Homework 2: Neural Networks

Joe Arriaga
CS 445: Machine Learning

March 9, 2019

1 Description

A two-layer neural network was implemented using the Python programming language to identify handwritten digits from the MNIST dataset. The network had ten output units, corresponding to the ten digits (0-9). All hidden and output units used the sigmoid activation function, all weights in the network were initialized to random values $[-0.5, 0.5)$, and the network was trained for fifty epochs using back-propagation with stochastic gradient descent with a learning rate of $\eta = 0.1$.

A "baseline" performance was acquired by running the neural network with 100 hidden units (plus a bias unit), a momentum value of $\alpha = 0.9$, and training on a set of 60,000 training examples. This baseline performance could then be used to evaluate the performance of the network in a series of experiments using 10 hidden units, 20 hidden units, momentum values of $\alpha = 0.0$, $\alpha = 0.5$, $\alpha = 1.0$, training on a set of 30,000 examples, and training on a set of 15,000 examples. In each experiment, the "baseline" values were used for all characteristics of the neural network except the characteristic being investigated. Some characteristics, such as the learning rate and the number of training epochs, remained constant across all experiments. Initial weights were randomized in the range $[-0.5, 0.5)$ for each experiment which is believed to reduce the chance of the initial weights favoring any particular configuration of the network and thus avoid a source of bias despite the values not being constant across all experiments.

2 Results

2.1 Varying the number of hidden units

Three networks were tested which had varying numbers of hidden units while all other parameters were the same. The networks had 10, 20, and 100 hidden units respectively, a learning rate of $\eta = 0.1$, a momentum value of $\alpha = 0.9$, and were trained for 50 epochs over 60,000 examples using backpropagation with stochastic gradient descent.

Increasing the number of hidden units in the network resulted in only a slight, but unmistakable, improvement in accuracy. With 10 hidden units the network achieved a final accuracy on the test set of 91%, with 20 hidden units it achieved 93%, and with 100 hidden units it achieved 97% accuracy. This pattern indicates that having more hidden units results in better accuracy, however it also suggests a trend of diminishing returns since increasing the number of hidden units by a factor of 2 (from 10 to 20) yielded a 2% improvement while increasing the number of hidden units by a factor of 10 (from 10 to 100) yielded only a 6% improvement. This trend is largely expected but is nevertheless an important reminder that at some point the computing costs of more hidden units will not be worth the marginal increase in accuracy. A disadvantage of this experiment's methodology was that the initial accuracies were so high that there was little improvement to be gained when increasing the number of hidden units. Applying a network to a more difficult problem that provided such room for improvement might cast a different light and show more clearly the potential gains of increasing the number of hidden units.

The number of hidden units did not significantly affect the number of epochs necessary for training to converge. With 10 hidden units the network was within 3% of its final accuracy after the first epoch but oscillated until after 5 epochs it would always be within 1% of its final accuracy. With 20 hidden units the network was immediately within 1% of its final accuracy after the first epoch and remained above that threshold throughout training. With 100 hidden units the network took 2 epochs to reach an accuracy within 1% of its final accuracy and remained above that threshold for the remainder of training. The network with the fewest hidden units took the longest to converge but all three neural networks made the majority of their improvements on the very first epoch.

It does not appear that any of the networks overfit to the training data since the accuracy since the accuracy on the test data was not significantly worse than the accuracy on the training data. For all three networks the accuracy on the training data was always less than 3% better than the accuracy on the test data.

2.1.1 100 hidden units

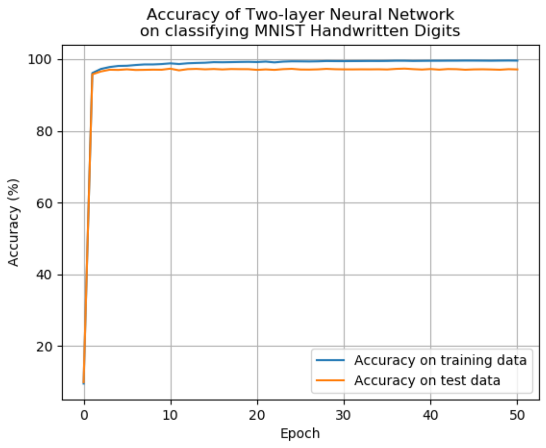
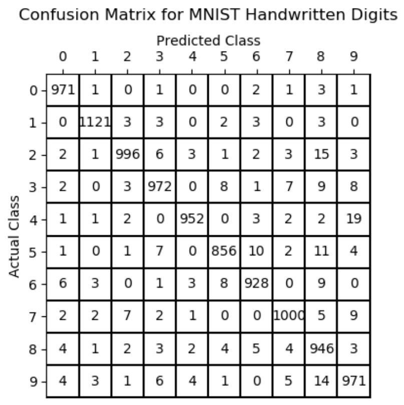
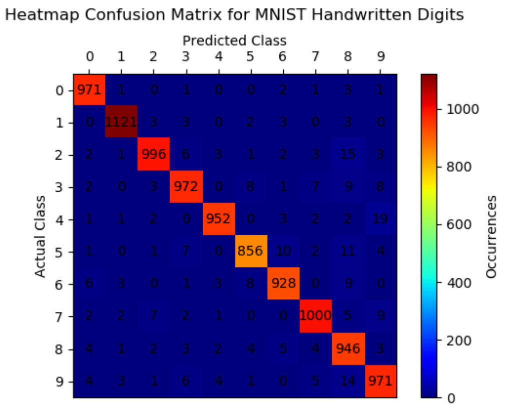


Figure 1: Accuracy with 100 hidden units


Confusion Matrix for MNIST Handwritten Digits

Actual Class \ Predicted Class	0	1	2	3	4	5	6	7	8	9
0	971	1	0	1	0	0	2	1	3	1
1	0	1121	3	3	0	2	3	0	3	0
2	2	1	996	6	3	1	2	3	15	3
3	2	0	3	972	0	8	1	7	9	8
4	1	1	2	0	952	0	3	2	2	19
5	1	0	1	7	0	856	10	2	11	4
6	6	3	0	1	3	8	928	0	9	0
7	2	2	7	2	1	0	0	1000	5	9
8	4	1	2	3	2	4	5	4	946	3
9	4	3	1	6	4	1	0	5	14	971

(a)



(b)

Figure 2: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set for network with 100 hidden units

2.1.2 20 hidden units

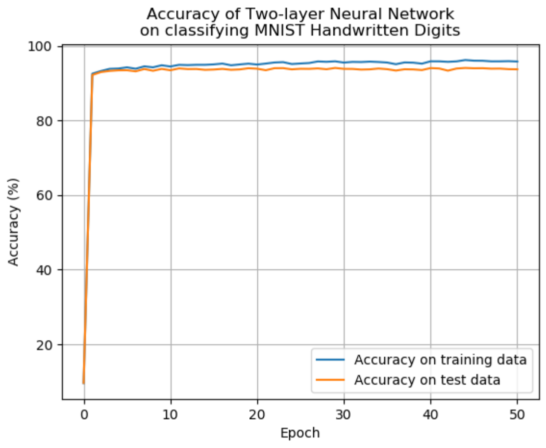
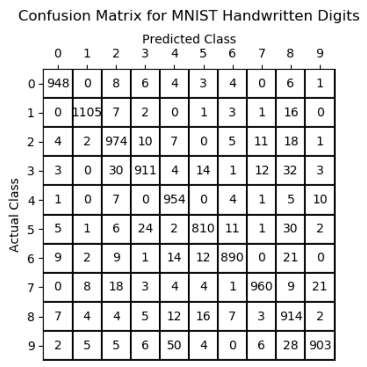
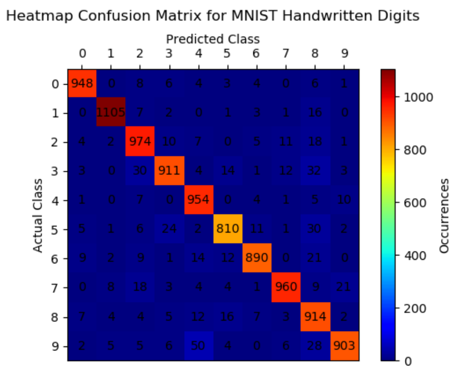


Figure 3: Accuracy with 20 hidden units


Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class

	0	1	2	3	4	5	6	7	8	9
0	948	0	8	6	4	3	4	0	6	1
1	0	1105	7	2	0	1	3	1	16	0
2	4	2	974	10	7	0	5	11	18	1
3	3	0	30	911	4	14	1	12	32	3
4	1	0	7	0	954	0	4	1	5	10
5	5	1	6	24	2	810	11	1	30	2
6	9	2	9	1	14	12	890	0	21	0
7	0	8	18	3	4	4	1	960	9	21
8	7	4	4	5	12	16	7	3	914	2
9	2	5	5	6	50	4	0	6	28	903

(a)



(b)

Figure 4: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set for network with 20 hidden units

2.1.3 10 hidden units

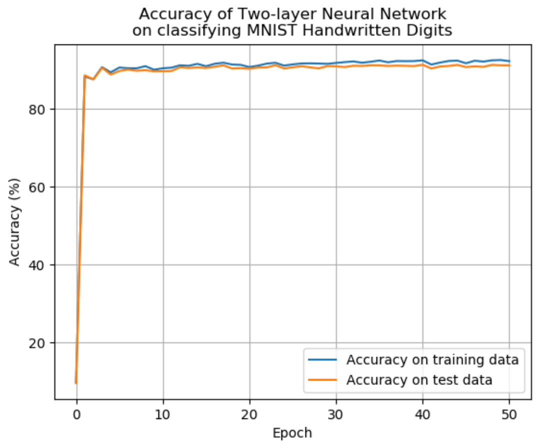
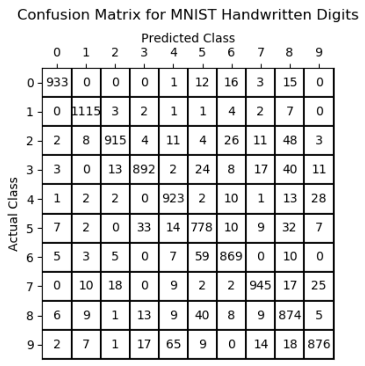
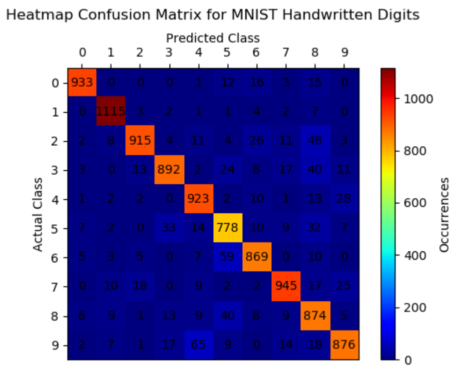


Figure 5: Accuracy with 10 hidden units


Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class

	0	1	2	3	4	5	6	7	8	9
0	933	0	0	0	1	12	16	3	15	0
1	0	1115	3	2	1	1	4	2	7	0
2	2	8	915	4	11	4	26	11	48	3
3	3	0	13	892	2	24	8	17	40	11
4	1	2	2	0	923	2	10	1	13	28
5	7	2	0	33	14	778	10	9	32	7
6	5	3	5	0	7	59	869	0	10	0
7	0	10	18	0	9	2	2	945	17	25
8	6	9	1	13	9	40	8	9	874	5
9	2	7	1	17	65	9	0	14	18	876

(a)



(b)

Figure 6: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set for network with 10 hidden units

The neural networks all performed better than the system of perceptrons which achieved an accuracy of 87% on the same training set and test set. The increase in accuracy from the system of perceptrons to the neural network with 10 hidden units was roughly equivalent to the increases in accuracy between the different neural networks which seems to indicate that the perceptron and the neural network belong in the same class of computational power. That is to say that a neural network with fewer hidden units, perhaps 5, would achieve the same accuracy as the system of perceptrons. A notable difference, however, was that the accuracies produced by each neural network were quite consistent throughout across epochs and only varied about 3%, but the accuracies produced by the system of perceptrons varied quite frequently, almost oscillating within the 5% range of accuracies.

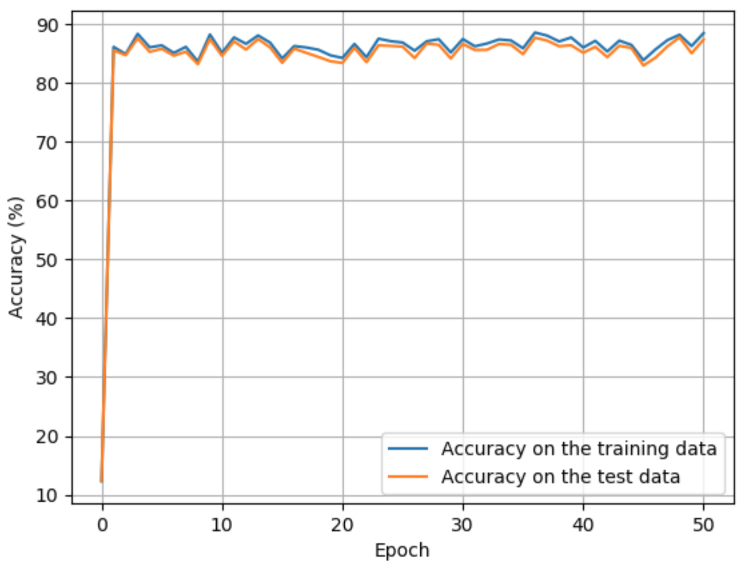


Figure 7: Accuracy of a system of ten perceptrons classifying MNIST handwritten digits

2.2 Varying the momentum value

Three networks were tested using three different momentum values while all other parameters were the same. The networks had a momentum value of $\alpha = 0.0$, $\alpha = 0.5$, and $\alpha = 0.1$ respectively. All three networks had 100 hidden units, a learning rate of $\eta = 0.1$, and were trained for 50 epochs over 60,000 examples using backpropagation with stochastic gradient descent. These networks can also be compared to the network from the previous section which had a momentum value of $\alpha = 0.9$ and matched all other parameters.

Varying the momentum value does not significantly affect the final accuracy of the neural network except when $\alpha = 1.0$. For all values except 1.0 the networks achieved a final accuracy of $97\% \pm 0.4\%$. For $a\alpha = 1.0$ the network was never able to perform better than random chance. It achieved only 9.8% accuracy on the test set whereas we would expect randomly selecting one of the ten possibilities for each example to yield a 10% accuracy. This slight deficiency is intriguing but may be due to something as banal as the distribution of the test set or some inherent bias of the initial random weights to predict a particular number.

The momentum value did not have any significant effect on the number of epochs needed for training converge. A pattern did emerge that the greater the momentum value the more quickly training to converged but the results from the different values are so similar and had such little impact on the performance of the networks that it is unlikely to be significant.

α	0.0	0.5	0.9	1.0
epochs	6	3	2	1

Table 1: The epochs needed until accuracy was within 1% of the final accuracy for each momentum value α

Again, there is no evidence that any of the models overfit to the training data. In all three networks the accuracy on the test set was very similar to the accuracy on the training set indicating that the networks were able to generalize well. Even when $\alpha = 1.0$ and the network did not learn at all it still generalized that training to the test set and did equally terribly!

It is surprising that all values except 1.0 produced similar results while a value of 1.0 produced such a wildly different outcome. One might expect that a value of 0.0 at the other extreme might also produce dramatic results, although perhaps of a different nature. One might also expect the results to change gradually along with the values. That is, that a value of 0.1 would be similar to 0.0, a value of 0.9 would be similar to 1.0, and a value of 0.5 would be some combination or compromise between the two extremes. The reason this behavior was not observed may be attributable to the power of the 100 hidden units. It is possible that with 100 hidden units the network was able to compensate for any deficiencies of learning produced by the momentum term. However, this is insufficient to explain why a value of 1.0 would produce such drastically poor results and a value of 0.9 would not since there is very little mathematical difference between the two.

2.2.1 $\alpha = 0.0$

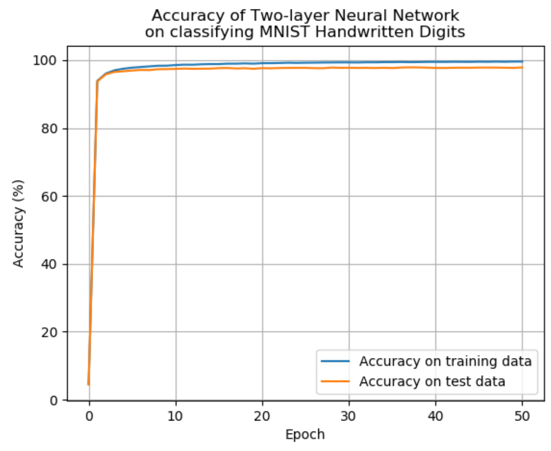
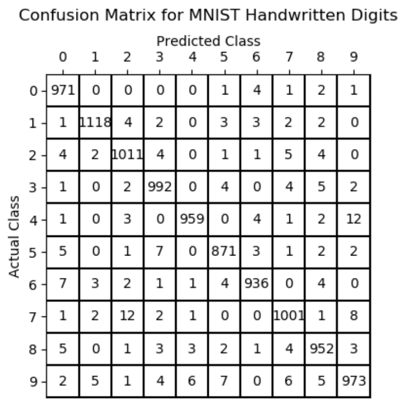
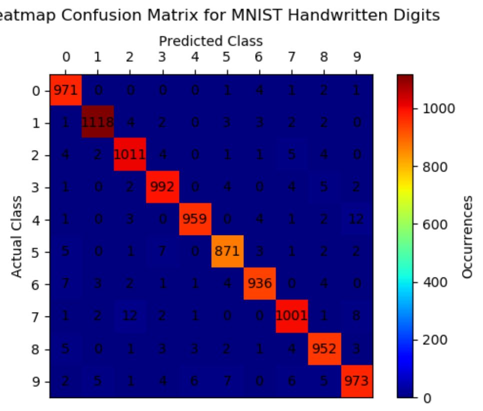


Figure 8: Accuracy with $\alpha = 0.0$


Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class

	0	1	2	3	4	5	6	7	8	9
0	971	0	0	0	0	1	4	1	2	1
1	1	1118	4	2	0	3	3	2	2	0
2	4	2	1011	4	0	1	1	5	4	0
3	1	0	2	992	0	4	0	4	5	2
4	1	0	3	0	959	0	4	1	2	12
5	5	0	1	7	0	871	3	1	2	2
6	7	3	2	1	1	4	936	0	4	0
7	1	2	12	2	1	0	0	1001	1	8
8	5	0	1	3	3	2	1	4	952	3
9	2	5	1	4	6	7	0	6	5	973

(a)



(b)

Figure 9: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set with $\alpha = 0.0$

2.2.2 $\alpha = 0.5$

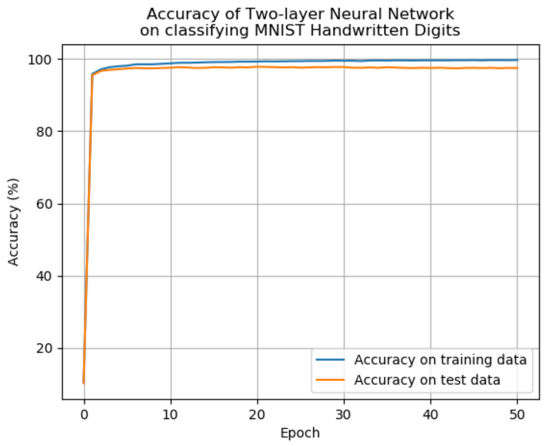
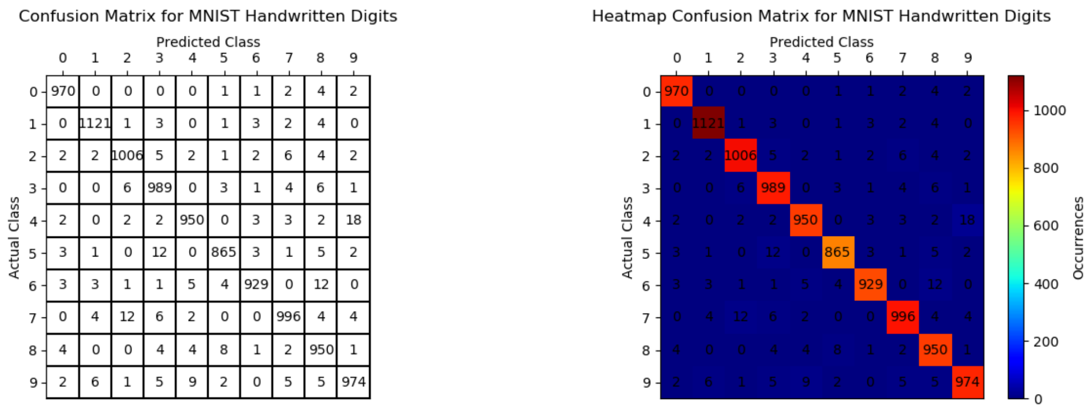


Figure 10: Accuracy with $\alpha = 0.5$


Confusion Matrix for MNIST Handwritten Digits
Heatmap Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class
Occurrences

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	970	0	0	0	0	1	1	2	4	2
1	0	1121	1	3	0	1	3	2	4	0
2	2	2	1006	5	2	1	2	6	4	2
3	0	0	6	989	0	3	1	4	6	1
4	2	0	2	2	950	0	3	3	2	18
5	3	1	0	12	0	865	3	1	5	2
6	3	3	1	1	5	4	929	0	12	0
7	0	4	12	6	2	0	0	996	4	4
8	4	0	0	4	4	8	1	2	950	1
9	2	6	1	5	9	2	0	5	5	974

(a) (b)

Figure 11: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set with $\alpha = 0.5$

2.2.3 $\alpha = 1.0$

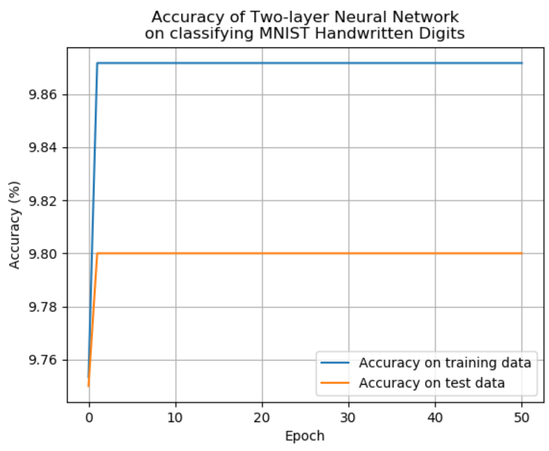
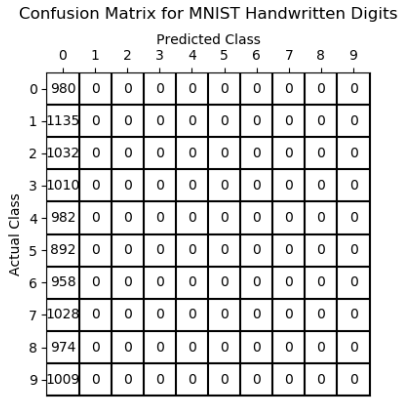
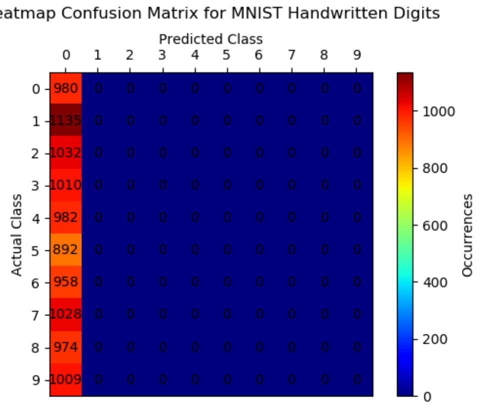


Figure 12: Accuracy with $\alpha = 1.0$


Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	980	0	0	0	0	0	0	0	0	0
1	1135	0	0	0	0	0	0	0	0	0
2	1032	0	0	0	0	0	0	0	0	0
3	1010	0	0	0	0	0	0	0	0	0
4	982	0	0	0	0	0	0	0	0	0
5	892	0	0	0	0	0	0	0	0	0
6	958	0	0	0	0	0	0	0	0	0
7	1028	0	0	0	0	0	0	0	0	0
8	974	0	0	0	0	0	0	0	0	0
9	1009	0	0	0	0	0	0	0	0	0

(a)



(b)

Figure 13: (a) Confusion matrix and (b) Heatmap of confusion matrix on test set with $\alpha = 1.0$

2.3 Varying the number of training examples

Two new networks were tested which used $\frac{1}{4}$ of the training set (15,000 examples) and $\frac{1}{2}$ of the training set (30,000 examples), and these can be compared to the network from the first section which used the full training set (60,000 examples). All three networks had 100 hidden units, a learning rate of $\eta = 0.1$, a momentum value of $\alpha = 0.9$, and were trained for 50 epochs using backpropagation with stochastic gradient descent.

Both reduced training sets had an equal distribution of all 10 digits. The 15,000 example set was produced by taking the first 15,000 examples from the full set. Since there were 10 digits, each digit should be represented by 1,500 examples. If a digit had more examples in the set, examples were deleted from the beginning of the set until the digit had 1,500 instances. If a digit had fewer examples in the set, additional examples were taken from the end of the full set to avoid duplicates and added to the reduced set until the digit had 1,500 instances. This methodology may have resulted in sequences of examples of the same digit which could bias a machine towards learning that digit at the expense of being able to generalize to other digits, but the order of examples from the data file are shuffled at the beginning of each epoch, resulting in a random ordering of examples for each epoch, and nullifying any effect of the original ordering of examples. An improvement might be to inspect the ordering for each epoch and adjust it so that examples of the same digit were never consecutive, but it is unlikely that there would be any improvement in performance despite the large amount of work necessary to inspect each and every example.

These experiments suggest that a larger training set produces a greater final accuracy on the test data but more investigation is required to reach a definitive conclusion. All three neural networks achieved at least 95% accuracy and the range across the three networks was less than 2%.

examples	60,000	30,000	15,000
accuracy (%)	97	96	95.6

Table 2: Final accuracies on the test set for training sets with varying numbers of examples

The pattern does suggest that a larger training set produces a greater final accuracy but because all three networks achieved such a high final accuracy and because the differences between them were so slight it is difficult to attribute the differences to the difference in training set size; it is possible that differences in initial random weights or example order may have contributed to the differences in final accuracy.

Three methodologies for further investigation are apparent:

1. Run the experiment many more times to perform a statistical analysis which would rule out the effects of random circumstance.
2. Run more experiments further reducing the size of the training set until there is a significant drop in accuracy, perhaps even to the point that the network fails to predict better than random chance.
3. Test a harder problem such that the network achieves only a moderate accuracy with the full test set, maybe 70% - 80%
4. Combine 2 and 3

The problem with the methodology used here was that even with the smallest training set the network still performed very well - too well! The difference in accuracy between the full training set and the smallest training set was not large enough to make a definitive conclusion. Because the smallest training set performed so well it did not allow any room for the larger training sets to show significant improvement.

The intention of the proposed methodologies is to create a significant disparity such that the full training set performs relatively well while the smallest training set largely fails. The intermediate set sizes can then be used to determine the relationship between set size and accuracy (linear, logarithmic, exponential, etc.) and perhaps even identify any inflection points where a small increase in set size results in a large increase in accuracy.

The size of the training set had no effect on the number of epochs needed for training to converge. In all three experiments the networks achieved an accuracy within 1% of their final accuracies within three epochs and remained within 1% for the remainder of training.

There is some evidence suggesting that networks with smaller training sets have a greater tendency to overfit to the training data. In all three experiments the final accuracies on the training sets were above 99%, however, the smaller the training set the lower the final accuracy on the test set, as shown above in Table 2. Inspecting the accuracies on the test set for each epoch revealed that all three experiments had a similar pattern in which the accuracies dipped slightly in the last ten to fifteen epochs, indicating a loss of generality.

2.3.1 One-half the number of training examples

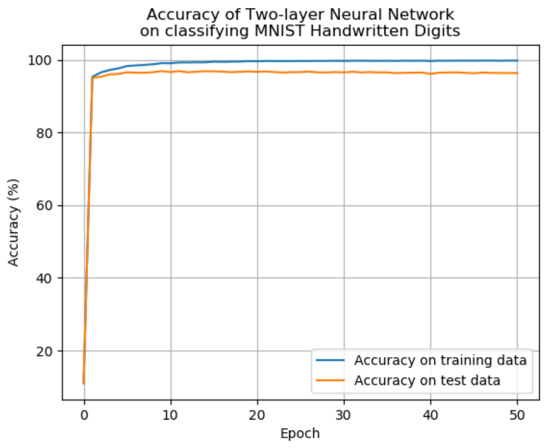
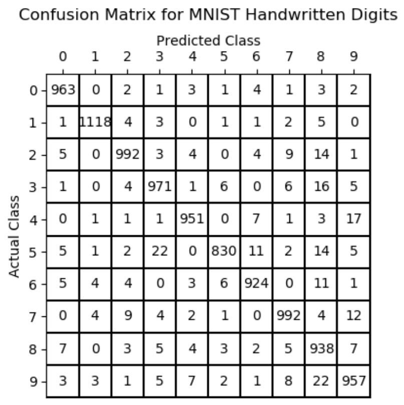


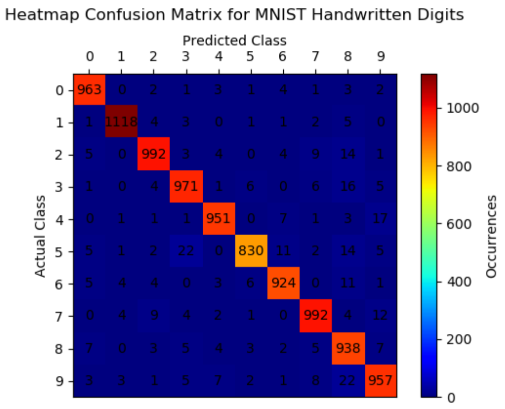
Figure 14: Accuracy after training on 30,000 examples


Confusion Matrix for MNIST Handwritten Digits

Predicted Class

Actual Class	0	1	2	3	4	5	6	7	8	9
0	963	0	2	1	3	1	4	1	3	2
1	1	1118	4	3	0	1	1	2	5	0
2	5	0	992	3	4	0	4	9	14	1
3	1	0	4	971	1	6	0	6	16	5
4	0	1	1	1	951	0	7	1	3	17
5	5	1	2	22	0	830	11	2	14	5
6	5	4	4	0	3	6	924	0	11	1
7	0	4	9	4	2	1	0	992	4	12
8	7	0	3	5	4	3	2	5	938	7
9	3	3	1	5	7	2	1	8	22	957

(a)



(b)

Figure 15: (a) Confusion matrix and (b) Heatmap of confusion matrix after training on 30,000 examples

2.3.2 One-quarter the number of training examples

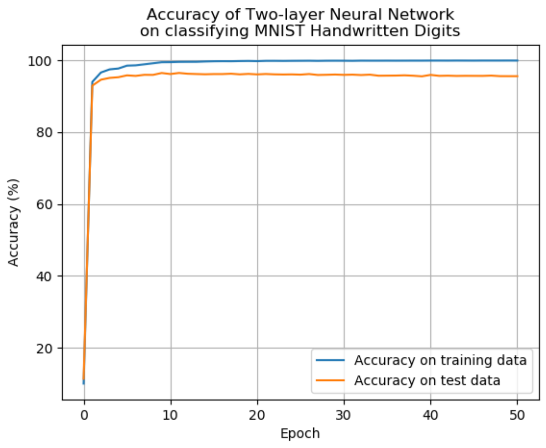
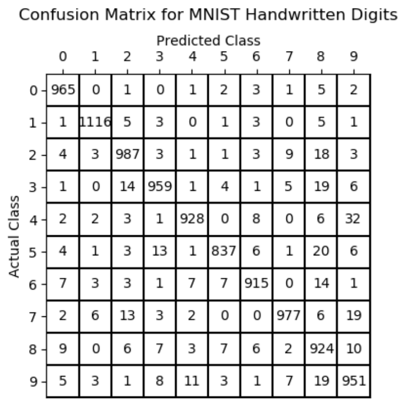
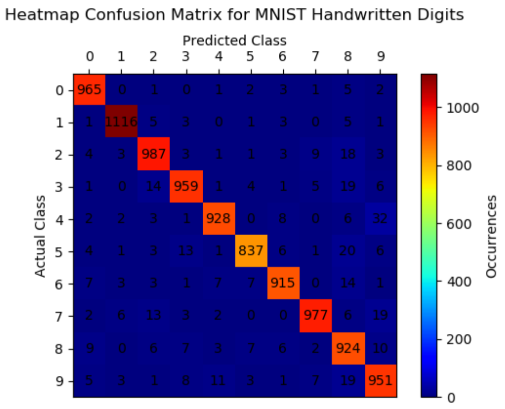


Figure 16: Accuracy after training on 15,000 examples


Confusion Matrix for MNIST Handwritten Digits
Predicted Class
Actual Class

	0	1	2	3	4	5	6	7	8	9
0	965	0	1	0	1	2	3	1	5	2
1	1	1116	5	3	0	1	3	0	5	1
2	4	3	987	3	1	1	3	9	18	3
3	1	0	14	959	1	4	1	5	19	6
4	2	2	3	1	928	0	8	0	6	32
5	4	1	3	13	1	837	6	1	20	6
6	7	3	3	1	7	7	915	0	14	1
7	2	6	13	3	2	0	0	977	6	19
8	9	0	6	7	3	7	6	2	924	10
9	5	3	1	8	11	3	1	7	19	951

(a)



(b)

Figure 17: (a) Confusion matrix and (b) Heatmap of confusion matrix after training on 15,000 examples

3 Conclusions

Increasing the number of hidden units led to a slight increase in accuracy but with diminishing returns; at some point the marginal increase in accuracy would not be worth the computational costs. Varying the momentum value seemed to have no appreciable effect as long as the value was less than one. Any value from 0.0 to 0.9 produced very similar results and only the extreme value of 1.0 produced different, and very poor, results. There was an indication that using a smaller training set reduced the final accuracy of the network but the resilience of the network and limitations of the experimental methodology did not allow for definitive evidence. The only indication that the network overfit to the training data in any of the experiments was with the smaller training sets; in all other cases the accuracy on the test set improved and declined along with the accuracy on the training set but was always slightly lower.