Shaan Ul Haque(180070053)
Samarth Singh (180050090)
Niraj Mahajan (180050069)

Question 6

5.1: Detecting Faces Outside our Trainset

The score that we used in the previous questions to classify an image to a person was the Euclidean Distance between the coefficients (for the PCA transform) of the test image and images in the train set. We thus select the image from the train set (say I) whose coefficients are closest to the coefficients of our test image (say I') and assign this image to the class to which the said image I belongs. But if draw our test images from the samples of unknown people (ie from out of the train set) then this image will have a high Euclidean distance from all the images in the train set (or gallery). Hence we can exploit this phenomenon to detect images outside the dataset. Our algorithm uses thresholding to classify images outside the trainset

The dataset was already divided into two parts - train and test in Q4 of this assignment. In our algorithm, we will need to fine tune the threshold hyper parameter, and hence we will further divide the data to create a validation dataset. Note that we cannot fine tune any hyper parameter on the test set, (which would be not just improper but ghastly erroneous), hence the creation of a validation set is a must. So our data is divided as follows:

- **Train set** - First 6 images for 32/40 people
- **Validation set** - 7th image for 32/40 people, and the first 4 images for the remaining 8/40 people
- **Test set** - 8th-10th images for 32/40 people, and the first 5th-10th images for the remaining 8/40 people

So, effectively the trainset has 192 images, the validation set has 64 (32+32) images, and the test set has 144 (96+48) images

The algorithm that we used is simple - For any test image I' compute the coefficients that are obtained using PCA, and get it's distance from the coefficients of all the images in the gallery (train set). Get the minimum such distance for our test image I'. If this distance d lies above some threshold λ, then the image I' is outside the trainset, and if it is lower than the same λ, then this image belongs to a person within the datset.

Results

As our threshold parameter increases, the number of False Positives increases, while te number of False Negatives Decreases. In a way, False Positives carry much more weight as compared to False negatives, as it is sometimes okay if a person in the system is not able to pass the security, but it can be detrimental if a person outside the database can pass the security.

By finetuning on our validation dataset, we establish that the optimal value for our thresholding parameter λ is 2090
At $lambda = 2090$,

- Number of False Positives = 2 out of 48 Negatives
- Number of False Negatives = 32 out of 96 Positives

This means that from our test set, out of the 48 images that lie outside the dataset, only two were classified as "inside", while out of the 96 images that lie inside the dataset, 32 were classified as "outside". This is somewhat acceptable, as we are ensure a tighter security by not allowing images that are outside the dataset to be classified as otherwise.

5.3: Usage of code

- Execute the **myMainScript.m** function to display the results and the plots. This takes less than 1 second.
- Since the dataset is being used for Q4, Q5, Q6 in the assignment, we have kept the datasets in a common directory just inside our submission directory. The ORL dataset is expected to be in a relative directory **'../../datasets/'**, and the Yale dataset is expected to be in a relative directory **'../../datasets/CroppedYale/'**
That is the per person directories should be as follows:
'../../datasets/ORL/s*/' and
'../../datasets/CroppedYale/yaleB/'**
- The **loadOrl.m** function loads the data and classies it further into train, test and validation.
- The **fitPCA.m** function generates the eigenvectors of the data matrix using the *svd* library function.
- The **getPredictor.m** functions returns a struct which stores the relevant eigenvectors, and the subspace transformation operator using the said eigenvectors.
- The **predict.m** functions returns the number of false positives and false negatives on the specified dataset, with the help of the struct generated in the previous part. This function is used while fine tuning the thresholding hyper parameter, as well as while testing.
- Lines 15 through 20 in the **myMainScript.m** file have the code that is used to fine tune the thresholding parameter. It is currently commented out to prevent it from flooding the stdout.