EECE6036 - Homework 4

Long Dang Vo

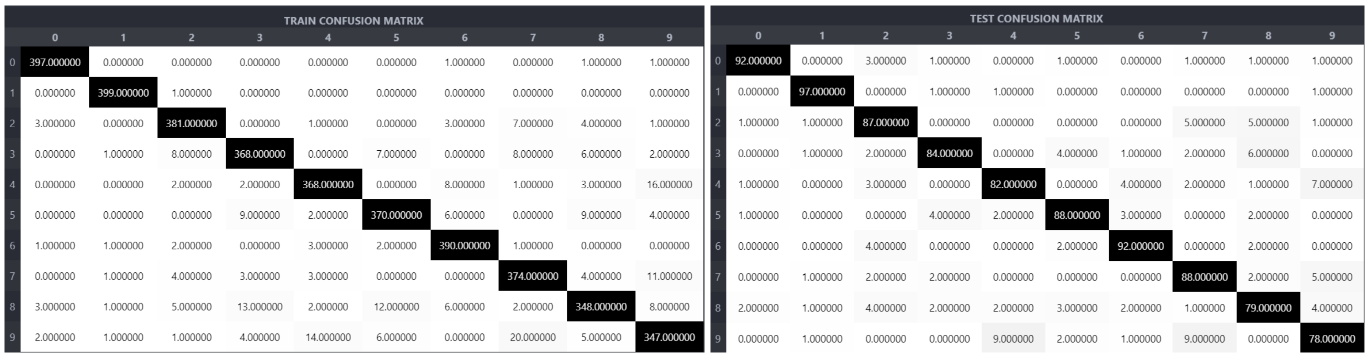
November 29th, 2022

1 Problem 1

1.1 System Description

This neural network consists of 784 input layers, each corresponding to a single pixel in a 28 x 28 MNIST dataset image, 180 hidden neurons, 10 output neurons, momentum (alpha) of 0.5, a high output threshold of 0.75, a low output threshold of 0.25, a learning rate of 0.01, and 800 epochs. Each of the 10 output neurons is assigned a unique number (from 0 - 9).
For case 1, I assigned the weight of input to hidden neurons using the weight generated by the autoencoder; however, I did not update it during the process of backpropagation. For case 2, I set up the weight of input to hidden neurons similar to case 1, but at this time, I update its weight and bias during the process of backpropagation.

1.2 Results



TRAIN CONFUSION MATRIX

	0	1	2	3	4	5	6	7	8	9
0	397.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	1.000000
1	0.000000	399.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	3.000000	0.000000	381.000000	0.000000	1.000000	0.000000	3.000000	7.000000	4.000000	1.000000
3	0.000000	1.000000	8.000000	368.000000	0.000000	7.000000	0.000000	8.000000	6.000000	2.000000
4	0.000000	0.000000	2.000000	2.000000	368.000000	0.000000	8.000000	1.000000	3.000000	16.000000
5	0.000000	0.000000	0.000000	9.000000	2.000000	370.000000	6.000000	0.000000	9.000000	4.000000
6	1.000000	1.000000	2.000000	0.000000	3.000000	2.000000	390.000000	1.000000	0.000000	0.000000
7	0.000000	1.000000	4.000000	3.000000	3.000000	0.000000	0.000000	374.000000	4.000000	11.000000
8	3.000000	1.000000	5.000000	13.000000	2.000000	12.000000	6.000000	2.000000	348.000000	8.000000
9	2.000000	1.000000	1.000000	4.000000	14.000000	6.000000	0.000000	20.000000	5.000000	347.000000

TEST CONFUSION MATRIX

	0	1	2	3	4	5	6	7	8	9
0	92.000000	0.000000	3.000000	1.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000
1	0.000000	97.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
2	1.000000	1.000000	87.000000	0.000000	0.000000	0.000000	0.000000	5.000000	5.000000	1.000000
3	0.000000	1.000000	2.000000	84.000000	0.000000	4.000000	1.000000	2.000000	6.000000	0.000000
4	1.000000	0.000000	3.000000	0.000000	82.000000	0.000000	4.000000	2.000000	1.000000	7.000000
5	1.000000	0.000000	0.000000	4.000000	2.000000	88.000000	3.000000	0.000000	2.000000	0.000000
6	0.000000	0.000000	4.000000	0.000000	0.000000	2.000000	92.000000	0.000000	2.000000	0.000000
7	0.000000	1.000000	2.000000	2.000000	0.000000	0.000000	0.000000	88.000000	2.000000	5.000000
8	2.000000	1.000000	4.000000	2.000000	2.000000	3.000000	2.000000	1.000000	79.000000	4.000000
9	0.000000	1.000000	0.000000	0.000000	9.000000	2.000000	1.000000	9.000000	0.000000	78.000000

Figure 1.1. Confusion Matrix for Training Set and Test Set (case 1)

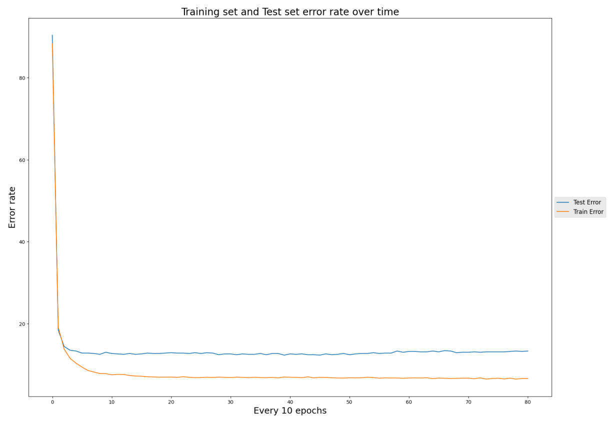


Figure 1.2. Training Set and Test Set error fraction rate over time (case 1)

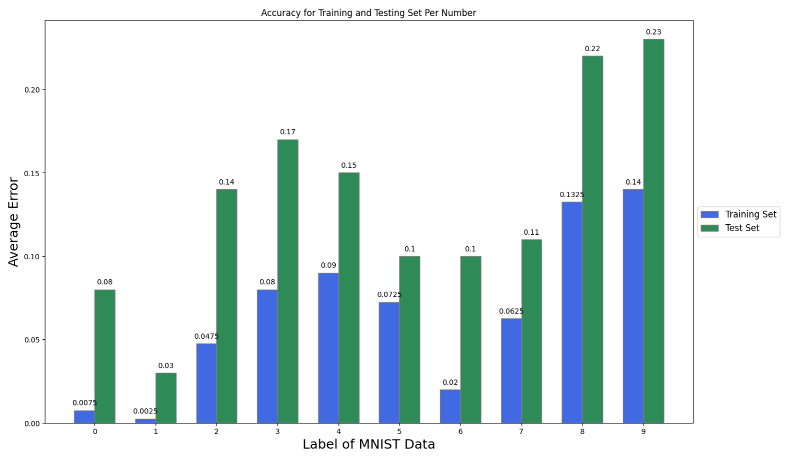


Figure 1.3. Average error fraction for each digit of the training set and test set (case 1)

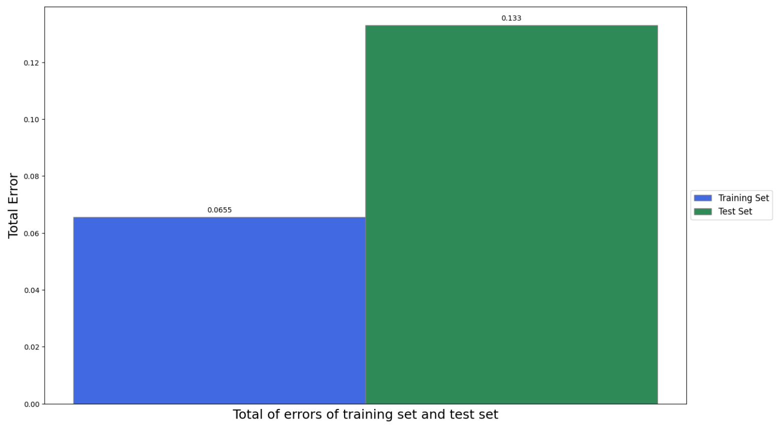
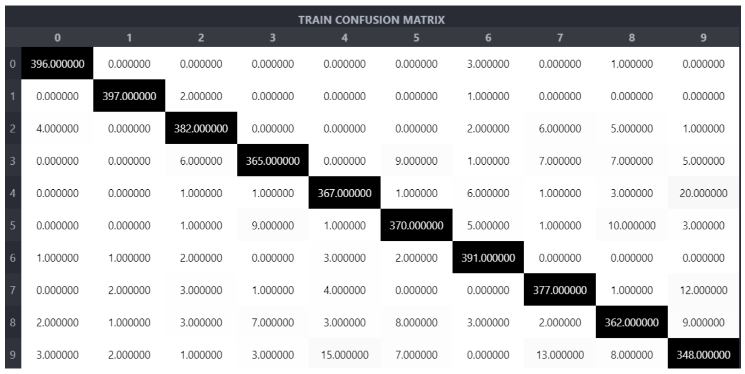
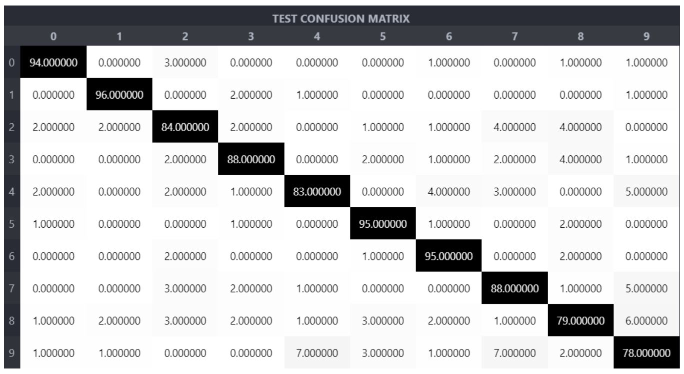


Figure 1.4. Total error fraction for training set and test set (case 1)



TRAIN CONFUSION MATRIX

	0	1	2	3	4	5	6	7	8	9
0	396.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.000000	0.000000	1.000000	0.000000
1	0.000000	397.000000	2.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000
2	4.000000	0.000000	382.000000	0.000000	0.000000	0.000000	2.000000	6.000000	5.000000	1.000000
3	0.000000	0.000000	6.000000	365.000000	0.000000	9.000000	1.000000	7.000000	7.000000	5.000000
4	0.000000	0.000000	1.000000	1.000000	367.000000	1.000000	6.000000	1.000000	3.000000	20.000000
5	0.000000	0.000000	1.000000	9.000000	1.000000	370.000000	5.000000	1.000000	10.000000	3.000000
6	1.000000	1.000000	2.000000	0.000000	3.000000	2.000000	391.000000	0.000000	0.000000	0.000000
7	0.000000	2.000000	3.000000	1.000000	4.000000	0.000000	0.000000	377.000000	1.000000	12.000000
8	2.000000	1.000000	3.000000	7.000000	3.000000	8.000000	3.000000	2.000000	362.000000	9.000000
9	3.000000	2.000000	1.000000	3.000000	15.000000	7.000000	0.000000	13.000000	8.000000	348.000000



TEST CONFUSION MATRIX

	0	1	2	3	4	5	6	7	8	9
0	94.000000	0.000000	3.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	1.000000
1	0.000000	96.000000	0.000000	2.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
2	2.000000	2.000000	84.000000	2.000000	0.000000	1.000000	1.000000	4.000000	4.000000	0.000000
3	0.000000	0.000000	2.000000	88.000000	0.000000	2.000000	1.000000	2.000000	4.000000	1.000000
4	2.000000	0.000000	2.000000	1.000000	83.000000	0.000000	4.000000	3.000000	0.000000	5.000000
5	1.000000	0.000000	0.000000	1.000000	0.000000	95.000000	1.000000	0.000000	2.000000	0.000000
6	0.000000	0.000000	2.000000	0.000000	0.000000	1.000000	95.000000	0.000000	2.000000	0.000000
7	0.000000	0.000000	3.000000	2.000000	1.000000	0.000000	0.000000	88.000000	1.000000	5.000000
8	1.000000	2.000000	3.000000	2.000000	1.000000	3.000000	2.000000	1.000000	79.000000	6.000000
9	1.000000	1.000000	0.000000	0.000000	7.000000	3.000000	1.000000	7.000000	2.000000	78.000000

Figure 1.5: Training Set and Test Set Confusion Matrix (case 2)

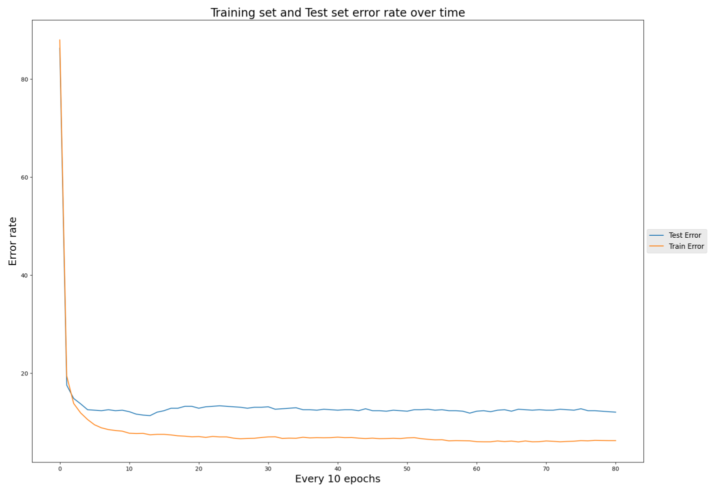


Figure 1.6. Training Set and Test Set error fraction rate over time (case 2)

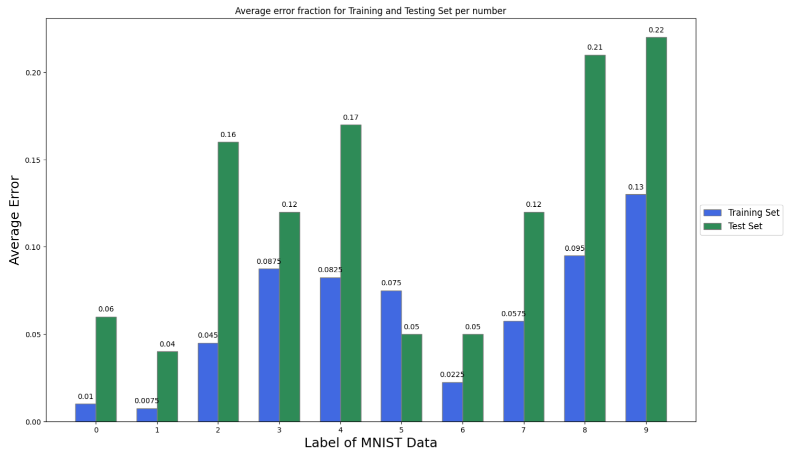


Figure 1.7. Average error fraction for each digit of the training set and test set (case 2)

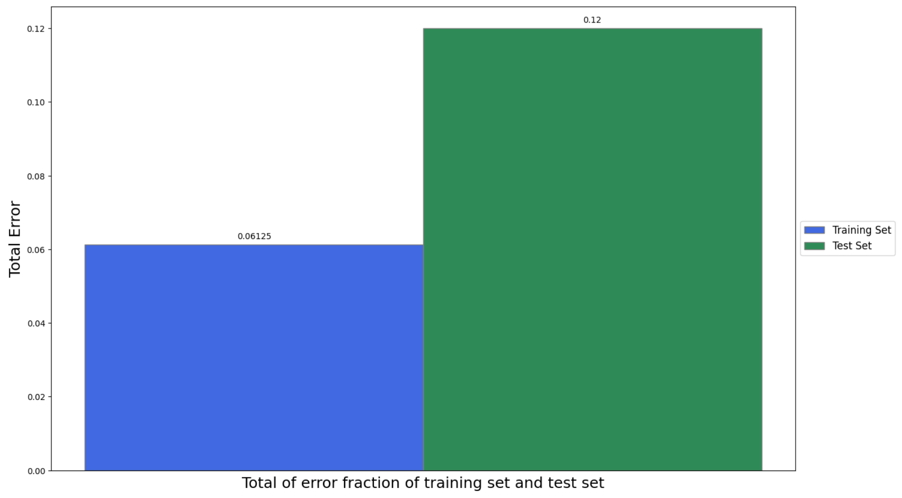


Figure 1.8. Total error fraction for training set and test set (case 2)

1.3 Analysis of Results

According to figures 1.1 and 1.5, by using the hidden weight generated by the autoencoder, the system worked faster toward getting the best result out of training because the weight was placed in a position that was closer to the actual output (the minimum loss) position that pushed the system in time proficiency. In other words, the autoencoder has rebuilt and encoded the input data to point it in the right direction. Furthermore, the autoencoder-generated weight produced a more accurate result than randomly generated weight. For example, in case 1, after training the accuracy result to approximately 94%, the system can classify the average 376/400 data point for each image digit, with the image of digit 1 having 99% accuracy (399/400); in case 2, after training the accuracy result to approximately 94%, the system can classify the average 376/400 data point for each image digit, with the image of digit 1 having 99% accuracy (397/400). In comparison to the randomly weighted system, the result with the image of digit 2 has an accuracy of 98% (393/400).

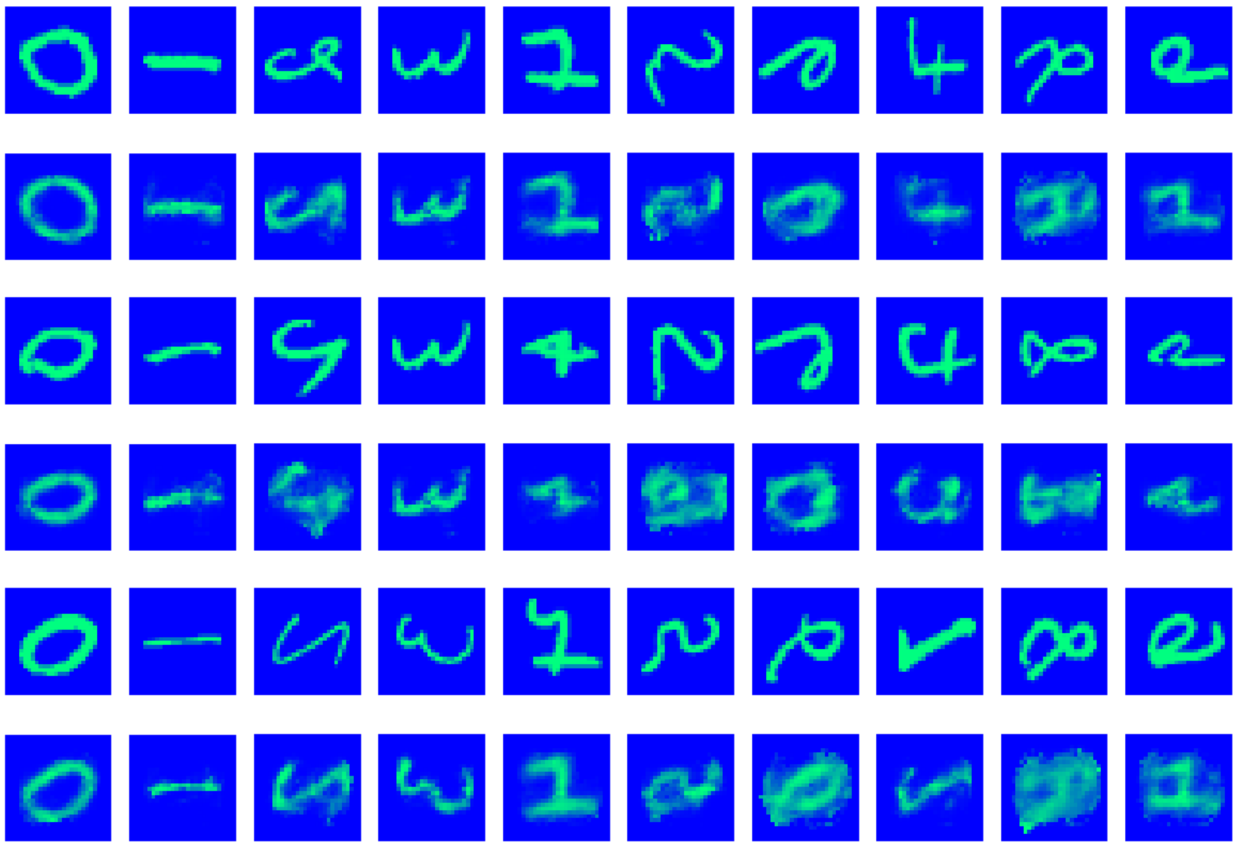
Figures 1.2 and 1.6 show that the error fraction rate drops significantly after 10 epochs, then has a small fluctuation toward 0, whereas with a randomly generated weight system, there were many fluctuations and the system required at least 100 epochs to achieve a good result. In comparison between case 1 and case 2, the results were not significantly different (training set around 94% for both cases, while test set around 87% for case 1 and 88% for case 2) because the autoencoder did a great job reconstructing the weight in a location closer to the minimum loss, and whether we updated the hidden weight during backpropagation did not affect much of the result, as shown in figures 1.4 and 1.8. Overall, the vast majority of data points had accurate classifications. The numbers that were incorrectly identified can be explained by the fact that, when considering some of the digits, they will have a similar draw to one another. For instance, 7 and 1 can occasionally be misunderstood if we write them too quickly.

2 Problem 2

2.1 System Description

This autoencoder consists of 784 input layers, each corresponding to a single pixel in a 28 x 28 MNIST dataset image, 180 hidden neurons, 784 output neurons, a momentum (alpha) of 0.5, a high output threshold of 0.75, a low output threshold of 0.25, a learning rate of 0.01, and 1100 epochs. The output will be the same as the input layers.
I use the same neural network system for the autoencoder to ensure that the autoencoder's goal is to compress the 784 input neurons to 180 hidden neurons and then have the same output as the input, especially when compressing and decompressing the data value. The change for the system in the autoencoder would be to set the bias with the value of 1 and unchange for the whole system which significantly decreases the error rate when training. However, at this time, the input data for training will only be the images of the digits 0 to 4.

2.2 Results



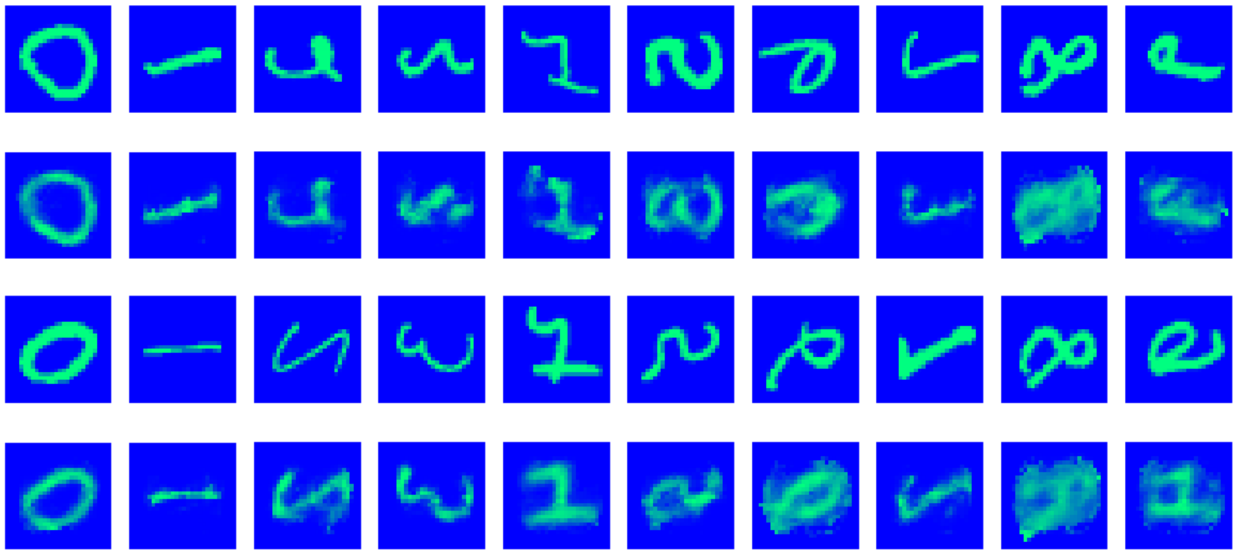


Figure 2.1. Sample output of the training set and the test set vs the actual output

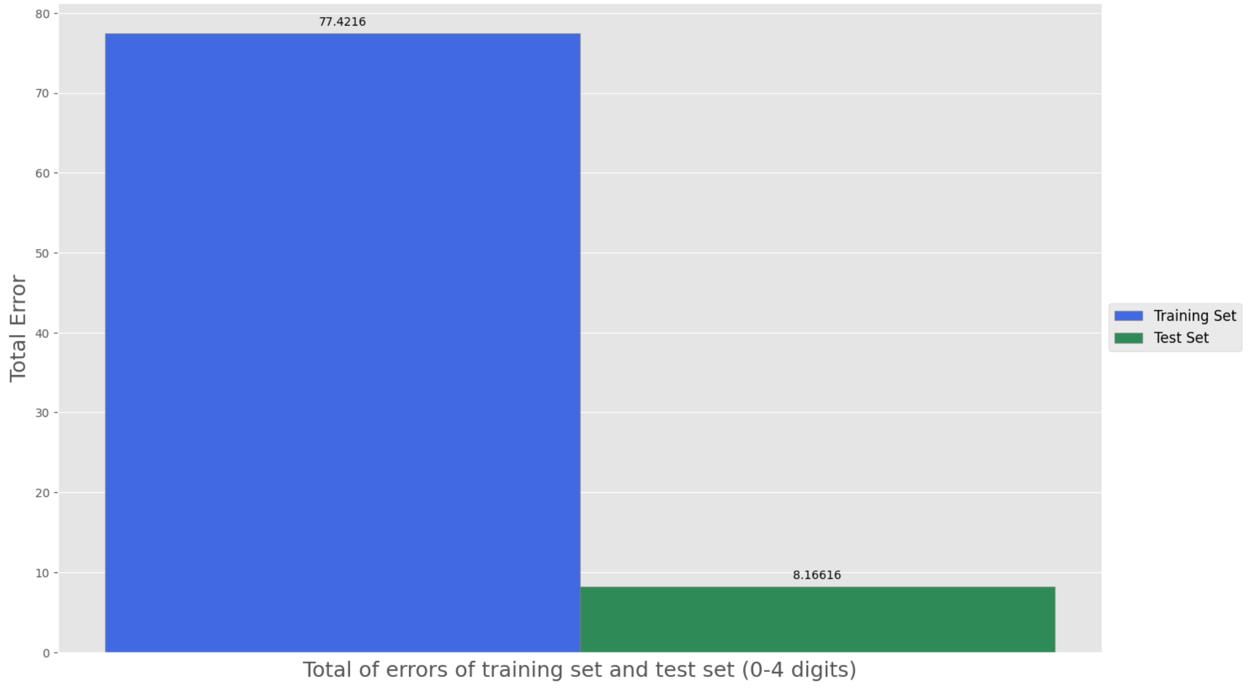
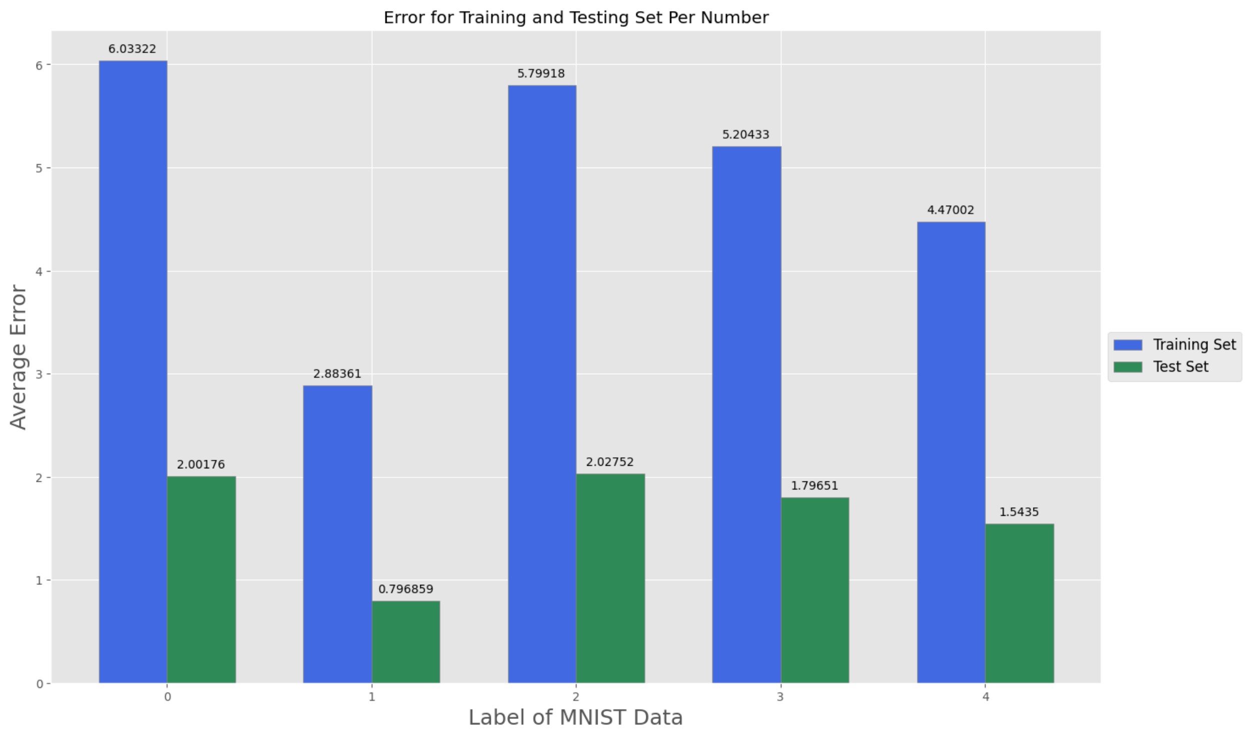


Figure 2.2. Final performance on the network of the training set and test set of 0-4 digits



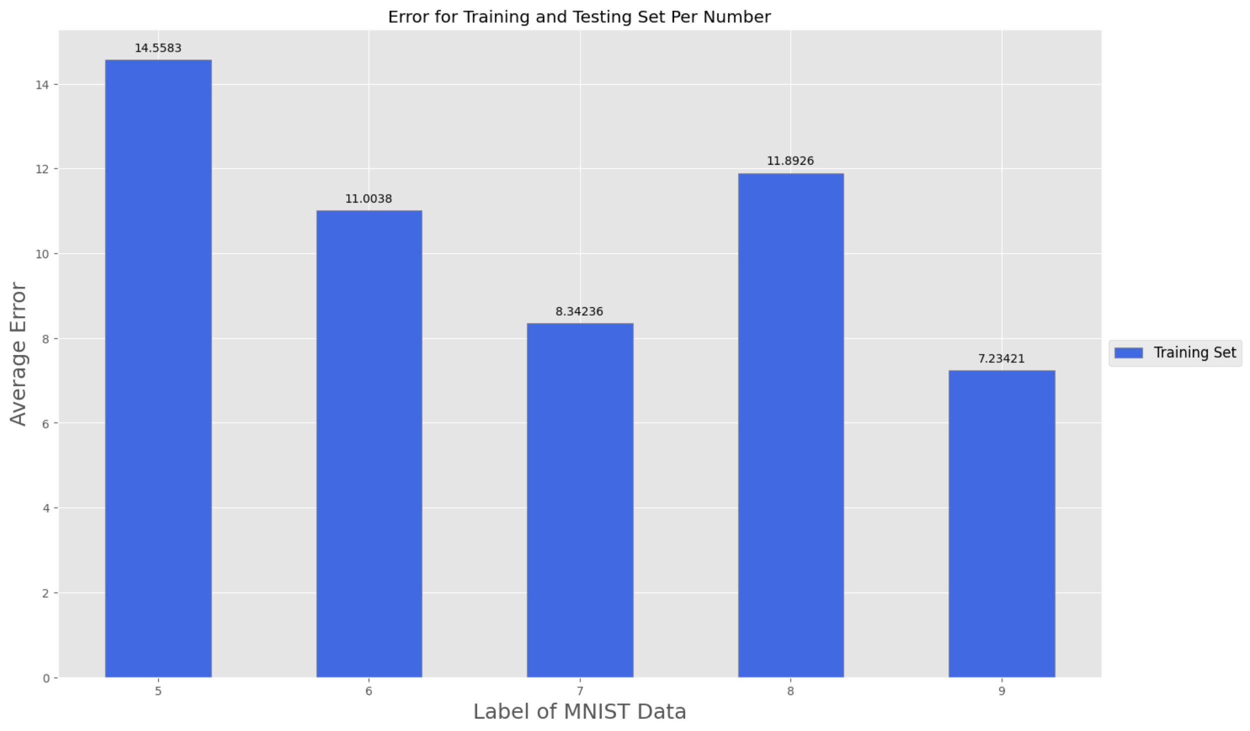


Figure 2.3. Error rate of each of the digits of the training set and test set

2.3 Analysis of Results

Although it is unclear from the first figure (Figure 2.1) what is to be classified, we can see that the numbers to be identified are from 0 to 9. Despite the fact that we trained the autoencoder using only the images of digits 0 to 4, some of them still could not be well classified. The high error rate caused a higher loss of data points because the feature sets were complex, making it difficult to identify or generalize, which ultimately resulted in a less-than-accurate result. Numbers 5, 7, and 9 were easier to reconstruct than other numbers when we tested the system on images of those numbers. The explanation is that 5 writes similarly to 2 in its writing pattern, 7 draws similarly to 1 in its drawing pattern, and 9 combines both 1 and 0. However, it was strange that the system was unable to recognize and rebuild 6 because 6 was also a combination of 1 and 0, as was 8 because it was a combination of two zeros. Figures 2.3 and 2.4 show that the test's overall loss is equal to one-third to almost two-thirds of the training set, indicating that higher error may result from the assumption that unseen data will be provided.