Computer Vision
Face Recognition – Eigen Faces | Fisher Faces

Part I: Face Recognition using Eigen Faces

1. Algorithm

a. Process(reshape) labeled training images (training).

b. Find mean μ for the processed images '*x*'. In our case we would calculate mean for the subset that we need to train and that will be the global mean. Subtract this mean from all the processed images to get a normalized matrix, A.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

c. Calculate covariance matrix, $\Sigma = AA^T$.

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

Here we can use the trick to reduce the computation and calculate pseudo covariance matrix, $\Sigma(\text{pseudo})=A^TA$.

d. Find d principal components (eigenvectors of Σ) based on the direction of maximum intensity variation. This can be done using the singular value decomposition function in Matlab.

e. Multiplying the transpose of this by the normalized matrix A, we get the eigenfaces.

$$W_{pca} = (w_{i1}, \ldots, w_{ik}) = (u_1^T(x_i - \mu), \ldots, u_k^T(x_i - \mu))$$

f. Given novel image x (testing). Project it onto the subspace.

$$(w_1, \ldots, w_k) = (u_1^T(x - \mu), \ldots, u_k^T(x - {}^\wedge\mu))$$

g. Classify as closest training face in k-dimensional subspace.

h. For reconstruction we use the following equation.

$$\hat{x} = \mu + w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + \ldots$$

Here, $(w_{1..}w_k)$ are weights obtained after projecting the test image on the *PCA* subspace, and $(u_1\ldots u_k)$ are the the eigenfaces.

2. Eigen Faces

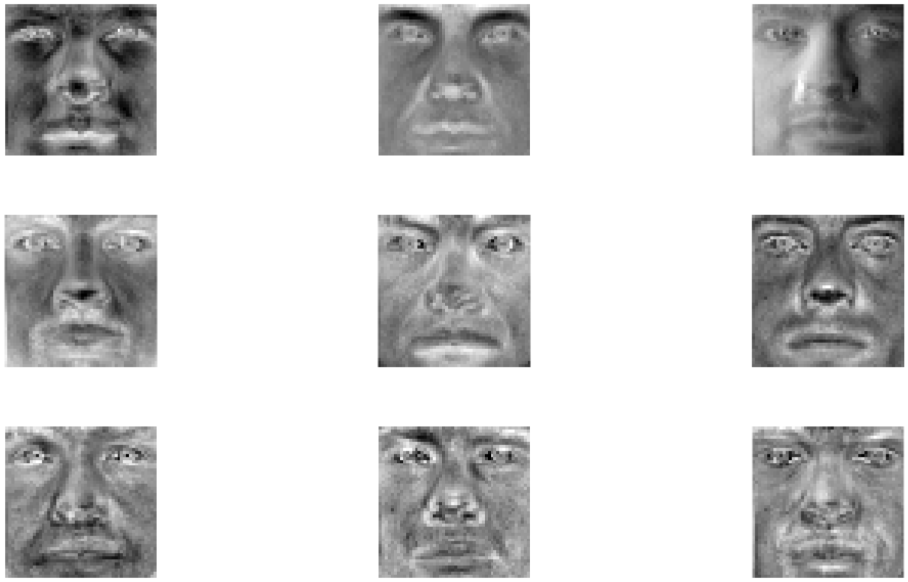

Figure 1: Eigen Faces for reduced dimensions, d=9.

Figure 2: Eigen Faces for reduced dimensions, d=30

3. Percentage Accuracy, Percentage Error, and other Results.

For d=9, following are the percentage accuracies:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	100	72.5	26.667	12.143	9.4737
'Subset 2'	60	100	29.167	15	13.684
'Subset 3'	8.5714	13.333	100	16.429	15.263
'Subset 4'	10	8.3333	13.333	100	10.526
'Subset 5'	4.2857	9.1667	19.167	19.286	100
'Subset 1+5'	100	31.667	15	21.429	100

For d=9, following are the percentage errors:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	0	27.5	73.333	87.857	90.526
'Subset 2'	40	0	70.833	85	86.316
'Subset 3'	91.429	86.667	0	83.571	84.737
'Subset 4'	90	91.667	86.667	0	89.474
'Subset 5'	95.714	90.833	80.833	80.714	0
'Subset 1+5'	0	68.333	85	78.571	0

For d=30, following are the percentage accuracies:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	100	74.167	27.5	12.143	10
'Subset 2'	61.429	100	30.833	15.714	13.158
'Subset 3'	8.5714	14.167	100	16.429	14.737
'Subset 4'	10	8.3333	15	100	11.579
'Subset 5'	4.2857	9.1667	19.167	21.429	100
'Subset 1+5'	100	35	15.833	22.143	100

For d=30, following are the percentage errors:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	0	25.833	72.5	87.857	90
'Subset 2'	38.571	0	69.167	84.286	86.842
'Subset 3'	91.429	85.833	0	83.571	85.263
'Subset 4'	90	91.667	85	0	88.421
'Subset 5'	95.714	90.833	80.833	78.571	0
'Subset 1+5'	0	65	84.167	77.857	0

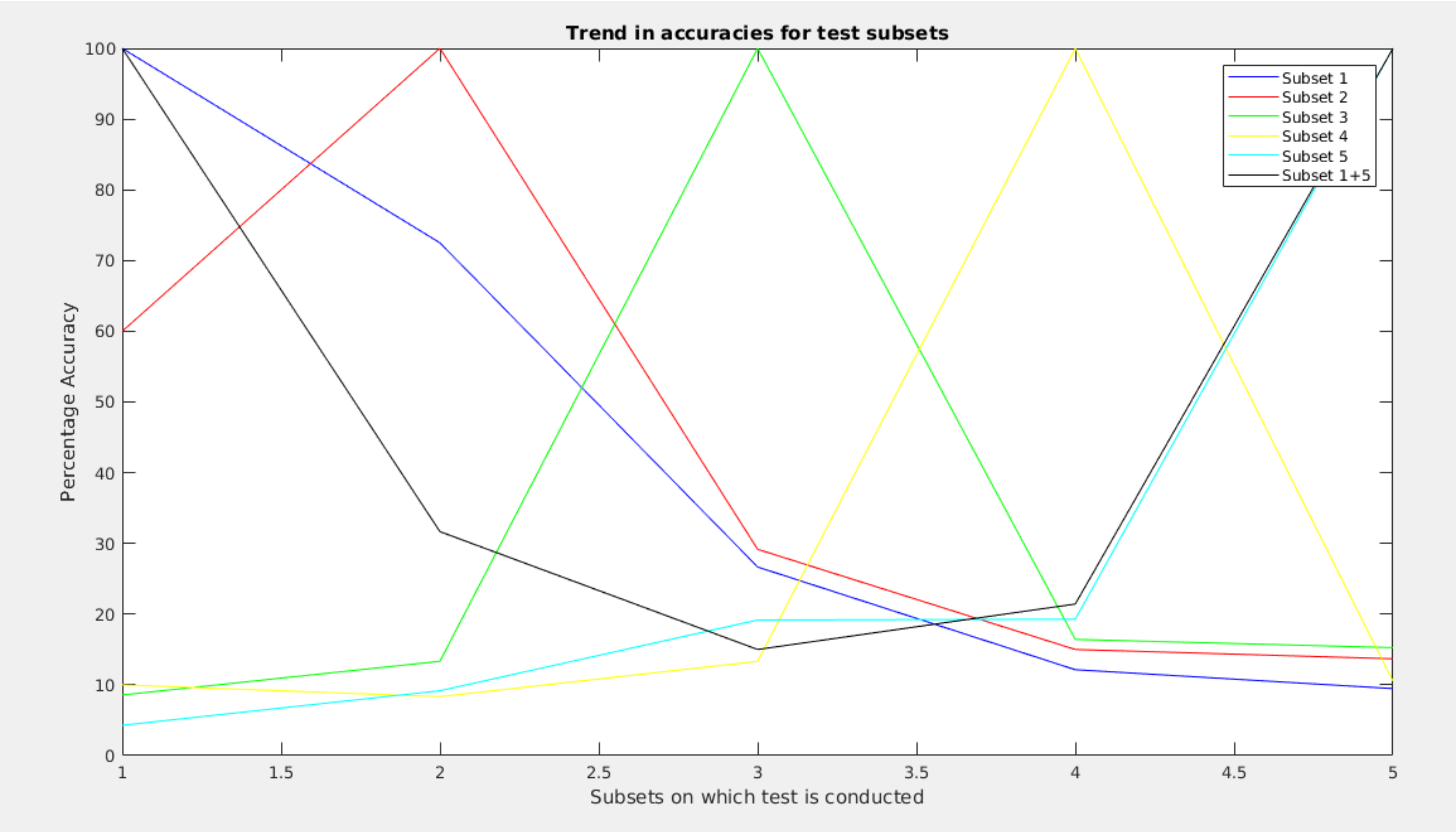

Figure 3: Trend of accuracies for d=9. NOTE: X axis are the subsets number 1 to 5 on which test is conducted.

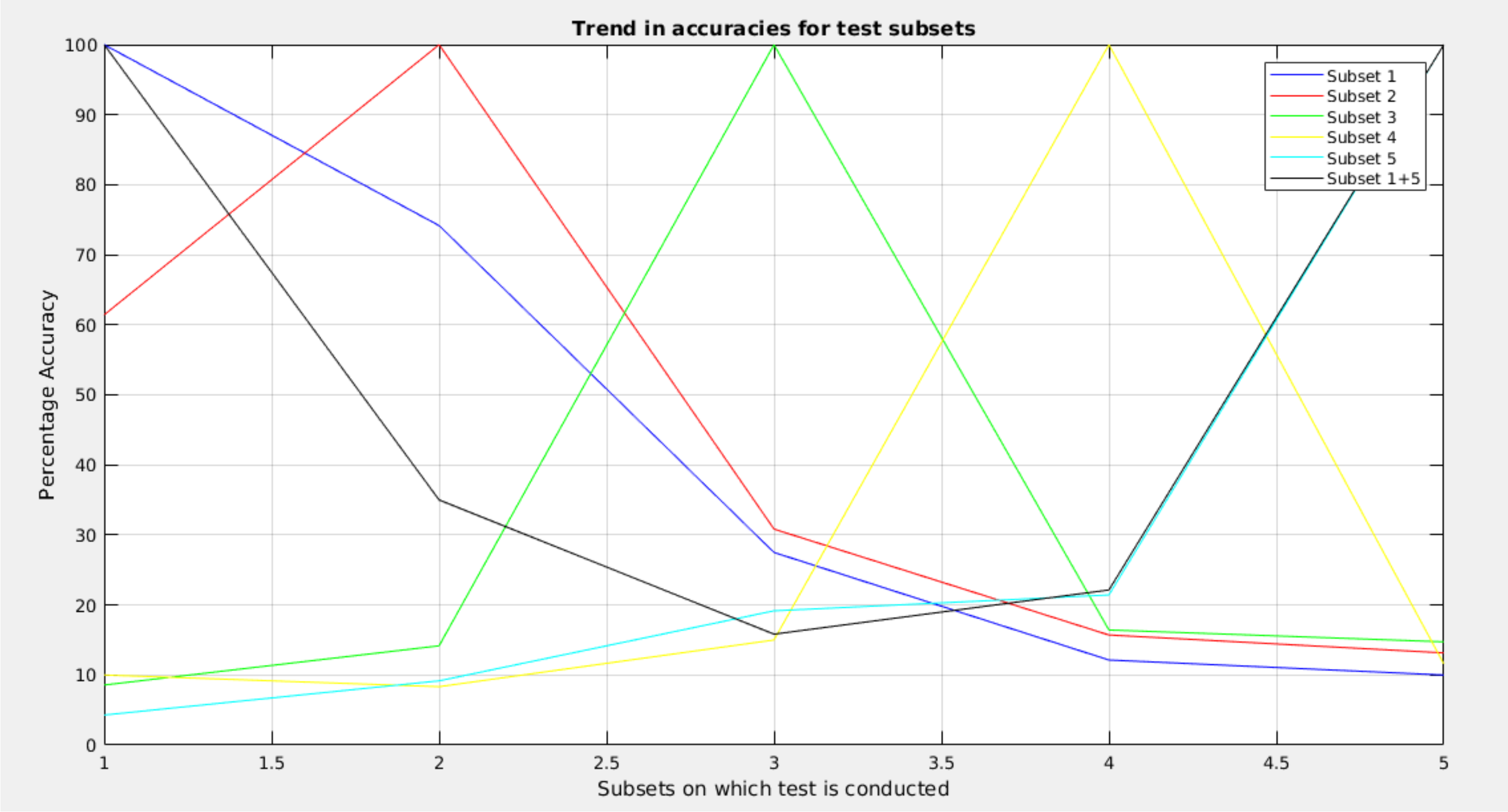

Figure 3: Trend of accuracies for d=30. NOTE: X axis are the subsets number 1 to 5 on which test is conducted.

4. Reconstruction

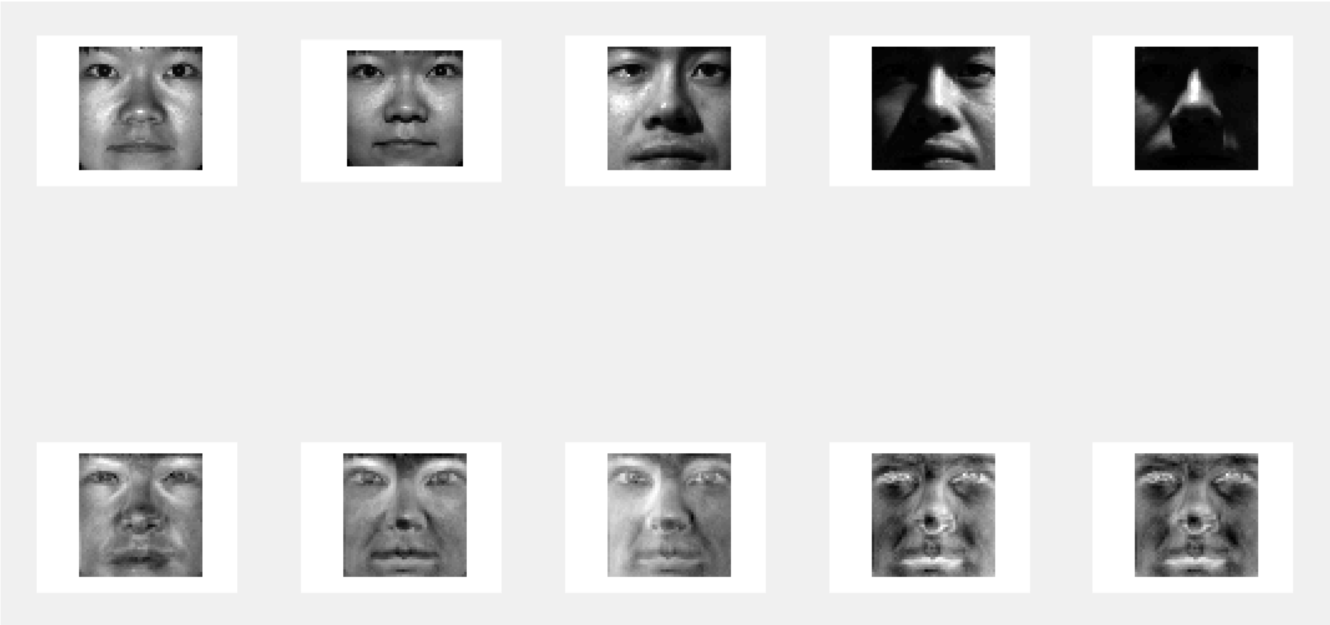

Figure 5: Original and Reconstructed Images for reduced dimensions, d=9

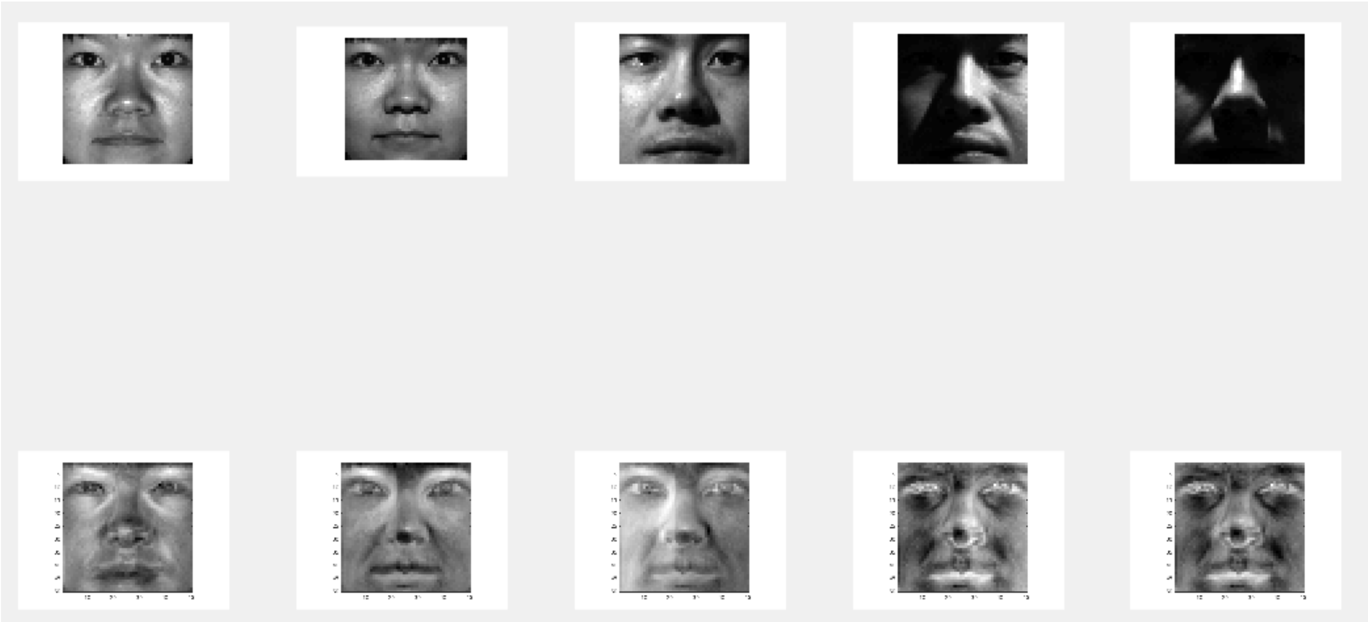

Figure 6: Original and Reconstructed Images for reduced dimensions, d=30

5. Inference

The above results are obtained from the Matlab code, main_eigenFaces. This is the main fucntion, calling other functions to compute required percentage errors.

When we collectively train subset 1 and subset 5, perfect matches are obtained when tested with subset 1 and subset 5 as we would expect.

Also if we eliminate first three Eigen vectors, we would see increase accuracy because the first three eigen vectors are due to high discrimination due to illumination or intensity.

Part II: Face Recognition using Fisher Faces

1. Algorithm

a. Some steps for this will be similar to eigenfaces. Process(reshape) labeled training images (training).

b. Find mean μ for the processed images. In our case we would calculate mean for the subset that we need to train and that will be the global mean. Subtract this mean from all the processed images to get a normalized matrix, A.

c. Calculate covariance matrix, $\Sigma = AA^T$. Here we can use the trick to reduce the computation and calculate pseudo covariance matrix, $\Sigma(\text{pseudo})=A^TA$.

d. Find d principal components (eigenvectors of Σ) based on the direction of maximum intensity variation. This can be done using the singular value decomposition.

e. Multiplying the transpose of this by the normalized matrix A, we get the eigenfaces. $W_{pca} = (w_{i1}, \ldots, w_{ik}) = (u_1^T(x_i - \mu), \ldots, u_k^T(x_i - \mu))$. Until this, everything is similar to the algorithm for PCA.

f. If you have N sample images, $\{x_1, \ldots, x_N\}$, for training subset 1 we have N=70 images. The sample classes will be the number of persons (/unique faces). Here, for subset 1 we have 7 different persons.

g. Compute average of each class:

$$\mu_i = \frac{1}{N_i} \sum_{x_k \in \chi_i} x_k$$

h. Compute average of the entire data:

$$\mu = \frac{1}{N} \sum_{k=1}^{N} x_k$$

i. Compute scatter of each class:

$$S_i = \sum_{x_k \in \chi_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

j. Compute within class scatter:

$$S_W = \sum_{i=1}^{c} S_i$$

k. Compute the scatter between class:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

l. Compute pseudo S_w and S_B:

$$\tilde{\boldsymbol{S}}_{\boldsymbol{B}} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{S}_{\boldsymbol{B}} \boldsymbol{W}$$
$$\tilde{\boldsymbol{S}}_{\boldsymbol{W}} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{S}_{\boldsymbol{W}} \boldsymbol{W}$$

Here, *W=W_{pca} or eigenfaces.*

m. Minimize the following equation using the eig(~S_B,~S_w):

$$\boldsymbol{W}_{\boldsymbol{opt}} = \mathbf{arg}\,\max_{\mathbf{W}} \frac{|\tilde{\boldsymbol{S}}_{\boldsymbol{B}}|}{|\tilde{\boldsymbol{S}}_{\boldsymbol{W}}|} = \mathbf{arg}\,\max_{\mathbf{W}} \frac{|\boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{S}_{\boldsymbol{B}} \boldsymbol{W}|}{|\boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{S}_{\boldsymbol{W}} \boldsymbol{W}|}$$

n. Given novel image x (testing). Project it onto the *FLD* subspace:

$$W^T{}_{opt} = W^T{}_{fld} W^T{}_{pca} \qquad \hat{x} = W_{opt}{}^T x$$

o. Classify by nearest neighbor and reconstruct in a similar way as mentioned in the algorithm for face recognition using eigenfaces.

2. Fisher Faces

Figure 7: Fisher Faces for reduced dimensions, c=10 (Top 10 instead of top 9)

Figure 8: Fisher Faces for reduced dimensions, c=31(Top 31 instead of top 30)

3. Percentage Accuracy, Percentage Error, and other results

For c=10, following are the percentage accuracies:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	100	94.167	70.833	28.571	11.579
'Subset 2'	88.571	100	64.167	23.571	15.789
'Subset 3'	87.143	98.333	100	65.714	17.368
'Subset 4'	27.143	35	60.833	100	24.737
'Subset 5'	11.429	16.667	22.5	35.714	100
'Subset 1+5'	100	82.5	22.5	25	100

For c=10, following are the percentage errors:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	0	5.8333	29.167	71.429	88.421
'Subset 2'	11.429	0	35.833	76.429	84.211
'Subset 3'	12.857	1.6667	0	34.286	82.632
'Subset 4'	72.857	65	39.167	0	75.263
'Subset 5'	88.571	83.333	77.5	64.286	0
'Subset 1+5'	0	17.5	77.5	75	0

For c=31, following are the percentage accuracies:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	100	96.667	42.5	12.143	7.8947
'Subset 2'	100	100	99.167	56.429	16.316
'Subset 3'	68.571	70	100	57.143	24.211
'Subset 4'	11.429	10.833	50.833	100	27.368
'Subset 5'	10	12.5	32.5	61.429	100
'Subset 1+5'	100	90.833	45	62.857	100

For c=31, following are the percentage errors:

Training_Subsets	Test_Subset_1	Test_Subset_2	Test_Subset_3	Test_Subset_4	Test_Subset_5
'Subset 1'	0	3.3333	57.5	87.857	92.105
'Subset 2'	0	0	0.83333	43.571	83.684
'Subset 3'	31.429	30	0	42.857	75.789
'Subset 4'	88.571	89.167	49.167	0	72.632
'Subset 5'	90	87.5	67.5	38.571	0
'Subset 1+5'	0	9.1667	55	37.143	0

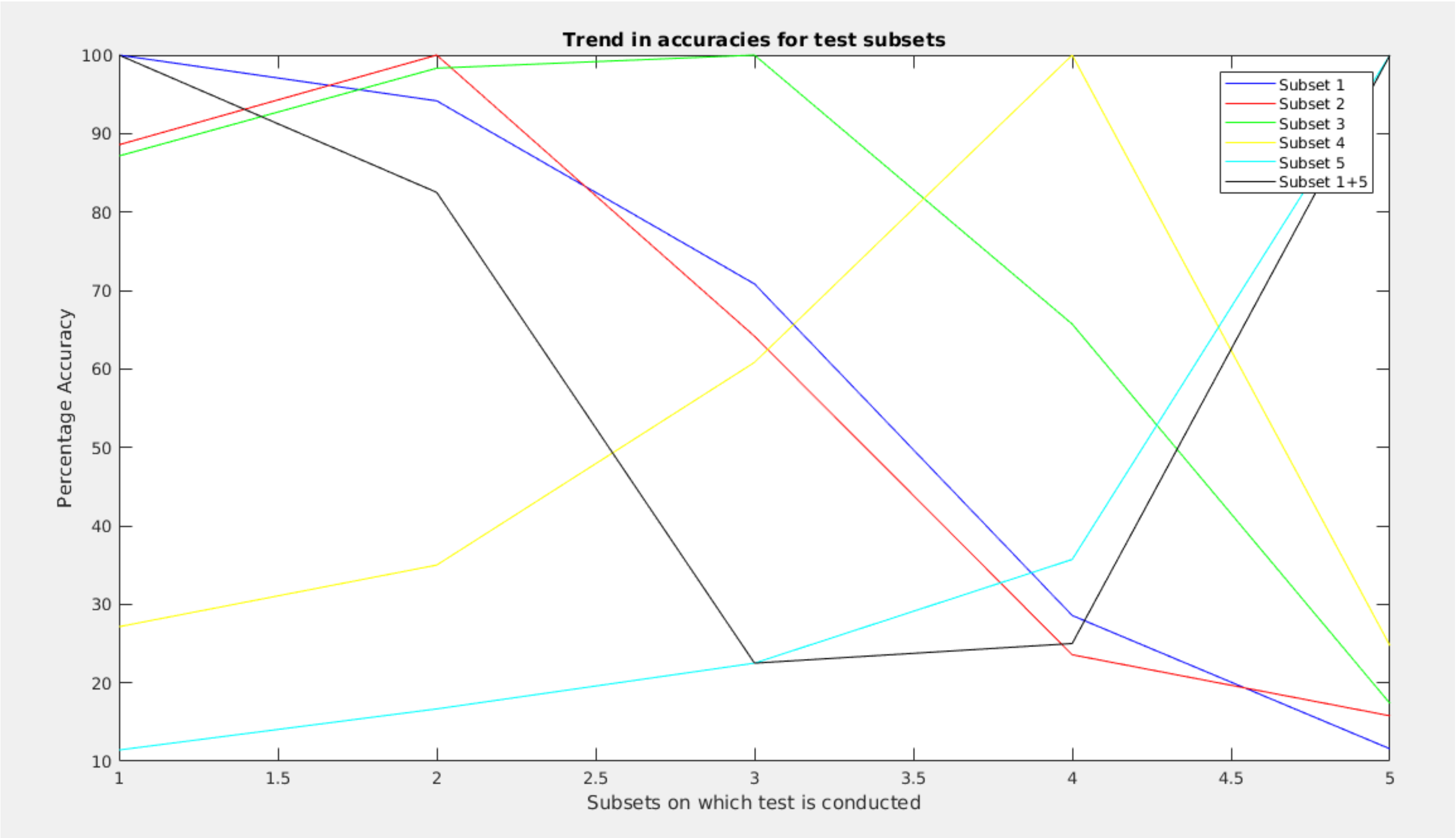

Figure 9: Trend of accuracies for c=10. NOTE: X axis are the subsets number 1 to 5 on which test is conducted

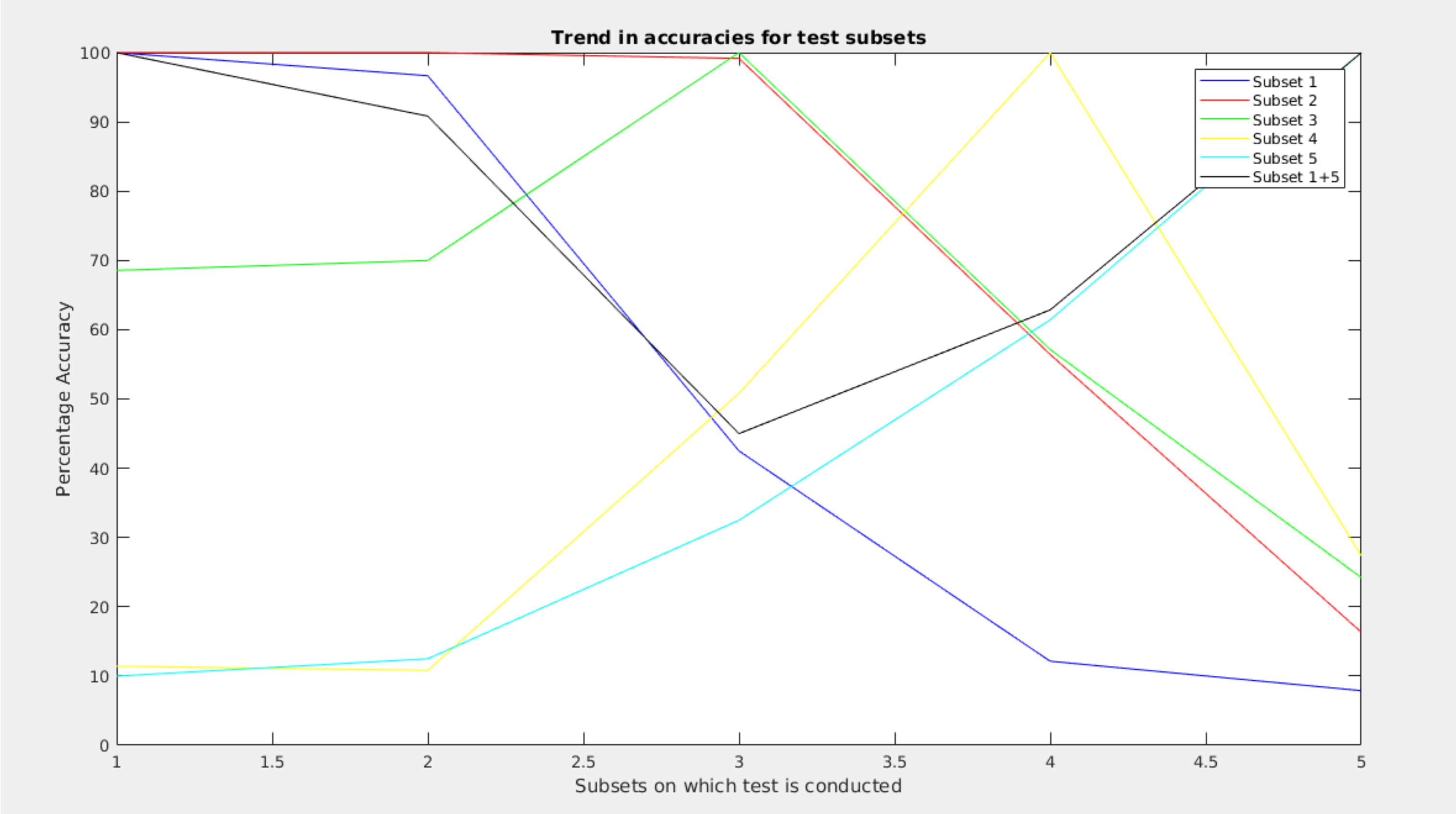

Figure 10: Trend of accuracies for c=31. NOTE: X axis are the subsets number 1 to 5 on which test is conducted

4. Reconstruction

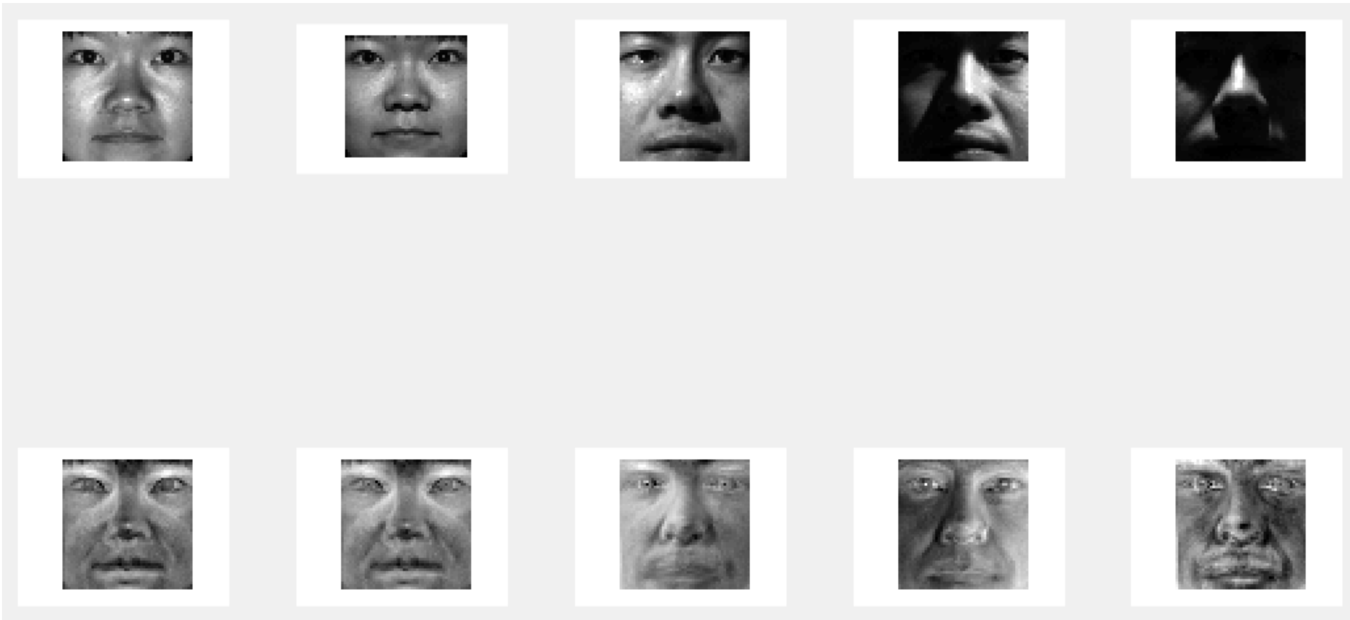

Figure 11: Original and Reconstructed Images for reduced dimensions, c=10

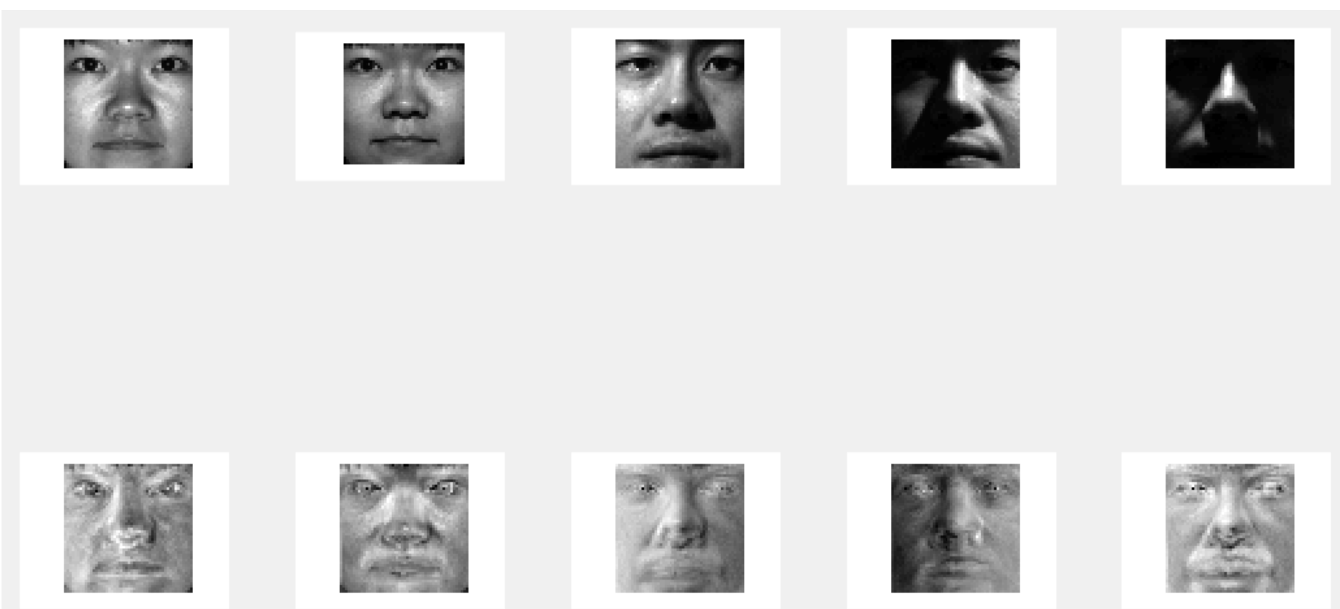

Figure 12: Original and Reconstructed Images for reduced dimensions, c=31

5.Training Subset 1 and Subset 5

The results of this are included in the tables above. Testing subset 1 and subset 5 with the trained image, the percentage error is 0. This is because the trained set has all the features including luminance combined from subset 1 and subset 5. For subset 3 reduces because the intensity/luminance difference between these is high.

6. Inference

The above results are obtained from the Matlab code, main_fisherFace. This is the main function, calling other functions to compute required percentage errors.

When we collectively train subset 1 and subset 5, perfect matches are obtained when tested with subset 1 and subset 5 as we would expect. The accuracy of match between the collectively trained subsets and subset 3 reduces because the intensity/luminance difference between these is high.

Again, if we eliminate first three Eigen vectors, we would see increase accuracy because the first three Eigen vectors are due to high discrimination due to illumination or intensity.

Taking a look at the reconstructed images using FLD and PCA, it is evident that, FLD does not perform well for reconstruction but does well on the discriminating the scatter between and within classes.

NOTE:
Run the code main_eigenFaces.m which takes approximately 83.037 s to compute the results for face recognition using eigenfaces.
Run the code main_fisherFace.m which takes approximately 211.147 s to compute the results for face recognition using fisherfaces.
I have trained each subset and tested with all the subsets out of inquisitiveness. Also, I have plotted the trends followed by the accuracy for each trained and tested subset.
Therefore, you might have to wait a bit until you get the result set.