MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Project Report

on

EMOTION DETECTION USING FACIAL RECOGNITION

A project report submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted by:

Prathmesh Soni

(0901IO201048)

Shubham Sahu

(0901IO201060)

Faculty Mentor:

Prof. Abhishek Dixit

Assistant Professor , Information Technology

Submitted to:

DEPARTMENT OF INFORMATION TECHNOLOGY

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005

Jan-May 2023

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Prathmesh Soni** (0901IO201048) has submitted the project report titled Emotion Detection using facial recognition under the mentorship of Prof. Abhishek Dixit in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Information Technology** from Madhav Institute of Technology and Science, Gwalior.

Prof. Abhishek Dixit	**Dr. Akhilesh Tiwari**
Assistant Professor	Professor and Head,
Information Technology	Department of IT

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Shubham Sahu** (0901IO201060) has submitted the project report titled Emotion Detection using facial recognition under the mentorship of Prof. Abhishek Dixit in partial fulfilment of the requirement for the award of degree of Bachelor of Technology in **Information Technology** from Madhav Institute of Technology and Science, Gwalior.

Prof. Abhishek Dixit	**Dr. Akhilesh Tiwari**
Assistant Professor	Professor and Head,
Information Technology	Department of IT

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

I hereby declare that the work being presented in this project report, for the partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Information Technology at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Prof. Abhishek Dixit, Assistant Professor,** Information Technology.

I declare that I have not submitted the matter embodied in this report for the award of any degree or diploma anywhere else.

Date:
Place: Gwalior

Prathmesh Soni
0901IO201048
III Year
Information Technology,

Shubham Sahu
0901IO201060
III Year
Information Technology

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of Technology and Science** to allow me to continue my disciplinary/interdisciplinary project as a curriculum requirement, under the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean Academics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Department of Information Technology,** for allowing me to explore this project. I humbly thank **Dr. Akhilesh Tiwari**, Professor and Head, Department of Information Technology, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Prof. Abhishek Dixit** , Assistant Professor , Information Technology, for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

Prathmesh Soni
0901IO201048
III Year
Information Technology,

Shubham Sahu
0901IO201060
III Year
Information Technology

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Emotion detection is a challenging task that has many applications, such as human-computer interaction, virtual reality, and robotics. Detecting emotions from facial expressions is an essential task in the field of affective computing. With the advancements in deep learning and computer vision, emotion detection using face recognition has become a popular research topic. In this project, we aim to build a model that can detect emotions from face images using unsupervised learning and Convolutional Neural Network (CNN) method. The model is designed to learn the features of the face images that represent different emotions such as happiness, sadness, anger, fear, surprise, and neutral expressions. The dataset used for this project is preprocessed and augmented to increase the diversity of the dataset. The batch image generator is used to load the dataset in batches, reducing memory usage and increasing the training speed. The model is trained, validated, and tested to ensure that it performs well on unseen data. Fine-tuning is done to optimize the hyperparameters and improve the accuracy of the model. The best-performing model is used for emotion detection on unseen face images. By building this model, we can achieve accurate and reliable emotion detection using face recognition, which has many practical applications in the real world.

Keyword: CNN, Deep Learning, Unsupervised Learning .

सारः

भावना का पता लगाना एक चुनौतीपूर्ण कार्य है जिसमें कई अनुप्रयोग हैं, जैसे कि मानव-कंप्यूटर इंटरैक्शन, आभासी वास्तविकता और रोबोटिक्स। चेहरे के भावों से भावनाओं का पता लगाना भावात्मक कंप्यूटिंग के क्षेत्र में एक आवश्यक कार्य है। गहरी शिक्षा और कंप्यूटर दृष्टि में प्रगति के साथ, चेहरे की पहचान का उपयोग करके भावना का पता लगाना एक लोकप्रिय शोध विषय बन गया है। इस परियोजना में, हम एक मॉडल बनाने का लक्ष्य रखते हैं जो असुरक्षित सीखने और संक्रामक तंत्रिका नेटवर्क (सीएनएन) विधि का उपयोग करके चेहरे की छवियों से भावनाओं का पता लगा सकता है। मॉडल को चेहरे की छवियों की विशेषताओं को सीखने के लिए डिज़ाइन किया गया है जो खुशी, उदासी, क्रोध, भय, आश्चर्य और तटस्थ अभिव्यक्तियों जैसी विभिन्न भावनाओं का प्रतिनिधित्व करते हैं। इस परियोजना के लिए उपयोग किए जाने वाले डेटासेट को डेटासेट की विविधता बढ़ाने के लिए पूर्व-संसाधित और संवर्धित किया जाता है। बैच छवि जनरेटर का उपयोग बैचों में डेटासेट लोड करने, स्मृति उपयोग को कम करने और प्रशिक्षण की गति बढ़ाने के लिए किया जाता है। मॉडल को प्रशिक्षित, मान्य और परीक्षण किया जाता है ताकि यह सुनिश्चित हो सके कि यह अनदेखे डेटा पर अच्छा प्रदर्शन करता है। हाइपरपैरामीटर को अनुकूलित करने और मॉडल की सटीकता में सुधार करने के लिए फाइन-ट्यूनिंग की जाती है। सबसे अच्छा प्रदर्शन करने वाले मॉडल का उपयोग अनदेखी चेहरे की छवियों पर भावनाओं का पता लगाने के लिए किया जाता है। इस मॉडल का निर्माण करके, हम चेहरे की पहचान का उपयोग करके सटीक और विश्वसनीय भावना का पता लगाने को प्राप्त कर सकते हैं, जिसके वास्तविक दुनिया में कई व्यावहारिक अनुप्रयोग हैं।

कीवर्ड: सीएनएन, डीप लर्निंग, अनसुपरवाइज्ड लर्निंग।

CHAPTER-1: PROJECT OVERVIEW

1.1 Introduction :-

Facial emotion recognition has become an important issue in many application nowadays. In recent years, the research on facial emotion recognition has become extensive. The aim of facial emotion recognition is to help identify the state of human emotion (eg; neutral, happy, sad, surprise, fear, anger, disgust, contempt) based on particular facial images. The challenge on facial emotion recognition is to automatically recognize facial emotion state with high accuracy.

1.2 Objective & Scope of Project :-

The objective of this project is to develop a facial emotion recognition system using unsupervised learning and Convolutional Neural Networks (CNN) with the scope of achieving accurate emotion recognition on unseen facial images. The project can have practical applications in mental health monitoring, human-computer interaction, and entertainment industries.

1.3 Project Features :-

- Sharp Perception
- Achievable
- Unique
- Easy to understand

1.4 System Requirements:-

1.4.1 Hardware Specification :

Finding the hardware requirements is important because the hardware is a key component in the development of a web project.

HARDWARE TOOLS	MINIMUM REQUIREMENTS

Processor	I5 (10th gen) or above
RAM	8 GB or above
Monitor	15.6” coloured
GPU	4 GB or above
Keyboard	122 keys
Hard Disk	128 GB

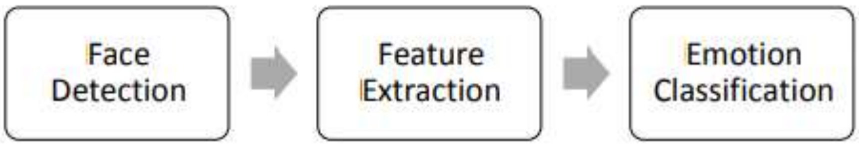
1.4.2 Software Specification :

SOFTWARE TOOLS	MINIMUM REQUIREMENTS
Platform	Windows
Operating System	Windows 10 or above
Technology	Deep Learning through Tensorflow
Scripting Language	Python (version 3.10)
IDE	Jupyter notebook

CHAPTER 2: LITERATURE REVIEW

The challenge on facial emotion recognition is to automatically recognize facial emotion state with high accuracy. It is challenging to find the similarity of the same emotion state between different person since they may express the same emotion state in various ways.

Generally FER is divided into three major stages as shown in Figure : (i) Face Detection, (ii) Feature Extraction, and (iii) Emotion Classification. At first stage, which is a pre-processing stage, an image of a face is detected and facial components of the face will be detected from the region. The facial components can be an eyes, brows, nose, and mouth. In the second stage, an informative features will be extracted from different parts of the face. In the last stage, a classifier need to be trained before been used to generate labels for the Emotions using the training data.



Facial Emotion Classification Stages.

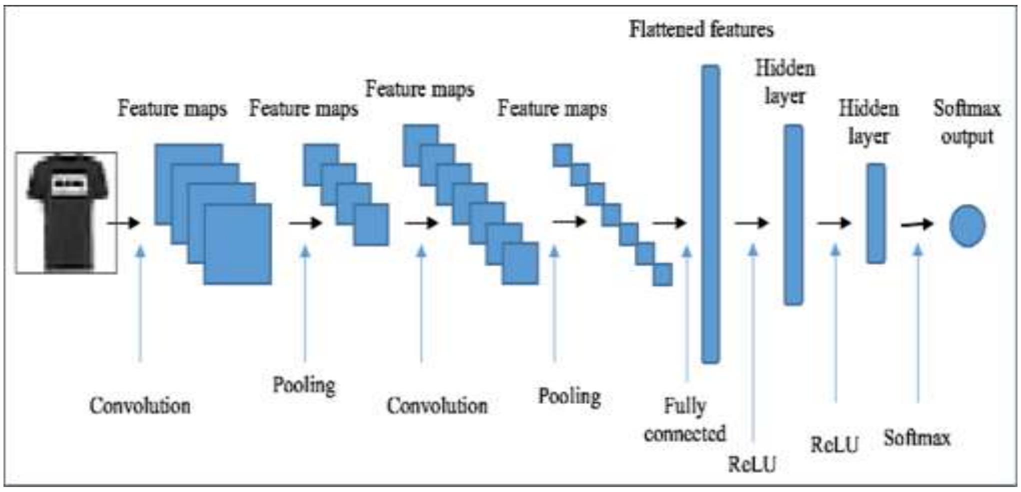
CHAPTER 3: PRELIMINARY DESIGN:-

3.1 Tools And Technologies Used:-

3.1.1 Technologies Used:-

3.1.1.1 Convolutional Neural Network :

A Convolutional Neural Network (CNN) is a type of neural network that is commonly used in computer vision tasks. CNNs are designed to learn features from images by using convolutional layers that apply a filter to the input image. In this project, we use a CNN model to detect emotions from face images.



3.1.1.2 Data Science:

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data. Using Data Science we prepare our data to feed it into a deep learning model.

3.1.1.3 Deep Learning:

Deep learning is a subset of machine learning that involves training deep neural networks to learn features and patterns from data. In this project, we use deep learning to build a Convolutional Neural Network (CNN) model to detect emotions from face images. The CNN model is designed to learn the features of the face images that represent different emotions. The model is trained on a large dataset of face images using backpropagation, and the weights of the model are optimized to minimize the loss function.

3.1.1.4 Machine Learning:

Machine learning is a subset of artificial intelligence (AI) that involves building algorithms that can automatically learn from and improve on data without being explicitly programmed. In other words, it is a method of teaching computers to learn from data and make decisions or predictions based on that data, without being explicitly programmed for every decision or prediction.

3.1.2 Tools :-

3.1.2.1 Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. This whole project is developed using python and its module. We used python as it is the most promising and easy implementing language for ML and Deep learning Algorithms.

3.1.2.2 Jupyter Notebook:

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. We have used this as our Integrated Development Tool (IDE) for writing python codes since it provides an interactive interface.

3.1.2.3 Anaconda Prompt:

Anaconda Prompt is a command line interface with Anaconda Distribution. Terminal is a command line interface that comes in Linux. We have used anaconda prompt as a terminal for running linux command to perform different operations like connecting to TensorBoard, installing different python modules, etc.

3.1.2.4 Firefox :

Mozilla Firefox, or simply Firefox, is a free and open-source web browser developed by the Mozilla Foundation. In our project, Firefox is used as a software for running jupyter notebook, and as an internet explorer to gather information regarding the project.

3.1.3 Modules:

3.1.3.1 NumPy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, matrices. We have used numpy for storing our image data in 3-d array format.

3.1.3.2 Seaborn:

Seaborn is a Python data visualization library that builds on top of matplotlib and provides a high-level interface for creating informative and visually appealing statistical graphics. It offers a range of pre-built visualization types, such as scatter plots, line plots, and heatmaps, and includes built-in statistical functions for exploring relationships between variables. Seaborn is a popular choice among data scientists and analysts for its ease of use, extensive documentation, and ability to create sophisticated visualizations with minimal code.

3.1.3.3 Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. We used matplotlib for visualizing graphs and images to analyse the best parameter fit for our model.

3.1.3.4 OpenCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. We used this module to for rescaling , img to rgb 3d matrix conversion, for getting live image from camera etc.

3.1.3.5 Tensorflow :

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. We have used tensorflow’s different sub-modules like keras Sequential for developing our neural network , tensorflow-hub for loading the feature vector for transfer learning, TensorBoard to visualize the performace of Deep learning model through Bar graphs, historgrams and other analysing tools.

3.1.3.6 Keras :

Keras is an open-source deep learning library that provides a user-friendly interface for building neural networks. In this project, we use Keras to build the CNN model. Keras provides a set of high-level APIs for building neural networks, which makes it easier to build and train the model. We can also use Keras to load and preprocess the dataset, create a batch image generator, and evaluate the performance of the model.

CHAPTER 4: FINAL ANALYSIS AND DESIGN :-

4.1 Model Implementation :-

4.1.1 Data Gathering :

For our project we imported a image data set from Kaggle. The data set we get is divided into two classes : Training data , Validation . Further it is divided into seven sub classes : i.e. Anger, Fear, Happy, Sad, Disgust, Surprise, Neutral. Our dataset contains approx. 32000 raw images which is pre classified in the categories shown above.

4.1.2 Data Preprocessing :

4.1.2.1 Resizing:

Resizing images into a standard dimention is very essential, since our deep learning model needs to create a generalized model to handle all the images, we reduced the size of image from random sizes to (224 X 224 X 3) using OpenCV module.

4.1.2.2 Scaling:

Through OpenCV module we get an image in the form of 3-D vector with RGB value ranges between (0-255), We then Scale it within the range (0-1). We scale all the image vector before feeding it to a Deep Learning model because it reduces the complexity and tends to perform well on when feeded with data that ranegs between 0 to 1.

4.1.3 Batch Image Generator:

A batch image generator is created to load the preprocessed and augmented dataset in batches. The batch image generator loads a specified number of images at a time, reducing the memory usage and increasing the training speed. This step is especially useful when dealing with large datasets.

4.1.4 Building CNN Model :

A Convolutional Neural Network (CNN) model is built to train the model on the collected dataset. The CNN model is designed to learn the features of the face images that represent different emotions. In this step, we need to select an appropriate architecture for the CNN model.

4.1.5 Training :

The model is trained on the preprocessed and augmented dataset using the batch image generator. During training, the model learns the weights that optimize the accuracy of the model. We need to select an appropriate optimizer and loss function for the model.

4.1.6 Testing :

After training, the model is tested on a test dataset to evaluate the performance of the model on unseen data. We need to evaluate the accuracy and precision.

4.1.7 Fine Tuning :

Fine-tuning is done to improve the accuracy of the model further. In this step, the hyperparameters such as the learning rate, batch size, and number of epochs are optimized to improve the performance of the model. We need to perform a grid search or random search to find the optimal hyperparameters.

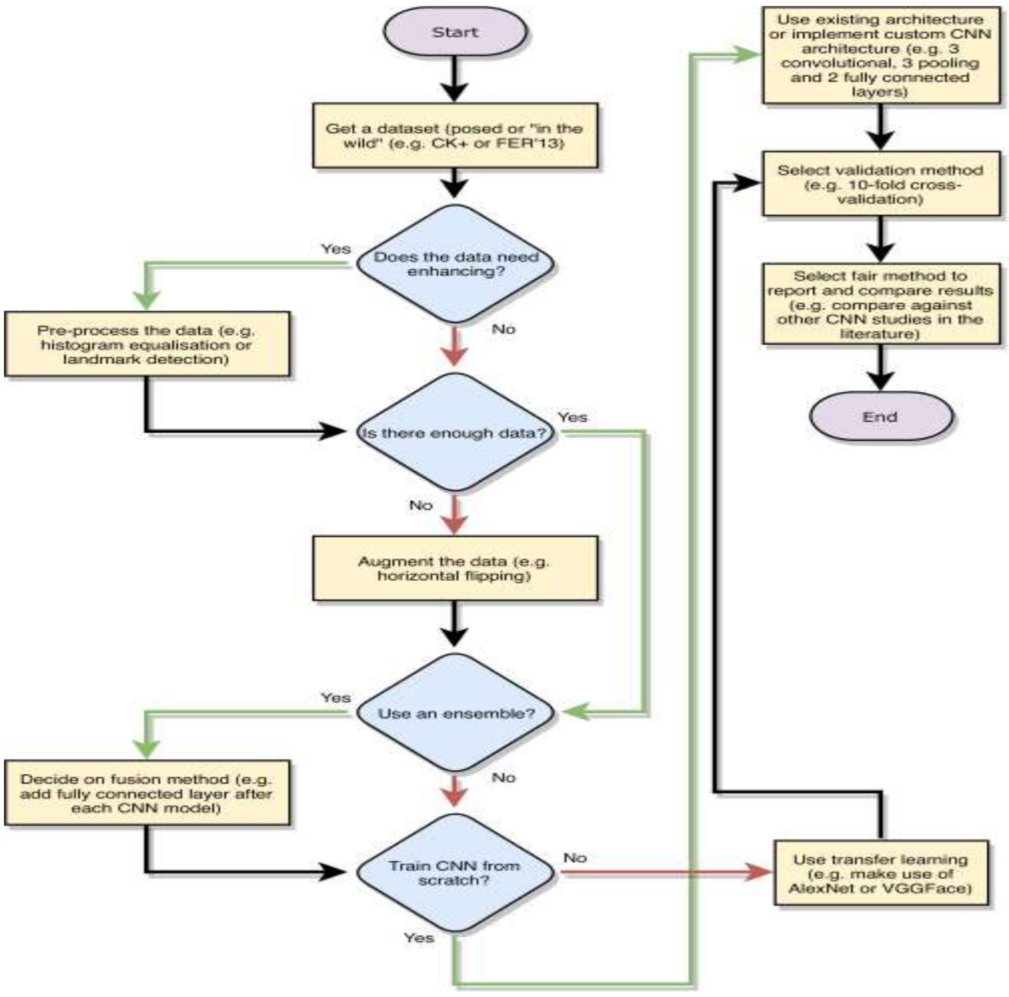
4.1.8 Emotion Detection :

Finally, the model is used to detect emotions on unseen face images. The face image is fed into the model, and the output of the model is the predicted emotion label. We need to use the best-performing model from the previous steps for emotion detection.

4.1.9 Interactive Interface :

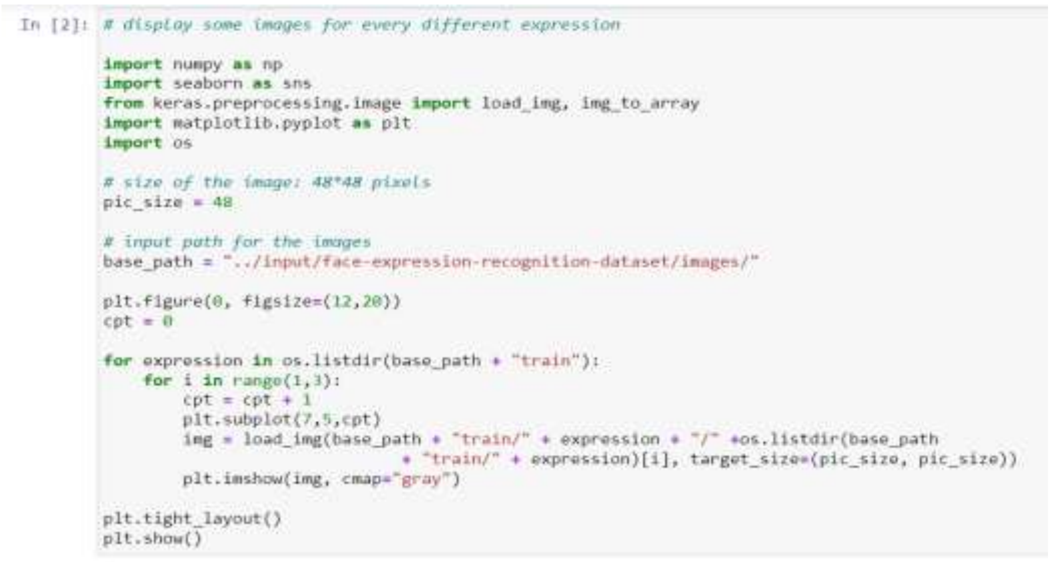
With the help of OpenCv, we designed an interactive live video capturing software that detects the emotion on the basis of image.

4.2Flowchart :-


Start
Get a dataset (posed or "in the wild" (e.g. CK+ or FER'13)
Does the data need enhancing?
Yes
No
Pre-process the data (e.g. histogram equalisation or landmark detection)
Is there enough data?
Yes
No
Augment the data (e.g. horizontal flipping)
Use an ensemble?
Yes
No
Decide on fusion method (e.g. add fully connected layer after each CNN model)
Train CNN from scratch?
No
Yes
Use transfer learning (e.g. make use of AlexNet or VGGFace)
Use existing architecture or implement custom CNN architecture (e.g. 3 convolutional, 3 pooling and 2 fully connected layers)
Select validation method (e.g. 10-fold cross-validation)
Select fair method to report and compare results (e.g. compare against other CNN studies in the literature)
End

4.3 Code Snippets:

4.3.1 Data Gathering :



```
In [2]: # display some images for every different expression

        import numpy as np
        import seaborn as sns
        from keras.preprocessing.image import load_img, img_to_array
        import matplotlib.pyplot as plt
        import os

        # size of the image: 48*48 pixels
        pic_size = 48

        # input path for the images
        base_path = "../input/face-expression-recognition-dataset/images/"

        plt.figure(0, figsize=(12,20))
        cpt = 0

        for expression in os.listdir(base_path + "train"):
            for i in range(1,3):
                cpt = cpt + 1
                plt.subplot(7,5,cpt)
                img = load_img(base_path + "train/" + expression + "/" +os.listdir(base_path
                                + "train/" + expression)[i], target_size=(pic_size, pic_size))
                plt.imshow(img, cmap="gray")

        plt.tight_layout()
        plt.show()
```

4.1.2 Data Preprocessing :

4.1.2.1 Scaling & Batch Image Generator:

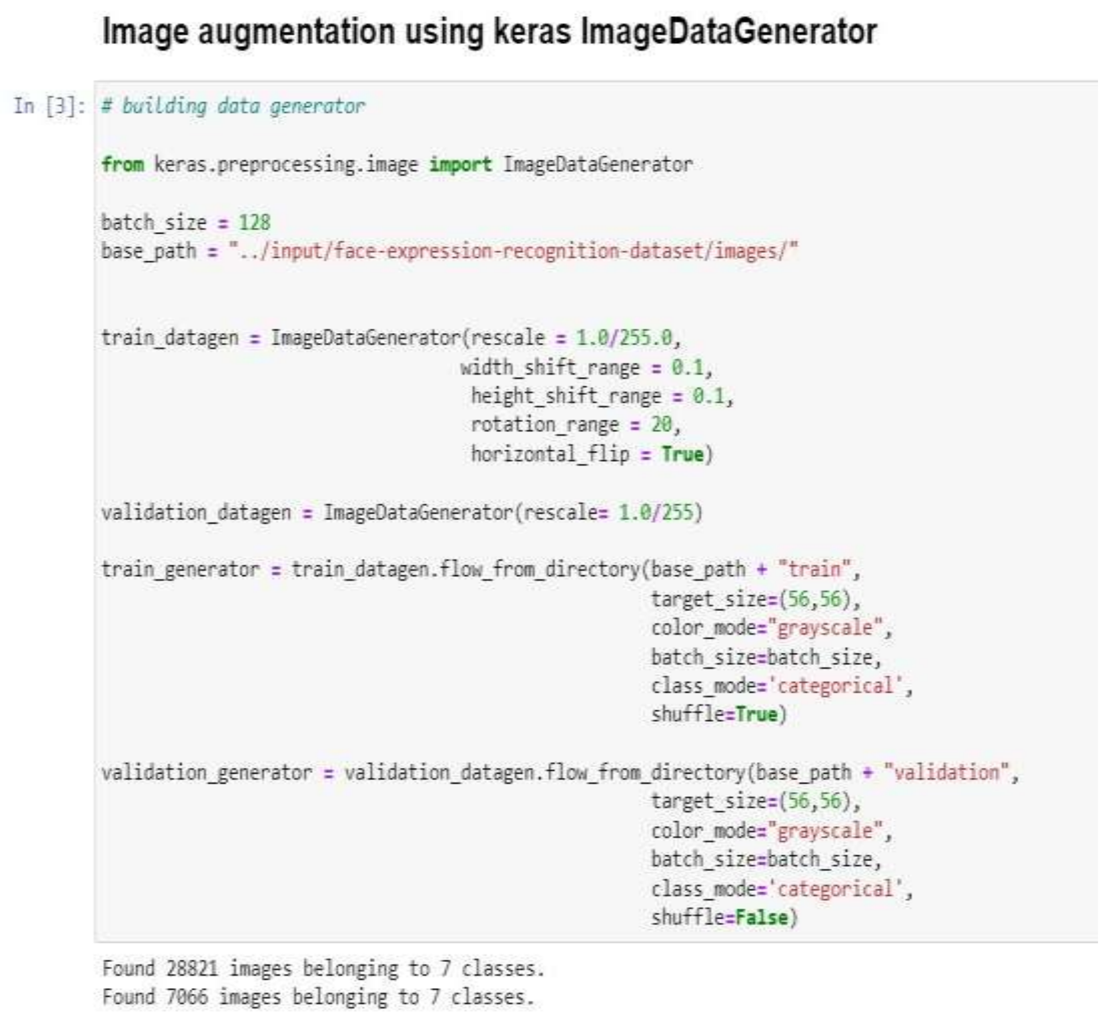
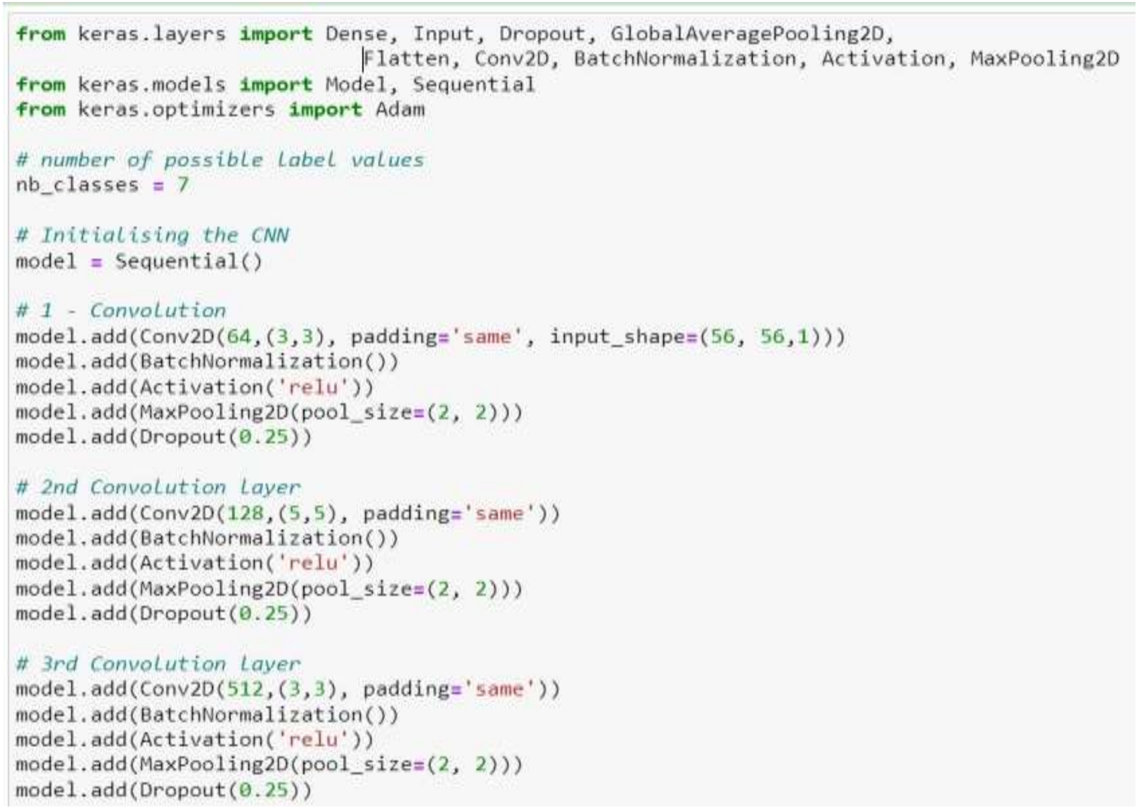


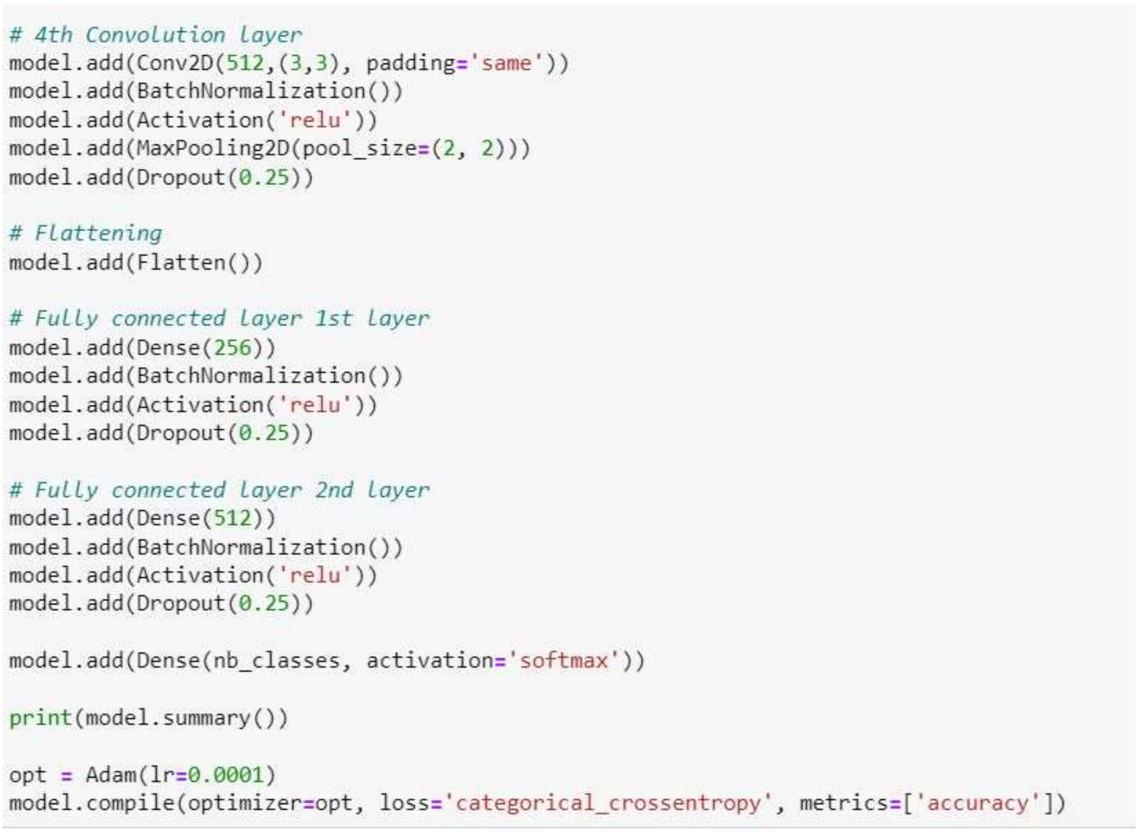
Image augmentation using keras ImageDataGenerator

```
In [3]: # building data generator

        from keras.preprocessing.image import ImageDataGenerator

        batch_size = 128
        base_path = "../input/face-expression-recognition-dataset/images/"


        train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                          width_shift_range = 0.1,
                                           height_shift_range = 0.1,
                                           rotation_range = 20,
                                           horizontal_flip = True)

        validation_datagen = ImageDataGenerator(rescale= 1.0/255)

        train_generator = train_datagen.flow_from_directory(base_path + "train",
                                                            target_size=(56,56),
                                                            color_mode="grayscale",
                                                            batch_size=batch_size,
                                                            class_mode='categorical',
                                                            shuffle=True)

        validation_generator = validation_datagen.flow_from_directory(base_path + "validation",
                                                            target_size=(56,56),
                                                            color_mode="grayscale",
                                                            batch_size=batch_size,
                                                            class_mode='categorical',
                                                            shuffle=False)

        Found 28821 images belonging to 7 classes.
        Found 7066 images belonging to 7 classes.
```

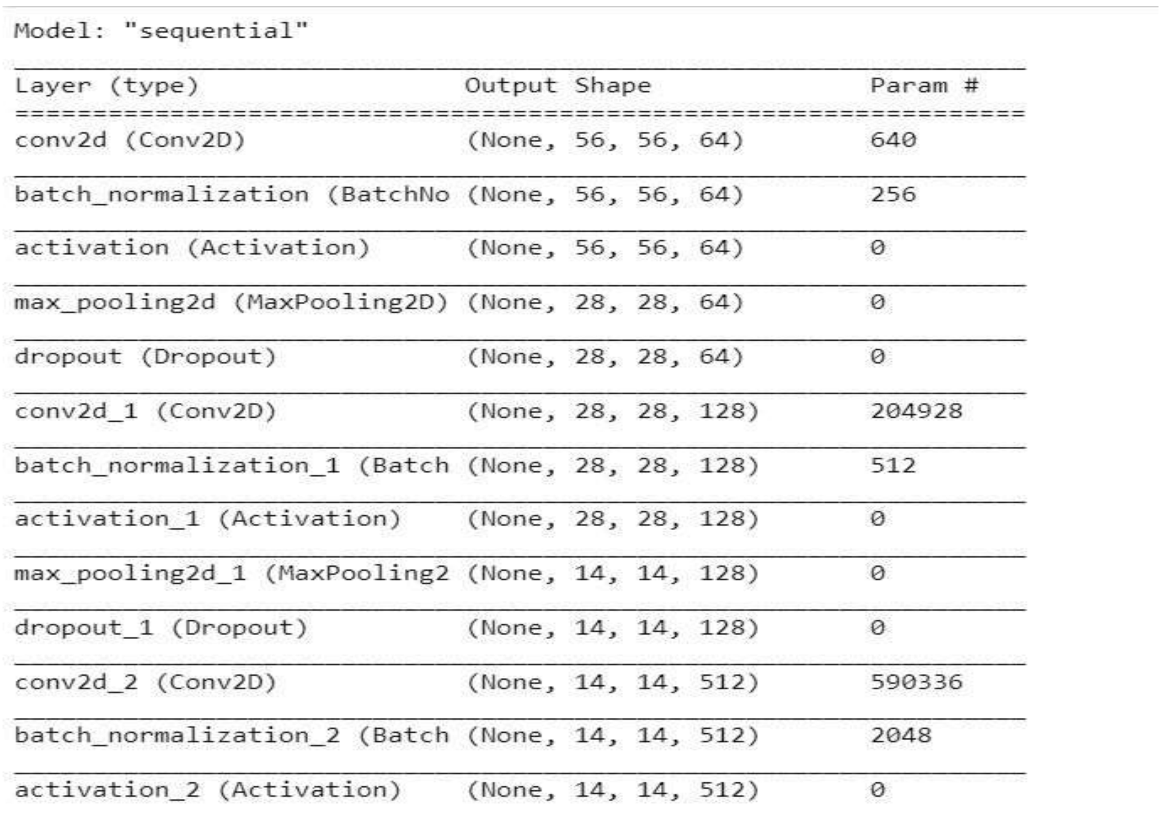
4.1.4 Building CNN Model :



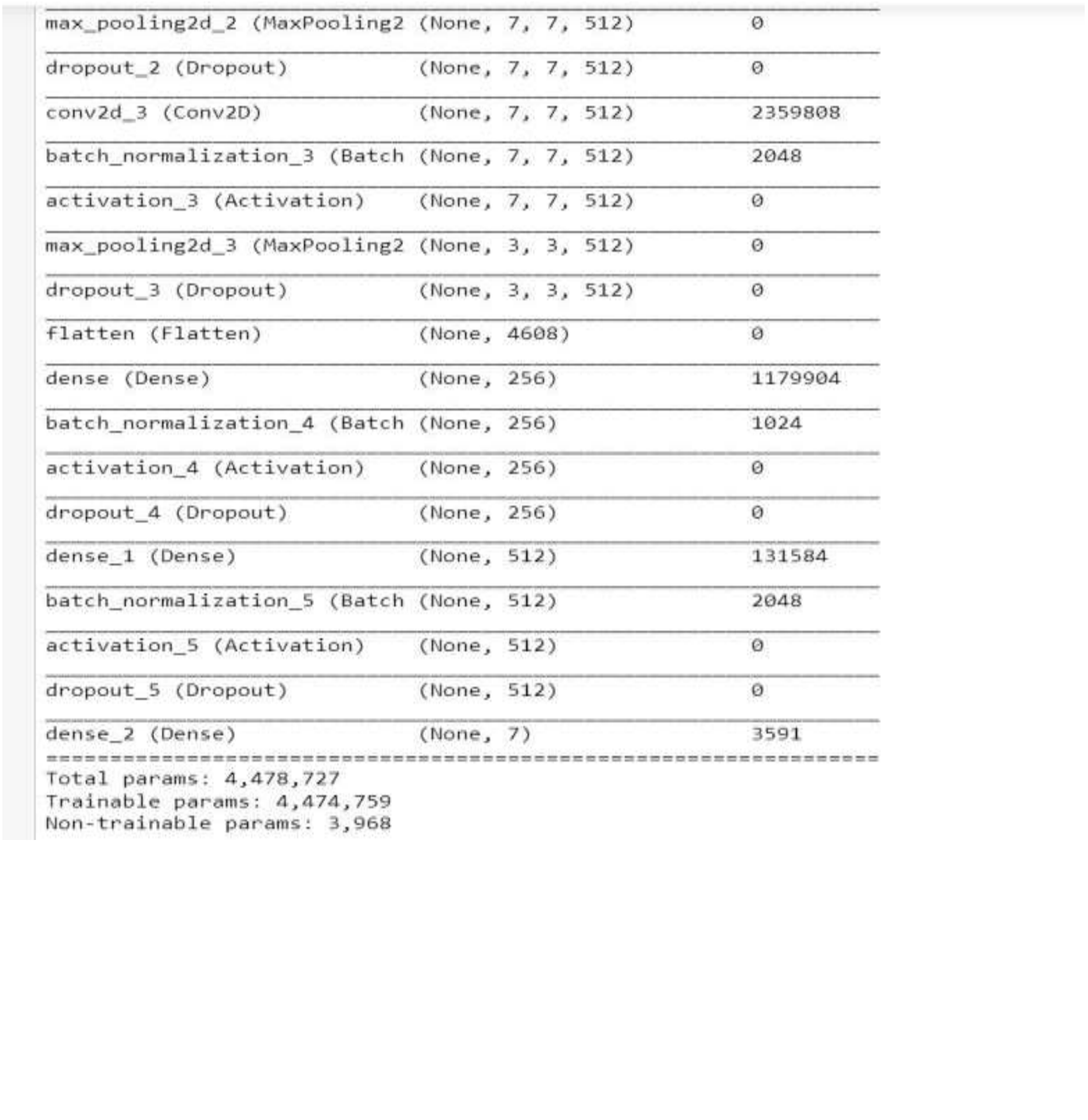
```python
from keras.layers import Dense, Input, Dropout, GlobalAveragePooling2D,
                              Flatten, Conv2D, BatchNormalization, Activation, MaxPooling2D
from keras.models import Model, Sequential
from keras.optimizers import Adam

# number of possible label values
nb_classes = 7

# Initialising the CNN
model = Sequential()

# 1 - Convolution
model.add(Conv2D(64,(3,3), padding='same', input_shape=(56, 56,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 2nd Convolution layer
model.add(Conv2D(128,(5,5), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 3rd Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```



```python
# 4th Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(nb_classes, activation='softmax'))

print(model.summary())

opt = Adam(lr=0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

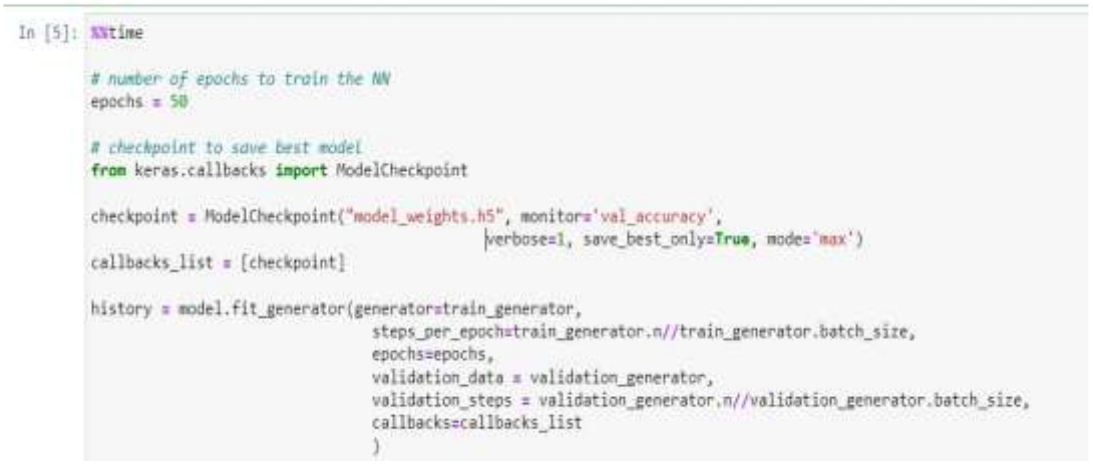


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 56, 56, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 56, 56, 64)        256
_________________________________________________________________
activation (Activation)      (None, 56, 56, 64)        0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 28, 28, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 128)       204928
_________________________________________________________________
batch_normalization_1 (Batch (None, 28, 28, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 28, 28, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 512)       590336
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 512)       2048
_________________________________________________________________
activation_2 (Activation)    (None, 14, 14, 512)       0
```

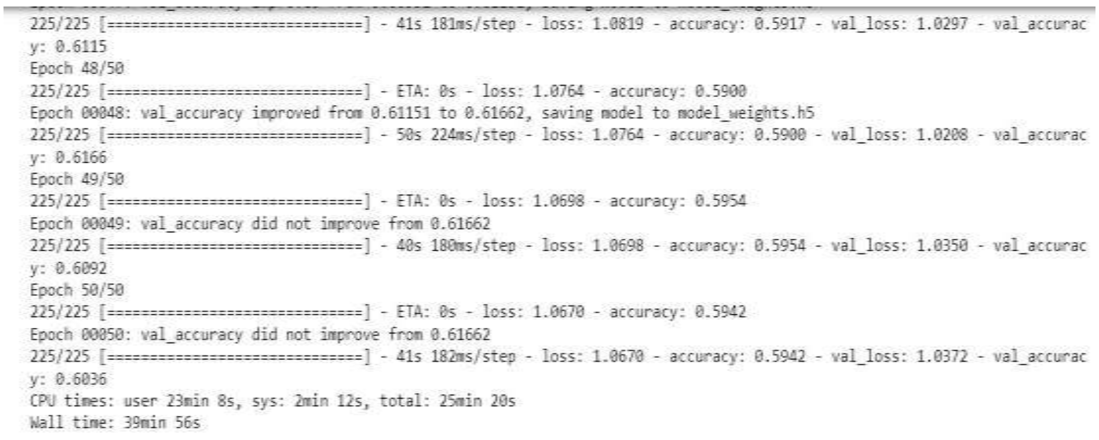


```
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 7, 7, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 7, 7, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
```

4.1.5 Training & Testing:

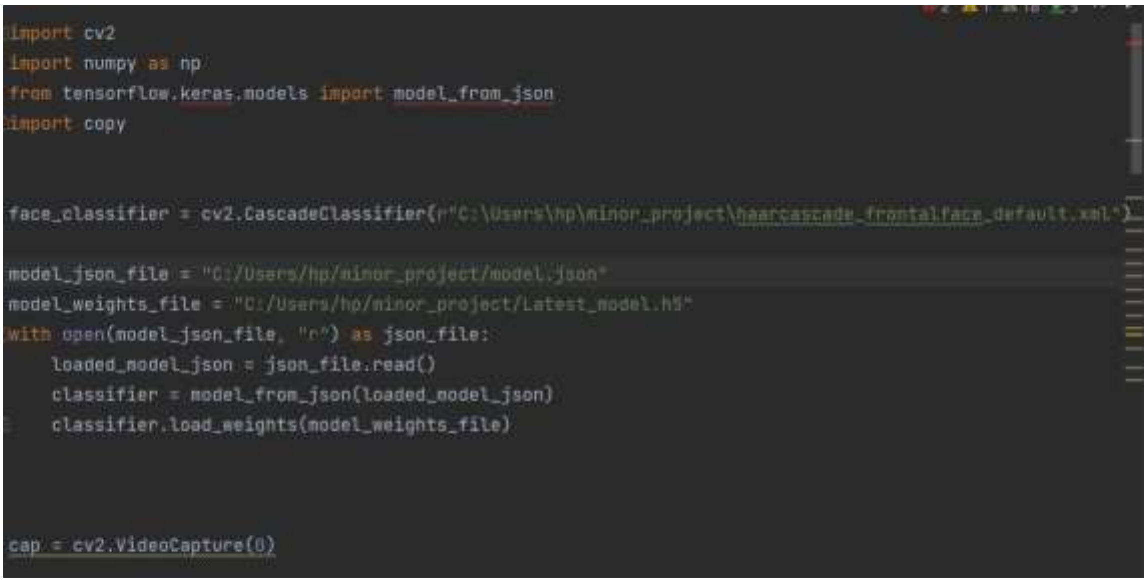


```
In [5]: %%time

        # number of epochs to train the NN
        epochs = 50

        # checkpoint to save best model
        from keras.callbacks import ModelCheckpoint

        checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_accuracy',
                                     verbose=1, save_best_only=True, mode='max')
        callbacks_list = [checkpoint]

        history = model.fit_generator(generator=train_generator,
                                      steps_per_epoch=train_generator.n//train_generator.batch_size,
                                      epochs=epochs,
                                      validation_data = validation_generator,
                                      validation_steps = validation_generator.n//validation_generator.batch_size,
                                      callbacks=callbacks_list
                                      )
```

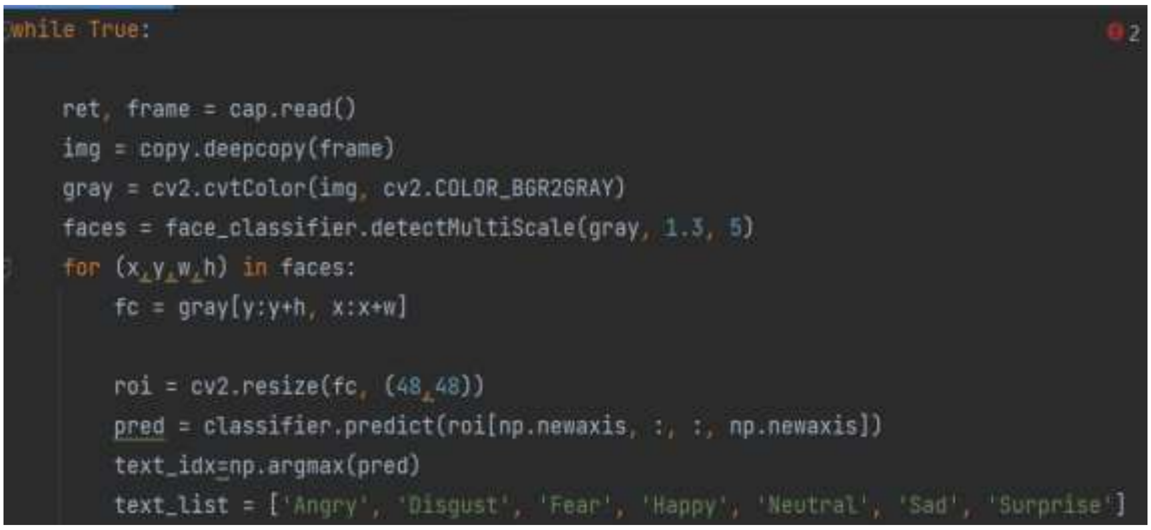


```
225/225 [==============================] - 41s 181ms/step - loss: 1.0819 - accuracy: 0.5917 - val_loss: 1.0297 - val_accurac
y: 0.6115
Epoch 48/50
225/225 [==============================] - ETA: 0s - loss: 1.0764 - accuracy: 0.5900
Epoch 00048: val_accuracy improved from 0.61151 to 0.61662, saving model to model_weights.h5
225/225 [==============================] - 50s 224ms/step - loss: 1.0764 - accuracy: 0.5900 - val_loss: 1.0208 - val_accurac
y: 0.6166
Epoch 49/50
225/225 [==============================] - ETA: 0s - loss: 1.0698 - accuracy: 0.5954
Epoch 00049: val_accuracy did not improve from 0.61662
225/225 [==============================] - 40s 180ms/step - loss: 1.0698 - accuracy: 0.5954 - val_loss: 1.0350 - val_accurac
y: 0.6092
Epoch 50/50
225/225 [==============================] - ETA: 0s - loss: 1.0670 - accuracy: 0.5942
Epoch 00050: val_accuracy did not improve from 0.61662
225/225 [==============================] - 41s 182ms/step - loss: 1.0670 - accuracy: 0.5942 - val_loss: 1.0372 - val_accurac
y: 0.6036
CPU times: user 23min 8s, sys: 2min 12s, total: 25min 20s
Wall time: 39min 56s
```

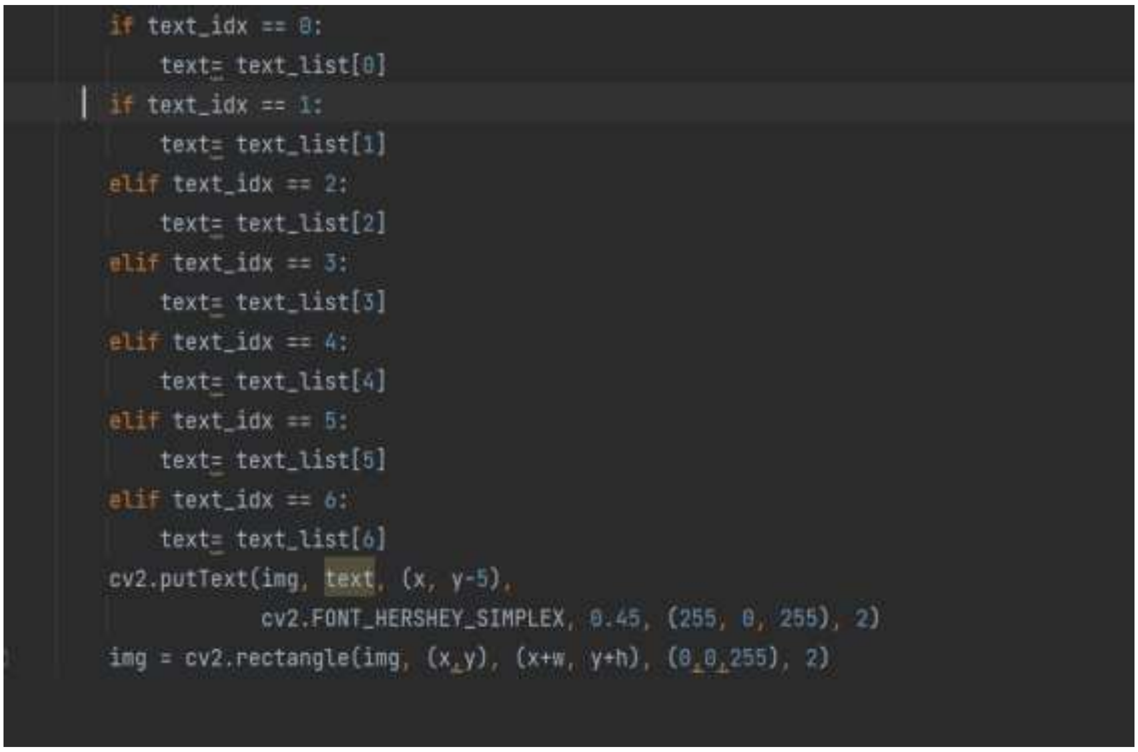
4.1.6 Interactive Interface :



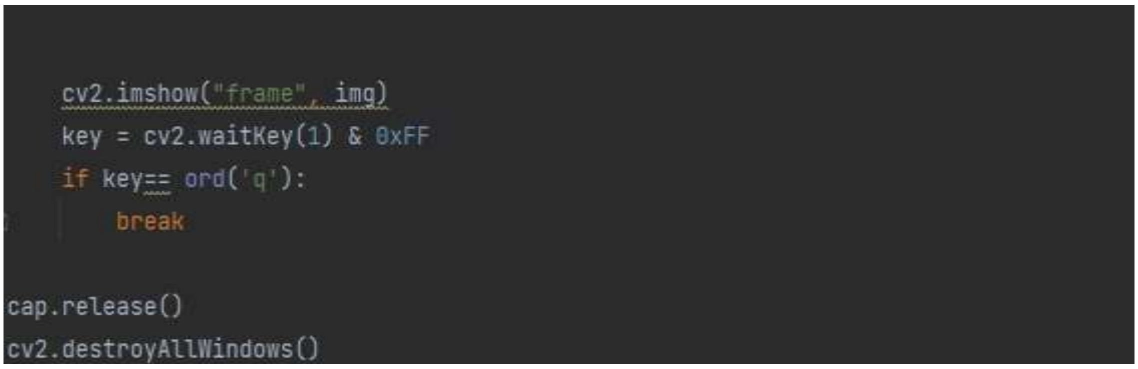
```
import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
import copy


face_classifier = cv2.CascadeClassifier(r"C:\Users\hp\minor_project\haarcascade_frontalface_default.xml")

model_json_file = "C:/Users/hp/minor_project/model.json"
model_weights_file = "C:/Users/hp/minor_project/Latest_model.h5"
with open(model_json_file, "r") as json_file:
    loaded_model_json = json_file.read()
    classifier = model_from_json(loaded_model_json)
    classifier.load_weights(model_weights_file)



cap = cv2.VideoCapture(0)
```



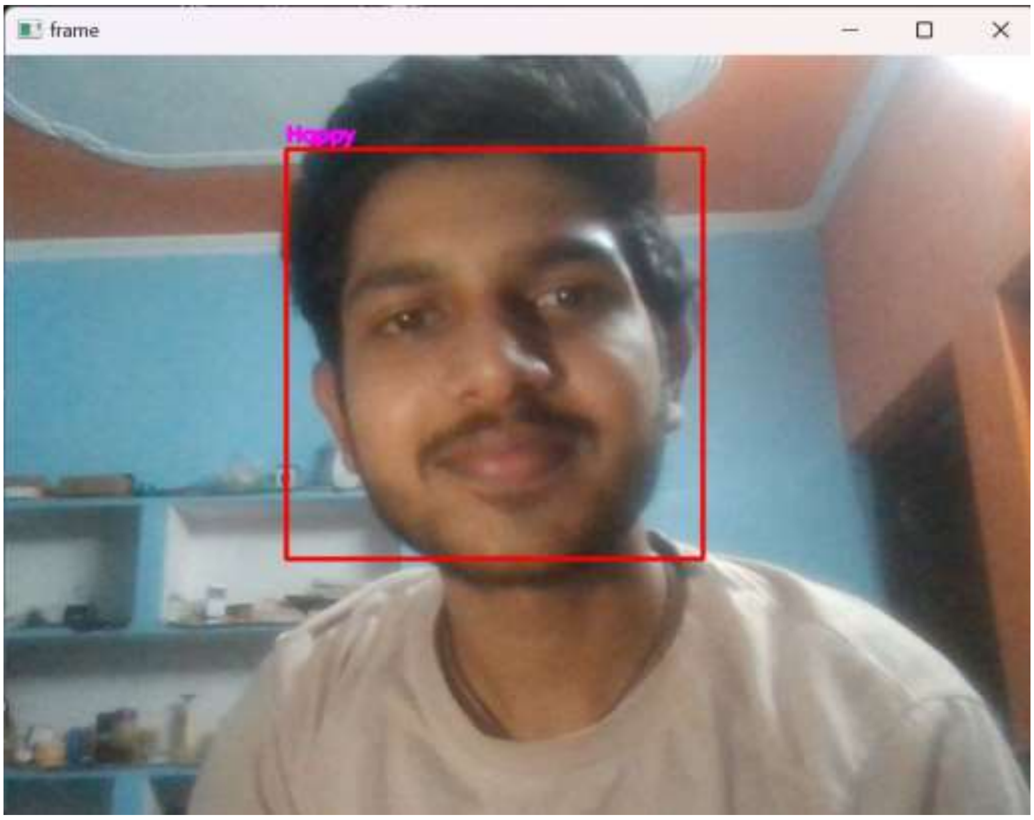
```python
while True:

    ret, frame = cap.read()
    img = copy.deepcopy(frame)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        fc = gray[y:y+h, x:x+w]

        roi = cv2.resize(fc, (48,48))
        pred = classifier.predict(roi[np.newaxis, :, :, np.newaxis])
        text_idx=np.argmax(pred)
        text_list = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
```



```python
        if text_idx == 0:
            text= text_list[0]
        if text_idx == 1:
            text= text_list[1]
        elif text_idx == 2:
            text= text_list[2]
        elif text_idx == 3:
            text= text_list[3]
        elif text_idx == 4:
            text= text_list[4]
        elif text_idx == 5:
            text= text_list[5]
        elif text_idx == 6:
            text= text_list[6]
        cv2.putText(img, text, (x, y-5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, (255, 0, 255), 2)
        img = cv2.rectangle(img, (x,y), (x+w, y+h), (0,0,255), 2)
```

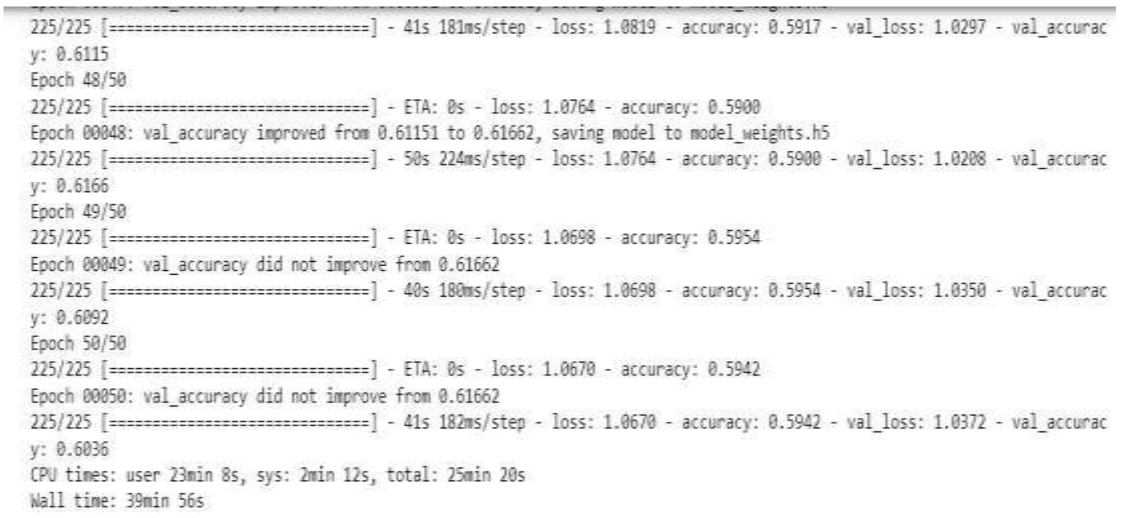


```python
    cv2.imshow("frame", img)
    key = cv2.waitKey(1) & 0xFF
    if key== ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```



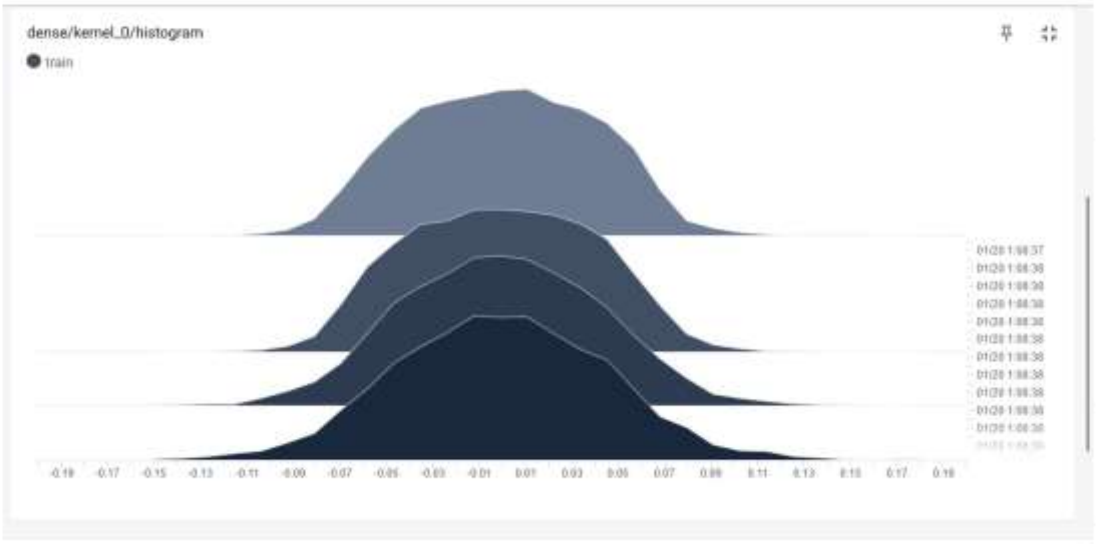
4.4 Result :-

After completion of this project, we have successfully developed a CNN Deep Learning model which can classify human emotions. In our model we have achieved an accuracy of 59.42% on training data and an accuracy of 60.36% on a random testing data. We have tested the model on all aspects and can asure that this model will outperform in maximum situations.



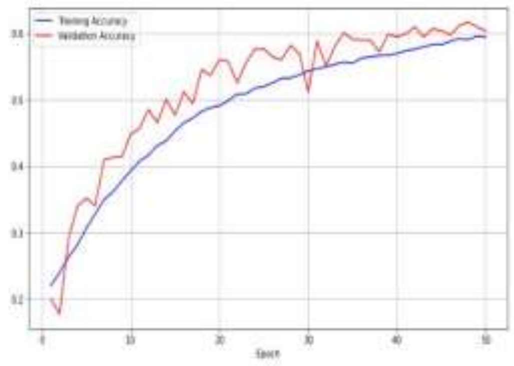
```
225/225 [==============================] - 41s 181ms/step - loss: 1.0819 - accuracy: 0.5917 - val_loss: 1.0297 - val_accurac
y: 0.6115
Epoch 48/50
225/225 [==============================] - ETA: 0s - loss: 1.0764 - accuracy: 0.5900
Epoch 00048: val_accuracy improved from 0.61151 to 0.61662, saving model to model_weights.h5
225/225 [==============================] - 50s 224ms/step - loss: 1.0764 - accuracy: 0.5900 - val_loss: 1.0208 - val_accurac
y: 0.6166
Epoch 49/50
225/225 [==============================] - ETA: 0s - loss: 1.0698 - accuracy: 0.5954
Epoch 00049: val_accuracy did not improve from 0.61662
225/225 [==============================] - 40s 180ms/step - loss: 1.0698 - accuracy: 0.5954 - val_loss: 1.0350 - val_accurac
y: 0.6092
Epoch 50/50
225/225 [==============================] - ETA: 0s - loss: 1.0670 - accuracy: 0.5942
Epoch 00050: val_accuracy did not improve from 0.61662
225/225 [==============================] - 41s 182ms/step - loss: 1.0670 - accuracy: 0.5942 - val_loss: 1.0372 - val_accurac
y: 0.6036
CPU times: user 23min 8s, sys: 2min 12s, total: 25min 20s
Wall time: 39min 56s
```

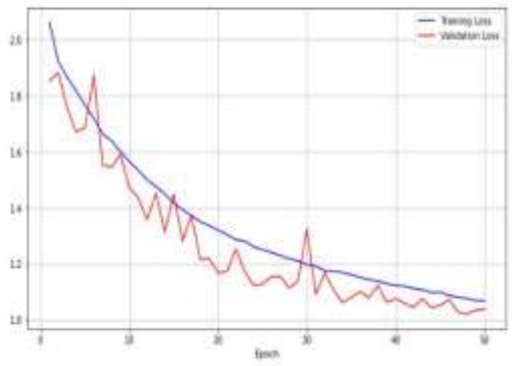
4.5 Result Analysis :-



```
def get_best_epcoh(history):
    valid_acc = history.history['val_accuracy']
    best_epoch = valid_acc.index(max(valid_acc)) + 1
    best_acc =  max(valid_acc)
    print('Best Validation Accuracy Score {:0.5f}, is for epoch {}'.format( best_acc, best_epoch))
    return best_epoch
```

```
plot_results(history)
best_epoch =get_best_epcoh(history)
```





```
Best Validation Accuracy Score 0.61662, is for epoch 48
```

4.6Application :-

4.6.1 Mental Health Monitoring

4.6.2 E - Learning

4.6.3 Entertainment

4.6.4 Human – Computer interaction

4.6.5 Monitoring

4.6.6 Gaming

CONCLUSION :-

In conclusion, the developed facial emotion recognition system utilizing unsupervised learning and CNN method showed promising results in accurately recognizing and classifying facial expressions of happiness, sadness, anger, surprise, fear, and neutral expressions. Preprocessing and augmenting the dataset played a significant role in improving the performance of the model. The potential practical applications of this project include mental health monitoring, human-computer interaction, and entertainment industries, indicating its importance in various aspects of daily life.

REFERENCES :-

1. lliana Azizan, Fatimah Khalid Facial Emotion Recognition: A Brief Review International Conference on Sustainable Engineering, Technology and Management (ICSETM -2018), Dec. 20, 2018, Negeri Sembilan, Malaysia.

2. Jonathan Oheix, Face Expression Recognition Data Set, KAGGLE.