Face Recognition Model

Introduction :

This project is made by using OpeCV and python. It has many application in real world problems. Face recognition may be used in automatic attendance by scanning students face or It may also use in home security locks which scans the face and allow access to only house members.

It consists 3 phases :
To create a complete project on Face Recognition, we must work on 3 very distinct phases:

- Face Detection and Data Gathering
- Train the Recognizer

There are some requirements needed to implement our model.
1. I am running my code in anaconda (spyder).
2. OpenCv must be installed.
3. Sufficient knowledge of language python.

Testing Camera

Enter the below Python code on your IDE:

```
import numpy as np
import cv2 cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
while(True):
ret, frame = cap.read()
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
cv2.imshow('frame', frame)
cv2.imshow('gray', gray)
k = cv2.waitKey(30) & 0xff
if k == 27: # press 'ESC' to quit
break
cap.release()
cv2.destroyAllWindows()
```

The above code will capture the video stream , displaying both, in BGR color and Gray mode.

Phase 1: Face Detection

The most basic task on Face Recognition is of course, “Face Detecting”. Before anything, you must “capture” a face (Phase 1) in order to recognize it, when compared with a new face captured on future (Phase 3).
The most common way to detect a face (or any objects), is using the “Haar Cascade classifier”.

Here is the code save it as detection.py

```
import numpy as np
import cv2 faceCascade = cv2.CascadeClassifier('Cascades/haarcascade_frontalface_default.xml')
#load classifier
cap = cv2.VideoCapture(0)
cap.set(3,640) # set Width
cap.set(4,480) # set Height
while True: ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# passing it some very important parameters, as scale factor, number of neighbors and minimum size of the detected face.

faces = faceCascade.detectMultiScale( gray, scaleFactor=1.2, minNeighbors=5, minSize=(20, 20) )

# The function will detect faces on the image. Next, we must “mark” the faces in the image, using, for example, a blue rectangle. This is done with this portion of the code

for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
cv2.imshow('video',img)
k = cv2.waitKey(30) & 0xff
if k == 27: # press 'ESC' to quit
break
cap.release()
cv2.destroyAllWindows()
```

Data Gathering

```
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('haarcascade_frontalface_default.xml') # For each person,
enter one numeric face id

face_id = input('\n enter user id end press <return> ==> ')
print("\n [INFO] Initializing face capture. 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("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 [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

Phase 2: Trainer

On this second phase, we must take all user data from our dataset and "trainer" the OpenCV Recognizer. This is done directly by a specific OpenCV function. The result will be a .yml file that will be saved on a "trainer/" directory.

```
import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create() #I am using as a recognizer, the LBPH (LOCAL
BINARY PATTERNS HISTOGRAMS) Face Recognizer, included on OpenCV package.
detector = cv2.CascadeClassifier("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') # convert it to 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 [INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
```

```
recognizer.train(faces, np.array(ids)) # Save the model into trainer/trainer.yml
recognizer.write('trainer/trainer.yml')
# Print the numer of faces trained and end program
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

As a result, a file named “trainer.yml” is saved in the trainer directory .

Phase 3 : Recognizer

Now, we reached the final phase of our project. Here, we will capture a fresh face on our camera and if this person had his face captured and trained before, our recognizer will make a “prediction” returning its id and an index, shown how confident the recognizer is with this match.

```
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(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX indiciate id counter id = 0 #names related to ids: example
==> Shivam: id=1, etc
names = ['None', 'Shivam', 'Arun', 'Ritu’] # including here a new array, so we will display
“names”
# Initialize and start real time 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 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]) # Check if confidence is less them 100 ==>
"0" is 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()
```

The recognizer.predict (), will take as a parameter a captured portion of the face to be analyzed and will return its probable owner, indicating its id and how much confidence the recognizer is in relation with this match.
And at last, if the recognizer could predict a face, we put a text over the image with the probable id and how much is the "probability" in % that the match is correct ("probability" = 100 — confidence index). If not, an "unknow" label is put on the face.

Conclusion

For details and final code, please visit my GitHub depository:
https://github.com/ShivamAgarwal99/Face-Recognizer

Here is Video link :
https://www.youtube.com/watch?v=xjjOtyX5wkQ&feature=youtu.be

Resources

1. https://www.tutorialspoint.com/opencv/
2. Machine Learning A-Z : Hands –On Python & R in Data Science Coursera from udemy.
3. Youtube Channel of Siraj Raval.
4. Sentdex Videos on Face recognition.

Contact me at : shivam.ag.1999@gmail.com