Face Recognition using Eigen Faces: Mid Evaluation Report

Team Number: 11
Team Name: 21 din mein CG double
Team Members:

- Tushar Choudhary (2019111019)
- Suyash Vardhan Mathur (2019114006)
- Tejas Chaudhari (2019111013)
- Rutvij Menavlikar (2019111032)

Repository Link: https://github.com/Rutvij-1/Eigen-Faces-SMAI-M21-project

Problem Statement

Developing a computational model of face recognition is quite difficult, because faces are complex, multidimensional, and meaningful visual stimuli. Unlike most early visual functions, for which we may construct detailed models of retinal or striate activity, face recognition is a very high level task for which computational approaches can currently only suggest broad constraints on the corresponding neural activity. Our aim is to develop a computational model of face recognition which is fast, reasonably simple, and accurate in constrained environments such as an office or a household.

Work Done Till Now

Read the paper and understood it's working

Implemented PCA and got the Eigen Faces

- Flattened the image data in order to make manipulations on it as a vector.
- Normalised the face matrix to perform SVD on.

- Performed SVD on the face matrix to get the Eigen Faces.
- Dimensionally reduced the face data using top 20 Eigen Faces.

Implemented Face Recognition using Nearest Neighbour Classifier

- For any image from the testing dataset, we first reduce it's dimensions with the transformation matrix calculated in PCA.
- Then with the coordinates of the projection of this image on face space, we find the image in the training dataset whose projection on face space is nearest to the projection of the test image, and assign the label of that image to the test image.

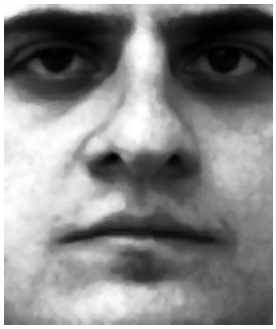
Datasets Used

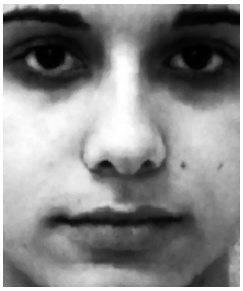
We have used the below two datasets for our tasks:

1. Dataset prepared by Centro Universitário da FEI. (example images from this dataset)
2. Dataset prepared by Yale.

Procedure and Experiments

- Following are the face images that we had taken from the dataset:

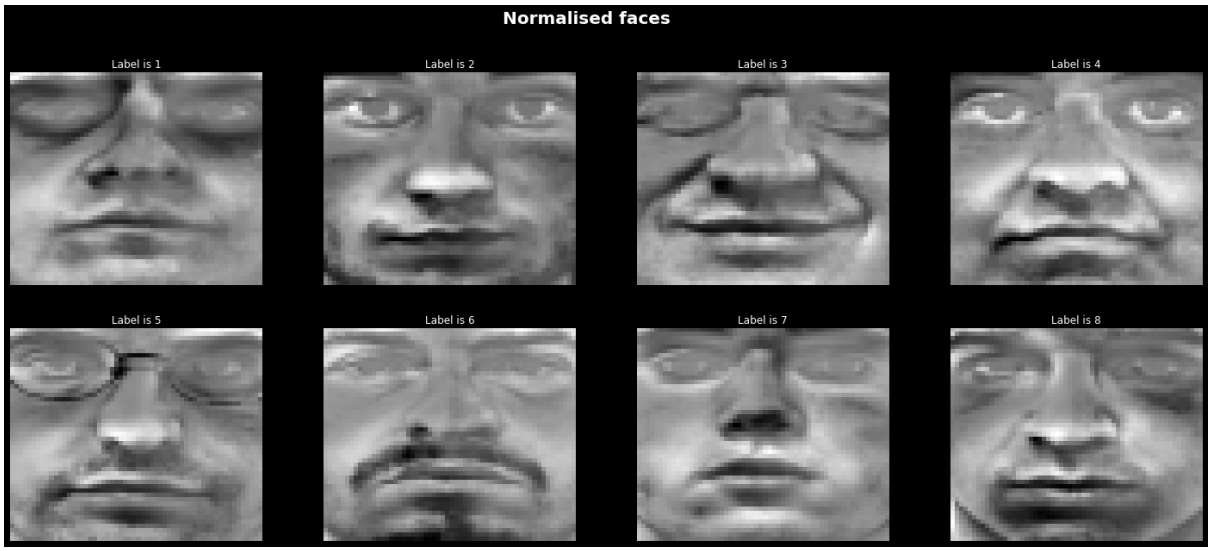





- Clearly, we can see that the facial features are aligned to a certain extent for all the faces. Thus, these images of faces, being similar in the overall configuration aren't randomly distributed in the huge image space, and thus can be described by a relatively low dimensional subspace. This gives us the idea of using PCA for the task.

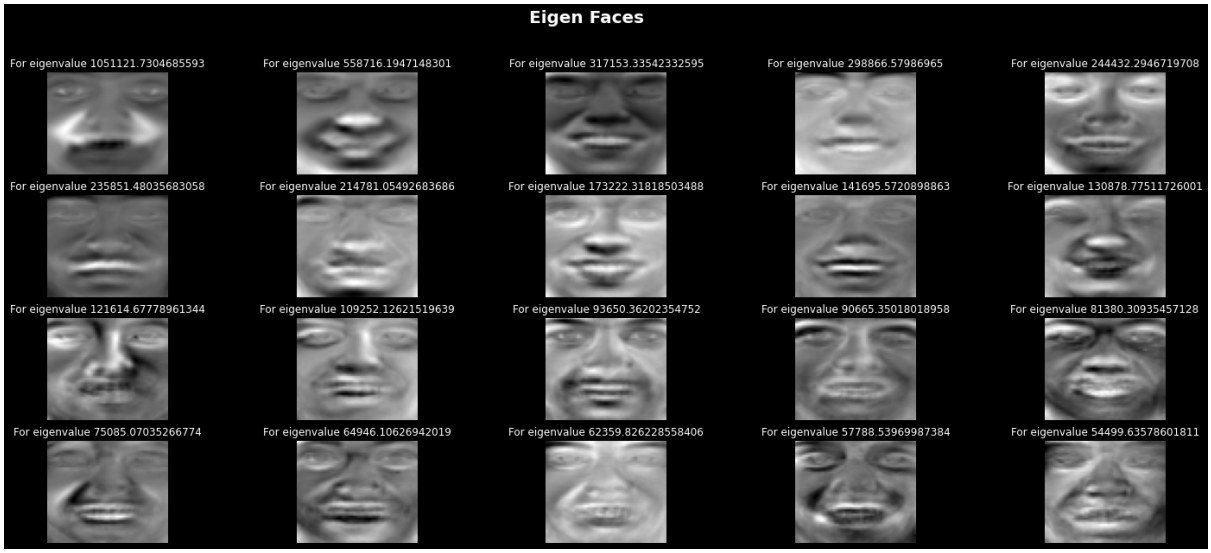
- Now, we find the mean/average face for all the faces, which can be seen below:

- Now, we normalize all the face images from our dataset using this mean face. These were normalized by taking the difference from the mean face, i.e. $\Gamma = \Gamma - \mu$, where μ represents the mean face. The normalized faces can be seen below:



- Then, we obtain M orthogonal vectors, the **eigenfaces** u_n with eigenvalues λ_k of the covariance matrix $C = \frac{1}{M}\sum_{n=1}^{M}\phi_n\phi_n^T = AA^T$, where $A = [\phi_1\phi_2...\phi_M]$**.** We obtained total 2842 eigenfaces after performing the SVD of the covariance matrix.
- **PCA:** Now, we sort the eigenfaces based upon their respective eigenvalues . This was done because the higher the eigenvalue of an eigenface, the more is its component in representing the faces.

- Now, we take the first 20 eigenfaces $A' = [\phi'_1 \phi'_2 ... \phi'_{M'}]$ where $M' = 20$, after sorting them. The first 20 eigenfaces taken are:



- Now, we obtained the dimensionally reduced weights corresponding to the normalized face images. We do this by taking $w = \Gamma \times A'$
- Then, for each image I which is flattened into a vector I_v, we take it's projection on face space by taking $T_v \times A'$
- Then we use nearest neighbour classifier with euclidean distance in the face space to get the label of I.

Pending Tasks

- Implementing Face Detection using Eigenfaces
- Study Accuracy of model with varying number of eigenvectors used for PCA and varying sizes of images.
- Try the same code on more datasets
- Compare performance with other methods, like Cascade classifier.
- Try to reduce the computation cost of SVD by changing the matrix on which SVD is done from $N \times N$ to $M \times M$, where N is $height \times width$ of an image and M is number of images in training dataset.