Facial-Recognition

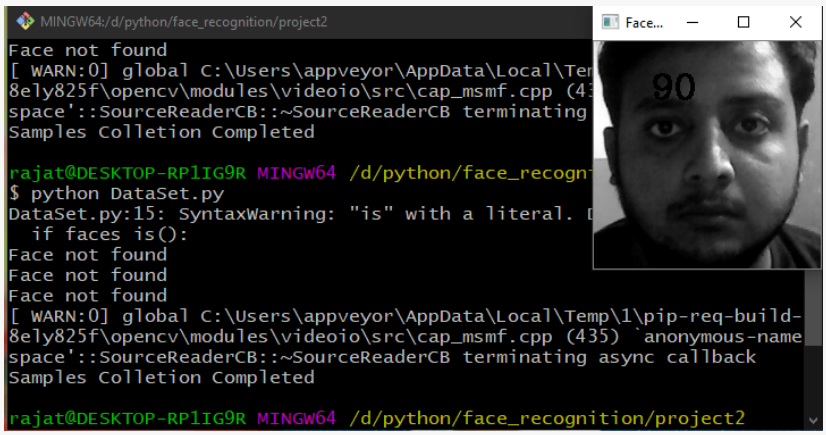
#Collect Data Sample from WebCam

```
import cv2
import numpy as np


#Face_Detect and Collect Sample

face_classifier =
cv2.CascadeClassifier('D:/IDM/Program/opencv/sources/data/haarcascades/haarcascade_frontalface_default.xml')


#Function which Detect Face

def face_extractor(img):

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #Convert Image into Gray
    faces = face_classifier.detectMultiScale(gray,1.3,5) #Rectangle on Face

    if faces is():
        return None

    for(x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w] #Size of Rectangle

    return cropped_face


#Open Web_cam

cap = cv2.VideoCapture(0)
count = 0

while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count+=1
        face = cv2.resize(face_extractor(frame),(200,200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

       #File Location where Sample Images Save

        file_name_path = 'D:/python/face_recognition/project2/Images/'+str(count)+'.jpg'

        cv2.imwrite(file_name_path,face)

        #Write text infront of Images

        cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
        cv2.imshow('Face Sample',face)
    else:
        print("Face not found")
        pass
```

#Face Sample wait for 100 counting or stop when we hit "Enter" Key ==13

```
    if cv2.waitKey(1)==13 or count==100:
        break
        cap.release()
        cv2.destroyAllWindows()
         print('Samples Colletion Completed ')
```

#OUTPUT



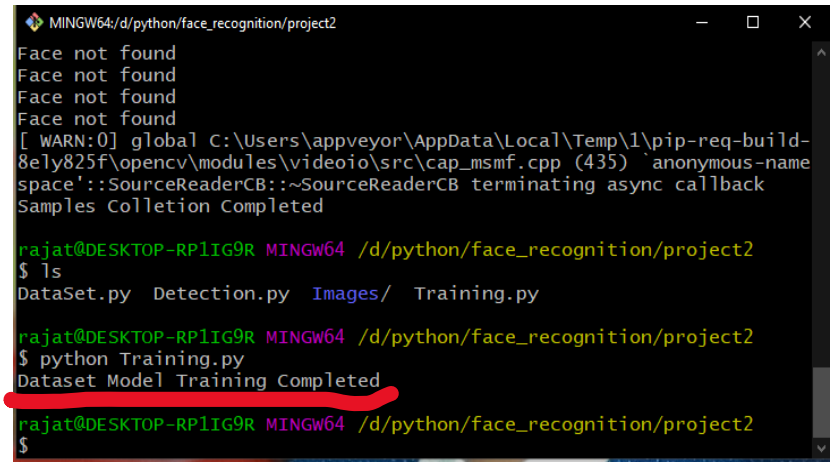
#Trained Data Sample

```
import cv2
import numpy as np
from os import listdir #import file from location
from os.path import isfile, join #import path

data_path = 'D:/python/face_recognition/project2/Images/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))] #Connect file

Training_Data, Labels = [ ], [ ] #martix store in both array

for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)


#Local Binary face algorithm
#LBPH is one of the easiest face recognition
algorithms. # It can represent local features in
the images.
# It is possible to get great results (mainly in a controlled
environment). # It is robust against monotonic gray scale
transformations.
```

3

```
# It is provided by the OpenCV library (Open Source Computer Vision Library).

model = cv2.face.LBPHFaceRecognizer_create() #Store Binary code Of PIC

model.train(np.asarray(Training_Data), np.asarray(Labels)) #Call Functions

print("Dataset Model Training Completed ")
```

#OUTPUT

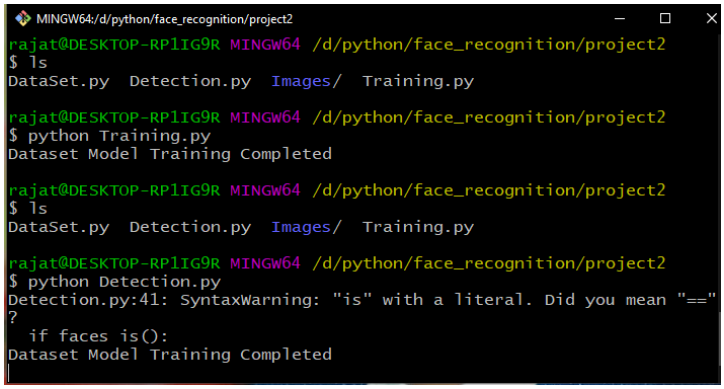



```
# extract face modal
face_classifier =
cv2.CascadeClassifier('D:/IDM/Program/opencv/sources/data/haarcascades/haarcascade_frontalface_default.xml')

# Function which Detect Face
def face_detector(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #Convert to grey scale
    #multiple scale style and at the same time following sliding window strategy
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():
        return img,[] #If face is empty
        #rectangle on face
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,0),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))

    return img,roi
        #Open WebCam
cap = cv2.VideoCapture(0)
while True:

    ret, frame = cap.read()

    image, face = face_detector(frame)
  #comapre face sample from face
```

4

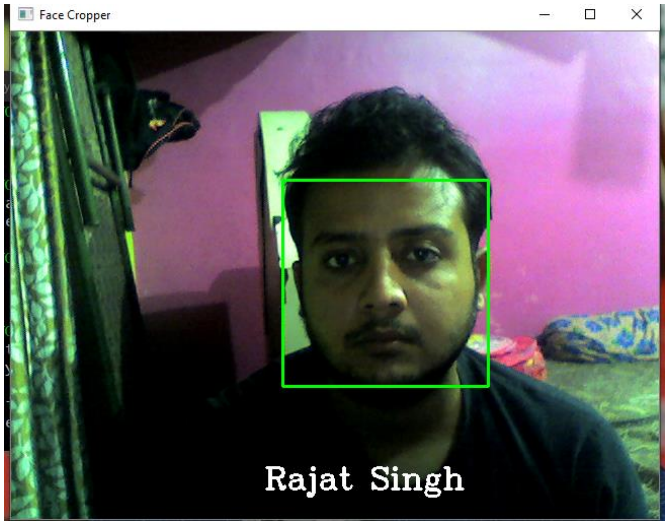
```
try:
    face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
    result = model.predict(face)

    if result[1] < 500:
    confidence = int(100*(1-(result[1])/300))

    if confidence > 82:
          #print name of face on screen
          cv2.putText(image, "Rajat Singh", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
          cv2.imshow('Face Cropper', image)

    else:
          #for unknown faces
          cv2.putText(image, "Unknown", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
          cv2.imshow('Face Cropper', image)



  except:
          #when face not found
          cv2.putText(image, "Face Not Found", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)
          cv2.imshow('Face Cropper', image)
          pass

  if cv2.waitKey(1)==13:
     break

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

#OUTPUT

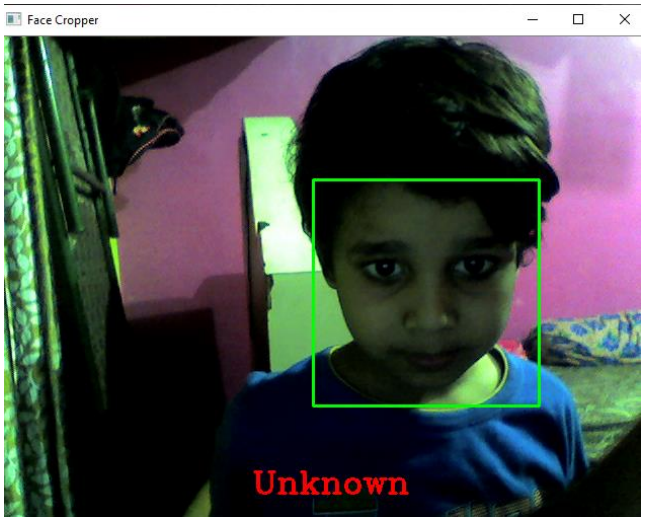


#When Face Match

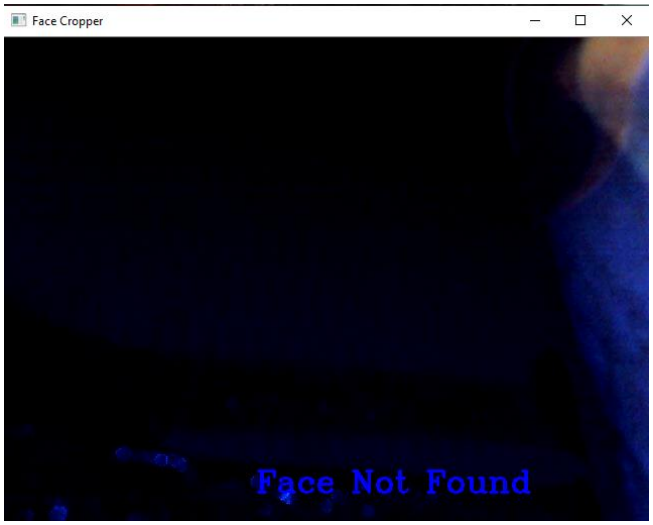
5



#When Face does not Match



#When No Face Detect



6