Imperial College London

Pattern Recognition
Coursework 1

Alberto Garcia Matachana
ag4116
01205901
ag4116@ic.ac.uk

Gustavo Brito Rodriguez
gb1216
01198785
gb1216@ic.ac.uk

November 19, 2019

Contents

Abstract

Face recognition is the challenge of classifying whose face is in an input image. This is different from face detection, where the challenge is determining if there is a face in the input image. A naïve way of accomplishing this is to take the new image, flatten it into a vector, and compute the Euclidean distance between it and all of the other flattened images in our database. The downsides of this approach are the memory of the database (more images, more memory required) and the image sizes. For a single $m \times n$ image, one would have to flatten it out into a single mn vector to feed into our neural network as input. For large image sizes, this will most certainly hurt run-time complexity. The objective of this report is to take high-dimensional images and boil them down to a smaller dimensionality while retaining the essence or important parts of the image through different methods and compare their performance. All the calculations were carried out in Python.

1 Question 1 - Computationally Efficient Eigenfaces

1.1 Data Partition

The data set provided (face.mat) consists of 10 raster-scanned 46x56 (2576 pixels) grey-scale face images for each of 52 individuals. All the 520 images are frontal, but the facial expressions and directions of gazes are varied.

Following the example stated in the coursework definition, 8 images per individual were taken to form the training data set (416 total) and 2 were taken to form the test data set (104 total). The allocation of these images was done randomly in order to improve generalisation of the algorithm[4]. For sample training data please refer to Appendix A.1.

Each image $n \in N = \{1, 2, ..., 416\}$ in the training set, was represented as a single vector $x_n \in \mathbb{R}^D : D = 2576$. The training data set matrix was therefore $X_{train} \in \mathbb{R}^{N \times D}$.

1.2 Batch PCA

Batch Principal Component Analysis (PCA) is the first method that will be explored. Its objective is to transform the correlated input, in the form of images, into linearly uncorrelated Principal Components. These components are calculated in such a way so that the first one accounts for as most variance as possible between the input images. Each subsequent component accounts for the most variance possible with the constraint that it is orthogonal to every preceding component[1].

The first step in Batch PCA is to calculate the average face $\bar{x}$ (can be seen in Appendix A.3):

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{1}$$

consequently, one is then able to calculate the matrix A defined as:

$$A = [\phi_1, \phi_2, ..., \phi_N] \in \mathbb{R}^{N \times D} \tag{2}$$

where $\phi_n \in \mathbb{R}^D$ is the difference between an image n and the average face $\bar{x}$, i.e.:

$$\phi_n = x_n - \bar{x} \tag{3}$$

The purpose of matrix A, is that one can then compute the covariance matrix S given by:

$$S = \frac{1}{N} A A^T \in \mathbb{R}^{D \times D} \tag{4}$$

The normalised eigenvectors of S, $u_i : i = \{1, 2, ..., D\}$, and their corresponding eigenvalues λ_i, which will satisfy the equation $Su_i = \lambda_i u_i$, will form the eigenspace from which all the images in X_{train} can be reconstructed through:

$$\tilde{x}_n = \bar{x} + \sum_{i=1}^{D} a_{ni} u_i \tag{5}$$

where $\tilde{x}_n$ is the reconstruction of image n and a_{ni} is the weight corresponding to eigenvector i for image n. These eigenvectors are also known as the eigenfaces (for examples refer to Appendix A.4). It is important to note that there will only be D eigenfaces as long as S is full rank and therefore has non-zero eigenvalues.

When calculating all the eigenvectors of S, there was a total of 2160 non-zero eigenvalues (defined as anything below 0.1). The computation took a total of 17.7 seconds due to the high dimensionality of S and hence, the high complexity of computing its eigenvalues and eigenvectors.

1.3 Low-dimensional Batch PCA

The Low-dimensional Batch PCA method allows to significantly reduce the time complexity of the algorithm through the computation of the matrix S_{low} given by:

$$S_{low} = \frac{1}{N} A^T A \in \mathbb{R}^{N \times N} \tag{6}$$

The dimensions of S_{low} ($R^{416 \times 416}$), are significantly reduced from that of S ($R^{2576 \times 2567}$) and the eigenvectors of v_i of S_{low} have a one to one mapping to those of S through $u_i = Av_i$. This method will be only reduce the complexity, however, as long as $N < D$, or, in other words, as long as there are less training images than pixels in an image.

Using this method, one is only able to calculate a total of 416 eigenfaces compared to the 2576 of S, albeit much faster. The calculation of all the eigenvalues v_i of S_{low} took a total of 0.2 seconds (in comparison to 17.7s of Batch PCA) and had 0 non-zero eigenvalues. These 416 eigenvalues of S_{low} were the same as the largest 416 eigenvalues of S.

The biggest eigenvalue, u_1, was significantly bigger than the rest and is therefore not included on the graph below. However, to see the graph with all eigenvalues please refer to Appendix A.2.

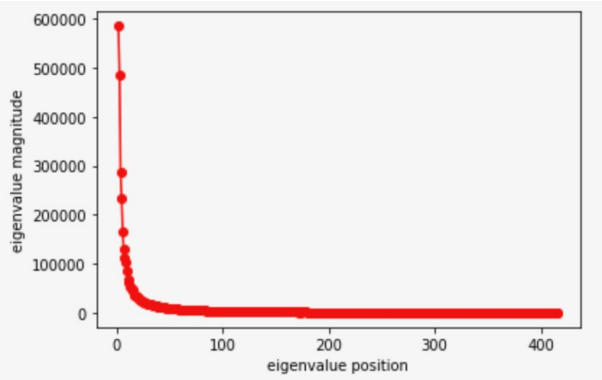

Figure 1: Graph of eigenvalue magnitude $|\lambda_i|$ against i, excluding $i = 1$

From Figure 1, it can be appreciated how the first 20 eigenfaces hold much more information than the rest, as mentioned at the start of the section. This will be discussed further in 1.4 and 1.5.

1.4 Image Reconstruction

From here onward, the eigenvalues used, u_i, are the 416 obtained through the Low-Dimensional Batch PCA method in 1.3.

As stated in Equation (5), all that is required to reconstruct $\tilde{x}_n$ is: the average face, $\bar{x}$; the eigenfaces u_i and the weights a_{ni}. The average face and eigenfaces have already been calculated with the training data set. Therefore, one must calculate the weights a_{ni} defined as:

$$a_{ni} = \phi_n^T u_i \in \mathbb{R} \tag{7}$$

For a single image n, the vector ϕ_n, which represents the variance with the average face (defined in Equation (3)), has to be projected into the whole eigenspace by multiplication with each eigenface u_i forming the vector $w_n : w_n = \{a_{n1}, a_{n2}, ..., a_{nM}\}$.

The number of eigenvalues used for reconstruction M can be varied, which lowers both memory and time consumption (refer to Table 1).

M	Run-time(s)	Memory(MB)
416	3.07	17.15
100	0.73	4.23
10	0.09	0.45
5	0.05	0.28
1	0.01	0.04

Table 1: Run-time and memory consumption, for weights and eigenfaces, with different M

However, reducing M (and hence the number of eigenfaces used) will also reduce the quality of the image (for sample reconstructions refer to Appendix A.3) as can be seen on Table 2 below, where the reconstruction errors for different values of M are shown. The error used is defined as:

$$e_M = \frac{1}{N} \sum_{n=1}^{N} ||\tilde{x}_n - x_n|| \in \mathbb{R} \tag{8}$$

M	e
416	15
100	18
50	22
10	29
5	32

Table 2: Reconstruction error e with different M

As stated previously in 1.3, the reconstruction error remains similar from $M = 416$ to $M = 50$ and afterwards increases sharply due to the progressive removal of large eigenvalued eigenfaces.

1.5 Nearest Neighbour Classification

The Nearest Neighbour (NN) Classification algorithm is used to classify images into one of the categories in the training set by finding the label l of image n which satisfies:

$$n = \underset{i}{\text{argmin}} ||w - w_i|| : w, w_i \in \mathbb{R}^M \tag{9}$$

One can see from the equation above, that, once again, changing the value of M will result in differences in the L2 norm: $||w - w_i||$. The confusion matrix for different values of M were calculated and can be seen below.

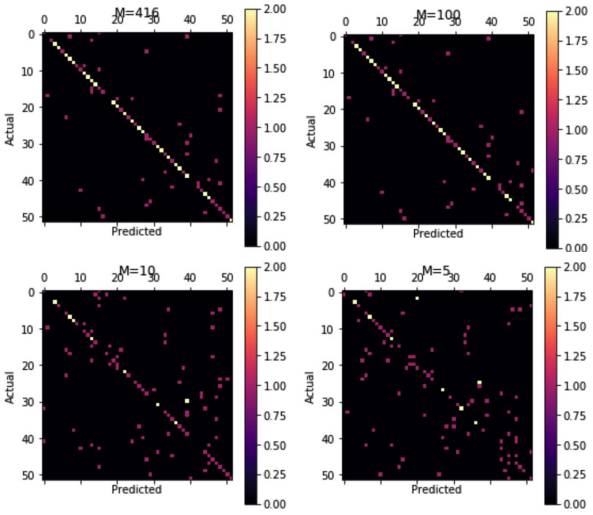

Figure 2: Confusion Matrices for M values of 416, 100, 10 and 5 respectively

As can be seen above, the accuracy of the algorithm declines sharply with very low values of M; but, it must

also be noted that there is virtually no change in performance from $M = 416$ to $M = 100$.

The different accuracies of the algorithm with the different values of M are shown below:

M	Accuracy %
416	68.27
100	68.30
50	67.03
20	58.65
10	41.34
5	31.73
1	1.92

Table 3: Accuracy percentages of different M

Once again, as mentioned in 1.3 and 1.4, the classification accuracy decreases sharply for values of $M < 50$, further reinforcing the significance of eigenfaces corresponding to larger eigenvalues. For sample success and failure please refer to Appendix A.5.

Overall, increasing the number of Principal Components used increases the recognition accuracy, since there is more information to reconstruct and more face variance to account for.

2 Question 2 - Incremental PCA

2.1 Data Partition

For Question 2, the training set used for Question 1 (consisting of 416 images) was divided into 4 subsets of 104 images, namely $X_A, X_B, X_C, X_D \in \mathbb{R}^{104\times D}$. The subsets were divided in such a way that every X_i contained 2 images of each face. The test data set remained the same as the one specified for Question 1 in 1.1. Each image is still being represented as a single vector $x_n \in \mathbb{R}^D : D = 2576$.

2.2 Incremental PCA

Batch PCA performed in Question 1 makes the assumption that all the images will be available at the point where the algorithm has to be trained. In practice, this is rarely the case. Additionally, computing a large number of images would take too long with the Batch PCA method. For this reason, Incremental PCA, which allows the eigenspace to be added to over time is really convenient.

The objective of Incremental PCA is to form an eigenspace Ξ_3, defined by $\{N_3, \mu_3, P_3, \Lambda_3\}$, from two smaller eigenspaces, $\Xi_1 = \{N_1, \mu_1, P_1, \Lambda_1\}$ and $\Xi_2 = \{N_2, \mu_2, P_2, \Lambda_2\}$.

N_i represents the number of vectors in Ξ_i; μ_i is the average vector (average face) on that eigenspace; P_i is a Matrix with the largest d eigenvectors (eigenfaces) forming Ξ_i and Λ_i is a diagonal matrix containing the largest d eigenvalues.

$N_3 \in \mathbb{R}$ is simply the sum of N_1 and N_2, given that Ξ_3 will contain all the vectors in Ξ_1 and Ξ_2.

μ_3 is given by the weighted average of the averages of Ξ_1 and Ξ_2:

$$\mu_3 = \frac{N_1\mu_1 + N_2\mu_2}{N_1 + N_2} \in \mathbb{R}^D \tag{10}$$

P_3 is defined as:

$$P_3 = \Phi R \in \mathbb{R}^{(d_1+d_2+1)\times D} \tag{11}$$

where Φ is the orthonormal matrix spanning the combined covariance matrix S_3 and R is a rotation matrix. Φ can be calculated by concatenating P_1, P_2 (found using the Low-dimensional Batch PCA from 1.3) and $(\mu_1 - \mu_2)$:

$$\hat{\Phi} = \big[\overbrace{u_1 \quad \cdots \quad u_d}^{P_1 \in \mathbb{R}^{D\times d_1}} \ \overbrace{v_1 \quad \cdots \quad v_d}^{P_2 \in \mathbb{R}^{D\times d_2}} \ \overbrace{(\mu_1 - \mu_2)}^{\in \mathbb{R}^D}\big] \tag{12}$$

where u_i, represents the ith biggest eigenvector of P_1 and v_i represents the ith biggest eigenvector of P_2. In practice, d_1 and d_2 were chosen to be 103 (equal to the number of images on each of the subsets minus 1) due to the fact that when the eigenvalues of $S_{A,B,C,D}$ (as defined in 1.2) were calculated, the 103 largest eigenvalues, on all cases, were larger than 10^3 and the rest were found to be 0.

Subsequently, one must apply Gram-Schmidt Orthogonalisation to the columns of $\hat{\Phi}$ in order to achieve aforementioned orthonormal matrix Φ [3].

R can then be founds as follows:

From the definition it is true that:

$$S_3 = P_3\Lambda_3P_3^T \tag{13}$$

S_3, the covariance matrix of Ξ_3, can be found from S_1 and S_2 through:

$$S_3 = \frac{N_1}{N_3}S_1 + \frac{N_2}{N_3}S_2 + \frac{N_1N_2}{N_3^2}(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \tag{14}$$

where $S_3 \in \mathbb{R}^{D\times D}$ and hence from Equations (11) and (13) it follows that:

$$\Phi^T S_3 \Phi = R\Lambda_3R^T \tag{15}$$

$\Phi^T S_3 \Phi$ can be calculated straightforwardly and is a square matrix with each side of length $d_1 + d_2 + 1$. Using eigen-decomposition, R and Λ_3 can be found as the eigenvectors and eigenvalues of $\Phi^T S_3 \Phi$ respectively. It is important to note that this eigen-decomposition has a run-time complexity of $O((d_1 + d_2 + 1)^3)$, which combined with the run-time complexity of eigen-decomposition of P_1 and P_2 using the low dimensional method, $O(N_{1,2}^3)$, is still significantly reduced to that of the Batch PCA method of $O(D^3)$.

Finally, P_3 can be found using Equation (11).

The process explained in this section, can be applied to join Ξ_A and Ξ_B to result in Ξ_{AB}. Ξ_C and Ξ_D can then be joined iterating the method resulting in the Ξ_{ABC} and Ξ_{ABCD} subspace respectively.

The run-time per step is 1.2 seconds on average. Therefore, it took an overall of 4.8 seconds to train the whole data set. This results in a significant improvement from the 17.7s of Batch PCA. Additionally, Incremental PCA provides the added flexibility of not needing all the images from the get-go.

2.3 Image Reconstruction

The image reconstruction performed in this section follows the same principles as that of 1.4. The eigenvalues used for this section vary as the different incremental sub-spaces Ξ_{AB}, Ξ_{ABC} and Ξ_{ABCD} are calculated.

The following table summarises the number of eigenvectors taken from each subset, the time it took to compute them and the reconstruction error per subset.

Subset	EigVectors	Train-time(s)	Error
Ξ_A	103	0.07	28
Ξ_{AB}	207	1.47	24
Ξ_{ABC}	311	2.48	21
Ξ_{ABCD}	415	3.61	17

Table 4: Reconstruction and train time at different incremental steps

Calculating the eigenvalues of each of $S_{A,B,C,D}$ took 0.7s (using the low dimensional PCA method). These were all very similar in value given that their dimensions were all the same. The break down of training time for each individual subset, and the merging of these is found on the table below.

Subset	Train-time(s)	Merging-time(s)
Ξ_A	0.07	0
Ξ_{AB}	0.13	1.34
Ξ_{ABC}	0.20	2.28
Ξ_{ABCD}	0.27	3.34

Table 5: Reconstruction and train time at different incremental steps

2.4 Nearest Neighbour Classification

Nearest Neighbour Classification was performed at each incremental step to see the change in recognition accuracy with the testing set. Once again, Equation (9) is used to determine the labels of the images in the training set.

The confusion matrices for different incremental steps are shown on Figure 3. For this calculation, all the eigenvectors available at each step were used, which correspond to those on Table 4.

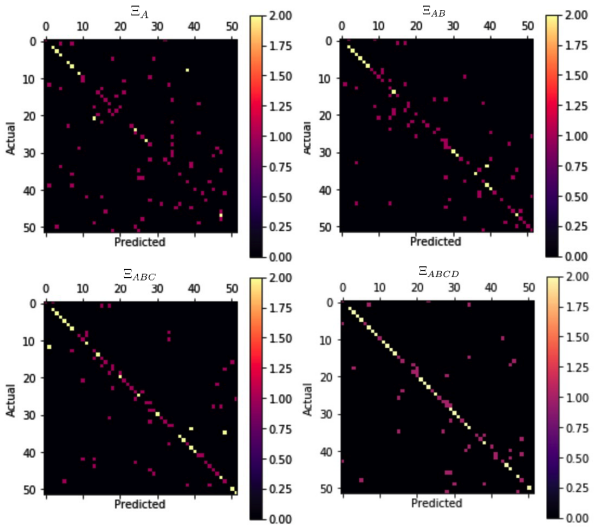

Figure 3: Confusion Matrices for the 4 different incremental subsets: $\Xi_A, \Xi_{AB}, \Xi_{ABC}, \Xi_{ABCD}$

As can be seen above, the accuracy improves significantly at each additional step. For sample success and failure at the last step please refer to Appendix B.2.

The accuracy at different incremental steps was as follows:

Subsets	Accuracy %
Ξ_A	32.69
Ξ_{AB}	43.26
Ξ_{ABC}	57.69
Ξ_{ABCD}	61.53

Table 6: Varying accuracy percentage with different incremental steps

As stated at the start of 2.2, the maximum performance obtained by the Incremental PCA method (61.53) was less than that of the Low-dimensional Batch PCA from 1.3 (68.27). In the former, a total of 415 eigenfaces were used and in the latter a total of 416 eigenfaces were used. The eigenfaces corresponding for the 16 highest eigenvalues of each can be found in Appendices B.1 and A.4 respectively.

For further testing, performance with different data partitions could have been performed, as well as using a data validation set[4]. A graph illustrating how the different subset accuracy varied with M can be found in Appendix B.2.

Finally, equivalently to Question 1, higher values of M led to better performance.

3 Question 3 - PCA-LDA Ensemble for Face Recognition

3.1 Data Partition

The data partition used for Question 3 is synonymous to that used in Question 1.

3.2 LDA

Linear Discriminant Analysis (LDA) is a method whose objective is to find the set of directions $W = [w_1, w_2, ..., w_M] \in \mathbb{R}^{D \times M}$ (known as Fisherfaces) that optimally separate data of different classes. This is fundamentally different from PCA, explored in Questions 1 and 2, whose objective is to find the direction of maximum data variance regardless of image labels. M is a parameter set by the user which will be explained later.

To perform LDA, one must first define the Within-Class scatter matrix and the Between-class scatter matrix.

The Within-Class scatter matrix S_W is a measure of how scattered the points of a class are from the mean face of that class for the whole data set. It is calculated by adding the scatter matrices S_i for each class $c_i : i = \{1, 2, ..., C\}$. S_i is given by:

$$S_i = \sum_{x \in c_i} (x - m_i)(x - m_i)^T \in \mathbb{R}^{D \times D} \tag{16}$$

where m_i is the mean face for class i, calculated as:

$$m_i = \frac{1}{N_i} \sum_{x \in c_i} x \in \mathbb{R}^D \tag{17}$$

and where N_i is the number of images belonging to class c_i.

S_W is therefore:

$$S_W = \sum_{i=1}^{C} S_i \in \mathbb{R}^{D \times D} \tag{18}$$

Following the definition above, S_W is symmetric and positive semi-definite; but, will be non-singular as long as $N > D$. The rank of this matrix is given by $N - C$ where N so was 364 in our training set.

The Between-Class scatter matrix S_B is a measure of how far the means of different classes m_i are from each other and is given by:

$$S_B = \sum_{i=1}^{C} N_i (m_i - m)(m_i - m)^T \in \mathbb{R}^{D \times D} \tag{19}$$

where m is the average face as defined in Equation (1). The rank of this matrix is, when calculated was 51 given that there were 52 classes and the rank of S_B of the matrix is given by $C - 1$.

As mentioned previously, the objective of LDA is to find the directions which optimally separate the data of different classes. This is equal to finding the direction along which the ratio of Between-class scattering to Within-class scattering is maximised so that, on that direction, the images of a class are as close to the mean image of that class as possible and the class means are as far away from each other as possible. Mathematically:

$$W_* = \underset{W}{\operatorname{argmax}} \frac{|W^T S_B W|}{|W^T S_W W|} \tag{20}$$

where W_* is the optimum set of Fisherfaces which can be found as an optimisation problem in which one tries to $\max_W W^T S_B W$ subject to $k = W^T S_W W$, where k is a constant. The problem can be written as the following Lagrangian function:

$$\mathcal{L}(W) = W^T S_B W - \Lambda W^T S_W W \tag{21}$$

where Λ is a square matrix in $\mathbb{R}^{M \times M} : 0 < M \leq D$ with Lagrangian parameters in its diagonal. The derivative can then be taken and equated to 0 to find the maximum:

$$S_W^{-1} S_B W = \Lambda W \tag{22}$$

where the optimal Fisherfaces are the columns of W and the values in the diagonal of Λ correspond to the eigenvectors and eigenvalues of $S_W^{-1} S_B$ respectively. The value of $M \in \mathbb{R}$ represents the eigenvectors corresponding to the M largest eigenvalues of $S_W^{-1} S_B$ which will affect recognition accuracy (similar to 1.5 and 2.4) and whose impact will be tested later.

As mentioned previously, S_W only has an inverse when $N > D$ which is not the case in our data set. For this reason, PCA has to be applied before to reduce the dimensionality and therefore make the finding of the inverse possible.

3.3 PCA-LDA

Because finding the inverse of S_W is not possible in our data set, the PCA-LDA method has to be used. This does not result in a loss of discriminatory information given that performing PCA first only eliminates the joint null space of S_W and S_B[2]. PCA-LDA involves reducing the dimensions of S_W and S_B through PCA and then apply LDA to find the Fisherfaces, W_{Opt}, defined as:

$$W_{Opt} = W_{PCA} W_{LDA} \in \mathbb{R}^{D \times M_{LDA}} \tag{23}$$

Where W_{PCA} is:

$$W_{PCA} = \underset{W}{\operatorname{argmax}} |W S_T W| \in \mathbb{R}^{D \times M_{PCA}} \tag{24}$$

which can be solved by the method described in 1.2 and using $S_T = S_W + S_B$ as the S from Equation (4). M_{PCA} represents the eigenvectors corresponding to the M_{PCA} largest eigenvalues. For samples please refer to Appendix C.2.

W_{LDA} is then defined as[2]:

$$W_{LDA} = \underset{W}{\operatorname{argmax}} \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|} \tag{25}$$

where one can use the regular LDA method described in 3.2 to find $W_{LDA} \in \mathbb{R}^{M_{PCA} \times M_{LDA}}$, which then allows one to get W_{Opt} (for samples please refer to Appendix C.2), as per Equation (23). Therefore, the PCA-LDA method, allows one to reduce the dimensions of the data from $D \times D$ to $D \times M_{LDA}$. In practice, we found that the maximum reasonable number of eigenvalues was $i = 51$ due to all other corresponded to 0% of the Trace of Λ (Appendix C.1). The optimal values of M_{PCA} and M_{LDA} corresponded to the values closest to their ranks as mentioned in 3.2. This will be discussed further in 3.4.

Unlike PCA, higher the values of M did not necessarily lead to higher performance because in PCA-LDA not all information is discriminatory.

3.4 PCA-LDA Nearest Neighbour Classification

In order to carry out Nearest Neighbour Classification, on must project an image $x_n \in \mathbb{R}^D$ into the Fisherfaces' eigenspace and then check what the nearest projection from the training set (refer to graph in Appendix C.3) is as in Equation (9).

Experiments were carried out varying the values of M_{PCA} in the range $[51, 350]$ and M_{LDA} in the range $[1, 51]$ and the best accuracy achieved was 90.02 which was achieved with $M_{LDA} = 51$ and $M_{PCA} = 350$, the values closest to their respective ranks. All the results achieved from testing along with the best-performing Confusion Matrix can be found in Appendix C.3. These values resulted in a good compromise between maximising the Between-class scatters and the minimisation of the Within-class scatters in LDA whilst still maximising the variance between data in PCA and resulted in a massive accuracy improvement from the PCA methods.

3.5 PCA-LDA Ensemble

PCA-LDA Ensemble is a way to regularise the model and hence improve generalisation. There are two ways in which this can be done. Bagging (randomly select data to train the model) and Feature Randomisation (randomly selecting the parameters for the model). Additionally, more than one model will be trained with these random features and classification will be decided by 'majority voting' within the models which form a Committee Machine.

The one that is going to be explored in this report will be Feature Randomisation. T PCA-LDA models were trained with M_{PCA} and M_{LDA} selected at random. Initially, acceptable range for M_{PCA} was $[51, 350]$ and for M_{LDA} was $[1, 51]$. These were decided from the testing carried out in part 3.4 and had a total of 15000 possible combinations. These ranges along with T were varied during testing.

Naturally, run-time increased with the number of models being calculated as can be seen below. The performance could have been improved through parallel calculation (i.e. multithreading), but the environment in which it was tested did not allow for this.

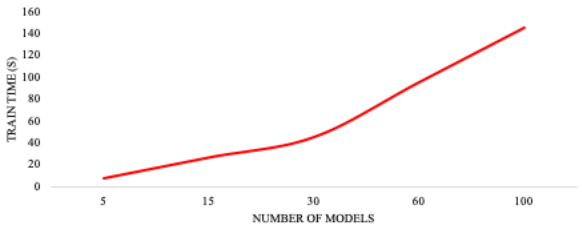

Figure 4: Run-time(s) against Number of models

The average accuracy of individual models Acc_{av} was compared with the accuracy of the Committee Machine Acc_{com} to check performance. Because of the increased generalisation, $Acc_{av} \leq Acc_{com}$ was the behaviour expected. The results are recorded in Table 8 in Appendix C.4 for varying T and ranges.

The Committee Machine outperformed the individual models once the number of models got big enough so that the majority voting was effective. The best accuracy achieved was 91.42% with 100 models. It is important to note that generalisation improved with additional models but improvement flattened out considerably after a 50 models.

The ranges which achieved maximum performance: $[300, 350]$ and $[20, 51]$ for M_{PCA} and M_{LDA} respectively, ended up being a good compromise between achieving high performance and introducing randomness into the system.

In terms of further improvement, the use of randomly selecting data for training (bagging) could help further enhance accuracy. Additionally, the use of another method before LDA could be tested such as latent semantic indexing[2] or the mixture of more models in the ensemble such as a combination of PCA, and PCA-LDA models.

References

[1] K. Fukunaga. *Introduction to Statistical Pattern Recognition (2Nd Ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[2] P. Howland and H. Park. *Equivalence of Several Two-stage Methods for Linear Discriminant Analysis*, pages 69–77. Society for Industrial and Applied Mathematics Publications, 2004.

[3] E. of Mathematics. Orthogonalization. `http://www.encyclopediaofmath.org/index.php?title=Orthogonalization&oldid=17050`. Accessed: 2019-11-10.

[4] H.-T. L. Yaser S. Abu-Mostafa, Malik Magdon-Ismail. *Learning From Data*. AMLBook, 2012.

A Question 1

A.1 Sample Training Data

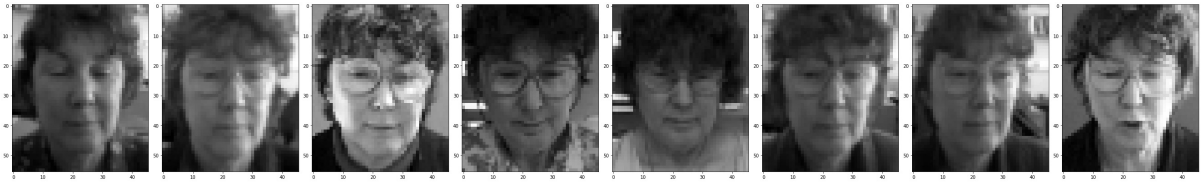

Figure 5: Sample training faces corresponding to label 1

A.2 Eigenvalues

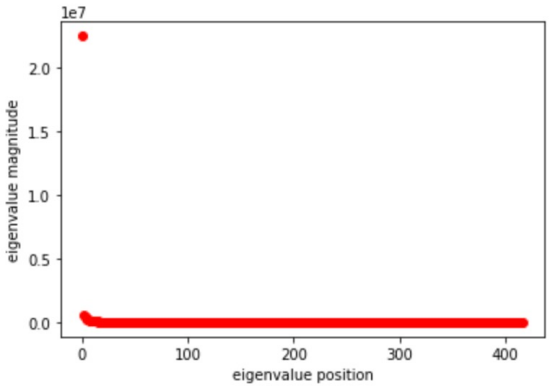

Figure 6: Graph of eigenvalue magnitude $|\lambda_i|$ against i

A.3 Reconstruction Images

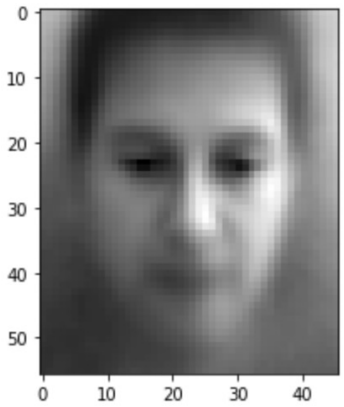

Figure 7: The average face $\bar{x}$

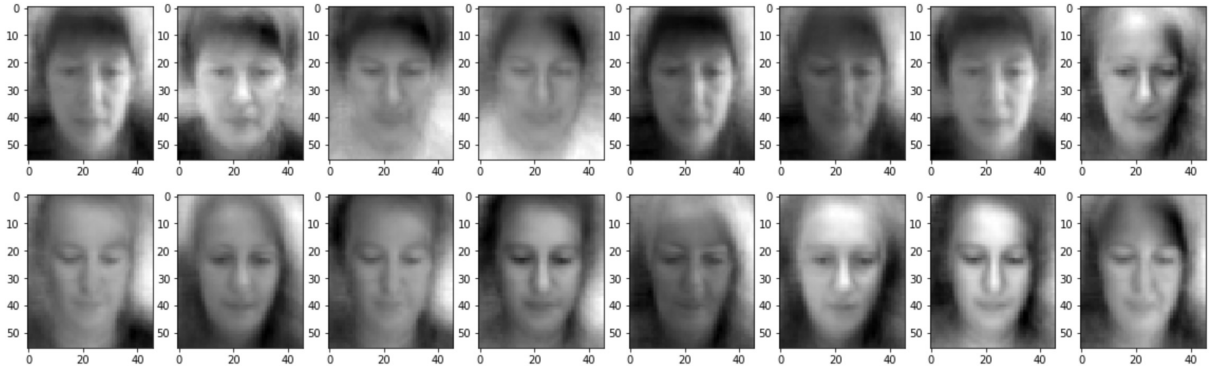

Figure 8: Image Reconstruction with $M = 5$

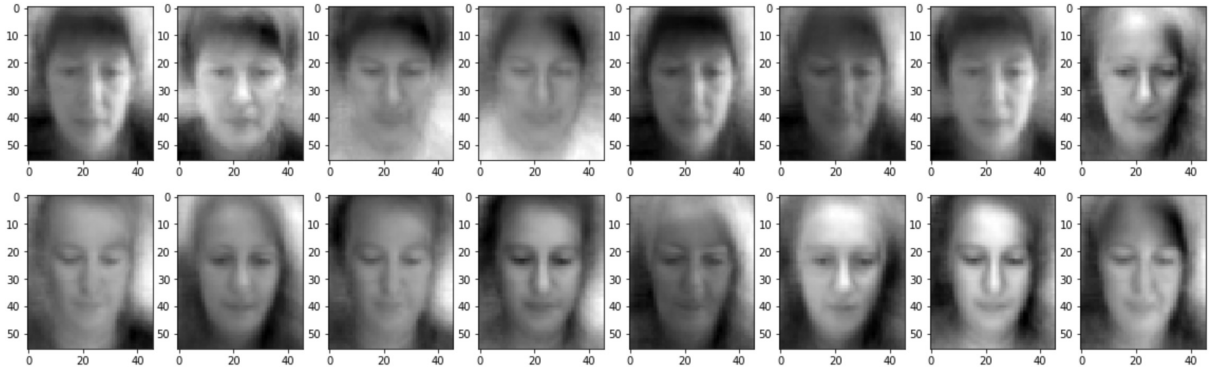

Figure 9: Image Reconstruction with $M = 10$

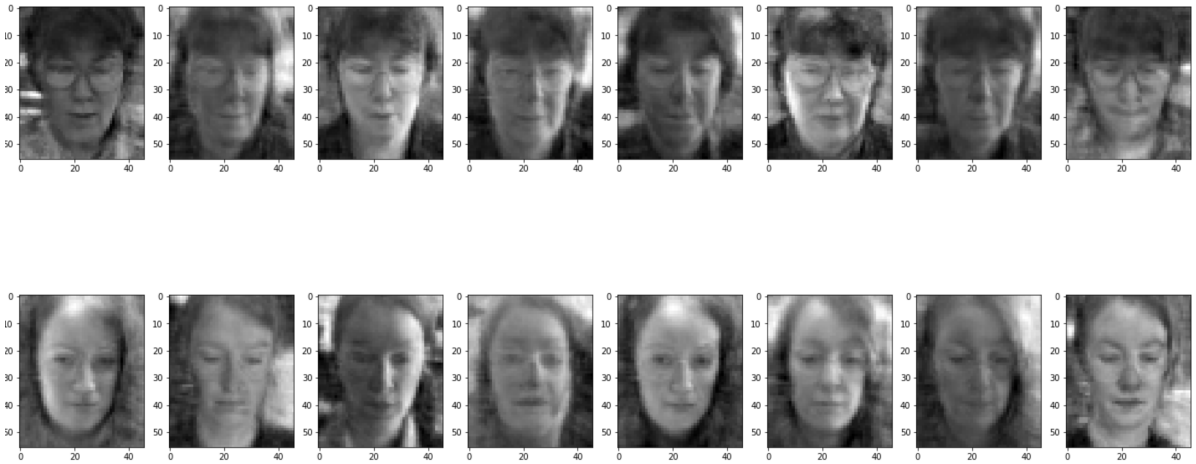

Figure 10: Image Reconstruction with $M = 100$

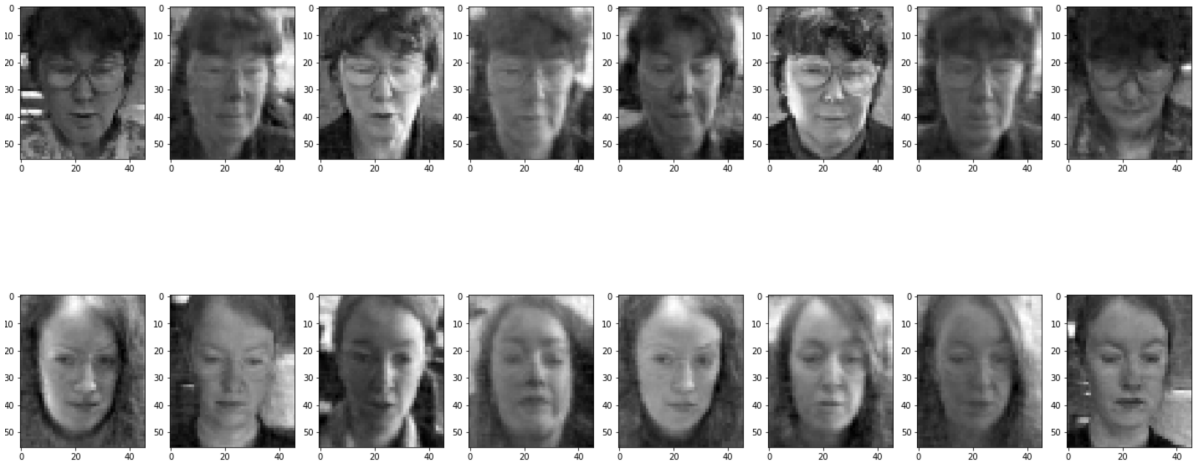

Figure 11: Image Reconstruction with $M = 200$

Figure 12: Image Reconstruction with $M = 416$

A.4 Eigenfaces

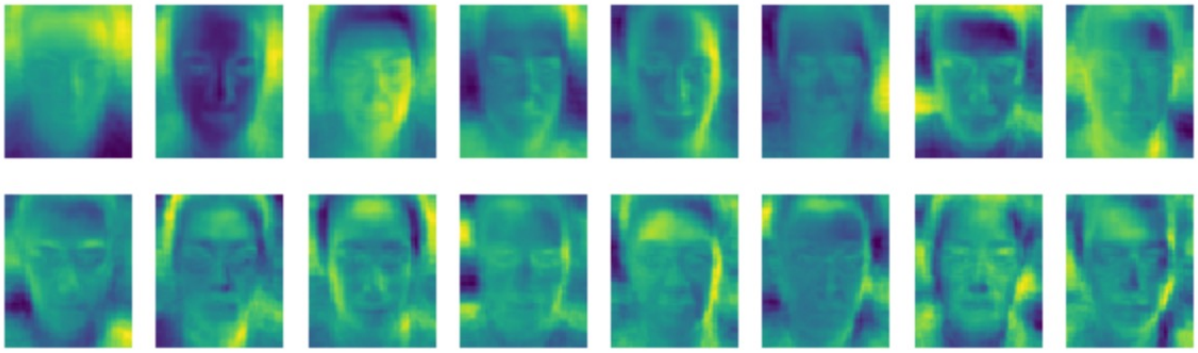

Figure 13: Eigenfaces corresponding to the 16 highest eigenvalues.

A.5 NN Classification

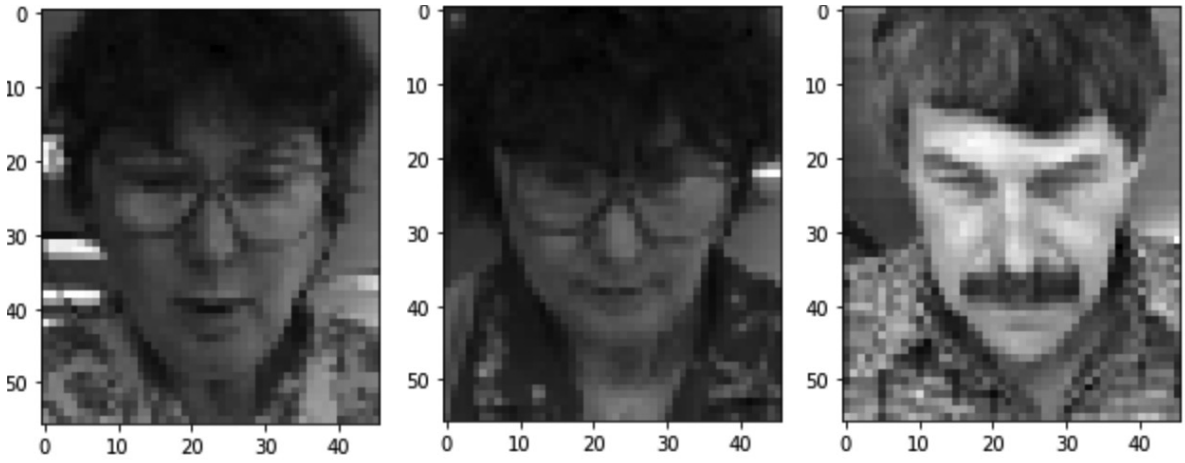

Figure 14: From left to right: Image label 1 classified correctly; image label 1 classified as label 42, training image with label 42

B Question 2

B.1 Eigenfaces

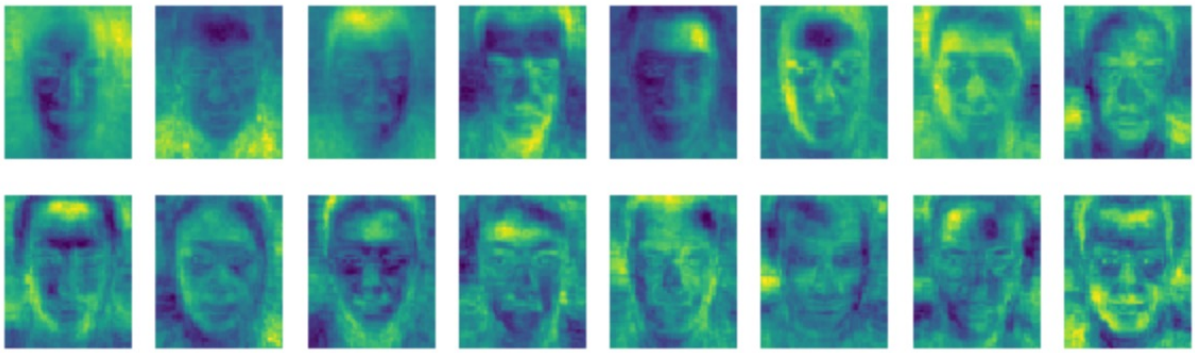

Figure 15: Eigenfaces corresponding to the 16 highest eigenvalues for Ξ_{ABCD}

B.2 NN Classification

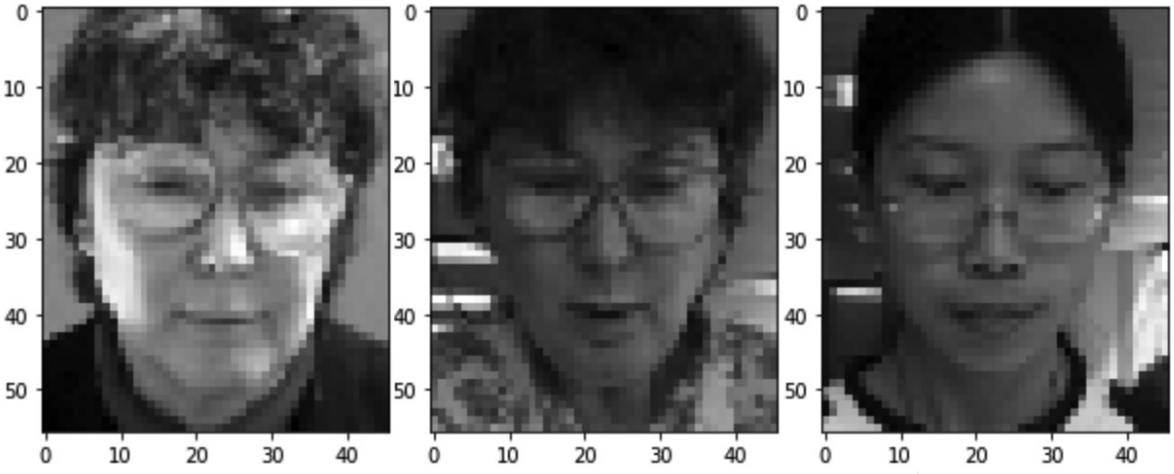

Figure 16: From left to right: Image label 1 classified correctly; image label 1 classified as label 24, training image with label 24

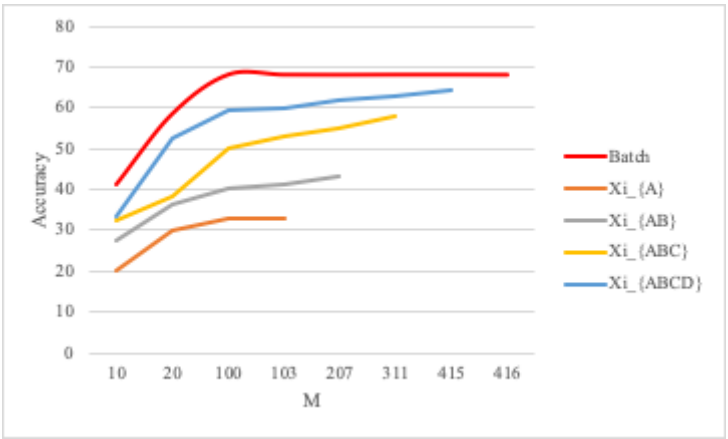

Figure 17: Accuracy of different subsets against M

C Question 3

C.1 Eigenvalues

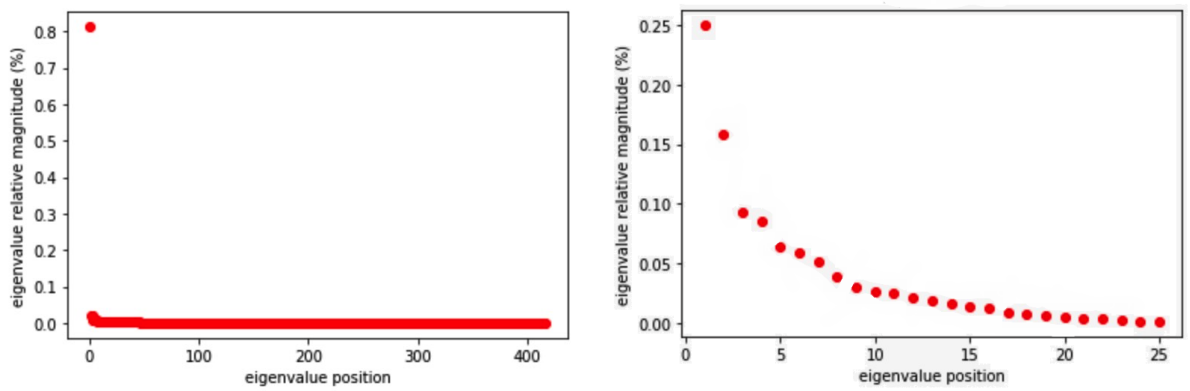

Figure 18: Graph of eigenvalue relative magnitude against i for $M_{PCA} = 416$ and $M_{PCA} = 25$

C.2 Fisherfaces

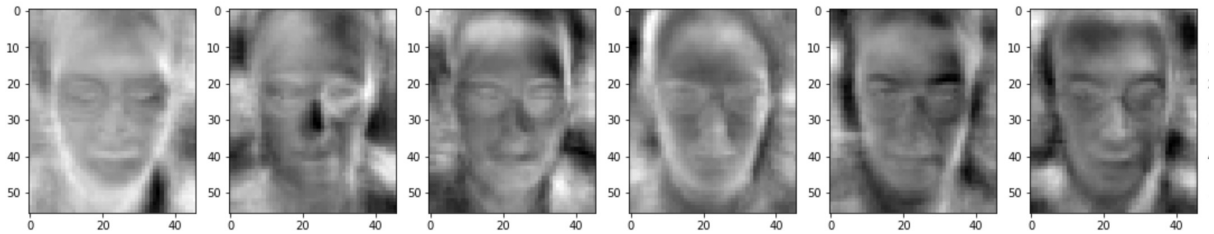

Figure 19: Samples of W_{PCA}

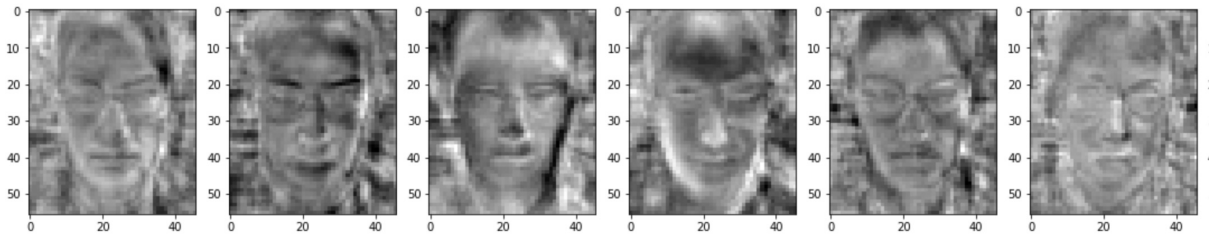

Figure 20: Sample Fisherfaces W_{Opt}

C.3 PCA-LDA NN

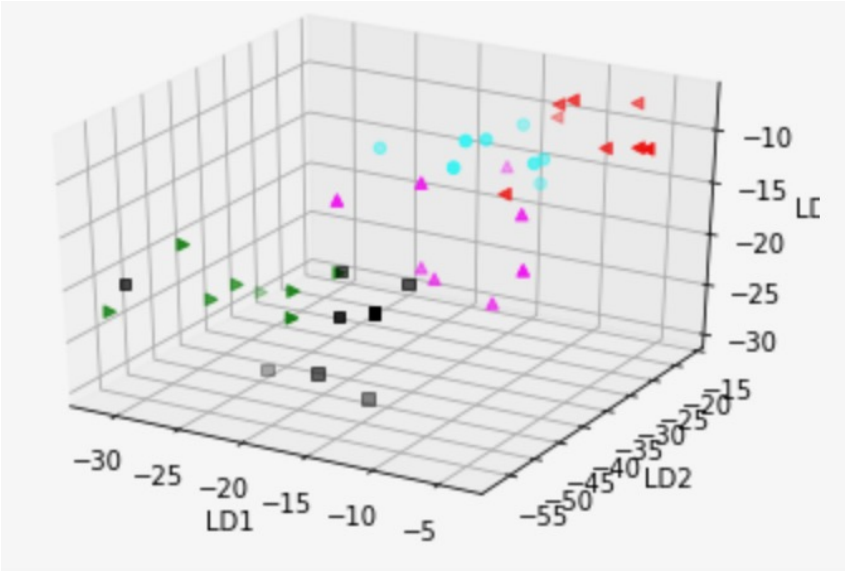

Figure 21: 5 training classes projected onto 3 Fisherfaces.

M_{LDA}	5	10	20	30	40	51
M_{PCA}						
350	50.21	65.32	82.01	84.02	89.08	90.02
300	48.12	58.65	58.75	74.69	86.29	88.30
200	42.01	52.88	52.89	62.88	78.95	80.21
100	38.68	40.22	48.20	50.96	65.94	72.40
75	30.22	37.22	45.51	50.00	59.20	63.15
51	25.12	36.35	38.22	42.10	47.84	50.96

Table 7: PCA-LDA accuracy with varying M_{PCA} and M_{LDA}

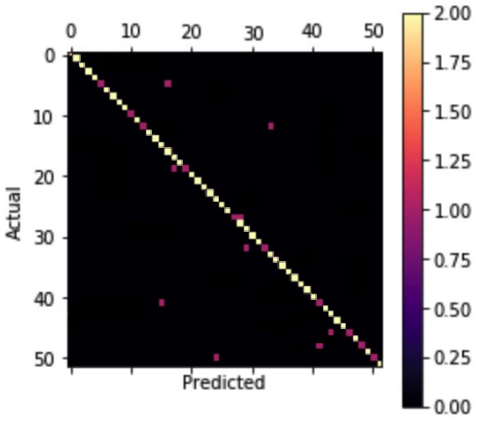

Figure 22: Confusion Matrix for $M_{PCA} = 350, M_{LDA} = 51$

C.4 PCA-LDA Ensemble NN

Models	5		10		50		100	
Ranges for M_{PCA} and M_{LDA}	Acc_{av}	Acc_{com}	Acc_{av}	Acc_{com}	Acc_{av}	Acc_{com}	Acc_{av}	Acc_{com}
$[51, 350], [1, 51]$	75.12	74.65	75.75	74.69	76.29	75.81	75.89	76.90
$[51, 350], [10, 51]$	76.01	75.67	75.92	76.88	74.95	76.02	75.91	77.05
$[150, 350], [10, 51]$	81.68	82.71	81.20	82.96	81.94	82.40	82.29	83.30
$[150, 350], [20, 51]$	85.91	86.02	86.17	87.00	85.21	86.15	86.01	87.01
$[300, 350], [20, 51]$	89.14	88.20	88.99	90.40	89.92	91.40	90.02	91.42

Table 8: Reconstruction and train time at different incremental steps