Ryan Filgas
Machine Learning
Program 1

Experiment 1: Test the network with 20, 50, and 100 hidden units.

(1) How does the number of hidden units affect the final accuracy on the test data?

The number of hidden units increases the final accuracy of the testing data from 93.5 up to 96.5 and reduces fluctuations in accuracy across epochs.

(2) How does it affect the number of epochs needed for training to converge?

Less epochs are required to converge when there are more hidden units.

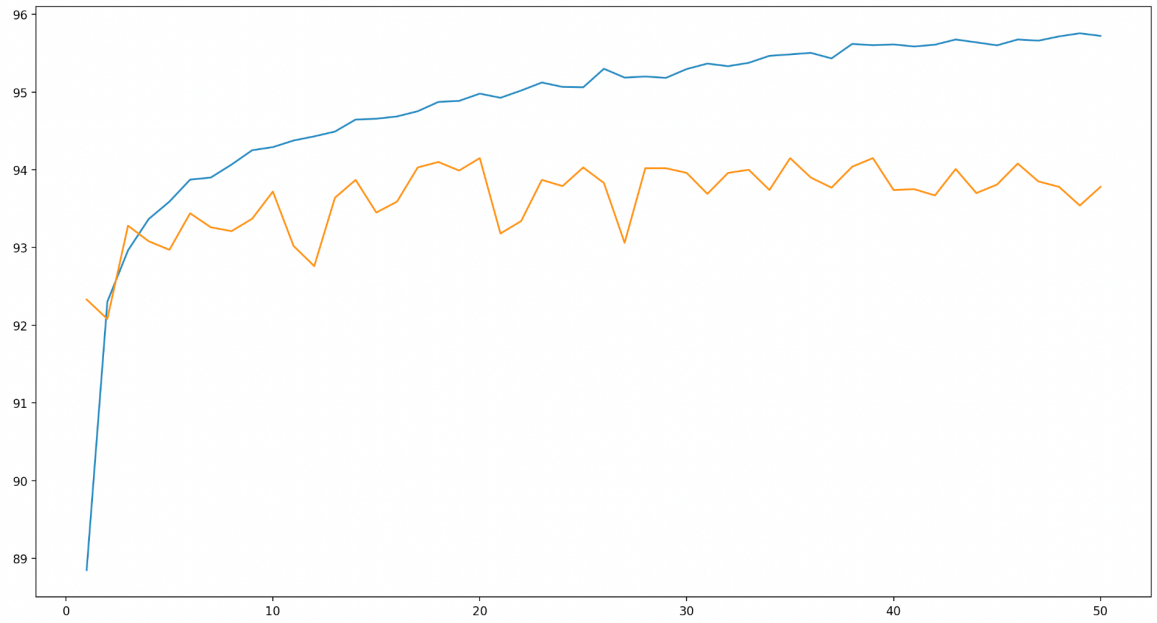
(3) Is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

There is evidence of overfitting to the training set in the data in the last model. The accuracy in the center of the confusion matrix has dropped between 50 and 100 epochs despite overall accuracy increasing. In general the model should be getting better across all fields for the testing set over time, but in this case it hasn't quite.

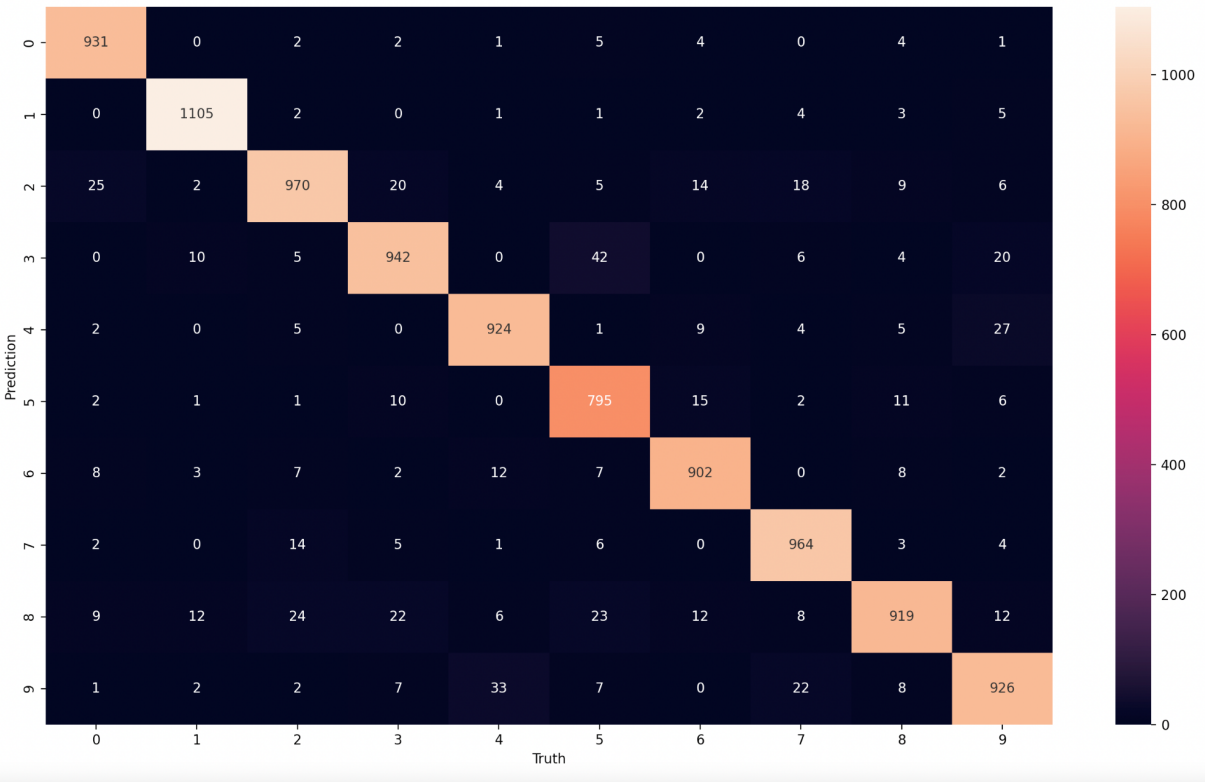
(4) How do your results compare to the results obtained by your perceptron in HW 1?

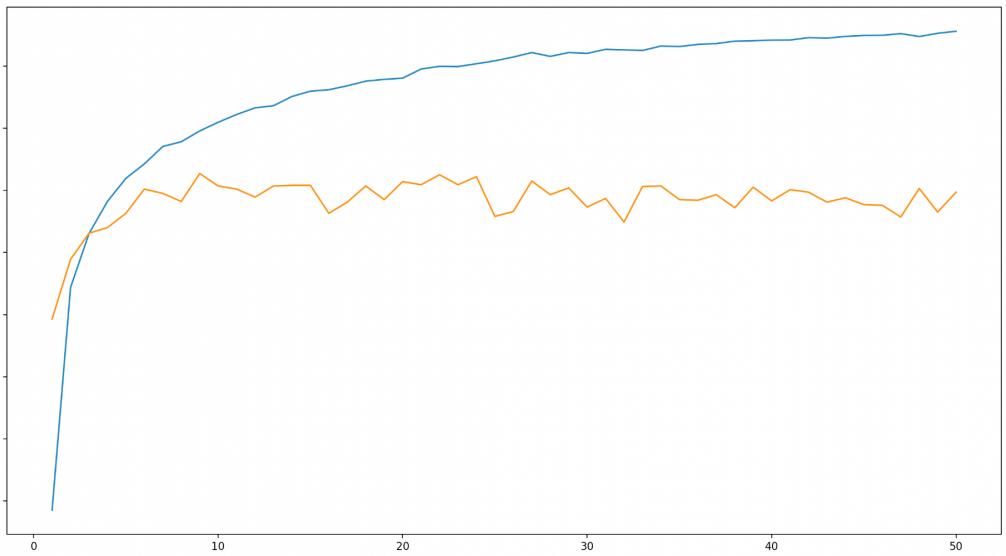
In homework 1 my program achieved around 88% accuracy with larger fluctuations. The algorithm implemented here is significantly better, performing at about 96% on the testing data. In addition the fluctuations in efficiency have been stabilized significantly and this model converges faster.

20 Hidden Units



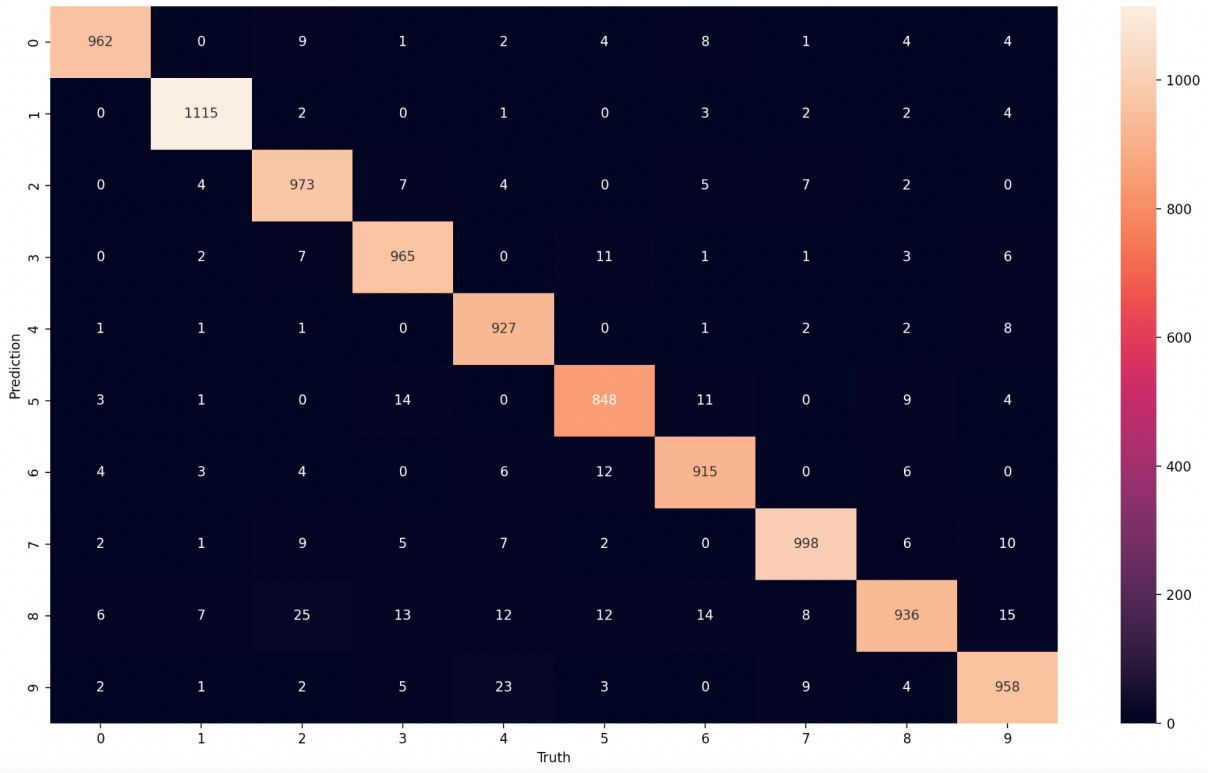
20 Hidden Units

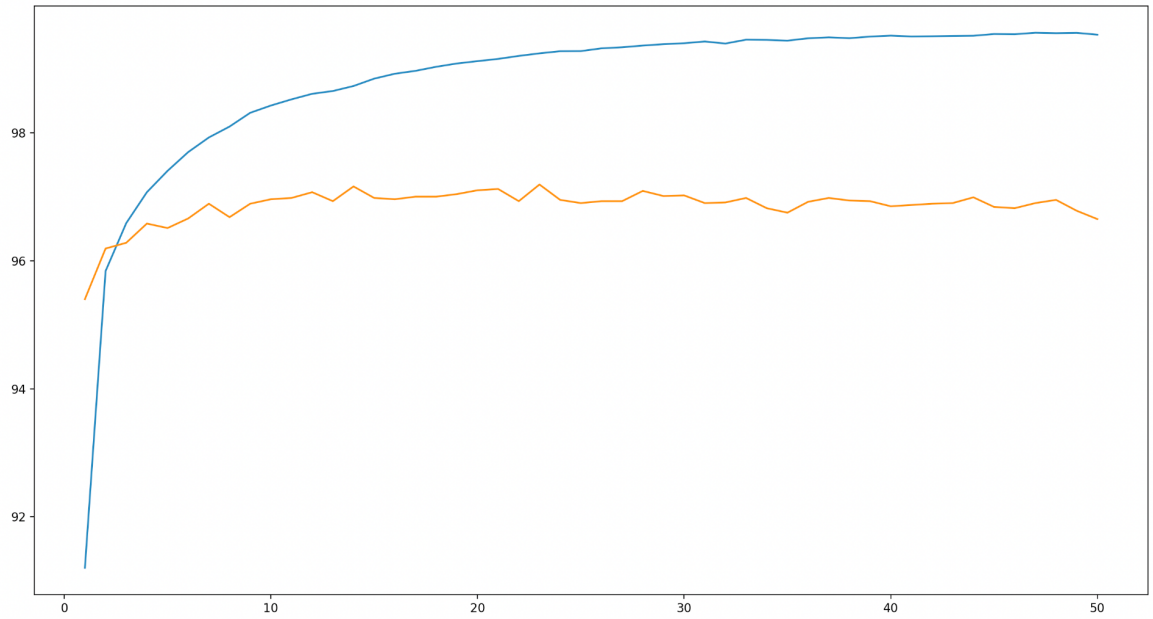




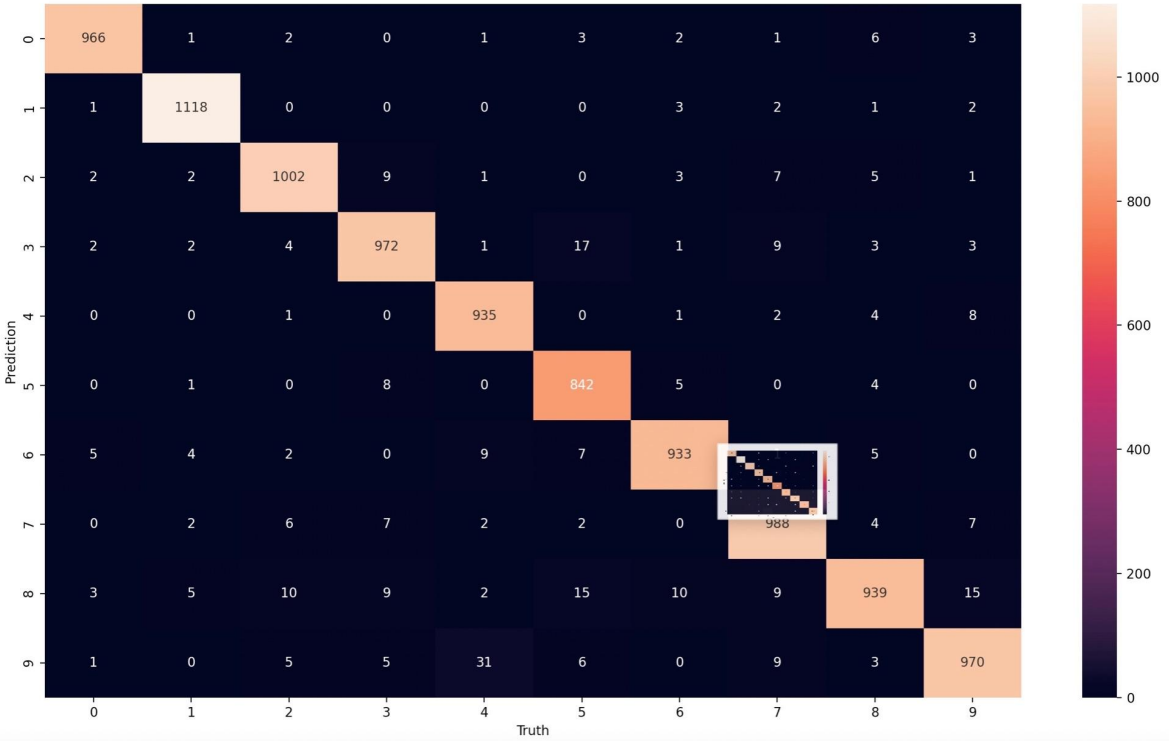
50 hidden units

50 hidden units





100 hidden units
100 Hidden Units



Experiment number 2 starts below:

Experiment #2: Vary the momentum value (0, .25, .5)

(1) How does the momentum value affect the final accuracy on the test data?

The momentum doesn't seem to significantly affect the final accuracy on the test data too significantly, however it converges to that accuracy faster. More training in this example meant a lower score on the test at the end than other models, and a wider gap between testing and training accuracy results.

(2) How does it affect the number of epochs needed for training to converge?

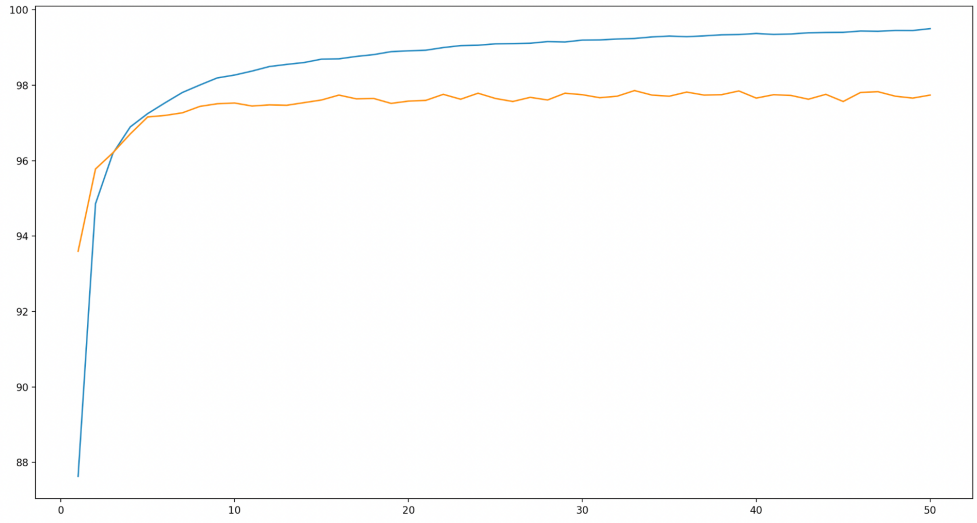
The number of epochs needed to converge is smaller. This is easiest to see in the testing data where the highest momentum value causes the testing data to converge almost immediately.

(3) Again, is there evidence that any of your networks has overfit to the training

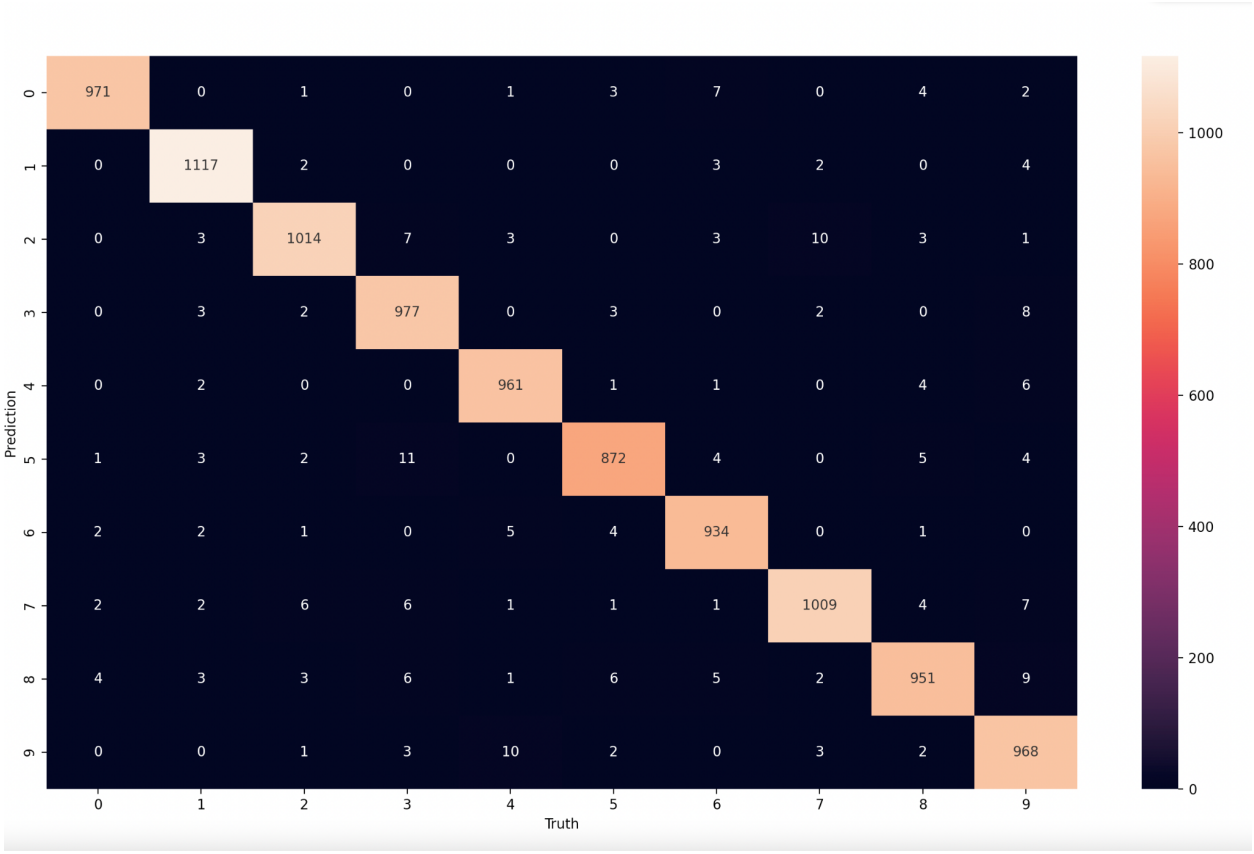
data? If so, what is that evidence?

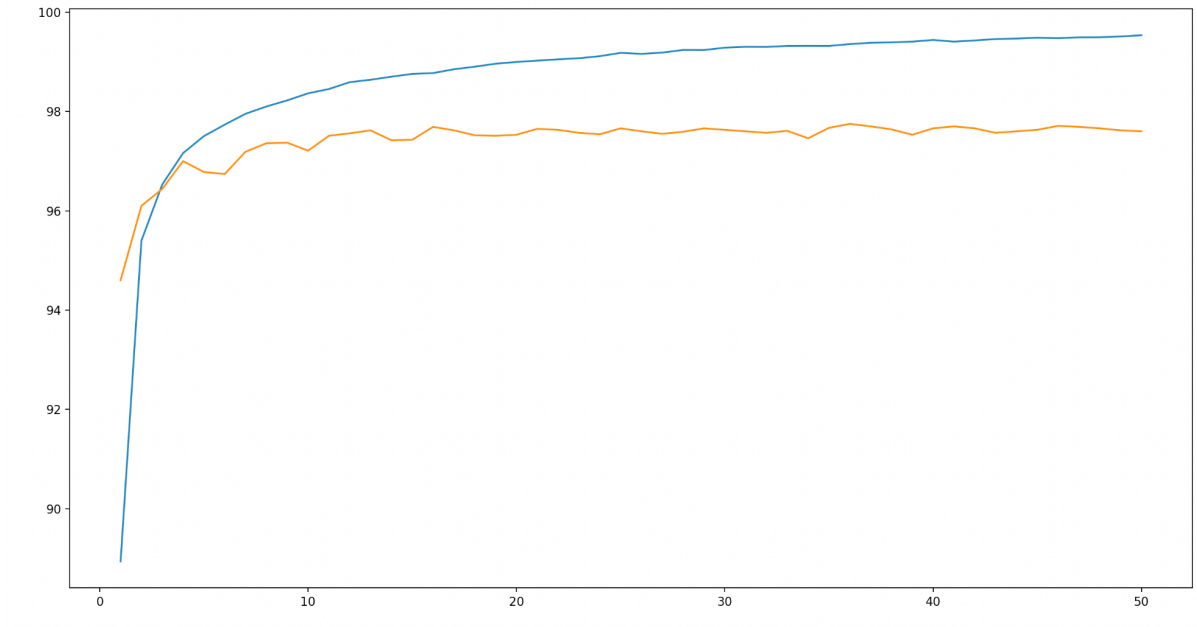
Converging faster in the case of the .5 momentum example meant lower testing accuracy by the end. More momentum past .25 didn't have as much positive effect. In the .25 data set you can see that the trajectory corrections are slightly more aggressive then when momentum is set to zero.

0 Momentum

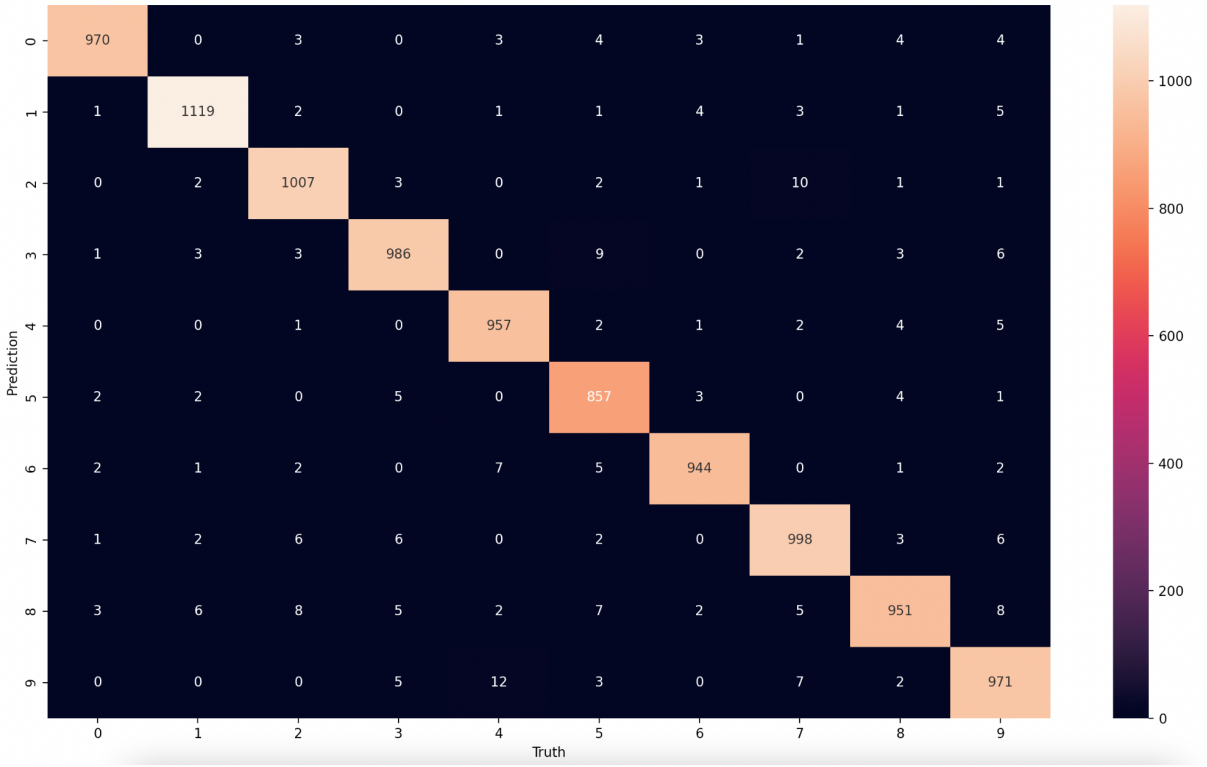


0 Momentum

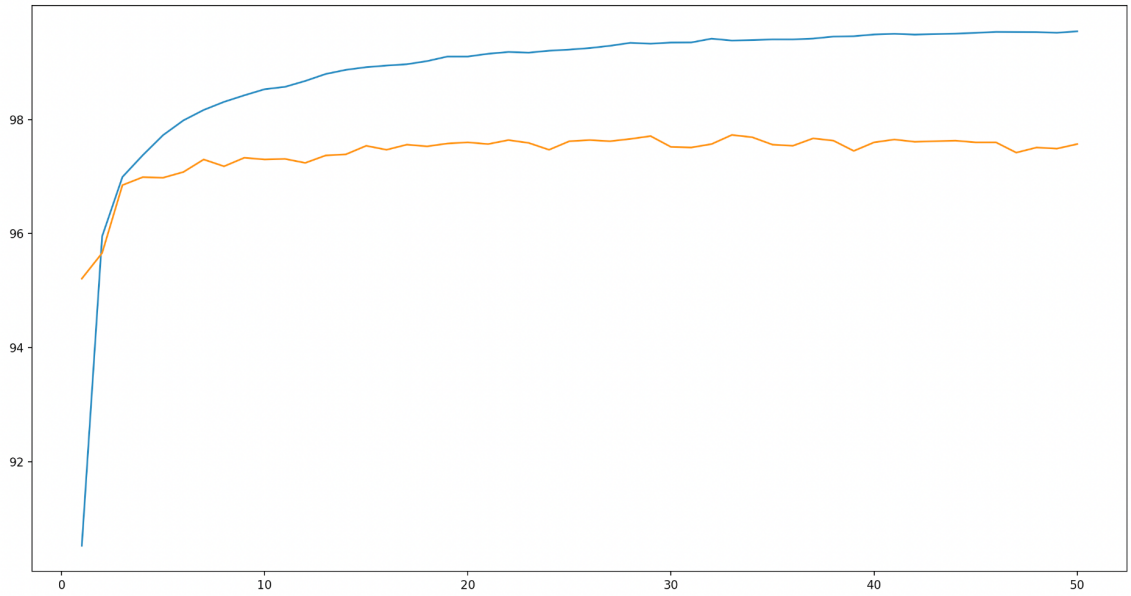




.25 Momentum



.25 Momentum



.50 Momentum

Experiment 3 is on the next page.

Experiment number 3:

(1) How does the size of the training data affect the final accuracy on the test data?

The final accuracy on the training data goes up with more training examples as would be expected.

(2) How does it affect the number of epochs needed for training to converge?

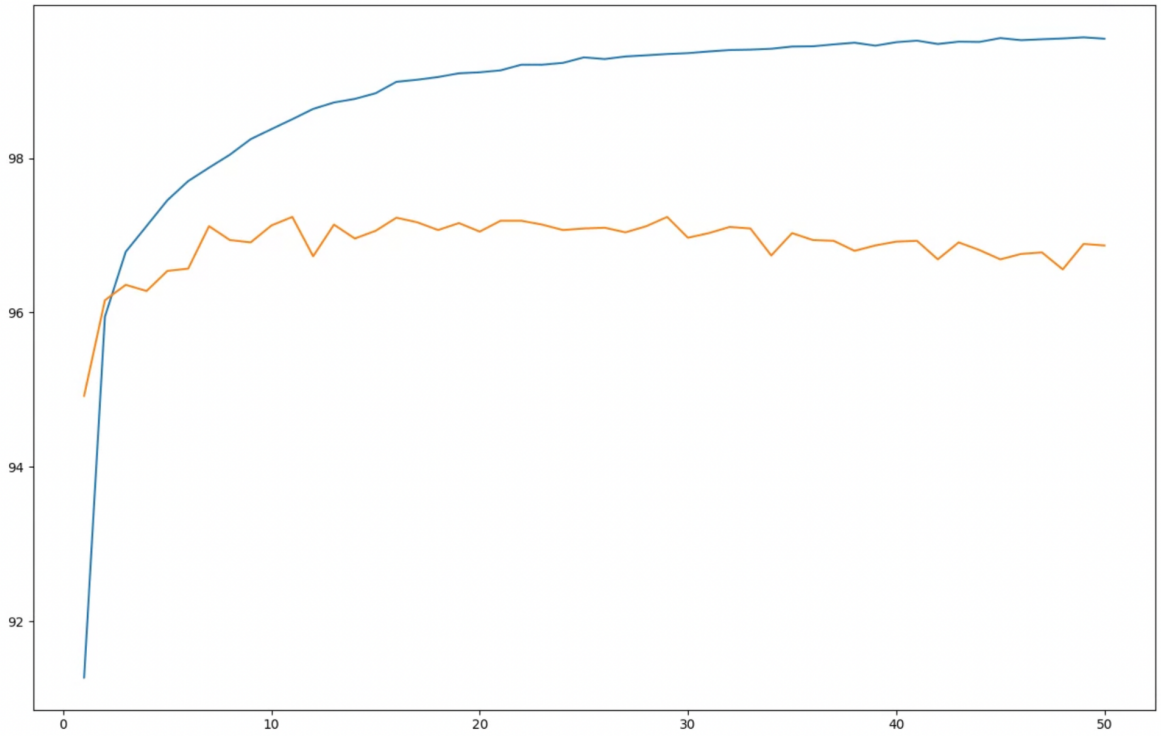
The model converges with less epochs when there is more data to process. This doesn;t necessarily mean it's converging "faster" yet as it crunched more data to do so, but it does mean it will converge in less epochs.

(3) Again, is there evidence that any of your networks has overfit to the training data? If so, what is that evidence?

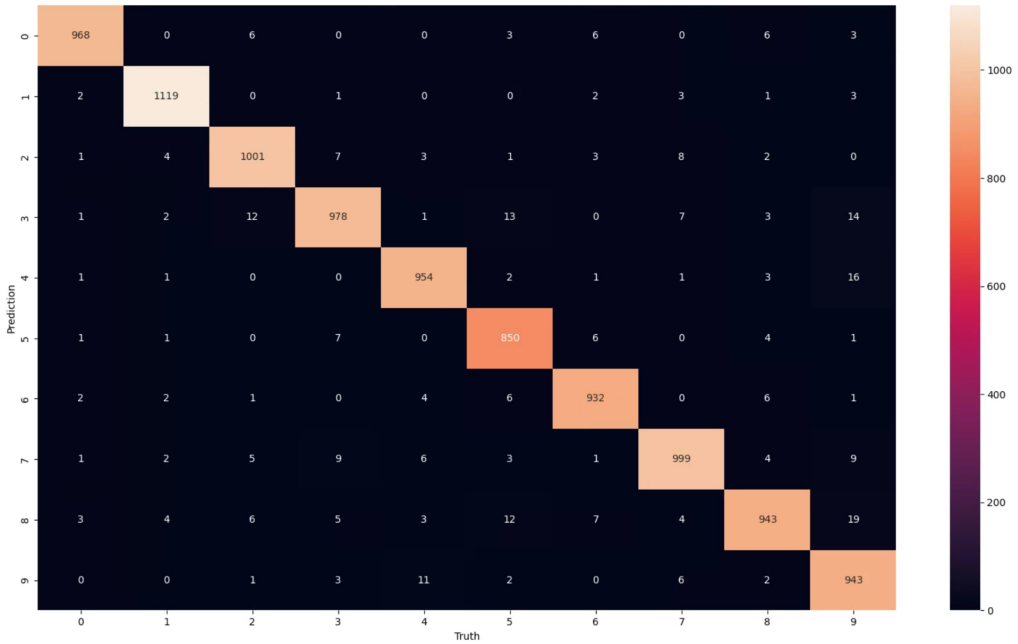
The network with less data shows more overfitting as the testing accuracy fluctuates more significantly than the other. It's also apparent that both graphs seem to get worse over time, and not better which would further indicate overfitting. The both show overfitting to an extent because the training data continues to go up in accuracy while the testing data is essentially flatlined or decreasing.

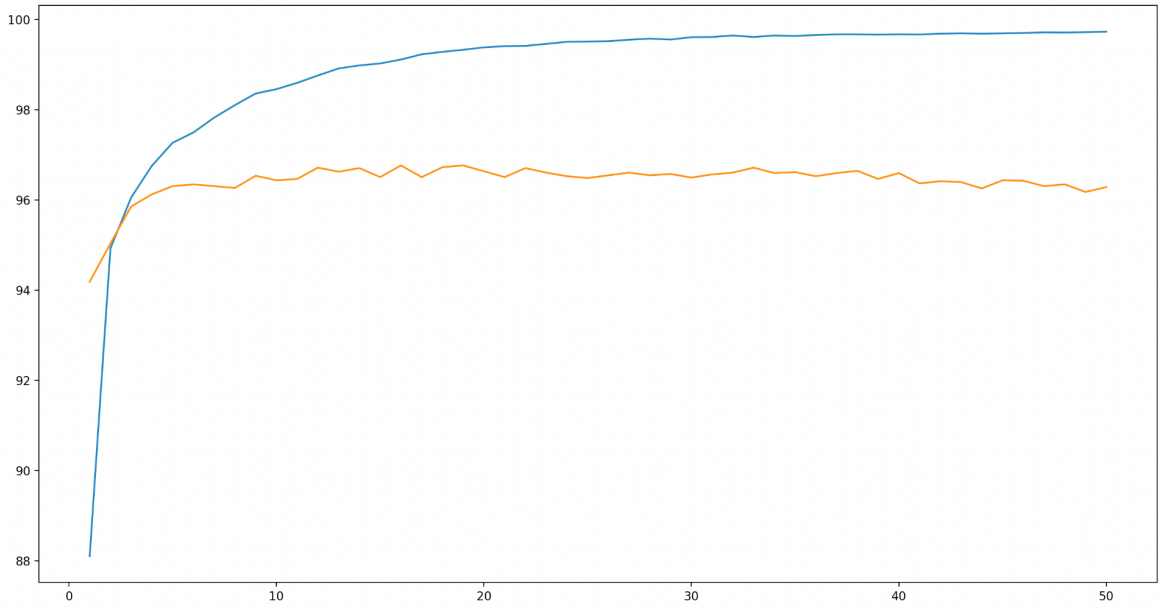
More diagrams below->

25% of training data



25% of Training Data





50% of training data

50% of training data