CMSC426 Project 1

Group members: Guok Wei Jie, Kane Tan, Javier Tan

Member contributions

All members attempted the whole project individually according to the case study. The only discussions that we had were regarding hyperparameter testing for the K value and the threshold value for the euclidean distance.

Preprocessing the training images

I read all of the faces from the Train1 file and preprocessed the images by subtracting the mean face image from each training face. Then, I saved the processed images in another file to check that the images look correct. I did not have to change the size of the images as I checked that they were all of the same size. At first, I thought of storing the training images in another file locally as I felt that storing it as a numpy array would be too big and slow. However, I realized that it was rather fast and more easily accessible. I also did not have to deal with grayscale conversion as images were saved as RGB.

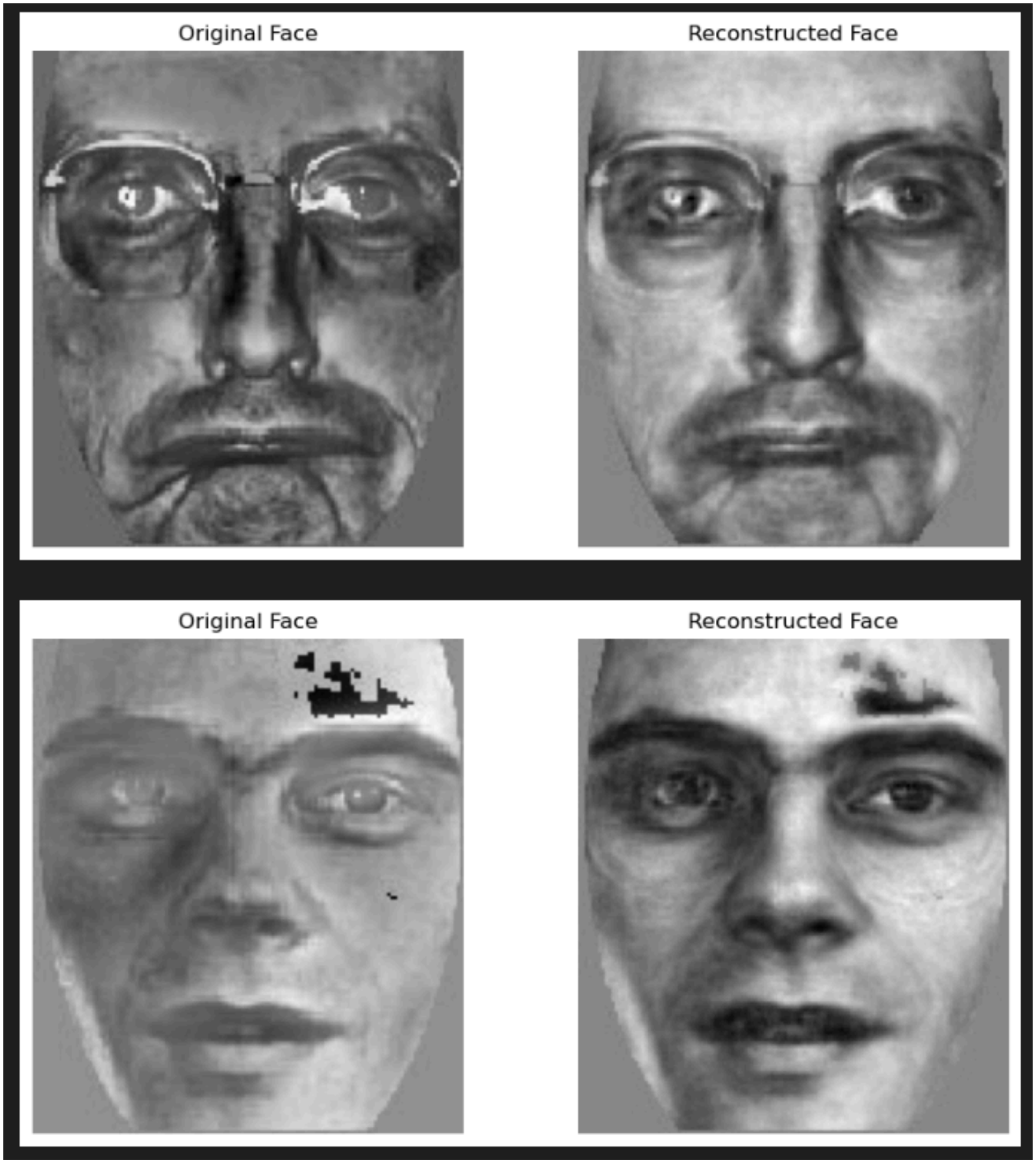
Choosing the lower dimensional basis vectors

Firstly, I flattened the processed faces and computed the covariance matrix for A^TA to take advantage of the smaller M x M matrix, which requires less computational power. Then, I derived the M best eigenvectors of the original AA^T matrix by sorting the eigenvalues and obtaining the corresponding indices of the largest eigenvalues and hence eigenvectors.

Lower dimensional representation of a face

To compare the similarity of the original training face and the corresponding reconstructed face I first projected the original training face onto the eigenvector space to find the weights of the eigenfaces. Thereafter, I did a dot product with the eigenfaces and added the mean face back to obtain the reconstructed face.

With this, I was able to begin experimenting with different K values. My group tested different K values so as to obtain reconstructed faces which look similar to the original face and are both the same person.

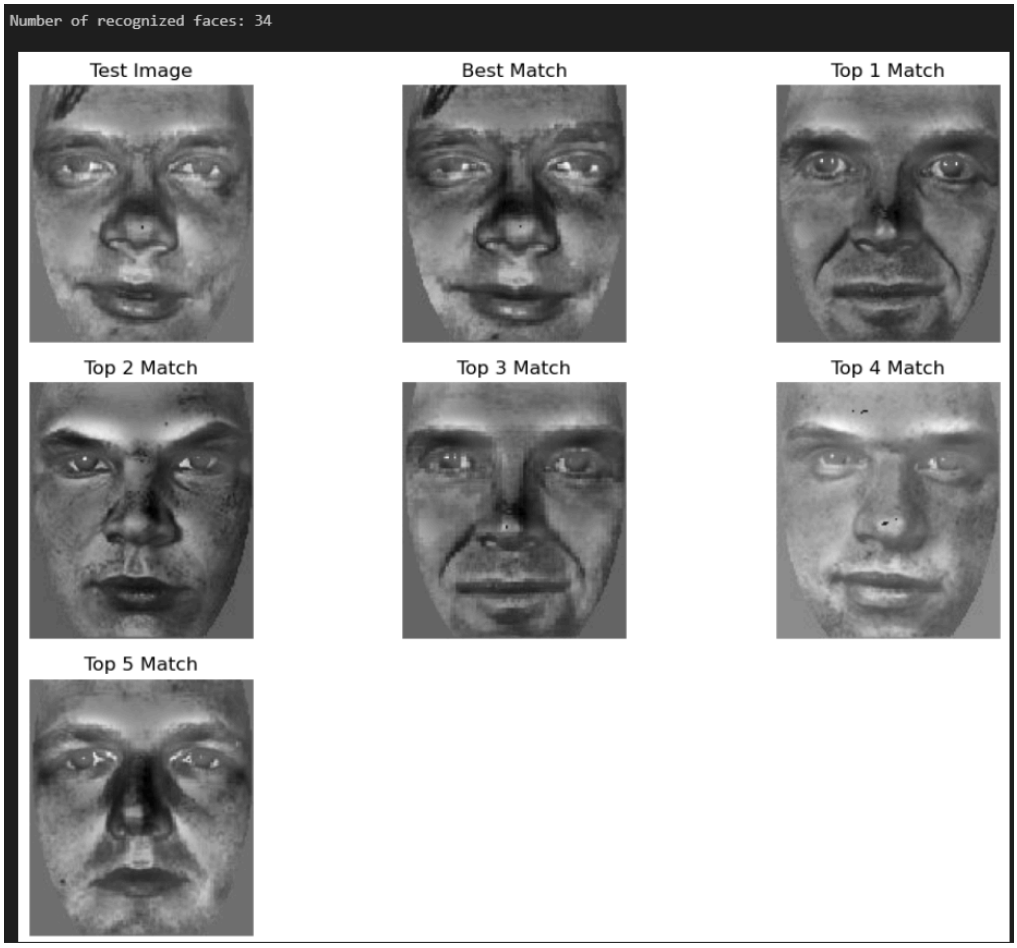


We avoided keeping too many eigenvectors to prevent overfitting to the training data. Additionally, according to a research study that we found[1], it suggested that 15% of training faces was the optimal number, which was about 570 faces. However, I decided to use K=150 purely for this set of test data, as I was able to obtain the same accuracy as with 570 faces. It was also interesting to look at the top few eigenfaces, as it shows which features are prioritized on the training images.

Accuracy of face recognition

Using the 150 best eigenfaces, I was able to achieve an accuracy of 34/44, which is about a 77.3% success rate. To test the accuracy of facial recognition, I first obtained and processed the test images by subtracting the mean face. Then, I projected the test images on the eigenvector space to obtain the weights of the eigenfaces for each test image. Thereafter, I took the minimum of the difference between the weights of each test image with respect to those of the training images.

To obtain the optimal threshold value, I tested different factors of the max of euclidean distances across all test images. I decided to use max(er) * 0.8, which was also supported by the research paper[1]. During testing, something interesting I found was that the threshold value would vary for different K values, as different K values would affect the distribution of distances between faces in the features space. For one of the test images shown below, the best match is the exact same person and exact same image, with the top 5 matches looking similar to the test image as well.

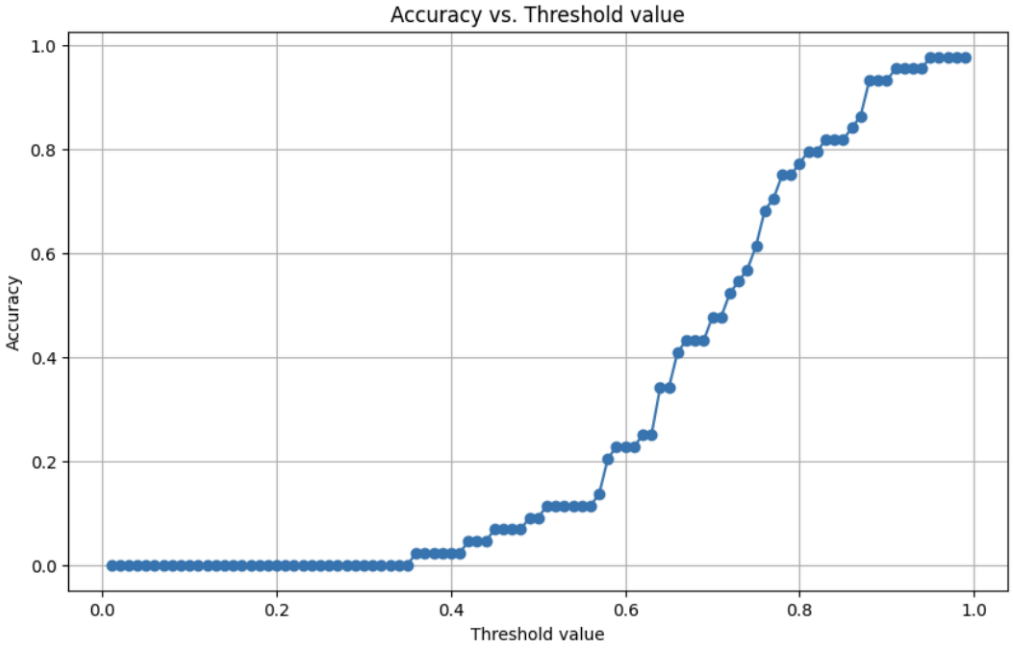


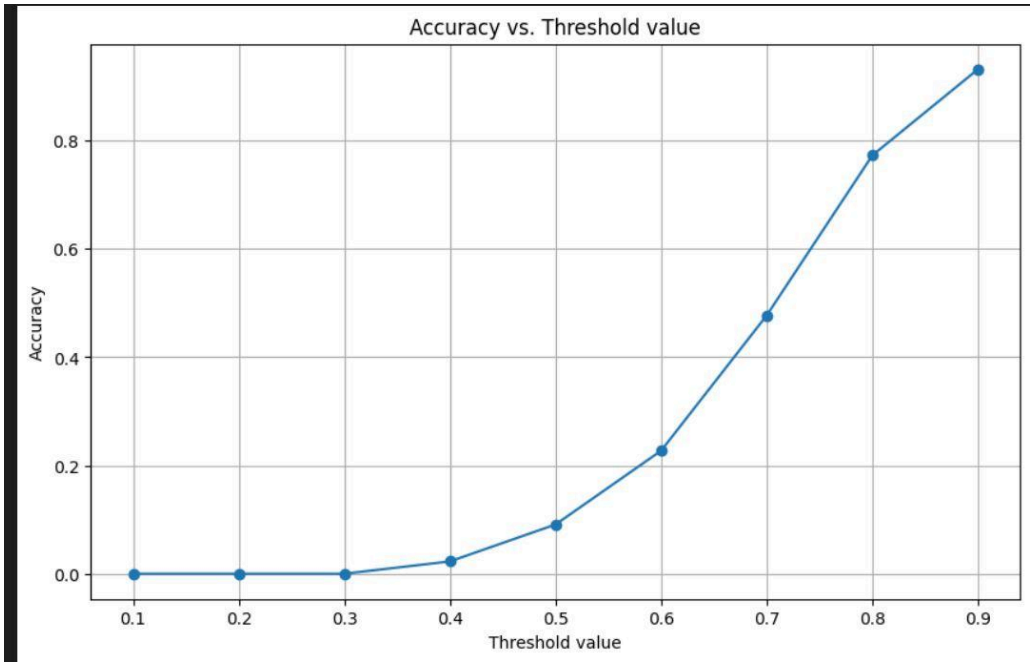
Learning points from the project

I learnt how to make use of eigenvectors in PCA to reduce the dimensionality of the original training images and keep the most important features for facial recognition of test images. I also learnt how to display graphical data effectively using matplotlib to plot images side by side, row by row, etc. The accuracy of facial recognition depends on the composition and size of the data set, and only by testing with more sets of testing data can we obtain a more robust and reliable accuracy ratio. I also learnt how to preprocess data, such as subtracting the mean face and flattening the images. I learnt how to do hyperparameter tuning for the K value and the threshold value, by observing and understanding how they affect recognition accuracy.

Graphs used for testing

Below are some of the graphs that were used for testing the different K values and threshold values.





References:

1. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1e1cfcd1da24ec87a4b59b248e826c14a82c66f8