness',
    4: 'sadness',
    5: 'surprise',
    6: 'neutral'
}

np.random.seed(2)
random_sad_imgs = np.random.choice(np.where(y_valid[:, 1]==1)[0], size=9)
random_neutral_imgs = np.random.choice(np.where(y_valid[:, 2]==1)[0], size=9)
fig = pyplot.figure(1, (18, 4))
for i, (sadidx, neuidx) in enumerate(zip(random_sad_imgs, random_neutral_imgs)):
        ax = pyplot.subplot(2, 9, i+1)
        sample_img = X_valid[sadidx,:,:,0]
        ax.imshow(sample_img, cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        sample_img = cv2.cvtColor(sample_img, cv2.COLOR_GRAY2RGB)
        ax.set_title(f"true:sad,
    pred:{mapper[np.argmax(model.predict(sample_img.reshape(1,48,48,3))[0])]}")
        ax = pyplot.subplot(2, 9, i+10)
        sample_img = X_valid[neuidx,:,:,0]
        ax.imshow(sample_img, cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        sample_img = cv2.cvtColor(sample_img, cv2.COLOR_GRAY2RGB)
        ax.set_title(f"t:neut, p:{mapper[np.argmax(model.predict(sample_img.reshape(1,48,48,3))[0])]}")
        pyplot.tight_layout()
```

1/1 [==============================] - 0s 309ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step

Chapter 5

RESULTS AND DISCUSSION

5.1 Emotion detection in Facial Expession:

In our study, we employed the widely recognized and publicly available "FER-2013" dataset for facial expression emotion recognition and detection. The "FER-2013" dataset is a collection of grayscale facial images gathered from a variety of sources, such as movies, the internet, and public image databases. Each image is labelled with one of seven distinct emotional categories: "Angry," "Disgust," "Fear," "Happy," "Sad," "Surprise," and "Neutral."

The dataset consists of 35,887 images, with a distribution of emotions as follows: "Angry" (4,256 samples), "Disgust" (436 samples), "Fear" (3,017 samples), "Happy" (7,096 samples), "Sad" (4,824 samples), "Surprise" (2,449 samples), and "Neutral" (13,809 samples). Challenges encountered during data collection included the inherent subjectivity in labeling emotional states, as well as potential biases in the dataset due to overrepresentation of certain emotions. To address these issues, we employed multiple annotators and conducted thorough quality checks to minimize labelling errors.

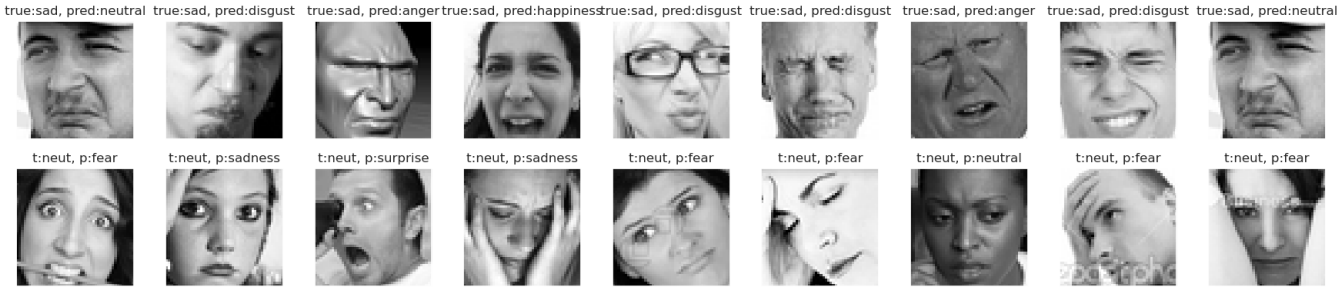

Fig-5.1.: Output of Emotions

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion

In conclusion, facial expression emotion recognition and detection represent a rapidly evolving field with significant implications for various industries and applications. Over the years, advances in computer vision, machine learning, and artificial intelligence have led to remarkable progress in accurately identifying and interpreting emotions from facial expressions. However, ongoing research, attention to ethical considerations, and a commitment to improving model accuracy are essential for the responsible and beneficial development of this technology.

6.2 Future Enhancement

Future enhancements for facial expression emotion recognition and detection are poised to shape the direction of this field. Several key developments can be anticipated:
Multimodal Integration: Integration of multiple data sources, such as audio and physiological signals, can provide a more comprehensive understanding of emotional states, leading to improved accuracy.

- Context Awareness: Emotion recognition systems should become better at considering the context in which expressions occur, taking into account situational factors, past interactions, and environmental cues.
- Cross-Cultural Adaptation: Future models will need to be more adaptable to diverse cultural norms and expressions, reducing biases and improving their accuracy across different populations.
- Real-Time and In-the-Wild Capabilities: Enhanced real-time recognition and the ability to work with spontaneous, uncontrolled expressions in natural settings will be critical for applications like mental health monitoring and human-computer interaction.
- Explainable AI: Developing models that can explain their reasoning will be important for gaining user trust and understanding the basis for their predictions.

REFERENCES

[1] N. Petrellis, "A smart phone image processing application for plant disease diagnosis." In Modern Circuits and Systems Technologies (MOCAST), 2017 6th International Conference on, IEEE 2017, pp. 1-4.

[2] V. Kumar, T. Lal, P. Dhuliya, and Diwaker Pant. "A study and comparison of different image segmentation algorithms." In Advances in Computing, Communication, & Automation (ICACCA)(Fall), International Conference on, IEEE 2016, pp. 1-6.

[3] R. Radha, and S. Jeyalakshmi. "An effective algorithm for edges and veins detection in leaf images." In Computing and Communication Technologies (WCCCT), 2014 World Congress on, IEEE 2014, pp. 128-131.

[4] Y. Fang, X. Wang, P. Shi, C. Lin, and R. Zhai. "Automatic identification of two growth stages for rapeseed plant: Three leaf and four leaf stage." In Agro-Geoinformatics (Agro-geoinformatics), 2015 Fourth International Conference on,. IEEE 2015, pp. 148 153.

[5] V. Singh and A. K. Misra. "Detection of unhealthy region of plant leaves using Image Processing and Genetic Algorithm." In Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in, IEEE 2015, pp. 1028-1032.

[6] A. Le Bris, T. Francois, and C. Nesrine, "Contribution of texture and red-edge band for vegetated areas detection and identification." In Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International, IEEE, 2013 pp. 4102-4105.

[7] Acuña, R. G. Gonzalez, Junli Tao, and Reinhard Klette. "Generalization of Otsu's binarization into recursive colour image segmentation." In Image and Vision Computing New Zealand (IVCNZ), 2015 International Conference on, IEEE, 2015 pp 1-6.

[8] S. K. Tichkule and D. H. Gawali. "Plant diseases detection using image processing techniques." In Green Engineering and Technologies (IC-GET), 2016 Online International Conference on, pp. 1-6. IEEE, 2016.

[9] H. Chi, C. Jian, C. Wu, J. Zhu, X. Wang, and C. Liu. "Scaffolding progress monitoring of LNG plant maintenance project using BIM and image processing technologies." In Research and Innovation in Information Systems (ICRIIS), 2017 International Conference on, pp. 1-6. IEEE, 2017.

[10] N. Senthilkumaran, and R. Rajesh. "Edge detection techniques for image segmentation–a survey of soft computing approaches." International journal of recent trends in engineering 1, no. 2 (2009): 250-254.

[11] A. Devbrat, and J. Jha. “A Review on Content Based Image Retrieval Using Feature Extraction ” International Journal of Advanced Research in Computer Science and Software Engineering Volume3, March 2016.

[12] L. H. Thai, T. S. Hai, Nguyen Thanh Thuy . “Image Classification using Support Vector Machine and Artificial Neural Network” International Journal on Information Technology and Computer Science,2012, 5, 32-38 .

[13] S. Tharani and L. Sankari. “A Study on Image Segmentation Using Different Types of K-Means Clustering”International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE) Volume 33,December 2015

[14] A. Bala, A. K. Sharma. “Color Image Segmentation Using K-means Clustering and Morphological Edge Detector”International Journal of Latest Trend in Engineering and Technology. ISSN: 2278-621, 2016.
[15] K. Sumithra, S. Buvana, R. Somasundaram. "A Survey on Various Types of Image Processing Technique" International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 4, March-2015
[16] P. Gupta, "A Survey Of Techniques And Applications For Real Time Image Processing." Journal of Global Research in Computer Science (UGC Approved Journal) 4, no. 8 (2013): 30-39.
[17] G. B. Souza, G. M. Alves, A. LM Levada, P. E. Cruvinel, and A. N. Marana. "A Graph-Based Approach for Contextual Image Segmentation" In Graphics, Patterns and Images (SIBGRAPI), 2016 29th SIBGRAPI Conference on,. IEEE 2016, pp. 281-288.