Face Authentication

Ritik Gautam, B180630CS

Abstract

Face recognition is a technology capable of identifying or verifying a subject through an image, video or any audiovisual element of his face. Generally, this identification is used to access an application, system or service. The security of information is becoming very significant and difficult. Security cameras are presently common in airports, Offices, University, ATM, Bank and in any locations with a security system. Face recognition is a biometric system used to identify or verify a person from a digital image. Face Recognition system is used in security. Face recognition system should be able to automatically detect a face in an image. This involves extracts its features and then recognize it, regardless of lighting, expression, illumination, ageing, transformations (translate, rotate and scale image) and pose, which is a difficult task. Human face detection has become a major field of interest in current research because there is no deterministic algorithm to find face(s) in a given image. A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces. Development beginning was as a form of computer application. Since their inception, facial recognition systems have seen wider uses in recent times on smartphones and in other forms of technology, such as robotics. Because computerized facial recognition involves the measurement of a human's physiological characteristics, facial recognition systems are categorized as biometrics. Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance and automatic indexing of images. Some face recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

1. Problem Statement

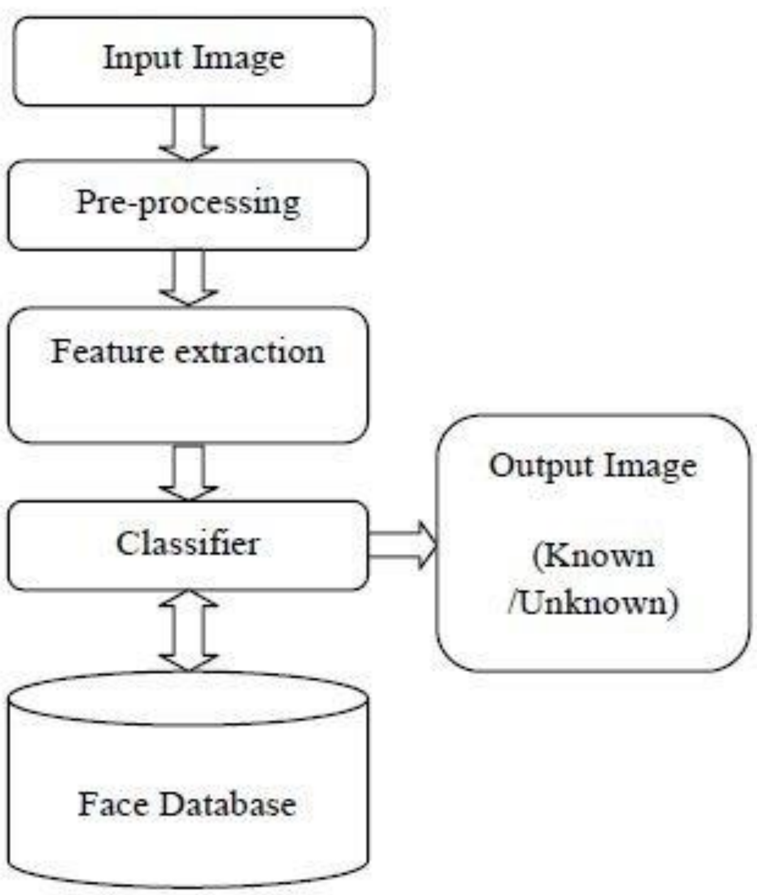
To build a Classifier that can be used for face Authentication.

Input: An image containing the front face photo of a person.

Output: ‘Yes’ if the image is containing your face and ‘No’ otherwise.

2. Design

A pattern recognition task performed exclusively on faces is termed as face recognition. It can be described as classifying a face either known or unknown, after matching it with stored known individuals as a database. It is also advantageous to have a system that has the capability of learning to identify unknown faces. There are five main functional blocks.



Face Recognition

In this phase the face part is recognized in the given image. Here it will try to make a boundary around the face region. And all the further operations are done in this region. I have used Viola-Jones Algorithm for face detection. Given an image the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

It has four stages,

1. Haar Feature Selection
2. Creating an Integral Image
3. Adaboost Training
4. Cascading Classifiers

Haar-like features are digital image features used in object recognition. All human faces share some universal properties of the human face like the eyes region is darker than its neighbor pixels, and the nose region is brighter than the eye region. A simple way to find out which region is lighter or darker is to sum up the pixel values of both regions and compare them. The sum of pixel values in the darker region will be smaller than the sum of pixels in the lighter region. If one side is lighter than the other, it may be an edge of an eyebrow or sometimes the middle portion may be shinier than the surrounding boxes, which can be interpreted as a nose This can be accomplished using Haar-like features and with the help of them, we can interpret the different parts of a face.

To calculate a value for each feature, we need to perform computations on all the pixels inside that particular feature. In reality, these calculations can be very intensive since the number of pixels would be much greater when we are dealing with a large feature. The integral image plays its part in allowing us to perform these intensive calculations quickly so we can understand whether a feature of several features fit the criteria. An integral image is the name of both a data structure and an algorithm used to obtain this data structure. It is used as a quick and efficient way to calculate the sum of pixel values in an image or rectangular part of an image.

The number of features that are present in the 24×24 detector window is nearly 160,000, but only a few of these features are important to identify a face. So, we use the AdaBoost algorithm to identify the best features in the 160,000 features. In the Viola-Jones algorithm, each Haar-like feature represents a weak learner. To decide the type and size of a feature that goes into the final classifier, AdaBoost checks the performance of all classifiers that you supply to it. To calculate the performance of a classifier, you evaluate it on all subregions of all the images used for training. Some subregions will produce a strong response in the classifier. Those will be classified as positives, meaning the classifier thinks it contains a human face. Subregions that don’t provide a strong response don’t contain a human face, in the classifier’s opinion. They will be classified as negatives. The classifiers that performed well are given higher importance or weight. The final result is a strong classifier, also called a boosted classifier, that contains the best performing weak classifiers.

The job of the cascade is to quickly discard non-faces, and avoid wasting precious time and computations. Thus, achieving the speed necessary for real-time face detection. We set up a cascaded system in which we divide the process of identifying a face into multiple stages. In the first stage, we have a classifier which is made up of our best features, in other words, in the first stage, the subregion passes through the best features such as the feature which identifies the nose bridge or the one that identifies the eyes. In the next stages, we have all the remaining features.

Feature Extraction

Now that we have cropped the face out of the image, we extract features from it. Here we are going to use face embeddings to extract the features out of the face. A neural network takes an image of the person's face as input and outputs a vector which represents the most important features of a face. In machine learning, this vector is called embedding and thus we call this vector as face embedding. While training the neural network, the network learns to output similar vectors for faces that look similar. After training the network, the network learns to output vectors that are closer to each other for faces of the same person. We use a CNN here.
Convolution preserves the relationship between pixels by learning image features using small squares of input data. Generally, increasing the number of convolutional layers will increase the learned feature complexity. The convolutional layer uses a set of learnable kernels sliding across the input image to extract the features. The network outputs a vector of 128 numbers which represent the most important features of a face. We pass all the images in our data to this pre-trained network to get the respective embeddings and save these embeddings in a file for the next step.

Face Recognition

Now that we have face embeddings for every face in our data, the next step is to recognize a new image that is not in our data. So, the first step is to compute the face embedding for the image using the same network we used above and then compare this embedding with the rest of the embeddings we have. We recognize the face if the generated embedding is closer or similar to any other embedding. For this we use a neural network which will compare embeddings of the given image with the known embeddings which we stored previously and return True for those that are similar and False otherwise. If we get False for all the images in data, it will return 'Unknown'. Otherwise, it will return the name of the person in the data which got more Trues while comparing.

2.1. Dataset

The dataset used for this model contains my images(Ritik Images) and some other images from google images(unknown images).
Data includes images and a csv file with the information about images. This data is used to train the model. It is the basic information provided based on which model makes the predictions.

3. Implementation Details & Results

Platform: Google Colab
Programming Language: Python
I also have used various open-source libraries like pandas, OpenCV, etc.

Steps for the implementation:

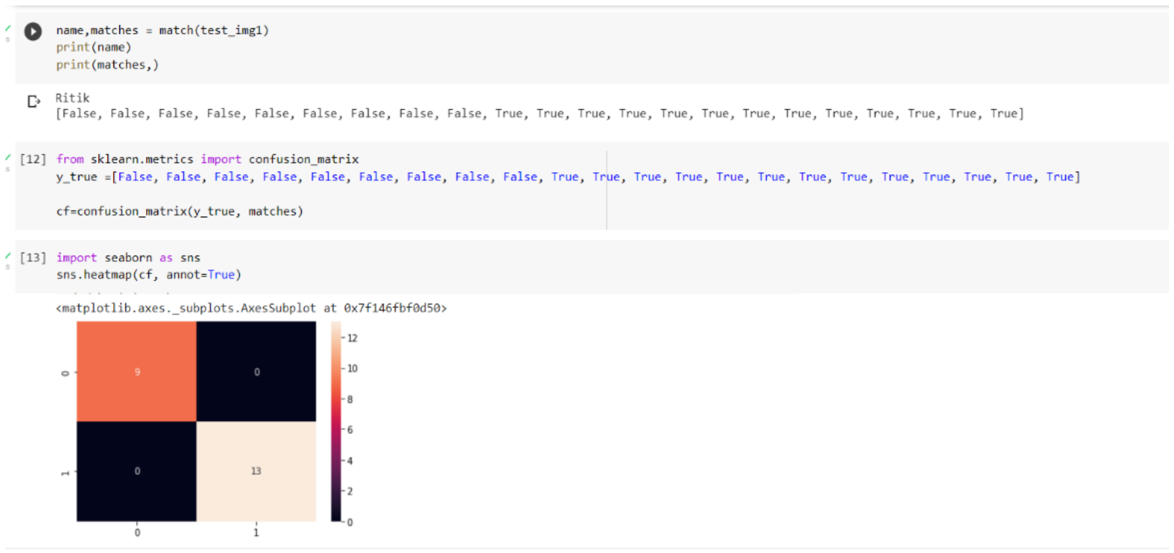
1. load the known faces and embeddings saved in file.
2. Find path to the image you want to detect face and pass it.
3. Do the facial embeddings for face in input.
4. Loop over the facial embeddings in case we have multiple embeddings for multiple faces.
5. Compare encodings with encodings in data["encodings"].
6. Matches contain array with Boolean values and True for the embeddings it matches closely and False for rest.
7. Set name = unknown if no encoding matches.
8. Check to see if we have found a match.
9. Find positions at which we get True and store them.
10. loop over the matched indexes and maintain a count for each recognized face.
11. Check the names at respective indexes we stored in matchedIdxs.
12. Set name which has highest count.
13. Give the output name as the final result for our face authentication.

Training Sample Images:

Test Images:

Results:

For input image of mine I got the result and confusion matrix as below, which give an output as my name ‘Ritik’ according to the problem statement. Also, by seeing the below plotted confusion matrix we can derive that the accuracy of the code written is well enough to identify a face.



```
name,matches = match(test_img1)
print(name)
print(matches,)
```

```
Ritik
[False, False, False, False, False, False, False, False, False, True, True, True, True, True, True, True, True, True, True, True, True, True]
```

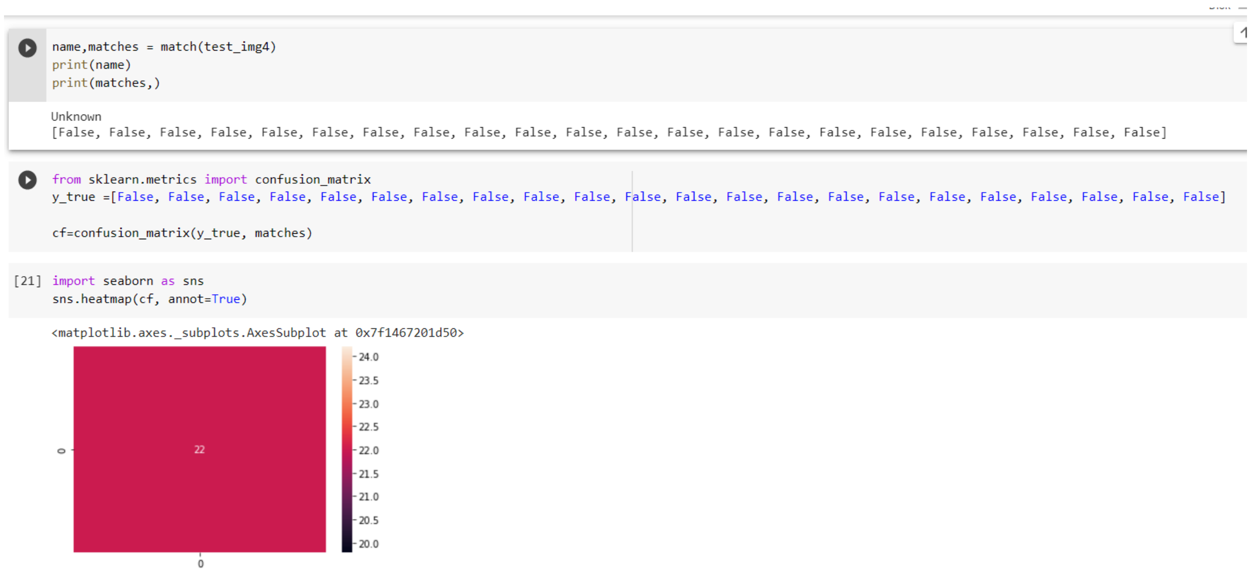
```
[12] from sklearn.metrics import confusion_matrix
     y_true =[False, False, False, False, False, False, False, False, False, True, True, True, True, True, True, True, True, True, True, True, True, True]

     cf=confusion_matrix(y_true, matches)
```

```
[13] import seaborn as sns
     sns.heatmap(cf, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f146fbf0d50>
```

And for the input image which I have inserted as random different from mile gives an output as ‘Unknown’ according the problem statement. Also, below is the plotted confusion matrix of the same.



```
name,matches = match(test_img4)
print(name)
print(matches,)
```

```
Unknown
[False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False]
```

```
from sklearn.metrics import confusion_matrix
y_true =[False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False, False]

cf=confusion_matrix(y_true, matches)
```

```
[21] import seaborn as sns
     sns.heatmap(cf, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1467201d50>
```

4. References

- https://www.mygreatlearning.com/blog/face-recognition
- https://www.researchgate.net/publication/262875649_Design_of_a_Face_Recognition_System
- https://ieeexplore.ieee.org/document/8974493