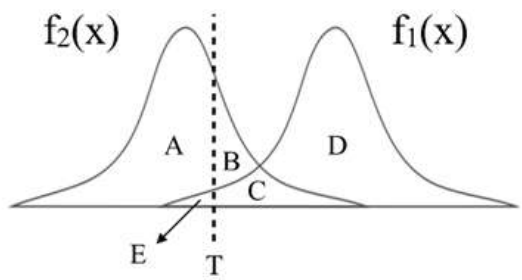
Computer Vision: from Recognition to Geometry
HW2

Name: 潘彥銘 Department: Electrical Engineering Student ID: B05901182

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.

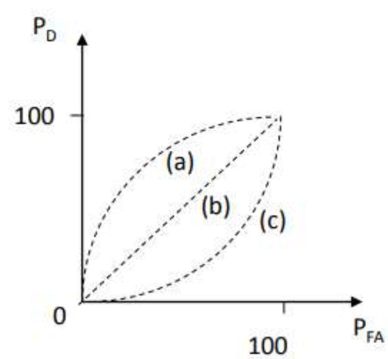


Ans:

Region B and C is False Positive, because region B, C and D will be classified as positive by the classifier. But only region B and C are negative instance.

Region E is False Negative, because region A and E will be classified as negative by the classifier. But only region E is positive instance.

(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:

Curve (a) and (b) are reasonable

Curve (a) has better discriminating ability since when threshold goes to right side, $P_D > P_{FA}$. Which means we can have higher chance to distinguish positive and negative instance.

Curve (b) has no discriminating ability since $f_1(x)$ and $f_2(x)$ overlap, and we can't tell the difference between $f_1(x)$ and $f_2(x)$.

When $f_1(x)$ and $f_2(x)$ overlap, no matter where the threshold T is located, P_D and P_{FA} are the same, so the ROC curve will become curve (b).

Problem 2

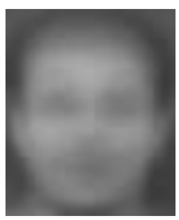
(a) PCA

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.

Ans:

(1) Mean face:



(2) First five eigenfaces

Eigenface_1 Eigenface_2 Eigenface_3 Eigenface_4 Eigenface_5

2. Take $\boldsymbol{person}_{8}\_\ \boldsymbol{image}_{6}$, 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 $\boldsymbol{person}_{8}\_\ \boldsymbol{image}_{6}$. Plot these reconstructed images with the corresponding MSE values in the report.

Ans:

n=5, MSE = 693.702

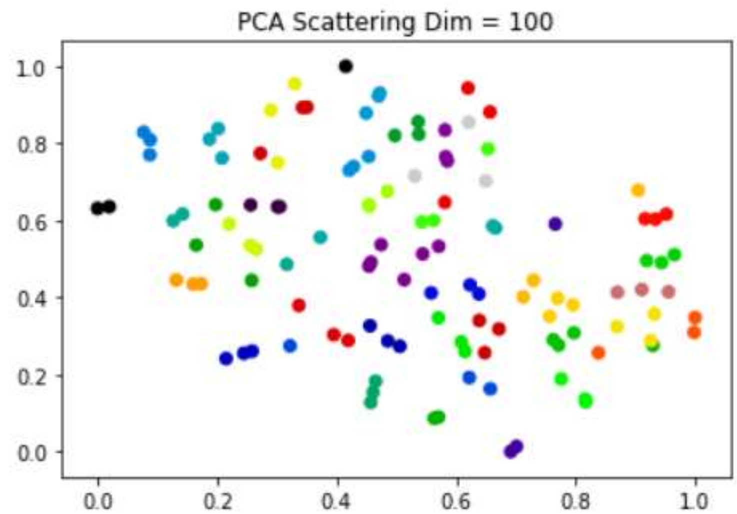
n=50, MSE = 119.2

n=150, MSE = 40.397

n=279, MSE = $5*10^{-24}$

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

Ans:

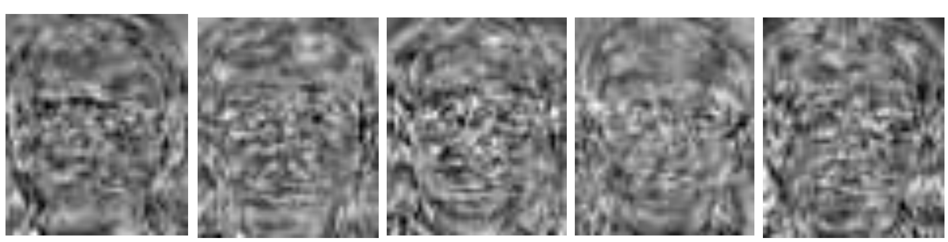


(b) LDA

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.

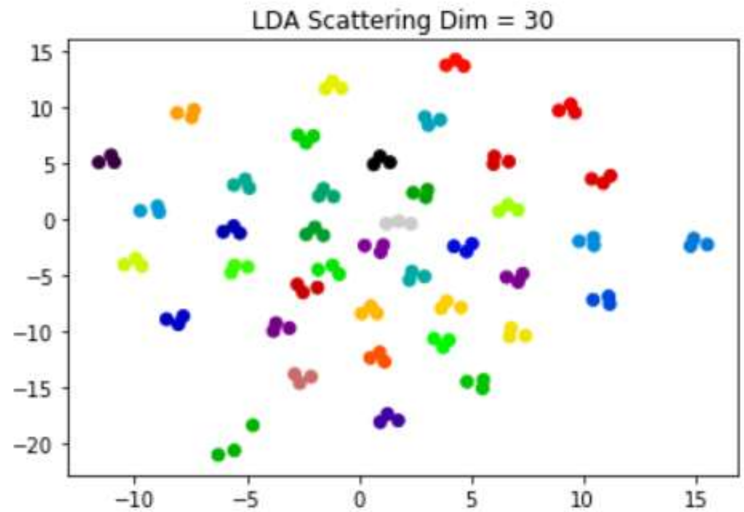
Ans:



Fisherface_1 Fisherface_2 Fisherface_3 Fisherface_4 Fisherface_5

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

Ans:



(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 cut the training data into three group, first group contains the first two images of each class, so first group has 80 images. Second group contains the third and the forth images of each class, so second group has 80 images. The third group contains the rest image of training data.

(1) PCA:

(n,k)	Recognition rate
(3,1)	0.7472222222222222
(3,3)	0.6444444444444445
(3,5)	0.5527777777777777
(10,1)	0.8902777777777778
(10,3)	0.7833333333333333
(10,5)	0.7541666666666668
(39,1)	0.9347222222222222
(39,3)	0.8847222222222223
(39,5)	0.8194444444444443

I choose (n,k) = (39,1), the test accuracy is 0.9533333333333. It is the highest among nine experiments.

(2) LDA:

(n,k)	Recognition rate
(3,1)	1.0
(3,3)	1.0
(3,5)	1.0
(10,1)	1.0
(10,3)	1.0
(10,5)	1.0
(39,1)	0.7166666666666667
(39,3)	0.6513888888888889
(39,5)	0.5805555555555556

It seems that overfit occur. I think the reason is I have used validation data to train my LDA model. So the recognition rate of validation data is very high.

Problem 3

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

Ans:

```
def Lenet_5():
    img_input = Input( shape = (28, 28, 1) )
    co1 = Conv2D(6, (5, 5), padding = 'valid', name = 'co1')(img_input)
    co1 = Activation('tanh')(co1)
    mp1 = MaxPooling2D(pool_size = 2, strides = 2, padding = 'SAME')(co1)
    co2 = Conv2D(16, (5, 5), padding = 'valid', name = 'co2')(mp1)
    co2 = Activation('tanh')(co2)
    mp2 = MaxPooling2D(pool_size = 2, strides = 2, padding = 'SAME')(co2)
    flat = Flatten()(mp2)
    fc1 = Dense(120, activation = 'tanh', name = 'fc1')(flat)
    fc2 = Dense(84, activation = 'tanh', name = 'fc2')(fc1)
    fc3 = Dense(10, activation = 'softmax', name = 'fc3_sm')(fc2)

    model = Model(img_input, fc3)
    return model
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 28, 28, 1)         0
_________________________________________________________________
co1 (Conv2D)                 (None, 24, 24, 6)         156
_________________________________________________________________
activation_3 (Activation)    (None, 24, 24, 6)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 12, 12, 6)         0
_________________________________________________________________
co2 (Conv2D)                 (None, 8, 8, 16)          2416
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 16)          0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 16)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0
_________________________________________________________________
fc1 (Dense)                  (None, 120)               30840
_________________________________________________________________
fc2 (Dense)                  (None, 84)                10164
_________________________________________________________________
fc3_sm (Dense)               (None, 10)                850
=================================================================
Total params: 44,426
Trainable params: 44,426
Non-trainable params: 0
_________________________________________________________________
```

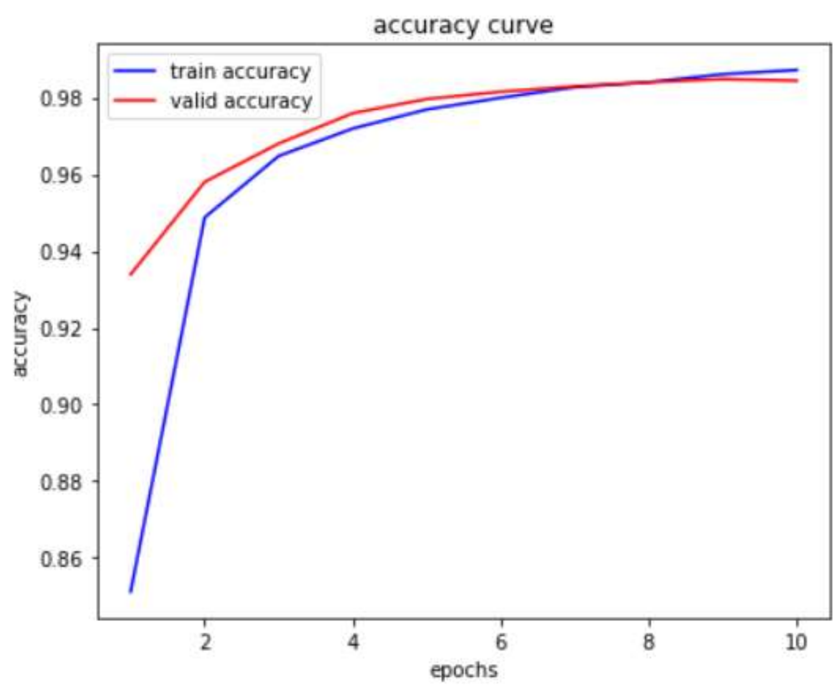
(b) Report your training / validation accuracy, and plot the learning curve (loss, accuracy) of the training process.

Ans:

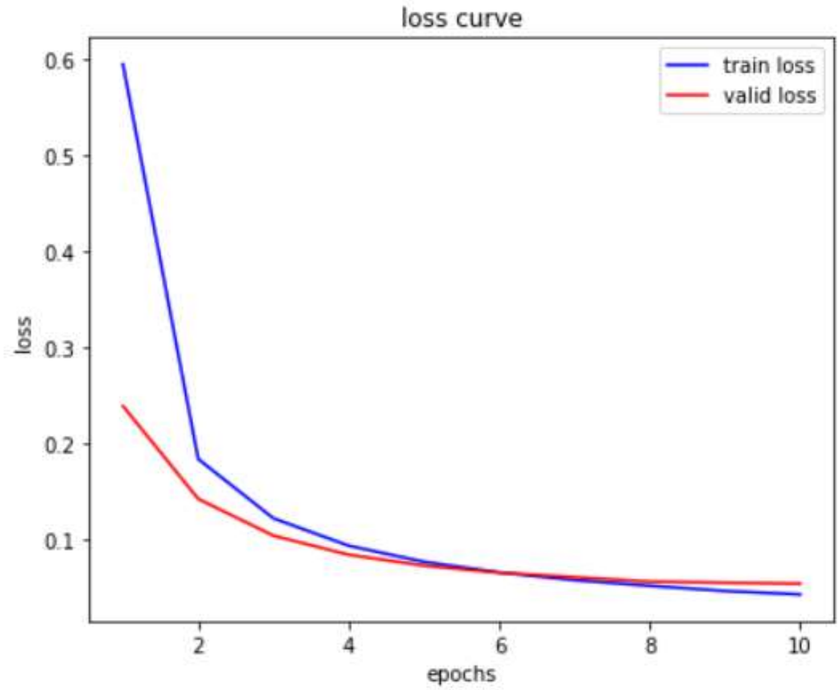
(1) Last epoch training accuracy: 98.72% ，validation accuracy: 98.44%

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000 [==============================] - 42s 850us/step - loss: 0.5954 - acc: 0.8512 - val_loss: 0.2392 - val_acc: 0.9339
Epoch 2/10
50000/50000 [==============================] - 47s 931us/step - loss: 0.1839 - acc: 0.9487 - val_loss: 0.1423 - val_acc: 0.9580
Epoch 3/10
50000/50000 [==============================] - 45s 905us/step - loss: 0.1221 - acc: 0.9648 - val_loss: 0.1040 - val_acc: 0.9680
Epoch 4/10
50000/50000 [==============================] - 46s 927us/step - loss: 0.0935 - acc: 0.9719 - val_loss: 0.0841 - val_acc: 0.9759
Epoch 5/10
50000/50000 [==============================] - 45s 894us/step - loss: 0.0767 - acc: 0.9769 - val_loss: 0.0727 - val_acc: 0.9796
Epoch 6/10
50000/50000 [==============================] - 44s 881us/step - loss: 0.0657 - acc: 0.9799 - val_loss: 0.0654 - val_acc: 0.9815
Epoch 7/10
50000/50000 [==============================] - 49s 977us/step - loss: 0.0576 - acc: 0.9827 - val_loss: 0.0607 - val_acc: 0.9829
Epoch 8/10
50000/50000 [==============================] - 45s 895us/step - loss: 0.0517 - acc: 0.9840 - val_loss: 0.0561 - val_acc: 0.9840
Epoch 9/10
50000/50000 [==============================] - 49s 990us/step - loss: 0.0463 - acc: 0.9861 - val_loss: 0.0550 - val_acc: 0.9848
Epoch 10/10
50000/50000 [==============================] - 49s 974us/step - loss: 0.0427 - acc: 0.9872 - val_loss: 0.0540 - val_acc: 0.9844
```

(2) Accuracy curve

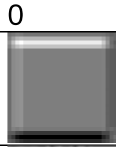
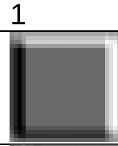
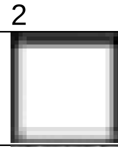
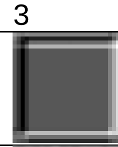
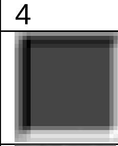
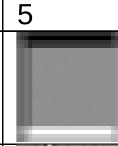
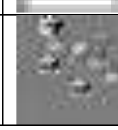


(3) Loss curve



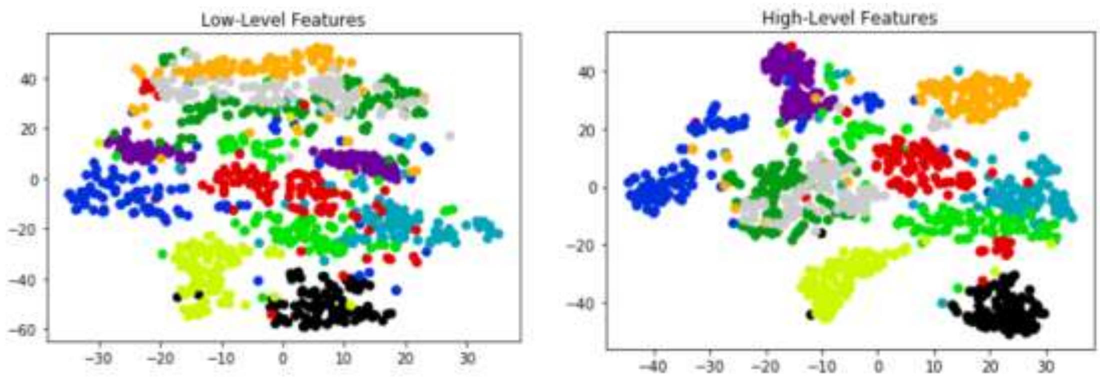
(c) Visualize at least 6 filters on both the first and the last convolutional layers.

Ans:

Layer\Filter	0	1	2	3	4	5
co1						
co2						

(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.

Ans:



From these two pictures above we can observe that features extracted from high level layer in the same class getting closer to each other, features in different class are getting far from each other. Which means the network becomes better at highlighting the features.