Kobe Davis (901199860)

Prof. Doliotis

CS 445

10 February 2020

Assignment 2: Neural Networks

Introduction

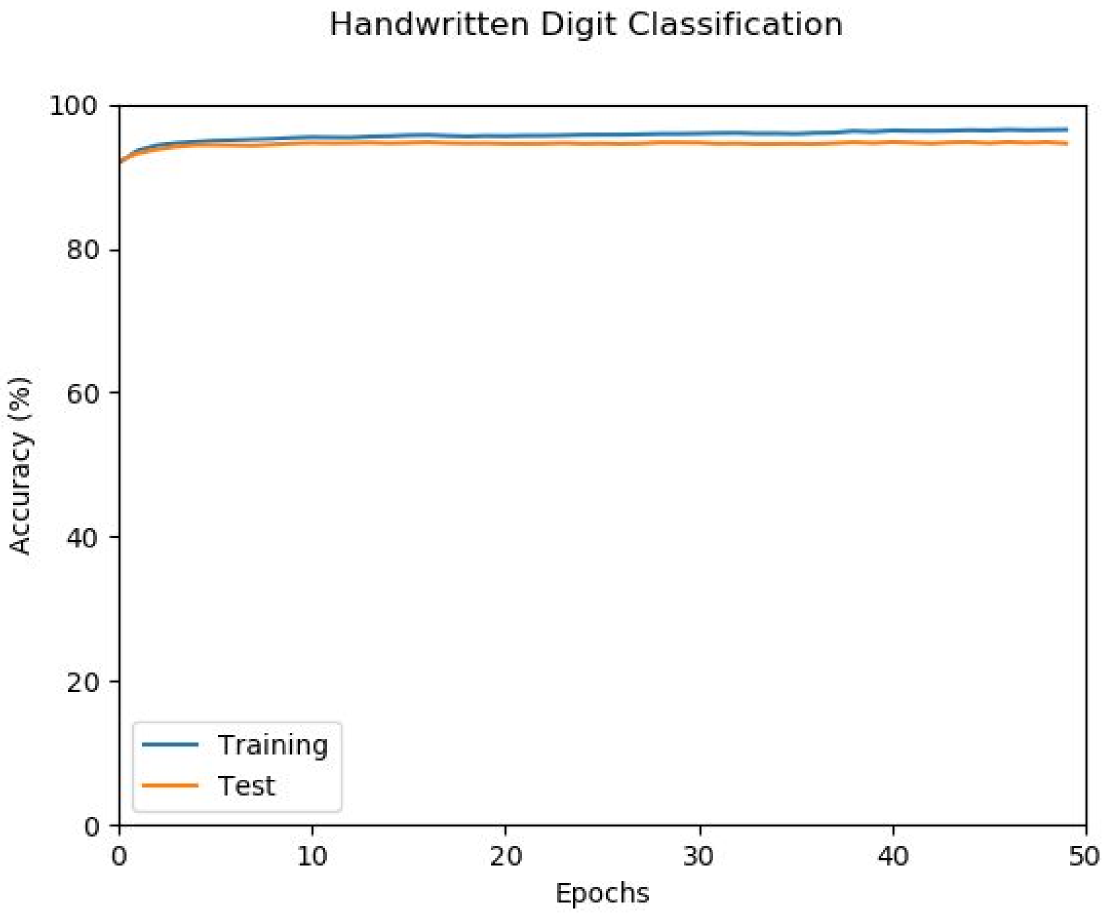
This assignment involved implementing and training a two-layer neural network to perform handwritten digit recognition. The neural network code was written from in Python (utilizing the Numpy library) and the mnist training and test datasets were used to train and record accuracy of the neural network. The images of handwritten digits used as input to the neural network are in a 28x28 pixel format. Hence, this neural network required an input layer consisting of 785 input units.

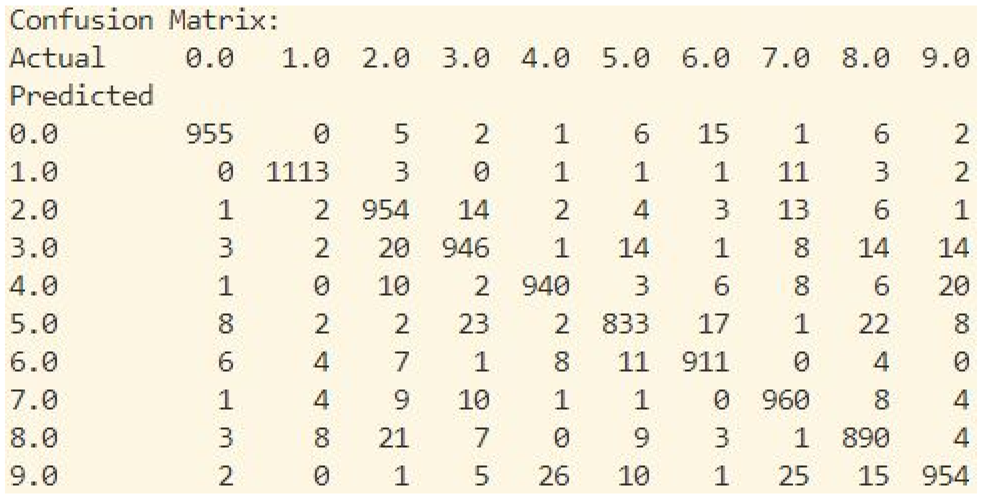
Similar to the first assignment, the input data is preprocessed such that pixel values are normalized to values between 0 and 1. The input data is shuffled as well. The primary difference between this assignment and the last is the presence of a hidden layer within the neural net, and thus requiring two weight matrices (input-to-hidden and hidden-to-output). Learning rate is kept static at 0.1 throughout each experiment, with that said, this assignment describes two experiments:

1. Vary the number of hidden units within the hidden layer between 20, 50, and 100.
2. Vary the number of training examples used between a quarter, half, and all.

Experiment 1

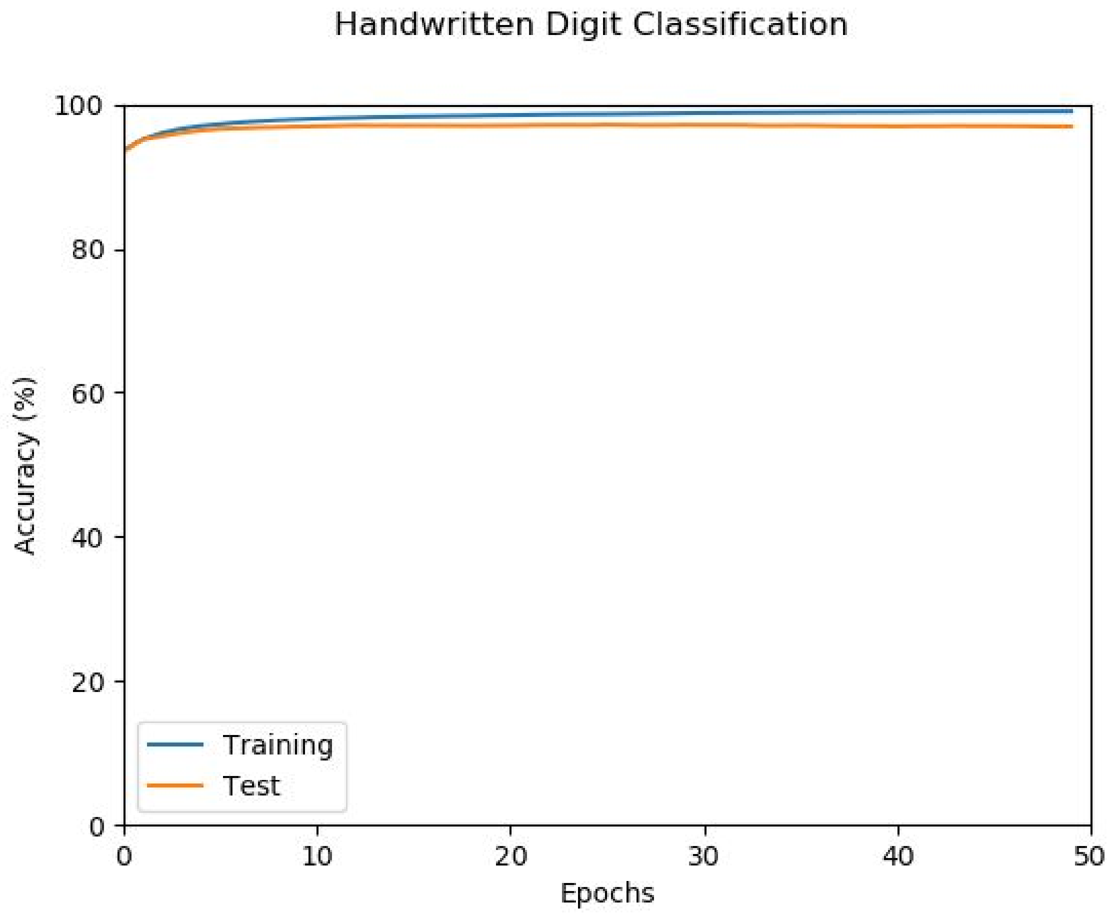
Twenty Hidden Units

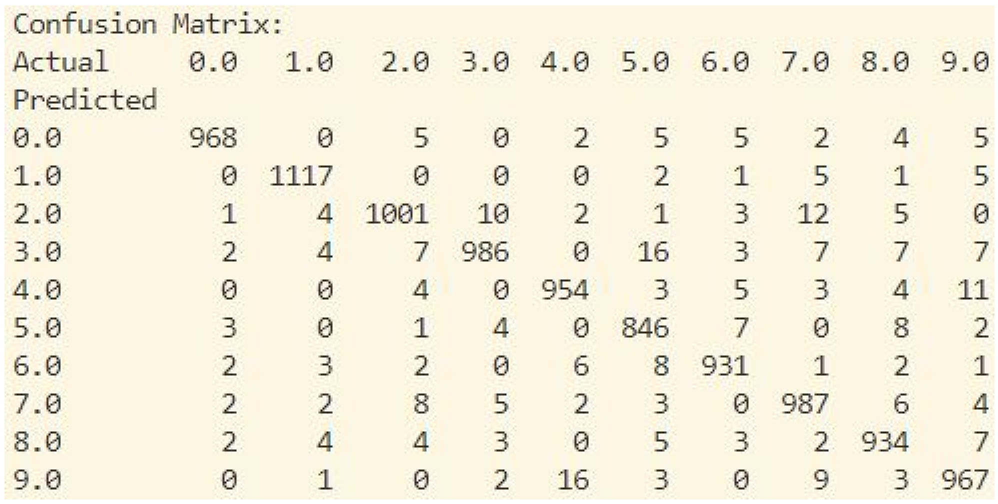




```
Confusion Matrix:
Actual      0.0   1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
Predicted
0.0         955     0    5    2    1    6   15    1    6    2
1.0           0  1113    3    0    1    1    1   11    3    2
2.0           1     2  954   14    2    4    3   13    6    1
3.0           3     2   20  946    1   14    1    8   14   14
4.0           1     0   10    2  940    3    6    8    6   20
5.0           8     2    2   23    2  833   17    1   22    8
6.0           6     4    7    1    8   11  911    0    4    0
7.0           1     4    9   10    1    1    0  960    8    4
8.0           3     8   21    7    0    9    3    1  890    4
9.0           2     0    1    5   26   10    1   25   15  954
```

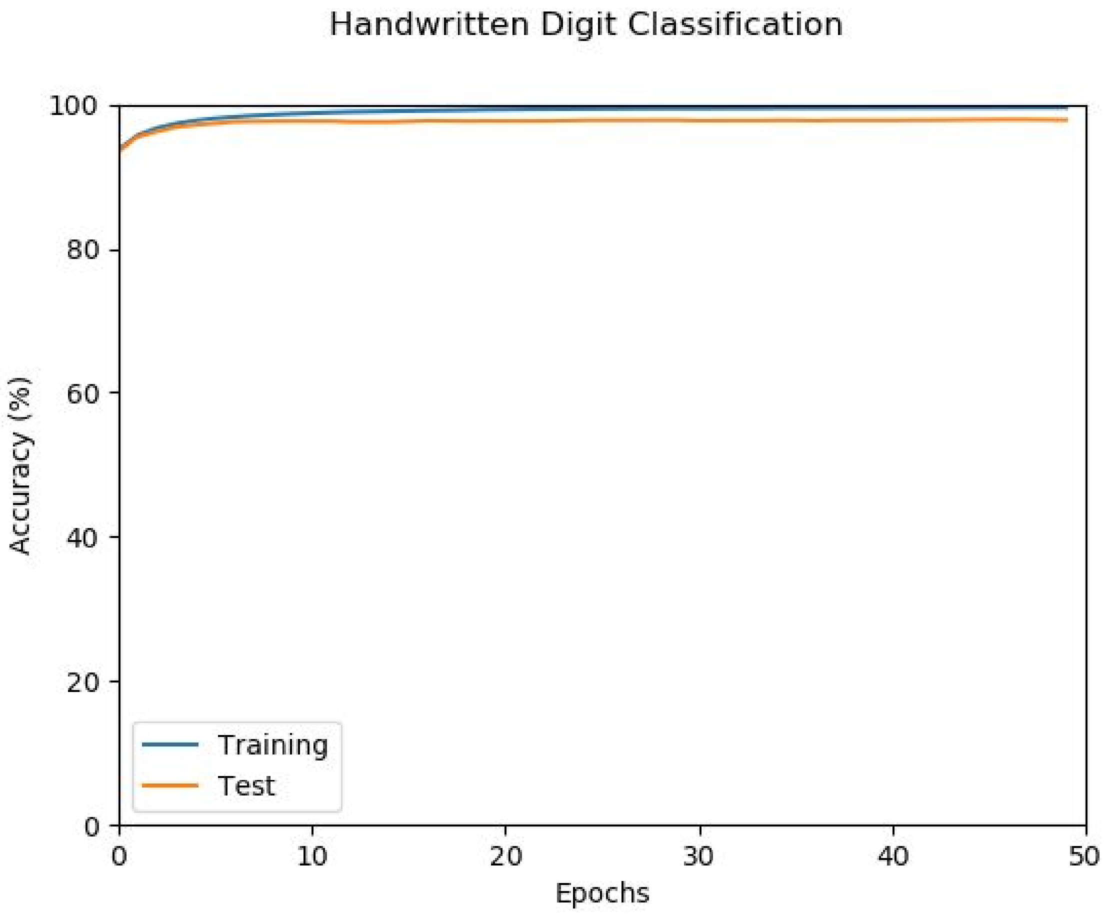
Fifty Hidden Units

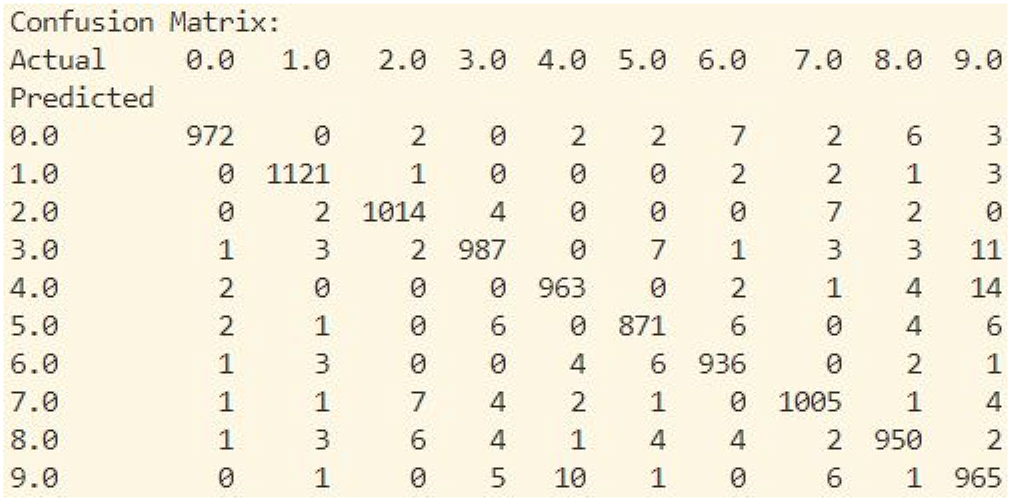




```
Confusion Matrix:
Actual       0.0   1.0   2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
Predicted
0.0          968     0     5    0    2    5    5    2    4    5
1.0            0  1117     0    0    0    2    1    5    1    5
2.0            1     4  1001   10    2    1    3   12    5    0
3.0            2     4     7  986    0   16    3    7    7    7
4.0            0     0     4    0  954    3    5    3    4   11
5.0            3     0     1    4    0  846    7    0    8    2
6.0            2     3     2    0    6    8  931    1    2    1
7.0            2     2     8    5    2    3    0  987    6    4
8.0            2     4     4    3    0    5    3    2  934    7
9.0            0     1     0    2   16    3    0    9    3  967
```

One Hundred Hidden Units





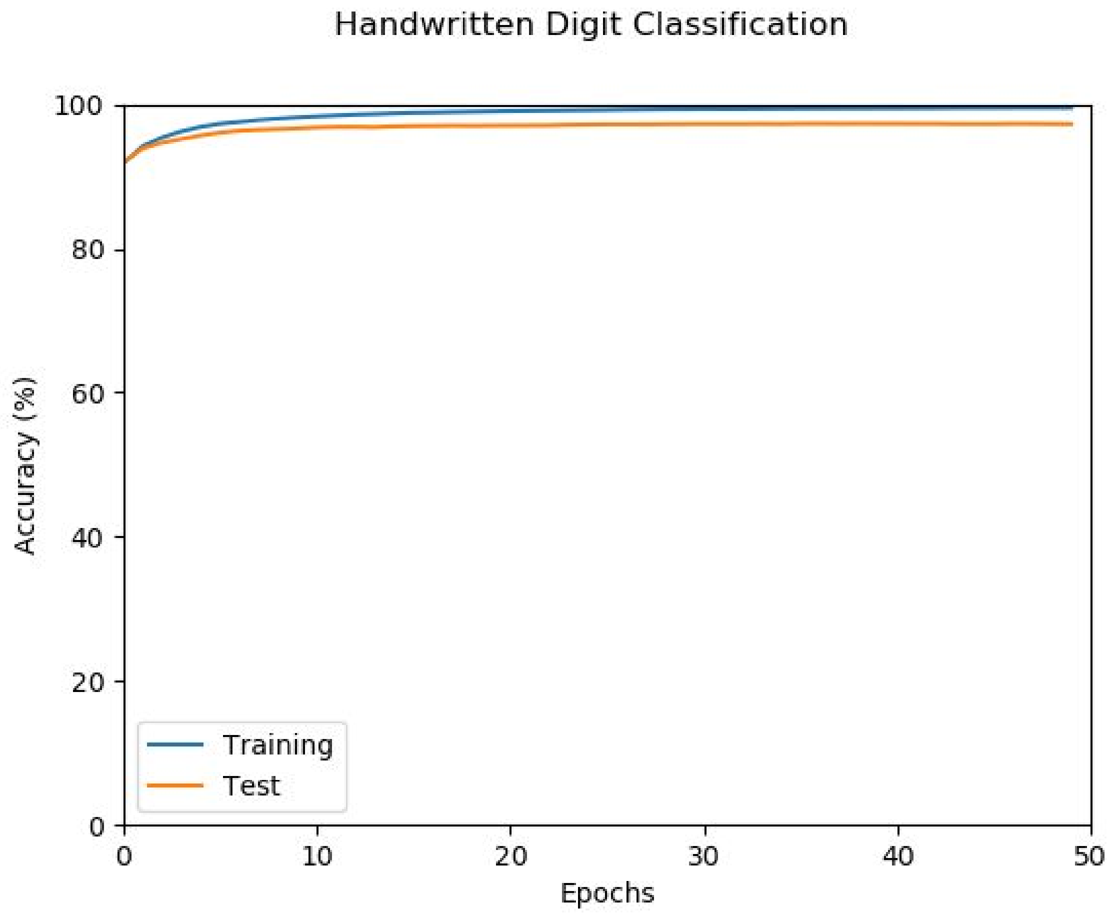
```
Confusion Matrix:
Actual      0.0   1.0   2.0  3.0  4.0  5.0  6.0   7.0  8.0  9.0
Predicted
0.0         972     0     2    0    2    2    7     2    6    3
1.0           0  1121     1    0    0    0    2     2    1    3
2.0           0     2  1014    4    0    0    0     7    2    0
3.0           1     3     2  987    0    7    1     3    3   11
4.0           2     0     0    0  963    0    2     1    4   14
5.0           2     1     0    6    0  871    6     0    4    6
6.0           1     3     0    0    4    6  936     0    2    1
7.0           1     1     7    4    2    1    0  1005    1    4
8.0           1     3     6    4    1    4    4     2  950    2
9.0           0     1     0    5   10    1    0     6    1  965
```

Discussion

1. How does the number of the hidden units affect the final accuracy on the test data?
    a. It appears that the more hidden units added to the hidden layer, the higher the resulting training and test accuracy will be (aka a direct relationship between the quantity of hidden units and the classification accuracy).
2. How does it affect the number of epochs needed for training to converge?
    a. The jumps in accuracy decrease enormously past 10 epochs (regardless of the number of hidden units), meaning any impact the quantity of hidden units has on the time (number of epochs) for convergence is minimal. That said, the plots do show *minimal* reduction in the number of epochs required for convergence as the quantity of hidden units increase. Though this change appears to be a result of the relationship between accuracy and the number of hidden units. (So we could say the number of epochs to convergence appears to be inversely related to the number of hidden units in the hidden layer.)
3. Is there evidence that any of your networks have overfit to the training data? If so, what is the evidence?
    a. While it is hard to discern due to the limited precision of the plot, it is possible that there could be some overfitting, demonstrated in the latter half of the epochs. The training accuracies *appear* to diverge from the test accuracies, though whether or not this behavior is exacerbated by more or less hidden units is unknown. This would imply that the neural network is continuing to improve its classification on the training set, but not able to transfer this learning over to the test set. A stronger observation on this behavior could be made, given more epochs were run.
4. How do your results compare to the results obtained by your perceptron in Assignment 1?
    a. Much better accuracy as well as minor to no oscillations.

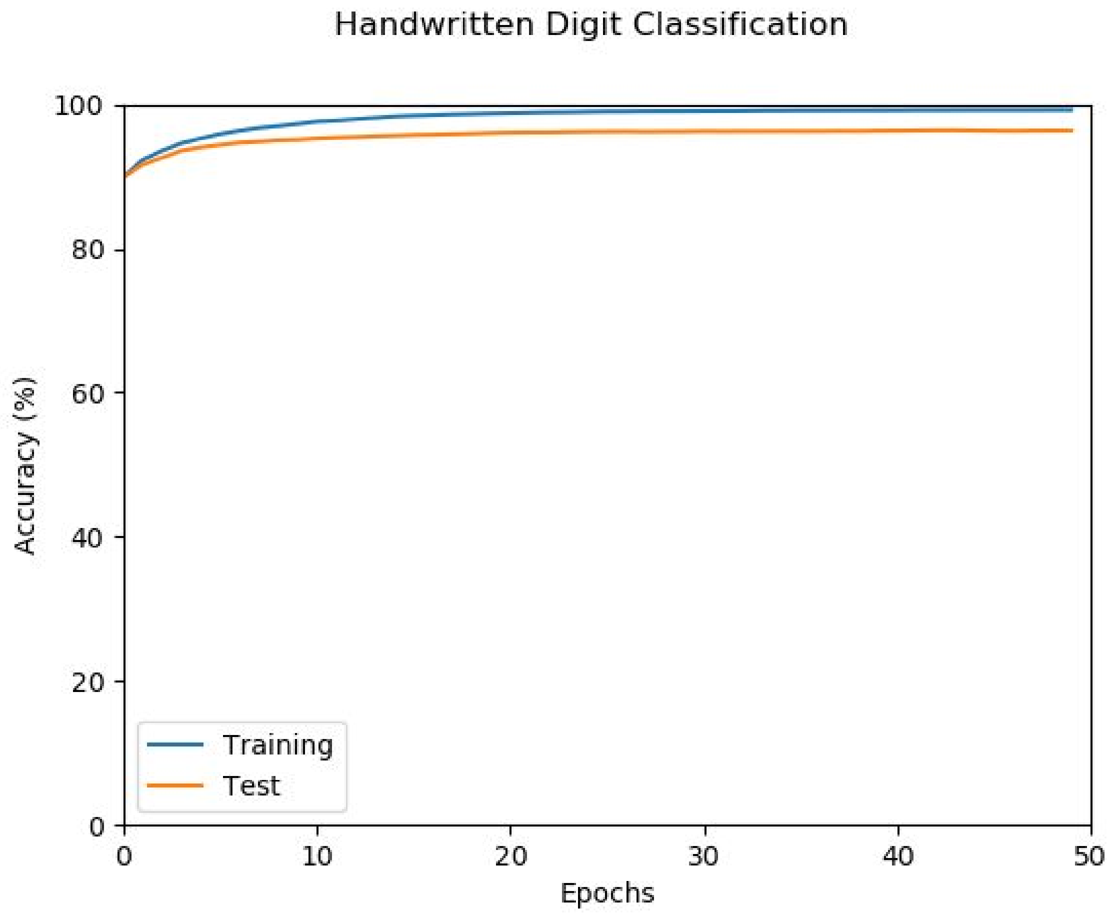
Experiment 2

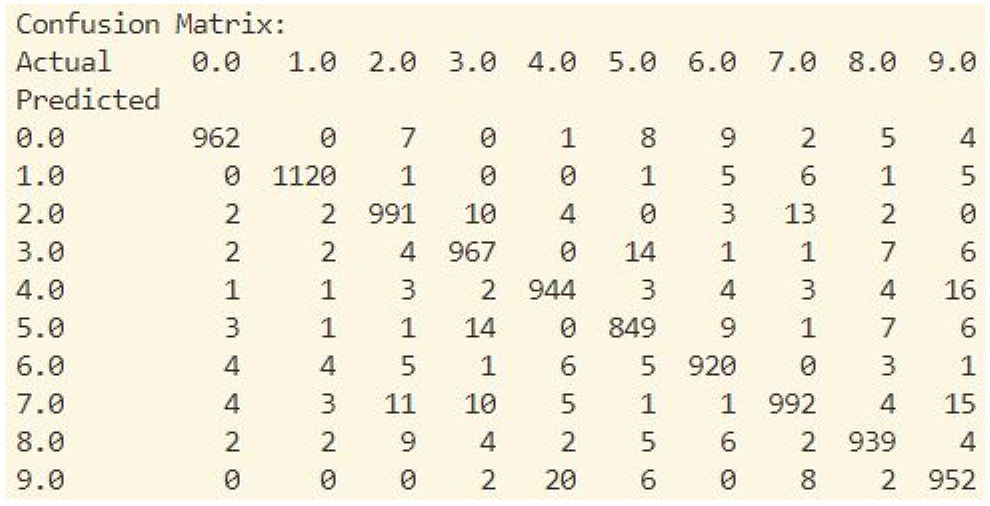
Half Training Examples



```
Confusion Matrix:
Actual     0.0   1.0   2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
Predicted
0.0        968     0     4    0    0    3    7    1    4    5
1.0          0  1121     1    0    0    0    2    6    0    3
2.0          1     3  1000   11    1    1    2   14    4    0
3.0          0     2     7  982    1    8    0    2    7    8
4.0          0     0     3    0  960    1    3    3    8   13
5.0          1     1     0    6    0  864    6    1    3    8
6.0          3     2     3    0    8    7  936    0    2    1
7.0          4     1     7    5    1    1    0  991    3    5
8.0          2     5     7    2    2    6    2    2  941    4
9.0          1     0     0    4    9    1    0    8    2  962
```

Quarter Training Examples





```
Confusion Matrix:
Actual      0.0   1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0
Predicted
0.0         962     0    7    0    1    8    9    2    5    4
1.0           0  1120    1    0    0    1    5    6    1    5
2.0           2     2  991   10    4    0    3   13    2    0
3.0           2     2    4  967    0   14    1    1    7    6
4.0           1     1    3    2  944    3    4    3    4   16
5.0           3     1    1   14    0  849    9    1    7    6
6.0           4     4    5    1    6    5  920    0    3    1
7.0           4     3   11   10    5    1    1  992    4   15
8.0           2     2    9    4    2    5    6    2  939    4
9.0           0     0    0    2   20    6    0    8    2  952
```

Discussion

1. How does the size of the training data affect the final accuracy on the test data?
    a. It appears that the larger the training data, the higher the final accuracy on both the test and training data. Consequently, the smaller the training data the lower the final accuracy.
2. How does it affect the number of epochs needed for training to converge?
    a. The number of epochs necessary for training to converge is observably larger for the smaller ‘quarter’ training set compared to the larger ‘half’ training set. Seeing as the smaller dataset reached a reasonable accuracy much more quickly than the ‘half’ and ‘all’ datasets, this experiment demonstrates how a reasonable accuracy can be reached in a relatively short amount of time given a smaller dataset, presuming time is in short supply. Though it can also be said that the final accuracy of neural network given a much smaller dataset will likely be lower than one given a larger dataset.
3. Is there evidence that any of your networks have overfit to the training data? If so, what is the evidence?
    a. In the example where only a quarter of the dataset is given to the neural network as inputs, the test accuracy appears to flatten at roughly twenty epochs. Meanwhile the training accuracy continues to grow steadily until it reaches fifty epochs.