HUMAN EMOTION RECOGNITION

A MINI PROJECT REPORT

Submitted by

Ujan Som (RA2011047010004)
Roshan Upadhyay (RA2011047010015)
Animesh Gupta (RA2011047010021)
Priyansh Srivastava (RA2011047010022)
Neelak Dasgupta (RA2011047010031)
Prakhar Sharma (RA2011047010034)

Under the guidance of
Dr. R. Babu
(Assistant Professor, Department of Computational Intelligence)

in partial fulfillment for the award of the degree
Of
BACHELOR OF TECHNOLOGY
in
COMPUTATIONAL INTELLIGENCE
of
FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District
JUNE 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"HUMAN EMOTION RECOGNITION"** is the bonafide work of **"Ujan Som (RA2011047010004), Roshan Upadhyay (RA2011047010015), Animesh Gupta (RA2011047010021), Priyansh Srivastava (RA2011047010022), Neelak Dasgupta (RA2011047010031), Prakhar Sharma (RA2011047010034)"**, who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr.R.BABU	Dr. ANNIE UTHRA
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Dept. of Computational Intelligence
Dept. of Computational Intelligence	

Signature of the Internal Examiner Signature of the External Exam

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, **Dr. R. Babu**, his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing the project. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this project work.

Ujan Som
Roshan Upadhyay
Animesh Gupta
Priyansh Srivastava
Neelak Dasgupta
Prakhar Sharma

TABLE OF CONTENTS

ABSTRACT

Detecting and recognizing human emotion is a big challenge in Computer Vision and Artificial Intelligence. Emotions are a big part of human communication. Most of the communication takes place through emotion. The main aim of our project is to develop a robust system which can detect as well as recognize human emotion from live feed. There are some emotions which are universal to all human beings like angry, sad, happy, surprise, fear, disgust and neutral. This has applications in fields of security, surveillance, robotics etc.

Chapter – 1

Introduction

1.1 OBJECTIVE

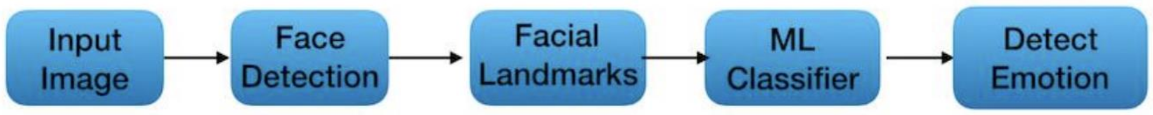
The goal of this project is to detect the emotional state of the person using Neural Networks and Machine Learning in real-time.

1.2 PROBLEM STATEMENT

The aim of our project is to create a model which can detect the emotions of a person, to understand their behaviour in different situations which will help various industries to get to know the wants and needs of different types of customers.

1.3 PROPOSED SOLUTION

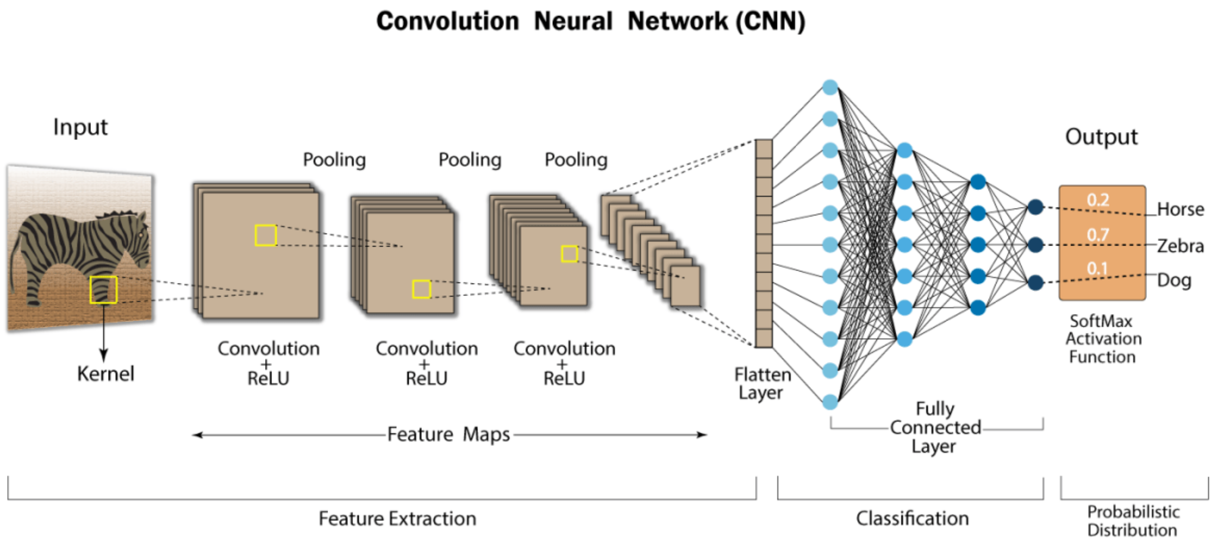
Our proposed solution is to build a Neural Network Model which can analyze the face of a person in real time and classify the kind of emotion showed. It will use Convolutional Neural Networks and a few parameters of OpenCV along with image pre-processing. The model will be tested on internal and external webcams and will show the current emotion on display.



CONCEPTS USED:

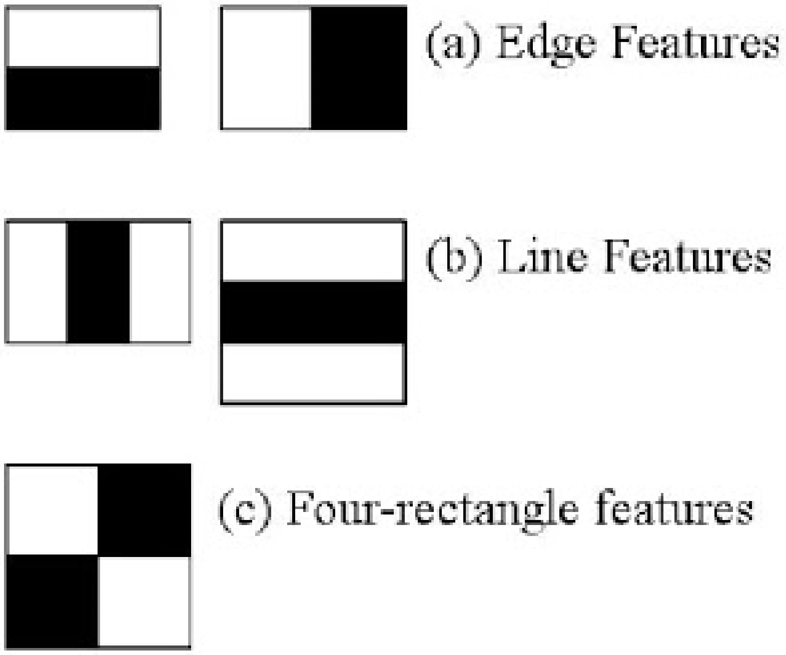
1.) Convolutional Neural Networks:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.



2.) Haar Cascades:

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them.



3.) Image pre-processing:

Image pre-processing techniques like flipping, rotation, gray scaling, etc. are performed to enhance the dataset. For real time face detection, frames are converted to images and compressed in size.

Chapter – 2

2.1 ALGORITHM AND ITS WORKING

Steps:

1.) All the relevant libraries are imported like pandas, numpy, tensorflow, etc.
2.) The dataset is imported and converted to a dataframe.
3.) Training and testing data is splitted.
4.) Data type of ‘Pixel’ column is converted to integer.
5.) Image is reshaped, rescales and converted to multiple images using rotation, flipping, etc.
6.) Train and test sets are optimized and batch size for the neural network is set.
7.) A CNN is formed of 3 convolution layers, 3 pooling layers, 1 flattening layer and finally a full connection.
8.) Output layer of dimensions 7 is formed.
9.) Model is fit into the data and accuracy is used as metric to train. Adam optimizer is used for Stochastic Gradient Descent.
10.) Model is saved for future use.
11.) Saved model is loaded to test in internal and external webcam.
12.) Haar cascade is used to detect face in the webcam.
13.) Model is tested and results are printed along with the detection of face in the dialogue box.

Chapter – 3

Tools and Software used:

3.1 Tools Description

1.) Python:
Python programming language will be used to solve the problem statement.

2.) Jupyter notebook:
The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

3.) Anaconda-navigator:
Anaconda Navigator is a desktop graphical user interface included in Anaconda that allows you to launch applications and easily manage conda packages, environments and channels without the need to use command line commands.

4.) MacOS (Monterey) and Windows 11:
Mac and Windows operating systems were used to access the tools.

5.) TensorFlow:
TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks

6.) Numpy:
NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

7.) Pandas:
Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

8.) OpenCV:
OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License.

9.) Webcam: The PC webcam or exterior webcam will be used for real-time testing.

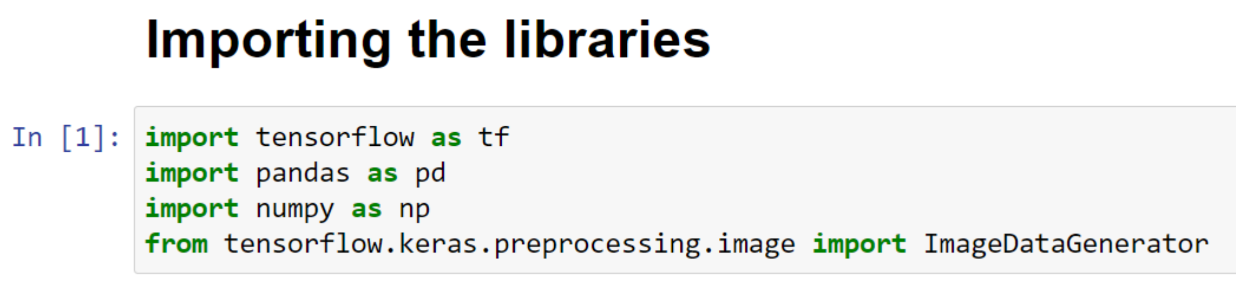
3.2 Dataset Description

- In our model we have used the fer2013 dataset which is an open-source data set that was made publicly available for a Kaggle competition.
- It contains 48 X 48-pixel grayscale images of the face. There are seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral) present in the data.
- The CSV file contains two columns that are emotion that contains numeric code from 0-6 and a pixel column that includes a string surrounded in quotes for each image.

Chapter 4

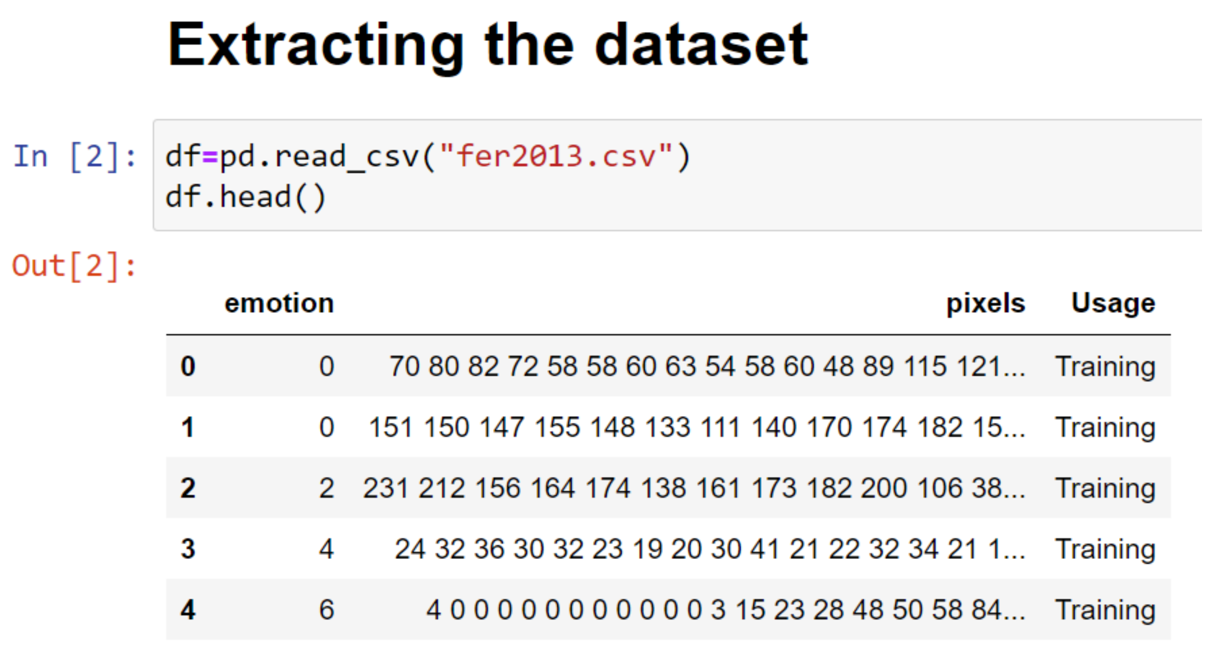
Result and Discussions

4.1 Code Implementation



Importing the libraries

```
In [1]: import tensorflow as tf
        import pandas as pd
        import numpy as np
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
```



Extracting the dataset

```
In [2]: df=pd.read_csv("fer2013.csv")
        df.head()
```

Out[2]:

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

```
In [3]: df.dtypes
```

```
Out[3]: emotion      int64
        pixels      object
        Usage       object
        dtype: object
```

```
In [4]: #Code cell to seperate out training and testing data
        X_train=[]
        y_train=[]
        X_test=[]
        y_test=[]
        for index,row in df.iterrows():
            p=row['pixels'].split(" ")
            if row['Usage']=='Training':
                X_train.append(np.array(p))
                y_train.append(row['emotion'])
            elif row['Usage']=='PublicTest':
                X_test.append(np.array(p))
                y_test.append(row['emotion'])
```

```
In [5]: #Conversion from object data type to integer
        X_train=np.array(X_train,dtype='uint8')
        y_train=np.array(y_train,dtype='uint8')
        X_test=np.array(X_test,dtype='uint8')
        y_test=np.array(y_test,dtype='uint8')
```

```
In [6]: #reshaping of images
        X_train=X_train.reshape(X_train.shape[0],48,48,1)
        X_test=X_test.reshape(X_test.shape[0],48,48,1)
```

```
In [7]: #rescaling and formation of multiple images
        train_datagen=ImageDataGenerator(rescale=1/255,shear_range=0.2,
                                         zoom_range=0.2,horizontal_flip=True,
                                         rotation_range=5)
        test_datagen=ImageDataGenerator(rescale=1/255)
```

In [8]:
```
#optimising data for efficient implementation of model and forming batches
train_flow=train_datagen.flow(X_train,y_train,batch_size=64)
test_flow=test_datagen.flow(X_test,y_test,batch_size=64)
```

Network

In [9]:
```
cnn=tf.keras.models.Sequential()
```

Convolution

In [10]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu',input_shape=[48,48,1]))
```

Pooling

In [11]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

Convolution

In [12]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
```

Pooling

In [13]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

Convolution

In [14]:
```
cnn.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,activation='relu'))
```

Pooling

In [15]:
```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2))
```

Flattening

In [16]:
```
cnn.add(tf.keras.layers.Flatten())
```

Full connection

```
In [17]: cnn.add(tf.keras.layers.Dense(units=64,activation='relu'))
         cnn.add(tf.keras.layers.Dense(units=128,activation='relu'))
         cnn.add(tf.keras.layers.Dense(units=256,activation='relu'))
         cnn.add(tf.keras.layers.Dense(units=512,activation='relu'))
         cnn.add(tf.keras.layers.Dense(units=1024,activation='relu'))
```

Output Layer

```
In [18]: cnn.add(tf.keras.layers.Dense(units=7,activation='softmax'))
```

Fitting of Model to data

```
In [19]: cnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
         history=cnn.fit_generator(train_flow,
                                   steps_per_epoch=len(X_train)/64,
                                   epochs=70,
                                   verbose=1,
                                   validation_data=test_flow,
                                   validation_steps=len(X_test)/64)

C:\Users\rosha\AppData\Local\Temp\ipykernel_17312\2649564826.py:2: UserWarning: `Model.fit_generator`
is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generat
ors.
  history=cnn.fit_generator(train_flow,
```

```
Epoch 1/70
448/448 [==============================] - 67s 146ms/step - loss: 1.7687 - accuracy: 0.2722 - val_los
s: 1.6249 - val_accuracy: 0.3650
Epoch 2/70
448/448 [==============================] - 61s 135ms/step - loss: 1.5903 - accuracy: 0.3711 - val_los
s: 1.4893 - val_accuracy: 0.4224
Epoch 3/70
448/448 [==============================] - 64s 142ms/step - loss: 1.4938 - accuracy: 0.4159 - val_los
s: 1.4260 - val_accuracy: 0.4372
Epoch 4/70
448/448 [==============================] - 71s 158ms/step - loss: 1.4360 - accuracy: 0.4419 - val_los
s: 1.3909 - val_accuracy: 0.4625
Epoch 5/70
448/448 [==============================] - 72s 161ms/step - loss: 1.4035 - accuracy: 0.4574 - val_los
s: 1.3819 - val_accuracy: 0.4795
Epoch 6/70
448/448 [==============================] - 71s 159ms/step - loss: 1.3561 - accuracy: 0.4740 - val_los
s: 1.3509 - val_accuracy: 0.4851
Epoch 7/70
448/448 [==============================] - 70s 157ms/step - loss: 1.3331 - accuracy: 0.4845 - val_los
s: 1.2879 - val_accuracy: 0.4968
```

.

.

.

```
Epoch 66/70
448/448 [==============================] - 69s 155ms/step - loss: 0.9633 - accuracy: 0.6339 - val_los
s: 1.2450 - val_accuracy: 0.5698
Epoch 67/70
448/448 [==============================] - 61s 137ms/step - loss: 0.9632 - accuracy: 0.6325 - val_los
s: 1.1927 - val_accuracy: 0.5734
Epoch 68/70
448/448 [==============================] - 63s 140ms/step - loss: 0.9534 - accuracy: 0.6368 - val_los
s: 1.2250 - val_accuracy: 0.5648
Epoch 69/70
448/448 [==============================] - 61s 136ms/step - loss: 0.9550 - accuracy: 0.6342 - val_los
s: 1.2058 - val_accuracy: 0.5673
Epoch 70/70
448/448 [==============================] - 61s 135ms/step - loss: 0.9518 - accuracy: 0.6374 - val_los
s: 1.2088 - val_accuracy: 0.5662
```

In [20]:
```
#save model
cnn.save('CNN model')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution
_op, _jit_compiled_convolution_op while saving (showing 3 of 3). These functions will not be directly
callable after loading.
```

```
INFO:tensorflow:Assets written to: CNN model\assets
```

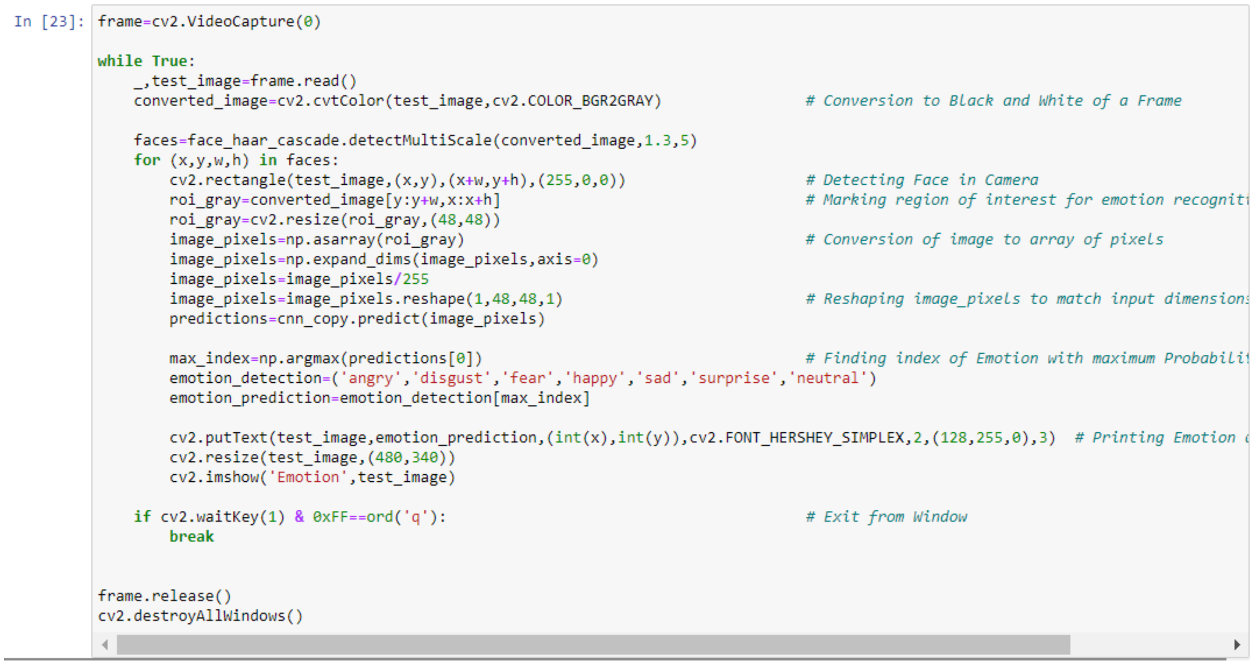
```
INFO:tensorflow:Assets written to: CNN model\assets
```

In [21]:
```
#load the model
cnn_copy=tf.keras.models.load_model('CNN model')
```

4.2 Performance Evaluation Metrics

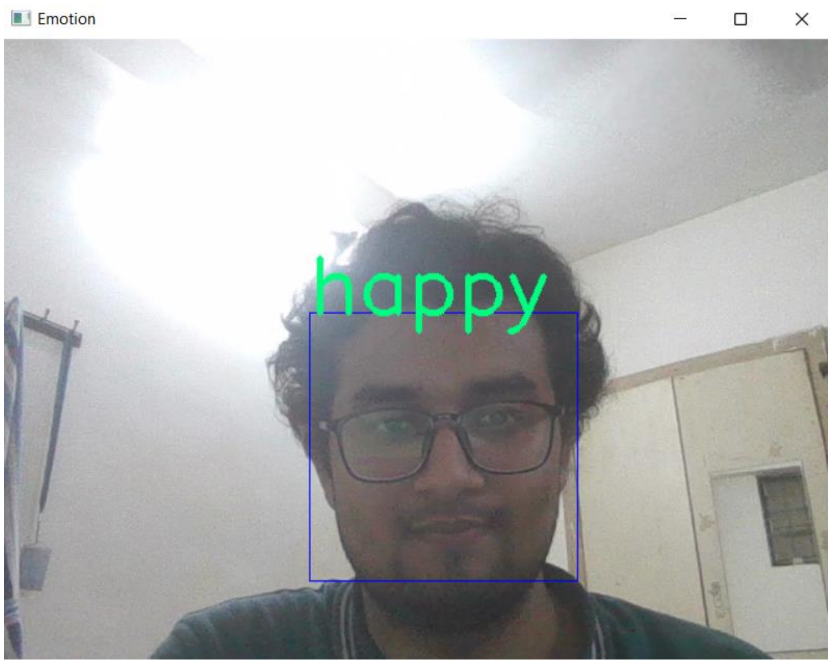
Real-Time Emotion Detection using Webcam

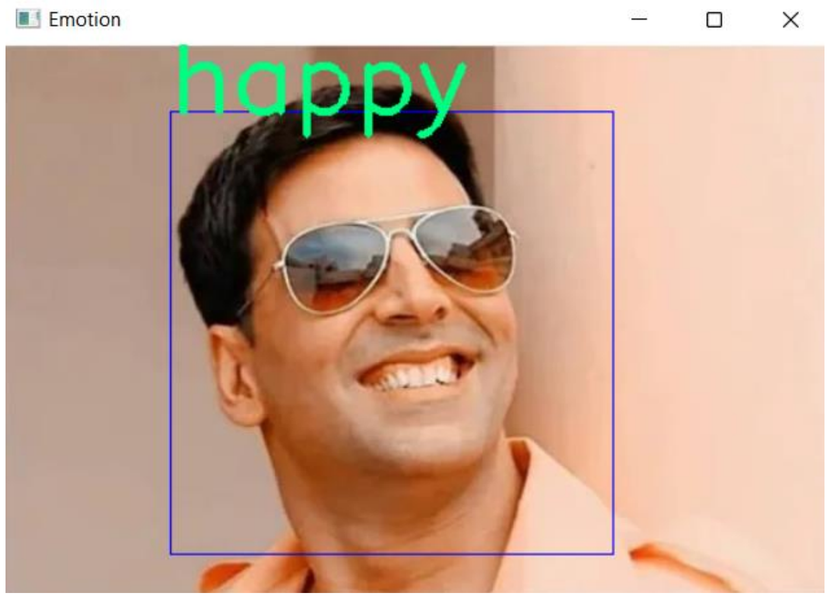
In [22]:
```
import cv2
face_haar_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
```

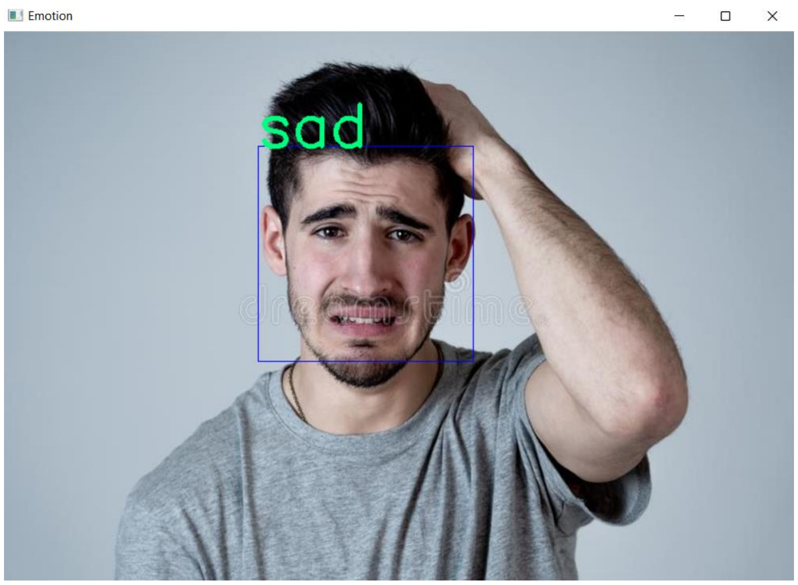


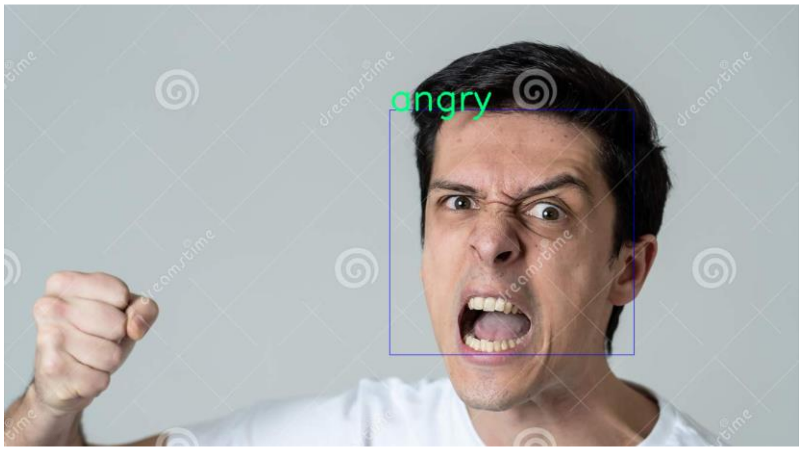
In [23]:
```
frame=cv2.VideoCapture(0)

while True:
    _,test_image=frame.read()
    converted_image=cv2.cvtColor(test_image,cv2.COLOR_BGR2GRAY)              # Conversion to Black and White of a Frame

    faces=face_haar_cascade.detectMultiScale(converted_image,1.3,5)
    for (x,y,w,h) in faces:
        cv2.rectangle(test_image,(x,y),(x+w,y+h),(255,0,0))                    # Detecting Face in Camera
        roi_gray=converted_image[y:y+w,x:x+h]                                   # Marking region of interest for emotion recogniti
        roi_gray=cv2.resize(roi_gray,(48,48))
        image_pixels=np.asarray(roi_gray)                                       # Conversion of image to array of pixels
        image_pixels=np.expand_dims(image_pixels,axis=0)
        image_pixels=image_pixels/255
        image_pixels=image_pixels.reshape(1,48,48,1)                            # Reshaping image_pixels to match input dimension
        predictions=cnn_copy.predict(image_pixels)

        max_index=np.argmax(predictions[0])                                     # Finding index of Emotion with maximum Probabili
        emotion_detection=('angry','disgust','fear','happy','sad','surprise','neutral')
        emotion_prediction=emotion_detection[max_index]

        cv2.putText(test_image,emotion_prediction,(int(x),int(y)),cv2.FONT_HERSHEY_SIMPLEX,2,(128,255,0),3)  # Printing Emotion

    if cv2.waitKey(1) & 0xFF==ord('q'):                                         # Exit from Window
        break


frame.release()
cv2.destroyAllWindows()
```

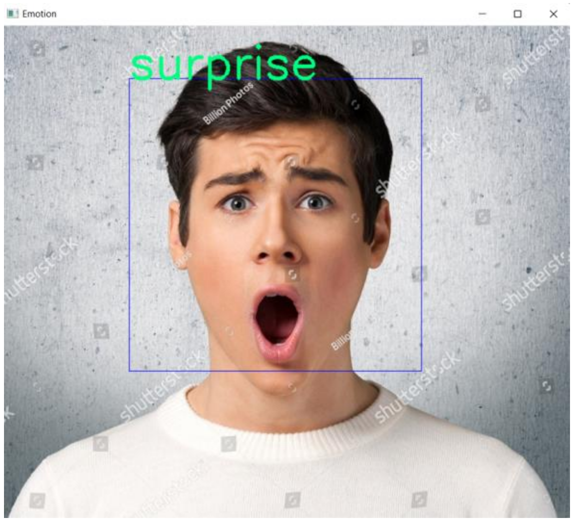
4.3 Result and Discussions






Emotion
sad


angry


Emotion
surprise

Chapter – 5

5.1 Conclusion

Hence successfully built a human emotion recognition model using Convolutional Neural Network. The trained model was used to categorise the emotions in real-time after we first defined the network and trained it to be capable of classifying the proper emotion. It provides good accuracy of approx. 60%. After designing a network that is capable of classifying the emotions, we use OpenCV for the detection of the faces and then pass it for prediction.