CA2 Project Report

Student name: LIANG AOLING Matric number: A0177210N

1. PCA based data distribution visualization

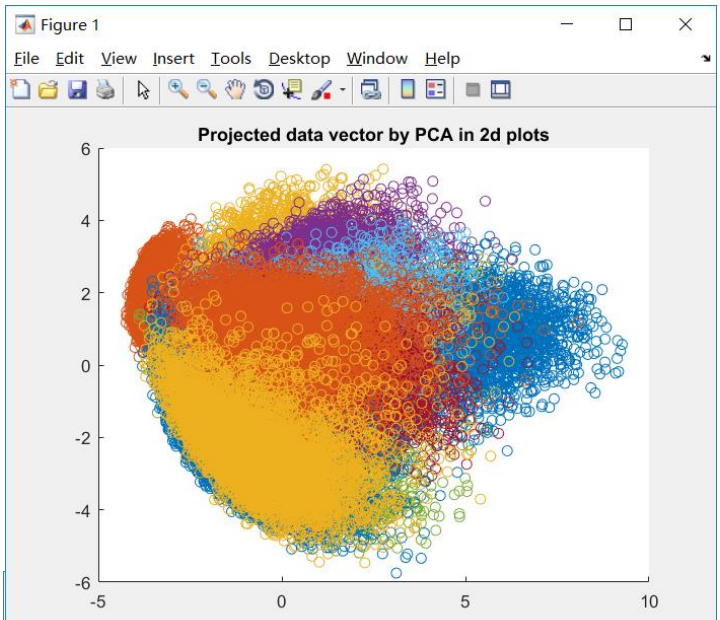


Figure 1

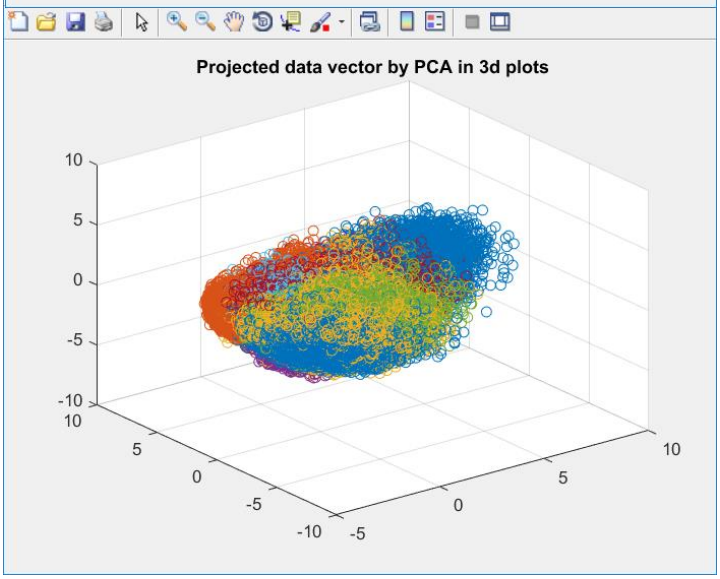


Figure 2

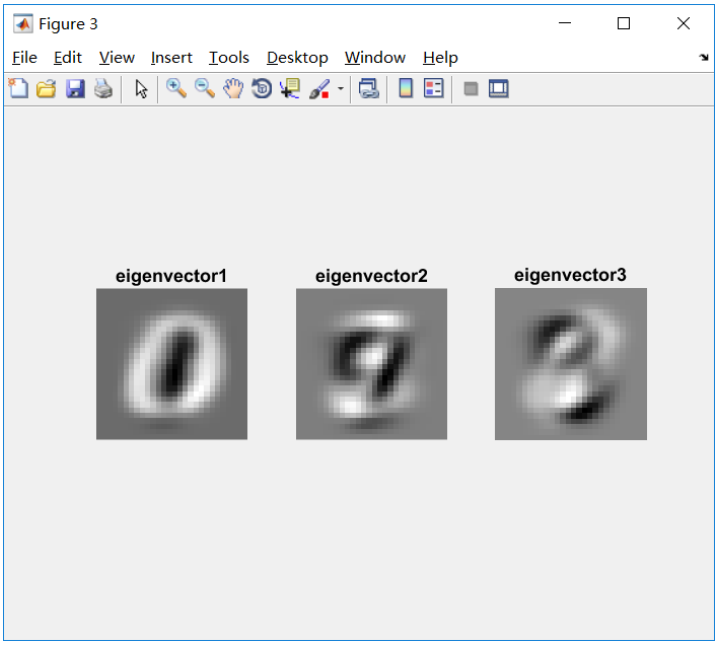


Figure 3

2. PCA plus nearest neighbor classification results

Dimension	40	80	200	154
Accuracy	97.35%	97.29%	96.91%	96.94%

Table 1

From the result table, we can see that after applying PCA to reduce the dimensionality of raw data from 784 to 40, the classification accuracy based on NN is 97.35%. After reducing to 80 dimension, the accuracy is 97.29% and after reducing to 200, the accuracy is 96.91%.

Using the below formula, the value of d preserving over 95% of the total energy after dimensionality reduction is 154. And after applying PCA to reduce data dimension to 154, the classification result based on NN is 96.94%.

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

Other criteria for automatically determining the value of d :

a. Using probabilistic PCA + Bayesian model selection to calculate and compare each posterior probability. This is called probabilistic principal component

analysis model.

b. Plotting a scee plot to show the number of eigenvalues ordered from biggest to smallest (Log-Eigenvalue Diagram). The correct number of components can be the number that appear prior to the elbow.

c. Selecting principal components by setting a threshold to the variance in the data

d. Using cross-validatory or computationally intensive methods

3. LDA based data distribution visualization

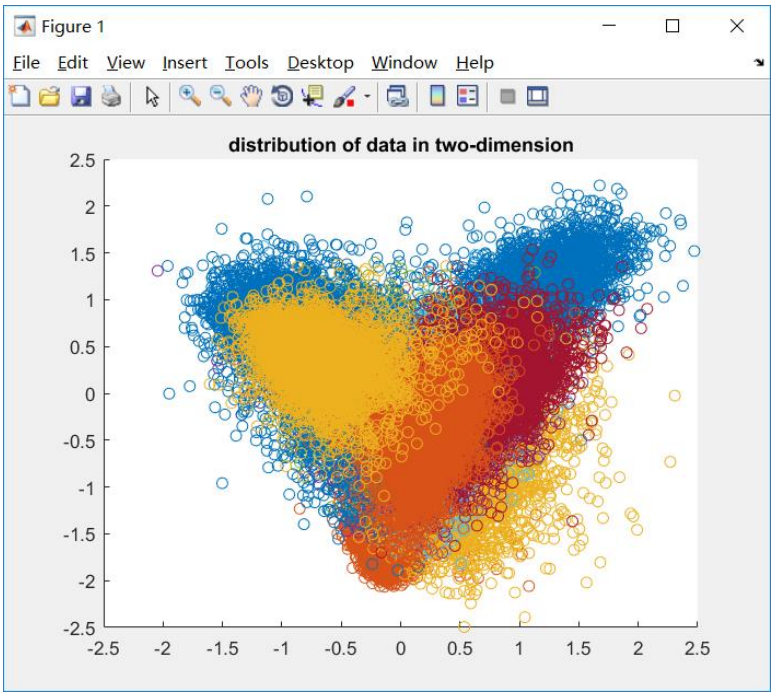


Figure 4

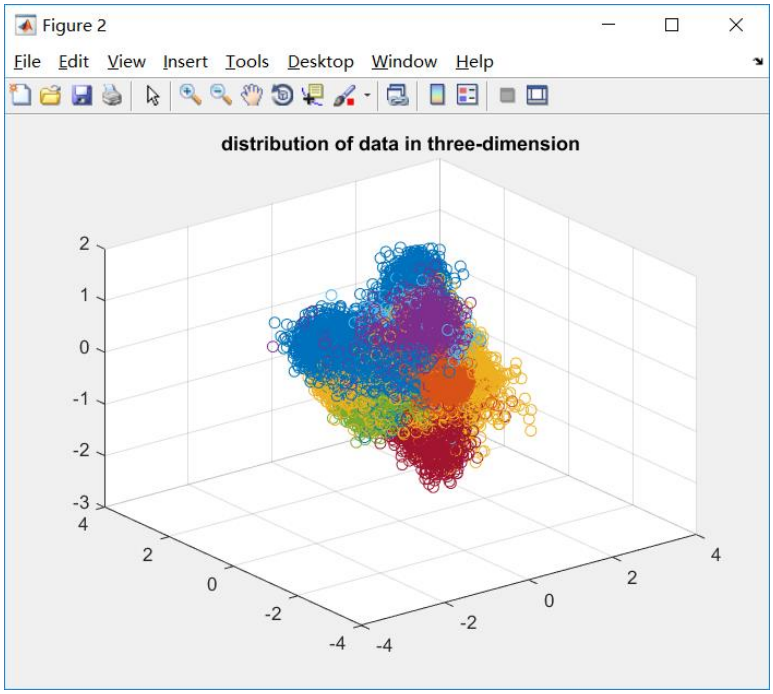


Figure 5

4. LDA plus nearest neighbor classification results:

Dimension	2	3	9	10
Accuracy	47.38%	67.09%	89.65%	90.14%

Table 2

From the table, we can view that after reducing the data to 2 dimension, the classification accuracy based on NN is 47.38%. After reducing to 3 dimension, the classification accuracy based on NN is 67.09%. After reducing to 9 dimension, the classification accuracy based on NN is 90.14%. The maximal dimensionality that data can be projected to via LDA is 10. Because the MINIST data only have 10 labels (0~9), it can be only classified to 10 classes and the maximal dimensionality is 10.

5. SVM classification results with different parameter values

A. Linear SVM result:

C	10^{-2}	10^{-1}	1	10
Raw data	94.38%	94.69%	93.98%	93.08%
PCA_40	93.25%	93.27%	93.21%	93.22%
PCA_80	94.00%	94.35%	94.31%	94.32%
PCA_200	94.31%	94.69%	94.24%	93.81%

Table 3

From this table, we can see that the linear SVM classification accuracies are all around 93%-94%, which are not very high and indicate the data is not separable for linear classification. The accuracy in all four kinds of C are very similar for both raw data and data which is reduced dimension to 200. Here the dimension has very low effect to the classification accuracy. But when the dimension is reduced to 80 or 40, the classification accuracy becomes lower.

When C is 10^{-2}, the slack variable can be ignored while when C is 10, the hard constrain will restrict the classification error. For all kinds of data, the accuracy in C = 10^{-1} is the highest, which means when the trade-off parameter C is 10^{-1}, for MNIST data set using SVM to classify, it will have relatively best classification result no matter whether it is reduced dimension or not.

For the raw data, the accuracy for C = 10 is the lowest. For the data of dimension 40, the accuracies are nearly the same for all C, which means the hard constraints have very small effect to the data whose dimension is reduced to 40. For data which is reduced dimension to 80, the accuracy in C = 10^{-2} is lower that other three Cs. When C = 10^{-1}, 1 and 10, the accuracies are very similar. For the data which is reduced dimension to 200 and the raw data, the accuracies in C = 10 are much lower than other three Cs. This phenomenon shows that when the data is not reduced dimension or only reduced to 200 dimension, the data distribution may be not-

separable, so when a larger hard constraint is added, the classification error will also be larger.

B. Kernel SVM：

I tune the g and c parameters for the data which has been reduced dimension to 40 to find the best accuracy. When c is 4, g is 19.6, the classification accuracy is 98.53%.

Here is some combinations of c and g parameters I tried:

C=4		
accuracy	96.76%	**g = 2^{-5}**
	98.24%	**g = 2^{-6}**
	97.94%	**g = 2^{-7}**
	97.08%	**g = 2^{-8}**

Table 4

I found that when c is fixed 4, and g = 2^{-5}, the result is best for 98.24%. When g becomes larger or smaller, the accuracy will reduce.

Then I made g fixed to 2^(-6) and tune c:

	c=1	c=2	c=4	c=8	c=10	c=16	c=18	c=19	c=19.2	c=19.3	c=19.4
accuracy (g = 2^(-6))	97.94%	98.19%	98.24%	98.46%	98.49%	98.49%	98.51%	98.52%	98.52%	98.53%	98.53%
	c=19.5	c=19.6	c=19.7	c=19.8	c=20	c=22	c=25	c=32	c=64	c=100	c=128
	98.53%	98.53%	98.51%	98.51%	98.51%	98.47%	98.45%	98.44%	98.46%	98.43%	98.42%

Table 5

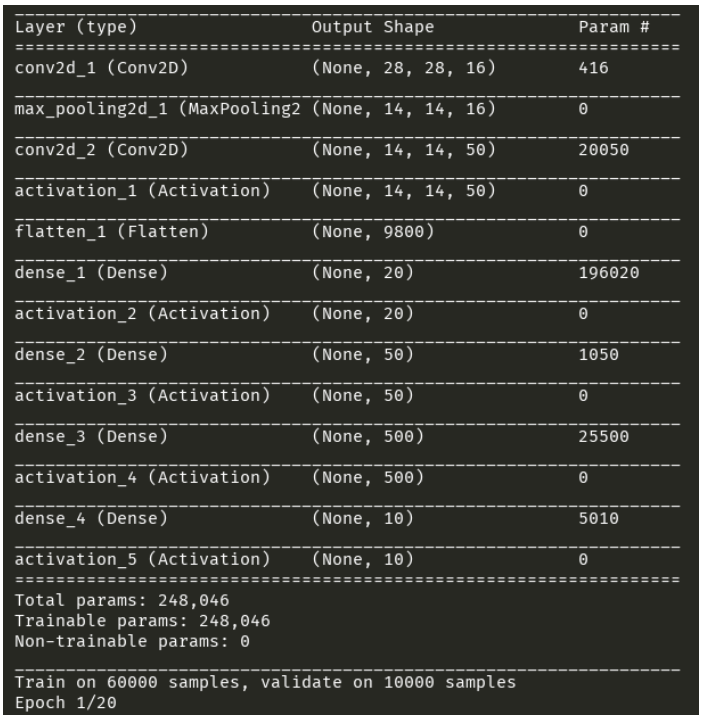
The table shows that when c = 19.3 – 19.6, the classification accuracy is the highest 98.53%. When c is below 19.3, the accuracy will reduce and when c is over 19.6, the accuracy will also reduce.

Compared with the above linear SVM's results, it can be found that the kernel function is much more suitable to classify the MNIST data set than the linear function. As the accuracy for kernel function is better than that for linear function. But the kernel function should have only very little parameters: C and G to perform best. For other C and G parameters' combination, the accuracies are also not very good.

6. Neural Networks

A. A CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-10. I use keras to train CNN and test. Firstly, I add only one max pooling after the first convolutional layer. Its classification accuracy is 98.87% and the loss is 5.4%. The classification error is 1.23%, which is very similar to *Reducing the dimensionality of data with neural networks 2006* in the web link.

Here is the requiring architecture and the training result:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 16)        416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 50)        20050
_________________________________________________________________
activation_1 (Activation)    (None, 14, 14, 50)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9800)              0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                196020
_________________________________________________________________
activation_2 (Activation)    (None, 20)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                1050
_________________________________________________________________
activation_3 (Activation)    (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 500)               25500
_________________________________________________________________
activation_4 (Activation)    (None, 500)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                5010
_________________________________________________________________
activation_5 (Activation)    (None, 10)                0
=================================================================
Total params: 248,046
Trainable params: 248,046
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
```

Figure 6

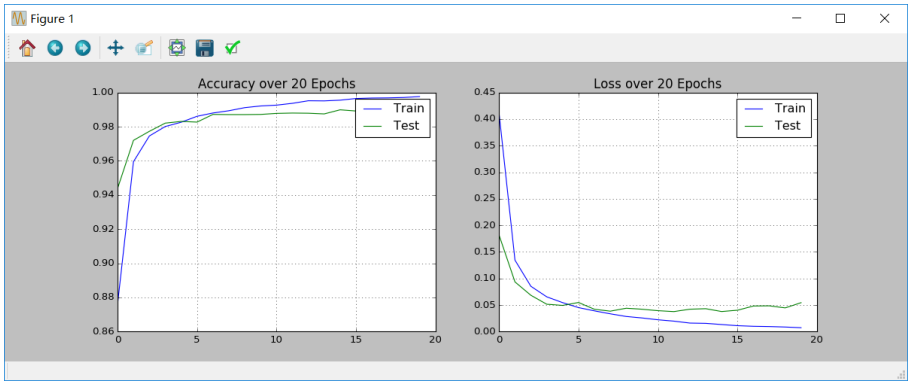


Figure 7

B. CNN classification results with different network architectures:

(1). When I changed the number of the fully connected layer, firstly I add one layer to the fully connected layer, the fully connected layer's architecture becomes 20-50-500-1000-10. Then the testing accuracy becomes 98.55% and the loss becomes 5.8%. The architecture is shown as below (the dense_4 and activation_5 are what I added).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 16)        416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 50)        20050
_________________________________________________________________
activation_1 (Activation)    (None, 14, 14, 50)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9800)              0
_________________________________________________________________
dense_1 (Dense)              (None, 20)                196020
_________________________________________________________________
activation_2 (Activation)    (None, 20)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                1050
_________________________________________________________________
activation_3 (Activation)    (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 500)               25500
_________________________________________________________________
activation_4 (Activation)    (None, 500)               0
_________________________________________________________________
dense_4 (Dense)              (None, 1000)              501000
_________________________________________________________________
activation_5 (Activation)    (None, 1000)              0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                10010
_________________________________________________________________
activation_6 (Activation)    (None, 10)                0
=================================================================
Total params: 754,046
Trainable params: 754,046
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
```

Figure 8

(2). I reduced the number of the fully connected layer to 20-50-10. The different

architecture is shown as below. The testing accuracy is 98.86% and the loss is 4.3%.

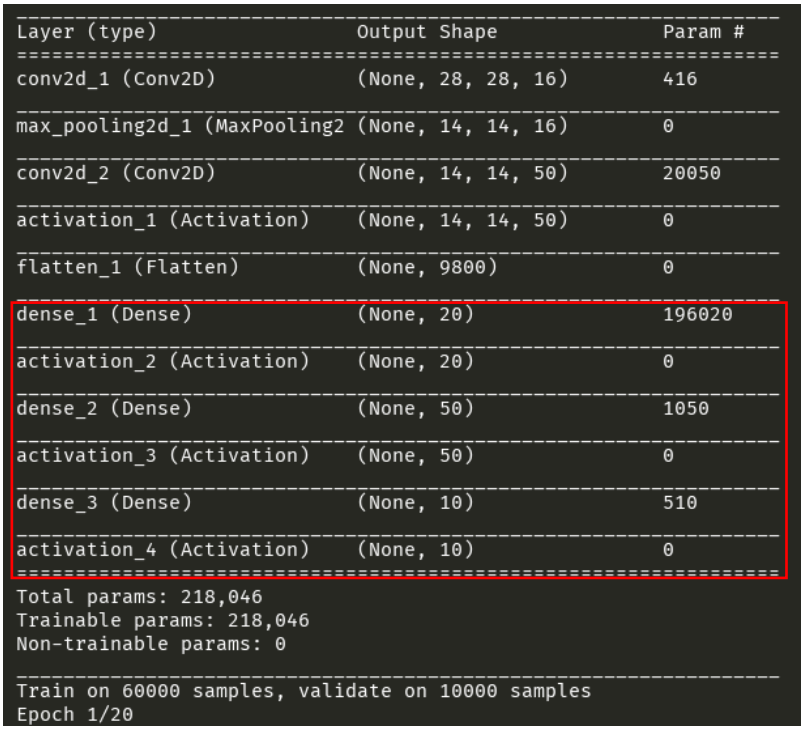


Figure 9

(3). I reduced the number of the fully connected layer to 20-10. The different architecture is shown as below. The testing accuracy is 98.72% and the loss is 4.7%.

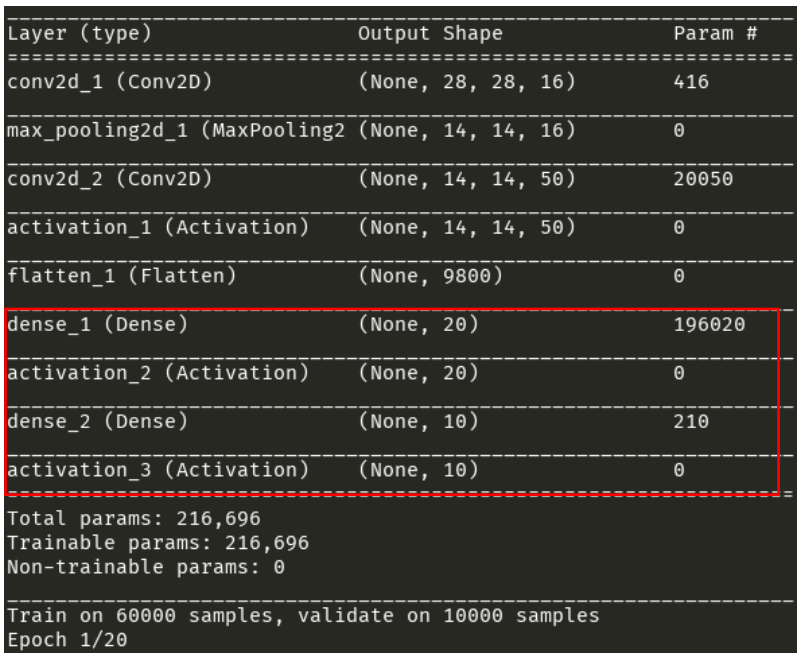


Figure 10

(4). I reduced the number of the fully connected layer to 10. The different architecture is shown as below. The testing accuracy is 98.89% and the loss is 4.17%.

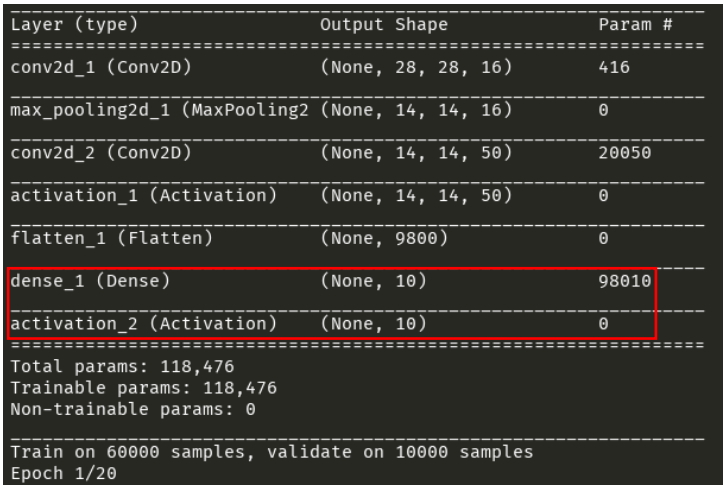


Figure 11

(5). I added one convolutional layer after the two default convolutional layers. The different architecture is shown as below. The testing accuracy is 98.92% and the loss is 5.44%.

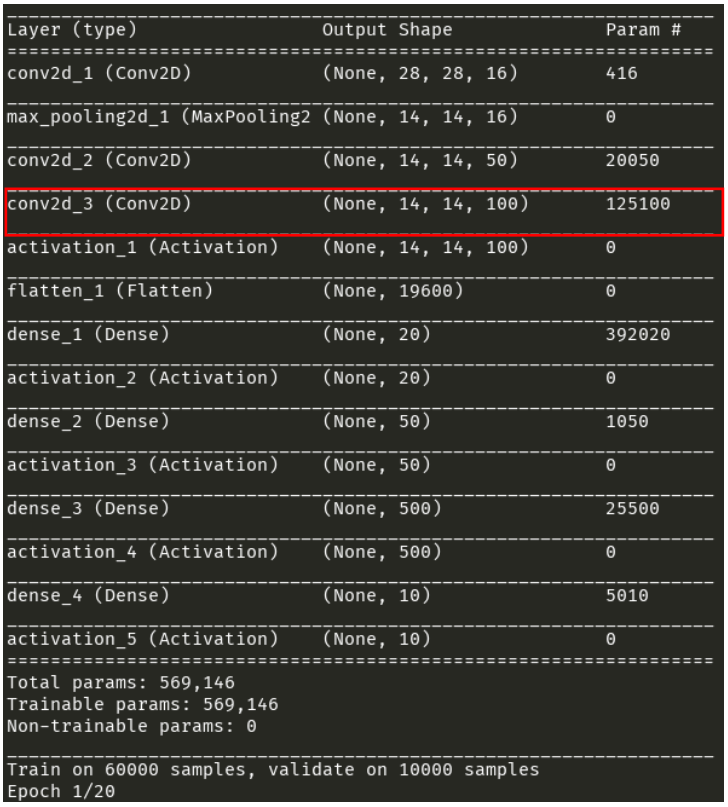


Figure 12

(6). I added one convolutional layer after the two default convolutional layers. The different architecture is shown as below. The testing accuracy is 98.93% and the loss is 5.5%.

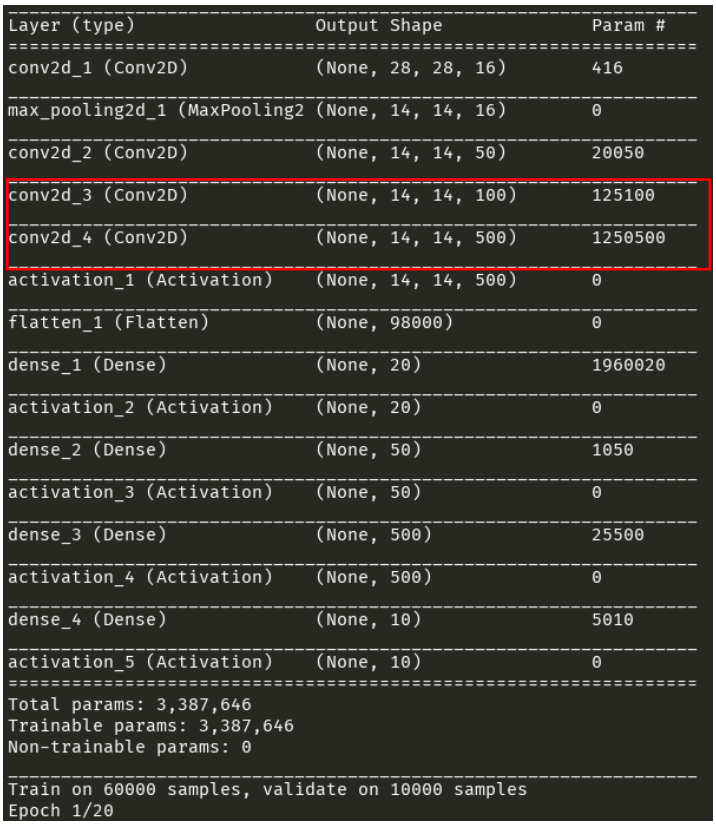


Figure 13

Also I changed more number of layers to view the testing results and I found that they have no big difference.

(7). Then I changed the types of layers. Firstly, I changed the kernel size of the first convolutional layer from 5*5 to 2*2. The different architecture is shown as below. The testing accuracy is 98.81% and the loss is 4.64%.

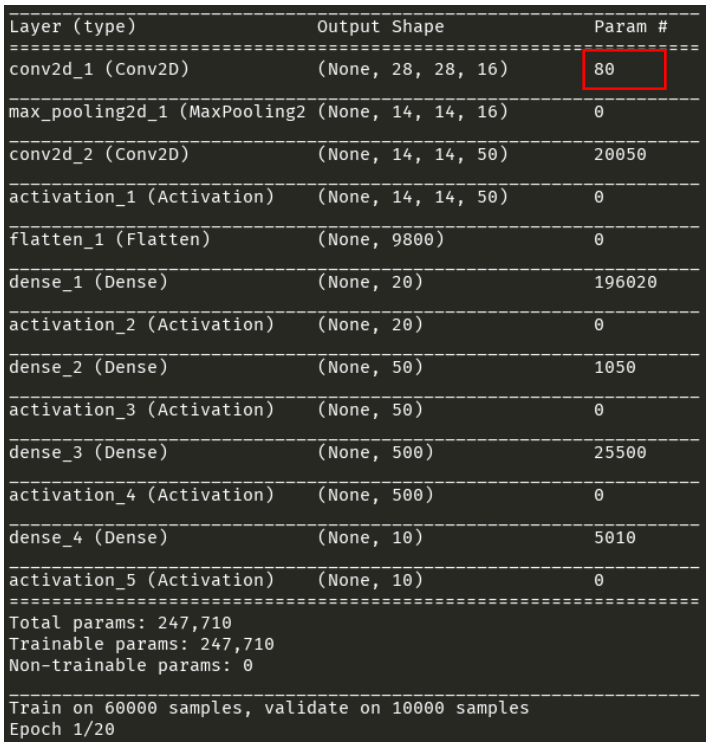


Figure 14

(8). I changed the kernel size of the first convolutional layer from 5*5 to 2*2. The different architecture is shown as below. The testing accuracy is 98.88% and the loss is 3.89%.

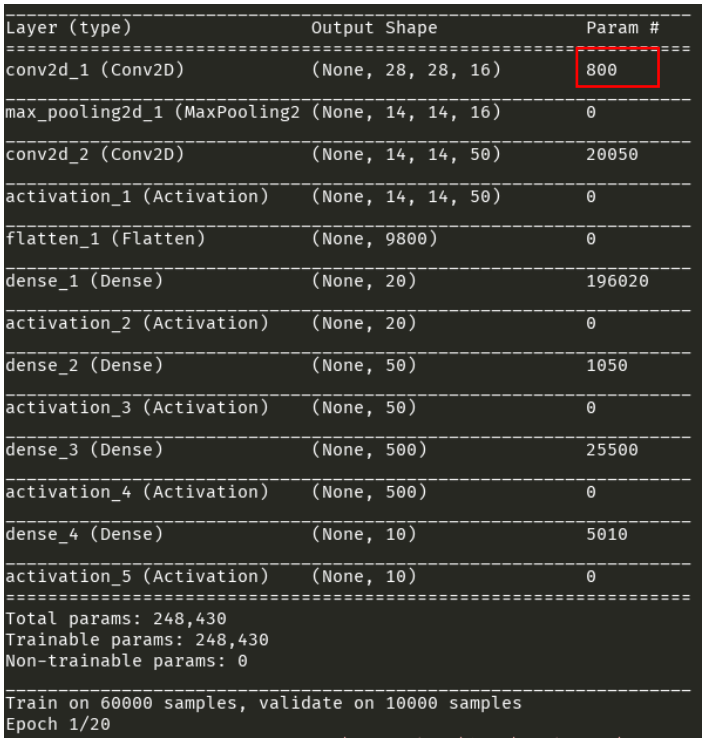


Figure 15

I also changed more kernel sizes and at last found the kernel size does not have a big

influence on the testing accuracy.

(9). I changed the filter size of the first convolutional layer from 16 to 10. The different architecture is shown as below. The testing accuracy is 98.99% and the loss is 4.14%.

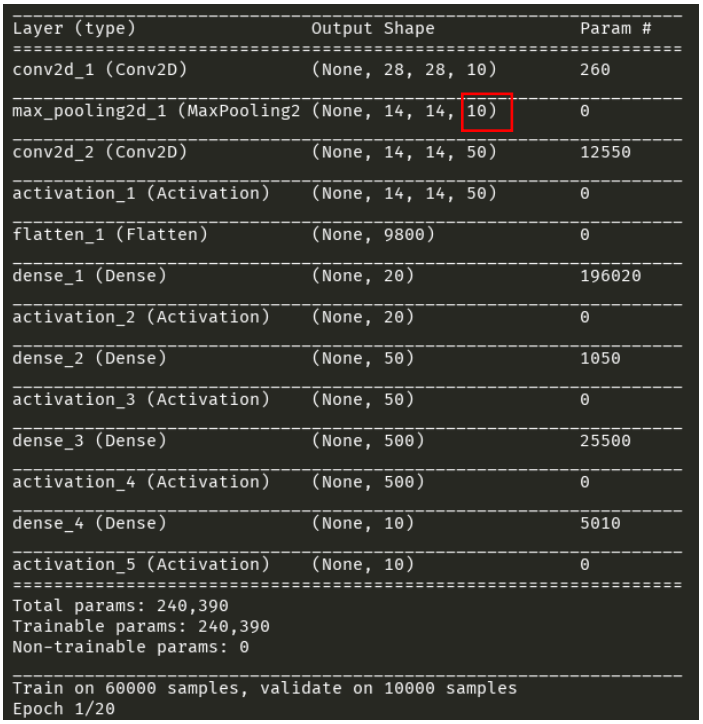


Figure 16

(10). I changed the filter size of the first convolutional layer from 16 to 30. The different architecture is shown as below. The testing accuracy is 98.69% and the loss is 5.2%.

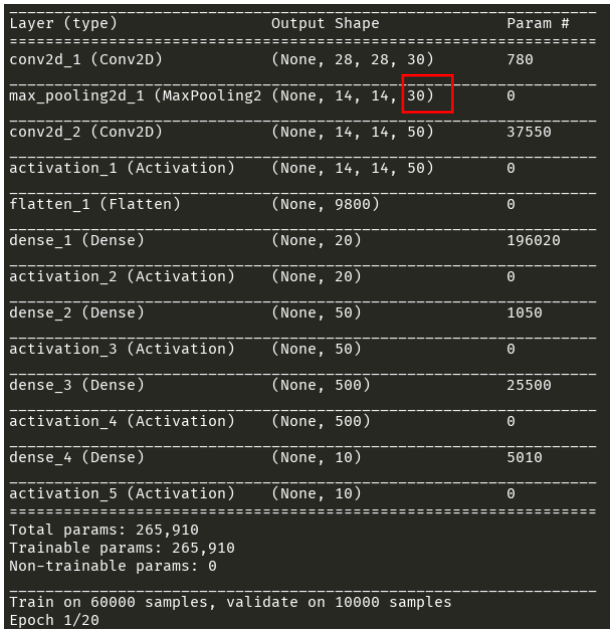


Figure 17

I also changed more filter sizes and at last found the kernel filter also does not have a big influence on the testing accuracy.

(11). I changed the pool size of the max pooling layer from 2 to 4. The different architecture is shown as below. The testing accuracy is 99.10% and the loss is 3.1%.

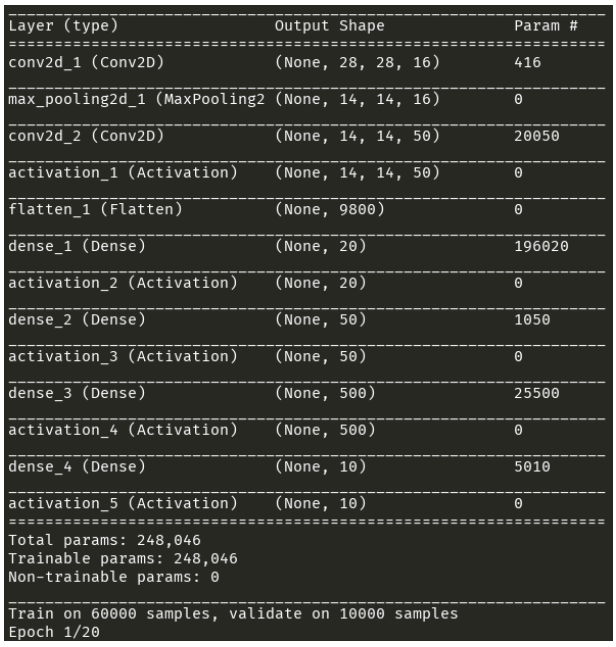


Figure 18

(12). I changed the pool size of the max pooling layer from 2 to 6. The different

architecture is shown as below. The testing accuracy is 99.27% and the loss is 2.5%.

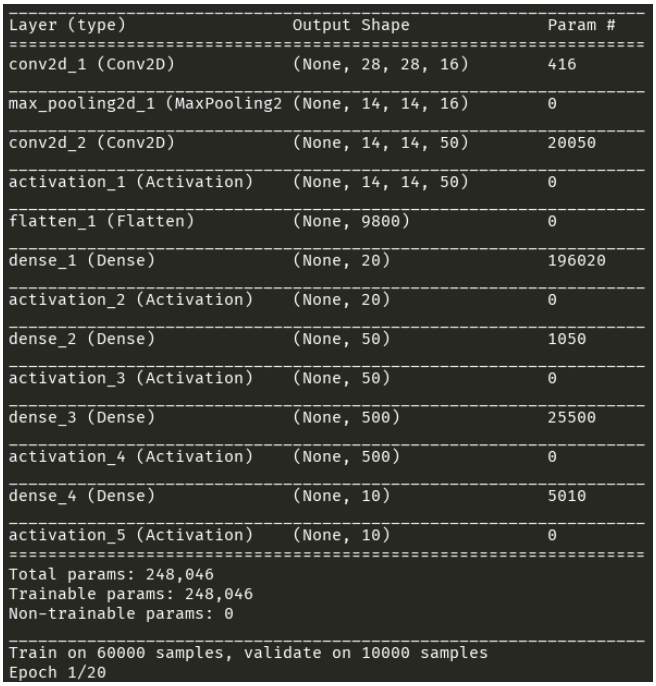


Figure 19

(13). I changed the pool size of the max pooling layer from 2 to 8. The different architecture is shown as below. The testing accuracy is 99.11% and the loss is 2.9%.

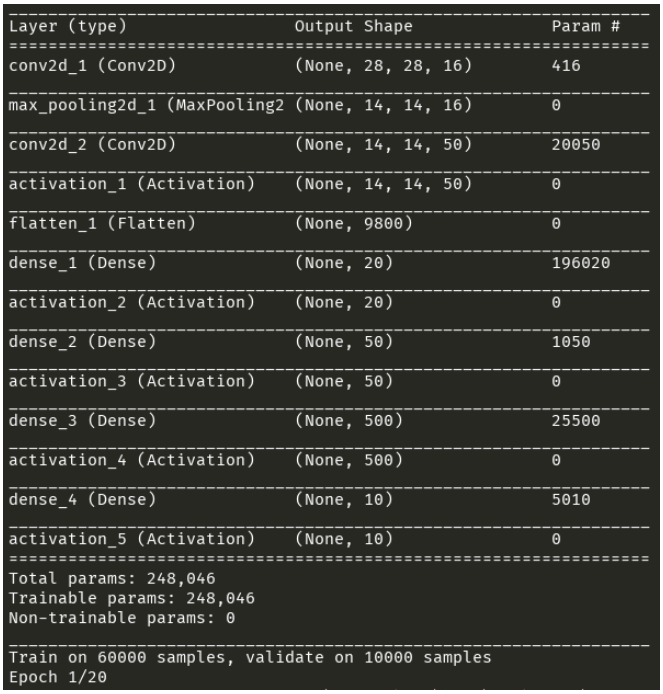


Figure 20

I found that when the max pooling size is larger than the previous 2, the testing accuracy can become better and over 99%. As the max pooling layer can better divide

the borders. The architecture will not change.

(14). Then I changed number of nodes per layer. Firstly, I changed fully connected layer's nodes from 20-50-500-10 to 10-50-500-10. The different architecture is shown as below. The testing accuracy is 98.88% and the loss is 4.29%.

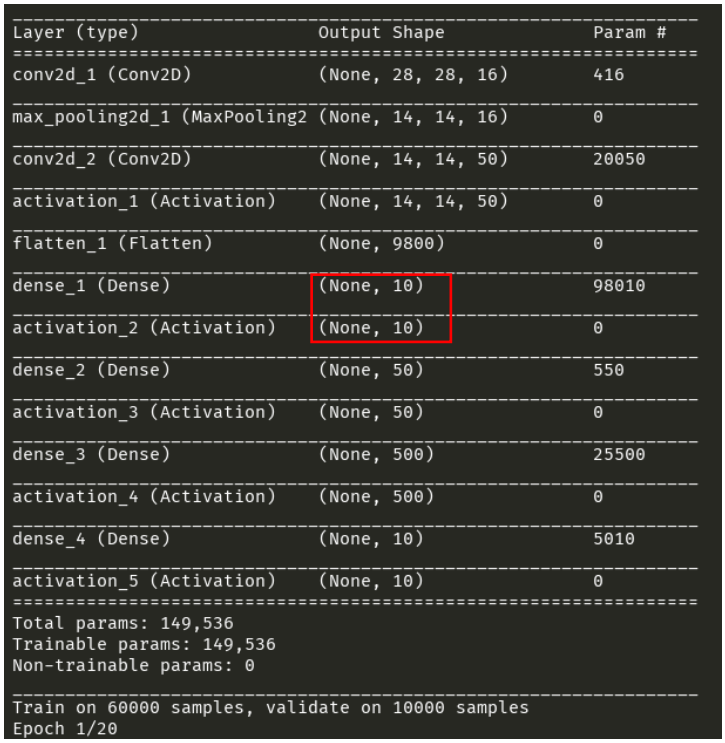


Figure 21

(15). I changed fully connected layer's nodes from 20-50-500-10 to 20-60-500-10. The different architecture is shown as below. The testing accuracy is 98.96% and the loss is 3.9%.

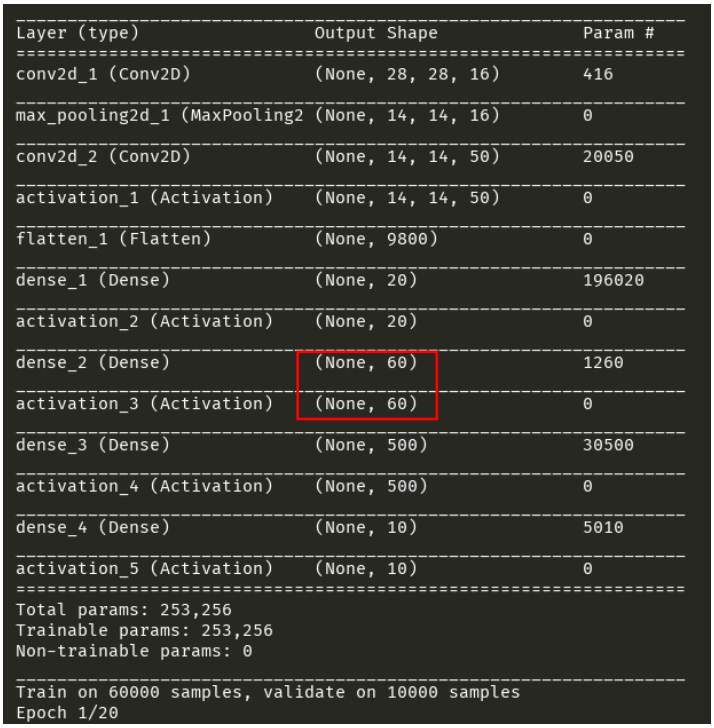


Figure 22

(16). I changed fully connected layer's nodes from 20-50-500-10 to 20-50-100-10 . The different architecture is shown as below. The testing accuracy is 98.85% and the loss is 4.45%.

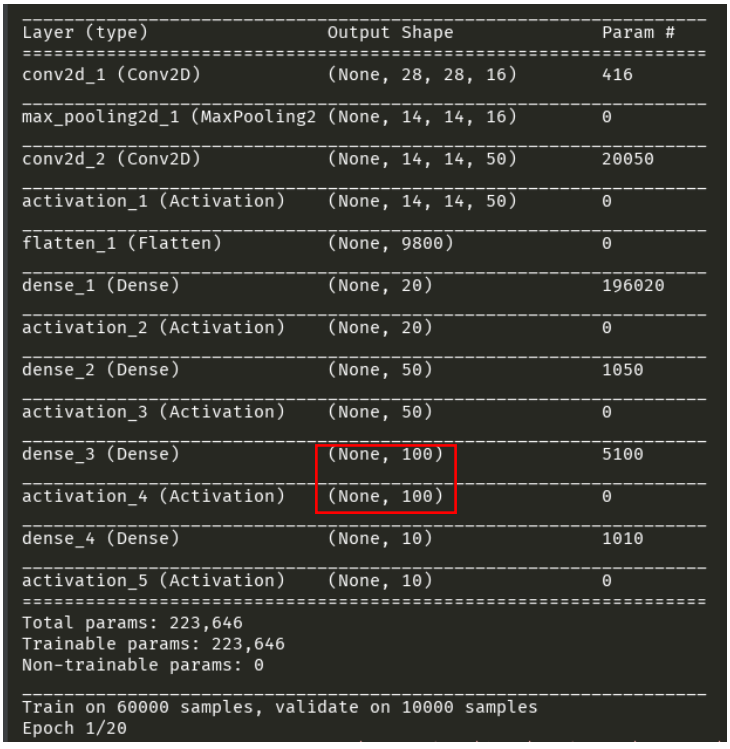


Figure 23

(16). I changed fully connected layer's nodes from 20-50-500-10 to 20-50-100-10 . The different architecture is shown as below. The testing accuracy is 99.00% and the loss is 4.07%.

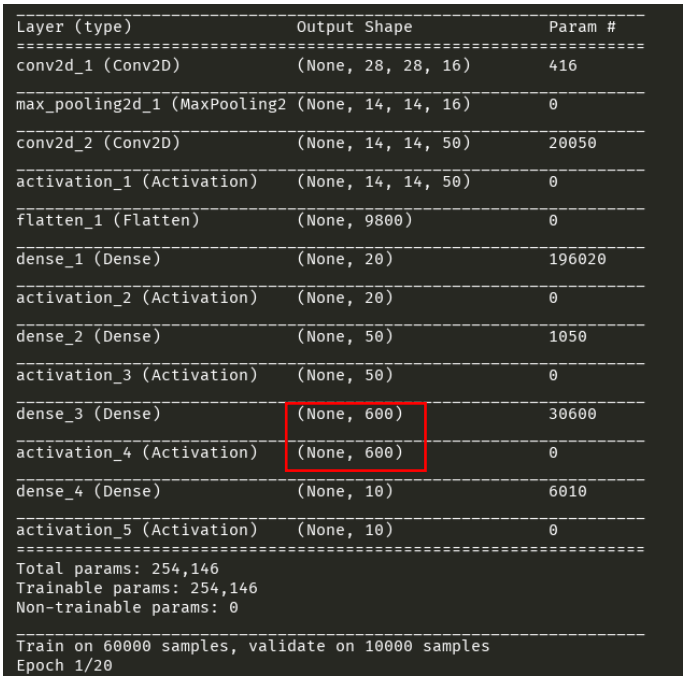


Figure 24

I also did more experiments to change the number of nodes in the fully connected layer and found that it will not affect the result too much. Above all, I think the type of layers especially the max pooling layer's pool size has the biggest influence on the neural networks' training and testing result. And because the MNIST data set has already been the simplest data set, which has only 0-1 grayscale so it can be classified to very high accuracies no matter how to change the architecture of the convolutional neuron network.