Project Name:

Human Emotion Recognition

[Deeplearning (Neural Network)]

Submitted by:

Satyam Gupta (11811891, A37)

Jassica Sahi (11807532, B52)

Section: KM073

GitHub link: https://github.com/coolsatyam9/facial-recognation.git
https://github.com/JassicaSahi/facial-recognition.git

Course Code: INT248

Submit to:

Mrs. Ankita Wadhawan

Lovely Professional University
PUNJAB INDIA

Human Emotion Recognition System

Introduction:

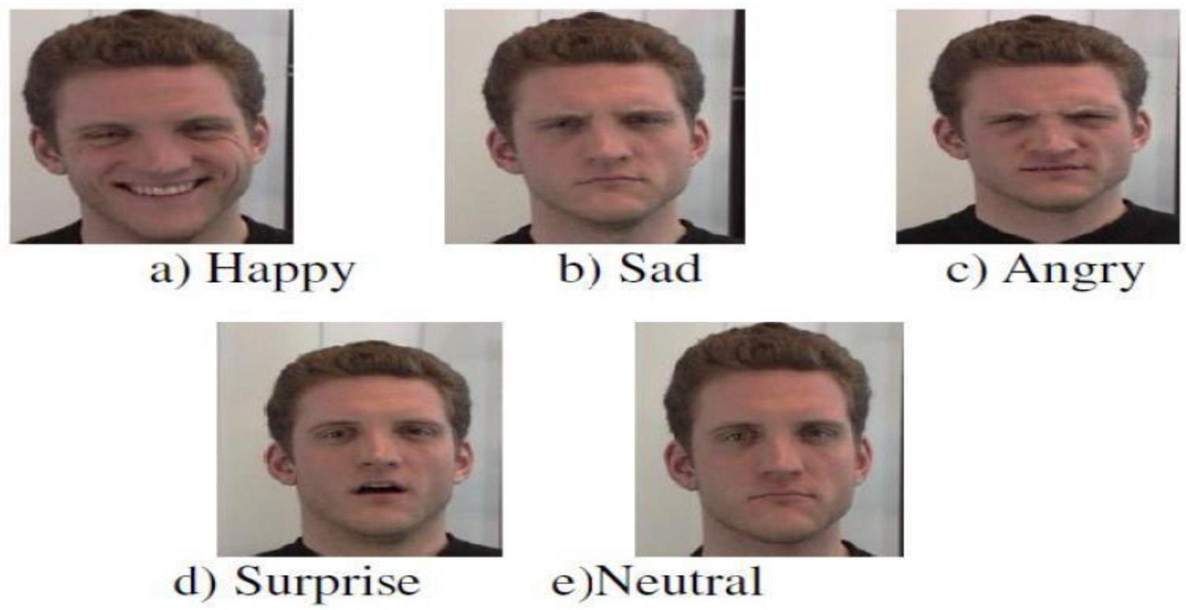
Emotion is an important aspect in the interaction and communication between people. Most of the communication between human beings involves non-verbal signs, and the social aspect of this communication is important. Humans also tend to include this social aspect when communicating with computers.

Problem Definition:

We have given the data set now using that data set we need to build a system for recognizing emotion detection. We have used existing data and the result of their analysis were 31 to 81 percentage correct. We used sequential model to train our neural network.

Many social media platforms using this functionality in their story feature for proving emojis based on user face emotion.

The output like be:



Contribution:

We are two person, Satyam Gupta, and Jassica Sahi. We both researched, analysed, implemented, and tested. It took around two weak to do analysis, implementation, and testing. As we are very passionate to learn Soft Computing and Machine learning, after allocation of project we start research and study about this project. And we both contribute equal effort to accomplish this project. And we are thanks to each other for giving such great time to work together.

Technology:

Python: for implementation of code.

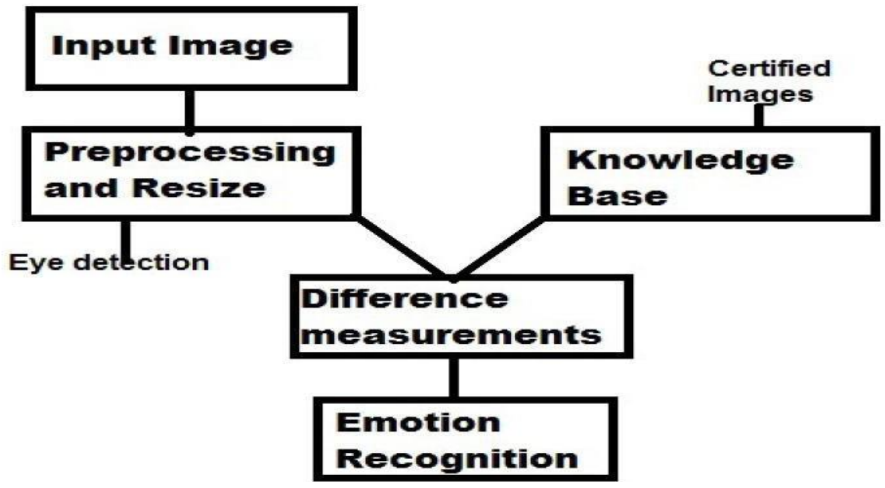
TensorFlow: for providing all needed libraries like Keras etc.

OpenCV: for processing image input.

And Time, Numpy etc.

Algorithm:

There is a model diagram is given to understand the implementation logic:



Knowledge base: It contains certified images which we will use for comparisons for the sake of emotion recognition. These images are highly qualified and these are stored in given database.

Pre-processing and resize: The main goal of this step is to enhance input image and also remove various type of noises.

Difference Measurements: It will find the difference between the input image and the certified images (stored in knowledge base) and give result to emotion recognition step.

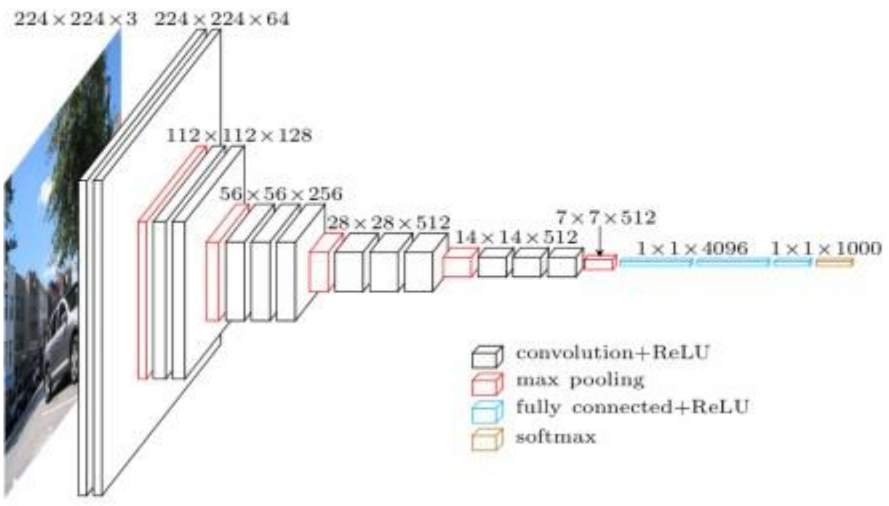
Implementation:

Part 1: Training Neural Network

For training a NN, it required sufficient relevant data for training and testing. We use 80% data for training and 20% data for testing.

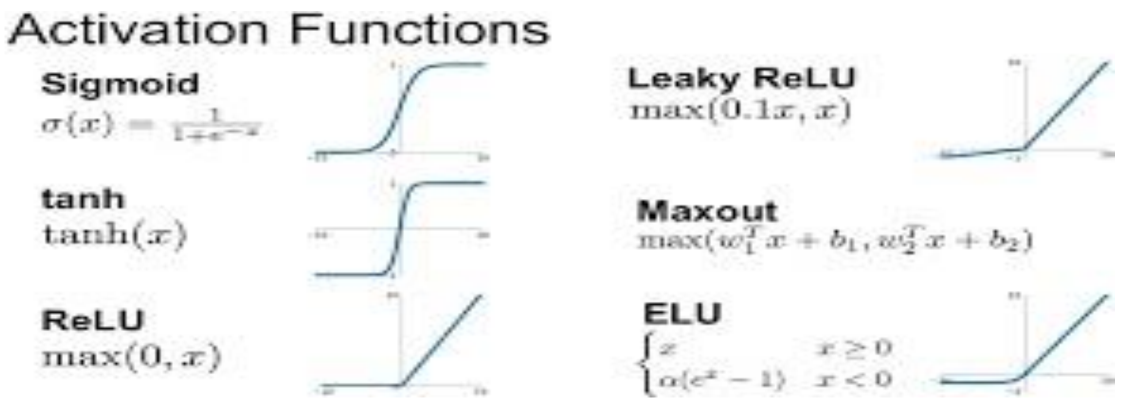
Step 1: We made 7 images from 1 image just changing its orientation and created another directory for validation.

Step 2: created a sequential model. Sequential model is NN that uses 2D data at input side and ant at the end of NN it changes that data into 1D. Here we made 7-layer NN model. At each layer (except input layer) it takes input from its previous layer and calculate the target value with appropriate weight and forward to next layer.



$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

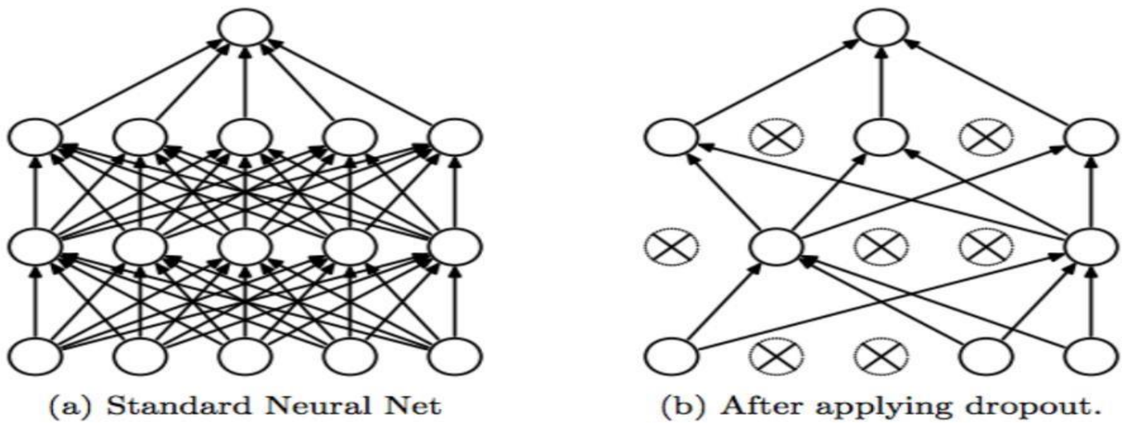
Step 3: The activation function. Activation function is function based which the weight is get updated. Here we used ELU activation function it a kind of exponential. ELU. Exponential Linear Unit or its widely known name ELU is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has a extra alpha constant which should be positive number. ELU is very similar to RELU except negative inputs.



Step 4: Batch Normalization and MaxPooling: Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer

for each mini batch. Max-Pooling: Maximum pooling, or max pooling, is a pooling operation that calculates the maximum, or largest, value in each patch of each feature map. The results are down sampled or pooled feature maps that highlight the most present feature in the patch, not the average presence of the feature in the case of average pooling.

Dropout class
The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.



(a) Standard Neural Net (b) After applying dropout.

Step 5: Repeat step 4 for 7 time but after 4th time we introduce Flatten: basically, flatten make 2D layer of matrix data into flat data. And in last two layers Dense layer come up: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activations of previous layer a.

Step 6: Start training to the NN for defined epoch and decided amount of data to be trained. During training figured out loss and accuracy if in continue three iteration loss increases and accuracy decreases the program need to stop training. At the end model is ready to test.

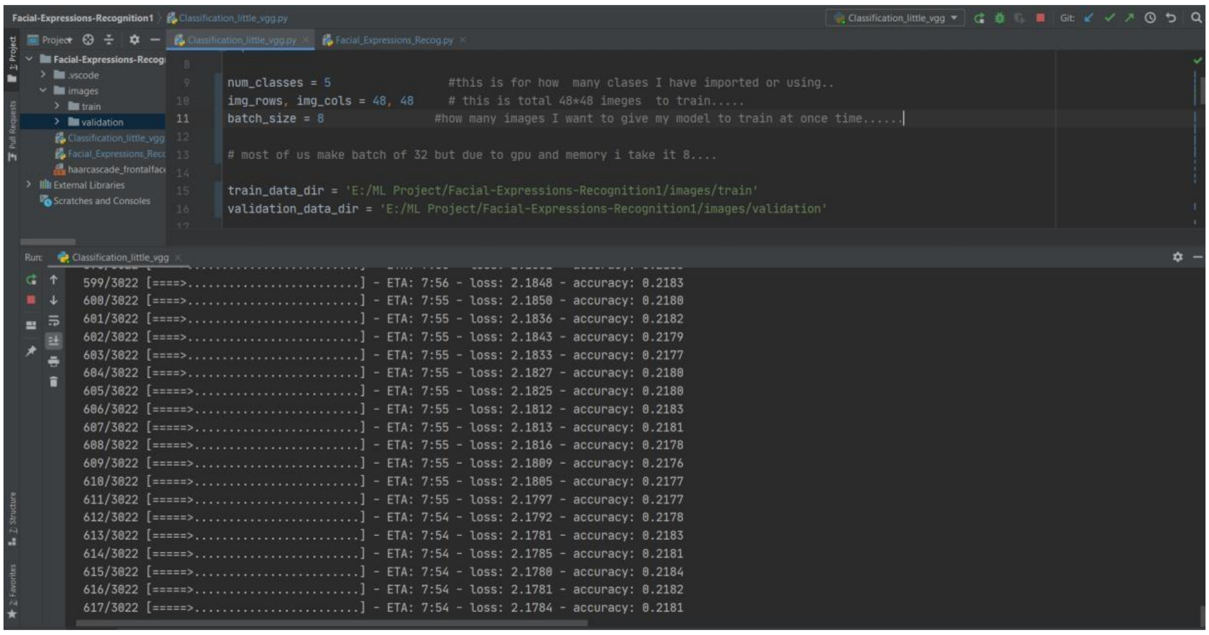
Part 2: Testing of Model

Step 1: Take input data using System webcam or from database of system. In our case we use OpenCV to take image as input. Then map with haarcascade file: a file that contain information about gesture. And apply to model for testing.

Step 2: After processing with input image give the output with target ans. Haarcascade file we easily find face in image ant the provided to input layer.

Step 3: Keep Showing Output with result status.

Output: 1 Training Part:


Facial-Expressions-Recognition1
Classification_little_vgg.py
Facial_Expressions_Recog.py
Facial-Expressions-Recog
.vscode
images
train
validation
Classification_little_vgg
Facial_Expressions_Reco
haarcascade_frontalface
External Libraries
Scratches and Consoles
Run: Classification_little_vgg

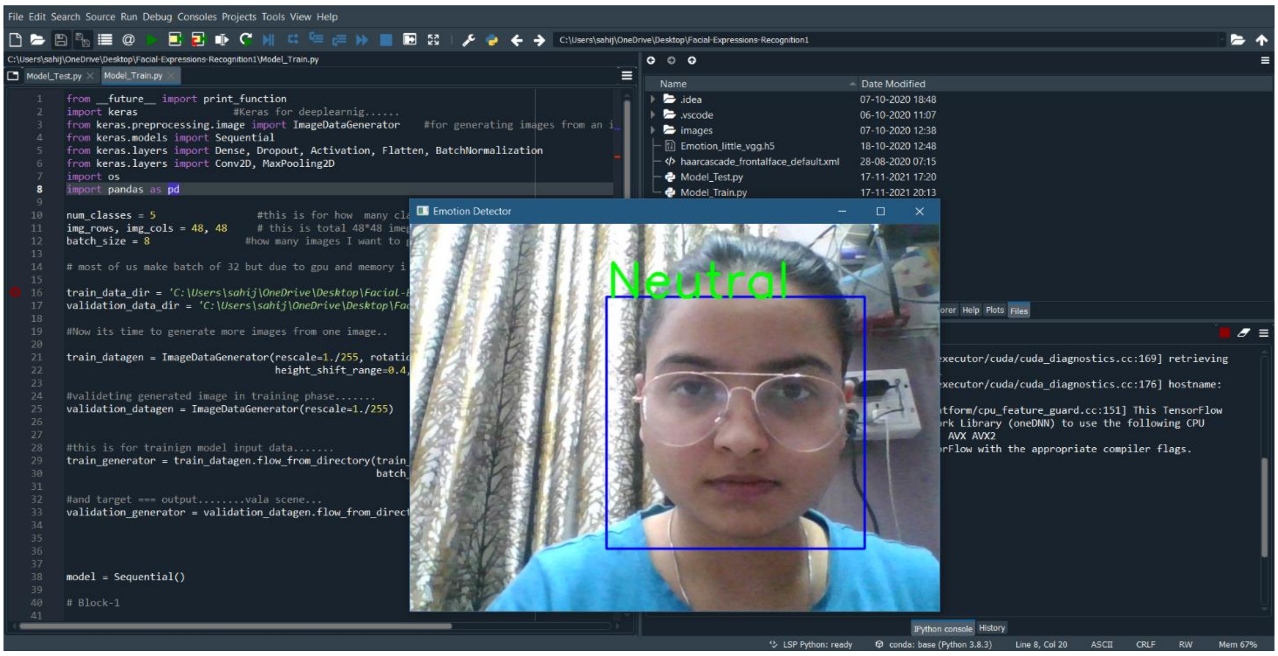
```
num_classes = 5                 #this is for how  many clases I have imported or using..
img_rows, img_cols = 48, 48     # this is total 48*48 imeges  to train.....
batch_size = 8                #how many images I want to give my model to train at once time......

# most of us make batch of 32 but due to gpu and memory i take it 8....

train_data_dir = 'E:/ML Project/Facial-Expressions-Recognition1/images/train'
validation_data_dir = 'E:/ML Project/Facial-Expressions-Recognition1/images/validation'
```

```
599/3022 [====>........................] - ETA: 7:56 - loss: 2.1848 - accuracy: 0.2183
600/3022 [====>........................] - ETA: 7:55 - loss: 2.1850 - accuracy: 0.2180
601/3022 [====>........................] - ETA: 7:55 - loss: 2.1836 - accuracy: 0.2182
602/3022 [====>........................] - ETA: 7:55 - loss: 2.1843 - accuracy: 0.2179
603/3022 [====>........................] - ETA: 7:55 - loss: 2.1833 - accuracy: 0.2177
604/3022 [====>........................] - ETA: 7:55 - loss: 2.1827 - accuracy: 0.2180
605/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1825 - accuracy: 0.2180
606/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1812 - accuracy: 0.2183
607/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1813 - accuracy: 0.2181
608/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1816 - accuracy: 0.2178
609/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1809 - accuracy: 0.2176
610/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1805 - accuracy: 0.2177
611/3022 [=====>.......................] - ETA: 7:55 - loss: 2.1797 - accuracy: 0.2177
612/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1792 - accuracy: 0.2178
613/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1781 - accuracy: 0.2183
614/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1785 - accuracy: 0.2181
615/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1780 - accuracy: 0.2184
616/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1781 - accuracy: 0.2182
617/3022 [=====>.......................] - ETA: 7:54 - loss: 2.1784 - accuracy: 0.2181
```

2 Testing Part:


File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1
C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1\Model_Train.py
Model_Test.py
Model_Train.py
Emotion Detector
Neutral
Help
Plots
Files
IPython console
History
LSP Python: ready
conda: base (Python 3.8.3)
Line 8, Col 20
ASCII
CRLF
RW
Mem 67%

```
from __future__ import print_function
import keras                    #Keras for deeplearnig......
from keras.preprocessing.image import ImageDataGenerator    #for generating images from an i
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from keras.layers import Conv2D, MaxPooling2D
import os
import pandas as pd

num_classes = 5                 #this is for how  many cla
img_rows, img_cols = 48, 48     # this is total 48*48 imeg
batch_size = 8                #how many images I want to g

# most of us make batch of 32 but due to gpu and memory i

train_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Facial-E
validation_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Fac

#Now its time to generate more images from one image..

train_datagen = ImageDataGenerator(rescale=1./255, rotatic
                                   height_shift_range=0.4,

#valideting generated image in training phase.......
validation_datagen = ImageDataGenerator(rescale=1./255)


#this is for trainign model input data.......
train_generator = train_datagen.flow_from_directory(train_
                                                   batch_

#and target === output........vala scene...
validation_generator = validation_datagen.flow_from_direct




model = Sequential()

# Block-1
```

Name	Date Modified
.idea	07-10-2020 18:48
.vscode	06-10-2020 11:07
images	07-10-2020 12:38
Emotion_little_vgg.h5	18-10-2020 12:48
haarcascade_frontalface_default.xml	28-08-2020 07:15
Model_Test.py	17-11-2021 17:20
Model_Train.py	17-11-2021 20:13

```
executor/cuda/cuda_diagnostics.cc:169] retrieving

executor/cuda/cuda_diagnostics.cc:176] hostname:

tform/cpu_feature_guard.cc:151] This TensorFlow
rk Library (oneDNN) to use the following CPU
 AVX AVX2
rFlow with the appropriate compiler flags.
```

Code Implementation:

1: Training:

```
from_future_import print_function

import keras            #Keras for deeplearnig......

from keras.preprocessing.image import ImageDataGenerator #for generating images from an images...

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization

from keras.layers import Conv2D, MaxPooling2D

import os


num_classes = 5          #this is for how many clases I have imported or using..

img_rows, img_cols = 48, 48    # this is total 48*48 imeges to train.....

batch_size = 8           #how many images I want to give my model to train at once time......

# most of us make batch of 32 but due to gpu and memory i take it 8....

train_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1'

validation_data_dir = 'C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1'

#Now its time to generate more images from one image..

train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=30, shear_range=0.3, zoom_range=0.3,
width_shift_range=0.4,

                    height_shift_range=0.4, horizontal_flip=True, fill_mode='nearest')

#valideting generated image in training phase.......

validation_datagen = ImageDataGenerator(rescale=1./255)

#this is for trainign model input data.......

train_generator = train_datagen.flow_from_directory(train_data_dir, color_mode='grayscale',
target_size=(img_rows, img_cols),

                              batch_size=batch_size, class_mode='categorical', shuffle=True)

#and target === output........vala scene...

validation_generator = validation_datagen.flow_from_directory(validation_data_dir, color_mode='grayscale',

                                    target_size=(img_rows, img_cols), batch_size=batch_size,

                                    class_mode='categorical', shuffle=True)

model = Sequential()

# Block-1
```

```
model.add(Conv2D(32, (3, 3), padding='same', kernel_initializer='he_normal', input_shape=(img_rows,
img_cols, 1)))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(Conv2D(32, (3, 3), padding='same', kernel_initializer='he_normal', input_shape=(img_rows,
img_cols, 1)))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))

# Block-2

model.add(Conv2D(64, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))

# Block-3

model.add(Conv2D(128, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(Conv2D(128, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))

# Block-4

model.add(Conv2D(256, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))

model.add(BatchNormalization())

model.add(Conv2D(256, (3, 3), padding='same', kernel_initializer='he_normal'))

model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
# Block-5
model.add(Flatten())
model.add(Dense(64, kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# Block-6
model.add(Dense(64, kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# Block-7
model.add(Dense(num_classes, kernel_initializer='he_normal'))
model.add(Activation('softmax'))
print(model.summary())
from keras.optimizers import RMSprop, SGD, Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
checkpoint = ModelCheckpoint('Emotion_little_vgg.h5',
                             monitor='val_loss',
                             mode='min',
                             save_best_only=True,
                             verbose=1)
earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )
reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                    patience=3,
                    verbose=1,
                    min_delta=0.0001)
callbacks = [earlystop, checkpoint, reduce_lr]
model.compile(loss='categorical_crossentropy',
        optimizer=Adam(lr=0.001),
        metrics=['accuracy'])
nb_train_samples = 24176
nb_validation_samples = 3006
epochs=25
history=model.fit_generator(
         train_generator,
         steps_per_epoch=nb_train_samples//batch_size,
         epochs=epochs,
         callbacks=callbacks,
         validation_data=validation_generator,
         validation_steps=nb_validation_samples//batch_size)
```

2:Testing:

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

face_classifier = cv2.CascadeClassifier(r'C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-
Recognition1\haarcascade_frontalface_default.xml')
classifier = load_model(r'C:\Users\sahij\OneDrive\Desktop\Facial-Expressions-Recognition1\Emotion_little_vgg.h5')

class_labels = ['Angry','Happy','Neutral','Sad','Surprise']

cap = cv2.VideoCapture(0)
```

```
while True:
    # Grab a single frame of video
    ret, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
    # rect,face,image = face_detector(frame)



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

        # make a prediction on the ROI, then lookup the class

            preds = classifier.predict(roi)[0]
            label=class_labels[preds.argmax()]
            label_position = (x,y)
            cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
        else:
            cv2.putText(frame,'No Face Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

--Thank You--