DIY Project

Computer Vision

Attendance monitoring system(Team H)

Team Members

Name	Roll number
Annu Kumari	20MF3IM03
Manish Kumar Roy	20AE30031
Saamarth Singh	20MF10029
Jai Marothiya	20ME30023
Kunal Kumar	20CE30015

Introduction

Monitoring attendance is a very important aspect in any environment where attendance is crucial. However, many of the available attendance monitoring methods are time taking, disturbing and it requires more human intervention. The proposed system is aimed at developing a less disturbing, cost-effective, and more efficient automated student attendance management system**.** We are using a camera which we will use to take pictures and send that image to our app or python code .All the faces in the image will be recognised. A rectangle would be created around their face in the output image and it will be sent to a telegram group. Along with that a csv file will also be sent to the telegram group where the name of all persons in the image will be there.

Description of code

Libraries used

1.face_recognition: Face-Recognition library built using the “dlib” library, which is built using C++ Language.This library helps to recognize the face of a person with 99.38% Accuracy. We can install this library using the following code:

```
pip3 install face_recognition
```

It helps in finding faces in an image using the following code:

```
import face_recognition

image = face_recognition.load_image_file("test.jpg")

face_locations = face_recognition.face_locations(image)
```

We can use this in live stream videos to detect faces using OpenCV.

We can also use this to compare faces in two images .For this we have compare_faces(). The following code can be used to compare faces:

```
import face_recognition

known_image = face_recognition.load_image_file("train.jpg")

unknown_image = face_recognition.load_image_file("test.jpg")

train_encoding = face_recognition.face_encodings(known_image)[0]

test_encoding = face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([train_encoding],
test_encoding)
```

2.cv2 :(import face_recognition)It is a library of Python bindings designed to solve computer vision problems.

3.numpy:(Numerical Python)Numpy provides a large set of numeric datatypes that can be used to construct arrays. At the time of Array creation, Numpy tries to guess a datatype, but functions that construct arrays usually also include an optional argument to explicitly specify the datatype.

4.os : The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

5. requests : Requests library is one of the integral part of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scrapping, requests is must to be learned for proceeding further with these technologies. When one makes a request to a URI, it returns a response. Python requests provides inbuilt functionalities for managing both the request and response.

Training

We created a list of our training images and used the following code for training:

```
for _ in images:

    image = fr.load_image_file(_)

    image_path =  _

    encoding = fr.face_encodings(image)[0]

    known_name_encodings.append(encoding)

    Known_names.append (os.path.splitext(os.path.basename(image_path))[0].
capitalize())
```

Here fr denotes face_recognition library.

Testing

For testing we found encodings for the test image using the following code:

```
test_image = "test.jpg"

image = cv2.imread(test_image)

face_locations = fr.face_locations(image)

face_encodings = fr.face_encodings(image, face_locations)
```

Comparing

Now for comparing our our test image encodings with our training encodings we used the following code:

```
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    matches = fr.compare_faces(known_name_encodings, *face_encoding)
    name = ""
    face_distances = fr.face_distance(known_name_encodings, face_encoding)
    best_match = np.argmin(face_distances)
    if matches[best_match]:
        name = known_names[best_match]
    cv2.rectangle(image, (left, top), (right, bottom), (0, 0, 255), 2)
    cv2.rectangle(image, (left, bottom - 15), (right, bottom), (0, 0, 255), cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    f.write(name)
    f.write("\n ")
    cv2.putText(image, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)
```

In which we are making a dictionary with our test image encodings and test image locations and then comparing those with image encodings .If the code is finding a match then it is putting a rectangle around the face in the image and also writing the name of the person in the rectangle.

Dataset

Training dataset:

1.Chandler

2.Joey

3.monica

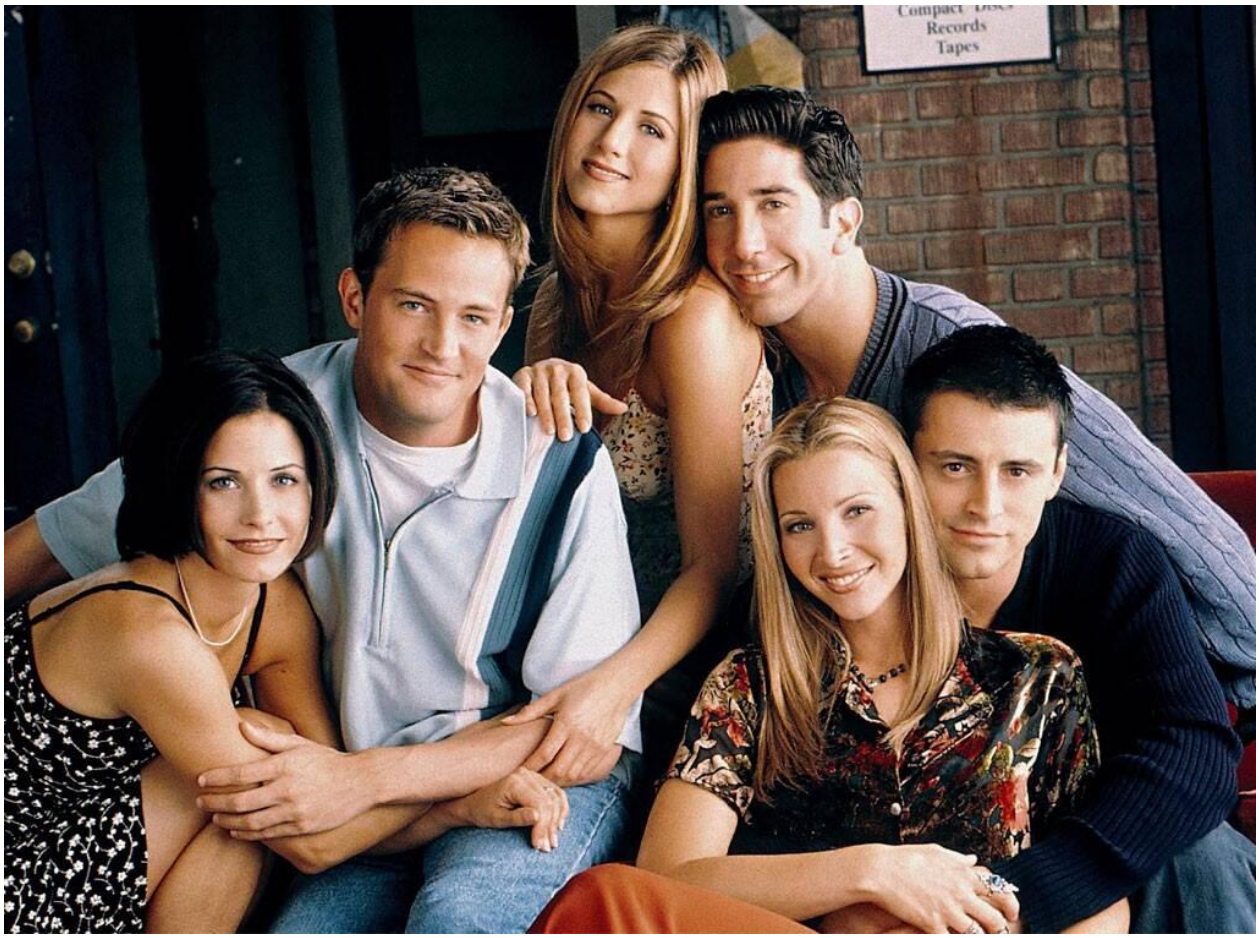
4.phoebe

5.rachel

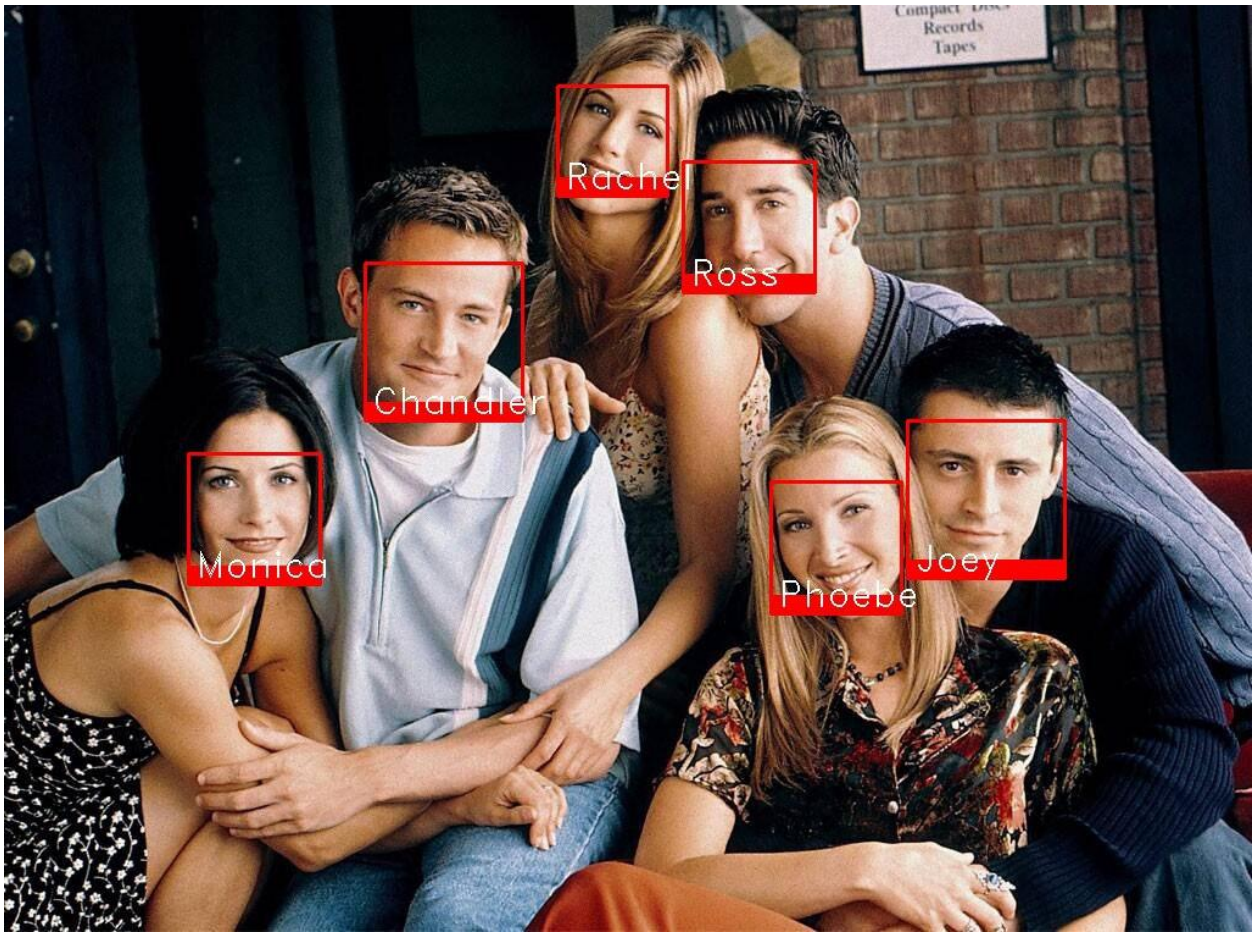
6.ross

These are the names of the people in the image.

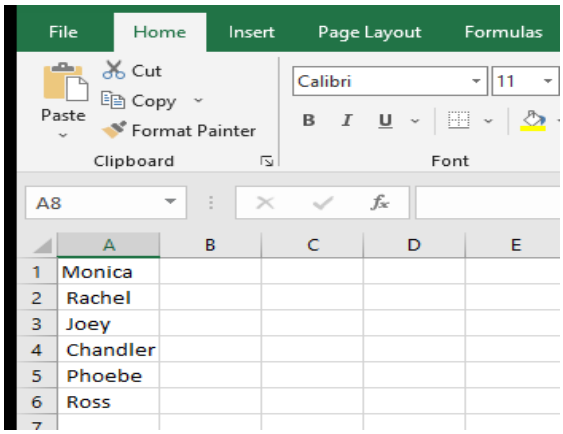
Test dataset:



Result



So this is the resulting image.Along with this we have created a csv file also with our code in which name of people in the image are there.



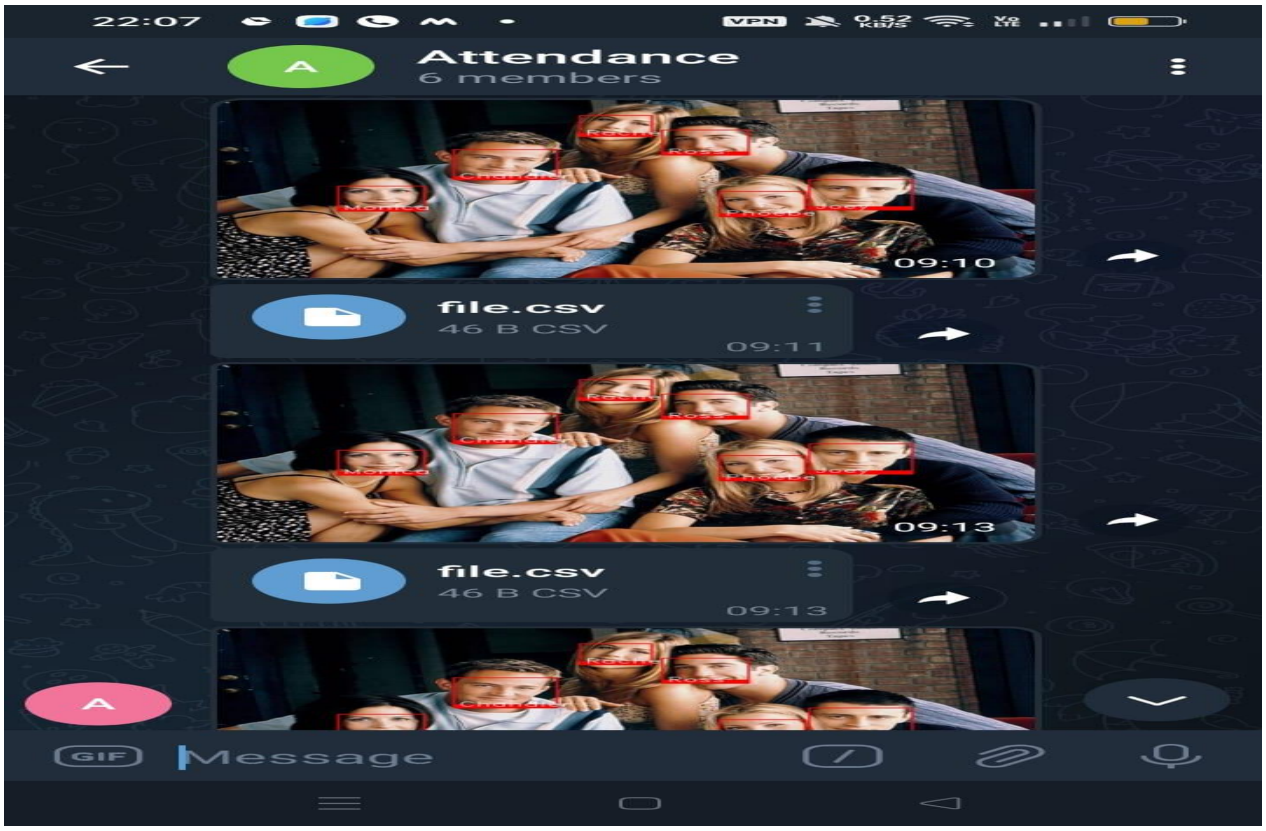
Sending the output image file and resulting csv file to Telegram

We have created a telegram group in which we are supposed to add all the members who need the image file as well as csv file .We have created a bot and added that bot to our telegram group. Now with the help of our code and bot we are able to send the image file and the csv file to the telegram group. Code to do so is:

```
files={'photo':open("output.jpg",'rb')}

resp=requests.post('https://api.telegram.org/bot5548975991:<token>/sendPhoto?chat_id=-624915028',files=files)

csv={'document':open("file.csv",'rb')}

resp=requests.post('https://api.telegram.org/bot5548975991:<token>/sendDocument?chat_id=-624915028',files=csv)
```

Replace the <token> in the code with your own token.

Some of the images of our telegram setup are given below:



Hardware Modeling

Some of the pictures of our hardware model are:

The End