PA1-Q2 Report

Michael Fielder NID: 4157982

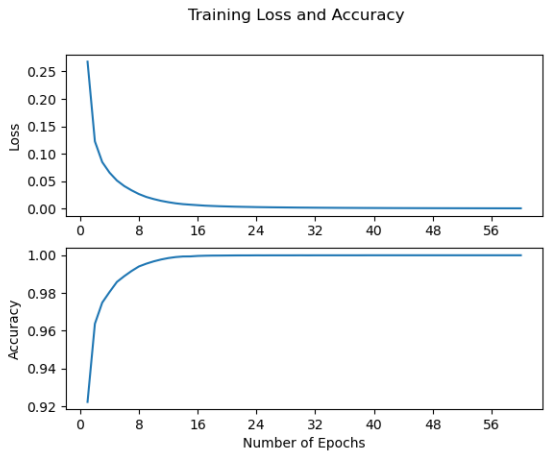
October 24, 2021

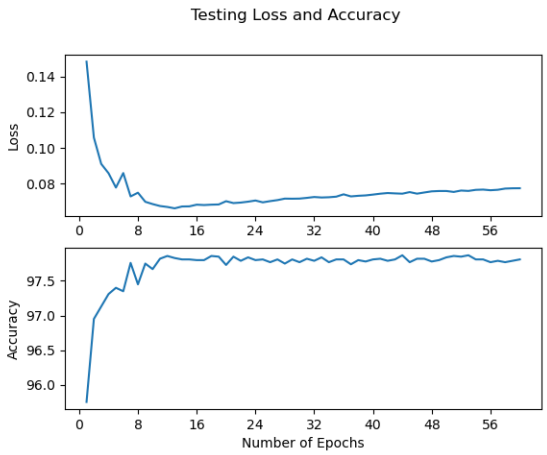
Introduction

For this assignment, I submitted text file containing all of the output text created by the model in the file `output.txt`. Next, there are the Python files `train_evaluate_CNN.py` and `ConvNet.py` which contain code written using the Pytorch library to create several AI models. Below is an explanation for each part of the question.

Part 1

For Part 1, I constructed a neural network with one hidden layer of 100 neurons and a sigmoid activation function. I calculated the loss using Cross-Entropy and performed back propagation with SGD at a learning rate of 0.1. The network ran for 60 epochs with a mini-batch size of 10. Below are the graphs of the training and test results,

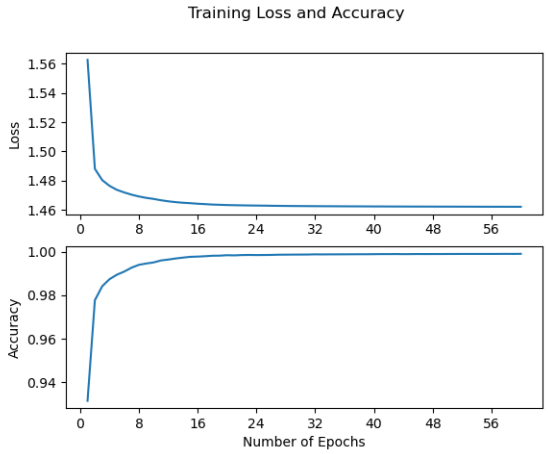


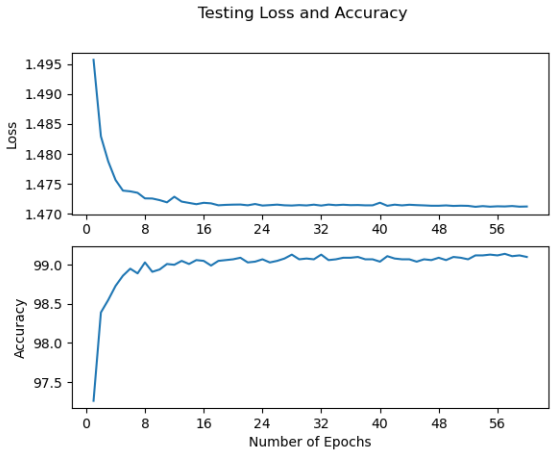


Overall, this neural network achieved a accuracy of 97.87%. This is very high for a neural network using

Part 2

The network for Part 2 is a direct continuation of the one found in Part 1. This time, the input is passed through two 2dConv layers with 40 kernels, a stride of 1, 40 kernels and a kernel size of 5. This network is trained with the exact same parameters as before. Below are the graphs of the training and test results,

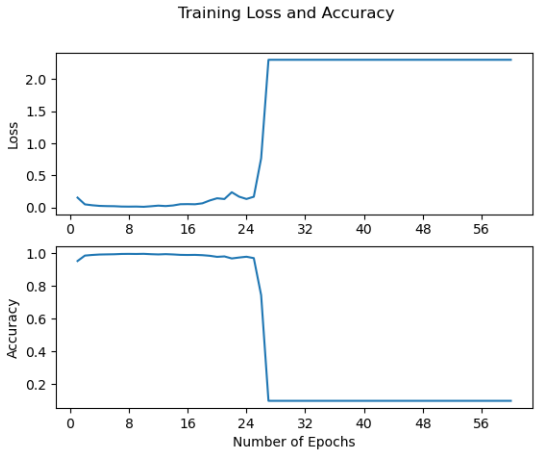


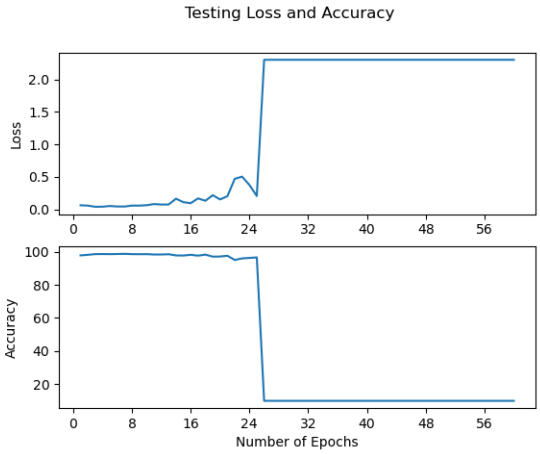


Part 2's network achieved an accuracy of 97.85%, which is very similar to the results of Part 1's simple neural network.

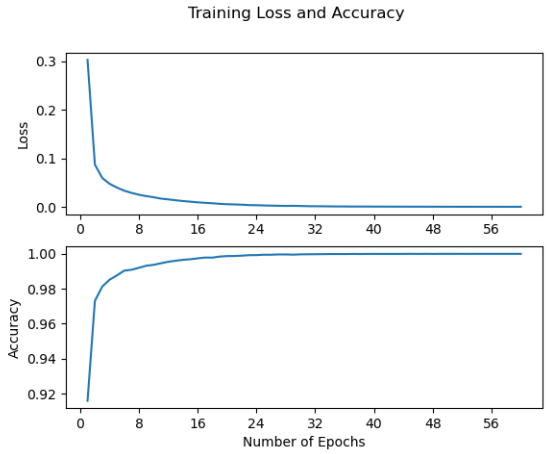
Part 3

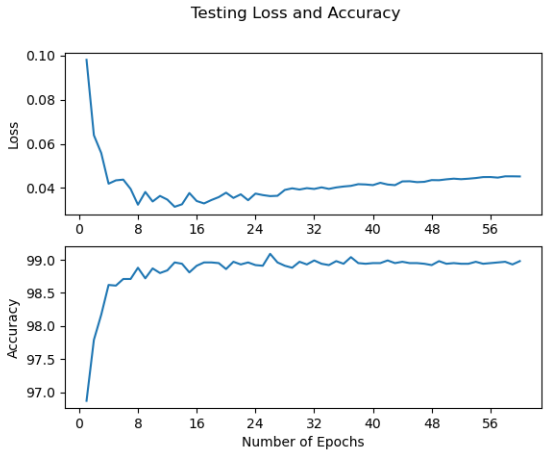
Part 3's neural network is the same as Part 2's except the Sigmoid activation function is replaced with a ReLU activation function. Also, the learning rate is changed to 0.03. Below are the results for this network,





In this Part's example, using a learning rate of 0.03 causes the network's loss value to jump up to above 2 around 24 epochs. This is not a desired output for my network as it causes the accuracy to drop to below 20%. A common cause of the learning rate jumping up like this is a high learning rate. I decided to run the same network again using a learning rate of 0.003 and these are the results that I achieved.

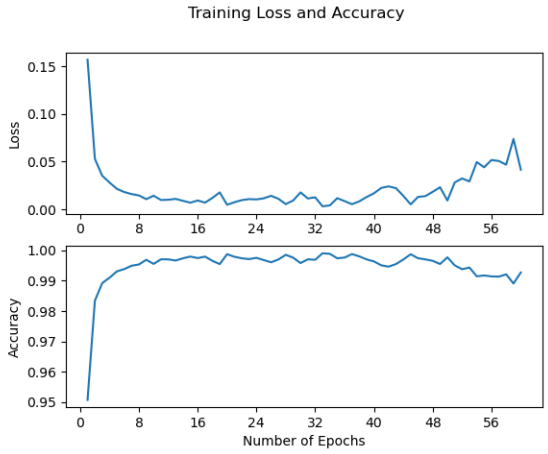




By lowering the learning rate, I was able to achieve a reasonable loss with a graph that follows a favorable trend downwards. Overall, the accuracy 99.09% which is significantly higher than the previous 2 Part's.

Part 4

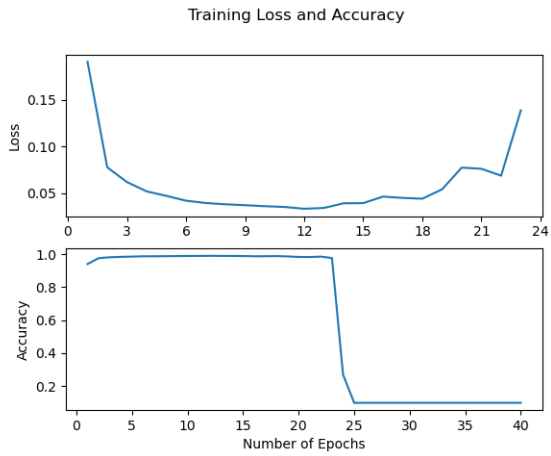
For Part 4, I constructed a network that is similar to the previous part with the only difference being an extra hidden layer with 100 neurons. It was also trained with learning rate of 0.03. This network did not seem to have the same issue as the previous Part's and I was able to get a graph that maintained a consistent loss,

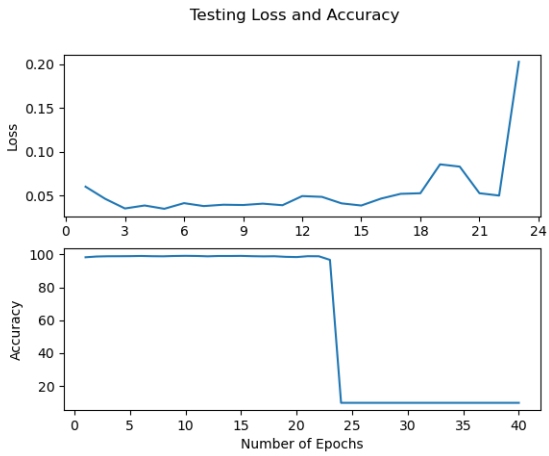


This Part's network achieved an accuracy of 99.11% which is very high and better than the Part 3's adjusted learning rate model. My explanation for why this model did not suffer the same learning rate pitfall was because of the inclusion of an extra hidden layer, as that was the only change between the two networks.

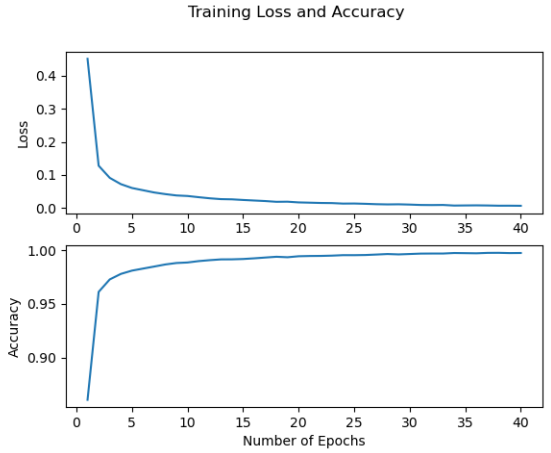
Part 5

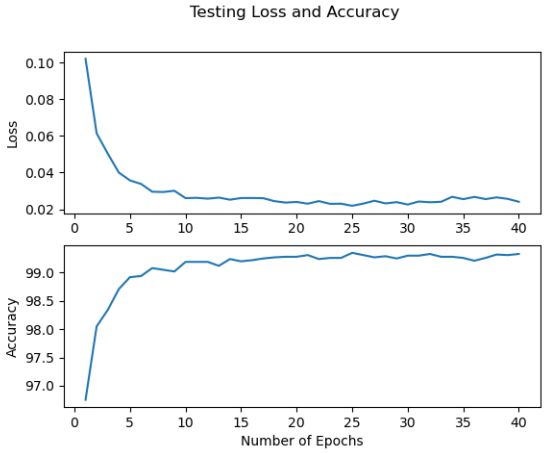
Part 5's model is the same as the previous Part 4's model with the addition of dropout layers with a probability of 0.5 and increasing the number of neurons in the hidden layers from 100 to 1000. Also, this model ran for 40 instead of 60 epochs and used a learning rate of 0.03. Here are the results,





Using a learning rate re-introduced the same problem found in the previous step. I solved this problem the same way I solved the previous problem. I reduced the learning rate and received the following,





This network achieved a accuracy of 99.35%. Which is the highest accuracy of all the models. The best explanation for this is that the regularization provided with the dropout layers and the inclusion of more neurons in the layer helped prevent overfitting in the network, leading to a better overall accuracy.

Conclusion

Below is a table of each models accuracy,

Model	Accuracy
Part 1	97.87
Part 2	97.85
Part 3	99.09
Part 4	99.11
Part 5	99.35

As you can see from the table, each part increases the overall accuracy percentage with a big jump between Parts 2 & 3. This could be down to changing the activation function, but it most likely came from lowering the learning rate from 0.1 to 0.003. There is minimal improvement between the models after Part 3. This is probably due to the addition of extra hidden layers and regularization.