Build Face Recognition Attendance System using Python

ADVANCED COMPUTER VISION IMAGE IMAGE ANALYSIS PYTHON

This article was published as a part of the Data Science Blogathon

Introduction

In this article, you will learn how to build a face-recognition system using Python. Face recognition is a step further to face detection. In face detection, we only detect the location of the human face in an image but in face recognition, we make a system that can identify humans.

"Face recognition is a broad challenge of verifying or identifying people in pictures or videos. Big tech giants are still working to make a faster and more accurate face recognition model."

Real-World Applications of Face Recognition

Face recognition is currently being used to make the world safer, smarter, and more convenient.

There are a few use cases :

- Finding Missing Person
- Retail Crime
- Security Identification
- Identifying accounts on social media
- School Attendance System
- Recognizing Drivers in Cars

There are several methods to perform facial recognition depending on the performance and complexity.

Traditional Face Recognition Algorithms:

During the 1990s holistic approaches were used for face recognition. Handcrafted local descriptors became popular In the early 1920s, and then the local feature learning approaches were followed in the late 2000s. Nowadays algorithms that are widely used and are implemented in OpenCV are as follows:

- **Eigenfaces (1991)**
- **Local Binary Patterns Histograms (LBPH) (1996)**
- **Fisherfaces (1997)**
- **Scale Invariant Feature Transform (SIFT) (1999)**
- **Speed Up Robust Features (SURF) (2006)**

Each method follows a different approach to extracting the image information and matching it with the input image.

Fischer-faces and *Eigenfaces* have almost similar approaches as well as SURF and SIFT.

LBPH is a simple yet very efficient method but it's slow compared to modern days face -recognizers.

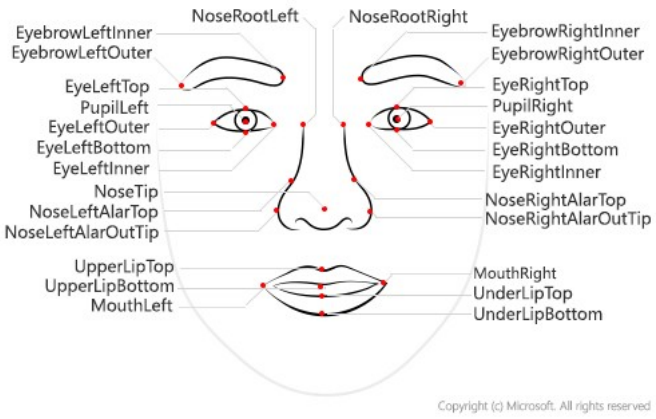
These algorithms are not faster compared to modern days face-recognition algorithms. Traditional algorithms can't be trained only by taking a single picture of a person.

Deep Learning for Face Recognition:

Some of the widely used Deep Learning-based Face Recognition systems are as follows:

- DeepFace
- **DeepID** series of systems
- VGGFace
- FaceNet

Face recognizers generally take face images and find the important points such as the corner of the mouth, an eyebrow, eyes, nose, lips, etc. Coordinates of these points are called facial-features points, there are such 66 points. In this way, a different technique for finding feature
points give different results.



source: https://www.pinterest.com/mrmosherart/face-landmarks/

Steps involved in a face recognition model:

1. **Face Detection:** Locate faces and draw bounding boxes around faces and keep the coordinates of bounding boxes.
2. **Face Alignments:** Normalize the faces to be consistent with the training database.
3. **Feature Extraction:** Extract features of faces that will be used for training and recognition tasks.
4. **Face Recognition:** Matching of the face against one or more known faces in a prepared database.

In the traditional method of face recognition, we had separate modules to perform these 4 steps, which was painful. -In this article, you will see a library that combines all these 4 steps in a single step.

Steps to Build the Face Recognition System

Install Libraries

We need to install 2 libraries in order to implement face recognition.

dlib: Dlib
is a modern C++ toolkit containing machine learning algorithms and
tools for creating complex software in C++ to solve real-world problems.

```
# installing dlib pip install dlib
```

face recognition: The face_recognition library, created and maintained by **Adam Geitgey**, wraps *around* **dlib** facial recognition functionality.

```
# installing face recognition pip install face recognition
```

Opencv for some image pre-processing.

```
# installing opencv pip install opencv
```

> **Note**: if in case you encounter any error while installing **dlib** ,i would recommend you to install the C++ development toolkit using vs_code community

Import Libraries

Now that you have downloaded all the important libraries let's import them to build the system.

```
import cv2 import numpy as np import face_recognition
```

Loading Images

After importing libraries you need to load an image.

face_recognition library loads images in the form of BGR, in order to
print the image you should convert it into RGB using OpenCV.

```
imgelon_bgr = face_recognition.load_image_file('elon.jpg') imgelon_rgb = cv2.cvtColor(imgelon_bgr,cv2.COLOR_BGR2RGB) cv2.imshow('bgr', imgelon_bgr) cv2.imshow('rgb', imgelon_rgb) cv2.waitKey(0)
```

Source: Local

As you see RGB looks natural so you will always change the channel to RGB.

Find Face Location and Draw Bounding Boxes

You need to draw a bounding box around the faces in order to show if the human face has been detected or not.

```
imgelon =face_recognition.load_image_file('elon.jpg') imgelon = cv2.cvtColor(imgelon,cv2.COLOR_BGR2RGB) #----------Finding face Location for drawing bounding boxes------- face = face_recognition.face_locations(imgelon_rgb)[0] copy = imgelon.copy() #-------------------Drawing the Rectangle------------------------- cv2.rectangle(copy, (face[3], face[0]),(face[1], face[2]), (255,0,255), 2) cv2.imshow('copy', copy) cv2.imshow('elon',imgelon) cv2.waitKey(0)
```

Source: Local

Train an Image for Face Recognition

This library is made in such a way that it automatically finds the face and work on only faces, so you don't need to crop the face out of
pictures.

Training:

At this stage, we convert the train image into some encodings and store the encodings with the given name of the person for that image.

```
train_elon_encodings = face_recognition.face_encodings(imgelon)[0]
```

Testing:

For testing, we load an image and convert it into encodings, and now match encodings with the stored encodings during training, this matching is based on finding maximum similarity. When you find the encoding matching to the test image you get the name associated with train encodings.

```
# lets test an image test = face_recognition.load_image_file('elon_2.jpg') test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB) test_encode = face_recognition.face_encodings(test)[0] print(face_recognition.compare_faces([train_encode],test_encode))
```

face_recognition.compare_faces returns **True** if the person in both images are the same other it returns **False.**

Building a Face Recognition System

Import Necessary Libraries

```
import cv2 import face_recognition import os import numpy as np from datetime import datetime import pickle
```

Define a folder path where your training image dataset will be stored

```
path = 'student_images'
```

> **Note**: for training, we only need to drop the training images in the **path** directory and the image name must be **person_name.jpg/jpeg** format.

for example:

as you see in my student_images path I have 6 persons. hence our model can recognize only these 6 persons. you can add more pictures in this directory for more persons to be recognized

Source: Local

- now create a list to store person_name and image array.
- traverse all image file present in **path** directory, read images, and append the image array to the **image** list, and file-name to **classNames.**

```
images = [] classNames = []mylist = os.listdir(path) for cl in mylist: curImg = cv2.imread(f'{path}/{cl}') images.append(curImg) classNames.append(os.path.splitext(cl)[0])
```

- create a function to encode all the train images and store them in a variable **encoded_face_train.**

```
def findEncodings(images): encodeList = [] for img in images: img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) encoded_face = face_recognition.face_encodings(img)[0] encodeList.append(encoded_face) return encodeList encoded_face_train = findEncodings(images)
```

- Creating a function that will create a **Attendance.csv** file to store the attendance with time.

Note: here you need to create **Attendance.csv** file manually and give the path in the function

```
def markAttendance(name): with open('Attendance.csv','r+') as f: myDataList = f.readlines() nameList = [] for line in myDataList: entry = line.split(',') nameList.append(entry[0]) if name not in nameList: now = datetime.now() time = now.strftime('%I:%M:%S:%p') date = now.strftime('%d-%B-%Y') f.writelines(f'n{name}, {time}, {date}')
```

with open("filename.csv",'r+') creates a file and **'r+'** mode is used to open a file for reading and writing.

We first check if the name of the attendee is already available in attendance.csv we won't write attendance again.

If the attendee's name is not available in attendance.csv we will write the attendee name with a time of function call.

Read Webcam for Real-Time Recognition

```
# take pictures from webcam cap = cv2.VideoCapture(0)while True: success, img = cap.read() imgS = cv2.resize(img, (0,0), None, 0.25,0.25) imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB) faces_in_frame = face_recognition.face_locations(imgS) encoded_faces = face_recognition.face_encodings(imgS, faces_in_frame)for encode_face, faceloc in zip(encoded_faces,faces_in_frame): matches = face_recognition.compare_faces(encoded_face_train, encode_face) faceDist = face_recognition.face_distance(encoded_face_train, encode_face) matchIndex = np.argmin(faceDist) print(matchIndex) if matches[matchIndex]: name = classNames[matchIndex].upper().lower() y1,x2,y2,x1 = faceloc # since we scaled down by 4 times y1, x2,y2,x1 = y1*4,x2*4,y2*4,x1*4 cv2.rectangle(img,(x1,y1),(x2,y2), (0,255,0),2) cv2.rectangle(img, (x1,y2-35),(x2,y2), (0,255,0), cv2.FILLED) cv2.putText(img,name, (x1+6,y2-5), cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2) markAttendance(name) cv2.imshow('webcam', img) if cv2.waitKey(1) & 0xFF == ord('q'): break
```

- Resize the image by 1/4 only for the recognition part. output frame will be of the original size.
- Resizing improves the Frame per Second.
- **face_recognition.face_locations()** is called on the resized image(**imgS**) .for face bounding box coordinates must be multiplied by 4 in order to overlay on the output frame.
- **face_recognition.distance()** returns an array of the distance of the test image with all images present in our train directory.
- The index of the minimum face distance will be the matching face.
- After finding the matching name we call the **markAttendance** function.
- Draw bounding box using **cv2.rectangle().**
- We put the matching name on the output frame using **cv2.putText().**

Source: Local

Attendance Report

Source: Local

Challenges faced by Face Recognition Systems

Although building facial recognition seems easy it is not as easy in the real world images that are being taken without any constraint. There are several challenges that are faced by the Facial Recognitions System are as follows:

- **Illumination:** It changes the face appearance drastically, it is observed that the slight changes in lighting conditions cause a significant impact on its results.
- **Pose:** Facial Recognition systems are highly sensitive to the pose, Which may result in faulty recognition or no recognition if the database is only trained on frontal face view.
- **Facial Expressions**: Different expressions of the same individual are another significant factor that needs to be taken into account. Modern Recognizers can easily deal with it though.
- **Low Resolution**: Training of recognizer must be done on a good resolution picture, otherwise the model will fail to extract features.
- **Aging:** With increasing age, the human face features shape, lines, texture changes which are yet another challenge.

Conclusion

In this article, we discussed how to create a face recognition system using the **face_recognition** library and made an attendance system. you can further design GUI using **Tkinter** or **Pyqt** for the face recognition attendance system.

Thanks for reading the article, please share if you liked this article.

Article From: Abhishek Jaiswal, Reach out to me on LinkedIn.

Article Url - https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python/

thetechwriters