CA2 Project Report

A0177273X Luo Ke

Part 1: PCA for Feature Extraction, Visualization and Classification

Apply PCA to reduce the dimensionality of vectorized images from 784 to 2 and 3 respectively. Visualize the projected data vector in 2d and 3d plots.

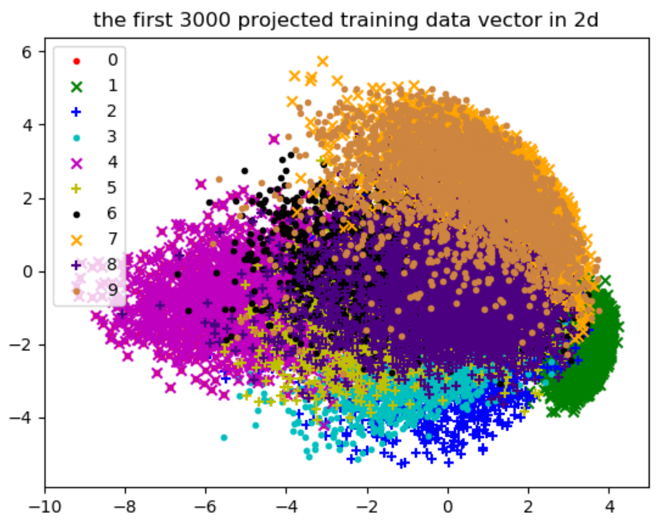


Figure 1: projected data vector in 2d

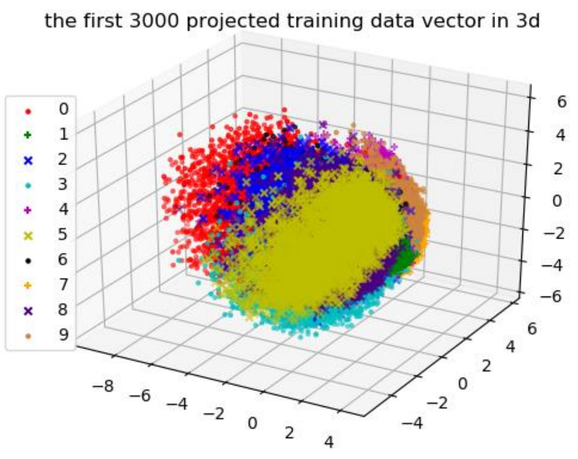


Figure 2: projected data vector in 3d

visualize the eigenvectors of sample covariance matrix used in PCA.

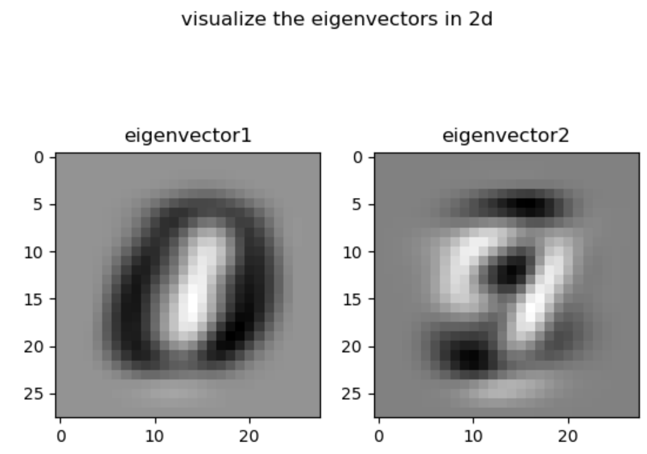


Figure 3: visualize the eigenvectors in 2d

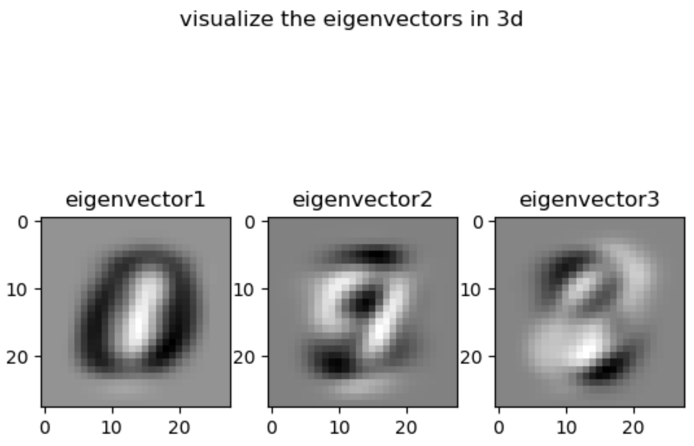
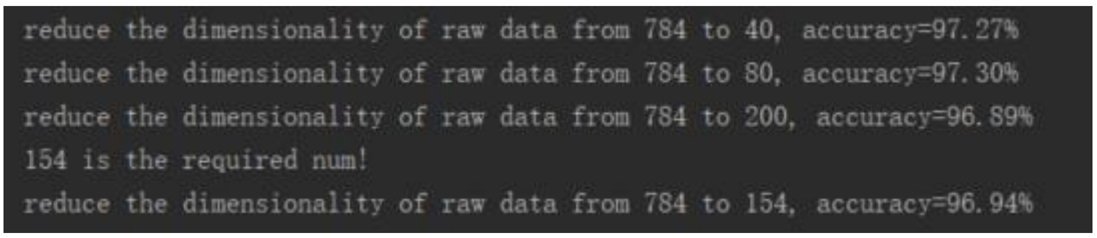


Figure 4: visualize the eigenvectors in 3d

Apply PCA to reduce the dimensionality of raw data from 784 to 40, 80 and 200 respectively. Classifying the test images using the rule of nearest neighbor. Report the classification accuracy.



```
reduce the dimensionality of raw data from 784 to 40, accuracy=97.27%
reduce the dimensionality of raw data from 784 to 80, accuracy=97.30%
reduce the dimensionality of raw data from 784 to 200, accuracy=96.89%
154 is the required num!
reduce the dimensionality of raw data from 784 to 154, accuracy=96.94%
```

Figure 5: PCA plus nearest neighbor classification results printed in pycharm console

Dimensionality	40	80	200	154
Accuracy	97.27%	97.30%	96.89%	96.94%

Table 1: PCA plus nearest neighbor classification results

In, Figure 5 and Table 1, we can see the required value of d preserving over 95% of the total energy after dimensionality reduction is 154. And after applying PCA to reduce data dimension from 784 to 154, the classification result based on NN is 96.94%.

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq Threshold\ (e.g., 0.95)$$

The value of d, 154, is given by the above formula.

Can you devise other criteria for automatically determining the value of d?

a. Kaiser's Rule, retaining all singular values greater than 1.
b. Select d to make the average squared prediction error / variation minimum, or reach a threshold

Part 2: LDA for Feature Extraction and Classification

Apply LDA to reduce data dimensionality from 784 to 2, 3 and 9. Visualize distribution of the data with dimensionality of 2 and 3 respectively.

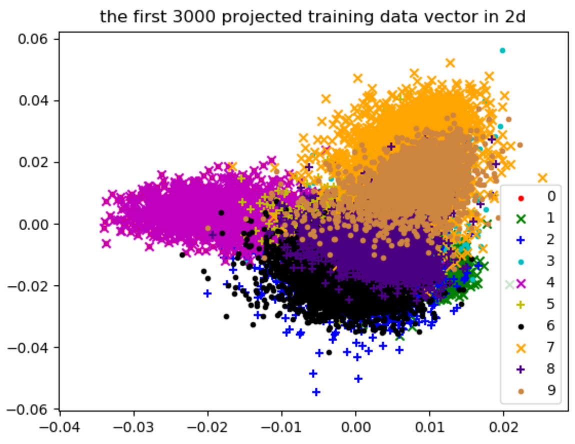


Figure 6: projected data vector in 2d

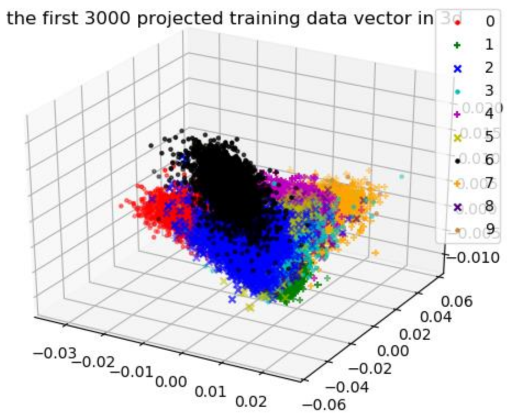
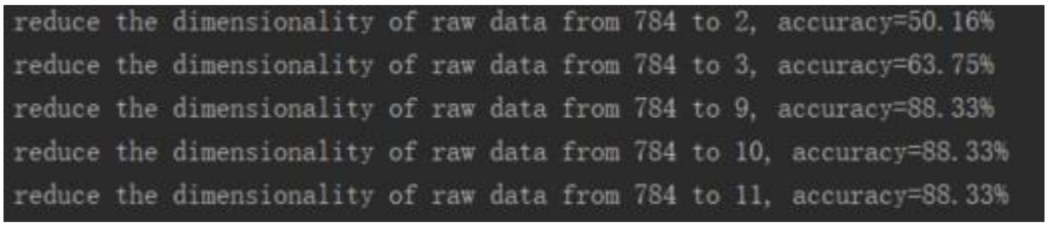


Figure 7: projected data vector in 3d

Report the classification accuracy for data with dimensions of 2, 3 and 9 respectively, based on nearest neighbor classifier.



```
reduce the dimensionality of raw data from 784 to 2, accuracy=50.16%
reduce the dimensionality of raw data from 784 to 3, accuracy=63.75%
reduce the dimensionality of raw data from 784 to 9, accuracy=88.33%
reduce the dimensionality of raw data from 784 to 10, accuracy=88.33%
reduce the dimensionality of raw data from 784 to 11, accuracy=88.33%
```

Figure 8: LDA plus nearest neighbor classification results printed in pycharm console

Dimensionality	2	3	9	10	11
Accuracy	50.16%	63.75%	88.33%	88.33%	88.33%

Table 2: LDA plus nearest neighbor classification results

Test the maximal dimensionality that data can be projected to via LDA. Explain the reasons.

From the Figure 8 and Table 2 we can see that the maximal dimensionality that data can be projected to via LDA is 9. LDA can generate up to C-1 dimensions subspace after dimension reduction (C is the number of categories). Because the MINIST data only have 10 categories (digital number 0~9), it can be only classified to 10 classes and the maximal dimensionality is 9.

Part 3 SVM for Classification

SVM classification results with different parameter values

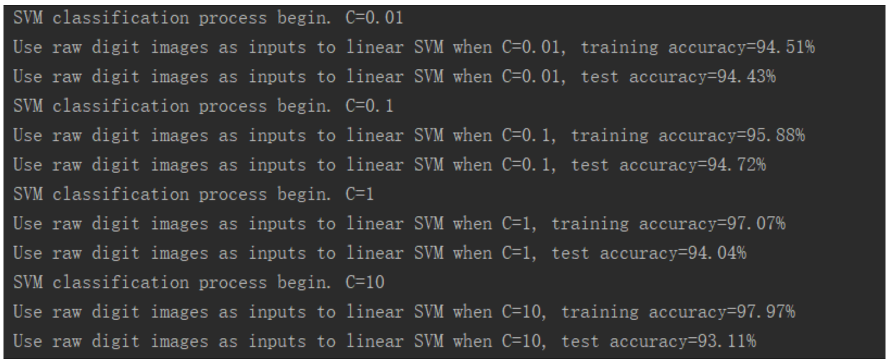
A. Linear SVM Test Accuracy Results:

C	0.01	0.1	1	10
Raw data	94.43	94.72	94.04	93.11
PCA_40	93.28	93.33	93.30	93.31
PCA_80	94.05	94.33	94.36	94.32
PCA_200	94.37	94.65	94.22	93.83

Table 3: Linear SVM results

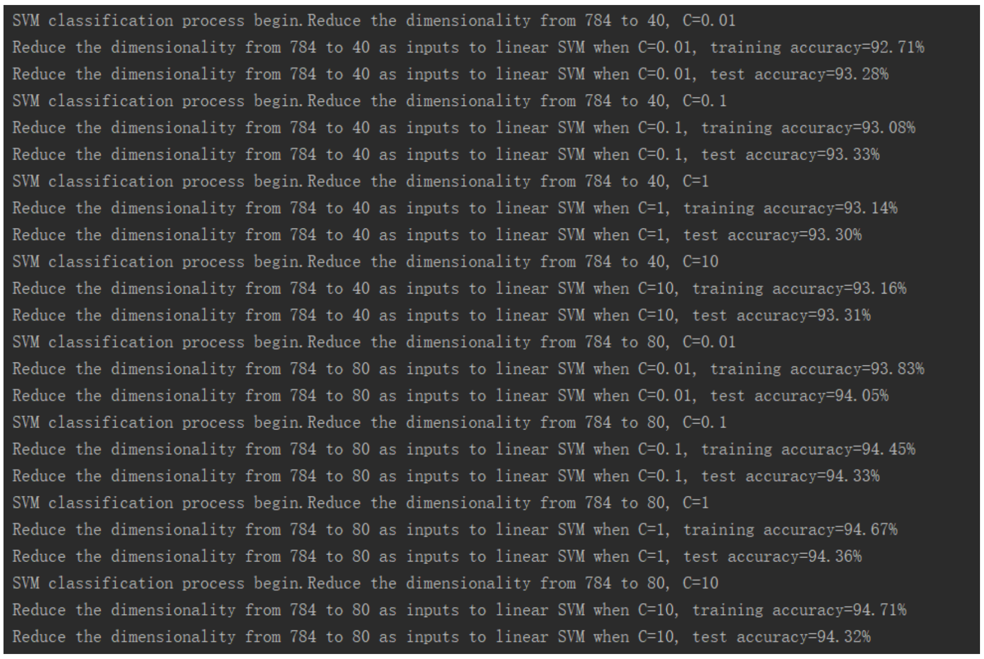
Report the classification accuracy with different parameters and dimensions. Explain the effect of data dimension and parameter C on the final classification accuracy.

From the above table, we can see that for linear SVM, the accuracy on test set is between 93% to 94%. And for the parameter C, when data dimension is fixed. C=0.1 has the highest accuracy. When C is less or greater than 0.1, the test accuracy will become lower. Reduce the dimension of data will lead to a slightly lower test accuracy compare to the raw data.



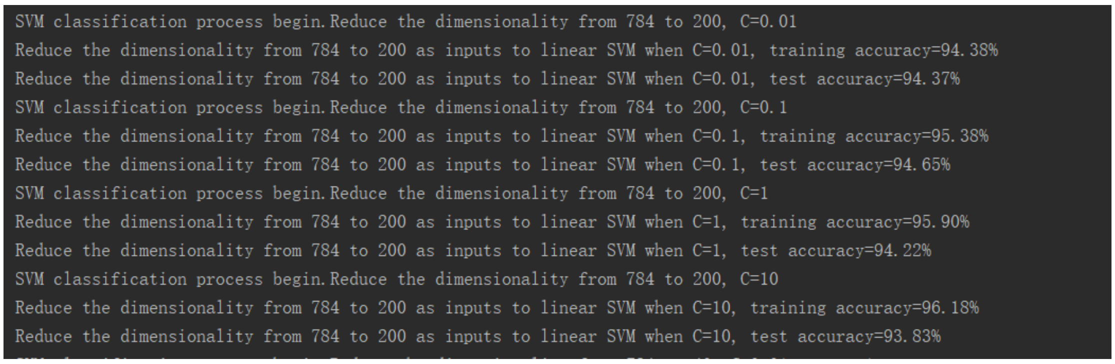
```
SVM classification process begin. C=0.01
Use raw digit images as inputs to linear SVM when C=0.01, training accuracy=94.51%
Use raw digit images as inputs to linear SVM when C=0.01, test accuracy=94.43%
SVM classification process begin. C=0.1
Use raw digit images as inputs to linear SVM when C=0.1, training accuracy=95.88%
Use raw digit images as inputs to linear SVM when C=0.1, test accuracy=94.72%
SVM classification process begin. C=1
Use raw digit images as inputs to linear SVM when C=1, training accuracy=97.07%
Use raw digit images as inputs to linear SVM when C=1, test accuracy=94.04%
SVM classification process begin. C=10
Use raw digit images as inputs to linear SVM when C=10, training accuracy=97.97%
Use raw digit images as inputs to linear SVM when C=10, test accuracy=93.11%
```

Figure 9: raw data as inputs to linear SVM



```
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.01
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=0.01, training accuracy=92.71%
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=0.01, test accuracy=93.28%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.1
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=0.1, training accuracy=93.08%
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=0.1, test accuracy=93.33%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=1
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=1, training accuracy=93.14%
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=1, test accuracy=93.30%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=10
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=10, training accuracy=93.16%
Reduce the dimensionality from 784 to 40 as inputs to linear SVM when C=10, test accuracy=93.31%
SVM classification process begin.Reduce the dimensionality from 784 to 80, C=0.01
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=0.01, training accuracy=93.83%
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=0.01, test accuracy=94.05%
SVM classification process begin.Reduce the dimensionality from 784 to 80, C=0.1
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=0.1, training accuracy=94.45%
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=0.1, test accuracy=94.33%
SVM classification process begin.Reduce the dimensionality from 784 to 80, C=1
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=1, training accuracy=94.67%
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=1, test accuracy=94.36%
SVM classification process begin.Reduce the dimensionality from 784 to 80, C=10
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=10, training accuracy=94.71%
Reduce the dimensionality from 784 to 80 as inputs to linear SVM when C=10, test accuracy=94.32%
```

Figure 10: PCA processed data as inputs to linear SVM



```
SVM classification process begin.Reduce the dimensionality from 784 to 200, C=0.01
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=0.01, training accuracy=94.38%
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=0.01, test accuracy=94.37%
SVM classification process begin.Reduce the dimensionality from 784 to 200, C=0.1
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=0.1, training accuracy=95.38%
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=0.1, test accuracy=94.65%
SVM classification process begin.Reduce the dimensionality from 784 to 200, C=1
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=1, training accuracy=95.90%
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=1, test accuracy=94.22%
SVM classification process begin.Reduce the dimensionality from 784 to 200, C=10
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=10, training accuracy=96.18%
Reduce the dimensionality from 784 to 200 as inputs to linear SVM when C=10, test accuracy=93.83%
```

Figure 11: PCA processed data as inputs to linear SVM

B. Kernel SVM Test Accuracy
   For saving training time, the data dimension in this case is PCA_40.

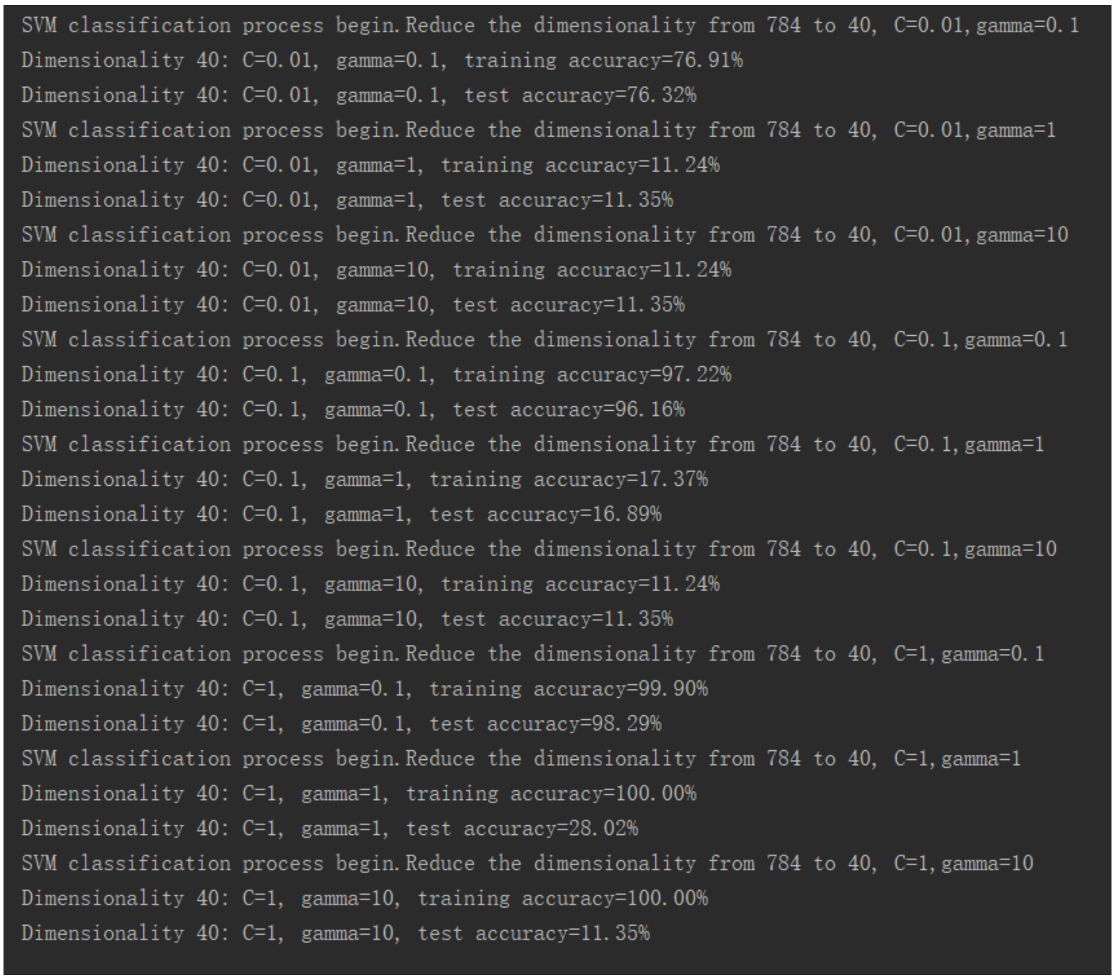
Gamma C	0.01	0.1	1
0.1	76.32%	96.16%	98.29%
1	11.35%	16.89%	28.02%
10	11.35%	11.35%	11.35%

Table 3: Kernel SVM results

Tune the parameters and report the parameters providing best classification performance. Compare the results with the one given by linear SVM and explain the observations.

From the above table, we can see the parameters providing best classification performance is C=1 and Gamma=0.1, and the test accuracy is 98.29%. For linear SVM, C=0.1 has the highest accuracy, and it is 94.72%. Therefore, we can see Kernel

SVM can leads to a much higher test accuracy than linear SVM.
C is the penalty parameter. It represents the degree of tolerance to error. The bigger the C is, the easier it is to have overfitting. For Gamma, it is a parameter of “RBF” function as the Kernel to SVM. The larger the gamma, the fewer the support vectors



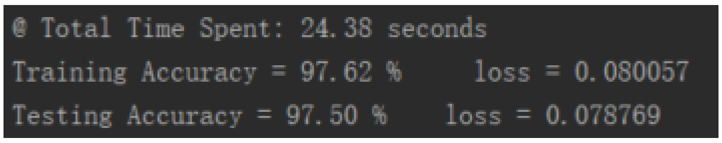
```
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.01,gamma=0.1
Dimensionality 40: C=0.01, gamma=0.1, training accuracy=76.91%
Dimensionality 40: C=0.01, gamma=0.1, test accuracy=76.32%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.01,gamma=1
Dimensionality 40: C=0.01, gamma=1, training accuracy=11.24%
Dimensionality 40: C=0.01, gamma=1, test accuracy=11.35%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.01,gamma=10
Dimensionality 40: C=0.01, gamma=10, training accuracy=11.24%
Dimensionality 40: C=0.01, gamma=10, test accuracy=11.35%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.1,gamma=0.1
Dimensionality 40: C=0.1, gamma=0.1, training accuracy=97.22%
Dimensionality 40: C=0.1, gamma=0.1, test accuracy=96.16%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.1,gamma=1
Dimensionality 40: C=0.1, gamma=1, training accuracy=17.37%
Dimensionality 40: C=0.1, gamma=1, test accuracy=16.89%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=0.1,gamma=10
Dimensionality 40: C=0.1, gamma=10, training accuracy=11.24%
Dimensionality 40: C=0.1, gamma=10, test accuracy=11.35%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=1,gamma=0.1
Dimensionality 40: C=1, gamma=0.1, training accuracy=99.90%
Dimensionality 40: C=1, gamma=0.1, test accuracy=98.29%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=1,gamma=1
Dimensionality 40: C=1, gamma=1, training accuracy=100.00%
Dimensionality 40: C=1, gamma=1, test accuracy=28.02%
SVM classification process begin.Reduce the dimensionality from 784 to 40, C=1,gamma=10
Dimensionality 40: C=1, gamma=10, training accuracy=100.00%
Dimensionality 40: C=1, gamma=10, test accuracy=11.35%
```

Figure 12: Kernel SVM with different parameters

Part 4 Neural Networks

1. **Required model. A CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-10. The kernel size is set to be 5.**

   Training the model for 1 epoch, the accuracy on training and test set is as follows:



```
@ Total Time Spent: 24.38 seconds
Training Accuracy = 97.62 %      loss = 0.080057
Testing Accuracy = 97.50 %     loss = 0.078769
```

Figure 13: CNN_01 results after trained 1 epoch

   Training the model for 20 epochs, the accuracy on training and test set is as follows:

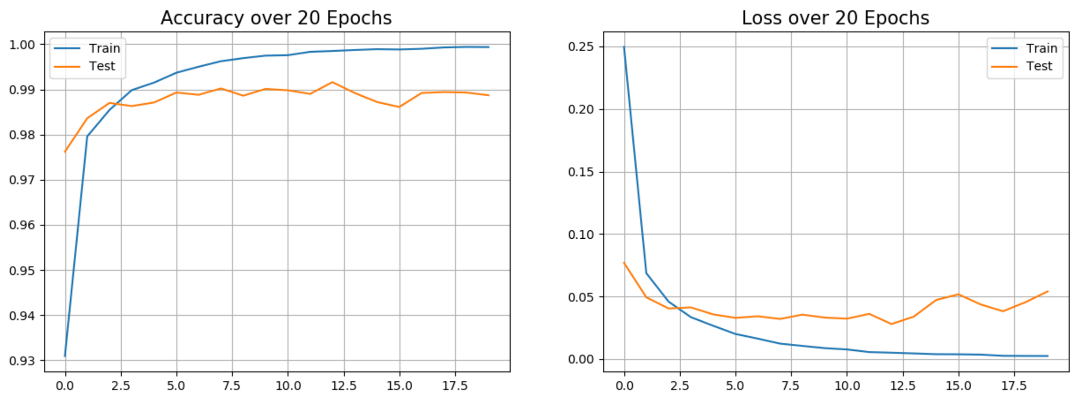
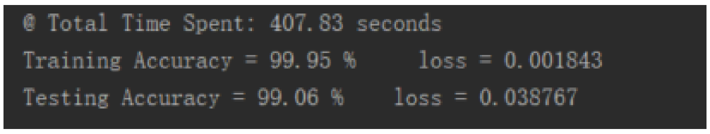


Figure 14: CNN_01 accuracy and loss changes line chart

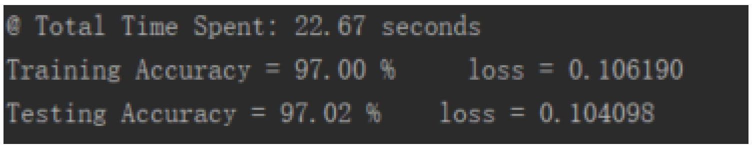


```
@ Total Time Spent: 407.83 seconds
Training Accuracy = 99.95 %      loss = 0.001843
Testing Accuracy = 99.06 %     loss = 0.038767
```

Figure 15: CNN_01 results after trained 20 epochs

2. **Required model. A CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-10. The kernel size is set to be 3.**

   Training the model for 1 epoch, the accuracy on training and test set is as follows:



```
@ Total Time Spent: 22.67 seconds
Training Accuracy = 97.00 %      loss = 0.106190
Testing Accuracy = 97.02 %     loss = 0.104098
```

Figure 16: CNN_02 results after trained 1 epochs

Training the model for 20 epochs, the accuracy on training and test set is as follows:

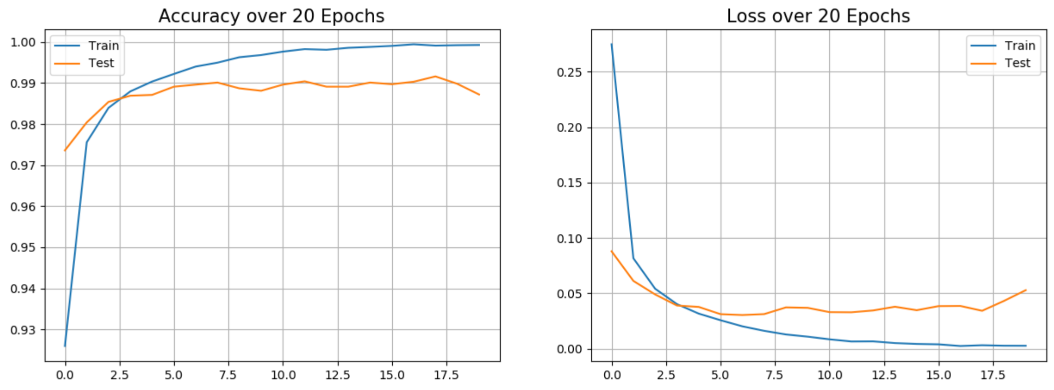
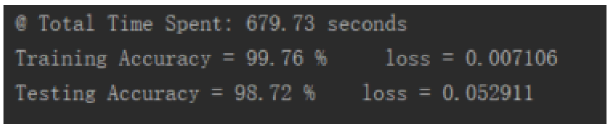


Figure 17: CNN_02 accuracy and loss changes line chart

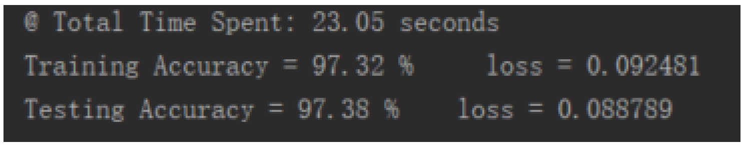


```
@ Total Time Spent: 679.73 seconds
Training Accuracy = 99.76 %      loss = 0.007106
Testing Accuracy = 98.72 %     loss = 0.052911
```

Figure 18: CNN_02 results after trained 20 epochs

3. **Required model, A CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-10. The kernel size is set to be 3. And after every max pooling layer, there is a ReLu activation function.**

   Training the model for 20 epochs, the accuracy on training and test set is as follows:



```
@ Total Time Spent: 23.05 seconds
Training Accuracy = 97.32 %      loss = 0.092481
Testing Accuracy = 97.38 %     loss = 0.088789
```

Figure 19: CNN_03 results after trained 1 epoch

Training the model for 20 epochs, the accuracy on training and test set is as follows:

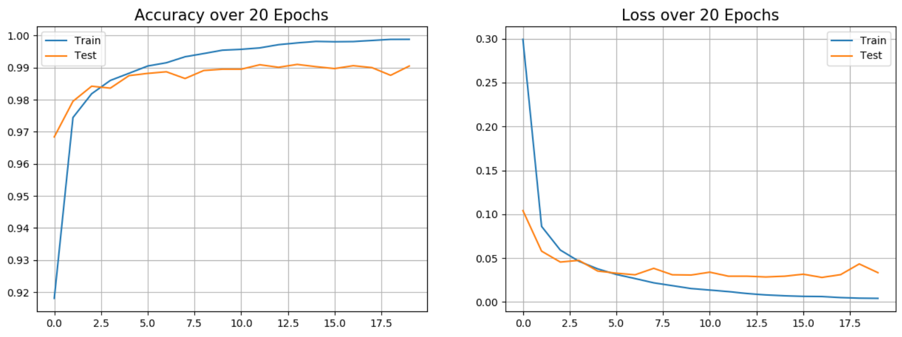
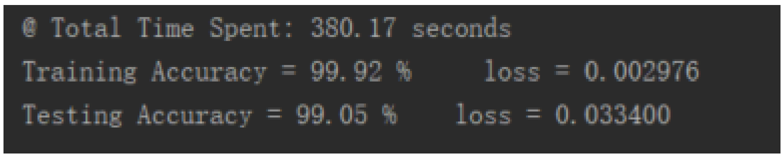


Figure 20: CNN_03 accuracy and loss changes line chart

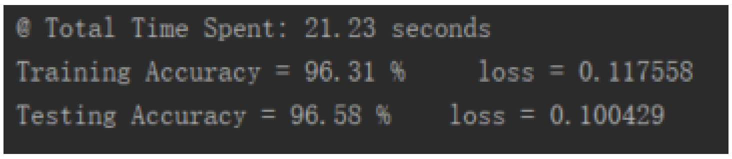


```
@ Total Time Spent: 380.17 seconds
Training Accuracy = 99.92 %      loss = 0.002976
Testing Accuracy = 99.05 %     loss = 0.033400
```

Figure 21: CNN_03 results after trained 20 epochs

4. **A CNN with three convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-50-500-10. The kernel size is set to be 3. And after every max pooling layer, there is a ReLu activation function.**

   Training the model for 1 epoch, the accuracy on training and test set is as follows:

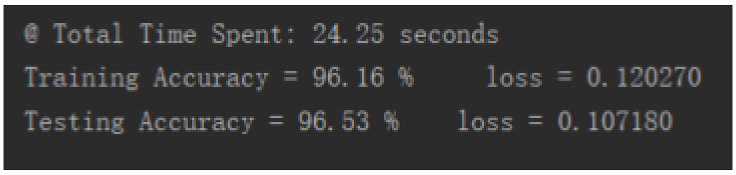


```
@ Total Time Spent: 21.23 seconds
Training Accuracy = 96.31 %      loss = 0.117558
Testing Accuracy = 96.58 %     loss = 0.100429
```

Figure 22: CNN_04 results after trained 1 epoch

5. **A CNN with three convolutional layers and two fully connected layers, with the architecture specified as follows: number of nodes: 20-50-50-500-1000-10. The kernel size is set to be 3. And after every max pooling layer, there is a ReLu activation function.**

   Training the model for 1 epoch, the accuracy on training and test set is as follows:



```
@ Total Time Spent: 24.25 seconds
Training Accuracy = 96.16 %      loss = 0.120270
Testing Accuracy = 96.53 %     loss = 0.107180
```

Figure 23: CNN_05 results after trained 1 epoch

From the above experiments results, we can see a very simple convolutional neural network can lead to a very high accuracy on the MNIST data set. And it is very clear that when train the model for 20 epochs, the model can fit the training set very well, the accuracy on training set is close to 1. But these results are overfitting. Generally, for CNNs, use L1, L2 and drop out regularization can prevent overfitting. And the deeper model usually leads to a higher accuracy. In addition, kernel size set to be 3 has a better performance in many cases than kernel size set to be 5. But because MNIST data set is simple, and for every image, it is 28*28*1, which is small. Some of the results do not follow these general laws. But they all has a very high accuracy.