Delhi Technological University

Department of Applied Mathematics

M.Sc Mathematics

Face Recognition In Python

Submitted By :
Vidipt Vashist
2K20/MSCMAT/33
Bharat Varshney
2K20/MSCMAT/07

Submitted To :
Prof. Goonjan JAIN

Face Recognition And Attendance Project In Python

Vidipt Vashist (2K20/MSCMAT/33)
Bharat Varshney (2K20/MSCMAT/07)

April 12, 2021

Contents

1 INTRODUCTION

- It involved building a system for face detection and face recognition using several classifiers available in the **open computer vision library(OpenCV)**.
- Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analysed at the same time.
- The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision "whose face is it ? ", using an image saved already.

2 HISTORY

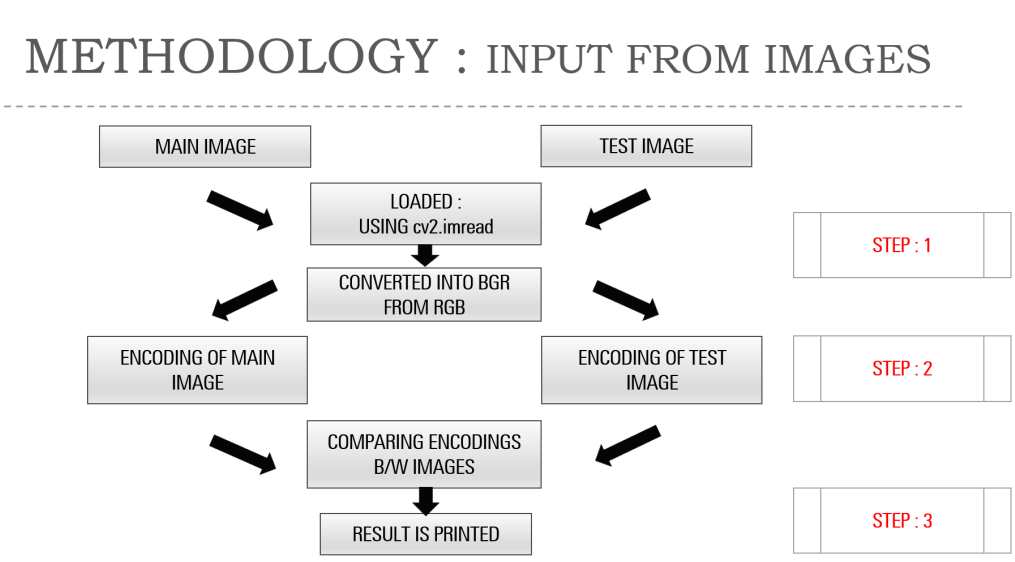
Face recognition began as early as 1977 with the first automated system being introduced By Kanade using a feature vector of human faces . In 1983, Sirovich and Kirby introduced the principal component analysis(PCA) for feature extraction . Using PCA, Turk and Pentland Eigenface was developed in 1991 and is considered a major milestone in technology . Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms(LBPH) . In 1996 Fisherface was developed using Linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigenface method . Viola and Jones introduced a face detection technique using HAAR cascades and ADABoost . In 2007, A face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes . In This project, HAAR cascades are used for face detection and Eigenface, Fisherface and LBPH are used for face recognition

3 METHODOLOGY

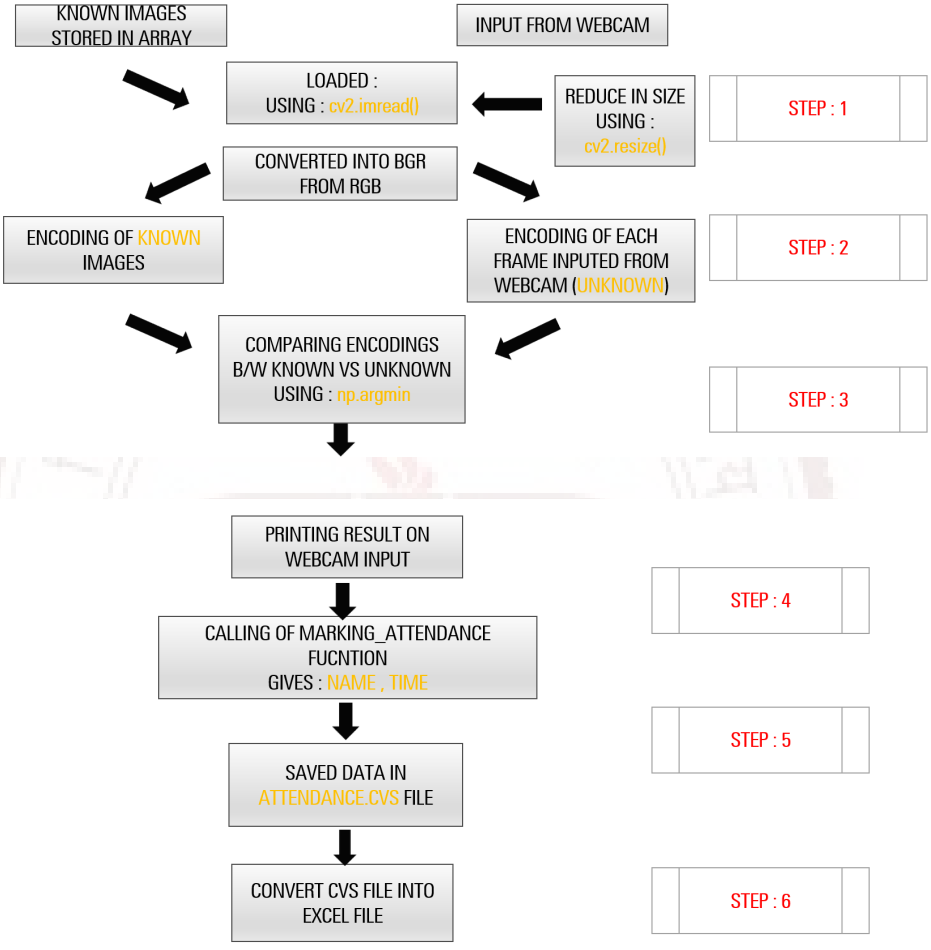
The project was coded in Python using **PYCharm IDE**.

For this project three steps are implemented independently

1. Collecting images IDs or **Loading Images**
2. Extracting unique features, classifying them and storing or **Cascading and storing**
3. Matching features of an input image to the features in the saved and predict identity or **Comparing Known Encoding vs Unknown Encoding**

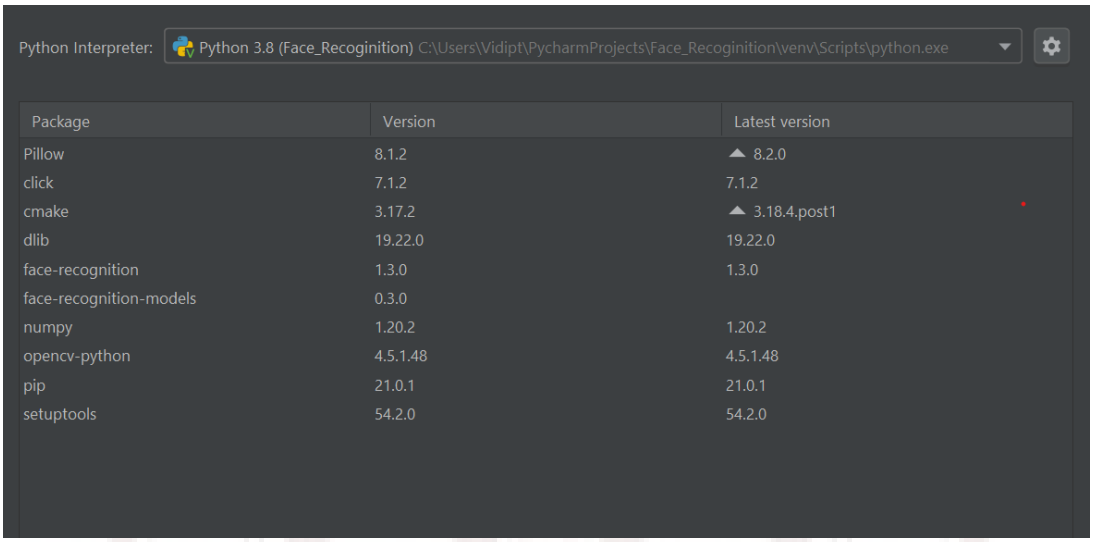


METHODOLOGY : ATTENDNACE PROJECT



4 LIBRARY IMPORTED

Followings are the library used in coding :



4.1 OPENCV

It is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human

4.2 NUMPY

NumPy is a Python library used for working with arrays. I NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

4.3 FACE RECOGNITION

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. This also provides a simple face recognition command line tool that lets you do face recognition on a folder of images from the command line.

4.4 OS

Miscellaneous operating system interfaces This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

4.5 GLOB

In Python, the glob module is used to retrieve files/pathnames matching a specified pattern. The pattern rules of glob follow standard Unix path expansion rules. It is also predicted that according to benchmarks it is faster than other methods to match pathnames in directories. With glob, we can also use wildcards (”*, ?, [ranges]) apart from exact string search to make path retrieval more simple and convenient.

5 CODE

```
while True :

print("Select : ")
print("1.Face Recognition Using Images")
print("2.Marking Attendance Using Webcam ")
print("3.Exit")
number  = int(input("Enter Number : "))
```

```
8
9  if number == 1:
10 import cv2
11 import face_recognition
12
13 # LOADING AND CONVERTING INTO RGB FROM BGR
14 imgElon = face_recognition.load_image_file('imagesAtttendance/Elon
       musk.jpg')
15 imgElon = cv2.cvtColor(imgElon, cv2.COLOR_BGR2RGB)
16
17 imgTest = face_recognition.load_image_file('ImagesBasic/Bill Gates.
       jpg')
18 imgTest = cv2.cvtColor(imgTest, cv2.COLOR_BGR2RGB)
19
20 # FINDING FACE IN IMAGE AND FINDING ENCODING
21 faceLoc = face_recognition.face_locations(imgElon)[0]
22 encodeElon = face_recognition.face_encodings(imgElon)[0]
23 cv2.rectangle(imgElon, (faceLoc[3], faceLoc[0]), (faceLoc[1],
       faceLoc[2]), (255, 0, 255), 2)
24
25 faceLocTest = face_recognition.face_locations(imgTest)[0]
26 encodeElonTest = face_recognition.face_encodings(imgTest)[0]
27 cv2.rectangle(imgTest, (faceLoc[3], faceLoc[0]), (faceLoc[1],
       faceLoc[2]), (255, 0, 255), 2)
28
29 # COMPARING THE FACE AND FINDING DIATANCE B/W THEM
30 results = face_recognition.compare_faces([encodeElon],
       encodeElonTest)
31 faceDis = face_recognition.face_distance([encodeElon],
       encodeElonTest)
32
33 print(results)
34 print(faceDis)
35
36 # SHOW RESULT ON TEST IMAGE
37 cv2.putText(imgTest, f'{results} {round(faceDis[0], 2)}', (50, 50),
       cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
38
39 cv2.imshow('Main_Image', imgElon)
40 cv2.imshow('Test_Image', imgTest)
41
42 cv2.waitKey(0)
43
44 elif number == 2:
45 import cv2
46 import numpy as np
47 import face_recognition
48 import os
49 import glob
```

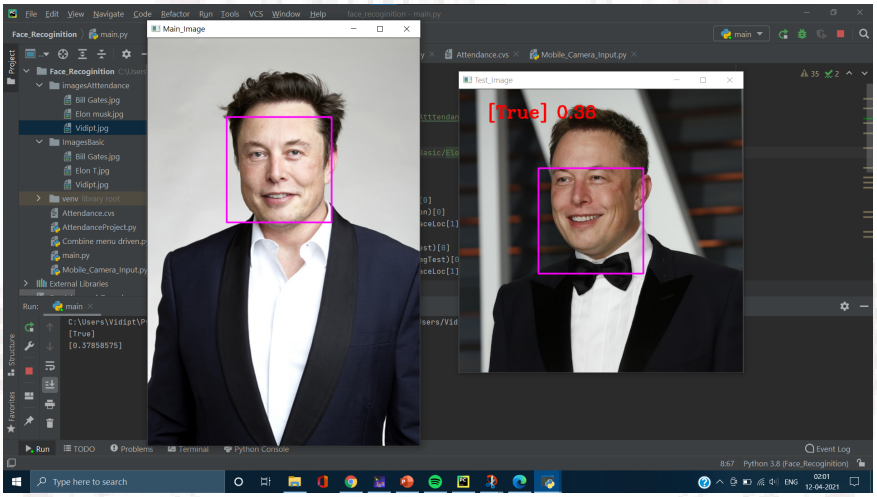
```
50
51 # USED IN MARKING ATTENDANCE
52 from datetime import datetime
53
54 path = 'ImagesAtttendance'
55 images = []
56 classNames = []
57 curImg = []
58
59 # RETURN NAMES OF ENTRIES IN DIRECTORY : ( ImagesAtttendance is this
       case ) I
60 mylist = os.listdir(path)
61 print(mylist)
62
63 # LOAD IMAGE : ( SPECIFLY USED GLOB FOR LOADING AS CV2,IMREAD UNABLE
       TO WORK ) : FOUND ON STACKEXCHANGE
64 for cl in mylist:
65 images = [cv2.imread(file) for file in glob.glob('ImagesAtttendance
      /*.jpg')]
66 curImg.append(images)
67 classNames.append(os.path.splitext(cl)[0])
68 print(classNames)
69
70
71 # FUCNTION CONVERTING BRG TO RGB AND ENCODING
72 def findEncodings(images):
73 encodeList = []
74 for img in images:
75 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
76 encode = face_recognition.face_encodings(img)[0]
77 encodeList.append(encode)
78 return encodeList
79
80
81 # FUCNTION FOR MAKRING ATTENDANCE (TO BE SHOWN IN ATTENDANCE.CVS)
82 def markAttendance(name):
83 with open('Attendance.cvs', 'r+') as f:
84 myDataList = f.readlines()
85 print(myDataList)
86 nameList = []
87 for line in myDataList:
88 entry = line.split(',')
89 nameList.append(entry[0])
90 if name not in nameList:
91 now = datetime.now()
92 dtString = now.strftime('%H:%M:%S')
93 f.writelines(f'\n{name},{dtString}')
94
95
```

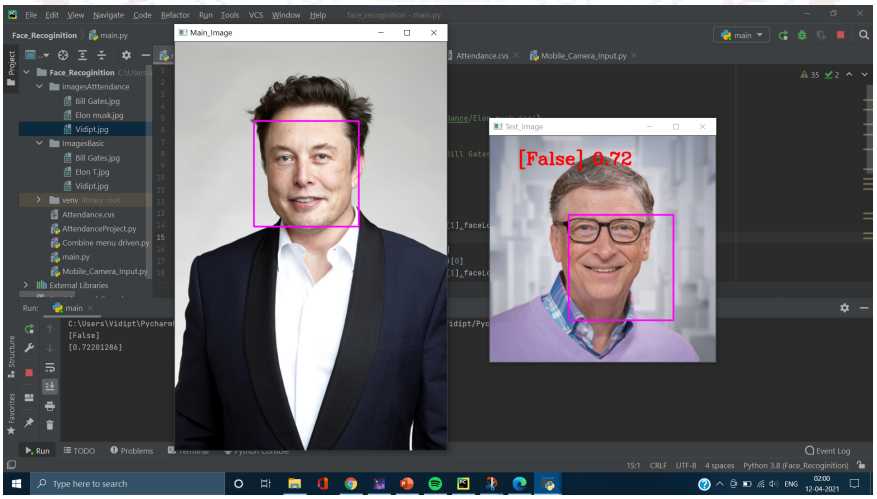
```
96 # TELLS ENCODING IS COMPLETED
97 encodeListKnown = findEncodings(images)
98 print('Encoding Complete ', len(encodeListKnown))
99
100 # CAPTURE IMAGE FROM WEBCAM
101 cap = cv2.VideoCapture(0)
102
103 while True:
104
105 # IMAGE CAPTURED IF REDUCED IN SIZE FOR FAST EXECUTION
106 success, img = cap.read()
107 imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
108 imgS = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
109
110 # IMAGES FROM WEBCAM IS ENCODED
111 faceCurFrame = face_recognition.face_locations(imgS)
112 encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)
113
114 # COMPARING IMAGES KNOW VS UNKNOWN(WEBCAM)
115 for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
116 matches = face_recognition.compare_faces(encodeListKnown, encodeFace
    )
117 faceDis = face_recognition.face_distance(encodeListKnown, encodeFace
    )
118 print(faceDis)
119 matchIndex = np.argmin(faceDis)
120
121 # PRINTING RESULT ON (WEBCAM INPUT)
122 if matches[matchIndex]:
123 name = classNames[matchIndex]
124 print(name)
125
126 y1, x2, y2, x1 = faceLoc
127
128 cv2.rectangle(img, (faceLoc[3], faceLoc[0]), (faceLoc[1], faceLoc
    [2]), (255, 0, 255), 2)
129 cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
130 cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX,
    1, (255, 255, 255), 2)
131
132 # CALLING OF ATTENDANCE FUNCTION FOR PRINTING NAME.
133 markAttendance(name)
134
135 cv2.imshow('webcam', img)
136 cv2.waitKey(1)
137
138 else :
139 break
140
```

```
141
142
143
```

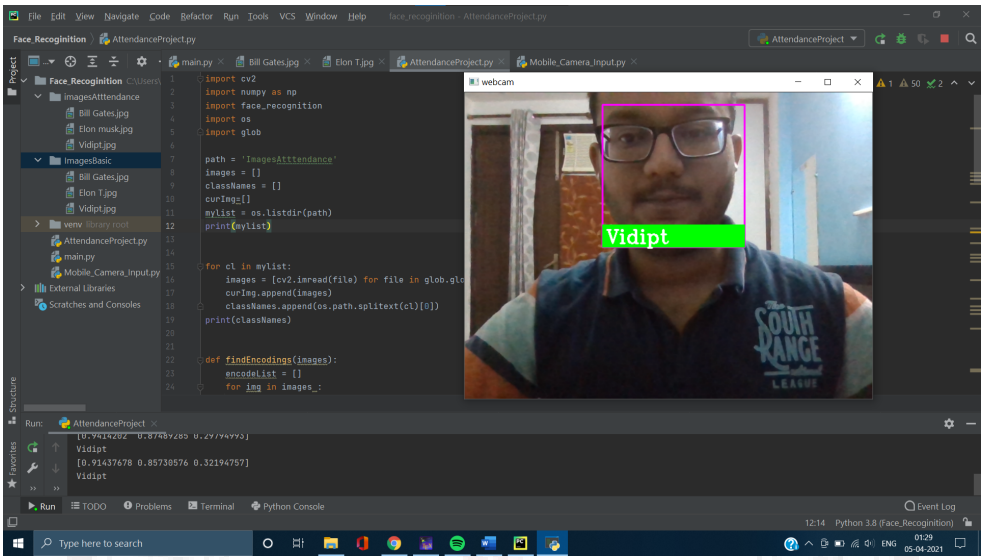
6 OUTPUT

6.1 Face Recogniton Using Input From Images



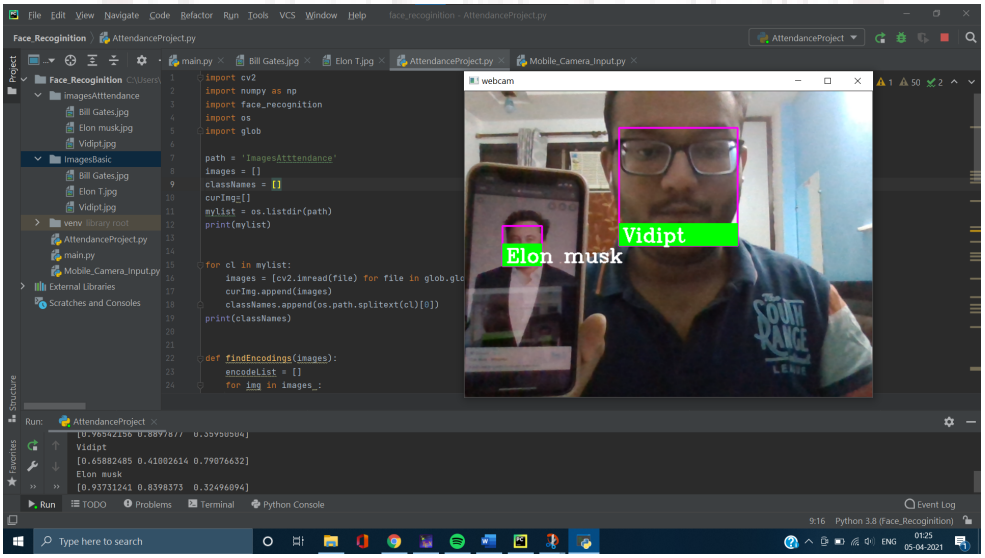


6.2 Attendance Using Face Recoginition

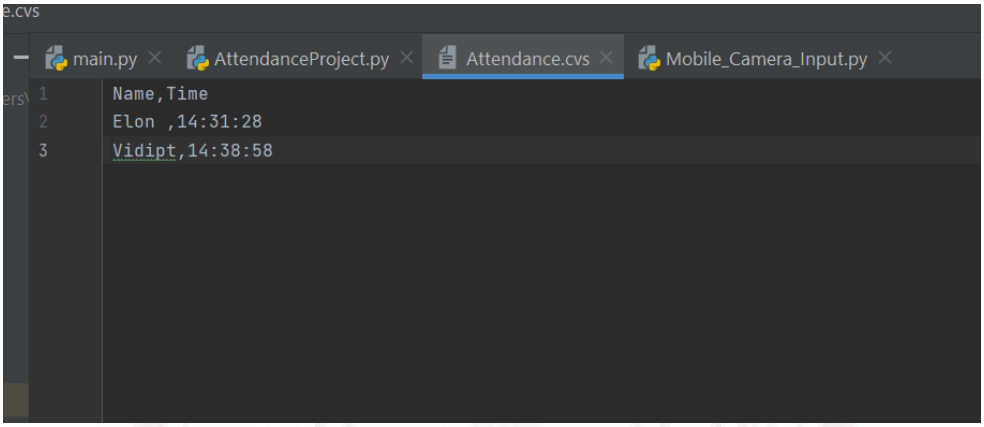


6.3 Results Of Comparing

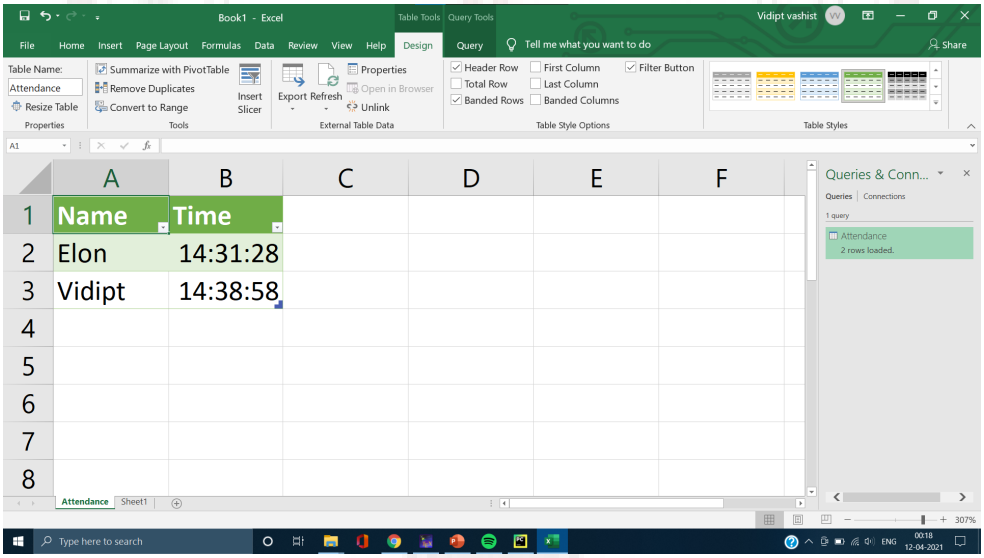
```
C:\Users\Vidipt\PycharmProjects\Face_Recoginition\venv\Scripts\python.exe C:/Users/Vidipt/PycharmProjects/face_recoginition/AttendanceProject.py
['Bill Gates.jpg', 'Elon musk.jpg', 'Vidipt.jpg']
['Bill Gates', 'Elon musk', 'Vidipt']
Encoding Complete  3
[0.94117458 0.84927747 0.32656679]
Vidipt
[0.90772781 0.85509047 0.32411795]
Vidipt
[0.91628655 0.83278238 0.33926461]
```



6.4 Attendance Marked in CVS file



6.5 CVS File Exported to Excel



7 CONCLUSION

This paper describes the mini-project for Face Recognition using opencv in python . Also,it explains the library used in the project and the methodology used. Real time video speed was satisfactory as well has a bit of noticeable frame lag. Considering that opencv can be implemented as a cost effective face recognition platform. An example is a system to identify known people for automatic Attendance.

8 REFERNCES

1. Takeo Kanade. Computer recognition of human faces, volume 47. Birkh¨auser Basel, 1977

2. https://opencv.org/

3. https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

4. https://www.copytrans.net/support/how-to-open-a-csv-file-in-excel/