A

MINOR PROJECT ON

FACE RECOGNITION USING PYTHON

A Project report for Minor Project

Submitted By -

SOUNAK SARKAR

ROHIT DEY

SAYAN PODDAR

MD. ZEESHAN

SHEEBA TAZEEN

in partial fulfillment for the award of the degree of

Bachelors in Computer Applications

DECLARATION

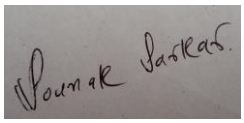
I undersigned, hereby declare that the project titled "Face Recognition Using Python" submitted in partial fulfillment for the award of Degree of Bachelors in Computer Application of MAKAUT is a bonafide record of work done by me under the guidance of Mrs. Rupa Saha.

BONAFIDE CERTIFICATE

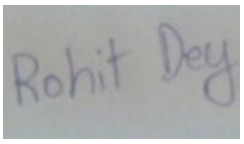
Certified that this project work was carried out under the supervision development of a feature-rich, "Face Recognition Using Python" is the bonafide work of

Student Name: ***Signature:***

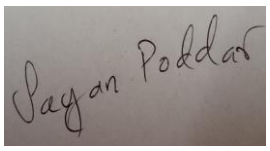
Sounak Sarkar (30601219083)



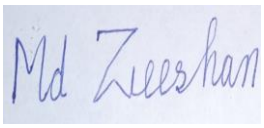
Rohit Dey (30601219081)



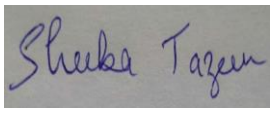
Sayan Poddar (30601219082)



Md. Zeeshan (30601219011)



Sheeba Tazeen (30601219003)



ACKNOWLEDGEMENT

We are over helmed in all humbleness and gratefulness to acknowledge our depth to all those who have helped us to put these ideas, well above the level of simplicity and into something concrete.

We would like to express our special thanks of gratitude to our professor and mentor Mrs. Rupa Saha as well as our principal who gave us the golden opportunity to do this wonderful project to

create a face detection program, which also helped us in doing a lot of Research and we came to know about so many new things. We are really thankful to them.

Any attempt at any level can ‘t be satisfactorily completed without the support and guidance of our mentors and group members.

We would like to thank our parents who helped me a lot in gathering different information, collecting data and guiding me from time to time in making this project, despite of their busy schedules, they gave me different ideas in making this project unique.

Sounak Sarkar

Rohit Dey

Sayan Poddar

Md. Zeeshan

Sheeba Tazeen

LIST OF TABLES

1. ABSTRACT

A face recognition system is one of the biometric information processes, its applicability is easier and working range is larger than others, i.e.; fingerprint, iris scanning, signature, etc. A face

recognition system is designed, implemented and tested at Atilim University, Mechatronics Engineering Department. The system uses a combination of techniques in two topics; face detection and recognition. The face detection is performed on live acquired images without any application field in mind. Processes utilized in the system are white balance correction, skin like region segmentation, facial feature extraction and face image extraction on a face candidate. Then a face classification method that uses Feedforward Neural Network is integrated in the system. The system is tested with a database generated in the laboratory with 26 people. The tested system has acceptable performance to recognize faces within intended limits. System is also capable of detecting and recognizing multiple faces in live acquired images.

2. INTRODUCTION

In computer vision, one essential problem we are trying to figure out is to automatically detect objects in an image without human intervention. Face detection can be thought of as such a problem where we detect human faces in an image. There may be slight differences in the faces of humans but overall, it is safe to say that there are certain features that are associated with all the human faces. There are various face detection algorithms but Viola-Jones Algorithm is one of the oldest methods that is also used today and we will use the same later in the article. You can go through the Viola-Jones Algorithm after completing this article as I'll link it at the end of this article.

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection can have very useful applications. The most successful application of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

2.1. BACKGROUND OF THE STUDY:

Importance of face recognition systems have speed up in the last few decades. A face recognition system is one of the biometric information processing. Applicability is easier and working range is larger than other biometric information processing, i.e.; fingerprint, iris scanning, signature, etc. A face recognition system is designed, implemented and tested in this thesis study. The system utilizes a combination of techniques in two topics; face detection and recognition. The face detection is performed on live acquired images without any application field in mind. Processes utilized in the system are white balance correction, skin like region segmentation, facial feature extraction and face image extraction on a face candidate. Then a face classification method that uses Feedforward Neural Network is integrated in the system. The system is tested with a database generated in the laboratory, with 26 people. The tested system has acceptable performance to recognize faces within

intended limits. System also capable of detecting and recognizing multiple faces in live acquired images.

2.2. NEED & SIGNIFICANCE OF THE STUDY:

Today, we are inundated with data of all kinds, but the plethora of photo and video data available provides the dataset required to make facial recognition technology work. Facial recognition systems analyze the visual data and millions of images and videos created by high-quality Closed-Circuit Television (CCTV) cameras installed in our cities for security, smartphones, social media, and other online activity. Machine learning and artificial intelligence capabilities in the software map distinguishable facial features mathematically, look for patterns in the visual data, and compare new images and videos to other data stored in facial recognition databases to determine identity.

2.3. STATEMENT OF THE PROBLEM:

Face recognition research still face challenge in some specific domains such as pose and illumination changes. Although numerous methods have been proposed to solve such problems and have demonstrated significant promise, the difficulties still remain. For these reasons, the matching performance in current automatic face recognition is relatively poor compared to that achieved in fingerprint and iris matching, yet it may be the only available measuring tool for an application. Error rates of 2-25% are typical. It is effective if combined with other biometric measurements.

2.4. OBJECTIVE OF THE STUDY:

The objective of face recognition is, from the incoming image, to find a series of data of the same face in a set of training images in a database. The great difficulty is ensuring that this process is carried out in real-time, something that is not available to all biometric facial recognition software providers.

The facial recognition process can perform two variants depending on when it is performed:

The one in which, for the first time, a facial recognition system addresses a face to register it and associate it with an identity, in such a way that it is recorded in the system. This process is also known as digital onboarding with facial recognition.

The variant in which the user is authenticated, prior to being registered. In this process, the incoming data from the camera is crossed with the existing data in the database. If the face matches an already registered identity, the user is granted access to the system with his credentials.

2.5. ORGANIZATION OF THE STUDY:

We will have to create three files; one will take our dataset and extract face embedding for each face using “dlib” library in Python. Next, we will save these embedding in a file.

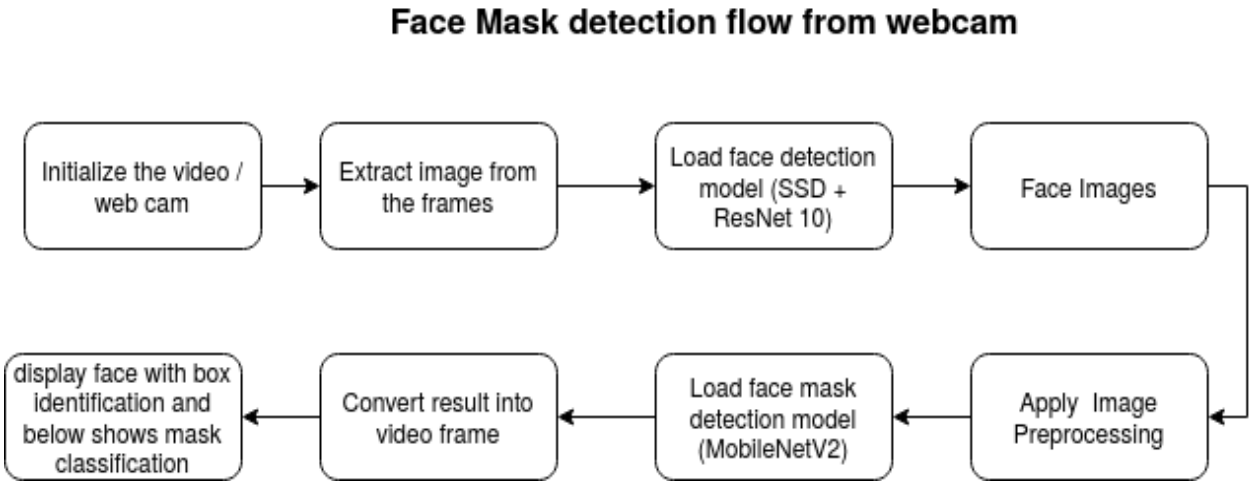
In the next file we will compare the faces with the existing the recognize faces in images and next we will do the same but recognize faces in live webcam feed.

3. RESEARCH METHODOLOGY:

3.1. OBJECTIVES:

The objective of face recognition is, from the incoming image, to find a series of data of the same face in a set of training images in a database. The great difficulty is ensuring that this process is carried out in real-time, something that is not available to all biometric facial recognition software providers.

3.2. RESEARCH DESIGN:



3.3. SOFTWARE REQUIREMENT:

1) This project uses the following tools for its implementation:
2) Python: The programming language used to implement this Project.
3) Jupyter notebook: Used to provide an interactive environment for python, for the implementation of this project.

4) OpenCV: Is an image and video processing library and is used for image and video analysis, like facial detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.
5) dlib: The dlib library, maintained by Davis King, contains our implementation of "deep metric learning" which is used to construct our face embeddings used for the actual recognition process.
6) Face Recognition: The face_recognition library, created by Adam Geitgey, wraps around dlib's facial recognition functionality, and this library is super easy to work with and we will be using this in our code. Remember to install dlib library first before you install face_recognition.

4. FACE RECOGNITION IN WEBCAM FEED:

 # IMPORT LIBRARIES NECESSARY FOR THIS PROJECT:

In [1]:
```
from imutils import paths
import face_recognition
import pickle
import cv2
import os
```

 # GET PATHS OF EACH FILE IN FOLDER IMAGES:

 # IMAGES HERE CONTAINS MY DATA (FOLDERS OF VARIOUS PERSONS):

In [2]:
```
#get paths of each file in folder named Images
#Images here contains my data(folders of various persons)
imagePaths = list(paths.list_images('Images'))
knownEncodings = []
knownNames = []
```

 # LOOP OVER THE IMAGE PATHS:

```
6  # loop over the image paths
7  for (i, imagePath) in enumerate(imagePaths):
```

 # EXTRACT THE PERSON'S NAME FROM THE IMAGE PATH:

 # LOAD THE INPUT IMAGE AND CONVERT IT FROM BGR (OPENCV ORDERING):

 # TO DLIB ORDERING (RGB):

```
8     # extract the person name from the image path
9     # load the input image and convert it from BGR (OpenCV ordering)
10    # to dlib ordering (RGB)
11    name = imagePath.split(os.path.sep)[-2]
12    image = cv2.imread(imagePath)
13    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

 # USE FACE_RECOGNITION TO LOCATE FACES:

```
14    #Use Face_recognition to locate faces
15    boxes = face_recognition.face_locations(rgb,model='hog')
16    # compute the facial embedding for the face
17    encodings = face_recognition.face_encodings(rgb, boxes)
18    # loop over the encodings
19    for encoding in encodings:
20        knownEncodings.append(encoding)
21        knownNames.append(name)
```

 # SAVE ENCODINGS ALONG WITH THEIR NAMES IN DICTIONARY DATA:

In [3]:
```
#save emcodings along with their names in dictionary data
data = {"encodings": knownEncodings, "names": knownNames}
#use pickle to save data into a file for later use
f = open("face_enc", "wb")
f.write(pickle.dumps(data))
f.close()
```

 # FIND PATH OF XML FILE CONTAINING HAARCASCADE FILE:

In [5]:
```
#find path of xml file containing haarcascade file
cascPathface = os.path.dirname(
 cv2.__file__) + "/data/haarcascade_frontalface_alt2.xml"
# load the harcaascade in the cascade classifier
faceCascade = cv2.CascadeClassifier(cascPathface)
# load the known faces and embeddings saved in last file
data = pickle.loads(open('face_enc', "rb").read())
```

 # LOOP OVER FRAMES FROM THE VIDEO FILE STREAM:

 # GRAB THE FRAME FROM THE THREADED VIDEO STREAM:

```
In [ ]:
print("Streaming started")
video_capture = cv2.VideoCapture(0)
# loop over frames from the video file stream
while True:
    # grab the frame from the threaded video stream
    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.1,
                                         minNeighbors=5,
                                         minSize=(60, 60),
                                         flags=cv2.CASCADE_SCALE_IMAGE)
```

 # CONVERT THE INPUT FRAME FROM BGR TO RGB:

```
    # convert the input frame from BGR to RGB
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # the facial embeddings for face in input
    encodings = face_recognition.face_encodings(rgb)
    names = []
```

 # COMPARE ENCODINGS WITH ENCODINGS IN DATA["ENCODINGS"]:

 # MATCHES CONTAIN ARRAY WITH BOOLEAN VALUES AND TRUE FOR THE EMBEDDINGS IT MATCHES CLOSELY AND FALSE FOR REST:

```
    # loop over the facial embeddings incase
    # we have multiple embeddings for multiple fcaes
    for encoding in encodings:
      #Compare encodings with encodings in data["encodings"]
      #Matches contain array with boolean values and True for the embeddings it matches closely
      #and False for rest
       matches = face_recognition.compare_faces(data["encodings"],
        encoding)
```

 # SET NAME = UNKNOWN IF NO ENCODING MATCHES:

 # CHECK TO SEE IF WE HAVE FOUND A MATCH:

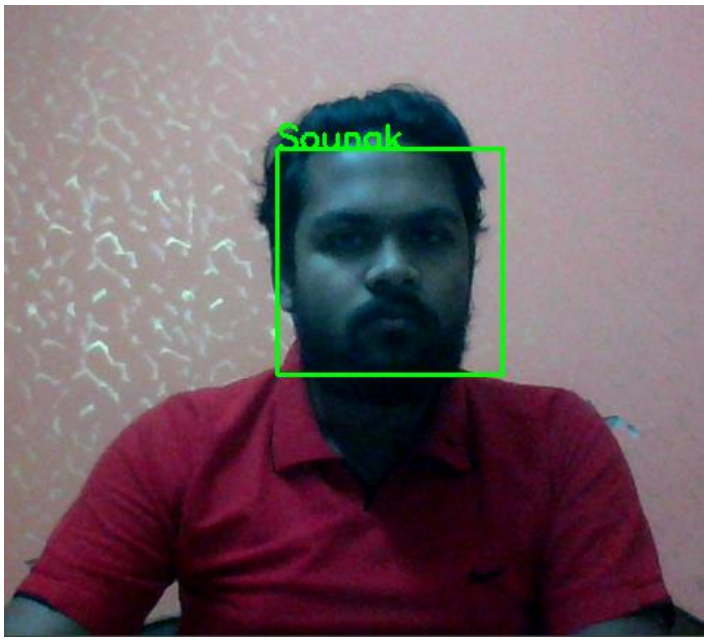
```
        #set name =inknown if no encoding matches
        name = "Unknown"
        # check to see if we have found a match
        if True in matches:
            #Find positions at which we get True and store them
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}
            # loop over the matched indexes and maintain a count for
            # each recognized face face
            for i in matchedIdxs:
                #Check the names at respective indexes we stored in matchedIdxs
                name = data["names"][i]
                #increase count for the name we got
                counts[name] = counts.get(name, 0) + 1
            #set name which has highest count
            name = max(counts, key=counts.get)
```

 # UPDATE THE LIST OF NAMES:

```
        # update the list of names
        names.append(name)
        # loop over the recognized faces
        for ((x, y, w, h), name) in zip(faces, names):
            # rescale the face coordinates
            # draw the predicted face name on the image
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(frame, name, (x, y), cv2.FONT_HERSHEY_SIMPLEX,
             0.75, (0, 255, 0), 2)
    cv2.imshow("Frame", frame)
    if cv2.waitKey(1) &0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

<u>OUTPUT:</u>

In the code above we have used haar cascade to detect faces.



5. CONCLUSION

We can conclude that:

Face recognition is an emerging technology that can provide many benefits. Face recognition can save resources and time, and even generate new income streams, for companies that implement it right.

Face recognition, also called identification involves comparing one input image to all images in an image library in order to determine who the input image belongs to. Or if it does not belong to the database at all.

An example of a face recognition system would be submitting a picture of a suspect to the police database to produce a match, such as they did at Panama's Tocumen airport.

6. REFERENCES

- https://www.researchgate.net/publication/262875649_Design_of_a_Face_Recognition_System
- https://face-recognition.readthedocs.io/en/latest/readme.html
- https://machinelearning.org.in/face-detection-from-webcam-in-python-using-opencv/