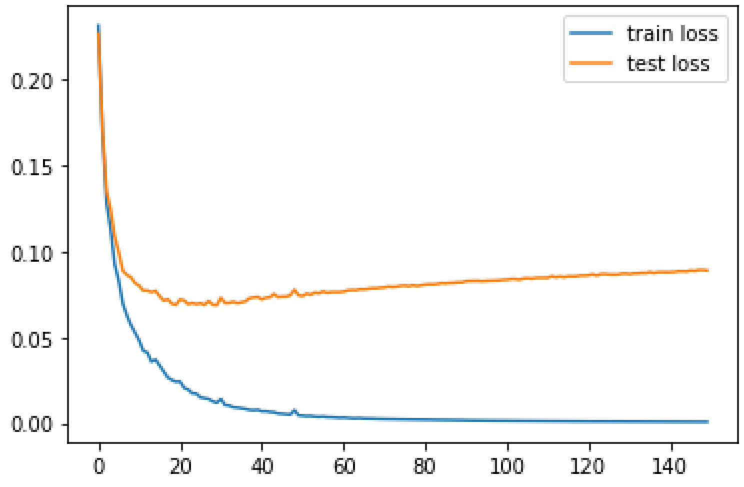
GR5241 Statistical Machine Learning Final Report

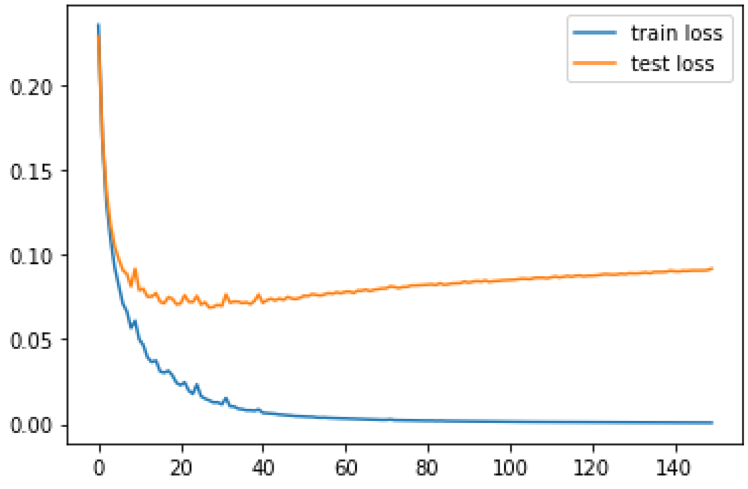
Shangjie Du, sd3497

Problem 3

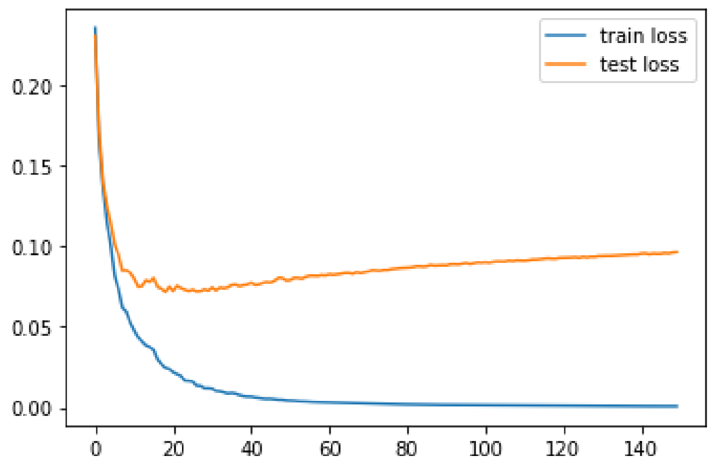
(a) In this part, we constructed a feedforward neural network with one hidden layer(100 hidden units) to predict handwritten digits' labels. The model is trained 5 times with different initialization. Weight parameters are initialized with Kaiming uniform initialization and bias parameters are initialized with 0. The loss function we used is cross-entropy loss. Here are the results of the training process.



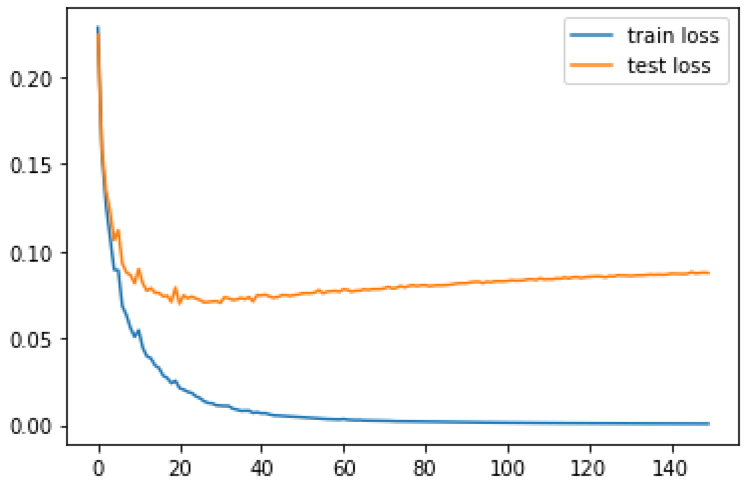
Run #1



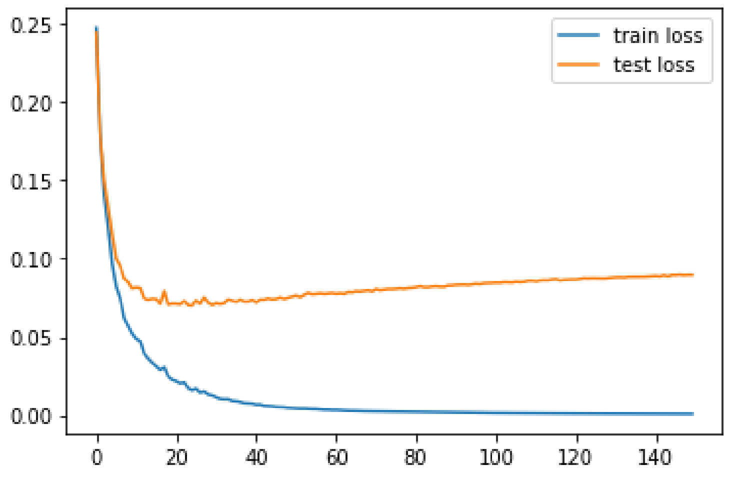
Run #2



Run #3



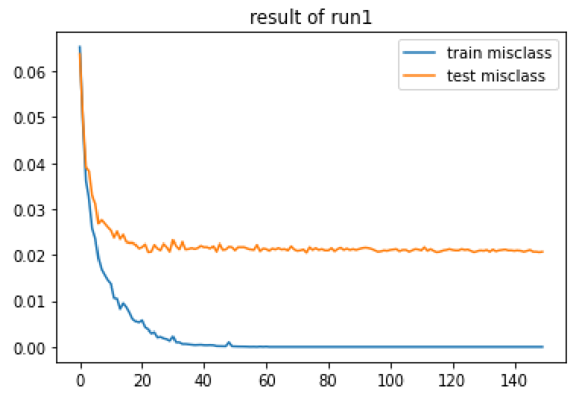
Run #4

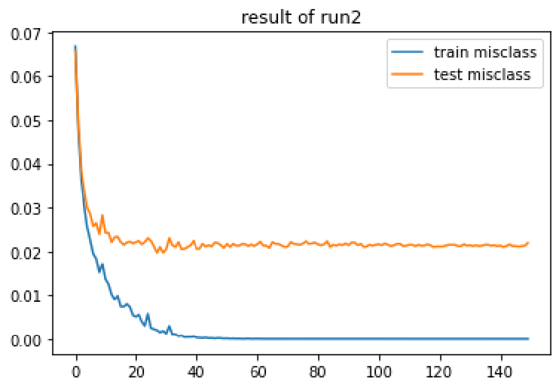


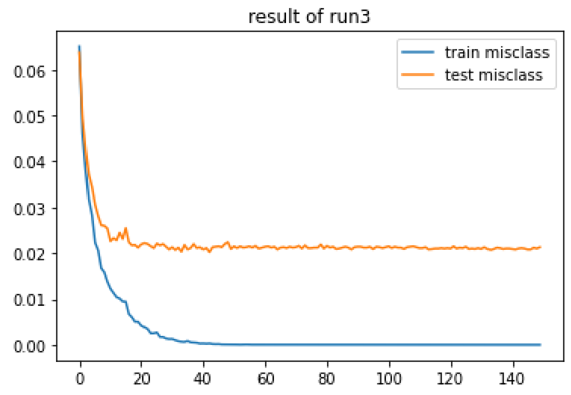
Run #5

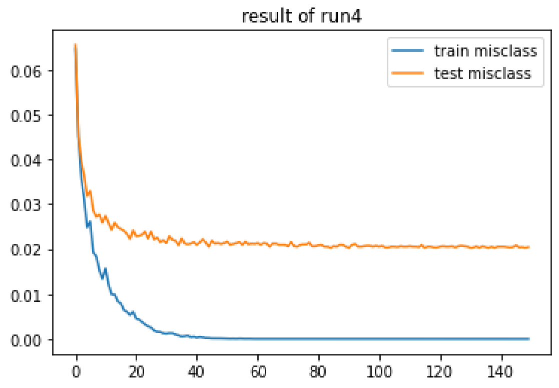
We can see that during the first few epochs, both train and test loss are decreasing. But after that, the training loss decreases while the test loss starts to increase. This is a sign of overfitting, which means the model can't be generalized to unseen samples.

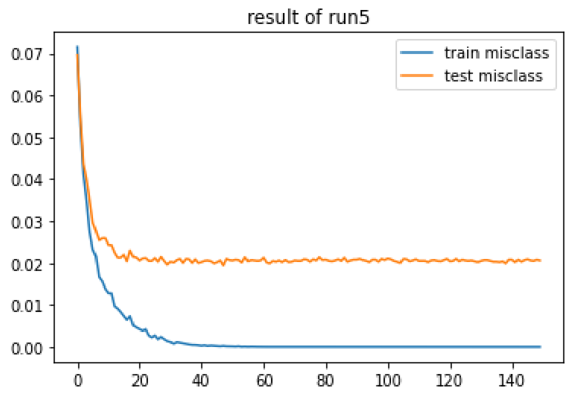
(b) If we change our loss measure to misclassification loss and record it during the training process, we'll get a slightly different result.









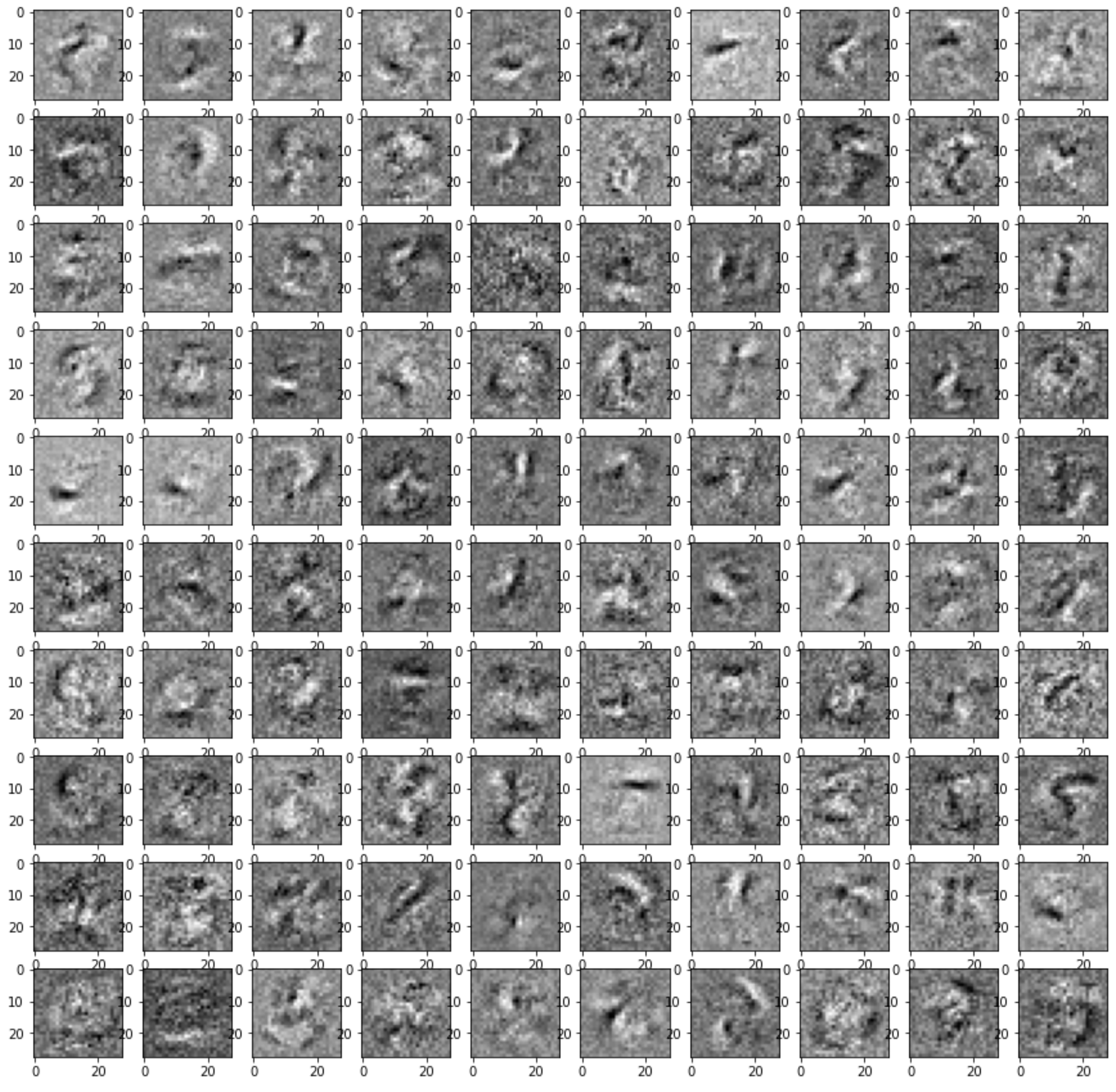


We can see that the training and testing misclassification error decreased in the first few epochs. The difference is that the test error doesn't increase after overfitting, instead, it fluctuates around the same level.

(c)The test accuracy is shown in the below table:

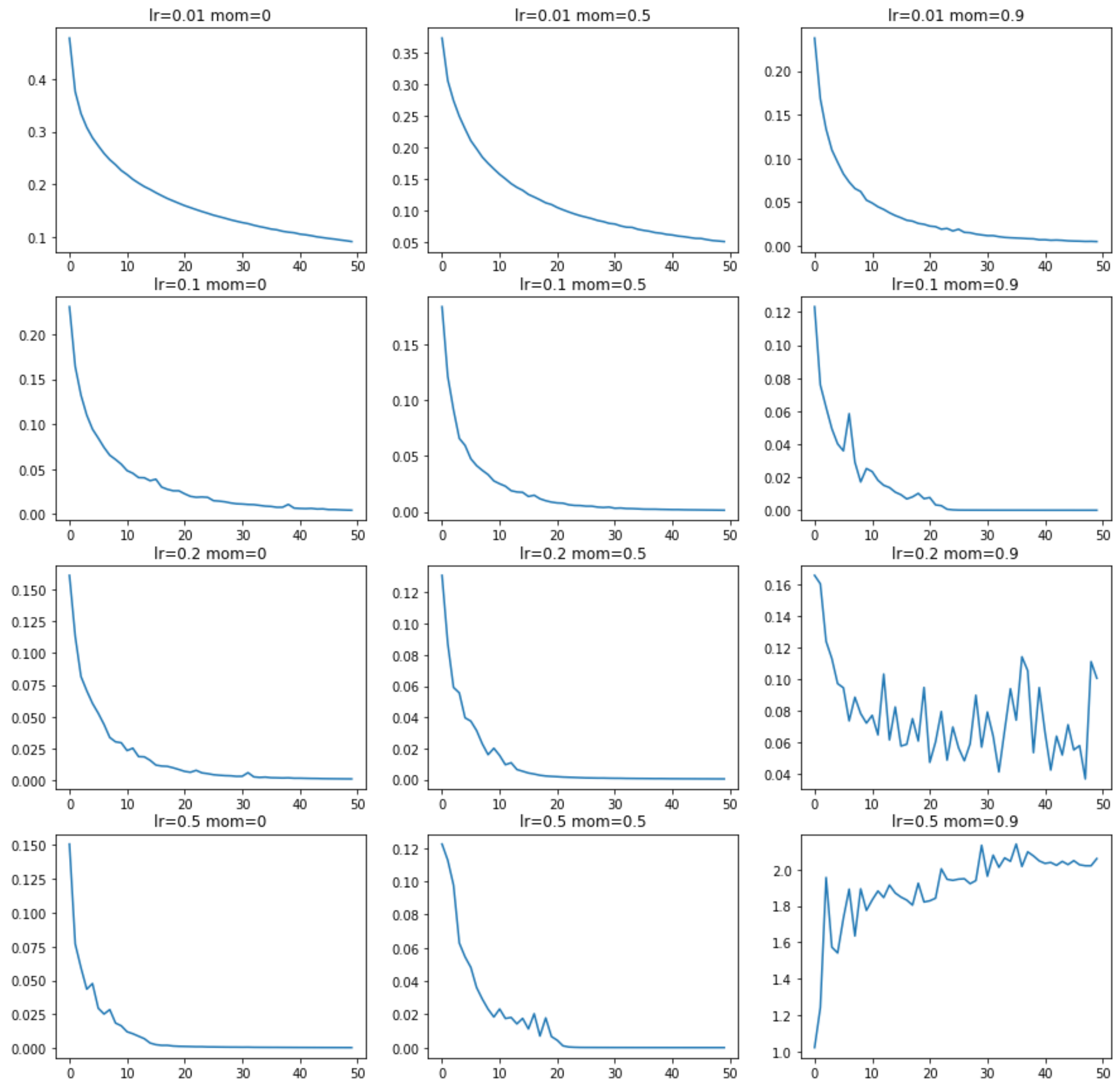
	Run #1	Run #2	Run #3	Run #4	Run #5
Test acc	97.93%	97.81%	97.87%	97.95%	97.94%

The best model is given by the 4th run. Now we can visualize the learned parameters as 100 graphs(28 x 28).



The learned parameters exhibit some structures, especially in the middle of the image. They are like some parts of a digit.

(d)We trained the model with SGD optimizers with a learning rate of 0.01, 0.1, 0.2, and 0.5, and momentum of 0, 0.5, and 0.9.



As shown in the above figure, we can make some general conclusions. The convergence is faster as the learning rate increases when momentum is fixed. The convergence is faster as momentum increases when the learning rate is fixed. But when the learning rate is 0.5 and momentum is 0.9, it failed to converge.

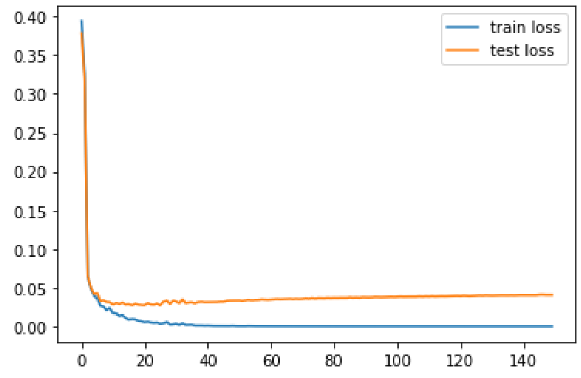
It seems that we can train the model for a few epochs with different choices of learning rate and momentum. If some pairs fail to converge we definitely won't choose them. And we can compare the training errors and choose the parameter pair with the lowest error.

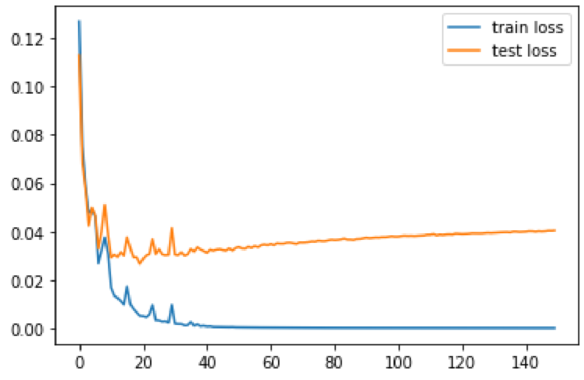
Problem 4

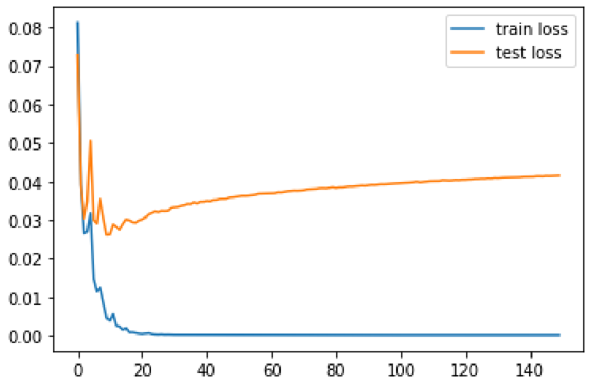
(a)In this problem, we constructed a CNN model with 2 convolutional layers and 2 fully connected layers for prediction. The structure of the network is shown below.

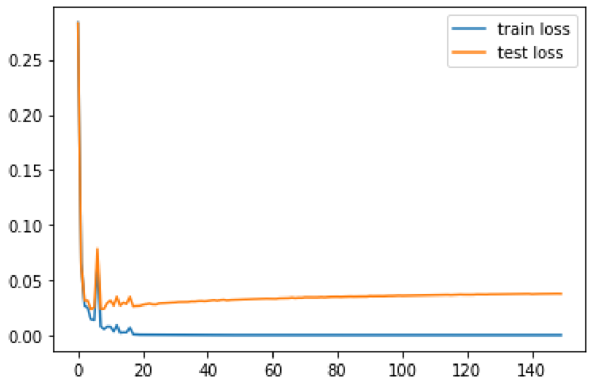
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 8, 28, 28]             208
         MaxPool2d-2            [-1, 8, 14, 14]               0
            Conv2d-3           [-1, 64, 12, 12]           4,672
         MaxPool2d-4             [-1, 64, 6, 6]               0
            Linear-5                  [-1, 256]         590,080
            Linear-6                   [-1, 10]           2,570
================================================================
Total params: 597,530
Trainable params: 597,530
Non-trainable params: 0
----------------------------------------------------------------
```

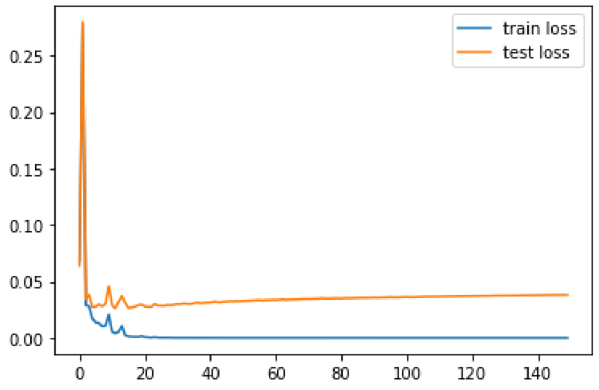
The model is trained 5 times with different initialization. Fully connected layers are initialized with Kaiming uniform initialization and convolution layers are initialized with Xavier uniform. All bias is initialized with 0. Here are the results of the training process.





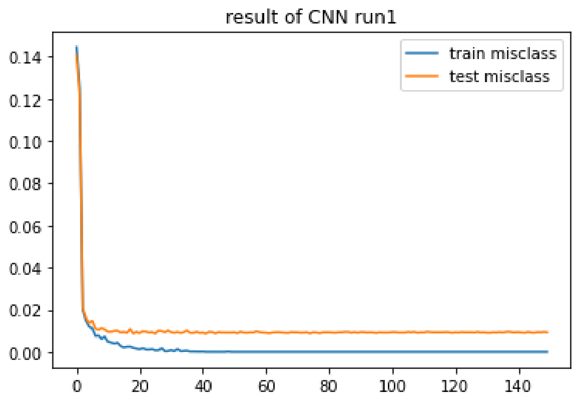


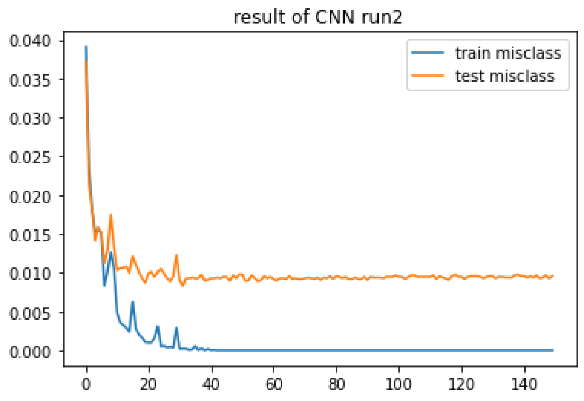


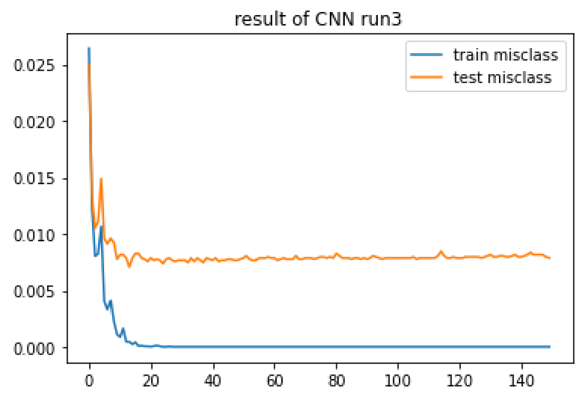


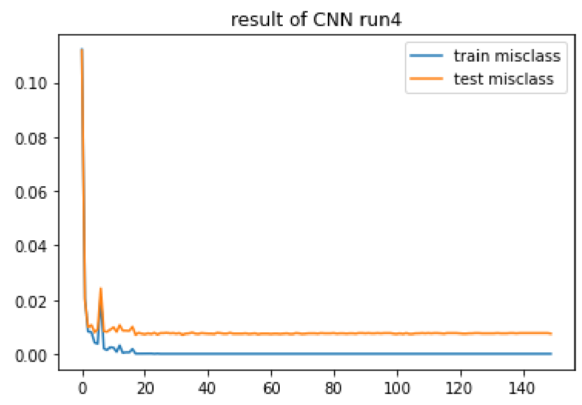
The overfitting issue occurs the same as that in problem3. Generally, the convergences are pretty fast, and for some runs, the cross-entropy loss is higher than the others.

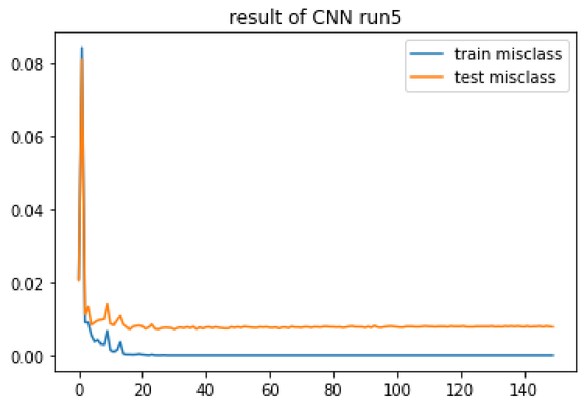
(b)The result of the misclassification error is basically the same: it decreased in the first few epochs and started to fluctuate at the same level.









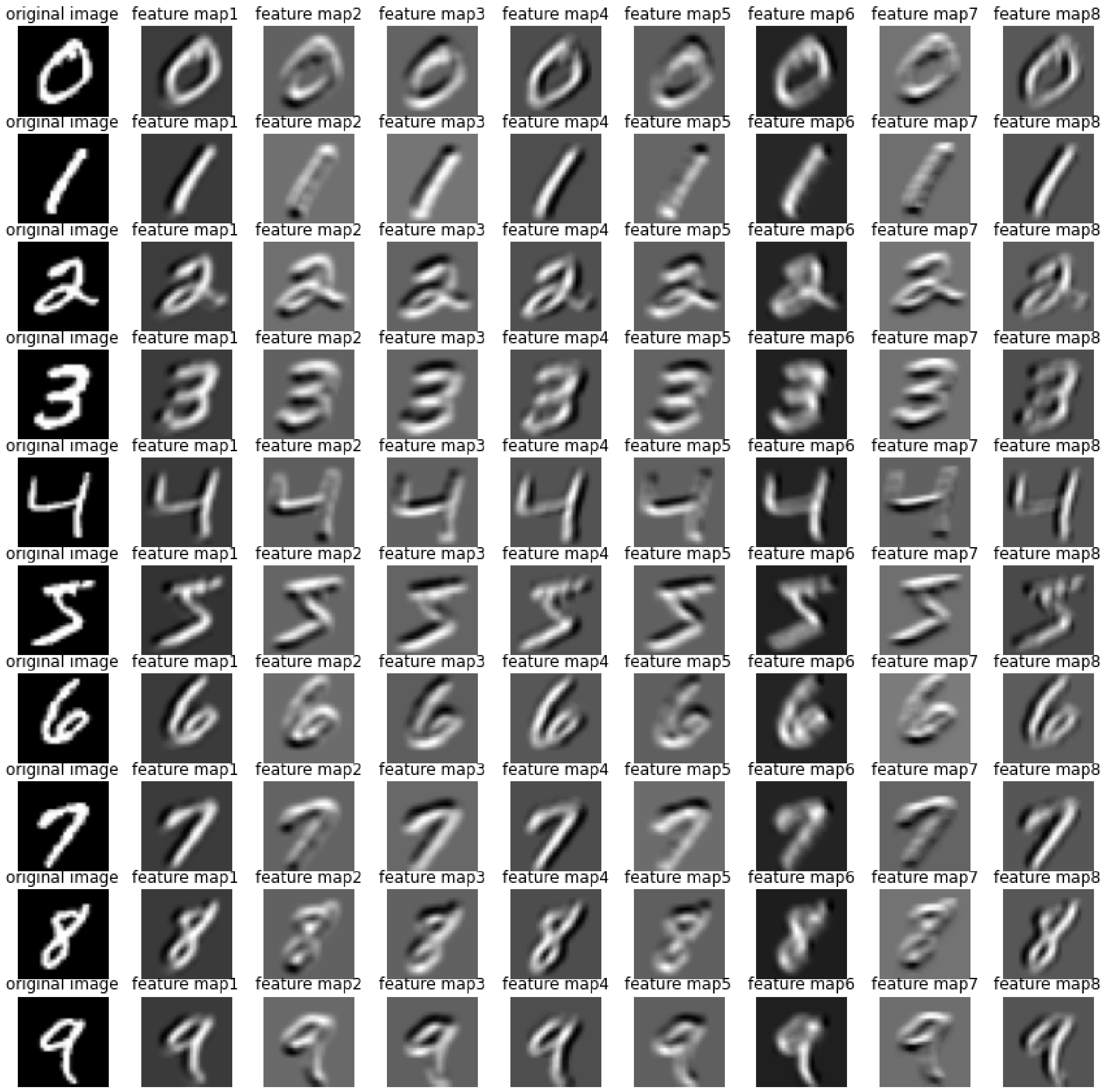


(c)The test accuracy of 5 CNN models is shown in the below table:

	Run #1	Run #2	Run #3	Run #4	Run #5
Test acc	99.07%	99.04%	99.21%	99.25%	99.21%

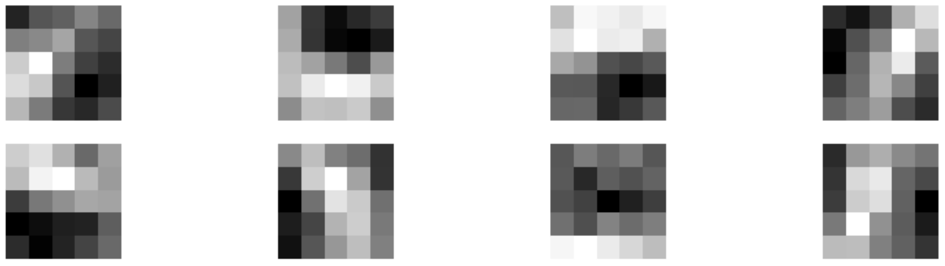
The best model is given by the 4th run. Now we can make some visualizations.

First, we can give some inputs to the model and take a look at the feature maps after the first convolution layer. The visualization result is shown below.



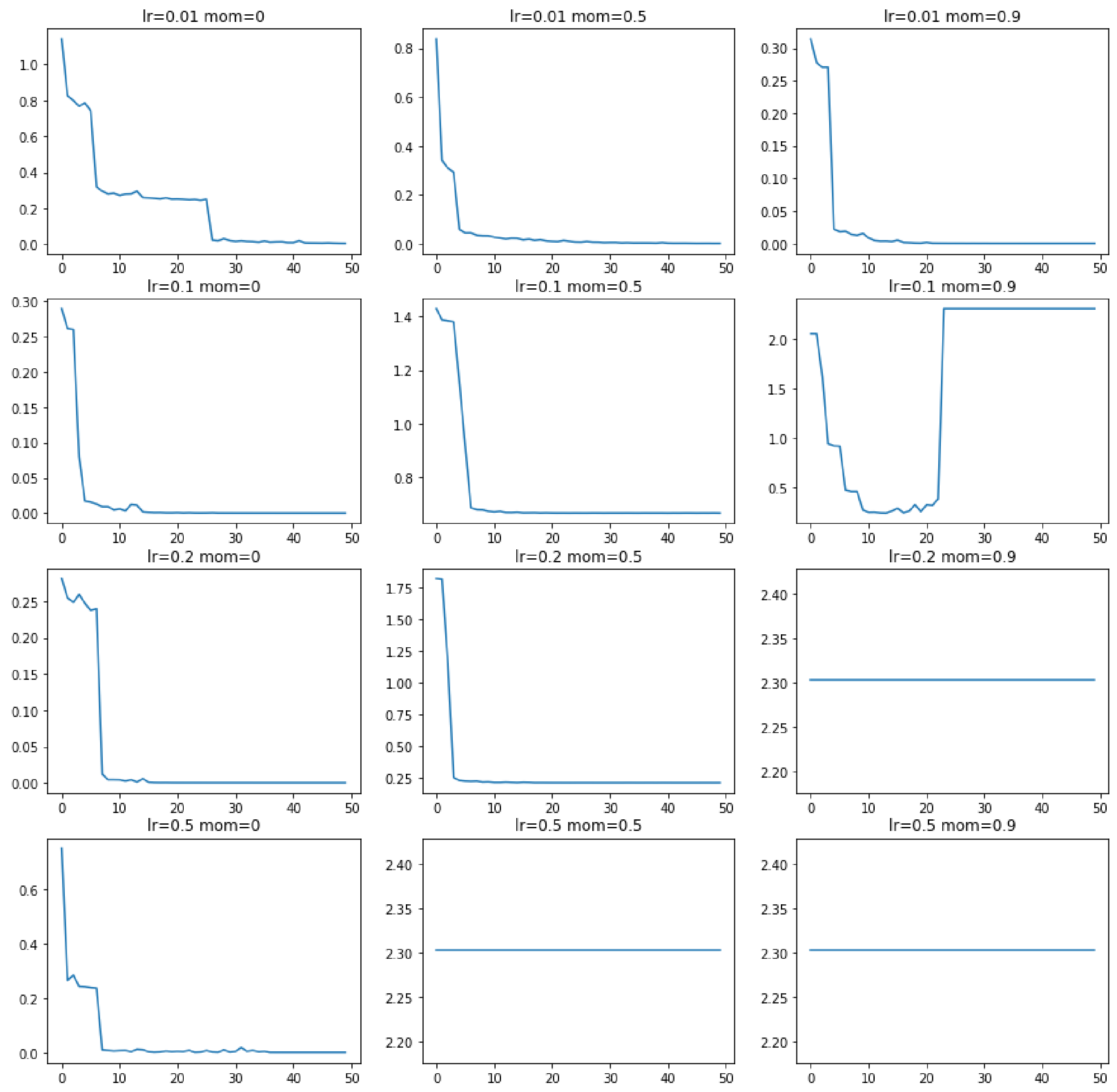
We can see that the feature maps after the first convolutional layer mainly preserve the digits' structure.

Then we can visualize the 8 kernels in the first convolutional layer.



The kernel size is 5x5. The visualization is hard to explain.

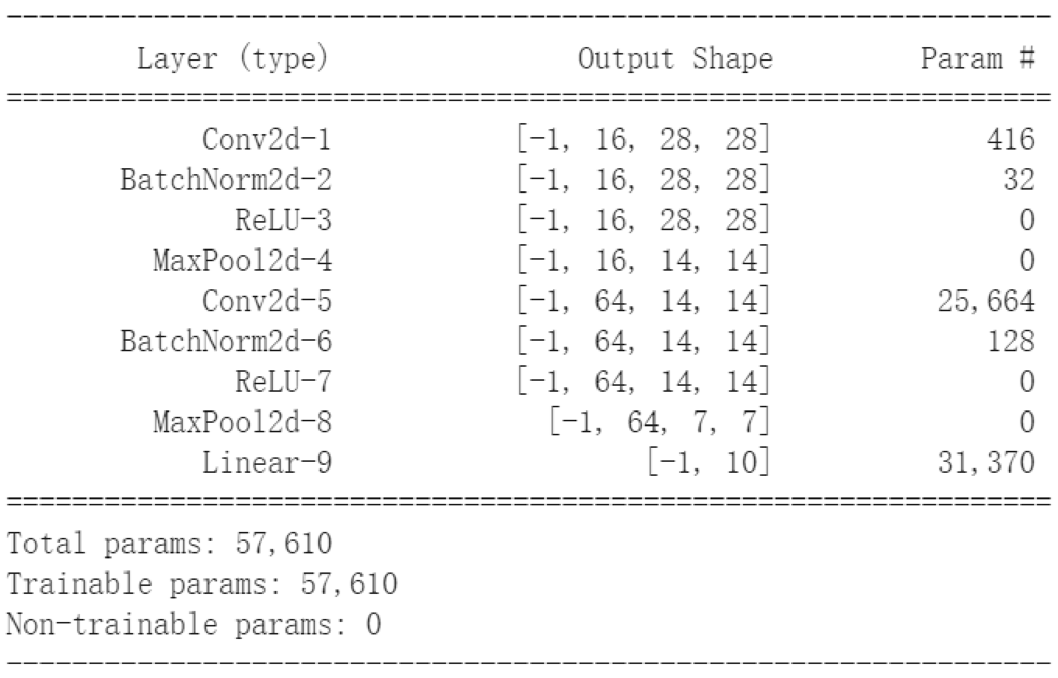
(d)We trained the model with SGD optimizers with a learning rate of 0.01, 0.1, 0.2, and 0.5, and momentum of 0, 0.5, and 0.9.



We can see that, when momentum is 0, it converges for all choices of the learning rate. For momentum is 0.5 and 0.9, it fails to converge when the learning rate is high. If we compare the results of momentum 0 and 0.5, we can see that the momentum method sometimes helps us avoid getting stuck at the local minimum.

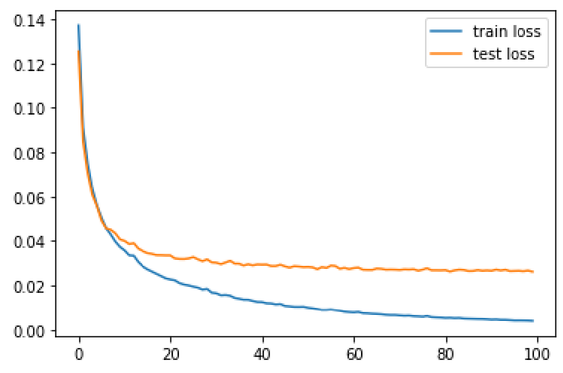
Problem 5

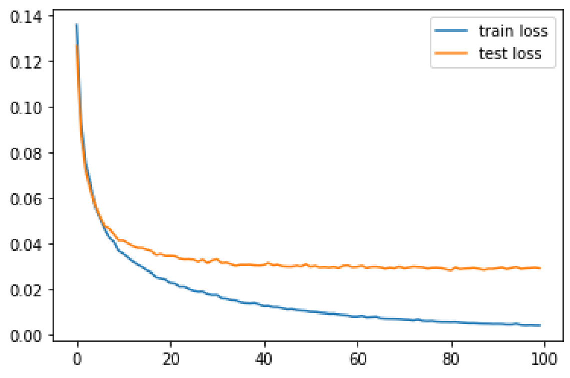
(a) In the problem, we introduce a 2D batch normalization layer to our convolutional neural network. The batch normalization layer standardizes every mini-batch of samples when they are passed through the network. The model structure is shown below:

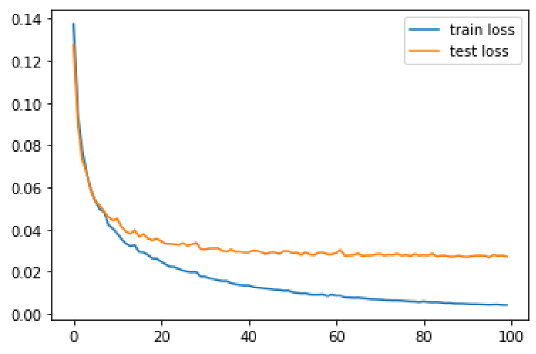


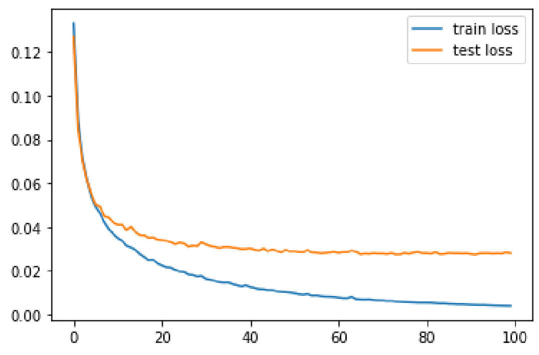
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 28, 28]             416
       BatchNorm2d-2           [-1, 16, 28, 28]              32
              ReLU-3           [-1, 16, 28, 28]               0
         MaxPool2d-4           [-1, 16, 14, 14]               0
            Conv2d-5           [-1, 64, 14, 14]          25,664
       BatchNorm2d-6           [-1, 64, 14, 14]             128
              ReLU-7           [-1, 64, 14, 14]               0
         MaxPool2d-8             [-1, 64, 7, 7]               0
            Linear-9                   [-1, 10]          31,370
================================================================
Total params: 57,610
Trainable params: 57,610
Non-trainable params: 0
----------------------------------------------------------------
```

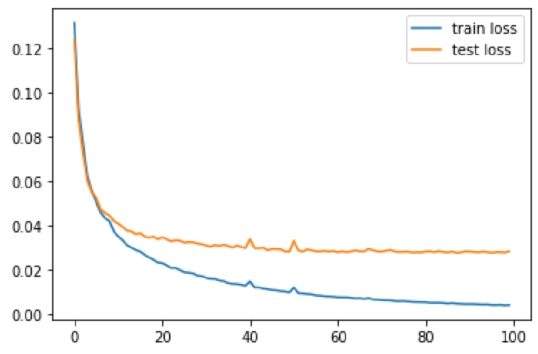
The model is trained 5 times with different initialization. Fully connected layers are initialized with Kaiming uniform initialization and convolution layers are initialized with Xavier uniform. All bias is initialized with 0. Here are the results of the training process.



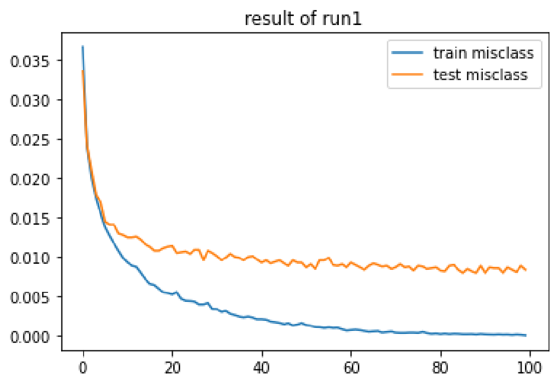


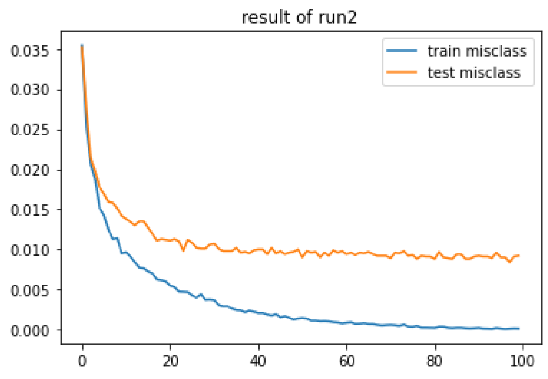


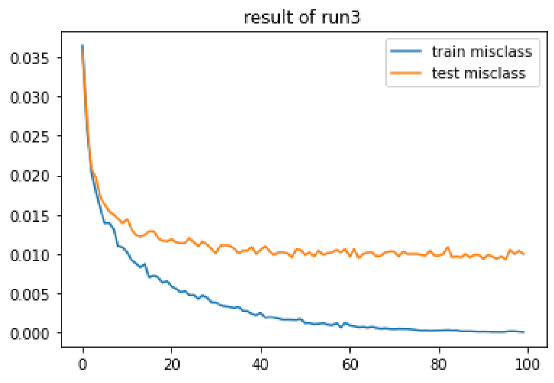


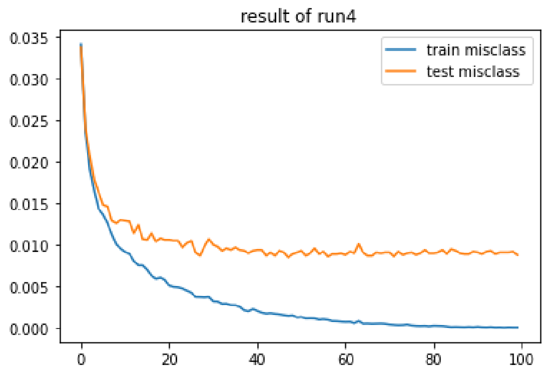


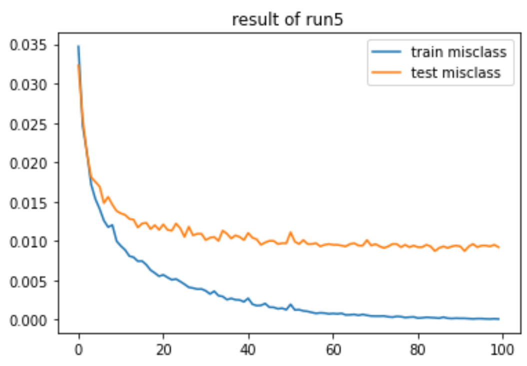
(b)The result of the misclassification error is basically the same: it decreased in the first few epochs and started to fluctuate at the same level.







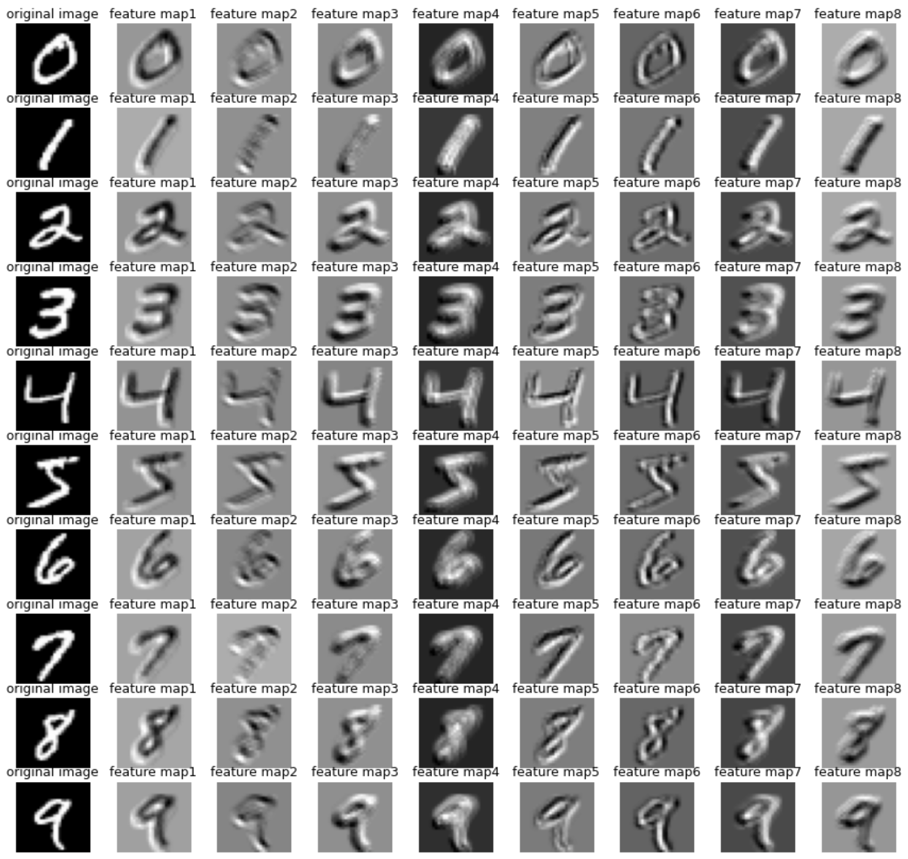




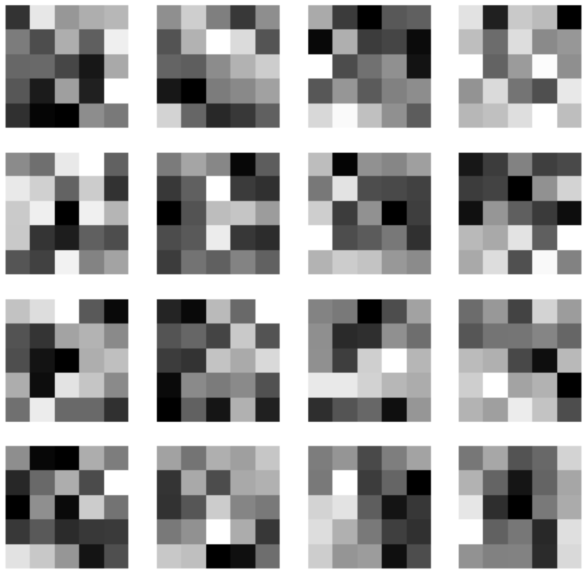
(c)The test accuracy of 5 CNN models is shown in the below table:

	Run #1	Run #2	Run #3	Run #4	Run #5
Test acc	99.16%	99.08%	99.00%	99.12%	99.08%

The best model is given by the first run. Now we can visualize the CNN in the same way as problem4.

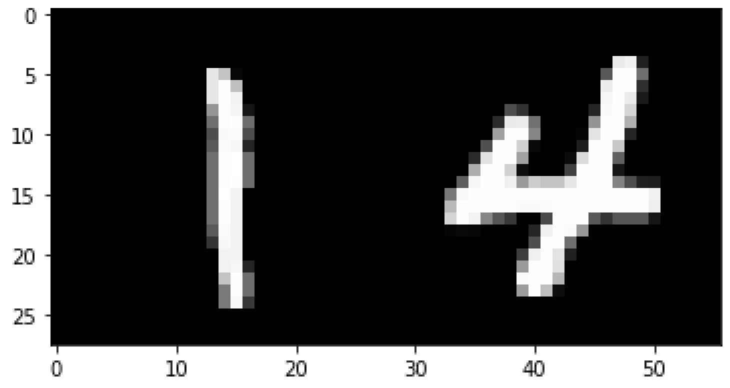


The feature maps for different inputs are shown above. And in this CNN structure, we have 16 output channels in the first convolutional layer. So there are 16 kernels to be visualized.

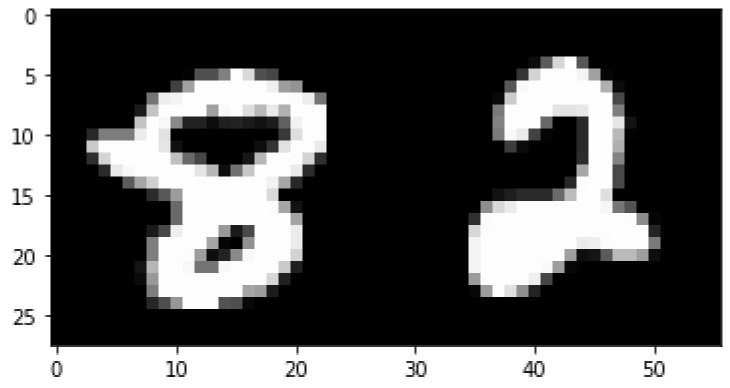


Problem 6

Data set visualization:



label = 5

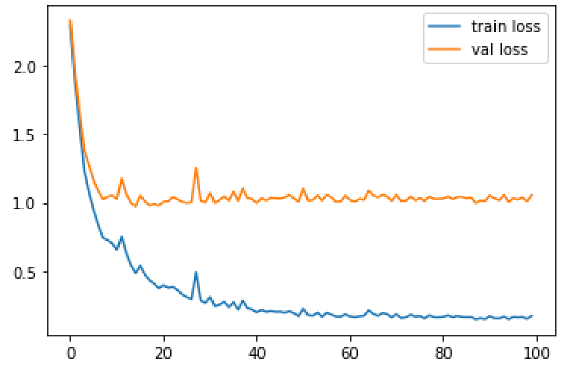


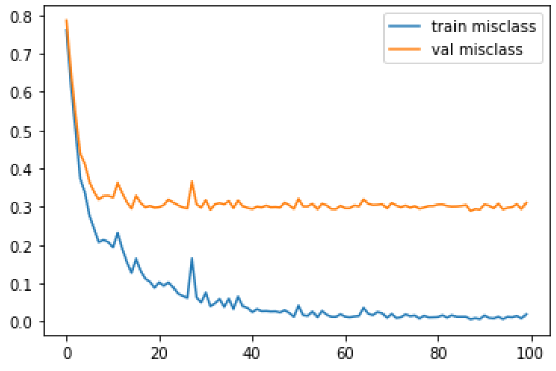
label = 10

For each sample in the data set, pixels are scanned in row-order. The last column of the data set represents labels for each sample. The label is the sum of the 2 digits in the figure.

Problem 7

Model 1: First we use a simple feedforward neural network with 1 hidden layer(200 hidden units). The input of the network is a vector of length 1568 and the output the predicted probability of 19 classes. The training process is shown below. During the training, the validation set is used to evaluate the model on unseen samples. If the current validation accuracy is higher than previous ones, then the current model state is recorded as the best model. In other words, the model selection is done during the training.





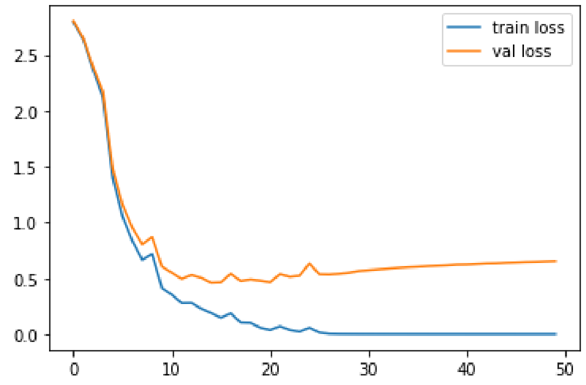
	Train Acc	Validation Acc	Test Acc
Model 1	98.06%	68.88%	71.02%

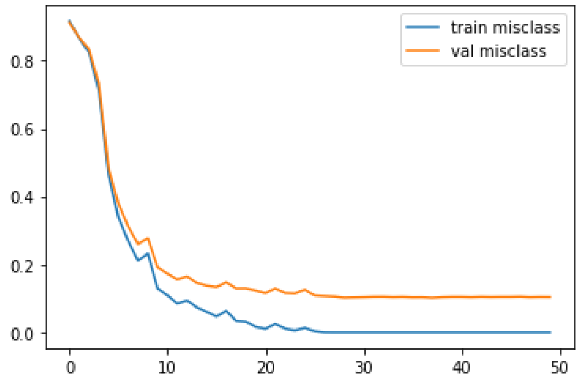
We can see that the simple neural network can only achieve 71.02% accuracy on the test data set. This result tells us that we need to try some complicated models. Visualization of feature maps is shown below.

Model 2: Second, we construct a model similar to the previous CNN. The model structure is shown below.

```
================================================================
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 8, 28, 56]             208
         MaxPool2d-2            [-1, 8, 14, 28]               0
            Conv2d-3           [-1, 64, 12, 26]           4,672
         MaxPool2d-4            [-1, 64, 6, 13]               0
            Conv2d-5            [-1, 32, 6, 13]           2,080
            Linear-6                  [-1, 256]         639,232
            Linear-7                   [-1, 19]           4,883
================================================================
```

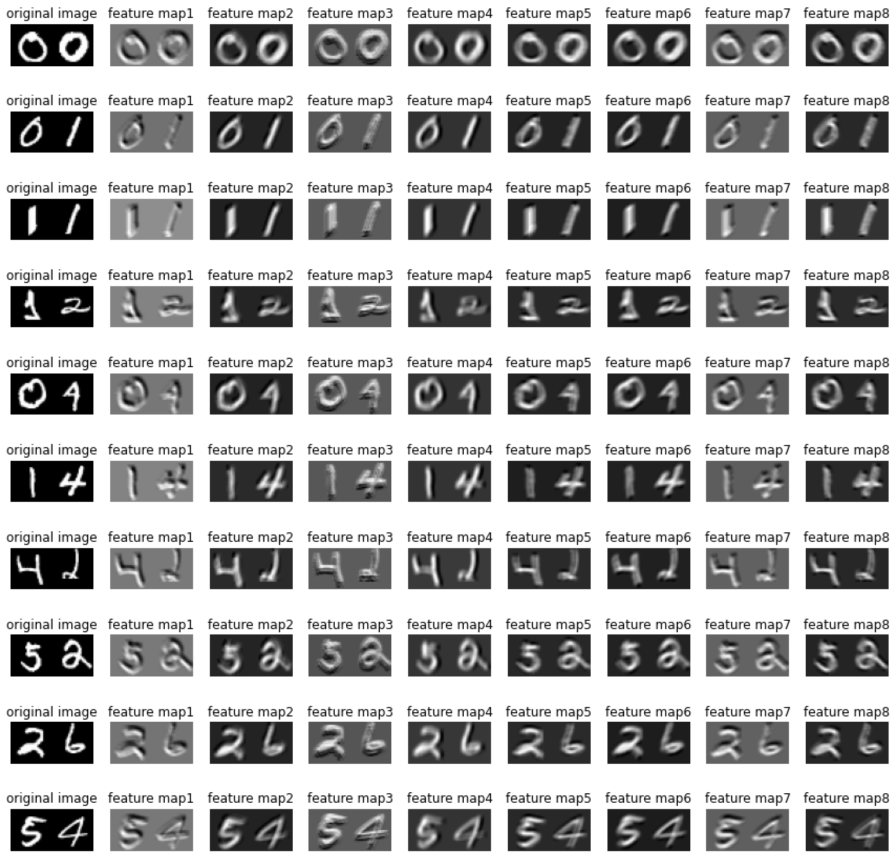
Similarly, the model selection is done during the training.



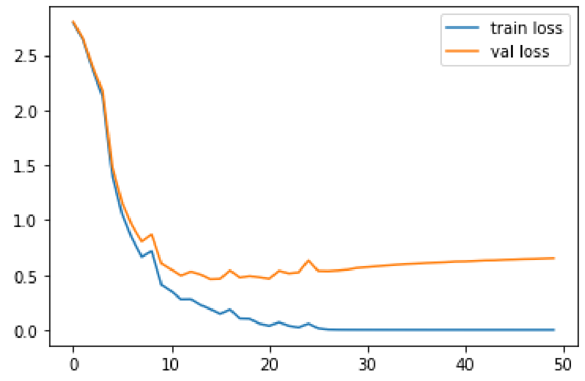


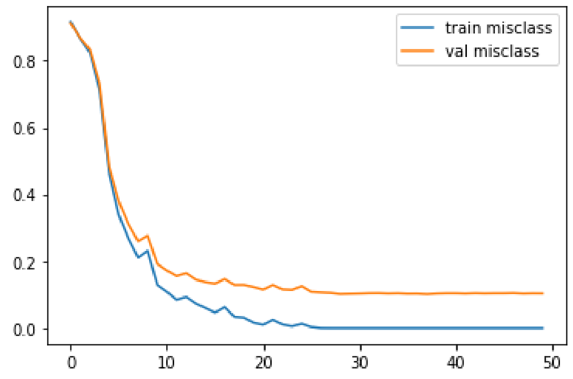
	Train Acc	Validation Acc	Test Acc
Model 2	100%	89.62%	91.02%

As we can see, the convolution structure can significantly increase the model accuracy. The model can achieve 91.02% accuracy on the test set. Visualization of feature maps is shown below.



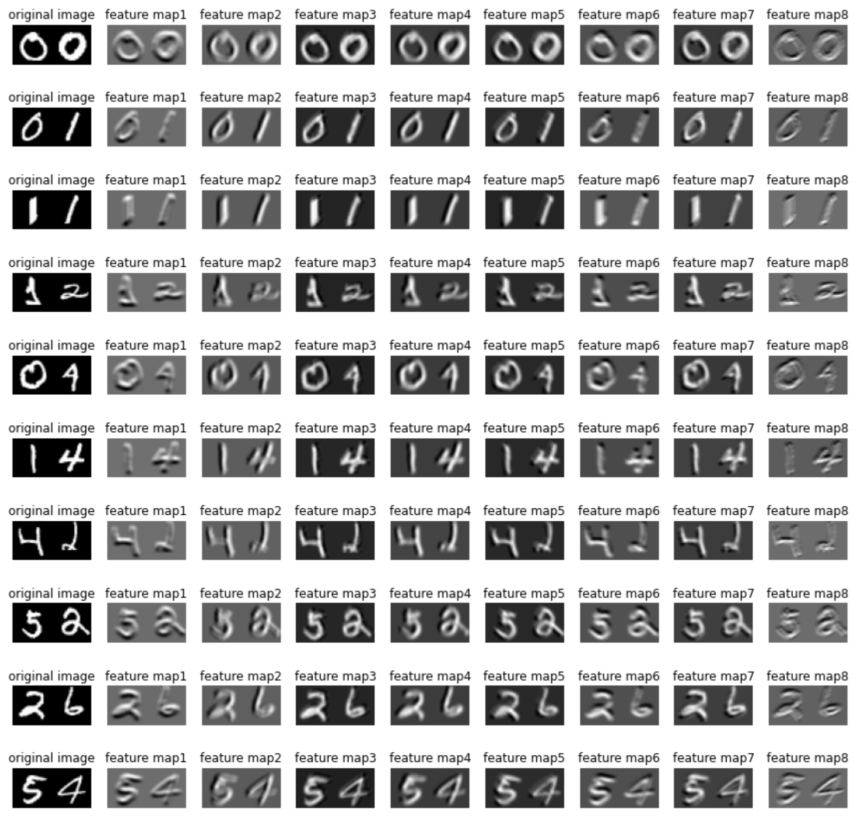
Model 3: Since we have known that the figure can be divided into left and right 2 parts, we can design a CNN to extract features from a figure of size 28 x 28 and concatenate 2 sets of features for classification. The training process and model evaluation are shown below.





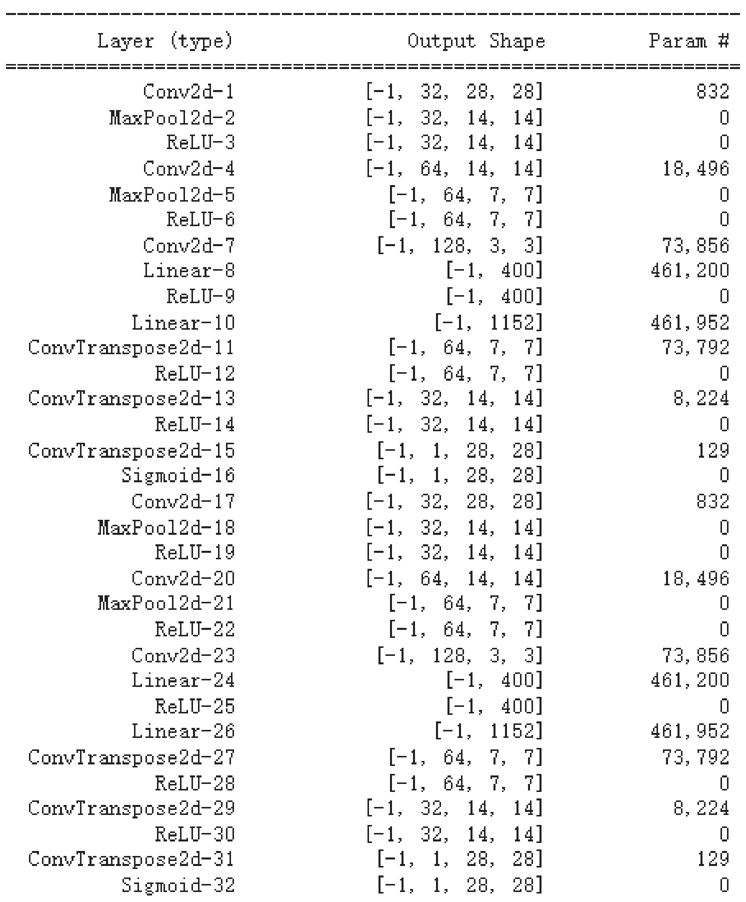
	Train Acc	Validation Acc	Test Acc
Model 3	100%	91.42%	92.28%

Visualization of feature maps is shown below.



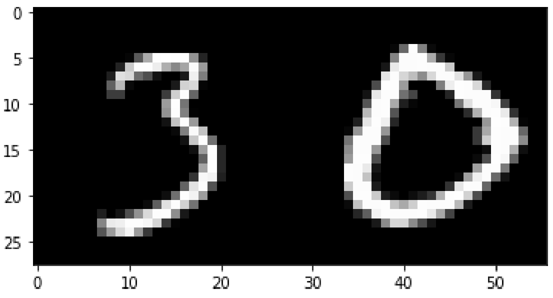
Model 4: Another idea is that we can predict 2 digits separately and do the summation by ourselves. In this case, we need to use the MNIST dataset to train a classifier for single digits. And to make use of the given training data, we propose the following model structure.

First we train a convolutional autoencoder on the training data set to extract useful features from every single-digit image. The input of the autoencoder is an image of size 28 x 28. The autoencoder structure is shown below:

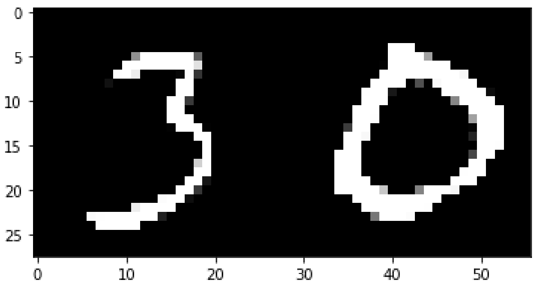


```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 28, 28]             832
         MaxPool2d-2           [-1, 32, 14, 14]               0
              ReLU-3           [-1, 32, 14, 14]               0
            Conv2d-4           [-1, 64, 14, 14]          18,496
         MaxPool2d-5             [-1, 64, 7, 7]               0
              ReLU-6             [-1, 64, 7, 7]               0
            Conv2d-7            [-1, 128, 3, 3]          73,856
            Linear-8                   [-1, 400]         461,200
              ReLU-9                   [-1, 400]               0
           Linear-10                  [-1, 1152]         461,952
  ConvTranspose2d-11             [-1, 64, 7, 7]          73,792
             ReLU-12             [-1, 64, 7, 7]               0
  ConvTranspose2d-13           [-1, 32, 14, 14]           8,224
             ReLU-14           [-1, 32, 14, 14]               0
  ConvTranspose2d-15            [-1, 1, 28, 28]             129
          Sigmoid-16            [-1, 1, 28, 28]               0
           Conv2d-17           [-1, 32, 28, 28]             832
        MaxPool2d-18           [-1, 32, 14, 14]               0
             ReLU-19           [-1, 32, 14, 14]               0
           Conv2d-20           [-1, 64, 14, 14]          18,496
        MaxPool2d-21             [-1, 64, 7, 7]               0
             ReLU-22             [-1, 64, 7, 7]               0
           Conv2d-23            [-1, 128, 3, 3]          73,856
           Linear-24                   [-1, 400]         461,200
             ReLU-25                   [-1, 400]               0
           Linear-26                  [-1, 1152]         461,952
  ConvTranspose2d-27             [-1, 64, 7, 7]          73,792
             ReLU-28             [-1, 64, 7, 7]               0
  ConvTranspose2d-29           [-1, 32, 14, 14]           8,224
             ReLU-30           [-1, 32, 14, 14]               0
  ConvTranspose2d-31            [-1, 1, 28, 28]             129
          Sigmoid-32            [-1, 1, 28, 28]               0
```

We can see that the reconstruction performance is not bad:

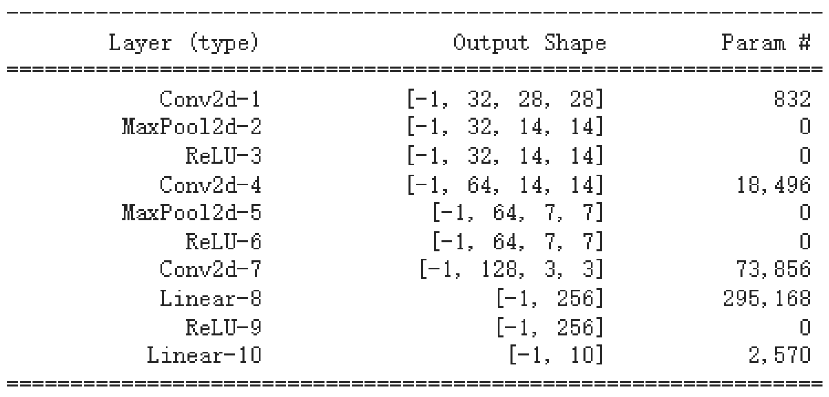


Original



Reconstructed

Then we use the autoencoder to extract 1152 features from each image and use 2 fc layers for classification. When training the classifier, we freeze the parameters from the encoder layers and only train the fc layers.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 28, 28]             832
         MaxPool2d-2           [-1, 32, 14, 14]               0
              ReLU-3           [-1, 32, 14, 14]               0
            Conv2d-4           [-1, 64, 14, 14]          18,496
         MaxPool2d-5             [-1, 64, 7, 7]               0
              ReLU-6             [-1, 64, 7, 7]               0
            Conv2d-7            [-1, 128, 3, 3]          73,856
            Linear-8                   [-1, 256]         295,168
              ReLU-9                   [-1, 256]               0
           Linear-10                    [-1, 10]           2,570
================================================================
```

After using the pre-trained autoencoder for feature extraction, a fully connected classifier can achieve 99.19% accuracy on the MNIST test set.

Finally, we can use the single digit classifier to predict the sum of two digits.

	Train Acc	Validation Acc	Test Acc
Model 4	100%	100%	98.4%

Our model 4 achieved 98.4% accuracy on the test set, but unfortunately didn’t achieve 99% test accuracy. The main reason is that if we want to achieve 99% accuracy, we need to have a 99.5% test accuracy single-digit classifier, which is not achieved in all previous models.