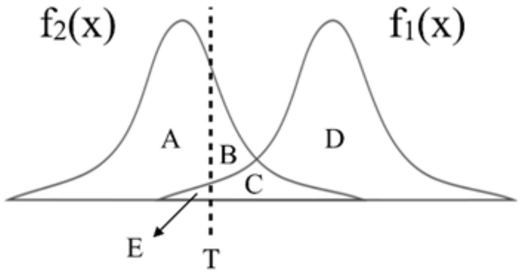
Computer Vision: from Recognition to Geometry
HW2

Name: 毛弘仁 Department: 電機四 Student ID: B04901117

Problem 1

(a) Assume X is a continuous random variable that denotes the estimated probability of a binary classifier. The instance is classified as positive if $X > T$ and negative otherwise. When the instance is positive, X follows a PDF $f_1(x)$. When the instance is negative, X follows a PDF $f_2(x)$. Please specify which regions (A ~ E) represent the cases of *False Positive* and *False Negative*, respectively. Clearly explain why.

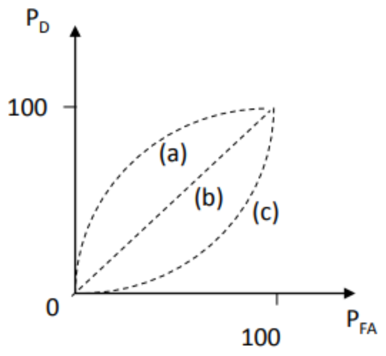


Ans:

False Positive: B+C, because they are both to the right of T (classified as positive) and under f_2(x) (true negative).

False Negative: E, because it is to the left of T (classified as negative) and under f_1(x) (true positive).

(b) There are three ROC curves in the plot below. Please specify which ROC curves are considered to have reasonable discriminating ability, and which are not. Also, please answer that under what circumstances will the ROC curve fall on curve (b)?



Ans:

We see that (a) has reasonable discriminating ability, because given any point on the curve, P_D has a higher value than P_{FA}, which means there is a higher probability of the classifier being right than wrong. The curve of (b) means the classifier has an equal probability of being correct or incorrect, thus it does not perform better than random guessing; the curve of (c) means the classifier is more often wrong than right, so even random guessing would yield better performance. For these reasons, (b) and (c) do not have reasonable discriminating ability, as they are no better than pure guesswork.

A possible reason for the ROC curve falling on curve (b) may be that $f_1(x)$ and $f_2(x)$ overlap and cannot be differentiated, thus no matter which value of T we choose, performance is always the same as random guessing.

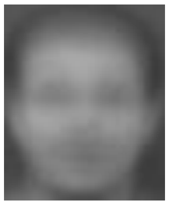
Problem 2

(a) PCA

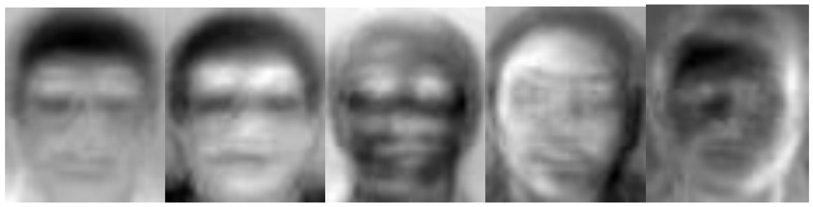
In this task, you need to implement PCA from scratch, which means you cannot call PCA function directly from existing packages.

1. Perform PCA on the training data. Plot the mean face and the first five eigenfaces and show them in the report.

Mean Face



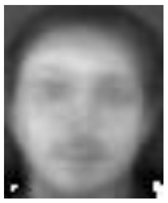
First 5 eigenfaces



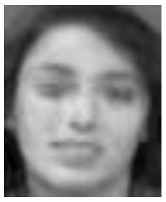
2. Take $\boldsymbol{person_8\_\ image_6}$, and project it onto the above PCA eigenspace. Reconstruct this image using the first n = { 5, 50, 150, all } eigenfaces. For each n, compute the mean square error (MSE) between the reconstructed face image and the original $\boldsymbol{person_8\_\ image_6}$. Plot these reconstructed images with the corresponding MSE values in the report.

Original

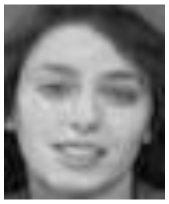
n=5, MSE=693.7



n=50, MSE=119.2

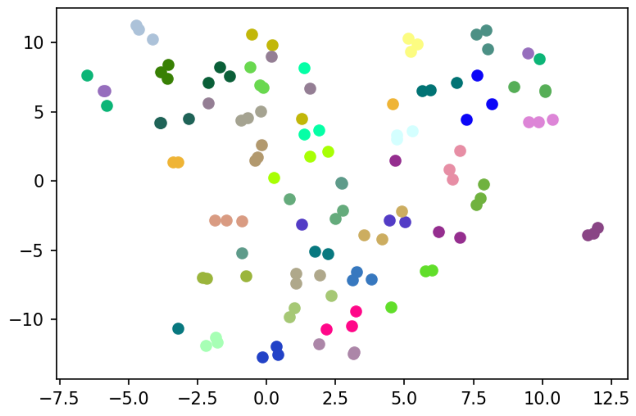


n=150, MSE=38.8



n=279, MSE=0

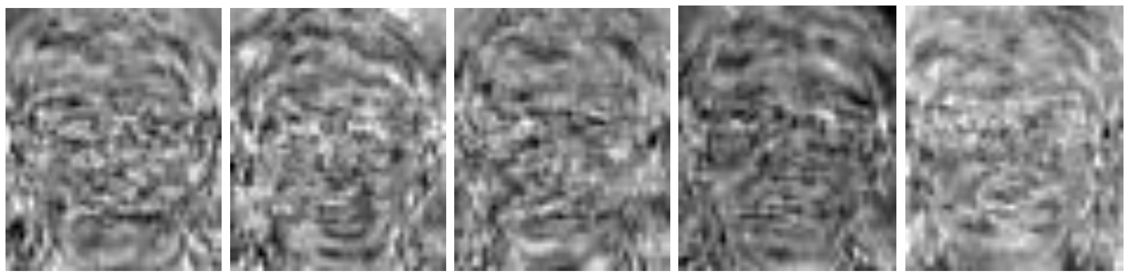
3. Reduce the dimension of the image in testing set to dim = 100. Use t-SNE to visualize the distribution of test images.



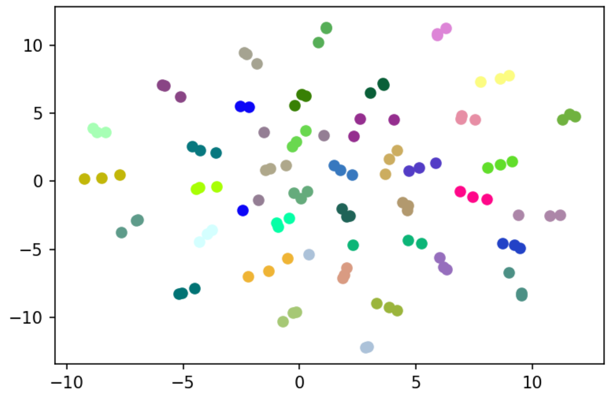
(b) LDA

In this task, you need to implement LDA from scratch, which means you cannot call LDA function directly from existing packages.

1. Implement LDA and plot first 5 Fisherfaces.



2. Use t-SNE to visualize the distribution of the projected testing data, which has the dimension of 30.



(c) To apply the k-nearest neighbors (k-NN) classifier to recognize the testing set images, please determine the best k and n values by 3-fold cross-validation.

For simplicity, the choices for such hyper-parameters are:

k = {1, 3, 5} and n = {3, 10, 39}.

Please show the cross-validation results and explain your choice for (k, n). Also, show the recognition rate on the testing set using your hyper-parameter choice. Please apply the above comparing method on both PCA and LDA.

Do you observe an improved recognition rate using fisherfaces (compared to eigenfaces obtained by PCA)? If so (or if not), what might be the possible explanation?

Ans:

I randomly split the training set into 3 folds for cross-validation. For each fold, I recalculated the eigenvectors for PCA and LDA such that the validation set is unseen, better mimicking the test set. When evaluating on the test set, I chose the hyper-parameters that yielded the best validation accuracy.

PCA validation accuracies

	n=3	n=10	**n=39**
k=1	0.69	0.89	**0.92**
k=3	0.61	0.78	0.87
k=5	0.55	0.70	0.78

Test accuracy using (k=1,n=39): **0.96**

LDA validation accuracies

	n=3	n=10	**n=39**
k=1	0.42	0.78	0.92
k=3	0.42	0.78	0.93
k=5	0.41	0.79	**0.93**

Test accuracy using (k=5,n=39): 0.92

The recognition rate of the validation set is better if I use Fisherfaces, but the recognition rate of the test set is better if I use eigenfaces obtained by PCA. A possible reason could be that LDA fits too well to the training set, not being flexible enough to perform well on the test set. In fact, this phenomenon has been explored in previous research: https://pdfs.semanticscholar.org/cadb/9a014a4c5bbec57aaf30391f472fa4b69b4d.pdf

Problem 3

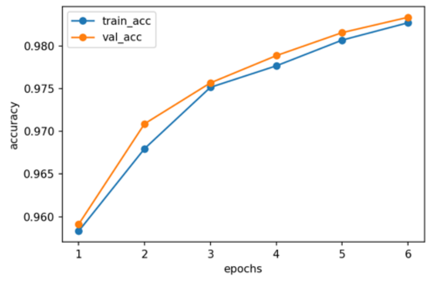
(a) Build a CNN model and train it on the given dataset. Show the architecture of your model in the report.

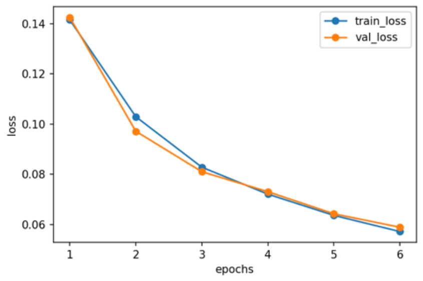
Ans:

Conv2D(1, 10, kernel_size=5) → Max_Pool2D(kernel_size=2) → Conv2D(10, 20, kernel_size=5) → Dropout2D(p=0.5) → Max_Pool2D(kernel_size=2) → flatten() → Linear(320, 50) → Dropout(p=0.5) → Linear(50, 10)

(b) Report your training / validation accuracy, and plot the learning curve (loss, accuracy) of the training process.

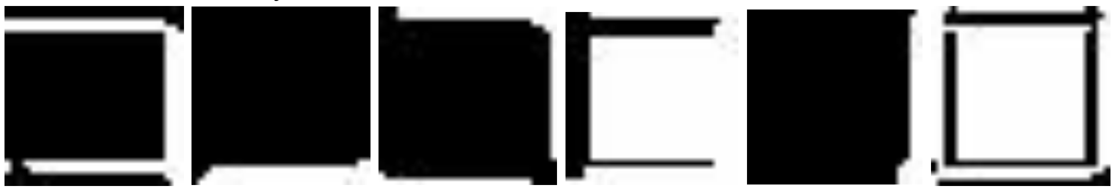
Ans:
After 6 epochs, the training accuracy is 0.9828, and the validation accuracy is 0.9834.



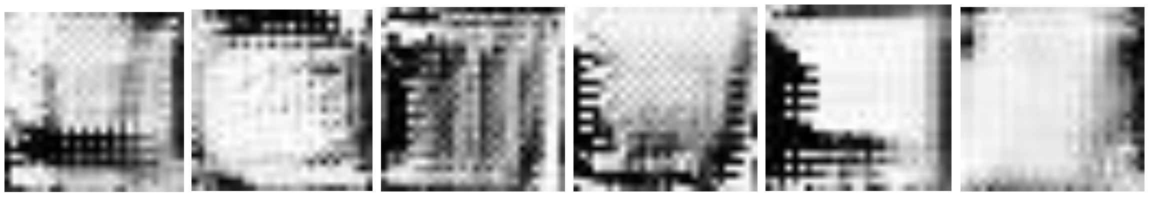


(c) Visualize at least 6 filters on both the first and the last convolutional layers.

First convolutional layer

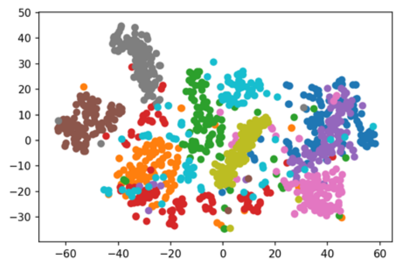


Last convolutional layer

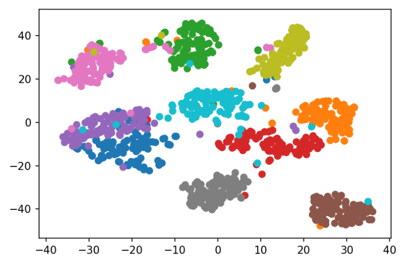


(d) Visualize high-level and low-level features of 1000 validation data (100 for each class) extracted from different layers, and explain what you have observed from the two t-SNE plots.

First convolutional layer



Last convolutional layer



We find that the classes are more separated from each other after passing through higher-level convolutional layers.