MLCV Coursework 1 Report

Shinyoung Yu
20200405
School of Computing
sahaa0918@kaist.ac.kr

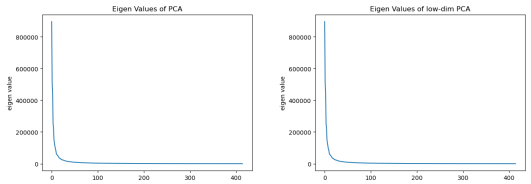
Jiyun Park
20210263
School of Computing
jiyun02@kaist.ac.kr

1. Eigenfaces

Face data is the base dataset for our experiment. This consists of 520 data with 52 classes and 10 data for each class. We will conduct the upcoming experiments by dividing the data for each class into an 8:2 train-test split.

1.1. Eigenfaces

There are two method to conduct PCA: PCA($S = AA^T$) and low dimensional PCA($S = A^TA$). We compare two methods on training dataset with size 2576*416(D*N).



(a) eigenvalues of PCA (b) eigenvalues of low-dimensional PCA

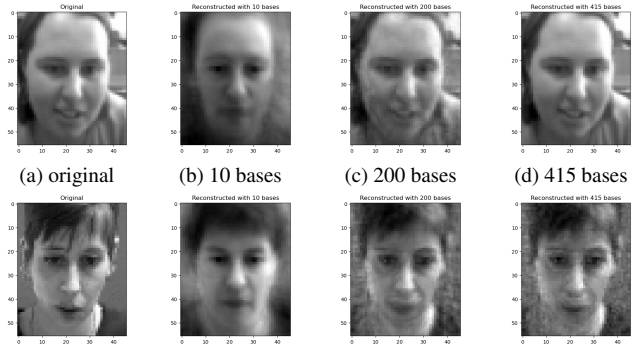
Figure 1. Comparison between eigenvalues of PCA methods

As shown in Fig. 1 both methods have the same nonzero eigenvalues. Specifically, PCA has 2576 nonzero eigenvalues, and low-dimensional PCA has 416. However, when selecting values bigger than $\frac{1}{100}$, both methods yield 415 eigenvalues. This is because, given matrix sizes of 2576x2576 and 416x416, the maximum ranks are 2576 and 416, respectively. But since we have 416 data points, the PCA projection subspace can at most capture 415 meaningful dimensions, resulting in 415 eigenvalues larger than $\frac{1}{100}$. From this comparison (Fig. 1), we conclude that both methods yield the same eigenvalues, and their top 415 eigenvectors are also identical.

The only difference between them is computation time: PCA takes 8.015s, while low-dimensional PCA takes 1.092s. Low-dimensional PCA is faster because, in our case, the data dimension exceeds the number of data points, making its covariance matrix smaller than that of PCA. Based on this time efficiency, we use low-dimensional PCA going forward.

1.2. Image reconstuction using Eigenfaces

Using pca basis from above, we perform face reconstruction. Fig. 2shows reconstruction results using different numbers of PCA bases. As seen in the images, the reconstruction become more accurate with more bases. When using only 10 bases, the results resemble the mean face(Appendix A) of training data. This occurs because the PCA basis is oriented to capture common features by maximizing data variance. Thus, with a small number of bases, the reconstruction reflects common features of the dataset more strongly than specific details of each image.



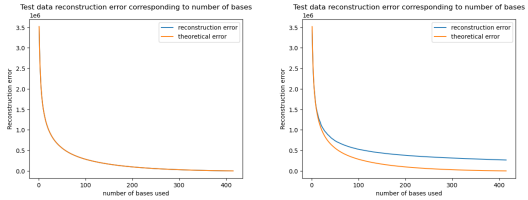
(a) original (b) 10 bases (c) 200 bases (d) 415 bases

(e) original (f) 10 bases (g) 200 bases (h) 415 bases

Figure 2. Data reconstruction results (1st row: training data, 2nd row: test data)

For more detailed analysis, we computed the reconstruction error. The theoretical error is given by $\sum_{i=n+1}^{M} \lambda_i$ where $n$is the number of bases we used for the reconstruction, $M$is total number of PCA bases, and λ_i is eigenvalues of unused eigenvectors. The actual reconstruction error can be calculated using L2-norm of difference between the original and reconstructed data.

As shown in Fig. 3, the reconstruction error from the training data matched the theoretical result. However, for the test data, while the graph's shape was similar to the theoretical result, the actual values differed slightly. This may be because the PCA bases used for reconstruction were derived from the training data, not the test data.



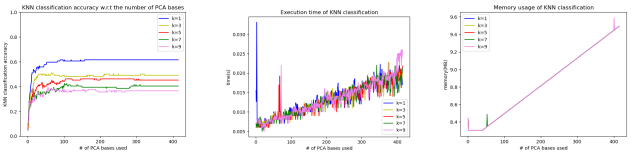
(a) training data (b) test data

Figure 3. PCA reconstruction error

1.3. KNN classification using Eigenfaces

We can also perform classification using PCA with NN classifier. As shown in Fig. 4a, classification accuracy converges after approximately 90 PCA bases. Therefore, it can be concluded that even when the original image dimension of 2576 is significantly reduced to 90, we can preserve sufficient feature information for classification. For the number of neighbors, since we are using only 8 training images per class, using fewer neighbors is better at preserving class boundaries, thus increasing classification accuracy.

In terms of time and memory, the number of neighbors made no significant difference. However, as shown in Fig. 4b and Fig. 4c, the execution time and memory usage of the k-NN classifier linearly increase with the number of PCA bases increase. This is because, the execution time and memory usage is $O(N \cdot M)$ where N is the number of total data and M is the number of PCA bases we used. Since k only affects on final selection process, its impact during the computation is very small.



(a) accuracy (b) running time (c) memory usage

Figure 4. Analysis on KNN classification using PCA

However, its overall prediction accuracy is about 0.6, which is not that high. This might because, PCA is not very sophisticated model to capture all the detailed features of data. As we also check from the confusion matrix(Fig. 13a), there are many components outside the diagonal, that failed to be predicted.

Detailed prediction examples can be found in Appendix B and Appendix C.

2. Incremental PCA

2.1. Important parameter in implementation: d_3

To implement incremental PCA, we utilized the algorithm from the "Online Learning" slides presented in class. Here, a key parameter is d_2 and d_3. When new data arrives in incremental PCA, computing the eigenspace model for this subset requires $O(\min(D, N')^3)$ time, where N' is the number of data points in the subset. Additionally, merging this new eigenspace model with the existing data takes $O((d_1 + d_2 + 1)^3)$ time, where d_1 is equal to the previously computed eigenspace model's d_3 value. Therefore, to enhance time efficiency in incremental PCA, it is essential to keep d_3 small, although this results in a time-accuracy tradeoff by dropping less-significant eigenvector information, which is represented by our experiment result Fig. 5

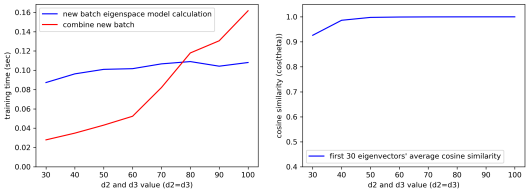


Figure 5. Incremental PCA's time-accuracy tradeoff according to the value of d_3

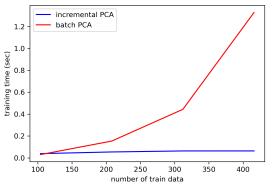
2.2. Comparison with other PCA

We compared the results of each incremental PCA stage (i.e.adding training data in four batches) with the results of batch PCA in the following four aspects. In summary, incremental PCA is a good approximation of batch PCA, and even requires less training time.

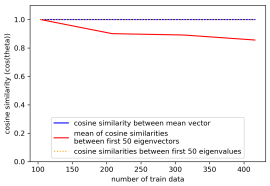
- Training time: For batch PCA, we measured training time by re-training the model each time new data was added. The results are shown in Fig. 6a. Using one subset, the training time is approximately the same for both methods. However, as we add more data, the value of N increases for batch PCA, while the values of N' and d_3 remain constant for incremental PCA, maintaining constant time and improving time-wise efficiency.
- Accuracy of incremental PCA: In incremental PCA, time-accuracy tradeoff occurs since less-significant eigenvectors are dropped during $d_1 + d_2$ merging, retaining only the top d_3 eigenvectors. We calculated the cosine similarity of eigenvectors, eigenvalues, and mean vectors between incremental PCA at each stage and batch PCA calculated by corresponding data, as shown in Fig. 6b. As more training data is added, the number of discarded less-significant eigenvectors increases, resulting in decreased similarity between eigenvectors; the similarity after adding the last subset is 0.856. For mean vectors and eigenvalues, cosine similarities are 1 and close to 1 respectively, indicating that the incremental PCA is a good approximation.
- Reconstruction error: Referring to Fig. 6c, the reconstruction error for incremental PCA using all training data (i.e., after adding the 4th batch) is almost identical to that of batch PCA with the same amount of data. This also

demonstrates that incremental PCA gives similar result to batch PCA.

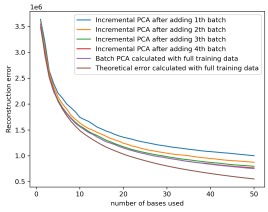
- Face recognition accuracy: In the optimal settings of batch PCA found in Question1 (i.e. K=1 and bases=90) the accuracy ranks as follows: full-data batch PCA > full-data incremental PCA > PCA using only the first training set. This indicates that classification accuracy improves progressively with the increase in training data. Additionally, using incremental PCA in place of batch PCA results in an accuracy drop of approximately 6.67%, also indicating a time-accuracy tradeoff.



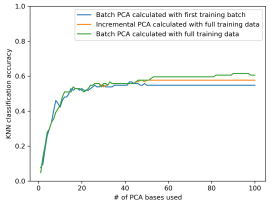
(a) Training time



(b) Cosine similarity of results



(c) Reconstruction error



(d) NN-classification face recognition accuracy ($n\_nearest = 1$)

Figure 6. Comparison between Incremental and Batch PCA methods

3. LDA Ensemble for Face Recognition

PCA can effectively reduce the dimension of input data preserving important features. And LDA can maximize the variance of between-class while minimize between-class. Thus, using PCA-LDA is expected to increase computing efficiency and classification accuracy. In this section, we try to figure out the effect of PCA-LDA via some experiments.

3.1. Recognition accuracy of PCA-LDA

To implement PCA-LDA, we set M_{pca} and M_{lda} by measuring classification accuracy with M_{pca} from 1 to 415 and M_{lda} from 1 to $min(M_{pca} - 1, 51)$. This is because, the maximum possible projection dimension for PCA is $N - 1$ since the total number of principal components can not be larger than overall data, and for LDA is $n_{class} - 1$ since the number of direction for maximizing the distance of between-class and minimizing within-class cannot be larger than the total number of classes. As shown in Fig. 7, accuracy peaked at $M_{pca} = 150$ and $M_{lda} = 50$. Higher M_{lda} improves performance by enhancing data discrimination, while optimal M_{pca} is between 100 and 200, balancing information retention and overfitting risk.

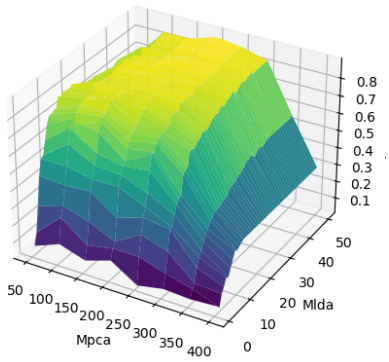


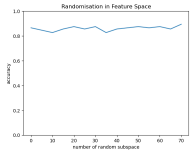
Figure 7. Classification accuracy varying Mpca and Mlda

In LDA, the rank of within-class scatter matrix(S_w) is $min(364, M_{pca})$, where $N - n_{class} = 416 - 52 = 364$ and the rank of between-class scatter matrix(S_b) is $n_{class} - 1 = 51$. Since $\sum_{x \in D_i}(x - m_i) = 0$, each class is linearly dependent, so, $S_w = \sum_{i=1}^{c} \sum_{x \in D_i}(x - m_i)(x - m_i)^T$ has at most $N - n_{class}$ linearly independent row vectors. After reducing dimensions using PCA, the rank of S_w cannot exceed M_{pca} as the vectors are confined to the PCA subspace. For S_b defined as $S_b = \sum_{i=1}^{c}(m_i - m)(m_i - m)^T$, it depends only ionly on class means, and since their relationship of class mean does not change after PCA projection since their relationships remain unchanged under PCA (a linear transformation), the maximum possible rank for S_b is $N - n_{class}$.

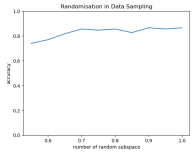
Based on this observation, we decided to fix $M_{pca} = 150$ and $M_{lda} = 50$ for future experiments. Detailed result including confusion matrix and images can be found in Appendix B and Appendix C.

3.2. PCA-LDA Ensemble

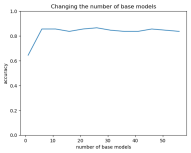
There are 3 hyperparameter that we can handle randomness: the number of random vector in feature space, the proportion of training data for subsampling, and the number of models for the ensemble. We will examine the impact of each one one by one.



(a) Randomization in feature space



(b) Random data subsampling



(c) randomization in the number of base-models

Figure 8. Accuracy comparison between randomness parameters

First, we look into randomization in feature space($M1$). As Fig. 8a indicates, the accuracy is almost highest when

the number of random samples is around 30. This is because if the number of random vectors is too large, important features cannot be preserved. However if it is too small, since each base model become more similar, the advantages of ensembling is diminished. Therefore, it is estimated that optimization occurred when the number of random vectors was 30.

Next, we observe the impact of data subsampling proportion(α). Fig. 8b shows that the more data we use, slightly better accuracy we get. Since the number of training data is small, the benefit of generalizing each model outweighs the benefit gained from randomness.

Lastly, regarding the effect of the number of base models(n_{model}), we varied the number from 1 to 60, as shown in Fig. 8c. When there are fewer than 30 models, the results are similar to a single PCA-LDA model. However, with more than 30 models, performance improves significantly, though further increases beyond 30 yield minimal gains. This is because up to 30 models, the ensemble can capture diverse features for generalization, but the PCA-LDA model's limited ability to capture complex patterns prevents substantial improvement beyond that point. We used "majority voting" for combining predictions, as it's most suitable for classification tasks where each class lacks numeric meaning, making it more appropriate than methods like averaging or maximum selection.

Considering both accuracy and computation cost, we conclude that the ensemble model with $M1 = 30$, $\alpha = 0.9$, $n_{model} = 30$ has the best performance.

3.3. Result of PCA-LDA Ensemble

The accuracy of committee machine, which gather all prediction results from each base models from ensemble model, is 0.846. And the average of individual models in ensemble model is 0.736. From this result, we can check that the performance of committee machine is better than individual as we learned. (Ensemble Learning LN, p.11-13)

In addition, by comparing the number of off-diagonal components of confusion matrices, (Fig. 13b and Fig. 13c) we can visually check that the performance of ensemble model is better.

4. Generative and Discriminative Subspace Learning

PCA is a generative model, which aims to reconstruct the input. And LDA is a discriminative model, which conducts classification. If we want to take advantages of both methods, it is inevitable to redisign the objective function.

4.1. Objective function

To achieve our goal, it is natural to think of a function that combine objective functions of PCA and LDA.

Eq. (1) is identical to PCA objective function when $\alpha = 1$ and to LDA objective function when $\alpha = 0$. Our goal is to find W that maximizes $J(W)$ since reconstruction error is reduced and subspaces become more discriminative when PCA, LDA objective functions are maximized. Such W must satisfy Eq. (2) for some constant λ.

$$J(W) = \alpha W^T S W + (1-\alpha)\frac{W^T S_b W}{W^T S_w W} \tag{1}$$

$$(\alpha S + (1-\alpha)S_b)W = \lambda(\alpha I + (1-\alpha)S_w)W \tag{2}$$

If $\alpha I + (1-\alpha)S_w$ is invertible, we can easily get W using eigenvector-eigenvalue method. Using this method, we can reduce computation time and memory space required to store the train parameters since we only use one weignt matrix W. However, it is more difficult to visualize the result of feature extraction of PCA compared to PCA-LDA's sequential data transformation. Detailed derivation and the pros/cons of this method can be found in Appendix D.

5. RF classifier

The random forest model has several key hyperparameters, and testing them all simultaneously would require excessive effort and time. Therefore, we fixed the optimal parameters in the sequence described below. Also, to ensure result stability, each test was executed 10 times. The average of 10 results displayed in a graph, and the confusion matrix represents the best result among those is recorded in Appendix.

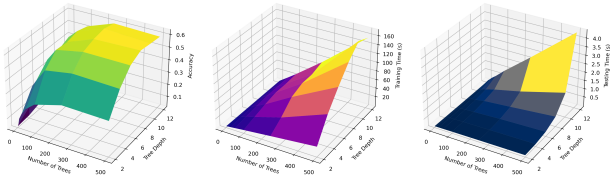
1. Number of Trees & Tree Depth: We tested using an axis-aligned weak learner with the split number fixed at 10.
2. Split Number (Randomness Parameter): Using the optimal number of trees and tree depth derived in step 1, with an axis-aligned weak learner.
3. Type of Weak Learner: With the optimal values for other parameters determined in steps 1 and 2, we tested different weak learners.

5.1. Number of trees & The depth of trees

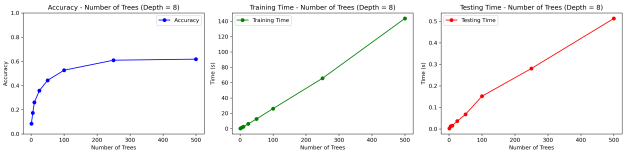
We varied the number of trees and their depths, obtaining the results shown in Fig. 9a. In the graph, the best accuracy 0.625 occurred at N (number of trees) $= 250$ and D (depth) $= 10$, explained as follows:

- Number of Trees: A single tree in random forests tends to overfit to data; therefore, we can generalize the model using ensembles of trees. As shown in Fig. 9a, the accuracy converges near $N = 250$, which is selected as optimal parameter.
- Tree Depth: When depth is D, maximum of $2^D - 1$ nodes are generated. As shown in Fig. 9a, for a given N, accuracy initially increases with tree depth but later decreases due to overfitting. Notably, the optimal depth D increases with N, indicating that a larger N result in less overfitting. Also, we chose $D = 8$ as the later experiment's standard value, accuracy of 0.610, which is high enough.

The theoretical training/testing time when adjusting the number of trees and tree depth is as follows, and our testing time results align with theoretical predictions Fig. 9b, Fig. 9c. However, due to the small size of the training data, some splits stop, not reaching the maximum tree depth (when data in the node less than 5), resulting in less training time than theory.



(a) Test accuracy, Training time, Testing time according to the number of tree and the depth of tree



(b) Test accuracy, Training time, Testing time according to the number of tree (D = 8)

(c) Test accuracy, Training time, Testing time according to the depth of tree (N = 250)

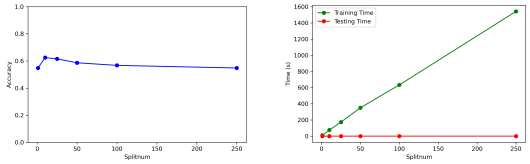
Figure 9. Test accuracy, training/testing time according to the number and depth of trees

- Number of Trees N: $O(N)$ time
- Tree Depth D: $O(2^D)$ time

Moreover, our code do not utilize parallelization of each tree's growth. Based on theoretical insights from course materials, training each tree in parallel could reduce the time for the tree numbers to $O(1)$, not $O(N)$.

5.2. Randomness parameter

We randomly select dimensions and threshold values to create the split function. For each split function, we attempt ρ random splits and choose the one with the highest information gain. There is a trade-off in the magnitude of ρ value: If the ρ value is too low, fewer splits are attempted, which reduces similarity between trees but increases the risk of the less-optimal split. Conversely, as ρ increases, more various split functions are tested, leading to greater similarity among trees, which decreases the advantage of ensemble multiple trees. As shown in Fig. 10a, accuracy initially increases with higher ρ values until $\rho = 10$ and then decreases, as expected. In Fig. 10b, training time is same as expected, $O(\rho)$. However, since ρ does not affect the resulting tree structure in training, it has constant testing time.



(a) Test accuracy according to the randomness parameter

(b) Training time, Testing time according to the randomness parameter

Figure 10. Test accuracy, training/testing time according to the randomness parameter

5.3. Impact of the different types of weak-learners

Fig. 11 shows the impact of weak learner. As expected, the axis-aligned method provides sufficiently good testing accuracy with shorter training and testing times. However, the two-pixel test achieves 8.32% higher accuracy under the same parameters, indicating greater information gain per split. Other methods, such as linear and non-linear weak learners, were also tested but omitted from the report due to impractically long training and testing times.

Figure 11. Axis-aligned vs Two-pixel test

The main results' confusion matrix, example success/failures, example node's histogram representing information gain along splitting are written in Appendix: E, F

5.4. Comparision between PCA and PCA-LDA

Since parallel programming was not used our random forest, both the training and testing times are significantly large compared to the PCA method. In addition, for PCA, Python NumPy's parallelization for matrix operations and scikit-learn's optimized KNN classification can handle multiple data points much more efficiently. About the accuracy, because raw pixel values were used directly as inputs, the optimal accuracy of Q1 PCA is almost identical to random forest's result. Overall, the accuracy is much lower compared to the PCA-LDA method, indicating that in situations where the size of the training dataset is small ($N << D$), applying the random forest with no PCA is relatively less suitable. Additionally, similar to the PCA-LDA ensemble, as the number of base models (tree) increases, there is a tendency for accuracy to improve, indicating that the 'ensemble models' is beneficial.

Appendix

A. Q1: Mean face

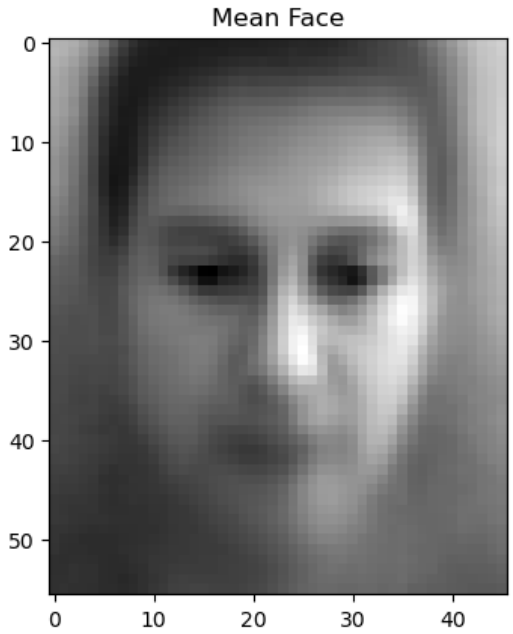
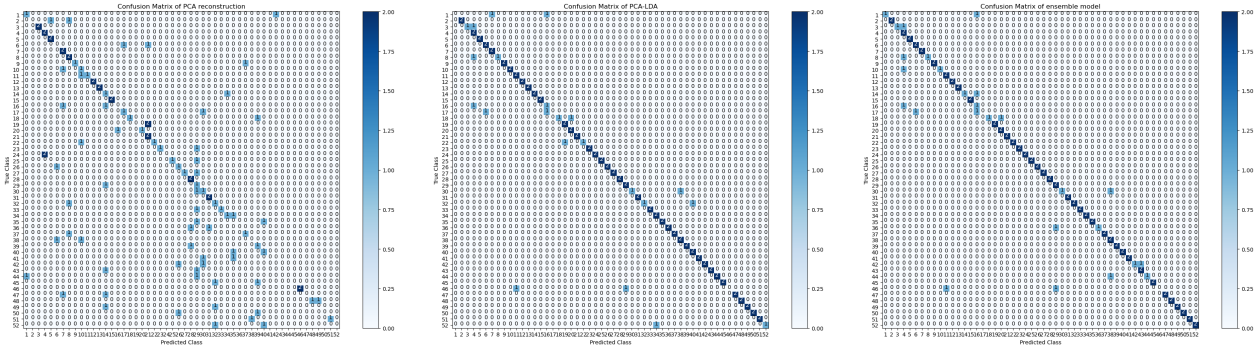


Figure 12. Mean face of training data

B. Q1,3: Confusion matrix



(a) Confusion matrix of PCA (b) Confusion matrix of PCA-LDA (c) Confusion matrix of Emsenble model

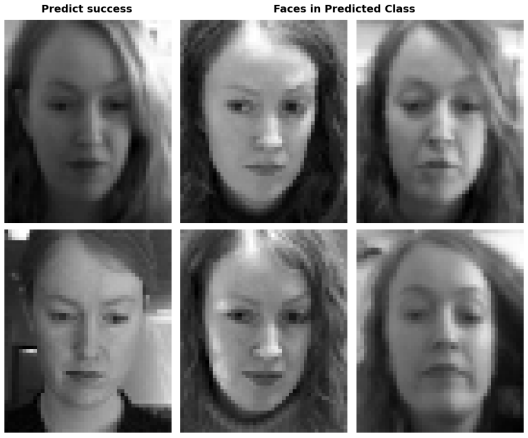
Figure 13. Confusion matrix of PCA, PCA-LDA, and PCA-LDA ensemble model

As shown in Fig. 13, confusion matrix of PCA has much more off-diagonal elements than PCA-LDA and Ensemble model. This shows that the prediction is much more successful using PCA-LDA and Ensemble model than PCA. Also, Ensemble model has slightly less off-diagonal elements than PCA-LDA. This indicates Ensemble model is more accurate in classification than the original model.

C. Q1,3: Prediction examples

(a) PCA classification success example

(b) PCA classification failure example



(c) PCA-LDA classification success example

(d) PCA-LDA classification failure example

Figure 14. The success/failuer examples of PCA and PCA-LDA with NN classification

Above images shows successful and failure examples of classification using PCA and PCA-LDA. If we take a closer look at successful cases, the model infer the class accurately despite different angles of face. This may because the model can generalize well across different angles of the face. For failure cases, the prediction appears to have failed because the glasses and facial expressions are similar.

D. Q4: Generative and Discriminative Subspace Learning

Below is the objective function for generative and discriminative subspace learning. Our goal is to find W that maximizes $J(W)$

$$J(W) = \alpha W^T S W + (1-\alpha)\frac{W^T S_b W}{W^T S_w W} \tag{3}$$

Eq. (3) can be changed as Eq. (4) since $W^T W = I$. (This is because, W is a projection matrix, hence $W^T W$ indicates re-projection after projection to the subspace. Therefore, projected vector returns to itself after applying $W^T W$.)

$$J(W) = \alpha\frac{W^T S W}{W^T W} + (1-\alpha)\frac{W^T S_b W}{W^T S_w W} = \frac{W^T(\alpha S + (1-\alpha)S_b)W}{W^T(\alpha I + (1-\alpha)S_w)W} \tag{4}$$

If we keep the denominator constant, the problem we have to solve changes to the problem to maximize the numerator. Let the denominator $k(constant)$ and use Lagrange-multiplier. (Eq. (5))

$$W^T(\alpha I + (1-\alpha)S_w)W = k \tag{5}$$

$$L(W,\lambda) = W^T(\alpha S + (1-\alpha)S_b)W + \lambda(k - W^T\alpha I + (1-\alpha)S_w)W) \tag{6}$$

To get the solution, each partial derivatives of $L(W,\lambda)$ with respect to W and λ must be 0.

$$\frac{\partial L}{\partial W} = 2 * W(\alpha S + (1-\alpha)S_b - \lambda(\alpha I + (1-\alpha)S_w)) = 0 \tag{7}$$

$$\frac{\partial L}{\partial \lambda} = k - W^T(\alpha I + (1-\alpha)S_w)W = 0 \tag{8}$$

Eq. (8) is satisfied from our assumption of the denominator. Now, the only thing we have to consider is Eq. (7). If we organize Eq. (7), we can get Eq. (9).

$$(\alpha S + (1-\alpha)S_b)W = \lambda(\alpha I + (1-\alpha)S_w)W \tag{9}$$

If $\alpha I + (1-\alpha)S_w$ is invertible, the equation becomes as below.(Eq. (10))

$$(\alpha I + (1-\alpha)S_w)^{-1}(\alpha S + (1-\alpha)S_b)W = \lambda W \tag{10}$$

Since λ is some constants, we can regard Eq. (10) as a eigenvector-eigenvalue problem where W is an eigenvector matrix and lambda is an eigenvalue matrix.

We can discuss the foreseeable behavior when applying the resulting W matrix from the above equation to data. Since this optimization involves both PCA and LDA, it can blend the advantages of both methods, generating high-accuracy classification results similar to Question 3's PCA-LDA method.

The pros and cons of this approach are as follows: In Question 3, we trained W_{pca} and W_{lda} separately for PCA and LDA and used $W_{opt}^T = W_{lda}^T W_{pca}^T$. This requires solving two eigenvector-eigenvalue problems to derive each of W_{pca} and W_{lda}. However, by using our new objective function, we only need to solve one eigenvector-eigenvalue problem and store just the W matrix, which reduces both time complexity and the space required to store the trained parameters.

There are also some drawbacks to this approach. Simultaneously tuning the hyperparameters, α and the final dimension D (where $W = (N \cdot D)$), may be challenging. In Question 3, since we trained W_{pca} and W_{lda} separately, we could verify the effectiveness of PCA and LDA process individually by applying it step-by-step to real images and printing intermediate outputs (e.g., displaying the PCA-reconstructed image). However, in the method proposed for Question 4, such verification is not possible. In other words, if the current test accuracy is not high enough, it would be difficult to determine whether the issue arose during the PCA or LDA phase.

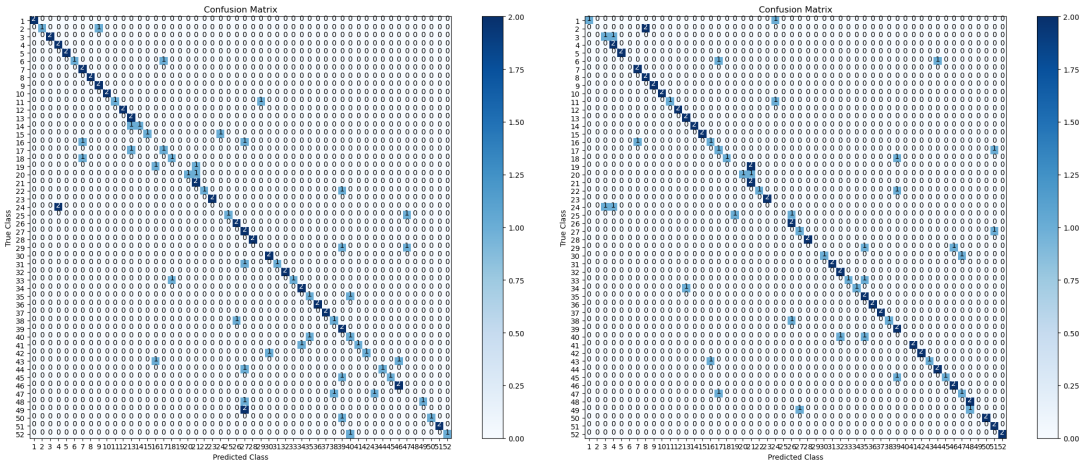
E. Q5: Test result's confusion matrix and success/failure cases

The optimal parameters determined for the random forest through experiments are $N = 250$, $D = 8$, and splitnum $= 10$. With this configuration, each real-time executable weak learner—axis-aligned and two-pixel tests—each was trained 10 times. The confusion matrix results in Fig. 15 represent the best test accuracy among the 10 repetitive train results.

Additionally, the example success and failure cases based on the confusion matrix results above are shown in Fig. 16, Fig. 17, Fig. 18, Fig. 19. Compared to the success cases, the failure cases show a higher similarity between the failed image and the predicted class, meaning that our model is reasonable.

F. Q5: Visualization of random forest's information gain process

Random forests utilize the entropy-based information gain method to reduce the diversity of classes within the nodes as they move down to the lower nodes. To illustrate the execution of this algorithm in our code, we have traced the path down the left child nodes of a single tree and represented the results in the histograms below. Each histogram corresponds to the left node of the previous figure, and the result Fig. 20 align with our expectations.



(a) Axis-aligned weak learner, test accuracy=0.644

(b) Two-pixel test weak learner, test accuracy=0.692

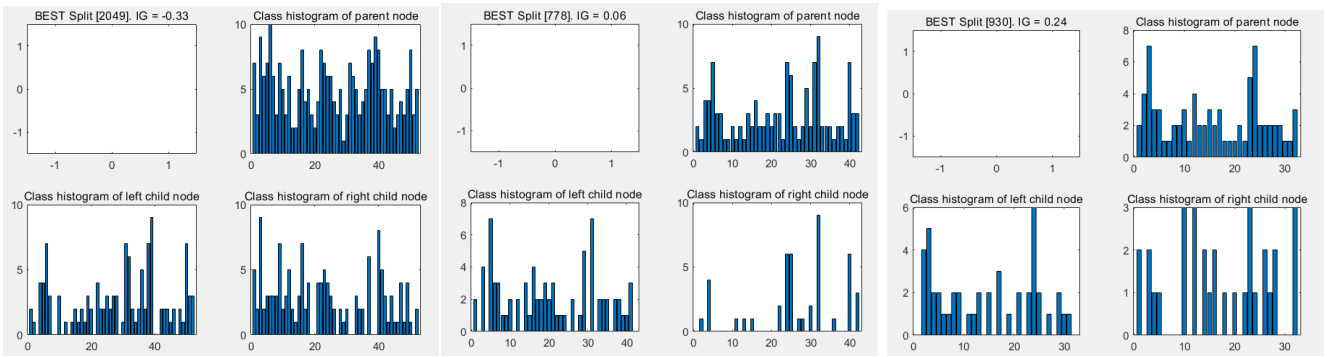
Figure 15. Confusion matrix of optimal cases ($N = 250$, $D = 8$, and splitnum $= 10$)

Figure 16. Axis-aligned weak learner: Example success case

Figure 17. Axis-aligned weak learner: Example failure case

Figure 18. Two-pixel test weak learner: Example success case

Figure 19. Two-pixel test weak learner: Example failure case



(a) Top node (depth=1) (b) depth=2 (c) depth=3

(d) depth=4 (e) depth=5 (f) Leaf (depth=6)

Figure 20. Visualization of random forest's information gain process