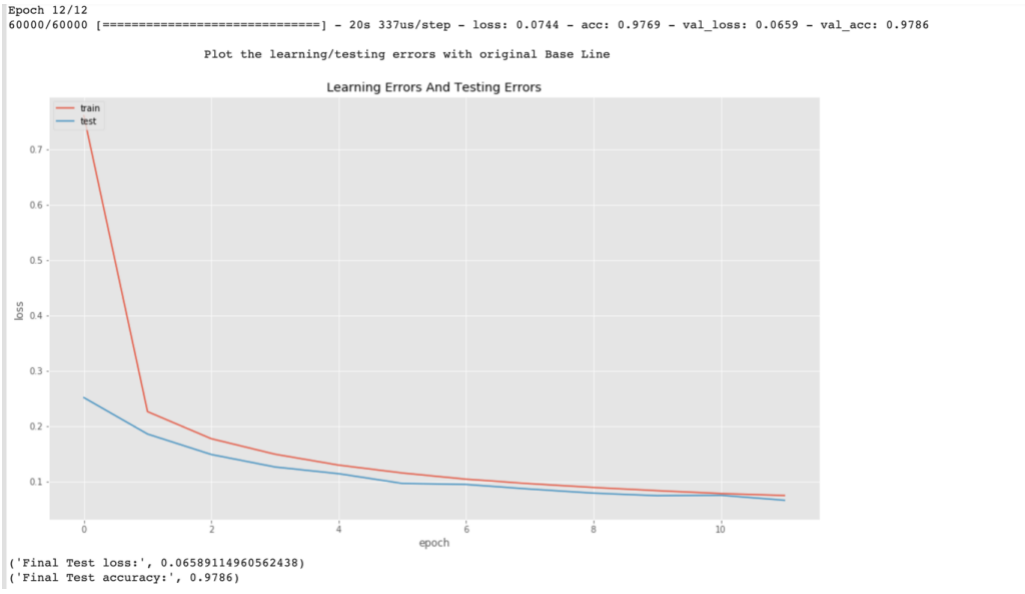
Ziming Dong
CSE 575
Project part 3: Classification Using Neural Networks and Deep Learning
12/7/2019

In this project, I will complete the classification using neural networks and deep learning to classify the data. I use the data which is the same as project 1 from the MINIST dataset. To save time, I only choose the subset of samples which only contain 6000 samples for training and 1000 samples for testing.

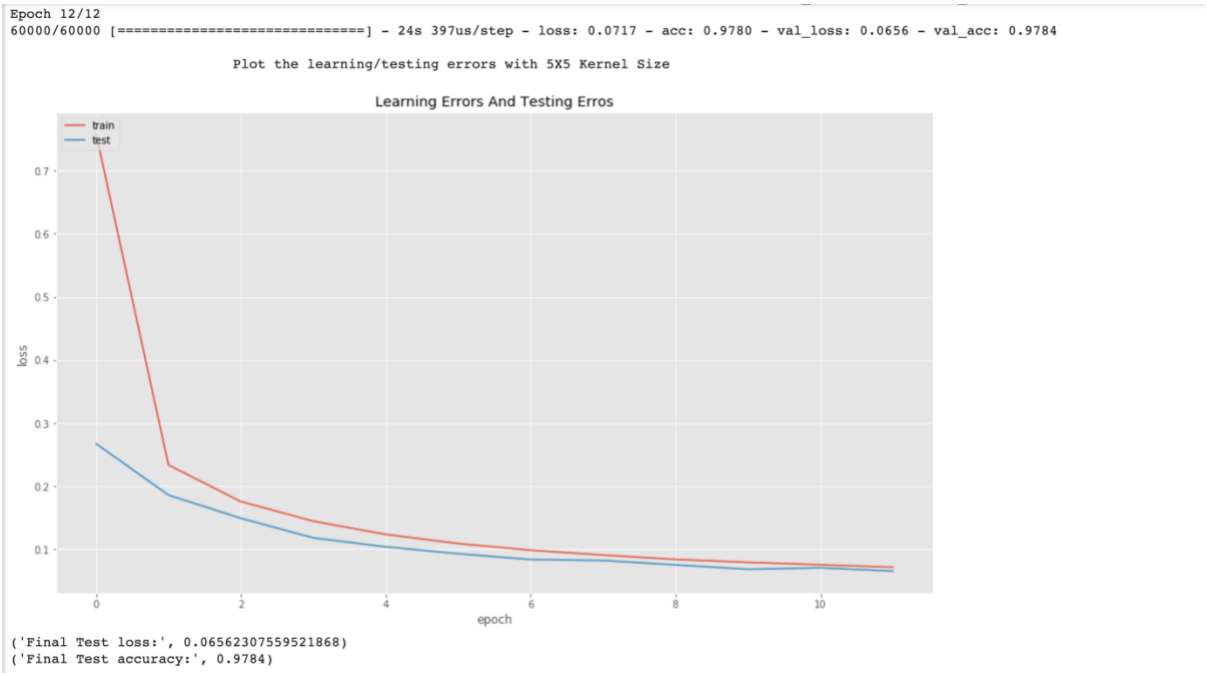
The input size of the image is 28x28 and we have two convolutional layers, first layer has 6 feature maps with 3x3 kernel size. The second convolutional layer has 16 feature maps with 3x3 kernel size. All layers follow a 2x2 pooling with stride 1. After max pooling, the layer is fully connected to the next hidden layer with 120 nodes and relus as the active function. The fully connected layer is followed by another fully connected layer with 84 nodes and relus as the active function. Then there will be 10 classes connected with 10 output nodes in the soft max layer. After implementing these requirements, I also plot the training errors and testing errors as a function of the learning epochs. To see how hyper-parameters influence the training and testing errors under the new setting, I have three experiments for this project. Firstly, I plot the test loss and report accuracy with baseline code. Then I change the kernel size for both layers from 3x3 to 5x5 and plot the test loss and report accuracy. Finally, I change the number of feature maps for both convolutional layers and see loss and accuracy on the testing set.

1. **Plot the learning/testing errors and final classification accuracy with baseline code**:



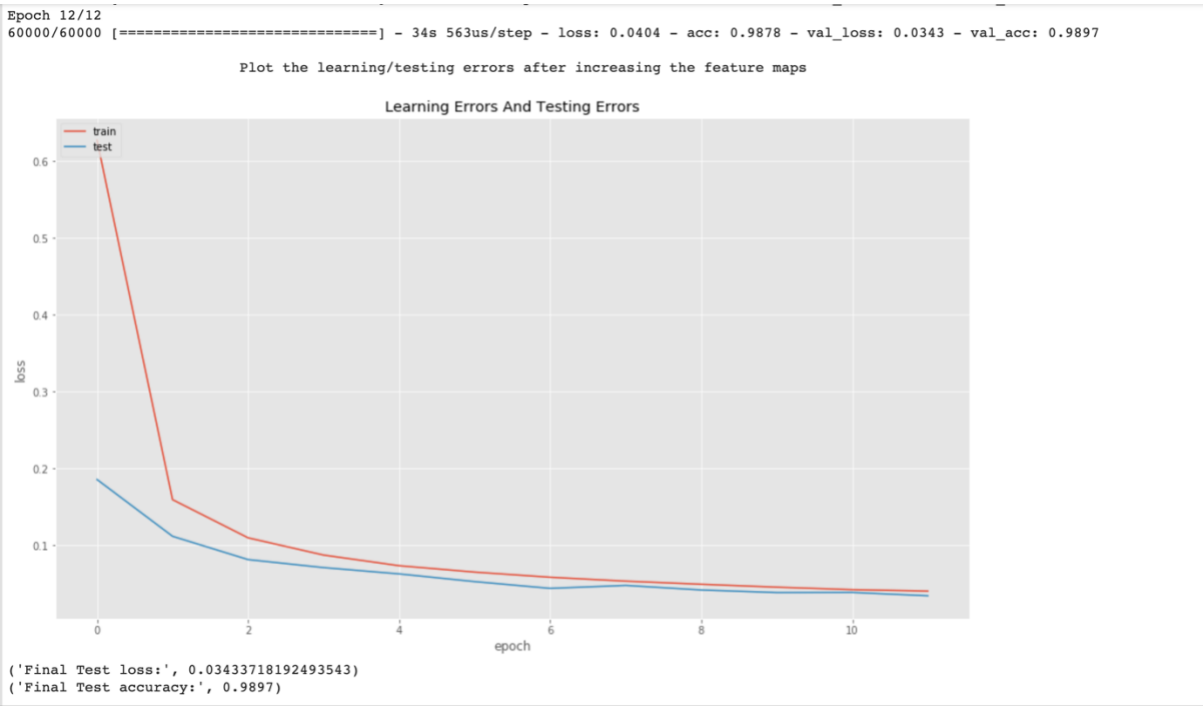
As you can see, the final accuracy is 0.9786 for the base line parameter setting, the training error changed from 0.76 to 0.074 with each epoch, and the testing error changed from 0.25 to 0.065. I think it is not too bad to see it, since the beginning of testing errors are not very high as training errors and both of their error rates are decreasing.

2. **Change the kernel size to 5*5, redo the experiment, plot the learning errors along with epoch, and report the testing error and accuracy on the test set.**



As we see, since we change the kernel size to 5*5 for both layers, the final testing accuracy is 0.9784, the training error changed from 0.7581 to 0.071 and testing errors changed from 0.2674 to 0.0655. I found that learning/testing errors and testing accuracy do not change too much, but if we see the detail of plots, the result with 5*5 kernel size shows a little higher error rates and lower accuracy than the result with 3*3 kernel size convolutional layers. Since this experiment is composed of 2 convolutions block 2 dense layers. The convolutional kernel size need for convolutional layer is $c * (k^2 * 1^2)$, to reduce the number of weights, we need $c * (k^2 * 1^2) + 1^2 * c^2$, let`s consider the 5*5 kernel size convolutional layer, k square is smaller than c >128, so we have a ratio of rough kernel surface: 9 or 25. At this point, I think both 3*3 and 5*5 kernel size convolutional layers will have good performance in this experiment.

3. **Increase the number of feature maps in the first convolutional from 6 to 16, and second convolutional layer from 16 to 64, then redo the experiment, plot the learning errors along with epoch and report the testing error and accuracy on the test set**.



After I increase the number of feature maps for both convolutional layers. The training error changed from 0.6272 to 0.404 and the testing error changed from 0.1856 to 0.0343, the final testing accuracy is 0.9897. I see the final test accuracy become higher than before, and the final test loss becomes lower than before, which means that increasing number of feature maps can improve the experiment performance for testing data.

In conclusion, after redoing the experiments three times with changing parameters, such as kernel size, the number of feature maps. I found the testing result does not change too much when I change the kernel size to 5*5, I assume both 3*3 and 5*5 kernel size are making sense when we doing neural networks and deep learning experiment when we have 2 convolutional layers and two dense layers, in some situations, 3*3 kernel size may perform a little bit better than 5*5 kernel size for convolutional layer. Furthermore, I found increasing the number of feature maps for convolutional layers can improve the testing result a lot, the training and testing errors will be lower and the testing accuracy will be higher. If I have more time in the future, I would like to use average pooling and use another optimizer to redo the experiment, I believe these hyper-parameters can influence the result of the experiment a lot.