**PROJECT REPORT**

# Attendance using Face Recognition

**Under the guidance of**

| | |
|---|---|
| Mr. Tushar Patnaik | Mr. Bhupendra Kumar |
| C-DAC, Noida | C-DAC, Noida |

**Submitted by**

Sahil Siddharth
(Trainee – Machine Learning)

C-DAC, Noida
May-July, 2019

सी डैक
CDAC

CENTER FOR DEVELOPMENT OF
ADVANCED COMPUTING

**July, 2019**

# CERTIFICATE

This is to certify that the summer internship project entitled Attendance Using Face Recognition has been prepared by Sahil Siddharth, a B.Tech student of The LNM Institute of Information Technology. The project consisted of preparing a software to mark attendance of students using face recognition technology. The project has been successfully completed during his internship from 15th May 2019 to 12th July 2019 at C-DAC Noida, under the supervision of Mr. Tushar Patnaik and Mr. Bhupendra Kumar.

The Project Report is hereby approved for submission.

**Mr. Tushar Patnaik**
(Guide)
**C-DAC, NOIDA**

# ACKNOWLEDGEMENT

I express my sincere thanks to **Mr. BHUPENDRA KUMAR** for his guidance and cooperation throughout the project. In spite of his busy schedule, he extended his full support with keen interest for discussion and valuable suggestion. He also provided priceless feedback which eventually led to the successful completion of this project.

I am also immensely thankful to **Mr TUSHAR PATNAIK** for his help and kind cooperation. I am also indebted to all faculty members and peers, who have directly and indirectly helped me in the process, for their kind cooperation and advice given to me from time to time.

**Sahil Siddharth**

Trainee – Machine Learning
May-July, 2019
C-DAC, Noida

## ABSTRACT

This project involves the design of a software to mark attendance of students using face recognition technology. It detects and recognises faces of attendees and marks their attendance by classifying the detected faces into one of the pre-existing classes in real time.

Although many such projects have been built on face recognition and it is among the most researched field of machine learning in these days. Learning from previously built models we designed this software to detect and recognize faces in live feed and mark their attendance.

As C-DAC Noida provides Academic qualifications with research, this project was built to digitize the attendance marking system. This project can be implemented through the classroom's camera to detect and recognize the face of students and mark their attendance. This project involves various libraries of python to detect and recognize faces in groups as well. Using this attendance can be marked daily when the class settles down and this program is ran for few minutes.

# INTRODUCTION

In this project we first detect faces in a given frame and make use of existing data on people's faces to classify the detected faces into one of those classes. The project is in two parts. The first part "face_setup" consists of the program to add a new class (new person) into our database. The second part "face_recog" consists of the program to first detect faces in a frame from live video feed and then classify them into one of the classes.

In "face_setup", the idea is to set a parameter, say *n*, which dictates the number of images of a person, we will feed into the dataset for training. It is presumed that during this step the only face in the frame will be of the person we wish to make the dataset for, else errors can be caused. We first detect the faces in the frame using feature extraction and then crop the faces and save them in a folder with the person's name.

In "face_recog", we initialise the number of pictures of each person we'll compare the detected face in the live feed with, say *n*. First we detect faces in the live feed and then we run each of the unknown faces against one picture each of all the existing classes in the dataset, until for a particular class *Z*, *n* matches are found. Then that unknown face is classified into that class *Z*. If a detected unknown face fails to have match *n* matches with any of the pre-existing classes, it is automatically assigned into the '*none'* class.

Finally, for every class that has been detected, the attendance variable is incremented. However, even if the same face is detected multiple times, the attendance variable is incremented only once, hence preventing overflow. This way, our project ensures attendance of students using face detection and face recognition technology.

# Technologies Used

## OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
Using OpenCV library, we can −

- ❑ Read and write images
- ❑ Capture and save videos
- ❑ Process images (filter, transform)
- ❑ Perform feature detection
- ❑ Detect specific objects such as faces, eyes, cars, in the videos or images.
- ❑ Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

## Dlib

Dlib is a general purpose cross-platform open source software library written in the C++ programming language. Its design is heavily influenced by ideas from design by contract and component-based software engineering. This means it is, first and foremost, a collection of independent software components, each accompanied by extensive documentation and thorough debugging modes.

Core to the development philosophy of dlib is a dedication to portability and ease of use. Therefore, all code in dlib is designed to be as portable as possible and similarly to not require a user to configure or install anything. To help achieve this, all platform specific code is confined inside the API wrappers. Everything else is either layered on top of those wrappers or is written in pure ISO standard C++. Currently the library is known to work on OS X, MS Windows, Linux, Solaris, the BSDs, and HP-UX. It should work on any POSIX platform but I haven't had the opportunity to test it on any others

## Face_recognition

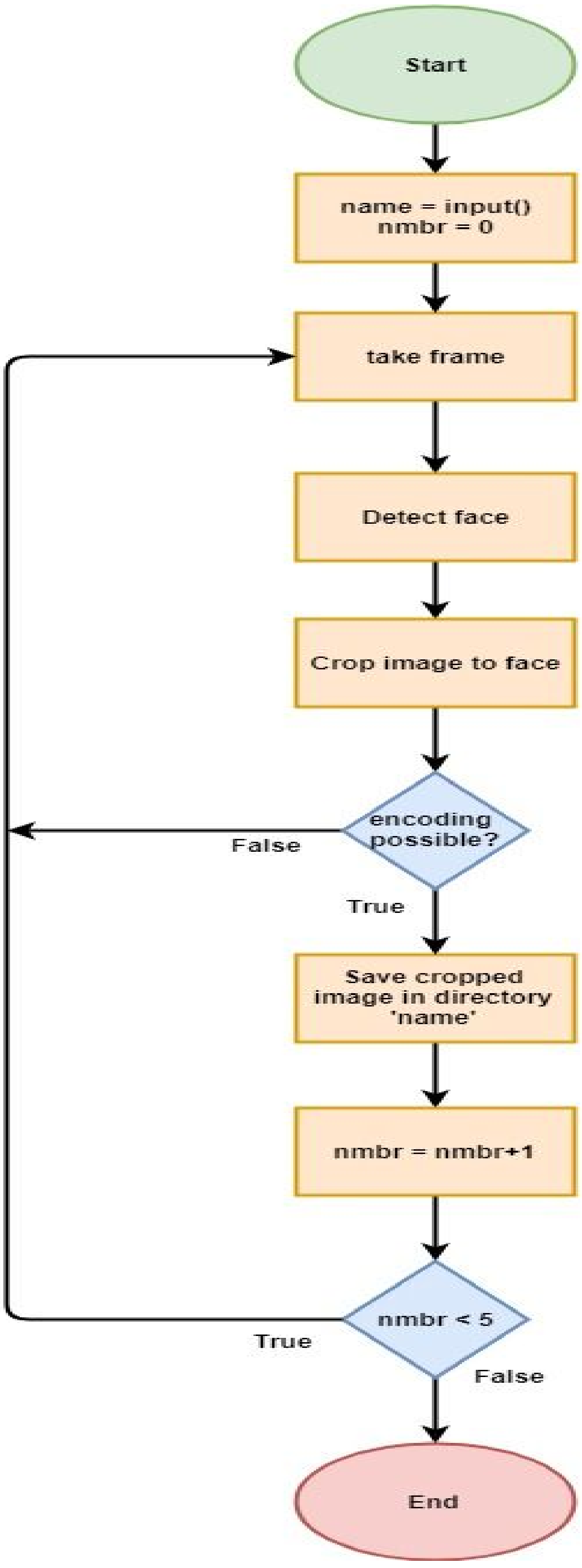
The face_recognition library empowers us to recognize and manipulate faces from Python or from the command line. It is built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the 'Labeled Faces in the Wild' benchmark. This also provides a simple face_recognition command line tool that lets us do face recognition on a folder of images from the command line!

Using the face_recognition library we can:

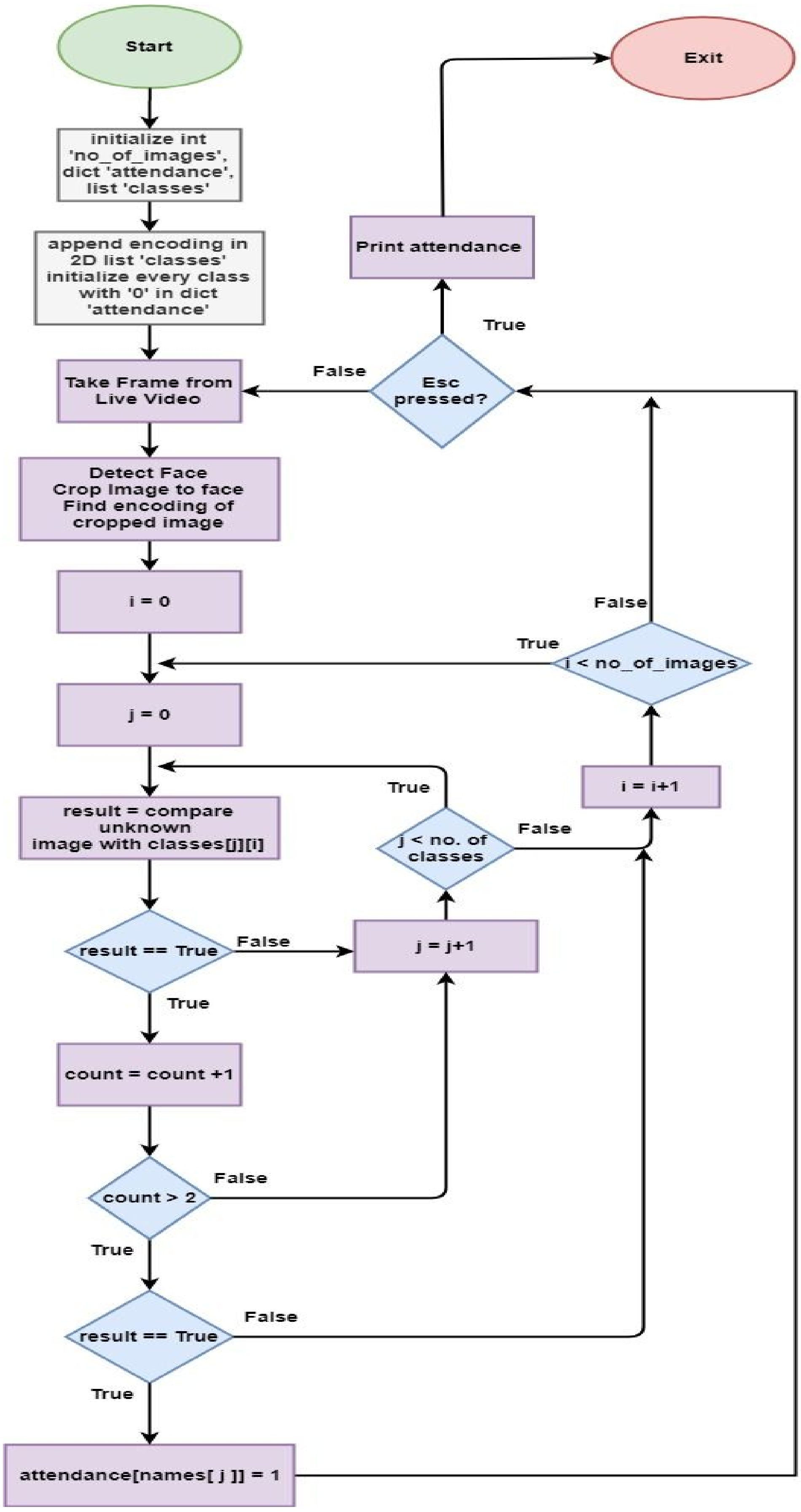
- ❏ Find faces in pictures
- ❏ Find and manipulate facial features in pictures
- ❏ Find and manipulate facial features in pictures

## Flow Diagrams

### A. Face Setup



## B. Face Recognition


Start
initialize int 'no_of_images', dict 'attendance', list 'classes'
append encoding in 2D list 'classes' initialize every class with '0' in dict 'attendance'
Take Frame from Live Video
Detect Face Crop Image to face Find encoding of cropped image
i = 0
j = 0
result = compare unknown image with classes[j][i]
result == True
True
count = count +1
count > 2
True
result == True
True
attendance[names[ j ]] = 1
False
j = j+1
False
j < no. of classes
True
False
i = i+1
i < no_of_images
True
False
Esc pressed?
False
True
Print attendance
Exit

# Source Code

## A. face_setup.py

```
import face_recognition
import cv2
import dlib
import numpy
import time
import os
from matplotlib import pyplot as plt

 ####################################################################################
##     A.face_setup : Attendance using Face Recognition Project
##            1. Detecting face
##            2. Cropping image
##            3. Saving it in directory
 ####################################################################################


cam = cv2.VideoCapture(0)                            # setting up camera

# Loading cascade files of frontalface, eyes, smile
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')


font = cv2.FONT_HERSHEY_SIMPLEX                      # setting font
nmbr=0                                               # intialising number of images to 0
name = input("enter name: ")                         # getting input name
newpath = r'cmpr_img/'+name                          # intialising location of new directory

if not os.path.exists(newpath):                      # make new path if path doesn't exist
    os.makedirs(newpath)
# loop runs if capturing has been initialized.
for i in range(50):

        ret, img = cam.read()                        # starting live video
             gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # converting img to
grayscale
        faces = face_cascade.detectMultiScale(gray, 1.3, 1) # detecting frontal face

        for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
                roi_gray = gray[y:y+h, x:x+w]
                roi_color = img[y:y+h, x:x+w]

                # Detects eyes and smile of different sizes in the input image
                eyes = eye_cascade.detectMultiScale(roi_gray)
                smile = smile_cascade.detectMultiScale(roi_gray, scaleFactor=1.9,
minNeighbors=4, minSize=(10, 10),flags = cv2.CASCADE_SCALE_IMAGE)

                #To draw a rectangle in eyes
                if len(eyes) != 0:
                    crop_img=img[y:y+h, x:x+w] # cropping image to the size of face
                    try:
                          # the 128-dimension face encoding for each face
                          encoding = face_recognition.face_encodings(crop_img)[0]

                    except (RuntimeError, TypeError, NameError, IndexError):
                        print('Incorrect format')
                        break
                    # saving cropped image to the directory
                    cv2.imwrite('cmpr_img\\'+name+'\\'+str(nmbr)+'img.jpg',crop_img)
                    nmbr+=1             # incrementing the number of images in the dir
                    print(nmbr)

        if nmbr>4:               # break before number of images in the dir exceeds 5
            break
        cv2.imshow('img',img)
        k = cv2.waitKey(30) & 0xff      # wait for Esc key to stop
        if k == 27:
             break

cam.release()

cv2.destroyAllWindows()
```

## B.face_recog.py

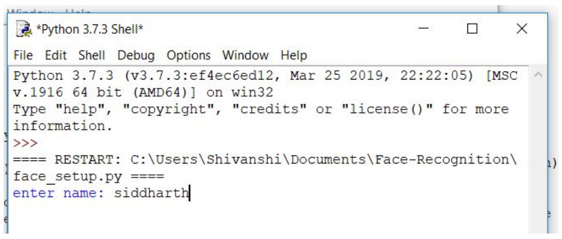
```
import face_recognition
import cv2
import dlib
import numpy as np
import os

 ##################################################################################
##    A.face_recog : Attendance using Face Recognition Project
##          1. Detecting face
##          2. Classifying faces by comparing unknown face with known faces
##          3. Marking attendance for checked classes
 ##################################################################################

cam = cv2.VideoCapture(0)       # setting up camera
names = os.listdir('cmpr_img')  # getting classes from directory
classes = []
nmbr_of_img = 5                 # number of images in a class
min_imgs = 2                    # minimum hits required

for i in names:
    encodings = []
    for j in range (nmbr_of_img):
        # loading images from dir
        img = face_recognition.load_image_file("cmpr_img/"+i+"/"+str(j)+"img.jpg")
        # 128-dimension face encoding for each face
        img_encoding = face_recognition.face_encodings(img)[0]
        encodings.append(img_encoding)  # appending encoded images in list
'encodings'
    classes.append(encodings)           # appending encodings in 2-D list 'classes'

attendance =  {key: 0 for key in names} # initialising attendance list
print(attendance)
detector = dlib.get_frontal_face_detector() # detecting frontal face
color_green = (0,255,0)                     # color of box
line_width = 3                              # width of box

# during live video feed
while True:
    ret_val, img = cam.read()               # getting frame from live feed
    dets = detector(img)                    # detecting frontal faces in frame

    for det in dets:
        # Crop Face from the whole image frame
        crop_img = img[det.left():det.right(), det.top():det.bottom()]

        # Create boundary box as(x, y, width, height) instead of four corner points
        bbox = (det.left()-5, det.top()-5, det.right() - det.left()+10 ,det.bottom()
- det.top()+10)
```

```
        # Detecting face in frame
        unknown_picture = face_recognition.face_locations(img)
        flag = 0                            # initialising flag
        count = [0]*len(names)              # initialising count
        try:
            unknown_face_encoding = 
face_recognition.face_encodings(img,unknown_picture)[0]

            for i in range (nmbr_of_img):
                # Draw rectangle around Face with color:Green and line width: 3pt
                cv2.rectangle(img,(det.left(), det.top()), (det.right(), 
det.bottom()), color_green, line_width)
                for j in range (len(names)):
                    # comparing unknown faces with known classes
                    results1 = face_recognition.compare_faces([classes[j][i]], 
unknown_face_encoding, tolerance = 0.45)
                    if results1[0] == True:         # checking matches
                          count[j] += 1             # increment count of that class

                # checking if count of that class exceeds min images
                if count[j] > min_imgs:
                    break
                # checking if matched and min images condition
                if results1[0] == True and count[j] > min_imgs:
                    flag = 1                        # setting flag to 1
                    attendance[names[j]] = 1        # marking present to that class
                    # putting name of that class beside box
                    cv2.putText(img, names[j], (det.left(), det.top()), 
cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2)
                    break

            if not flag:
                # setting that image to 'none' class
                cv2.putText(img, "none", (det.left(), det.top()), 
cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2)
        except (RuntimeError, TypeError, NameError, IndexError):
            print("no image found")

    cv2.imshow('LiveStream', img)                   # displaying live stream on screen

    # Exit if ESC pressed
    k = cv2.waitKey(1) & 0xff
    if k == 97:
       print(attendance)     # printing attendance if 'a' is pressed

    if k == 27 :
       print(attendance)       # printing final attendance when ESC is pressed
       break
```

## Output

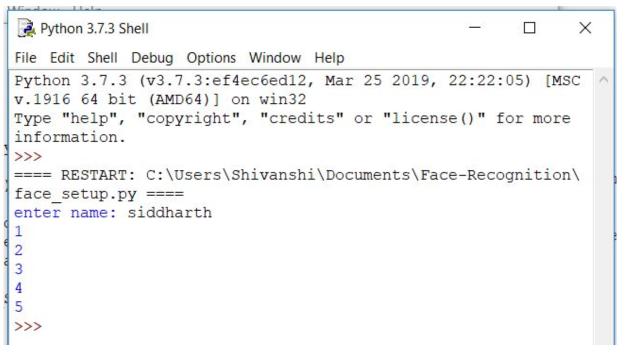
### A. face_setup

face_setup.py is run to add a new class to the database



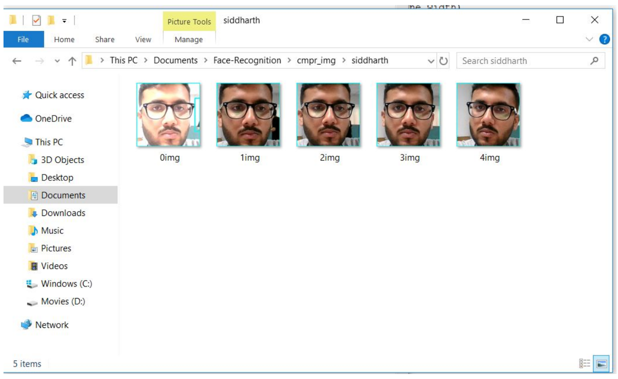
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC
v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\
face_setup.py ====
enter name: siddharth
```

Enter name of the class
Eg- "siddharth"



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC
v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\
face_setup.py ====
enter name: siddharth
1
2
3
4
5
>>>
```

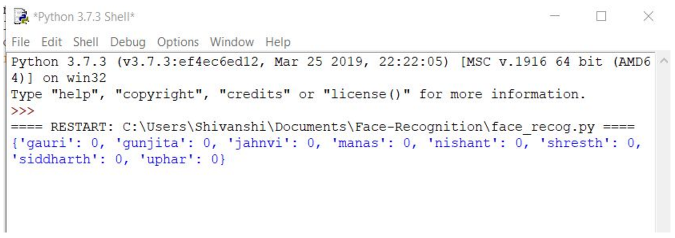
5 pictures for the class
"siddharth"
are taken



Directory by the name of the class
"siddharth" is made with
the 5 pictures named
0img, 1img… 4img

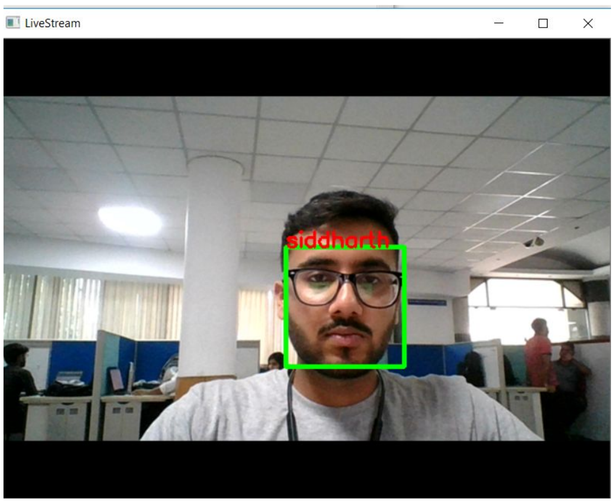
## B. face_recog

face_recog is run to detect and recognise faces from the live video feed

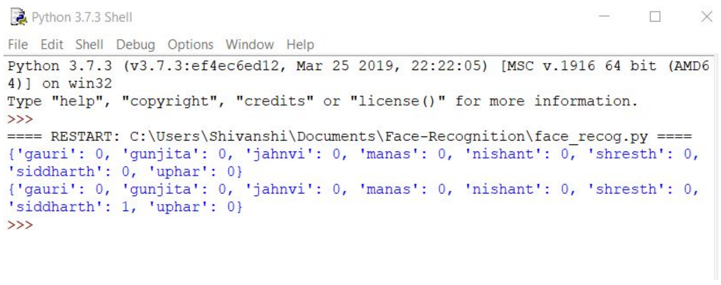


```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\face_recog.py ====
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 0, 'uphar': 0}
```

When face_recog is run it initially shows the attendance of all the students in the database



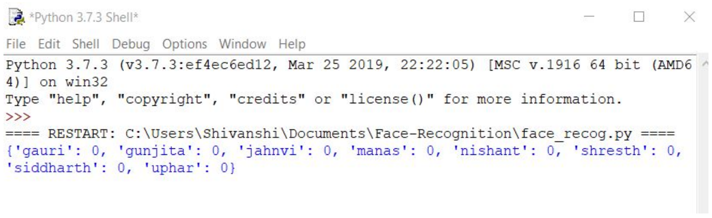
The face on the screen is detected and recognised



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\face_recog.py ====
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 0, 'uphar': 0}
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 1, 'uphar': 0}
>>>
```

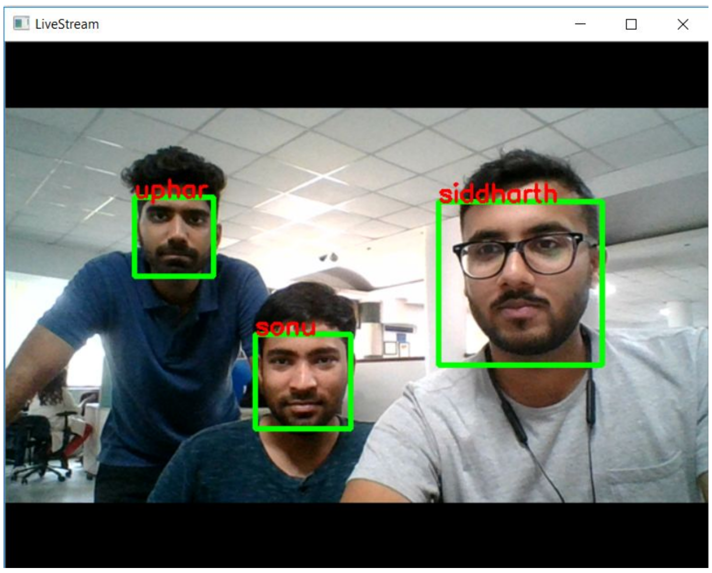
When ESC is pressed, the final attendance of all the students shown.
Note that only the attendance value of class “siddharth” is incremented.

## face_recog run for multiple faces in the same frame

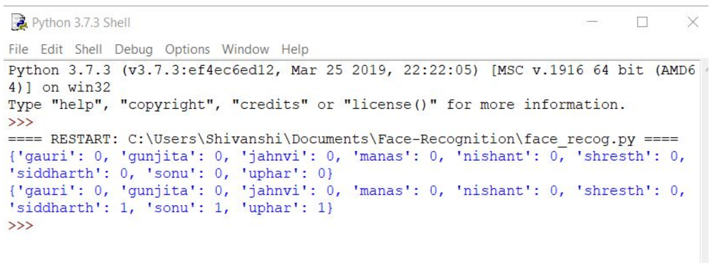


```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\face_recog.py ====
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 0, 'uphar': 0}
```

Initially no attendance is marked for any class



All three faces are correctly detected and recognised. Respective attendance is marked



```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\Shivanshi\Documents\Face-Recognition\face_recog.py ====
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 0, 'sonu': 0, 'uphar': 0}
{'gauri': 0, 'gunjita': 0, 'jahnvi': 0, 'manas': 0, 'nishant': 0, 'shresth': 0,
'siddharth': 1, 'sonu': 1, 'uphar': 1}
>>>
```

Attendance for the recognised classes is printed when the ESC key is pressed.

## Constraints

This project has been tested on 30 people and can be implemented on more than that. Constraints in this project are as follows:

- ❑ Number of images belonging to one class : 5
- ❑ Minimum number of images to be matched before marking attendance : 2
- ❑ Minimum dimensions of face that can be detected in live feed : 150*150 px
- ❑ The encoding of image saved should not be empty
- ❑ Tolerance of compare function of  face_recognition model is set to .45and can vary from 0 to 1 in decreasing level of strictness.

System constraints are as follows :

- ❑ Camera quality : 3 MP
- ❑ System RAM : 8 GB
- ❑ System Processor : Intel i7 8th Generation

## Result Analysis

This project is built to mark attendance using face detection and face recognition. It is divided into two parts, first where the data of faces is collected as known images, and second where the faces from the live feed is detected and compared to known images. After this the image is classified to its matched class and respective attendance is marked.

During testing we found out that lighting during the setting up of database is required to be proper and it takes some time to process the data as we compare each unknown image to minimum 3 images of the same class to mark it as present. Rest, all the trials were successful to mark the attendance for a person standing in front of the camera and the attendance of a person absent in front of the camera remained unmarked. We ran the program on different gender, race, and other factors, and found the results to be satisfactory.

It only requires a good processing speed to work smoothly to do all the computation in limited time. Modifications can be made by altering the strictness of 'face_recognition' model and increasing or decreasing the minimum number of images to cross check before marking the attendance.

The model however, is not perfect and can be improved in the future by using better hardware, like a better processor and better RAM. There exists a large scope of improvement for the software. A larger number of faces can be detected in a single frame by using a better camera and improving the minimum resolution on which faces can be detected.

By using better image quality, it would result in larger computations. Since the project is supposed to work for a live feed, this could cause problems in the live feed in the form of lag and frame freeze. A better algorithm would possibly lower the computations required and hence make the face recognition technology much more memory and time efficient and feasible.

## CONCLUSION

Through this project we first detected faces in a given frame and made use of existing data on people’s faces to classify the detected faces into one of those classes. The project is in two parts. The first part “face_setup” consists of the program to add a new class (new person) into our database. The second part “face_recog” consists of the program to first detect faces in a frame from live video feed and then classify them into one of the classes.

This project is build focusing just on detecting and recognizing faces to mark the attendance of the respective person. In the presence of proper camera for live feed and proper lighting, the project is functioning up to its standards.

In conclusion the program can classify images into respective classes, i.e names of people whose entries are made in database. This project can be implemented in classroom cameras with few modifications according to the requirements of the organization. The project is ready for use in a classroom environment and

This project was successfully designed and tested.

# REFERENCES

1. Gregory Koch, Richard Zemel and Ruslan Salakhutdinov. "**Siamese Neural Networks for One-shot Image Recognition.**" Department of Computer Science, University of Toronto. Toronto, Ontario, Canada https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
   Visited on : 29th May, 2019

2. OpenCV documentation: https://opencv.org/about/
   Visited on : 3rd June, 2019

3. Dlib documentation: http://dlib.net/
   Visited on : 3rd June, 2019

4. Face_recognition documentation:
   https://pypi.org/project/face_recognition/
   Visited on : 6th June, 2019

5. Erik Learned-Miller, Gary B. Huang, Aruni RoyChowdhury, Haoxiang Li, and Gang Hua. "**Labeled Faces in the Wild: A Survey.**" http://vis-www.cs.umass.edu/lfw/
   Visited on : 12th June, 2019