REPORT ON

Face Recognition

MOHD ZAHID FAIZ : MT2019514

SUBARNA KANTI SAMANTA : MT2019523

PRANJAL KUMAR : MT2019079

UNDER GUIDANCE OF:

PROF. DINESH BABU JAYAGOPI

CONTENTS

Introduction

This is the report on face recognition-based attendance system. This is done in two steps:

1. Face detection: Detecting the region of image where the face is located. Separating out the region of interest from the original image.
2. Face recognition: Classifying the faces based on the labels provided. This is done by first taking an image, detecting the facial region. Then resizing the image and reducing the dimensions of the image. These dimensions are reduced to get the necessary features and make the computations faster. The dimensions are reduced using two different techniques:

a. PCA (Principle Component Analysis)
b. LDA (Linear Discriminant Analysis)

After reducing the dimensions, Euclidean distance is used to find the similarity of test set from the training set. The shortest distance will give the best classification. Top 1^{st}, 5^{th} and 10^{th} accuracies are computes to check if the algorithm really works or not.

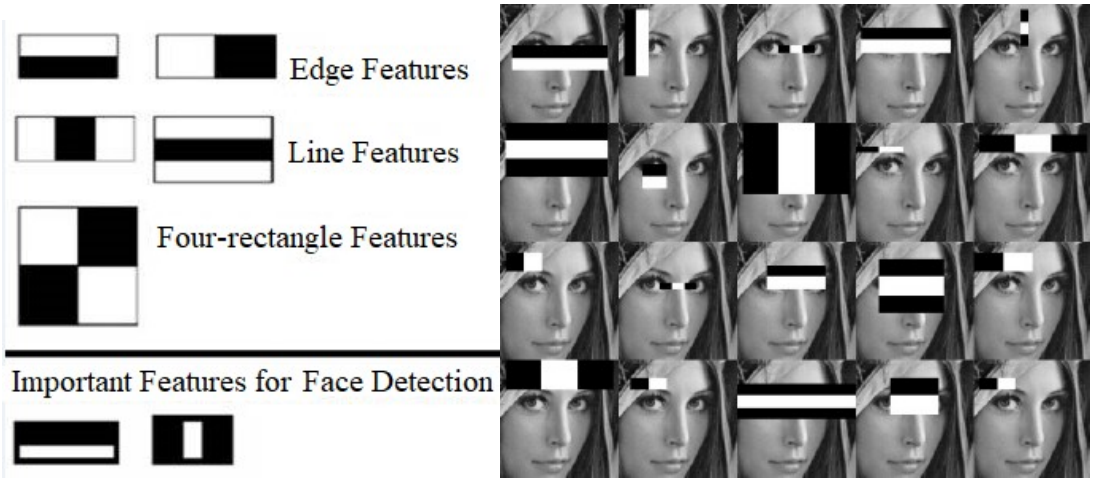
Face Detection

The features of the face like nose, eyes and mouth can be detected using HAAR features. The xml files for the HAAR features to detect the nose eyes mouth is available online. These are used as a cascade blocks to detect different regions of face. It works almost accurately but It has some drawbacks. If the proportions of the image are distorted in any way, then these features don’t work that well. It detects those faces only which are well proportioned as the feature's detector is made to detect.

This drawback quickly shortened the working dataset from 800+ images to only 624 images. Although that much dataset is enough to implement the algorithm, but it hindered to be able to generalize. There were some persons that were not present in the detected dataset.

- **Viola Jones algorithm haar feature detection**

Developed in 2001 by Paul Viola and Michael Jones, the Viola-Jones algorithm is an object-recognition framework that allows the detection of image features in real-time. This algorithm searches the whole image for the haar like features. The haar features are:

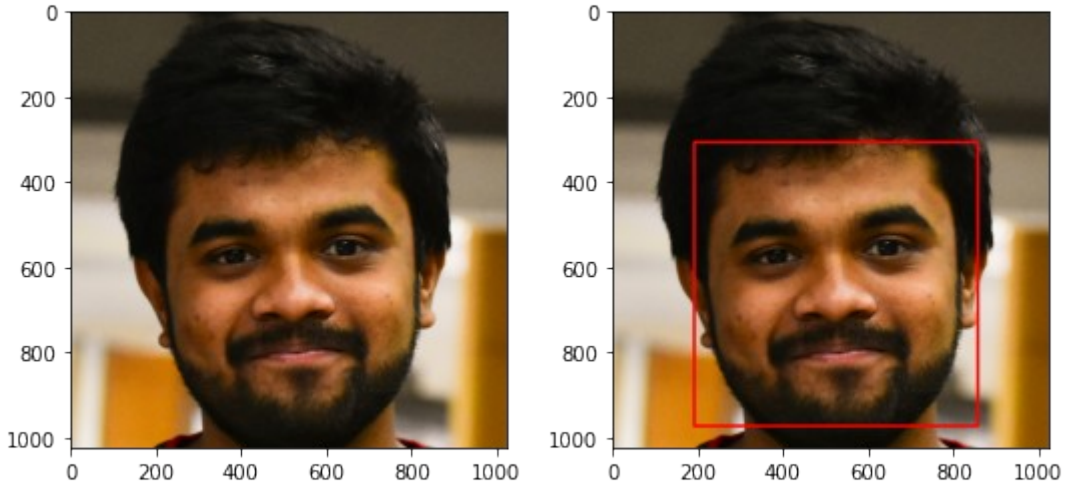


The image will match the wavelets like this.

After the detected features from the image, an ensemble of models are trained to separate the true positives from true negatives. These are nothing but just face or non-face classifiers. Ada boosting is used to train these ensembles of forest. Cascading is used to improve the speed of the algorithm.

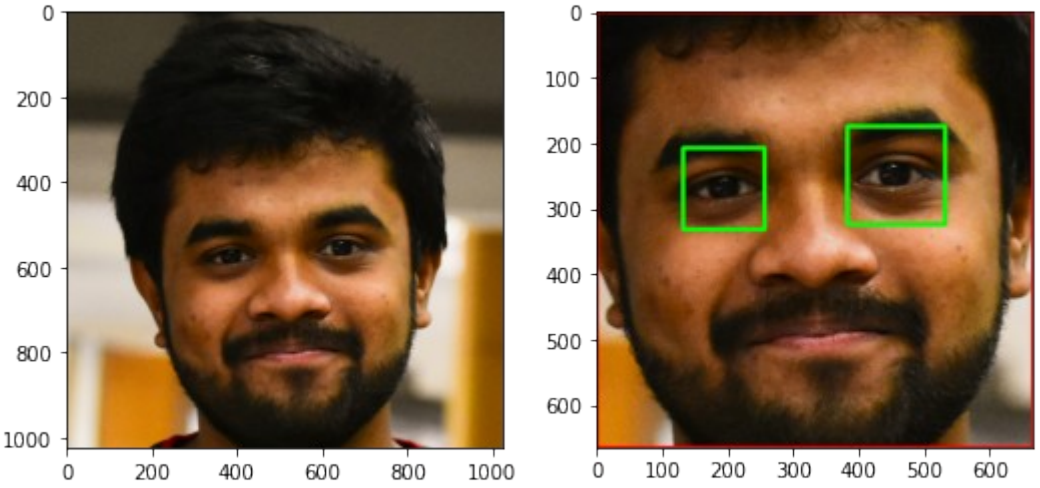
- **Region of interest:**

Face region is first cropped to get the region of interest. For this the HAAR feature for face detection is ran over the images. One of the example images is:

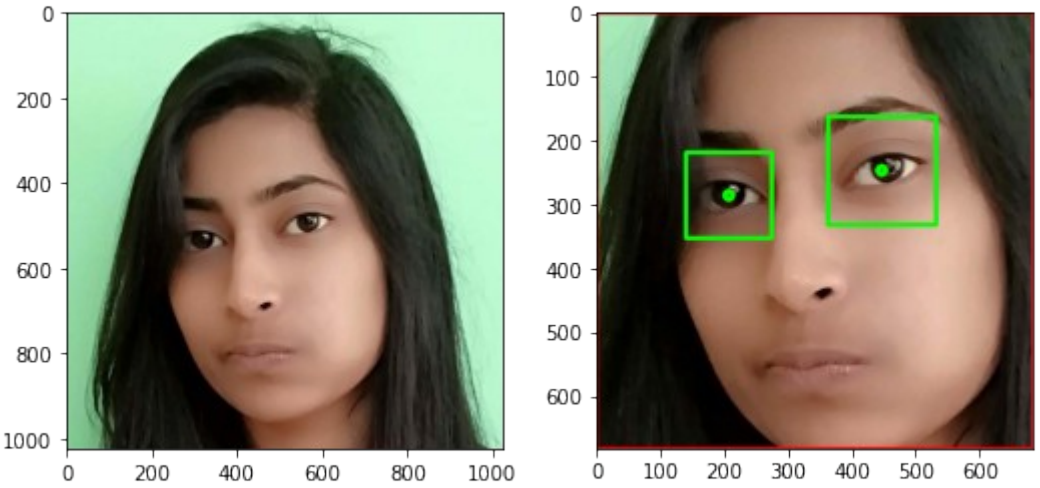


- **Eyes detection and centering:**

  - Eyes are detected again by using HAAR feature detector. One of the example images is shown below:

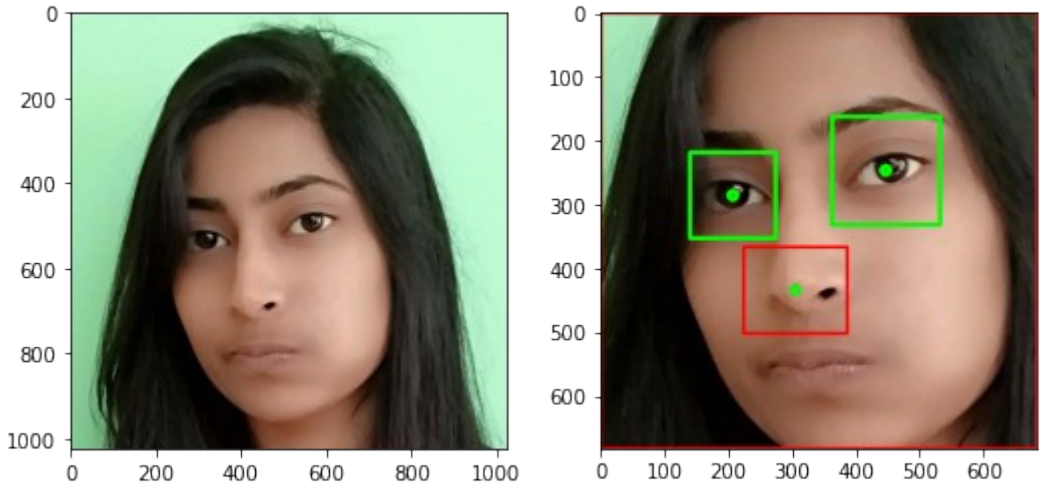


  - Detecting the center of the eye is done by finding the midpoint of the rectangles of the eyes detected as shown above. An example image is shown below:



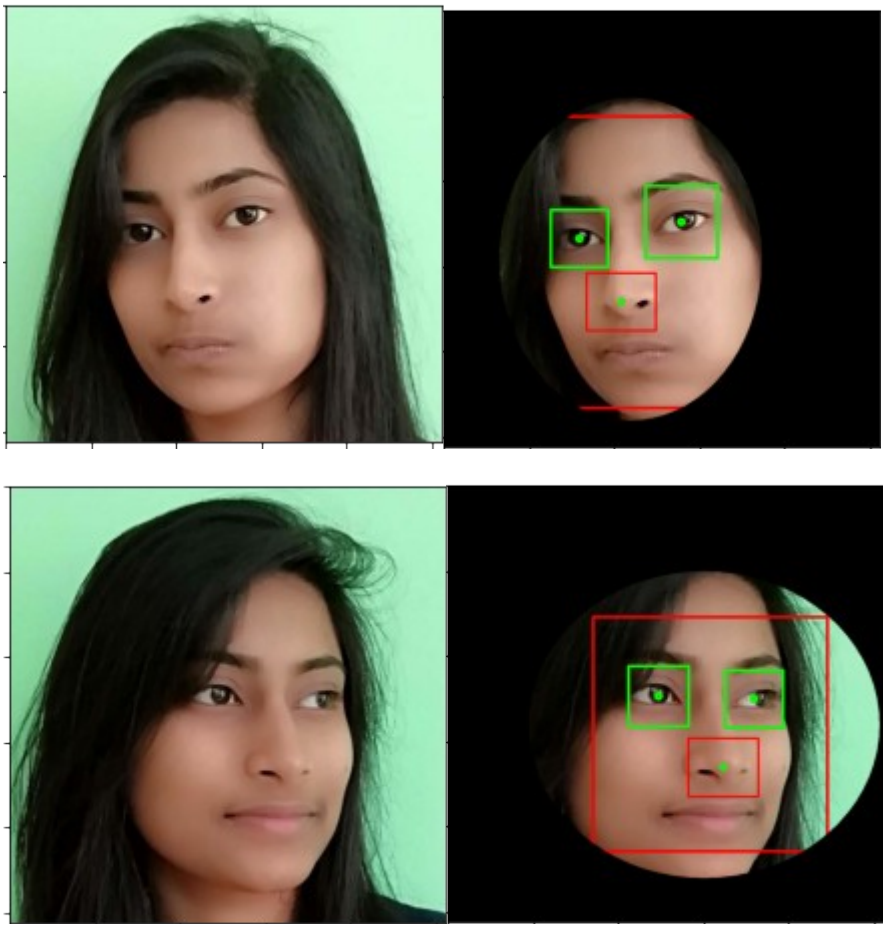
- **Nose detection:**

The nose detection is also done using the HAAR feature detection and it’s center is computed in similar way by finding the centroid if the rectangle detected. Example image is shown below,



- **Oval region cropping:**

The region around the face is unnecessary and hence must be removed. To do this a mask is made in the shape of ellipse. This mask is then put on the image. The example is shown below,



Dimensionality reduction

This is done to extract necessary and relevant features out of the image. Every image is an (n, n) dimensional vector, where every feature is represented by its corresponding eigen weights, And to get the best approximation of the and to consider only the necessary features the dimensionality is reduced by keeping the variance to above 95%. The two techniques used in the program are:

1. PCA (Principle Component Analysis)
2. LDA (Linear Discriminant Analysis)

- ## PCA

The technique is used to reduce the image dimensions and then computing the distances is also known as eigen face detection. PCA is an unsupervised learning algorithm so the number of dimensions are found by hit and trial. Math's behind PCA is explained in steps:

1. The stacked images are first flattened. The code flattens the (64,64) image into 4096*1 array. And then the images are stacked column wise. The faces detected from the image dataset is 624. So the dataset made is having the dimension (4096, 624).

   All training images are stacked as
   $s = \{I_1, I_2, ..., I_{624}\}$

2. The mean of the stacked images is calculated column wise and the mean vector computed is,

   Mean = {u_1 ,u_2 ,u_3 ……..u_{624}}

3. The mean is subtracted from the S matrix to center the data to zero. The resulting matrix will be,

   Ωn $_{= \{}$In – u_n }

   Ω = {Ω1 , Ω2 …….. Ω624}

4. After this the covariance matrix calculated using the formula,

   Cov = Ω.T * Ω

   The covariance calculated will have the dimensions (4096, 4096). This is a very high number to calculate the distance even with high computation power. So, the dimensions are reduced by calculating the eigen values and eigen vectors. The top K eigen values will contain the most variance (>95%).

   The eigen values matrix calculated is represented by V.

   The top K columns of this V matrix is kept.

5. Then the last step is to multiply this reduced V matrix with the centered matrix and the dimensionality is reduced.

   Weight matrix = V * Ω.

   The dimensionality of the matrix is reduced to (624 , K), here K is the relevant eigen vectors to be kept.

- ## LDA (Linear Discriminant Analysis)

This is another dimensionality reduction technique, but it is supervised. It preserves the ability to do classification after reduction of the dimensions. In short it preserves the inter class variance. The sklearn library contains the LDA function that is used in the code. The best dimensions came was 40. The steps of LDA is summarized below:

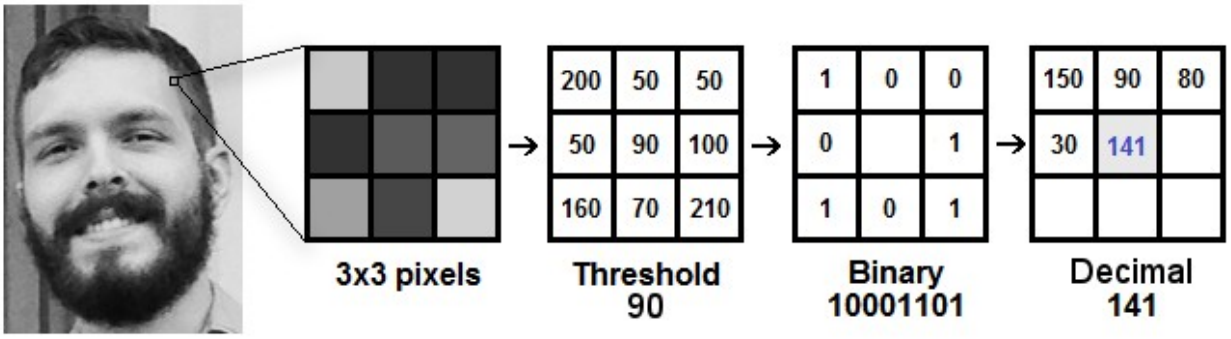
1. Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ($\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix $\boldsymbol{W}$ (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $\boldsymbol{Y} = \boldsymbol{X} \times \boldsymbol{W}$ (where $\boldsymbol{X}$ is a $n \times d$-dimensional matrix representing the n samples, and $\boldsymbol{y}$ are the transformed $n \times k$-dimensional samples in the new subspace).

LBP (Local Binary Pattern)

It is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It first creates an intermediate image which best describes the original image in a better way. For this the algorithm uses a sliding window, based on the radius and neighbours.

Taking an example image,



The above process is described in the following steps:

1. The small part (3*3 in this example) of a gray scale image is taken.

2. In this the centre value is taken as threshold value. In the above example threshold is 90.

3. For each neighbour a binary value is set according to the following rules,

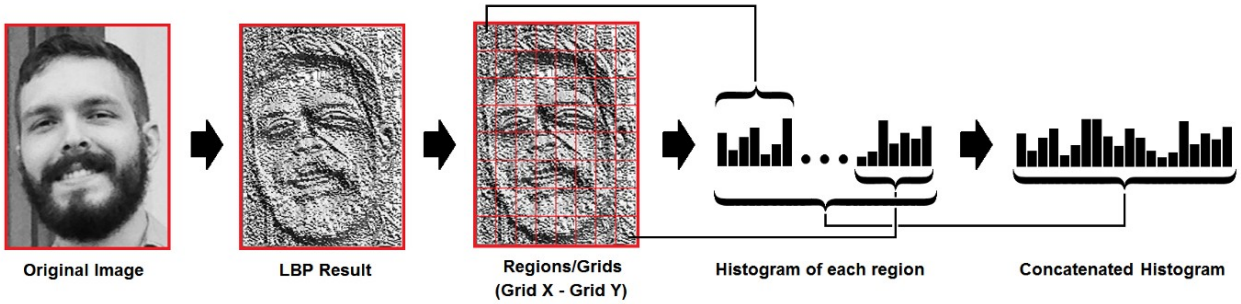
    a. If the pixel value is larger then threshold then set that pixel to 1.

    b. If the pixel value is smaller then threshold then it is set to 0.

4. This step is very important. This takes all the binary values in the clockwise order. The binary value for the above image is shown. The binary value is converted to the decimal value an put in the central pixel.

The resulting image will better represent the original image.

Next the histograms are computed in grids. As shown in the image below,



The histograms of the computed images are then compared using and distance matrix like euclidean distance, chi square or absolute value. The euclidean distance is used the code to compare the histograms.

Accuracy comparison

LDA:

Top 1 : 60.45%

Top 3 : 74.84%

Top 5 : 80.94%

Top 10 : 87.10%

PCA:

Top 1 : 69.68%

Top 3 : 80.64%

Top 5 : 87.10%

Top 10 : 90.97%

LBP:

86.45%