Capstone Project Report

Face Detection and Recognition

Name: Mukul Kumar

Course: AI and ML (Batch-3)

Duration: 10 months

Problem Statement: Build a Machine Learning model for face detection and recognition.

Prerequisites:

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following url can be referred to as download python

https://www.python.org/downloads/

Once you have python downloaded and installed, you will need to set up PATH variables (if you want to run a python program directly, detailed instructions are below in how to run the software section)

https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/

Setting up the PATH variable is optional as you can also run the program without it and more instructions are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url

https://www.anaconda.com/download/

You will also need to download and install below 3 packages after you install either python or anaconda from the steps above

Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages

pip install -U scikit-learn

pip install numpy pip install

scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages

conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

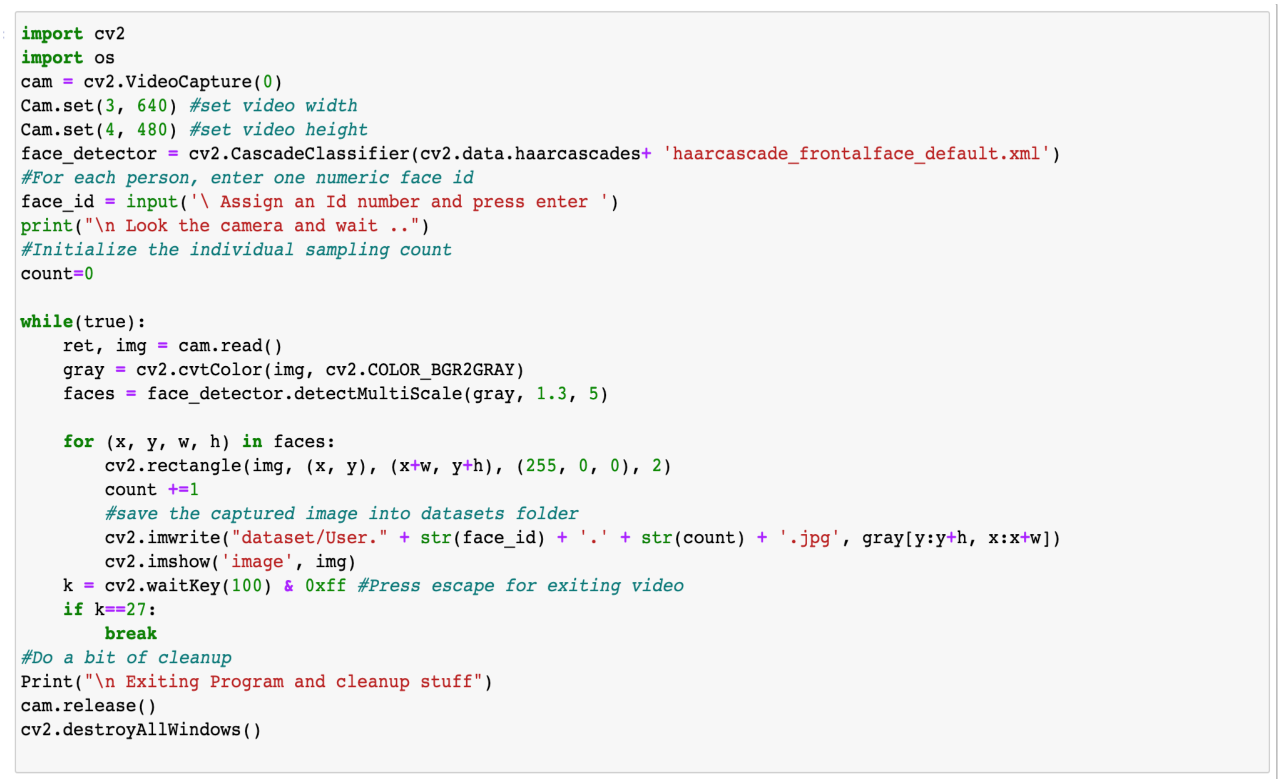
Dataset used

The data source used for this project is by capturing live images. The screenshots of datasets have also been shared in this document .

Method used for detection

Haar cascade classifier

1. Importing the libraries and capturing images:

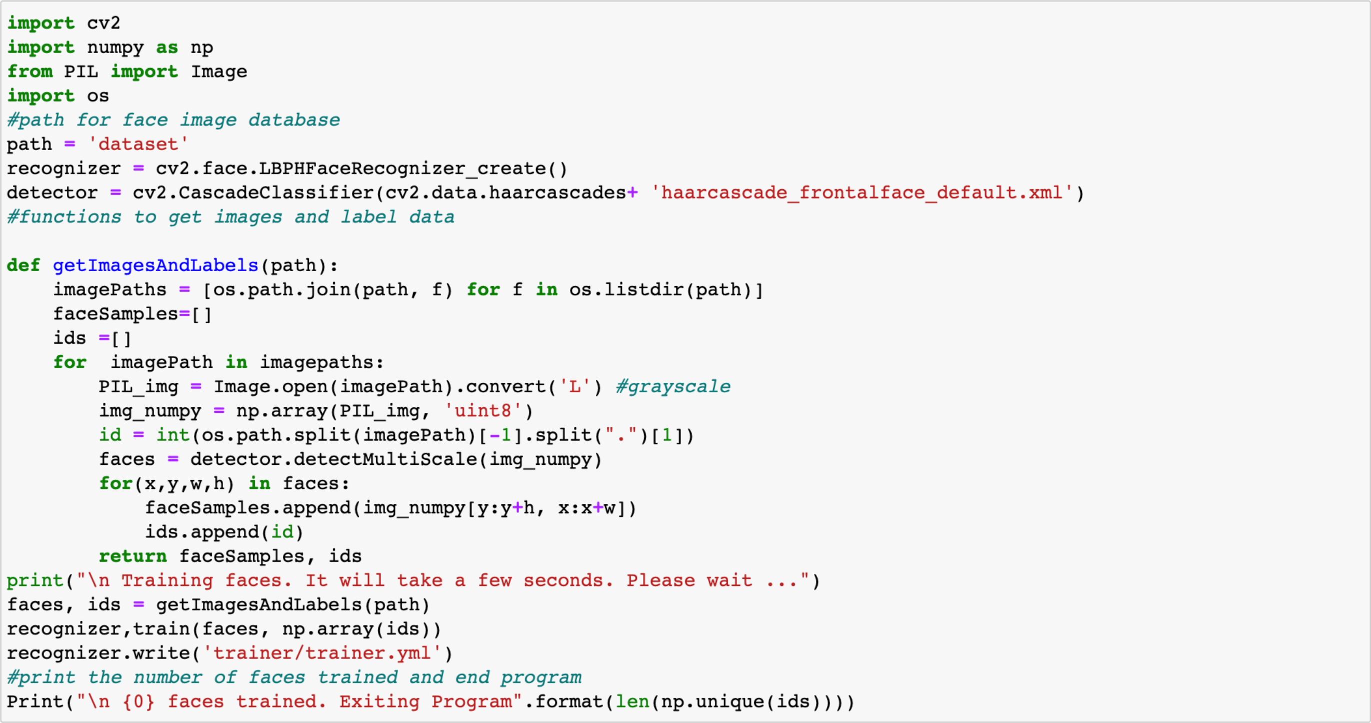


```
import cv2
import os
cam = cv2.VideoCapture(0)
Cam.set(3, 640) #set video width
Cam.set(4, 480) #set video height
face_detector = cv2.CascadeClassifier(cv2.data.haarcascades+ 'haarcascade_frontalface_default.xml')
#For each person, enter one numeric face id
face_id = input('\ Assign an Id number and press enter ')
print("\n Look the camera and wait ..")
#Initialize the individual sampling count
count=0

while(true):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
        count +=1
        #save the captured image into datasets folder
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + '.jpg', gray[y:y+h, x:x+w])
        cv2.imshow('image', img)
    k = cv2.waitKey(100) & 0xff #Press escape for exiting video
    if k==27:
        break
#Do a bit of cleanup
Print("\n Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

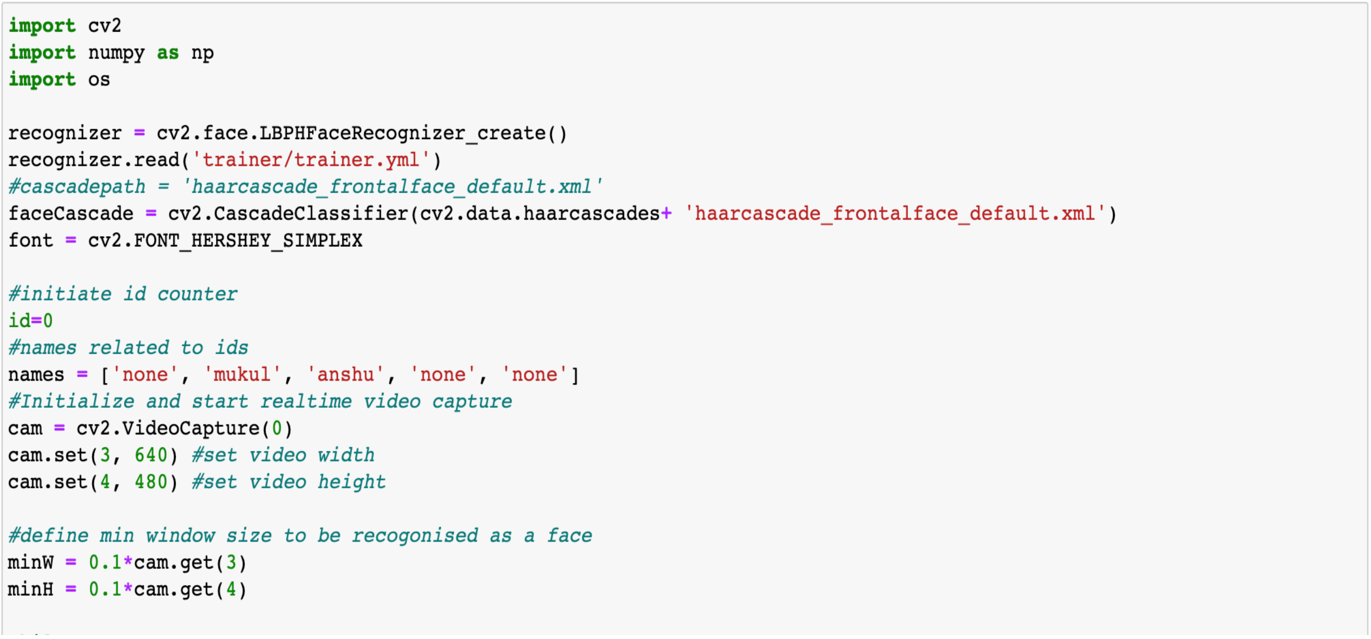
2. Training Data:



```
import cv2
import numpy as np
from PIL import Image
import os
#path for face image database
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier(cv2.data.haarcascades+ 'haarcascade_frontalface_default.xml')
#functions to get images and label data

def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faceSamples=[]
    ids =[]
    for  imagePath in imagepaths:
        PIL_img = Image.open(imagePath).convert('L') #grayscale
        img_numpy = np.array(PIL_img, 'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for(x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h, x:x+w])
            ids.append(id)
        return faceSamples, ids
print("\n Training faces. It will take a few seconds. Please wait ...")
faces, ids = getImagesAndLabels(path)
recognizer,train(faces, np.array(ids))
recognizer.write('trainer/trainer.yml')
#print the number of faces trained and end program
Print("\n {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

3. Recognition using the trained data:

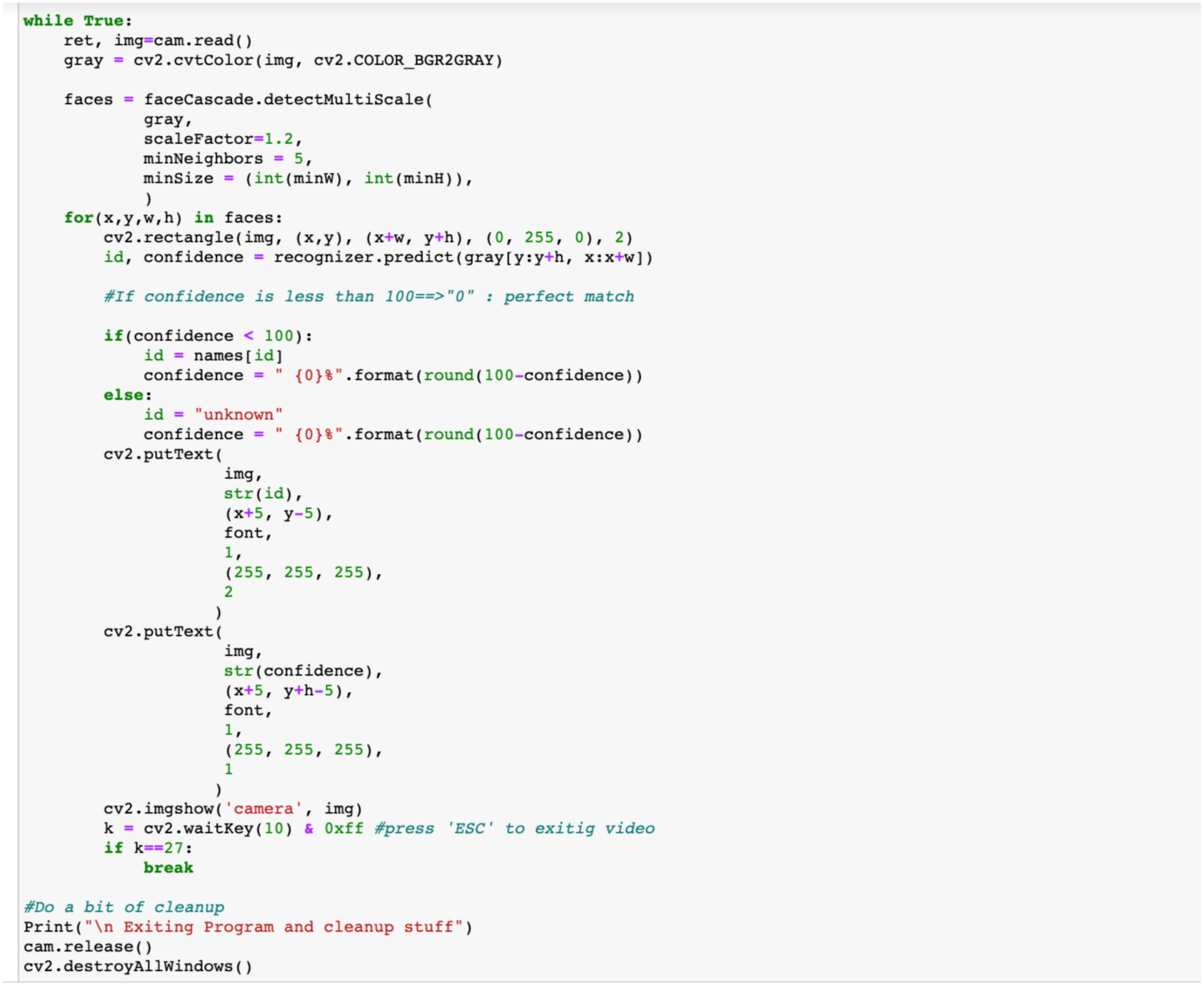


```
import cv2
import numpy as np
import os

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
#cascadepath = 'haarcascade_frontalface_default.xml'
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades+ 'haarcascade_frontalface_default.xml')
font = cv2.FONT_HERSHEY_SIMPLEX

#initiate id counter
id=0
#names related to ids
names = ['none', 'mukul', 'anshu', 'none', 'none']
#Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) #set video width
cam.set(4, 480) #set video height

#define min window size to be recogonised as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
```



```
while True:
    ret, img=cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor=1.2,
            minNeighbors = 5,
            minSize = (int(minW), int(minH)),
            )
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w, y+h), (0, 255, 0), 2)
        id, confidence = recognizer.predict(gray[y:y+h, x:x+w])

        #If confidence is less than 100==>"0" : perfect match

        if(confidence < 100):
            id = names[id]
            confidence = " {0}%".format(round(100-confidence))
        else:
            id = "unknown"
            confidence = " {0}%".format(round(100-confidence))
        cv2.putText(
                    img,
                    str(id),
                    (x+5, y-5),
                    font,
                    1,
                    (255, 255, 255),
                    2
                   )
        cv2.putText(
                    img,
                    str(confidence),
                    (x+5, y+h-5),
                    font,
                    1,
                    (255, 255, 255),
                    1
                   )
        cv2.imgshow('camera', img)
        k = cv2.waitKey(10) & 0xff #press 'ESC' to exitig video
        if k==27:
            break

#Do a bit of cleanup
Print("\n Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```