Capstone Project Report

Face Detection and Recognition

Name: Shreya Das

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 latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction 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

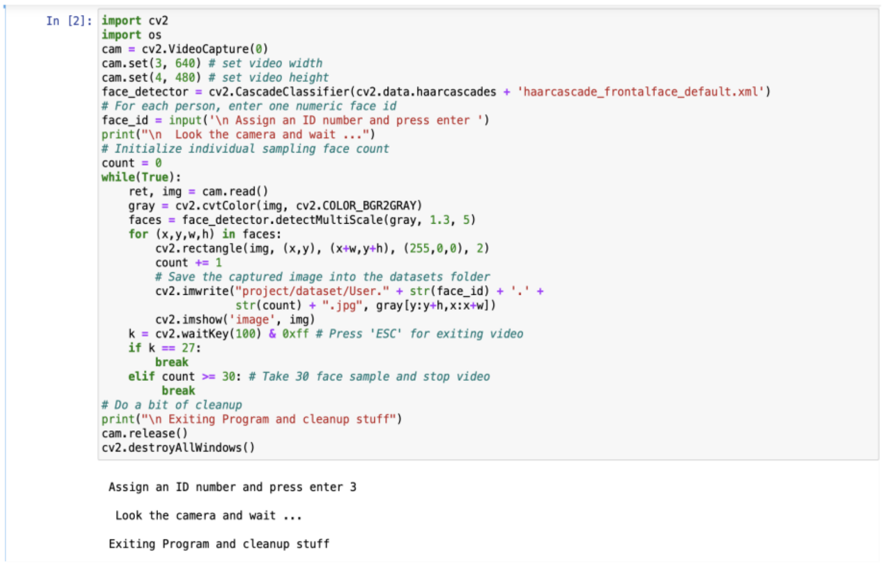
Dataset used

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

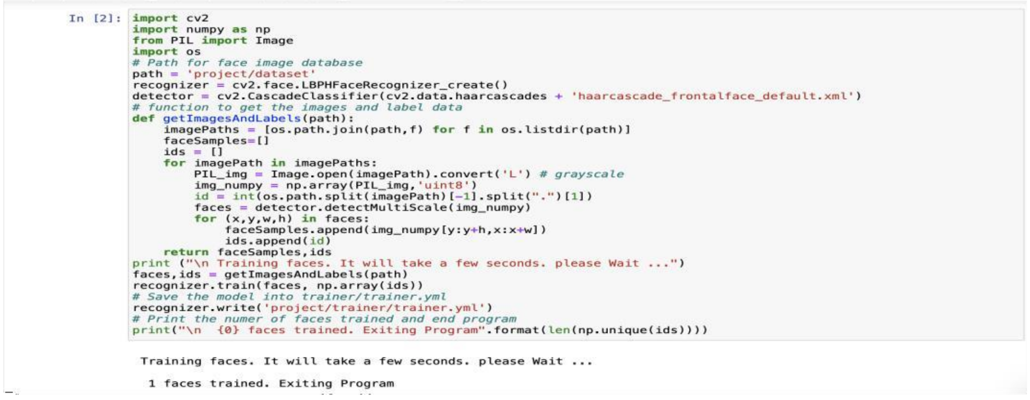
Harr cascade classifier

Importing the libraries and capturing images:



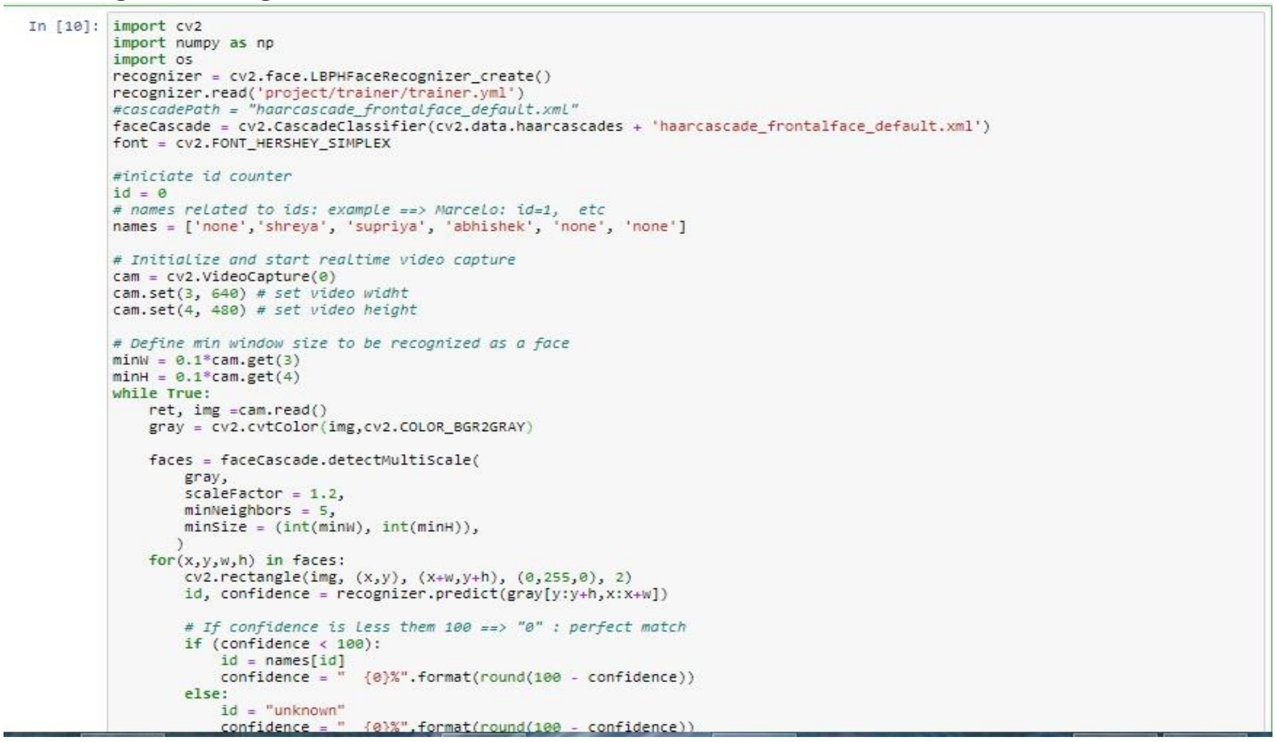
```
In [2]: import cv2
        import os
        cam = cv2.VideoCapture(0)
        cam.set(3, 640) # set video width
        cam.set(4, 480) # set video height
        face_detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        # For each person, enter one numeric face id
        face_id = input('\n Assign an ID number and press enter ')
        print("\n  Look the camera and wait ...")
        # Initialize individual sampling face 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 the datasets folder
                cv2.imwrite("project/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 'ESC' for exiting video
            if k == 27:
                break
            elif count >= 30: # Take 30 face sample and stop video
                 break
        # Do a bit of cleanup
        print("\n Exiting Program and cleanup stuff")
        cam.release()
        cv2.destroyAllWindows()

 Assign an ID number and press enter 3

  Look the camera and wait ...

 Exiting Program and cleanup stuff
```

2. Training Data:

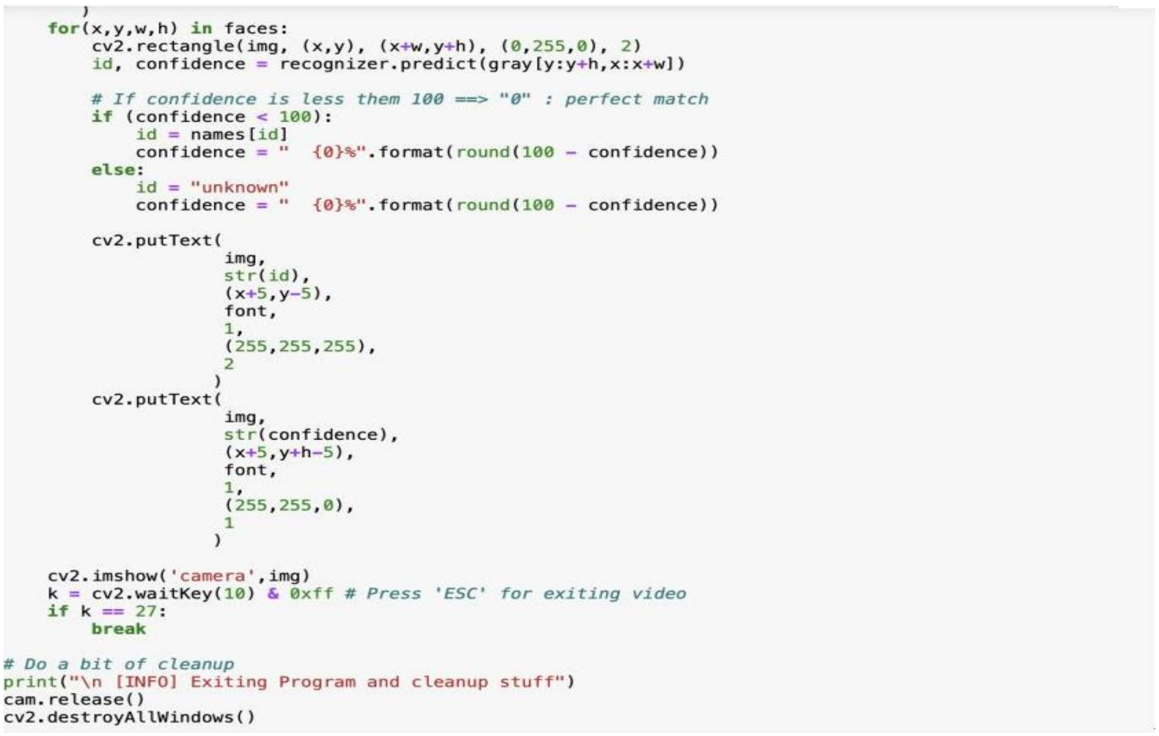


```
In [2]: import cv2
        import numpy as np
        from PIL import Image
        import os
        # Path for face image database
        path = 'project/dataset'
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        # function to get the 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))
        # Save the model into trainer/trainer.yml
        recognizer.write('project/trainer/trainer.yml')
        # Print the numer of faces trained and end program
        print("\n  {0} faces trained. Exiting Program".format(len(np.unique(ids))))

 Training faces. It will take a few seconds. please Wait ...

  1 faces trained. Exiting Program
```

3. Recognition using the trained data:

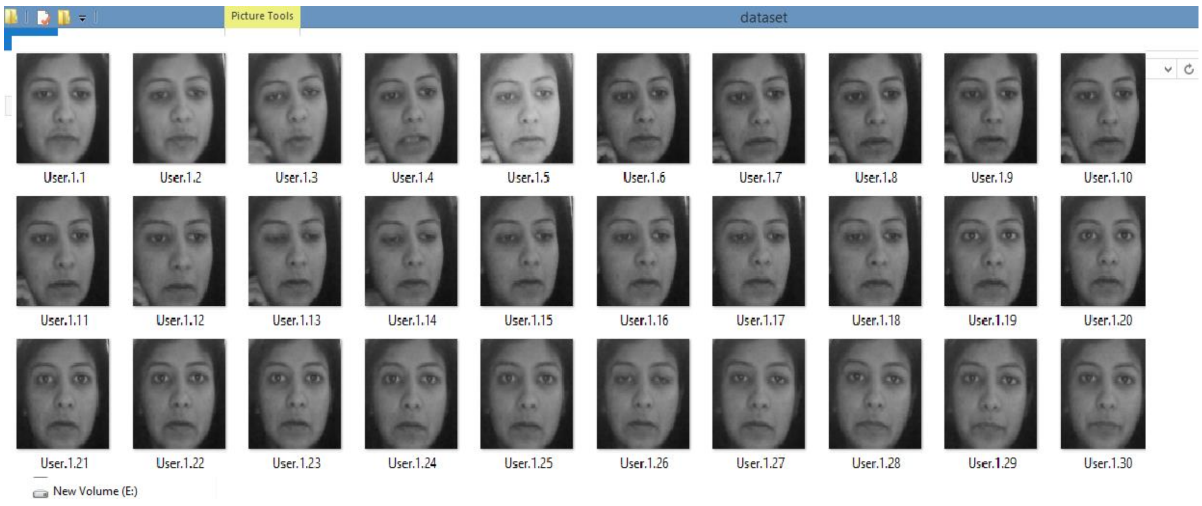


```
In [10]: import cv2
         import numpy as np
         import os
         recognizer = cv2.face.LBPHFaceRecognizer_create()
         recognizer.read('project/trainer/trainer.yml')
         #cascadePath = "haarcascade_frontalface_default.xml"
         faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
         font = cv2.FONT_HERSHEY_SIMPLEX

         #iniciate id counter
         id = 0
         # names related to ids: example ==> Marcelo: id=1,  etc
         names = ['none','shreya', 'supriya', 'abhishek', 'none', 'none']

         # Initialize and start realtime video capture
         cam = cv2.VideoCapture(0)
         cam.set(3, 640) # set video widht
         cam.set(4, 480) # set video height

         # Define min window size to be recognized 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 them 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))
```

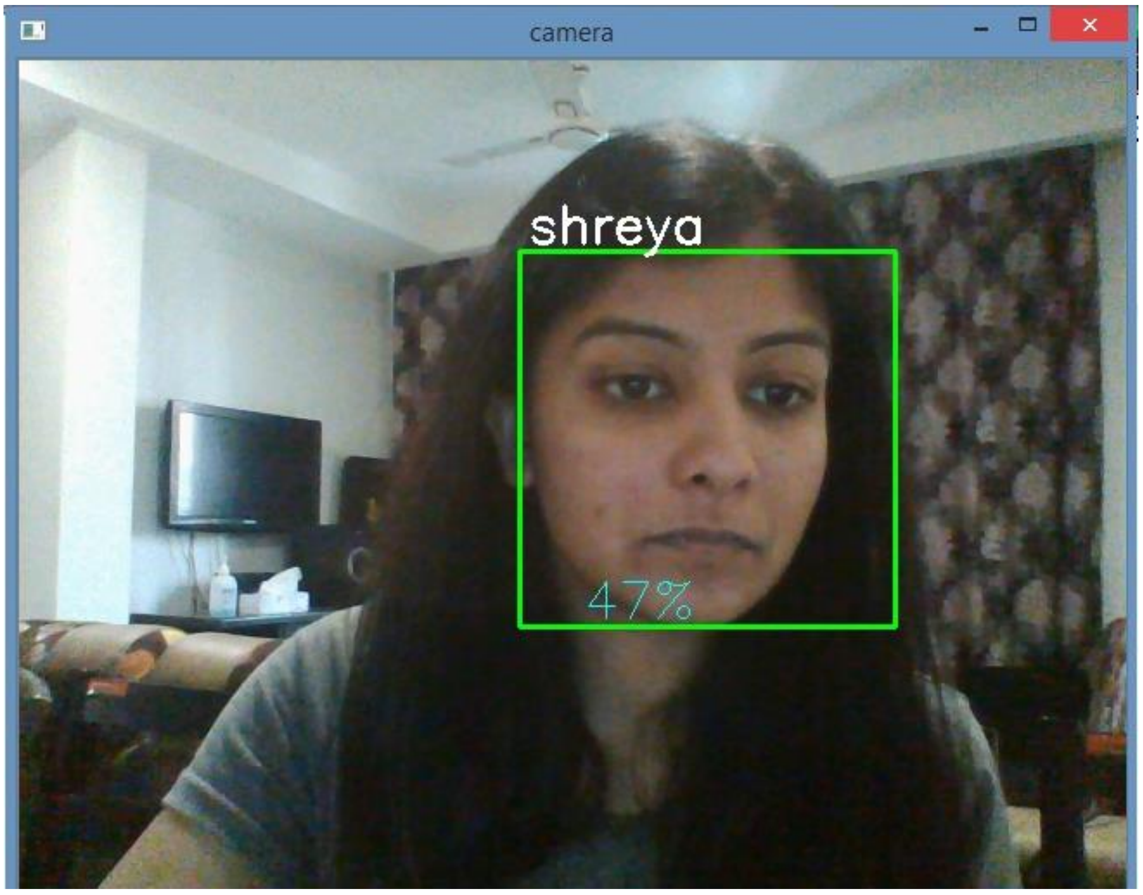


```
        )
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        # If confidence is less them 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,
                    (255,255,0),
                    1
                   )

    cv2.imshow('camera',img)
    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break

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

4. Output while capturing images for Data set.



5. FINAL OUTPUT


camera
shreya
47%