Coursework 1

20170781 Seoyun Yang, 20170790 Deokmin Hwang

I. Computationally Efficient EigenFaces

In this report, as following the overall steps of PCA, we analyze how it works and how much error has occurred. Also, by analyzing the relation between AAT and ATA, we compared which one is more efficient to get eigenvectors of covariance matrix.

A. Overall Flow of PCA and Face Reconstruction

In this step, we used 15 people's face images, each class has 10 images and each image is 56x46 pixels. And we used 8 images per person to train and rest images for the test. The mean face, reconstructed faces using various eigenvectors and corresponding errors are as follows. We can see in Fig 1.1 and 1.2, the faces are getting clear and error decreases as we use more eigenvectors.

Fig 1.1. The mean face and the constructed faces of class 1,2,3,4 using 1, 10, 30, 50, 70, 90, 150, 250, 400 eigenvectors.

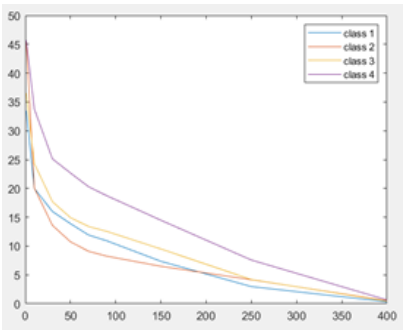


Fig 1.2. The errors of constructed faces

B. Comparison Between PCA and Low-Dimensional PCA

1) The Eigenvalues and Eigenvectors

Both PCA and Low-dimensional PCA have 416 nonzero eigenvalues and they are identical. The largest eigenvalue was 2.1751e^6 and the smallest non-zero eigenvalue was 83.1048.

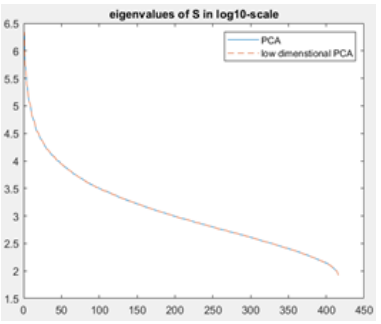


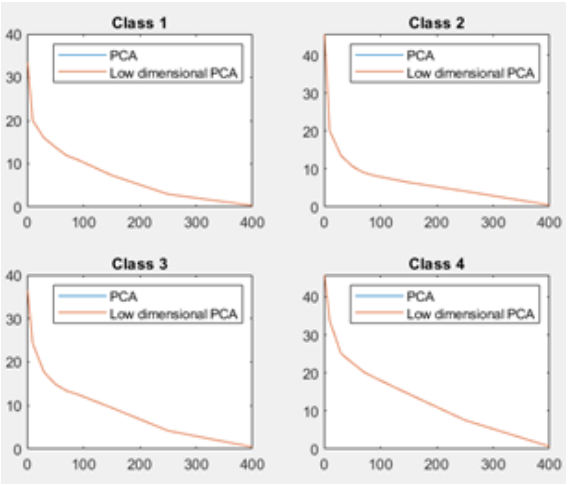
Fig 3. The eigenvalues of PCA and Low-dimensional PCA in log 10 scale

2) Consumed time

	PCA	Low dimensional PCA
Time for eigenvalue decomposition	2.680559 sec	0.035094 sec
Time for reconstruction	2.631873 sec	2.841501 sec
Total time	5.372432 sec	2.876595 sec

We can find low dimensional PCA shows 98.69% speed improvement in eigenvalue decomposition for our face dataset. And in the reconstruction process, low dimensional PCA takes 7% more time than PCA, but in total time it still shows greatly improved speed. It shows 46.46% improved speed.

3) Reconstructed faces and error



C. Discussion

After PCA, when the gray-scale image, 20608 bits of 56x46 (D=2576) pixels is expressed as 100 weights (projected to the eigenvectors, single type), 3200 bits. Therefore, it shows 84% of compression ratio. Furthermore,

the reconstructed data loss was only 12% on average of 10 classes. In addition, when performing low-dimensional PCA through , it is very efficient because the exact same eigenvector and reconstruction error can be obtained at 46% faster speed than the original PCA.

The eigenvectors of AA^T and the eigenvectors of A^TA are also can be obtained by the Singular Value Decomposition.

$$A = U\Sigma V^T$$

The k eigenvectors of AA^T corresponding to nonzero eigenvalues are same as the k column vectors of U, $\{u_1, u_2, \cdots u_k\}$. And the eigenvectors of A^TA are same as the column vectors of V, $\{v_1, v_2, \cdots v_k\}$. And they are the orthonormal basis for col(A) and the orthonormal basis for row(A). So, the following equation is obvious.

$$u_i = Av_i / \|Av_i\|$$

II. Incremental PCA

IPCA (Incremental PCA) is an alternative to PCA when additional data arrives and the model needs to be updated or when the data is too large to be trained at once. Instead of training the whole data set altogether, IPCA divides the dataset into batches and updates the model incrementally retraining only on the new batch each time.

The crucial part in the IPCA algorithm is to compute the eigenvalue of the combined data using the sufficient spanning set. This is what makes the IPCA capable of decreasing the computation cost from $O(min(D, N)^3)$ to $O((d_1 + d_2 + 1)^3)$, where D is the number of features, N is the number of data samples in the combined data, and d_1, d_2 are the number of eigenvectors stored in the eigenvector matrix of the two eigenspace models to be combined. For detailed explanation of the algorithm, refer to the lecture slide [1].

A. Experimental Setup

The same face data is used and divided into training and testing data as in *I. Computationally Efficient Eigenfaces*. For IPCA, the training data is additionally divided into 4 subsets, each with 104 images (i.e., 2 images per face identity).

3 models compared are IPCA, batch PCA, and 1-PCA (PCA trained only by the first subset of the training data). 1-PCA can be thought of as either a batch PCA trained on only one subset, or an IPCA before any updates.

Important parameter to consider is m, the number of principal components chosen. That is, for batch PCA and 1-PCA, the number of best eigenvectors to keep from the eigenvector matrix of the whole training data and for IPCA, from the newly arriving training batch.

B. Experimental Goal

We seek training time, face recognition accuracy and reconstruction error. Training time only considers the time consumed for eigen-decomposition. The face recognition accuracy is estimated using the NN classifier. Lastly, reconstruction error is the root mean square error between the original data and the reconstructed data.

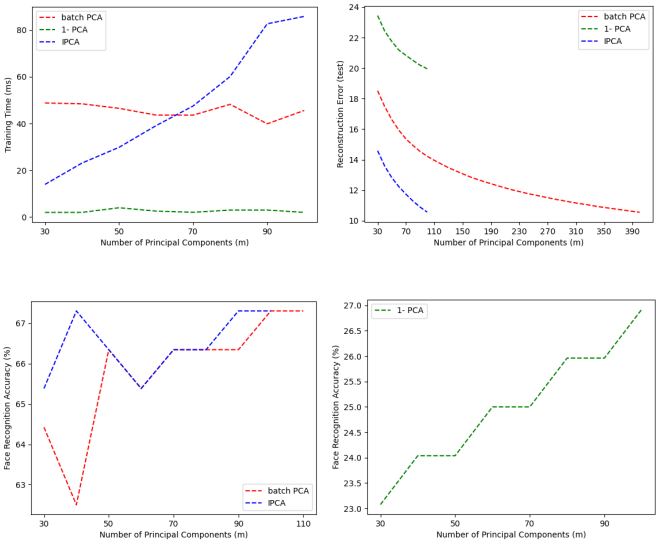


Fig. 2 Evaluation results of batch PCA, 1-PCA and IPCA. Respectively, training time, reconstruction error, and face recognition accuracy according to m. 1-PCA and IPCA have an upper bound 104 for m. Face recognition accuracy plots are divided into two because the range of 1-PCA accuracy is very low compared to the other two. The range of m, which is the range of the x-axis, is controlled by whether the results for batch PCA converge or not.

C. Result and Discussion

Comparing the results in Fig. 2, we can understand the relationship between a model and its parameters and the relationship between the models.

First, consider the relationship between each model and its parameter. Increase in parameter m increases the training time for IPCA, but remains similar for batch PCA and 1-PCA. For IPCA, the computation cost according to m is $O((2m + 1)^3)$, by substituting m for d_1 and d_2 in the formula mentioned above. We can see directly that the training time increases with increasing m. However, for batch PCA and 1-PCA, the eigen-decomposition is held on the same covariance matrix regardless of m. So, the training time remains the same. The slight differences in the training time are due to the inaccuracy in time measurement by *Python*.

Also, increase in parameter m increases and then converges the accuracy to a certain point for all three models. This is because after the best eigenvectors are preserved to a certain extent, additional eigenvectors with small importance don't affect the classification result of the model. Also, the same tendency appears in all three models for reconstruction error. Bigger parameter value means that more eigenvectors are kept, and more eigenvectors allow detailed reconstruction.

Second, consider the relationship between the models. The training time and accuracy is the lowest for 1-PCA because it trains only on the first subset of the training data, which is very small compared to the other two models.

Comparing the results of IPCA and batch PCA, we can clearly observe that the training time of IPCA is smaller than batch PCA, which was the reason for preferring IPCA over batch PCA when dealing with large data. The training time of IPCA increases as d gets larger, but small d is enough to get the max accuracy. One weakness of IPCA is that there is information loss as updates are made in IPCA. However,

batch PCA and IPCA achieve the same upper bound in accuracy and similar lower bound in reconstruction error. This is because only 3 updates were made and there wasn't much information loss. We can predict that if the data were divided into more batches, differences between the two would occur.

III. PCA-LDA

LDA (Linear Discriminant Analysis) is another dimensionality reduction technique as PCA. The difference between the two is that while PCA finds the direction for maximum data variance, LDA finds the direction that optimally separates data of different classes [2]. Using PCA followed by LDA, we seek to obtain both the generative and discriminative features from the data for better classification.

LDA is achieved by utilizing the scatter matrices. Simply stated, by maximizing the ratio between the within-class scatter matrix S_B and the between-class scatter matrix S_W. For detailed explanation of the algorithm, refer to the lecture slide [2].

A. Experimental Setup

The same face data is used and divided into training and testing data as in *I. Computationally Efficient Eigenfaces*. Two important parameters used are, *m_pca* and *m_lda*, the number of best eigenvectors to keep for PCA and LDA respectively.

B. Experimental Goal

We mainly focus on the face recognition accuracy estimated by the NN classifier by varying the parameters *m_pca* and *m_lda*. After that, we compute the confusion matrix and compare time and memory with that of naïve PCA calculated above. We also check whether the scatter matrices have the correct rank and the example success and failure cases.

C. Result and Discussion

The evaluation results in Fig. 3 first shows that the increase in *m_pca* decreases the face recognition accuracy for any *m_lda*. As the scope of decline increases as *m_lda* is small, we assume that underfitting happens when too much data information is kept after PCA. Which is also opposed to the purpose of executing PCA before LDA.

Also, Fig. 3 shows that the increase in *m_lda* tends to increase the face recognition accuracy when *m_pca* is 140 or more. This is because the increase in *m_lda* leads to better extraction of discriminative features of the data. However, when *m_pca* is smaller than 140, this tendency disappears. This is because if too little information is extracted from the PCA, the discriminative features would be damaged before LDA is applied afterwards. Importance of maintaining discriminative features is the difference compared to the naïve PCA, where the highest accuracy was achieved when the parameter *m* was around 100.

Fig. 7.1 shows the confusion matrix for PCA-LDA when (*m_pca*, *m_lda*) = (148, 50), which are the parameters when the face recognition accuracy is the highest. As the confusion matrix visualizes the performance of PCA-LDA, we can have a better understanding of the result.

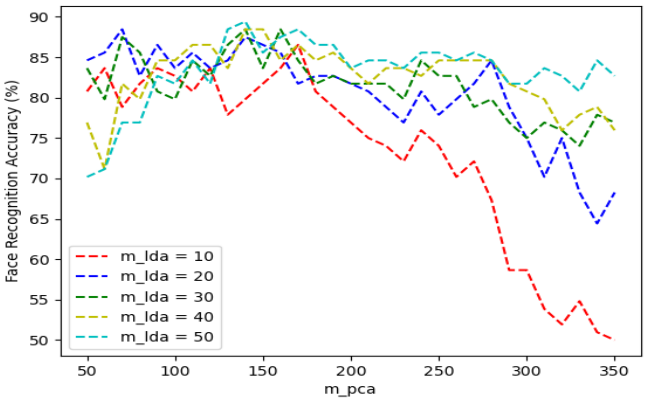


Fig. 3. Evaluation results of PCA-LDA. The face recognition accuracy is calculated according to parameters *m_pca* and *m_lda*. Conditions that m_pca is smaller or equal to the number of training data samples minus the number of classes, which is 416 – 52 = 364 and that m_lda is smaller or equal to the number of classes minus one, which is 51 is met.

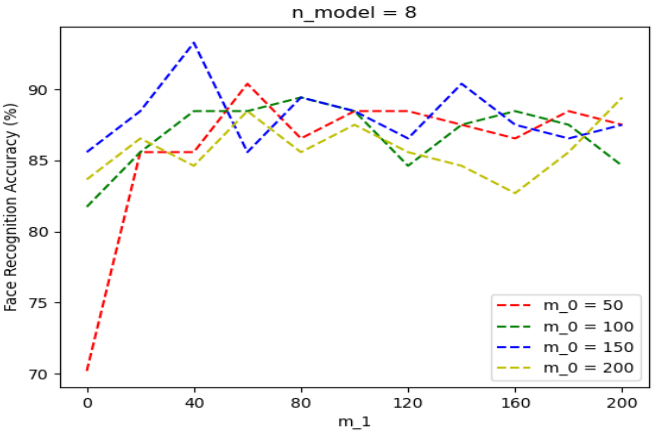


Fig. 4.1. Face recognition accuracy of PCA-LDA ensemble when parameter *n_model* is 8.

The rank of the scatter matrices is 364 and 51 for S_W and S_B respectively. For this experiment, *N*, the number of training data samples is 416 and *c*, the number of classes is 52. Which leads to the result that rank of S_W is 364 and rank of S_B is 51.

Lastly, we compare the training time and memory of PCA-LDA and naïve PCA. As PCA-LDA contains naïve PCA in the algorithm, the training time of PCA-LDA is always bigger than that of PCA. We have discussed earlier that the parameters do not affect the training time of PCA. Therefore, the difference in training time would be the time needed for training LDA, which is proportional to *m_pca*. The memory needed depends completely on the parameters of each model. For naïve PCA, the memory needed is proportional to *d*, number of features, *N*, and *m*. For PCA-LDA, the memory needed is also proportional to *d*, *N*, *m_pca* and *m_lda*. Fig. 8 shows the example of success and failure cases.

IV. PCA-LDA Ensemble

Ensemble classification trains a set of classifiers randomly different from one another and combines their results to get a more generalized final model and avoid overfitting. The results are combined using a fusion rule, e.g., sum, majority voting, min, max.

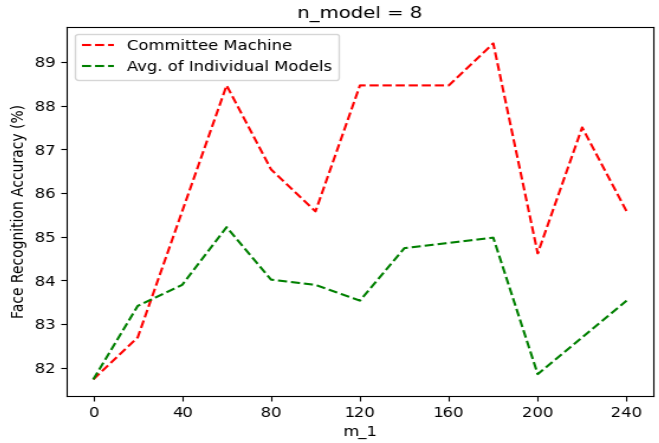


Fig. 4.2. Comparison between the accuracy of the committee machine and the average accuracy of individual models used to build the committee machine when *m_0* is 100 and *m_lda* is 50.

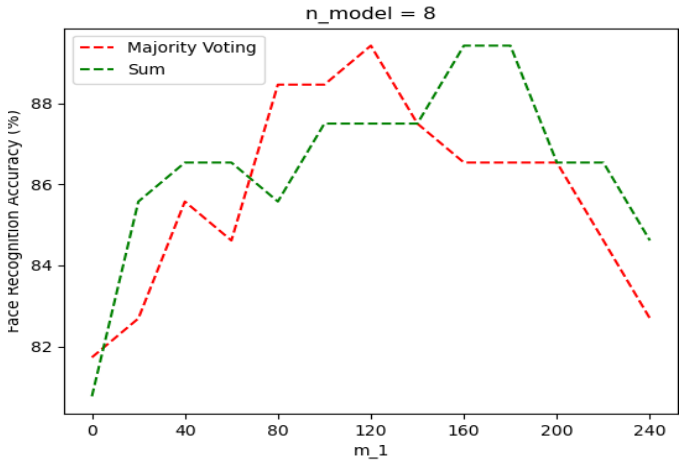


Fig. 4.3. Comparison between the accuracy of the committee machine when using majority voting and sum respectively, when m_0 is 100.

By giving randomness to the eigenvectors chosen in the PCA process, we execute the PCA-LDA ensemble. The fusion rules for combining the set of classifiers are majority voting and sum. For detailed explanation of the algorithm, refer to the lecture slide [3].

A. Experimental Setup

The same face data is used and divided into training and testing data as in *I. Computationally Efficient Eigenfaces*. Four important parameters used are, *m_0, m_1* and *m_lda* and *n_model*. *m_0* is the fixed number of best eigenvectors selected and *m_1* is the number of randomly selected eigenvectors from the rest. Both are used for PCA, and linking it to *m_pca*, *m_pca* is the sum of *m_0* and *m_1*. *m_lda* is the number of best eigenvectors to keep from LDA.

B. Experimental Goal

We aim to check the effectiveness of ensemble classification with respect to the randomization in the feature space by varying parameters *m_0*, *m_1* and the number of base models. Also, we compare the accuracy between the combined model and the average accuracy of individual models in the set of combined models. Fusion rules, majority voting and sum are compared. Finally, we visualize the confusion matrix.

C. Result and Discussion

Fig 4.1 shows the plot when *n_model* is 8. It shows the face recognition accuracy according to *m_0* and *m_1*. *m_lda* is set to 50 because we have seen earlier that high *m_lda* returns higher accuracy and 51 is the upper bound. First, compare it to Fig 3.1. The sum of *m_0* and *m_1* corresponds to *m_pca*. In Fig 3.1, for large *m_pca* when *m_lda* is 50, the face recognition accuracy is between 80-85 percent. On the other hand, in Fig 4.1, even though the sum of the two parameters is large, the accuracy is mostly over 85 percent. This shows that the ensemble method is better than using a single PCA-LDA classifier. Fig. 9 shows when *n_model* is 3 and 5.

Second, compare Fig 4.1 to Fig. 9, where the only difference in the setting is the parameter *n_model*. When there are a greater number of models for ensemble classification, we can infer that better generalization happens, and that the face recognition accuracy would increase. Through the plot, we can actually check that this is true. The accuracies for the same parameters are higher when the *n_model* is bigger.

Fig 4.2 supports the fact that PCA-LDA ensemble has a better performance compared to naïve PCA-LDA. Fig 4.3 shows the accuracy difference according to fusion rules, majority voting and sum. The NN classifier outputs one closest class to the new data, but not the probability distribution. The majority voting rule chooses the class that the majority of classifiers used for the ensemble has predicted. However, the sum rule demands the probability of each class, so that it can add up the probability of all classifiers and choose the class with the highest probability. Because it is impossible to use the sum rule directly to the NN classifier, we came up with a conjugated sum rule. This new sum rule adds up the distance between the new data and the training data with same class labels, and chooses the class that the sum of the distance over all classifiers is the least. In Fig 4.3, we can check that the majority voting rule and the sum rule bring similar accuracy. In Fig 10, we can see that if the number of models is small, the majority rule is better.

V. K-means Codebook

In this step, we tried to find the best clustered results by tuning the number of vocabularies which are used for the clustering. We tried 3 ways to visualize how well the classification is done. After these experiments, we found that the classification was best when the vocabulary size is 128.

A. # of vocabularies = 128

Fig 5.1 is the histogram of distribution of 100k descriptions extracted from 150 images (15 images per each class) and the bag of words corresponding to the images in Fig 11. And lastly, we tried to visualize the distribution of bags of words in 3 dimensions in 3 ways. There are 128 vocabularies, and we randomly selected 3 vocabularies for each bag of words, and then marked them on 3 dimensional coordinates. But as we can see in Fig 5.3, it is hard to determine how well the clustering is done based on randomly picked vocabularies. So, we proceed to the PCA to find eigenvectors which have

maximum variance, then we compress a bag of words into those 3 eigenvectors corresponding to the largest 3 eigenvalues. After PCA, data points lie on the 3 branches divided from the center where most points are tangled.

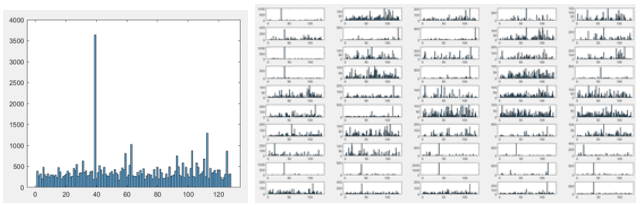


Fig 5.1. The histogram showing how many descriptors belong to each vocabulary when vocabulary size equals 128(left).
Fig 5.2. Bag of words corresponding to the image in Figure 1. Images in the same row are in the same cluster(right).

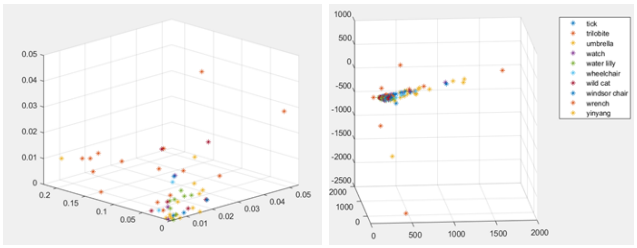


Fig 5.3. Visualization of bag of words in 3D coordinates. 3 Axes are randomly selected from 128 vocabularies(left).
Fig 5.4 Visualization of compressed data of bag of words into 3 major principle component weights.(right)

B. # of vocabularies = 256 and 512

We proceed with the same procedures with k=256 and k=512. And the classified data is followed as.

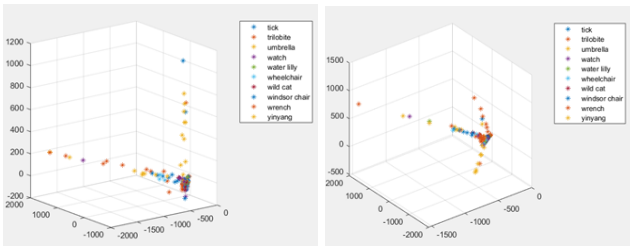


Fig 5.5 Visualization of compressed data of bag of words into 3 major principle component weights when k=256 and k=512.

C. Visualization of Classification

Visualized data after PCA still has difficulty to interpret how well the clustering is done. So, we tried to find the similarities between the mean distributions of training data for each class and the distribution of test data using Kullback-Leibler divergence.We regard the test data belong to the class that records the highest similarity. The confusion matrices are as follows.

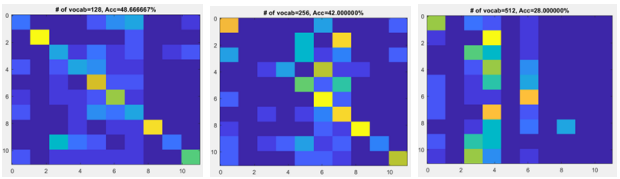


Fig 5.6 The confusion matrix when k=128, 256, and 512. Their accuracy was 48%, 42%, and 28%.

When the vocabulary size is 128, the accuracy was highest at 48.67%. According to the Kullback-Leibler divergence, the data was effectively clustered when using 128 vocabularies.

D. Discussion

Looking at the histogram of classified descriptions (Fig 5.1, 5.2), we can see that it shows the highest frequency in a specific vocabulary.We can intuitively guess that the image which has white background has one specially highest frequency of vocabulary and it indicates the white background of the picture(Fig 11).

By using Kullback-Leibler divergence, 128 vocabularies showed the highest accuracy of 48.67%.

We assumed that by increasing the number of vocabularies, it will improve the accuracy of classification. But, as we can see in Fig 5.6, when we use 128 vocabularies shows the most precise result. From this result, we can guess that because the variance of each class data is high, it is highly overfitted to the train dataset when we use 512 vocabularies.

VI. Randomized Forest for Classification

Decision forest is actively used because it is good at generalization to the unseen new data and has very quick speed. But the internal part of the tree is like the black box, we have to manually set hyperparameters such as depth of tree, number of trees, number of vocabularies, split functions and how many times we try to find proper split function parameters. In this report, we tried to find the relationship between these hyperparameters and the accuracy. To sum up, the accuracy is higher when the tree is less overfit, trees are used as many as possible, and when 128 vocabularies are used to clustering the image features for Caltech image set.

A. Depth of Tree

First, we tried to find the proper depth of the tree in our dataset. Before tuning the depth value, we fixed other hyperparameters as follows.
{# of Vocabularies = 128, # of Trees = 1, # of split function = 5}

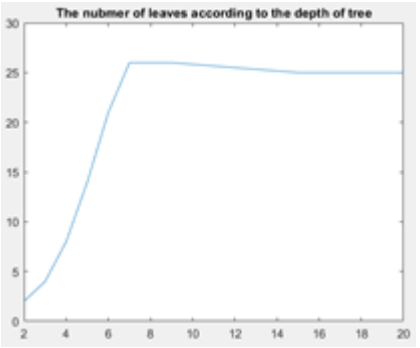


Fig 6.1 The number of constructed leaves according to the depth

As we can see in Fig 6.1, regardless of how deep the tree was, the number of constructed leaves had an upper limit. By the graph, we thought depth 7 is sufficient to classify our dataset. Actually, if depth was 5, there were 5 leaves which contain more than 5 data, so it can be thought as underfit. And if depth was 10, there was no leaf containing more than 6 data. So, we regard it as properly fit.

B. The number of Trees

Next, we measured the accuracy by increasing the number of trees and comparing two cases in which the depth of the tree is 5 and 7.

1) Depth = 7

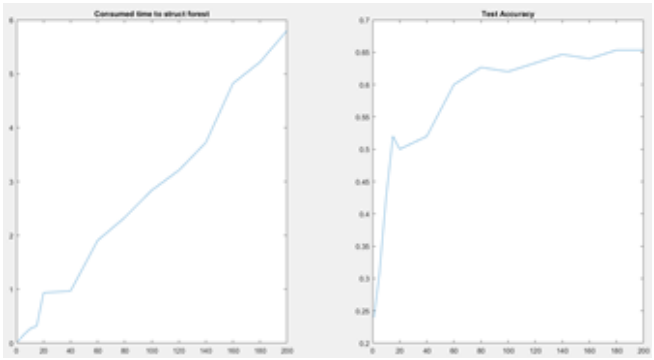


Fig 6.2 The consumed time(left) and test accuracy of RF(right) by tuning the number of trees

	1	5	10	15	20	40	60	80
Acc	0.24	0.30	0.43	0.52	0.50	0.52	0.60	0.63
	100	120	140	160	180	200		
Acc	0.62	0.63	0.65	0.64	0.65	0.65		

As we can see in the above graph and tables, the consumed time to construct forest is increased linearly. And accuracy is generally increased as we combine more trees. When we use 200 trees, it records 65% of accuracy. And in Fig 6.3, we can see the confusion matrix when we use 1,10, 20, 180 trees. In the case of 180 trees, 'trilobite' and 'yinyang' classes are classified well.

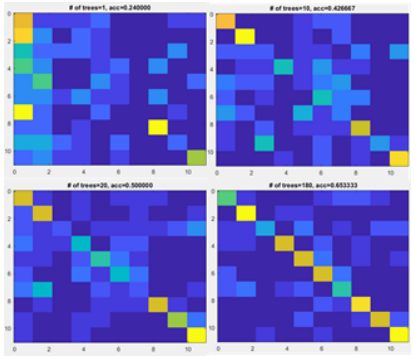


Fig 6.3 The confusion matrix when the number of trees are 1 , 10, 20, 180. The accuracy was 0.24, 0.43, 0.5, 0.65.

2) Depth = 5

The accuracy was better than when the depth was 7.

	1	5	10	15	20	40	60	80
Acc	0.19	0.27	0.43	0.45	0.36	0.53	0.63	0.68
	100	120	140	160	180	200		
Acc	0.65	0.67	0.67	0.67	0.68	0.69		

We can find that dividing train data as much as possible in the tree is not the optimal way for higher accuracy.

C. The Number of Vocabularies

1) # of vocabularies = 256

	1	5	10	15	20	40	60	80
Acc	0.18	0.26	0.26	0.39	0.32	0.53	0.51	0.53
	100	120	140	160	180	200		
Acc	0.53	0.53	0.56	0.59	0.58	0.58		

2) # of vocabularies = 512

	1	5	10	15	20	40	60	80
Acc	0.13	0.21	0.28	0.26	0.36	0.41	0.47	0.47
	100	120	140	160	180	200		
Acc	0.44	0.51	0.52	0.48	0.50	0.48		

As we can see in the above tables, accuracy has steadily increased but was not as high as when we use 128 vocabularies to make a bag of words. So, we can see that the result of Kullback-Leibler divergence in Q4 was quite reliable.

D. Two-Pixel Test

Next, we tried another weak learner way, 'Two-pixel test' instead of 'Axis-aligned' way. Hyperparameters are set as follows.

{# of Depth = 5, # of split function = 5, # of vocabularies = 128 # of trees = 200}

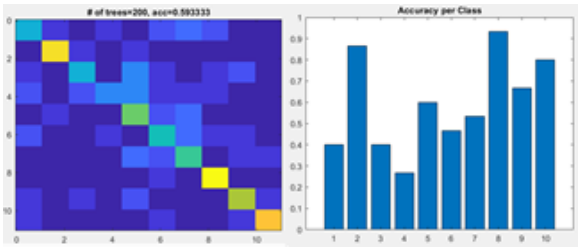


Fig 6.4 The confusion matrix and histogram of test dataset accuracy when using the 'two-pixel test'

In that result, it shows accuracy of 59% and it takes 5.62 seconds for 200 trees. Compared to the 'Axis-aligned' way, it takes 0.5 sec more, but the accuracy was decreased by 10%. We thought that the 'Two-pixel test' should randomly choose two components which are used for split function, so if we increase the number of split functions, the accuracy will be improved. So, we increased the number of split functions by 25. But the improvement was little(Fig 12).

E. Discussion

We tried to find rough causal relationships between hyperparameters and the accuracy.

First, in the case of the depth, since the number of leaves converges to the upper bound, depth doesn't need to be too big. Additionally, the depth in the underfit situation showed higher accuracy than the depth just before convergence.

Second, as we increase the number of trees, the accuracy shows the increasing trend.

And, in the case of the number of vocabularies, as we increase the number of vocabularies, the accuracy has decreased. We think this is because we only use 150 images for training, and 128 vocabularies are enough to effectively classify those images. If we use more vocabulary, then the tree will get overfitted.

And finally, we compared the two ways of weak learners, 'Axis-aligned' and 'Two-pixel test'. We thought if we use a more complex split function, the accuracy would be better. But it was faster and more accurate when we used 'Axis-aligned'.

References

[1] Online Learning

[2] Discriminant Analysis

[3] Ensemble Learning

Appendix I - Confusion matrices

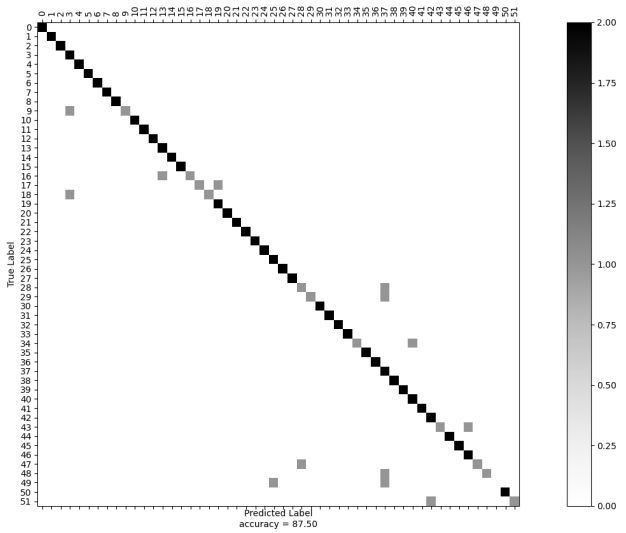


Fig. 7.1. Confusion matrix of PCA-LDA when *m_pca* = 148 and *m_lda* = 50.

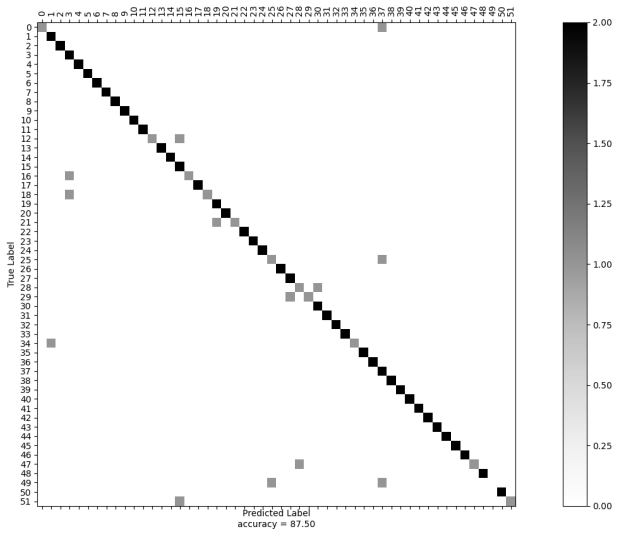


Fig. 7.2. Confusion matrix of PCA-LDA ensemble when *m_0* = 150, *m_1* = 140 and *m_lda* = 50 using majority voting rule

Appendix II - Example of Success and Failure cases

Fig. 8. Example success and failure cases of PCA-LDA. The top is the success case for label 23 and the bottom is the failure case for label 11. For both cases, the left side is the true image and the right side is the reconstructed image.

Appendix III- PCA-LDA Ensemble, *N_MODEL*

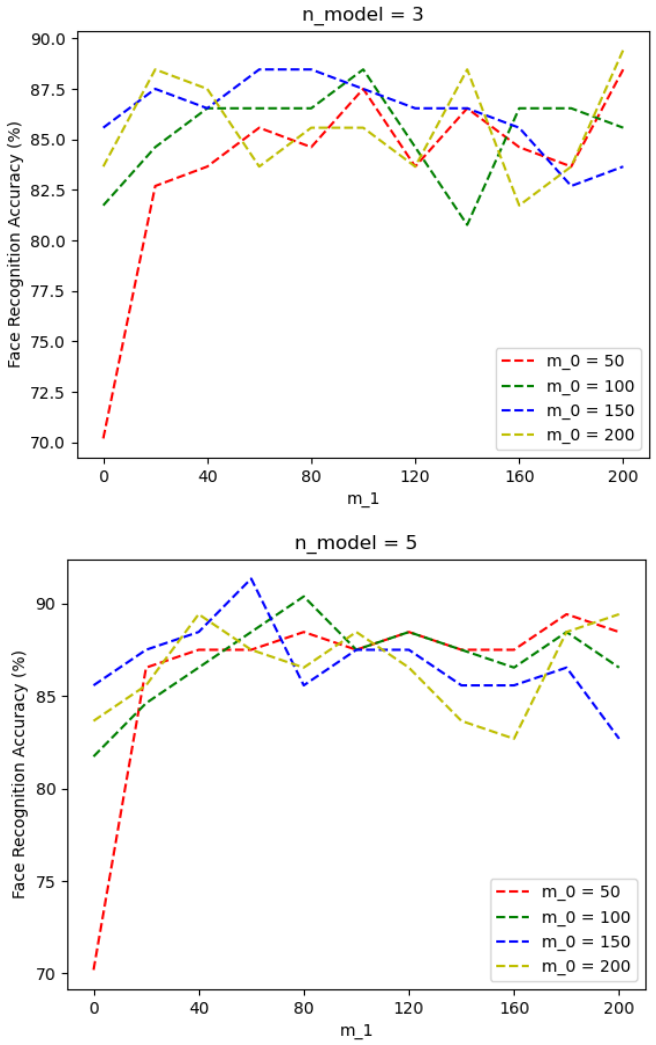


Fig. 9. Face recognition accuracy of PCA-LDA ensemble when parameter *n_model* is 3 and 5 respectively.

Appendix IV- Confusion Rule

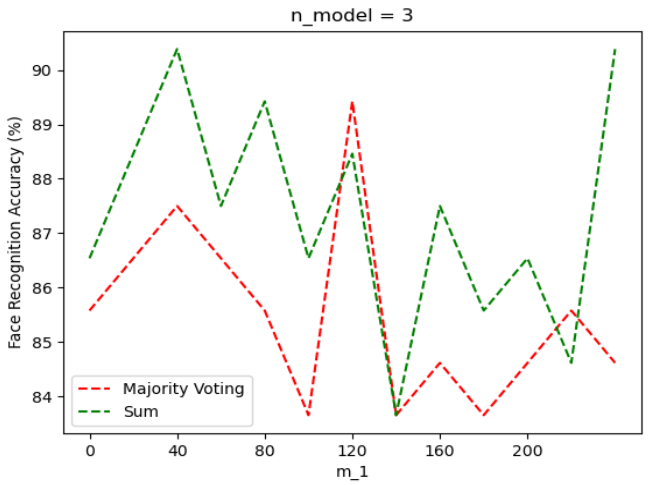


Fig. 10. Comparison between the accuracy of the committee machine when using majority voting and sum respectively, when *m_0* is 100.

Appendix V -Visualization of caltech data images

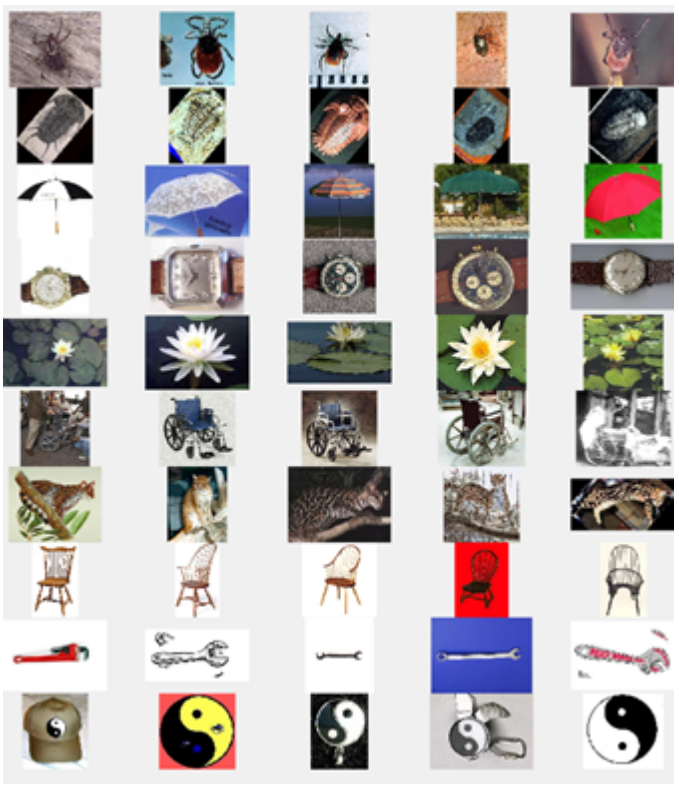


Fig 11. Randomly selected images used for training

Appendix VI - Confusion matrix when the split function is 'two-pixel test'

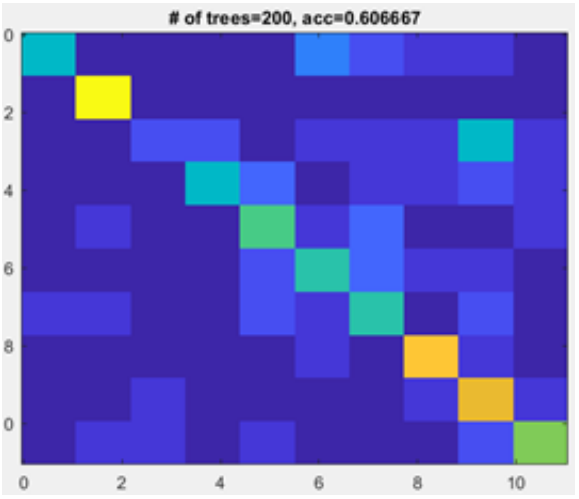


Fig 12. The Accuracy and the confusion matrix when the split function is 'two-pixel test'

DDDFGE