A PROJECT REPORT ON

Sentiment Analysis – Face Emotion Detection

By

Gargi Agarwal (2000290140046)

Pinkush Gupta (2000290140084)

Soyab Khan (2000290140124)

Session:2021-2022 (4th Semester)

Under the supervision of

Ms. Vidushi
Assistant Professor

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS
KIET GROUP OF INSTITUTIONS, DELHI-NCR,
GHAZIABAD-201206
(JUNE- 2022)

CERTIFICATE

Certified that **Gargi Agarwal (2000290140046), Pinkush Gupta (2000290140084), Soyab Khan (2000290140124),** have carried out the project work having "Title of Report- Sentiment Analysis (Face Emotion Detection)" for Master of Computer Application from Dr A.P.J. Abdul Kalam University AKTU (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies and caried out by the student himself/ herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date: 01-06-2022

GARGI AGARWAL (2000290140046)

PINKUSH GUPTA (2000290140084)

SOYAB KHAN (2000290140124)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge

Date: 01-06-2022

Ms. Vidushi

Assistant Professor

Department of Computer Applications

KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava

Head, Department of Computer ApplicationsKIET Group of Institutions, Ghaziabad

TITLE: SENTIMENT ANALYSIS (FACE EMOTION DETECTION)

ABSTRACT

Sentiments are feelings, likes and dislikes which can be expressed through text, images or videos. Sentiment Analysis on web data is now becoming a growing research area of social data analysis. End users express their sentiments through various web platforms such as Twitter, E-commerce websites, etc. by exchanging texts and uploading images. Users express their emotions through videos as well by online meeting platforms like Microsoft teams, Google Meet, etc. A lot of research work has been done for data collected in textual form; there has been limited work that focuses on analyzing the sentiment of data collected through images. There is a need to work upon image analysis. One of the major challenges is to predict the sentiments of the unlabeled images. Deep Learning models have the capability for effectively learning the image behavior or polarity. Image recognition, image prediction, image sentiment analysis, and image classification are some of the fields where Neural Network (NN) has performed well implying significant performance of deep learning in image sentiment analysis. This paper focuses on the noteworthy model of deep learning- Convolutional Neural Network (CNN) along with the suitability of their applications in image sentiment analysis and their limitations with perspectives of this rising field.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Vidushi** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would have been possible in time. They keep my life filled with enjoyment and happiness.

GARGI AGARWAL

(2000290140046)

PINKUSH GUPTA

(2000290140084)

SOYAB KHAN

(2000290140124)

TABLE OF CONTENTS

List of Figures

CHAPTER 1. INTRODUCTION

1.1 Introduction of the project

Sentiments are expressed using text, image or videos. A abundance of research papers are available for text sentiment analysis, but image sentiment analysis is not much explored. With the increasing use of web to express sentiments, it become one of the important area of research, therefore since last few years plenty of research has been done for the same to achieve optimum results. So we are deep diving into image analysis for sentiment analysis through face motion through deep learning. Deep learning is the subfield of machine learning that makes the computer enough intelligent, so that the machine is capable to learn from experience and perceive the world of concepts. Computer fetch knowledge from the experience of real world, without human help to make computer understand all the situations or to make decisions. CNN (Convolutional Neural Network) each input image will pass it through a series of convolution layers with filters. In order to perceive the same as humans, CNNs have digital color images that have red-blue-green (RGB) encoding. There is a Convolutional Layer, Activation Layer, Pooling Layer, and Fully Connected Layer, these are all interconnected so that CNNs can process and perceive data in order to classify images.

1.2 Problem Statement

- There are many researches and works has been done with text data there is need to work with image data.
- As we can connect through

1.3 Objective

To implement Convolutional Neural Networks for classification of facial expressions and to use this product in medical and educational field.

CHAPTER 2. LITERATURE REVIEW

2.1 Planning

In planning phase study of reliable and effective algorithms is done. On the other hand data were collected and were preprocessed for more fine and accurate results. Since huge amount of data were needed for better accuracy we have collected the data surfing the internet. Since, we are new to this project we have decided to use local binary pattern algorithm for feature extraction and support vector machine for training the dataset. We have decided to implement these algorithms by using OpenCv framework.

2.2 Literature Survey

Though many researchers has explored number of techniques for image sentiment analysis, machine learning based techniques are performing significantly well in this field. CNN(Convolutional Neural Network) is one such technique.CNN has four main four components- Convolution, Rectified Linear Unit, Pooling (sub-sampling) and Classification (Multilayer Perceptron). The primary purpose of Convolution in CNN is to extract features from the input image entered. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. The size of the Feature Map (Convolved Feature) is controlled by three parameters- Depth- Depth corresponds to the number of filters we use for the convolution operation. Stride- Stride is the size of the filter, if the size of the filter is 5x5 then stride is 5. Zero-padding- Sometimes, it is convenient to pad the input matrix with zeros around the border, so that filter can be applied to bordering elements of input image matrix. Using zero padding size of the feature map can be

controlled. An additional operation called ReLU has been used after every Convolution operation.

A Rectified Linear Unit (ReLU) is a cell of a neural network which uses the following activation function to calculate its output given x:

$$R(x) = Max(0,x)$$

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. The Fully Connected layer is a traditional Multi-Layer Perceptron that uses a softmax activation function in the output layer. The term “Fully Connected” implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.
Application areas of Face Emotion Detection are Schizophrenia, Mood Disorder,etc. Schizophrenia is a severe mental health state affecting the way a patient thinks, feels, and behaves. Mood Disorder- the diagnosis of mood disorder could be divided into two categories: Bipolar Disorder (BD) and Unipolar Depression (UD).

CHAPTER 3. SOFTWARE REQUIREMENTS ANALYSIS

3.1 Functional Requirements

The functional requirements for a system describe what the system should do. Those requirements depend on the type of software being developed, the expected users of the software. These are statement of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situation.

3.2 Non-Functional Requirements

Nonfunctional requirements are requirements that are not directly concerned with the specified function delivered by the system. They may relate to emergent system properties such as reliability, response time and store occupancy. Some of the nonfunctional requirements related with this system are hereby below:

a) Reliability:

Reliability based on this system defines the evaluation result of the system, correct identification of the facial expressions and maximum evaluation rate of the facial expression recognition of any input images.

b) Easy to use:

The system is simple, user friendly, graphics user interface implemented so any can use this system without any difficulties.

CHAPTER 4. SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Software Requirements

Following are the software requirement necessary of the project:

a) Python Language Editor (jupyter notebook)

b) OpenCV Framework

4.2 Hardware Requirements

Following are the software requirement necessary of the project:

a) Fluently working laptops

b) RAM 4Gb

c) Web Camera

CHAPTER 5: FEASIBILITY STUDY

Before starting the project, feasibility study is carried out to measure the viable of the system. Feasibility study is necessary to determine if creating a new or improved system is friendly with the cost, benefits, operation, technology and time. Following feasibility study is given as below:

5.1 Technical Feasibility

Technical feasibility is one of the first studies that must be conducted after the project has been identified. Technical feasibility study includes the hardware and software devices. The required technologies (Python language and Anaconda(Jupyter Notebook)) existed

5.2 Operational Feasibility

Operational Feasibility is a measure of how well a proposed system solves the problem and takes advantage of the opportunities identified during scope definition. The following points were considered for the project's technical feasibility:

- The system will detect and capture the image of face
- The captured image is then (identified which category)

5.3 Economic Feasibility

The purpose of economic feasibility is to determine the positive economic benefits that include quantification and identification. The system is economically feasible due to availability of all requirements such as collection of data from

- Kaggle

5.4 Schedule Feasibility

Schedule feasibility is a measure of how reasonable the project timetable is. The system is found schedule feasible because the system is designed in such a way that it will finish prescribed time.

CHAPTER 6: PROJECT METHODOLOGY

6.1 System Design

System design shows the overall design of system. In this section we discuss in detail the design aspects of the system:

6.1.1 System Diagram

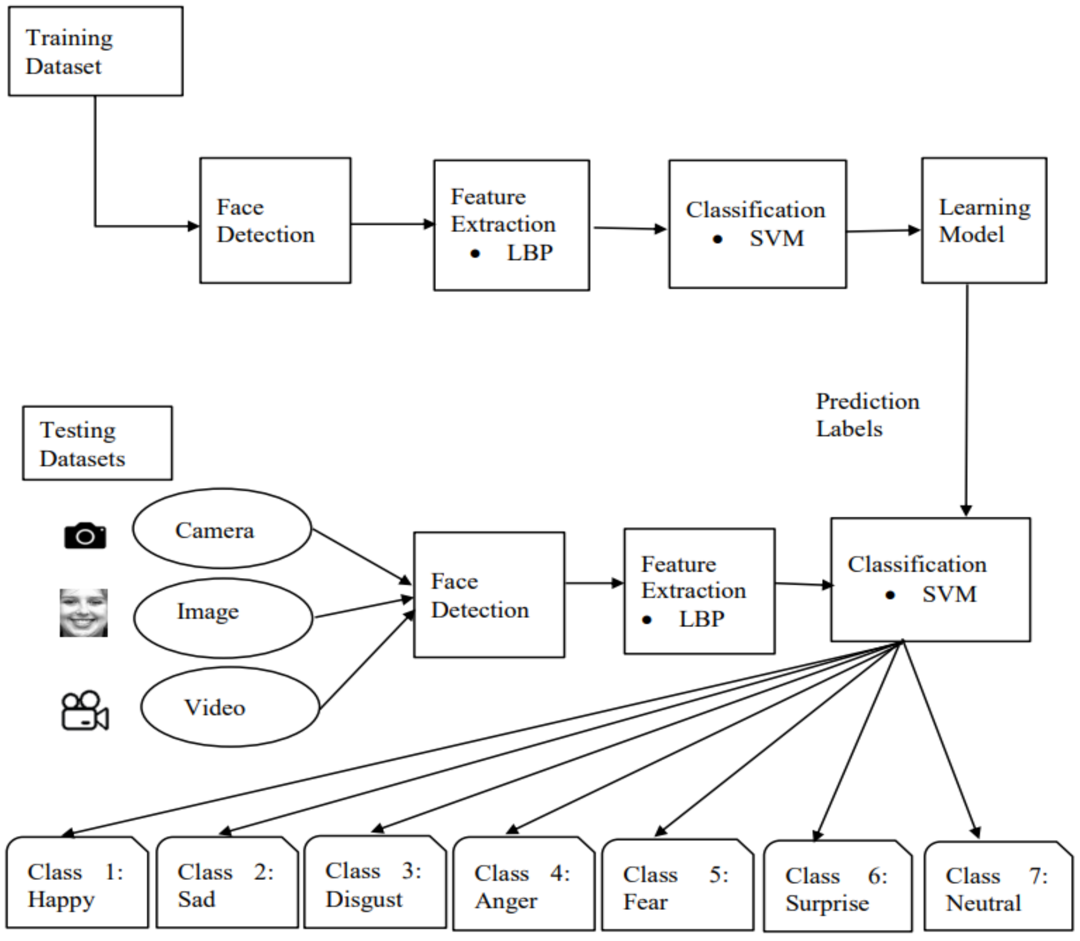


Fig: 6.1.1 System Diagram

6.1.2 System Flowchart

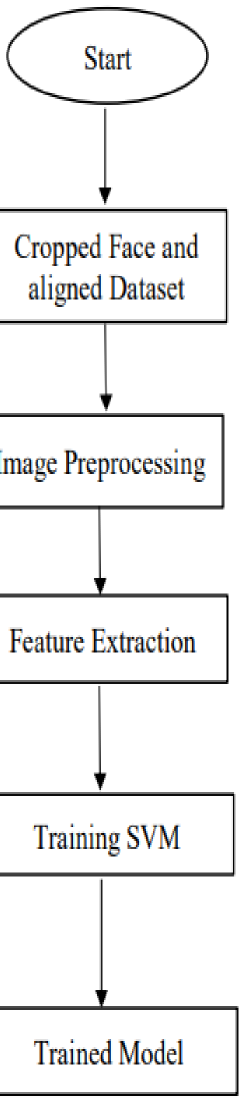


Fig: 6.1.2 Flowchart

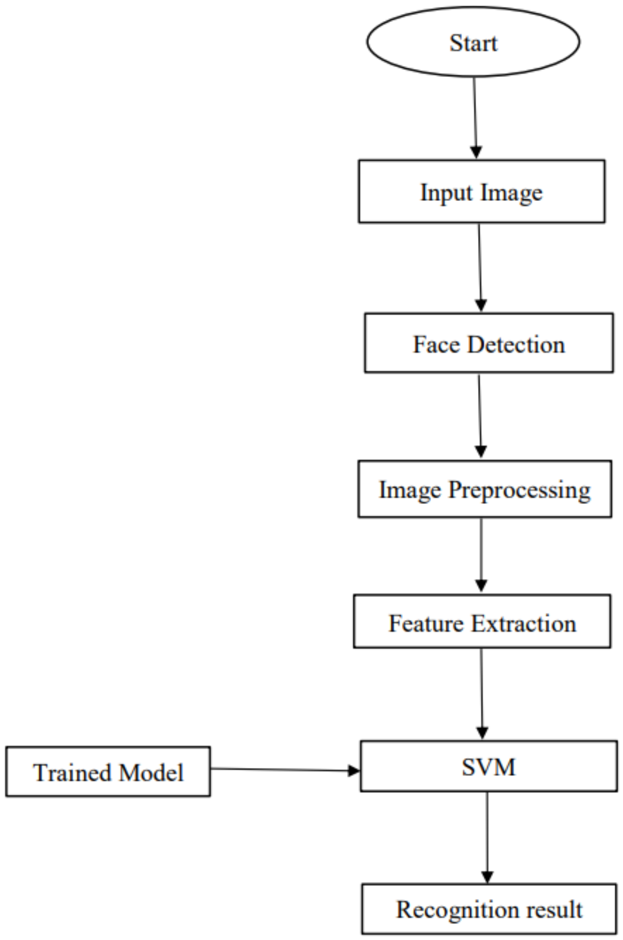
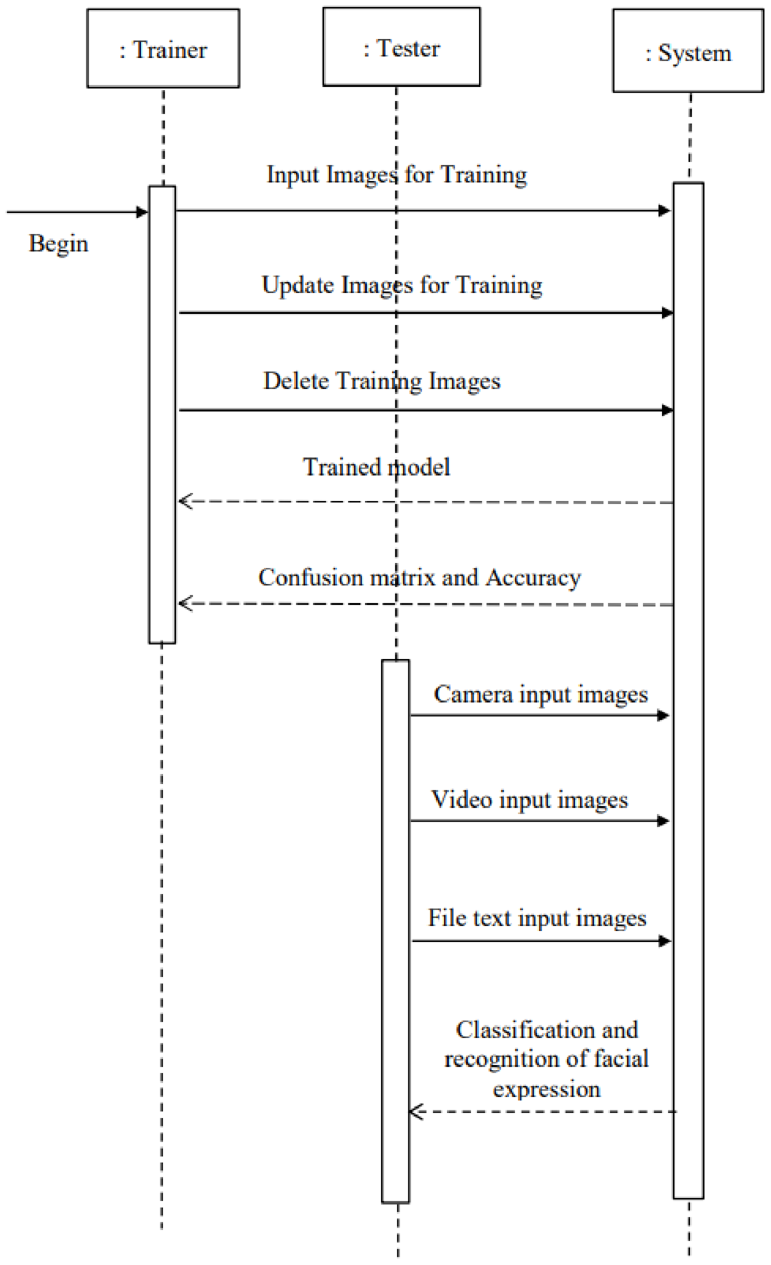


Fig 6.1.2.2: Flowchart of Testing/Prediction

6.1.3 Sequence Diagram:



: Trainer
: Tester
: System
Begin
Input Images for Training
Update Images for Training
Delete Training Images
Trained model
Confusion matrix and Accuracy
Camera input images
Video input images
File text input images
Classification and recognition of facial expression

Fig: 6.1.3 Sequence Diagram

6.2 Dataset

The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) is used for the training and testing. It comprises pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral. Dataset has training set of 35887 facial images with facial expression labels. The dataset has class imbalance issue, since some classes have large number of examples while some has few. The dataset is balanced using oversampling, by increasing numbers in minority classes. The balanced dataset contains 40263 images, from which 29263 images are used for training, 6000 images are used for testing, and 5000 images are used for validation.

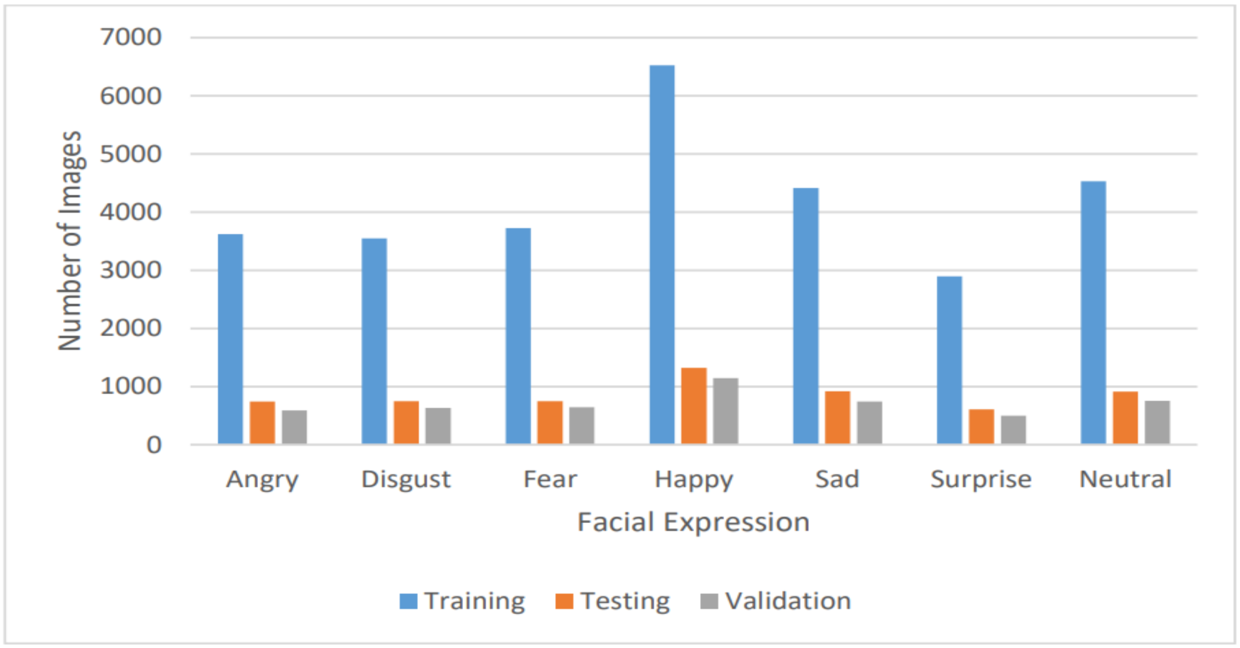


Fig: 6.2 Training, Testing, Validation

6.3 Architecture of CNN

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer. CNN is designed with some modification on LeNet Architecture [10]. It has 6 layers without considering input and output. The architecture of the Convolution Neural Network used in the project is shown in the following figure.

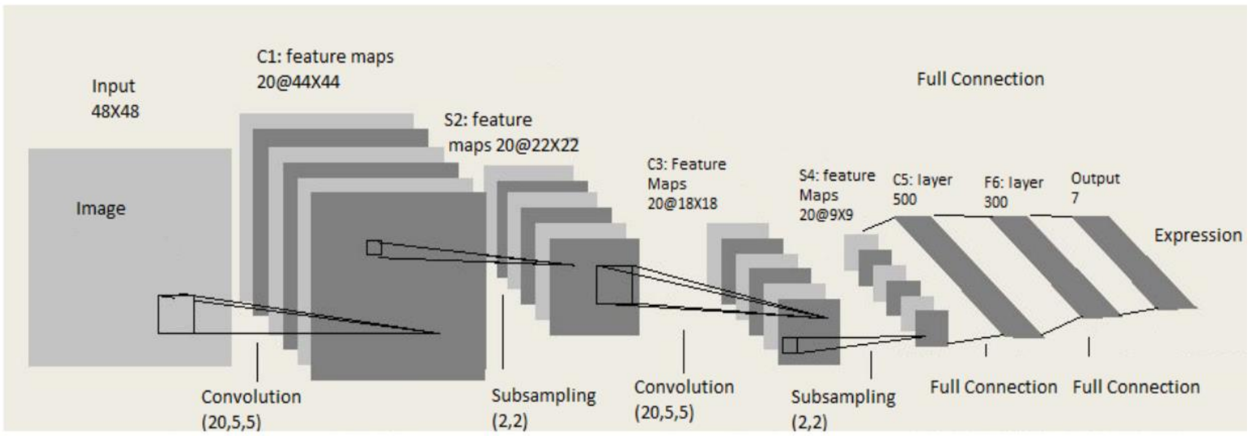


Fig 6.3: Architecture of CNN

a) Input Layer:

The input layer has pre-determined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. Normalized gray scale images of size 48 X 48 pixels from Kaggle dataset are used for training, validation and testing. For testing propose laptop webcam images are also used, in which face is detected and cropped using OpenCV Haar Cascade Classifier and normalized.

b) Convolution and Pooling (ConvPool) Layers:

Convolution and pooling is done based on batch processing. Each batch has N images and CNN filter weights are updated on those batches. Each convolution layer

takes image batch input of four dimension N x Color-Channel x width x height. Feature map or filter for convolution are also four dimensional (Number of feature maps in, number of feature maps out, filter width, filter height). In each convolution layer, four dimensional convolution is calculated between image batch and feature maps. After convolution only parameter that change is image width and height.

New image width = old image width – filter width + 1

New image height = old image height – filter height + 1

After each convolution layer downsampling / subsampling is done for dimensionality reduction. This process is called Pooling. Max pooling and Average Pooling are two famous pooling method. In this project max pooling is done after convolution. Pool size of (2x2) is taken, which splits the image into grid of blocks each of size 2x2 and takes maximum of 4 pixels. After pooling only height and width are affected.

Two convolution layer and pooling layer are used in the architecture. At first convolution layer size of input image batch is Nx1x48x48. Here, size of image batch is N, number of color channel is 1 and both image height and width are 48 pixel. Convolution with feature map of 1x20x5x5 results image batch is of size Nx20x44x44. After convolution pooling is done with pool size of 2x2, which results image batch of size Nx20x22x22. This is followed by second convolution layer with feature map of 20x20x5x5, which results image batch of size Nx20x18x18. This is followed by pooling layer with pool size 2x2, which results image batch of size Nx20x9x9.

c) Fully Connected Layer:

This layer is inspired by the way neurons transmit signals through the brain. It takes a large number of input features and transform features through layers connected with trainable weights. Two hidden layers of size 500 and 300 unit are used in fully-connected layer. The weights of these layers are trained by forward propagation of training data then backward propagation of its errors. Back propagation starts from evaluating the difference between prediction and true value, and back calculates the weight adjustment needed to every layer before. We can control the training speed and the complexity of the architecture by tuning the hyper-parameters, such as

learning rate and network density. Hyper-parameters for this layer include learning rate, momentum, regularization parameter, and decay.

The output from the second pooling layer is of size Nx20x9x9 and input of first hidden layer of fully-connected layer is of size Nx500. So, output of pooling layer is flattened to Nx1620 size and fed to first hidden layer. Output from first hidden layer is fed to second hidden layer. Second hidden layer is of size Nx300 and its output is fed to output layer of size equal to number of facial expression classes.

d) Output Layer:

Output from the second hidden layer is connected to output layer having seven distinct classes. Using Softmax activation function, output is obtained using the probabilities for each of the seven class. The class with the highest probability is the predicted class.

6.4 Phases in Facial Expression Recognition

The facial expression recognition system is trained using supervised learning approach in which it takes images of different facial expressions. The system includes the training and testing phase followed by image acquisition, face detection, image preprocessing, feature extraction and classification. Face detection and feature extraction are carried out from face images and then classified into six classes belonging to six basic expressions which are outlined below:

6.4.1 Image Acquisition

Images used for facial expression recognition are static images or image sequences. Images of face can be captured using camera

6.4.2 Face Detection

Face Detection is useful in detection of facial image. Face Detection is carried out in training dataset using Haar classifier called Voila-Jones face detector and implemented through Opencv. Haar like features encodes the difference in average intensity in different parts of the image and consists of black and white connected rectangles in which the value of the feature is the difference of sum of pixel values in black and white regions [2].

6.4.3 Image Pre-Processing

Image pre-processing includes the removal of noise and normalization against the variation of pixel position or brightness.

a) Color Normalization

b) Histogram Normalization

6.4.4 Feature Extraction

Selection of the feature vector is the most important part in a pattern classification problem. The image of face after pre-processing is then used for extracting the important features. The inherent problems related to image classification include the scale, pose, translation and variations in illumination level [6].

6.4.5 Classification

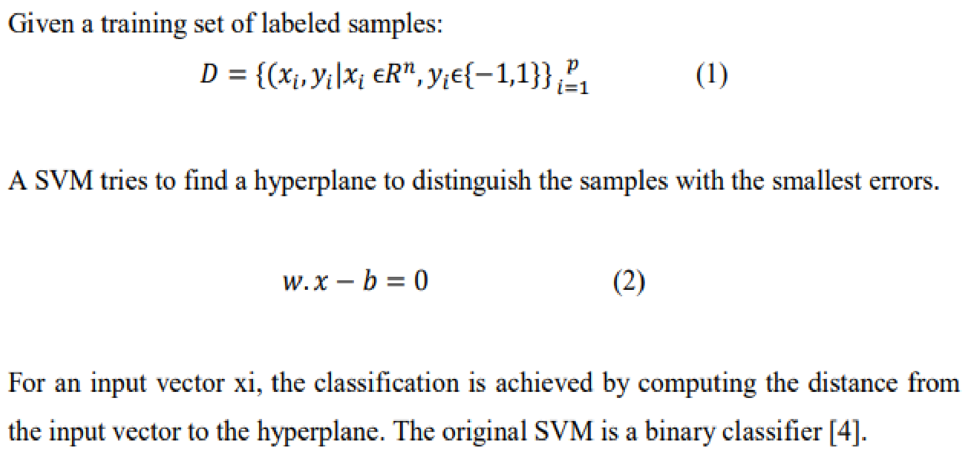
The dimensionality of data obtained from the feature extraction method is very high so it is reduced using classification. Features should take different values for object belonging to different class so classification will be done using Support Vector Machine algorithm.

6.4.5.2 Support Vector Machines

SVM is widely used in various pattern recognition tasks. SVM is a state-of-the-art machine learning approach based on the modern statistical learning theory. SVM can achieve a near optimum separation among classes. SVMs is trained to perform facial expression classification using the features proposed. In general, SVM are the maximal hyperplane classification method that relies on results from statistical learning theory to guarantee high generalization performance.

Kernel functions are employed to efficiently map input data which may not be linearly separable to a high dimensional feature space where linear methods can then be applied. SVMs exhibit good classification accuracy even when only a modest amount of training data is available, making them particularly suitable to a dynamic, interactive approach to expression recognition [10].

An ideal separation is achieved when the hyper plane and the training data of any class is the largest. This separating hyper plane works as the decision surface. SVM has been successfully employed for a number of classification tasks such as text categorization, genetic analysis and face detection [11].



Given a training set of labeled samples:

$$D = \{(x_i, y_i | x_i \epsilon R^n, y_i \epsilon \{-1,1\}\}_{i=1}^{p} \qquad (1)$$

A SVM tries to find a hyperplane to distinguish the samples with the smallest errors.

$$w.x - b = 0 \qquad (2)$$

For an input vector xi, the classification is achieved by computing the distance from the input vector to the hyperplane. The original SVM is a binary classifier [4].

CHAPTER 7: DEVELOPMENT AND TESTING

7.1 Implementation Tools

7.1.1 Programming Language and Coding Tools

a) Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. This versatility, along with its beginner-friendliness, has made it one of the most-used programming languages today. A survey conducted by industry analyst firm RedMonk found that it was the second-most popular programming language among developers in 2021.

What can you do with python? Some things include:

- Data analysis and machine learning
- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks

1) **Data Analysis and Machine Learning:**

Python has become a staple in data science, allowing data analysts and other professionals to use the language to conduct complex statistical calculations, create data visualizations, build machine learning algorithms, manipulate and analyze data, and complete other data-related tasks.

Python can build a wide range of different data visualizations, like line and bar graphs, pie charts, histograms, and 3D plots. Python also has a number of libraries that enable coders to write programs for data analysis and machine learning more quickly and efficiently, like TensorFlow and Keras.

2) Web Development:

Python is often used to develop the back end of a website or application—the parts that a user doesn't see. Python's role in web development can include sending data to and from servers, processing data and communicating with databases, URL routing, and ensuring security. Python offers several frameworks for web development. Commonly used ones include Django and Flask.

Some web development jobs that use Python include back end engineers, full stack engineers, Python developers, software engineers, and DevOps engineers.

3) Automation or Scripting:

If you find yourself performing a task over and over again, you could work more efficiently by automating it with Python. Writing code used to build these automated processes is called scripting. In the coding world, automation can be used to check for errors across multiple files, convert files, execute simple math, and remove duplicates in data.

Python can even be used by relative beginners to automate simple tasks on the computer—such as renaming files, finding and downloading online content or sending emails or texts at desired intervals.

4) Software Testing and Prototyping:

In software development, Python can aid in tasks like build control, bug tracking, and testing. With Python, software developers can automate testing for new products or features. Some Python tools used for software testing include Green and Requestium.

5) Everyday Tasks:

Python isn't only for programmers and data scientists. Learning Python can open new possibilities for those in less data-heavy professions, like journalists, small business owners, or social media marketers. Python can also enable non-programmer to simplify certain tasks in their lives. Here are just a few of the tasks you could automate with Python:

- Keep track of stock market or crypto prices
- Send yourself a text reminder to carry an umbrella anytime it's raining
- Update your grocery shopping list
- Renaming large batches of files
- Converting text files to spreadsheets
- Randomly assign chores to family members
- Fill out online forms automatically

b) Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running With Jupyter Notebook:

The Jupyter Notebook is not included with Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This article will focus on just two of them for the purposes of installing Jupyter Notebook. The most popular is CPython, which is the reference version of Python that you can get from their website. It is also assumed that you are using **Python 3**.

7.1.2 Framework

OpenCV:

OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000. **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.

OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development.

OpenCV-Python is the Python API for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

OpenCV-Python:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by **Guido van Rossum** that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

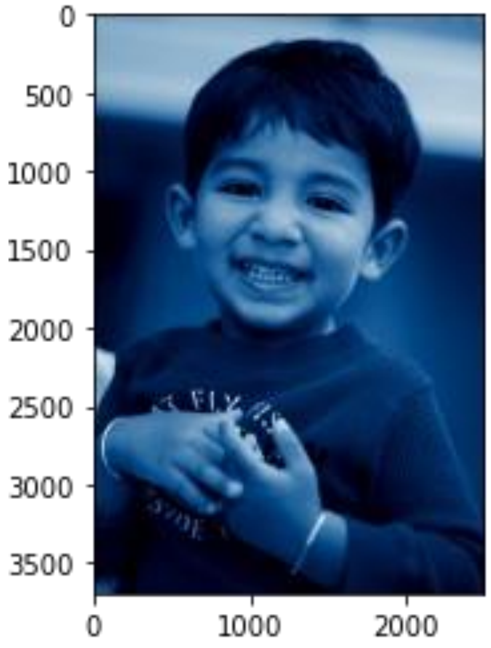
Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive

code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

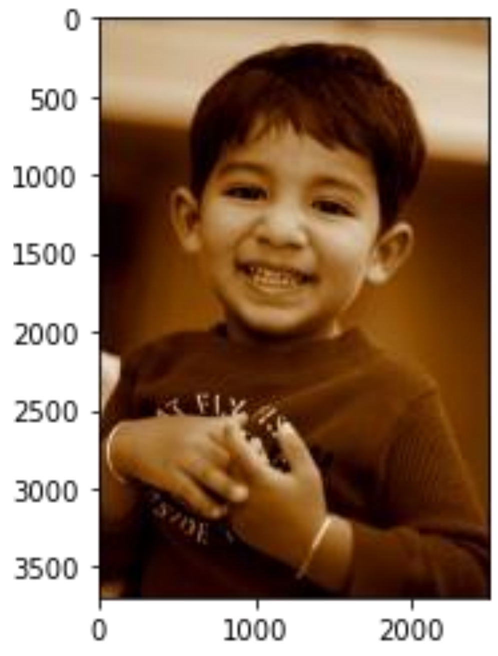
OpenCV-Python makes use of **Numpy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

7.2 Code

```
import cv2

from deepface import DeepFace

img = cv2.imread('happy-kid-2.jpg')import

matplotlib.pyplot as plt plt.imshow(img)

<matplotlib.image.AxesImage at 0x2792c7dcd90>
```



```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x27930425610>
```



```
predictions = DeepFace.analyze(img)
```

```
Action: race: 100%|
████████████████████████████████████████████████████████████
| 4/4 [00:15<00:00,  3.85s/it]
```

```
predictions
```

```
{'emotion': {'angry': 8.200909396016698e-09,'disgust':
   1.7151570148952328e-15,
   'fear': 0.00048818073992151767,
   'happy': 98.50794076919556,
   'sad': 0.029996942612342536,
   'surprise': 0.00017897715451908888,
   'neutral': 1.461394876241684},
 'dominant_emotion': 'happy',
 'region': {'x': 484, 'y': 575, 'w': 1425, 'h': 1425},
 'age': 32, 'gender':
 'Man',
 'race': {'asian': 28.015601275491562,
   'indian': 14.550685638107758,
   'black': 7.859540538842614,
   'white': 13.814799330854497,
   'middle eastern': 11.995498151123744,
   'latino hispanic': 23.76387432052203},
 'dominant_race': 'asian'}
```

```
type(predictions)
```

```
dict
```

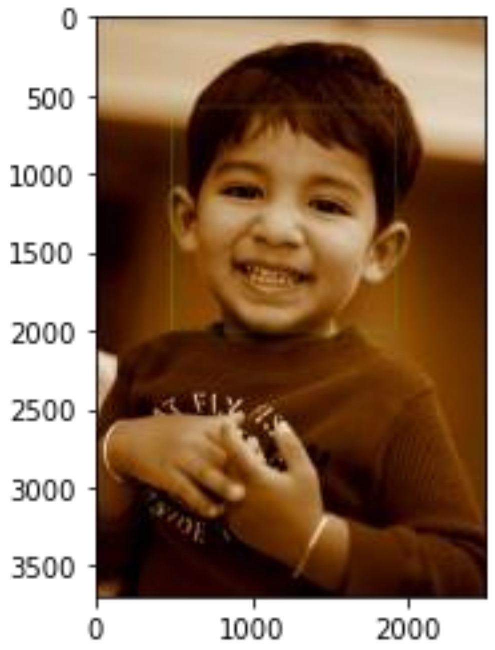
```
predictions['dominant_emotion']
```

```
'happy'
```

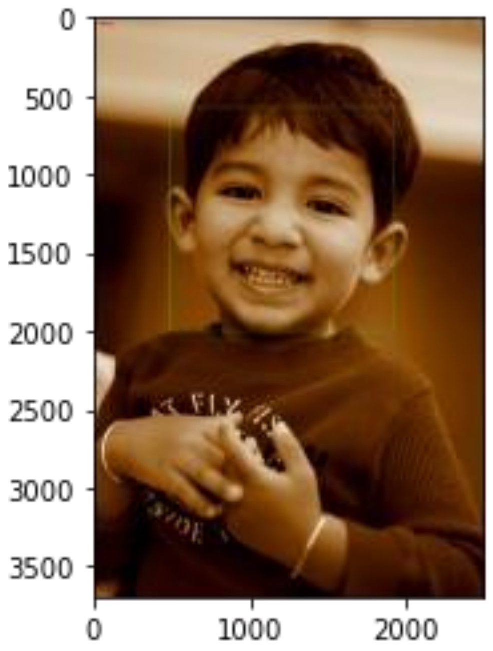
we are tyring to draw a rectangle across the face

```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
     #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

     # Draw a rectangle around the faces

for(x, y, w, h) in faces:

     cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x2793c25c610>
```



```
 font = cv2.FONT_HERSHEY_SIMPLEX
# use putText() method for#
inserting text on video
 cv2.putText(img,predictions['dominant_emotion'],(0, 50),
                              font, 1,
                              (0, 0, 255),
2, cv2.LINE_4) ;
 plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
 <matplotlib.image.AxesImage at 0x2793c34d640>
```



```
img = cv2.imread('OIP.jpg')
plt.imshow(cv2.cvtColor(img,
cv2.COLOR_BGR2RGB))
<matplotlib.image.AxesImage at 0x2792aae0e80>
```

```
predictions = DeepFace.analyze(img)
```

```
Action: race: 100%|
[redacted]
| 4/4 [00:06<00:00,   1.67s/it]
```

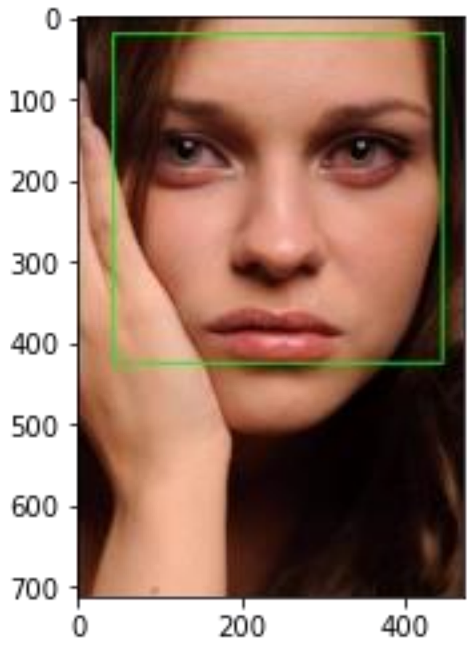
```
predictions
```

```
{'emotion': {'angry': 1.1243654540749048e-05,'disgust':
   1.3810860831247944e-16,
   'fear': 33.20895433425903, 'happy':
   1.069427098959741e-09, 'sad':
   66.78990125656128, 'surprise':
   3.654791736856485e-13,'neutral':
   0.0011304736290185247},
  'dominant_emotion': 'sad',
  'region': {'x': 44, 'y': 21, 'w': 404, 'h': 404},
  'age': 30, 'gender':
  'Woman',
  'race': {'asian': 12.49112337721865,
   'indian': 12.390831099847686,
   'black': 2.9823267662318664,
   'white': 23.851796847796063,
   'middle eastern': 19.23025926712266,
    'latino hispanic': 29.053664690692983},
  'dominant_race': 'latino hispanic'}
```

```
type(predictions)
```

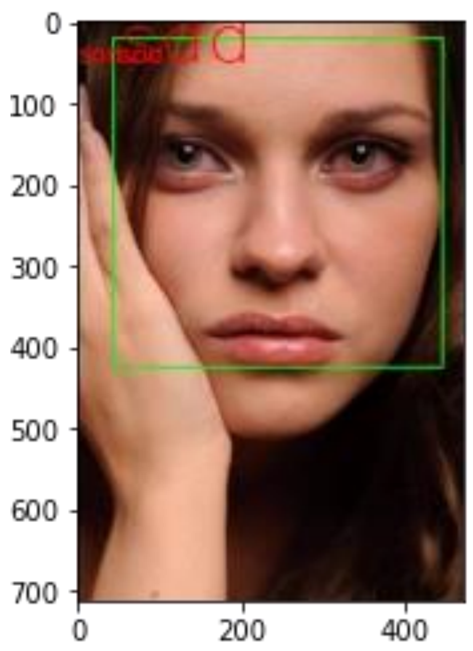
```
dict
```

```
predictions['dominant_emotion']

'sad'

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

    # Draw a rectangle around the faces

for(x, y, w, h) in faces:

    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792c1b5a00>
```



```
font = cv2.FONT_HERSHEY_SIMPLEX
# use putText() method for#
inserting text on video

cv2.putText(img,
                    predictions['dominant_emotion'],(50,
50),
font, 1,
(0, 0, 255),
2,
cv2.LINE_4) ;
```

```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
<matplotlib.image.AxesImage at 0x2792c25e0a0>
```



```
img = cv2.imread('man-looking-shocked-fear-28561169.jpg')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
<matplotlib.image.AxesImage at 0x2792c454100>
```

0
100
200
300
400
500
600
700
800
0
200
400

```
predictions = DeepFace.analyze(img)
```

```
Action: race: 100%|████████████████████████████████████████████████████████████████████████
████████████████████████████████████████
| 4/4 [00:05<00:00,   1.34s/it]
```

```
predictions
```

```
{'emotion': {'angry': 0.00012416076060617343,'disgust':
   1.3881703946061474e-10,
   'fear': 0.18115636194124818, 'happy':
   6.731361423817361e-06, 'sad':
   5.0911141968867923e-08, 'surprise':
   99.81871247291565, 'neutral':
   1.0084419193798211e-10},
 'dominant_emotion': 'surprise',
 'region': {'x': 139, 'y': 232, 'w': 380, 'h': 380},
 'age': 36, 'gender':
 'Man',
 'race': {'asian': 4.307474195957184,
   'indian': 8.894021064043045,
   'black': 71.70406579971313,
   'white': 2.3631012067198753,
   'middle eastern': 1.8535083159804344,
   'latino hispanic': 10.877827554941177},
 'dominant_race': 'black'}
```

```
type(predictions)
```

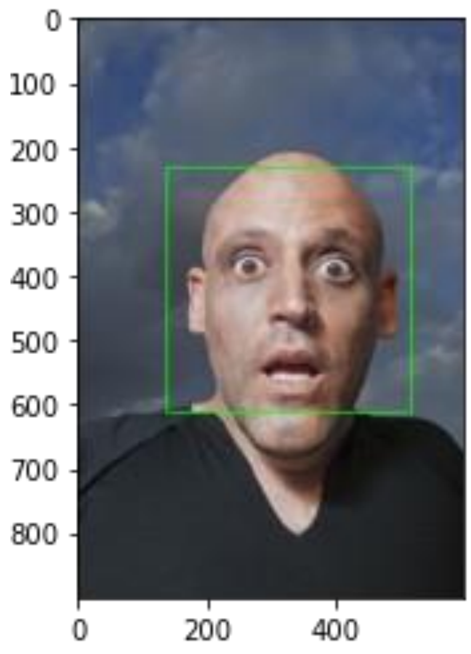
```
dict
```

```
predictions['dominant_emotion']
```

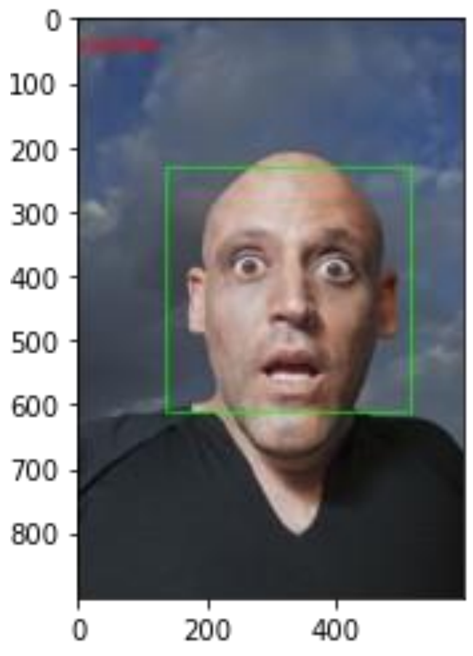
```
'surprise'
```

```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)
```

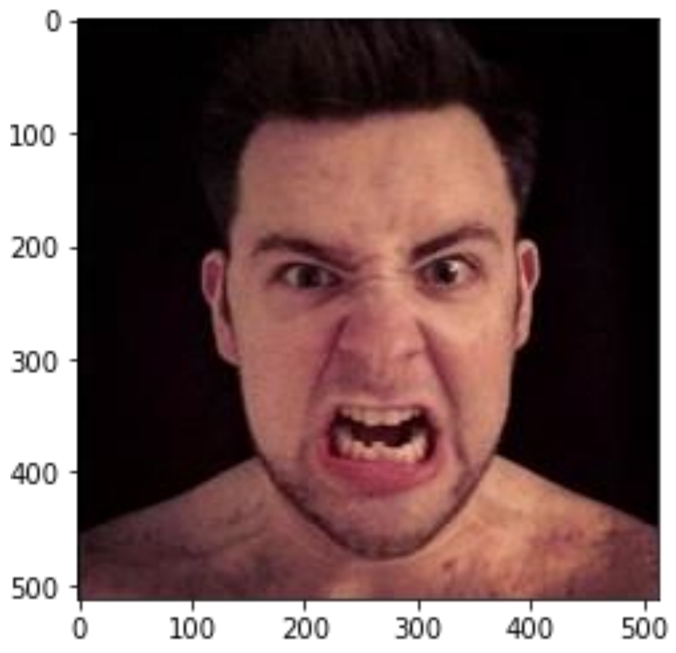
```
    # Draw a rectangle around the faces
for(x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792af3dbb0>
```



```
font = cv2.FONT_HERSHEY_SIMPLEX

    # use putText() method for#
    inserting text on video
cv2.putText(img,
                predictions['dominant_emotion'],(0,
                50),
                font, 1,
                (0, 0, 255),
                2,
                cv2.LINE_4) ;

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792c2d59a0>
```



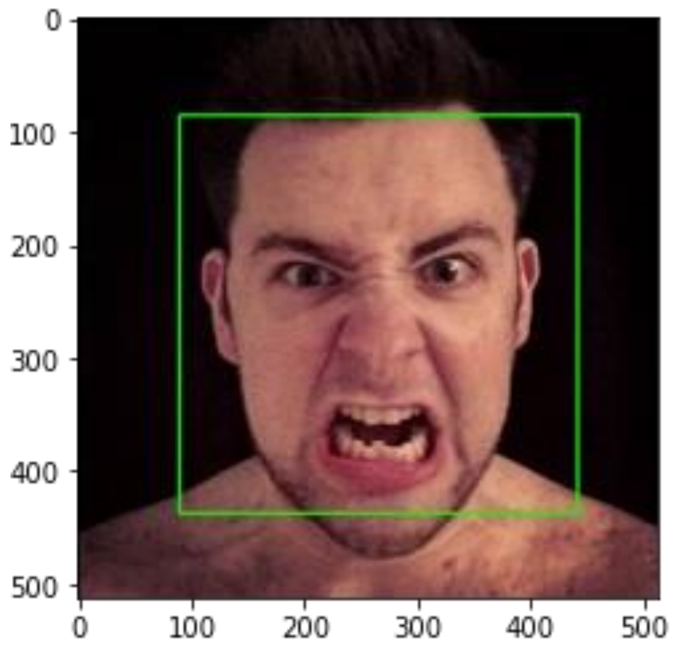
```
img = cv2.imread('OIP (1).jpg')
plt.imshow(cv2.cvtColor(img,
cv2.COLOR_BGR2RGB))
<matplotlib.image.AxesImage at 0x2792adacb80>
```



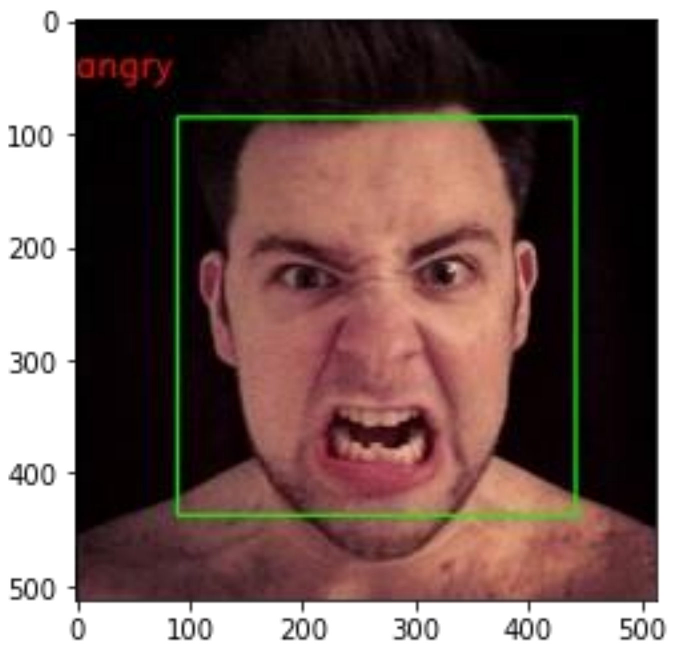
```
predictions = DeepFace.analyze(img)
```

```
Action: race: 100%|
[redacted]
| 4/4 [00:05<00:00,   1.48s/it]

predictions

{'emotion': {'angry': 99.77883097873104,
   'disgust': 0.0005056233169607907,
   'fear': 0.2154128417119791, 'happy':
   3.0685729618235155e-05,'sad':
   0.004931812733432306,
   'surprise': 0.0001574106437439904,
   'neutral': 0.00013888618987411126},
 'dominant_emotion': 'angry',
 'region': {'x': 90, 'y': 86, 'w': 351, 'h': 351},
 'age': 29, 'gender':
 'Man',
 'race': {'asian': 0.30703838979740267,
   'indian': 0.4968297541195447,
   'black': 0.06429813617825375,
   'white': 70.79094996560217,
   'middle eastern': 12.3315721951041,
   'latino hispanic': 16.009307979427817},
 'dominant_race': 'white'}

type(predictions)

dict

predictions['dominant_emotion']

'angry'

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

    # Draw a rectangle around the faces
```

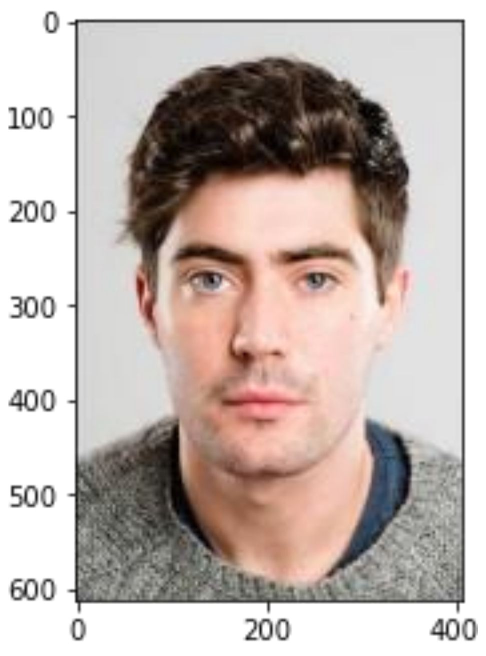
```
for(x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
<matplotlib.image.AxesImage at 0x2792ae6cb80>
```



```
font = cv2.FONT_HERSHEY_SIMPLEX

      # use putText() method for#
      inserting text on video
cv2.putText(img,
                  predictions['dominant_emotion'],(0,
                  50),
                  font, 1,
                  (0, 0, 255),
                  2,
                  cv2.LINE_4) ;

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792ae22fd0>
```



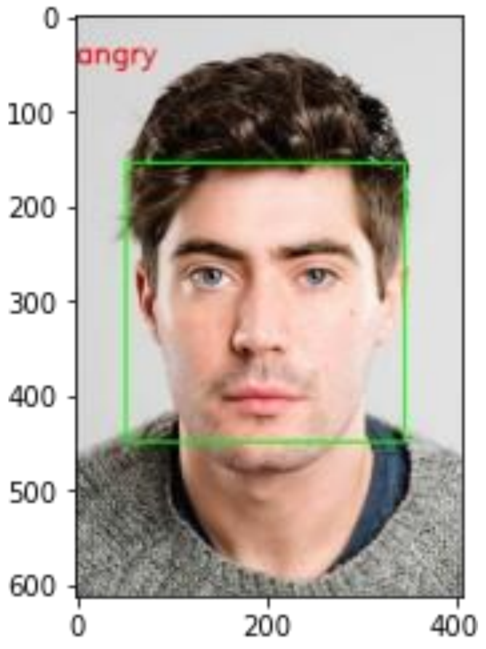
```
img = cv2.imread('istockphoto-178523753-612x612.jpg')

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792aeba640>
```



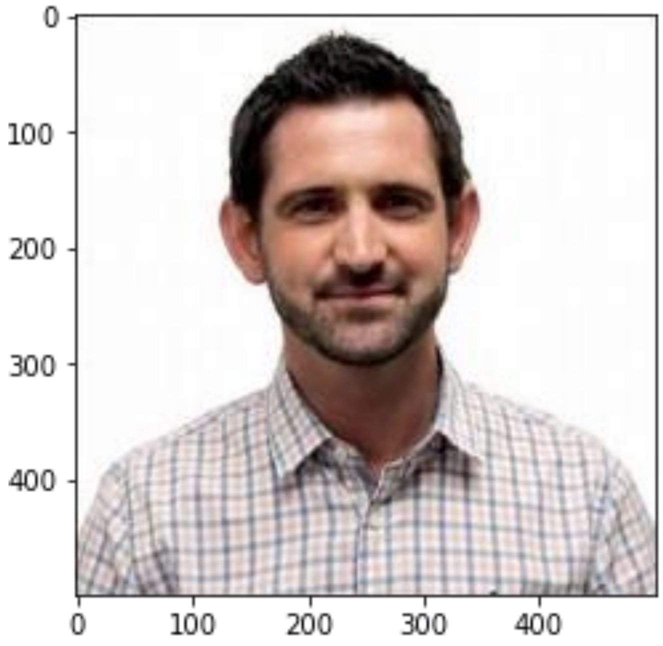
```
predictions = DeepFace.analyze(img)
```

```
Action: race: 100%|
```
[redacted]
```
| 4/4 [00:05<00:00,   1.28s/it]

predictions

{'emotion': {'angry': 37.543269991874695,
   'disgust': 5.384348522596838e-08, 'fear':
   1.425852719694376,
   'happy': 7.730822062512743e-05,
   'sad': 25.47897696495056,
   'surprise': 0.00025783735964068910,
   'neutral': 35.55156886577606},
  'dominant_emotion': 'angry',
  'region': {'x': 53, 'y': 155, 'w': 294, 'h': 294},
  'age': 26, 'gender':
  'Man',
  'race': {'asian': 0.7350302711910333,
   'indian': 5.313083417783191,
   'black': 0.1546381807827271,
   'white': 41.60953064078458,
   'middle eastern': 34.977052805072404,
   'latino hispanic': 17.21066739686339},
  'dominant_race': 'white'}

type(predictions)

dict

predictions['dominant_emotion']

'angry'

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
       #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

       # Draw a rectangle around the faces
```

```
for(x, y, w, h) in faces:

        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)font =

cv2.FONT_HERSHEY_SIMPLEX

        # use putText() method for#
        inserting text on video

cv2.putText(img,predictions['dominant_emotion'],(0, 50),
                                font, 1,
                                (0, 0, 255),
                                2,
                                cv2.LINE_4) ;

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792b181130>
```



```
img = cv2.imread('OIP (2).jpg')

plt.imshow(cv2.cvtColor(img,

cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x27939b41520>
```

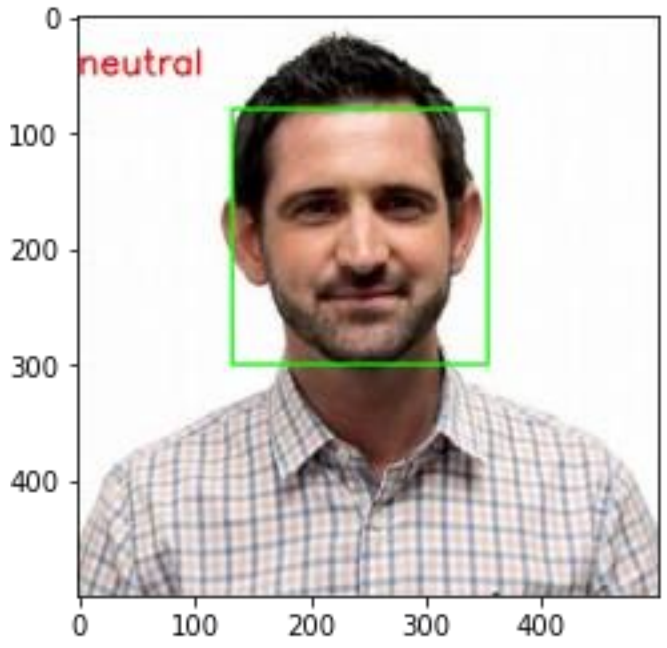


```
predictions = DeepFace.analyze(img)

Action: race: 100%|████████████████████████████████████████████████████
█████████████████████████
| 4/4 [00:04<00:00,   1.17s/it]

predictions

{'emotion': {'angry': 9.602805227041245,
   'disgust': 1.5423603149863907e-10, 'fear':
   0.19195474451407 79,
   'happy': 0.4071405157446861,
   'sad': 0.9448302909731865, 'surprise':
   3.3572996471775696e-05,'neutral':
   88.85324001312256},
  'dominant_emotion': 'neutral',
  'region': {'x': 133, 'y': 80, 'w': 220, 'h': 220},
  'age': 33, 'gender':
  'Man',
  'race': {'asian': 1.5129897334986708e-06,
   'indian': 0.007843961066100746,
   'black': 9.67733626566769e-07,
   'white': 89.91636633872986,
   'middle eastern': 9.800849854946136,
   'latino hispanic': 0.27493773959577084},
  'dominant_race': 'white'}
```

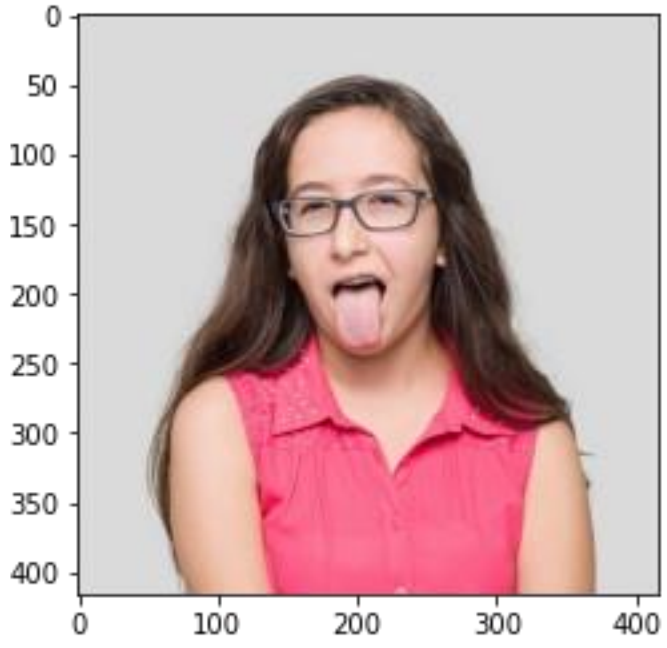
```
type(predictions)
```

```
dict
```

```
predictions['dominant_emotion']

'neutral'

faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

    # Draw a rectangle around the faces

for(x, y, w, h) in faces:

    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)font =

cv2.FONT_HERSHEY_SIMPLEX

    # use putText() method for#
    inserting text on video
cv2.putText(img,
                predictions['dominant_emotion'],(0,
                50),
                font, 1,
                (0, 0, 255),
                2,
                cv2.LINE_4) ;

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2793b863bb0>
```


neutral
0
100
200
300
400
0
100
200
300
400

```
img = cv2.imread('istockphoto-504877528-170667a.jpg')

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

<matplotlib.image.AxesImage at 0x2792b797130>
```



```
predictions = DeepFace.analyze(img)
Action: race: 100%|
```
[redacted]
```
| 4/4 [00:04<00:00,   1.08s/it]

predictions

{'emotion': {'angry': 24.011302230047647,
   'disgust': 0.11707974600380755,
   'fear': 44.403029822580855,
   'happy': 4.556556449757469,
   'sad': 19.149307811425253,
   'surprise': 0.19476007450504315,
   'neutral': 7.567953004132985},
  'dominant_emotion': 'fear',
  'region': {'x': 126, 'y': 82, 'w': 147, 'h': 147},
  'age': 37, 'gender':
```

```
'Woman',
'race': {'asian': 0.1393348560668528,
  'indian': 0.18270212458446622,
  'black': 0.009476066043134779,
  'white': 76.08774900436401,
  'middle eastern': 10.701151192188263,
   'latino hispanic': 12.879589200019836},
  'dominant_race': 'white'}
```

```
type(predictions)
```

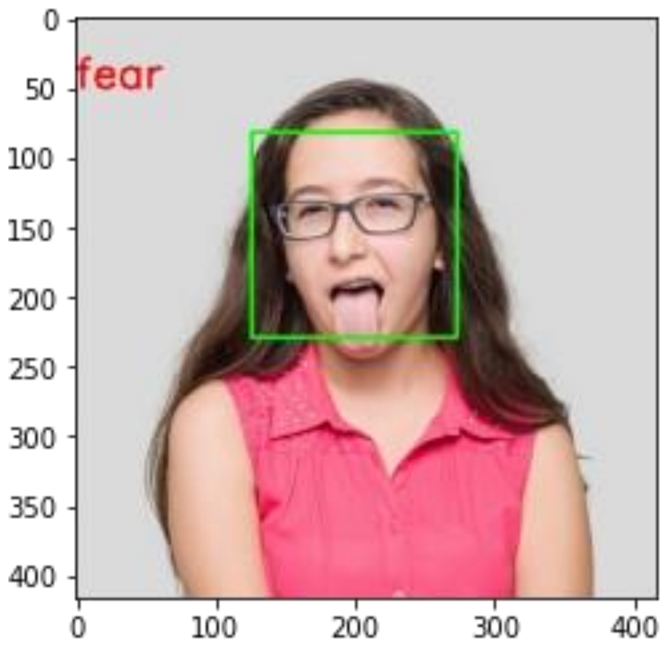
```
dict
```

```
predictions['dominant_emotion']
```

```
'fear'
```

```
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
      #print(faceCascade.empty())

faces = faceCascade.detectMultiScale(gray,1.1,4)

      # Draw a rectangle around the faces

for(x, y, w, h) in faces:

      cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)font =

cv2.FONT_HERSHEY_SIMPLEX

      # use putText() method for#
      inserting text on video
cv2.putText(img,
                        predictions['dominant_emotion'],(0,
                        50),
                        font, 1,
                        (0, 0, 255),
                        2,
                        cv2.LINE_4) ;

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x2792b7ff0a0>
```



FACE EMOTION RECOGNITION

```
from keras.models import load_model
from time import sleep
from keras.preprocessing.image import img_to_arrayfrom
keras.preprocessing import image
import cv2
import numpy as np

face_classifier = cv2.CascadeClassifier(r'C:\Users\Pinkush gupta\Desktop\new
p\haarcascade_frontalface_default.xml')
classifier =load_model(r'C:\Users\Pinkush gupta\Desktop\new p\model.h5')

emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad','Surprise']

cap = cv2.VideoCapture(0)



while True:

    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
```

```
faces = face_classifier.detectMultiScale(gray)

for (x,y,w,h) in faces:
```

```
            cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)roi_gray =
            gray[y:y+h,x:x+w]
            roi_gray = cv2.resize(roi_gray,
      (48,48),interpolation=cv2.INTER_AREA)



            if np.sum([roi_gray])!=0:
                  roi = roi_gray.astype('float')/255.0roi =
                  img_to_array(roi)
                  roi = np.expand_dims(roi,axis=0)

                  prediction = classifier.predict(roi)[0]
                  label=emotion_labels[prediction.argmax()]label_position
                  = (x,y)

      cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,1,
      (0,255,0),2)

     else:

                  cv2.putText(frame,'No Faces',
      (30,80),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
         cv2.imshow('Emotion Detector',frame)
         if cv2.waitKey(1) & 0xFF == ord('q'):

     break


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

7.3 System Testing

System testing was done by giving different training and testing datasets. This test was done to evaluate whether the system was predicting accurate result or not. During the phase of the development of the system our system was tested time and again. The series of testing conducted are as follows

7.3.1 Unit Testing

In unit testing, we designed the whole system in modularized pattern and each module was tested. Till we get the accurate output from the individual module we worked on the same module.

7.3.2 Integration Testing

After constructing individual modules all the modules were merged and a complete system was made. Then the system was tested whether the prediction given by training dataset to testing set was correct or not. We tried to meet the accuracy as higher as much as we can get. After spending a couple of days in integration testing the average accuracy of our system was 91%.

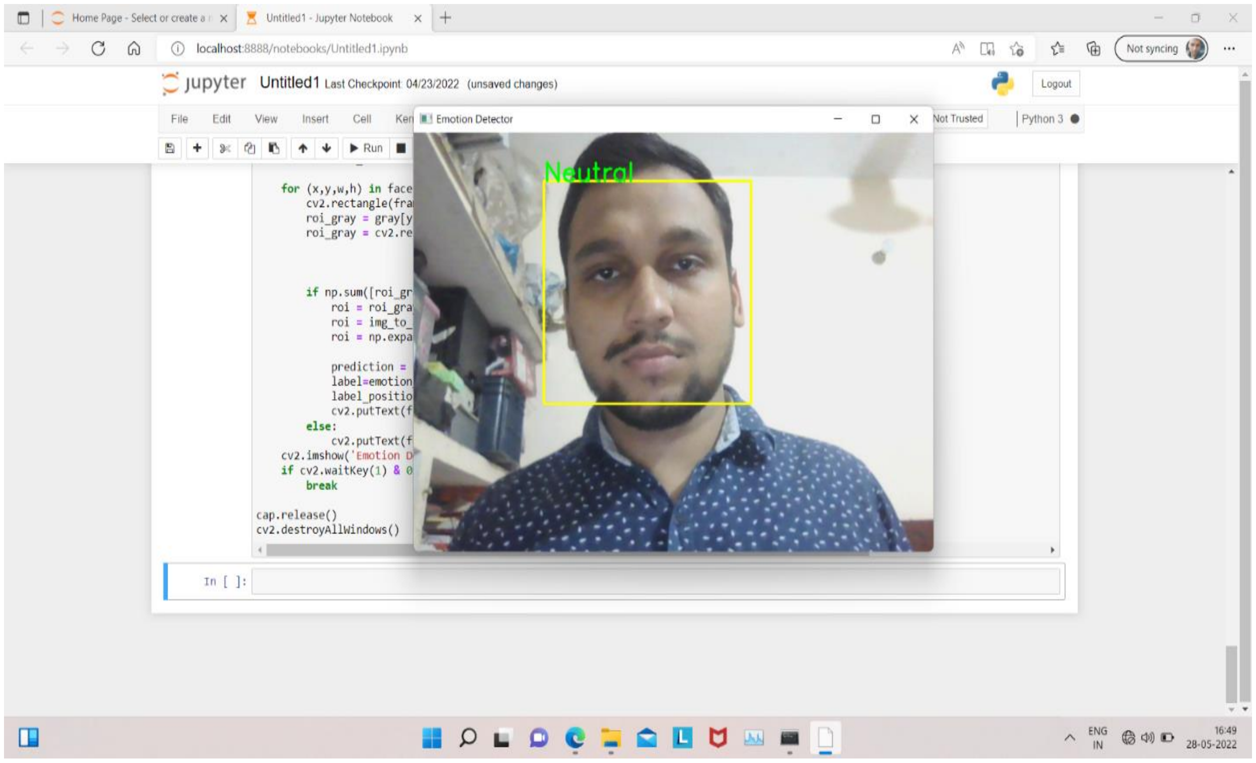
7.3.2.1 Alpha Testing

Alpha testing is the first stage of software engineering which is considered as a simulated or actual operational testing done by the individual member of the project. Alpha testing is conducted by the project developers, in context of our project.

7.3.2.2 Beta Testing

Beta testing comes continuously after alpha testing which is considered as a form of external user acceptance testing. The beta version of the program is developed to and provided to limited audience. This is the final test process in the case of this project. In this system the beta-testing is done by our colleagues and the project supervisor.

CHAPTER 8: RUNNING OUTPUT

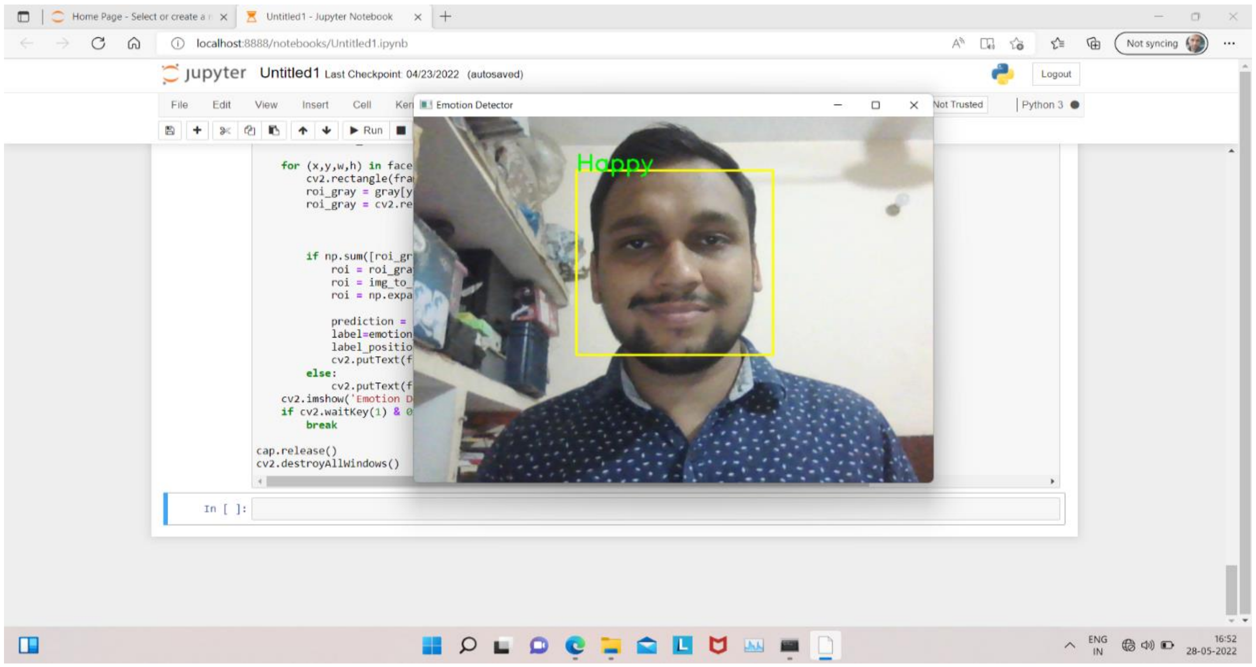


Home Page - Select or create a
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb
Not syncing
jupyter Untitled1 Last Checkpoint: 04/23/2022 (unsaved changes)
Logout
File Edit View Insert Cell Ker
Not Trusted
Python 3
Run
Emotion Detector
Neutral
ENG IN
16:49
28-05-2022

```
    for (x,y,w,h) in face
        cv2.rectangle(fra
        roi_gray = gray[y
        roi_gray = cv2.re



        if np.sum([roi_gr
            roi = roi_gra
            roi = img_to_
            roi = np.expa

            prediction =
            label=emotion
            label_positio
            cv2.putText(f
        else:
            cv2.putText(f
    cv2.imshow('Emotion D
    if cv2.waitKey(1) & 0
        break

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

```
In [ ]:
```

Fig 8.1: Neutral

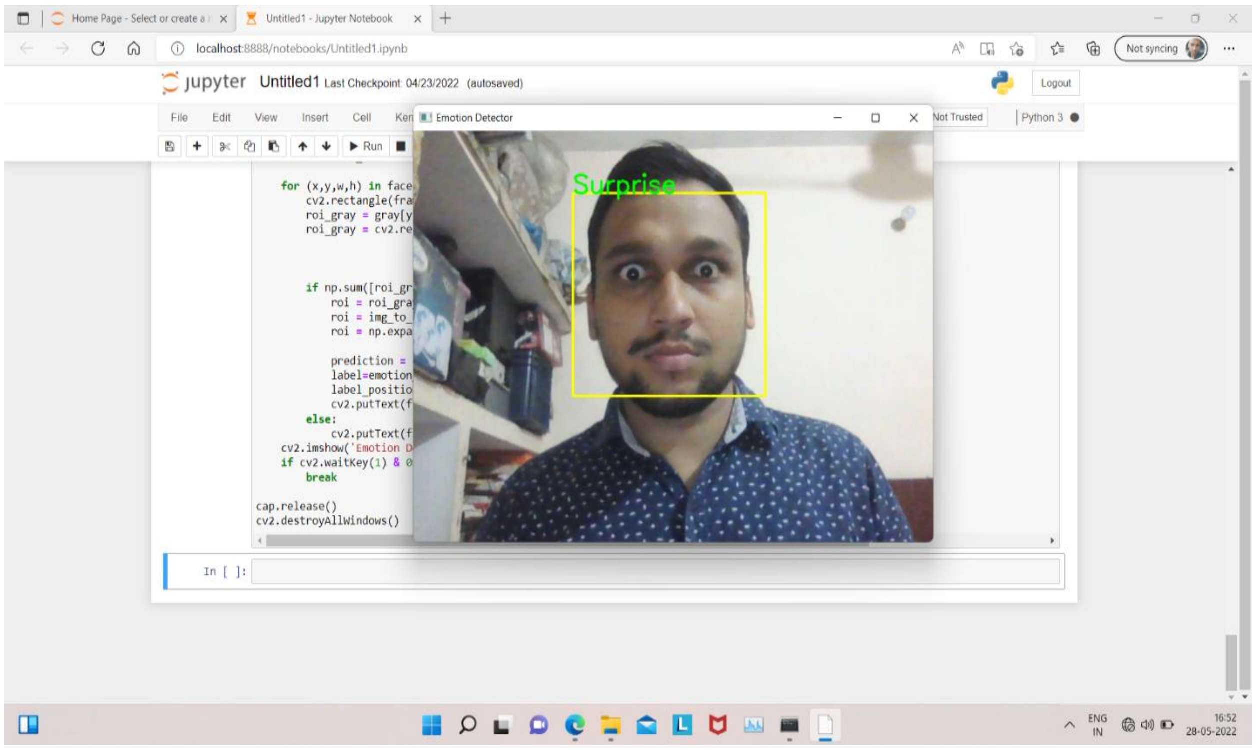


Home Page - Select or create a
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb
Not syncing
jupyter Untitled1 Last Checkpoint: 04/23/2022 (autosaved)
Logout
File Edit View Insert Cell Ker
Not Trusted
Python 3
Run
Emotion Detector
Happy
ENG IN
16:52
28-05-2022

```
    for (x,y,w,h) in face
        cv2.rectangle(fra
        roi_gray = gray[y
        roi_gray = cv2.re



        if np.sum([roi_gr
            roi = roi_gra
            roi = img_to_
            roi = np.expa

            prediction =
            label=emotion
            label_positio
            cv2.putText(f
        else:
            cv2.putText(f
    cv2.imshow('Emotion D
    if cv2.waitKey(1) & 0
        break

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

```
In [ ]:
```

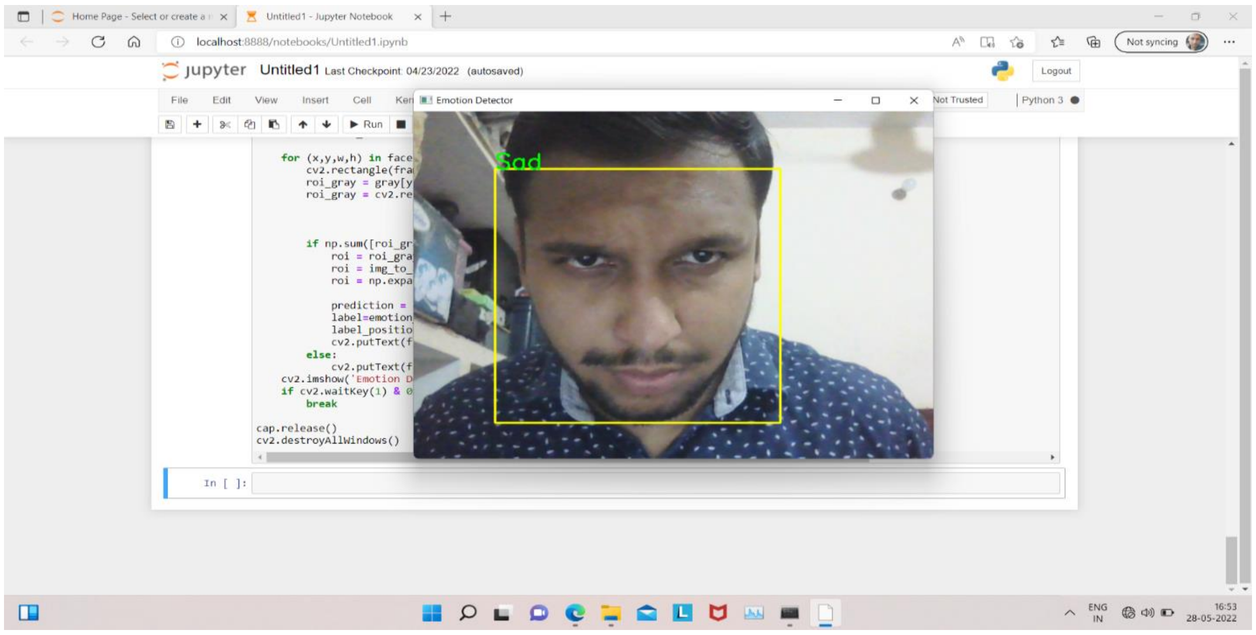
Fig 8.2: Happy


Home Page - Select or create a n
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb
Not syncing
jupyter
Untitled1 Last Checkpoint: 04/23/2022 (autosaved)
Logout
File
Edit
View
Insert
Cell
Ker
Run
Not Trusted
Python 3
Emotion Detector
Surprise
ENG IN
16:52
28-05-2022

```
    for (x,y,w,h) in face
        cv2.rectangle(fra
        roi_gray = gray[y
        roi_gray = cv2.re



        if np.sum([roi_gr
            roi = roi_gra
            roi = img_to_
            roi = np.expa

            prediction = 
            label=emotion
            label_positio
            cv2.putText(f
        else:
            cv2.putText(f
    cv2.imshow('Emotion D
    if cv2.waitKey(1) & 0
        break

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

```
In [ ]:
```

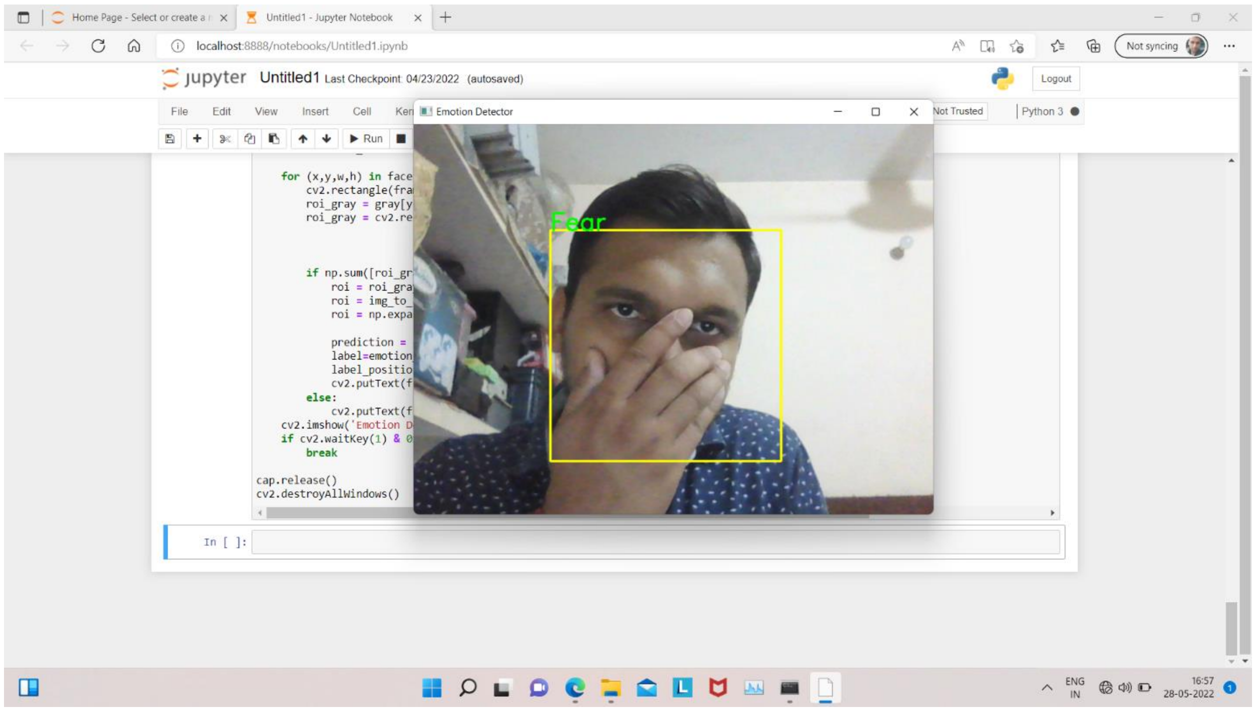
Fig 8.3: Surprise


Home Page - Select or create a n
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb
Not syncing
jupyter
Untitled1 Last Checkpoint: 04/23/2022 (autosaved)
Logout
File
Edit
View
Insert
Cell
Ker
Run
Not Trusted
Python 3
Emotion Detector
Sad
ENG IN
16:53
28-05-2022

```
    for (x,y,w,h) in face
        cv2.rectangle(fra
        roi_gray = gray[y
        roi_gray = cv2.re



        if np.sum([roi_gr
            roi = roi_gra
            roi = img_to_
            roi = np.expa

            prediction = 
            label=emotion
            label_positio
            cv2.putText(f
        else:
            cv2.putText(f
    cv2.imshow('Emotion D
    if cv2.waitKey(1) & 0
        break

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

```
In [ ]:
```

Fig 8.4: Sad


Home Page - Select or create a
Untitled1 - Jupyter Notebook
localhost:8888/notebooks/Untitled1.ipynb
Not syncing
jupyter
Untitled1 Last Checkpoint: 04/23/2022 (autosaved)
Logout
File
Edit
View
Insert
Cell
Ker
Emotion Detector
Not Trusted
Python 3
Run
Fear
In [ ]:
ENG
IN
16:57
28-05-2022

```
    for (x,y,w,h) in face
        cv2.rectangle(fra
        roi_gray = gray[y
        roi_gray = cv2.re



        if np.sum([roi_gr
            roi = roi_gra
            roi = img_to_
            roi = np.expa

            prediction = 
            label=emotion
            label_positio
            cv2.putText(f
        else:
            cv2.putText(f
    cv2.imshow('Emotion D
    if cv2.waitKey(1) & 0
        break

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

Fig 8.5: Fear

CHAPTER 9: CONCLUSION AND RECOMMENDATION

9.1 Conclusion

This project proposes an approach for recognizing the category of facial expressions. Face Detection and Extraction of expressions from facial images is useful in many applications, such as robotics vision, video surveillance, digital cameras, security and human-computer interaction. This project's objective was to develop a facial expression recognition system implementing the computer visions and enhancing the advanced feature extraction and classification in face expression recognition.

In this project, seven different facial expressions of different persons' images from different datasets have been analyzed. This project involves facial expression preprocessing of captured facial images followed by feature extraction using feature extraction using Local Binary Patterns and classification of facial expressions based on training of datasets of facial images based on Support Vector Machines. This project recognizes more facial expressions based on **Kaggle** face database.

9.2 Future Scope

Face expression recognition systems have improved a lot over the past decade. The focus has definitely shifted from posed expression recognition to spontaneous expression recognition. Promising results can be obtained under face registration errors, fast processing time, and high correct recognition rate (CRR) and significant performance improvements can be obtained in our system. System is fully automatic and has the capability to work with images feed. It is able to recognize spontaneous expressions. Our system can be used in Digital Cameras wherein the image can be captured only when the person smiles. In security systems which can identify a person, in any form of expression he presents himself. Rooms in homes can set the lights, television to a person’s taste when they enter the room. Doctors can use the system to understand the intensity of pain or illness of a deaf patient. Our system can be used to detect and track a user’s state of mind, and in mini-marts, shopping center to view the feedback of the customers to enhance the business etc.

CHAPTER 10: REFRENCES

1. M.S. Ratliff, E. Patterson

Emotion recognition using facial expressions with active appearance models
Proceedings of the Third IASTED International Conference on Human Computer Interaction, ACTA Press, Anaheim, CA, USA (2008), pp. 138-143

View Record in ScopusGoogle Scholar

2. Y.I. Tian, T. Kanade, F. Cohn J.

Recognizing action units for facial expression analysis
IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2) (2001), pp. 97-115

View Record in ScopusGoogle Scholar

3. Q. Mao, X. Pan, Y. Zhan, X. Shen

Using Kinect for real-time emotion recognition via facial expressions
Frontiers Inf Technol Electronic Eng, 16 (4) (2015), pp. 272-282

View Record in ScopusGoogle Scholar

4. B.Y.L. Li, S. Mian A., W. Liu, A. Krishna

Using Kinect for face recognition under varying poses, expressions, illumination and disguise
2013 IEEE Workshop on Applications of Computer Vision (WACV) (2013), pp. 186-192

ArticleDownload PDFGoogle Scholar

5. Microsoft Kinect, https://msdn.microsoft.com/en-us/library/jj131033.aspx

Google Scholar

6. Ahlberg J., *CANDIDE-3 - An Updated Parameterised Face*, 2001.

Google Scholar

7. S. Koelstra, I. Patras

Fusion of facial expressions and EEG for implicit affective tagging
Image and Vision Computing, 31 (2) (2013), pp. 164-174

ArticleDownload PDFView Record in ScopusGoogle Scholar

8. Ekman P., Friesen W., *Facial Action Coding System*, Consulting Psychologists Press, Stanford University, Palo Alto, 1977.

Google Scholar

9. Kinect, https://msdn.microsoft.com/en-us/library/jj130970.aspx

Google Scholar

10. Lundqvist D., Flykt A., Öhman A., The Karolinska Directed Emotional Faces - KDEF, CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, no. 1998,.

Google Scholar

11. M.R. Ogiela, R. Tadeusiewicz

Pattern recognition, clustering and classification applied to selected medical images
Studies in Computational Intelligence, 84 (2008), pp. 117-151

_View PDF

CrossRefView Record in ScopusGoogle Scholar