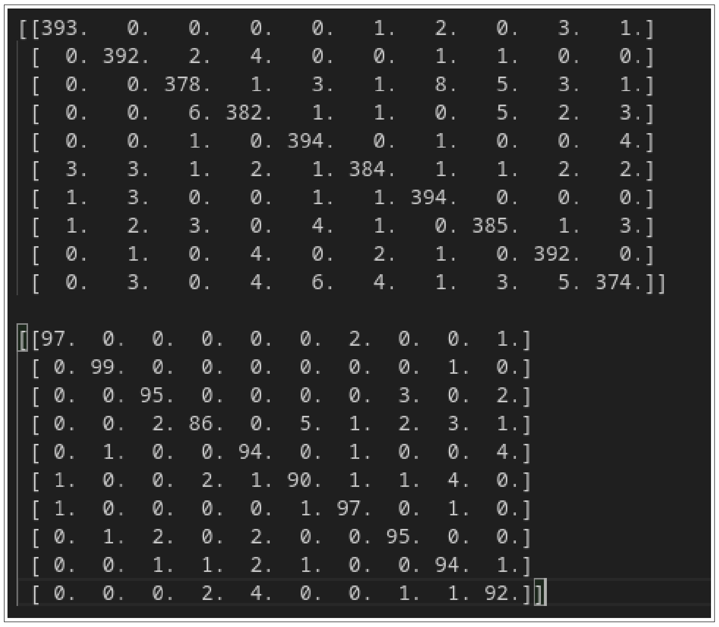
Homework 4
Chris Lee

Problem 1

System Description

I chose 125 hidden neurons, a learning rate of 0.015, and a momentum coefficient of 0.5. I stopped training when the testing error fraction started stagnating at 6.0%. My neurons used the sigmoid activation function, and I used Xavier initialization for all of the neurons. Lastly, I trained on a random selection of 1200 neurons with replacement for each epoch.

Results



```
[[393.   0.   0.   0.   0.   1.   2.   0.   3.   1.]
 [  0. 392.   2.   4.   0.   0.   1.   1.   0.   0.]
 [  0.   0. 378.   1.   3.   1.   8.   5.   3.   1.]
 [  0.   0.   6. 382.   1.   1.   0.   5.   2.   3.]
 [  0.   0.   1.   0. 394.   0.   1.   0.   0.   4.]
 [  3.   3.   1.   2.   1. 384.   1.   1.   2.   2.]
 [  1.   3.   0.   0.   1.   1. 394.   0.   0.   0.]
 [  1.   2.   3.   0.   4.   1.   0. 385.   1.   3.]
 [  0.   1.   0.   4.   0.   2.   1.   0. 392.   0.]
 [  0.   3.   0.   4.   6.   4.   1.   3.   5. 374.]]

[[97.  0.  0.  0.  0.  0.  2.  0.  0.  1.]
 [ 0. 99.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0. 95.  0.  0.  0.  0.  3.  0.  2.]
 [ 0.  0.  2. 86.  0.  5.  1.  2.  3.  1.]
 [ 0.  1.  0.  0. 94.  0.  1.  0.  0.  4.]
 [ 1.  0.  0.  2.  1. 90.  1.  1.  4.  0.]
 [ 1.  0.  0.  0.  0.  1. 97.  0.  1.  0.]
 [ 0.  1.  2.  0.  2.  0.  0. 95.  0.  0.]
 [ 0.  0.  1.  1.  2.  1.  0.  0. 94.  1.]
 [ 0.  0.  0.  2.  4.  0.  0.  1.  1. 92.]]
```

Figure 1.1. The confusion matrices for the training and test set respectively.

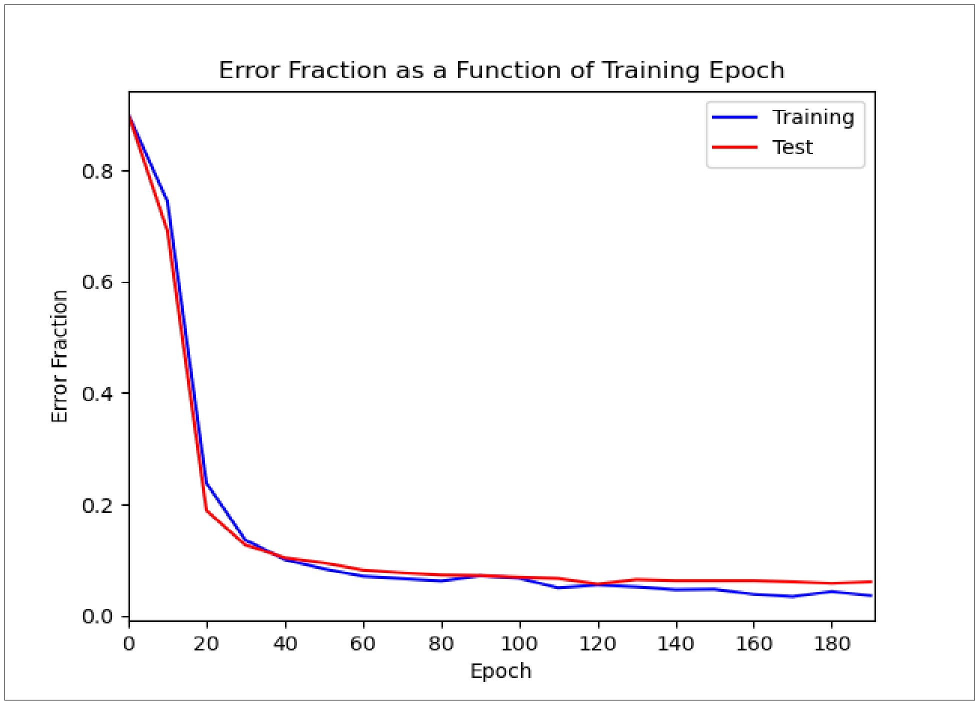


Figure 1.2. The error fraction of the entire training and test sets were recorded every 10th epoch.

Analysis of Results

As shown in Figure 1.2., my error fraction rapidly dropped for both the training and test sets within the first 60 epochs of training. The training error fraction continued to drop until the end of training, but the test set's error fraction stagnated at around 6% at around epoch 130. Given the time I've spent on this assignment, I have been unable to find a set of hyperparameters that has enabled me to decrease the testing error fraction any further. The set of hyperparameters I chose resulted in the lowest error fraction for the testing set that I could achieve before overfitting started to set in. I think that it is possible to decrease the error fraction even further by changing hyperparameters that I haven't changed so far, such as the usage of different learning rates for different layers.

My final results as shown in Figure 1.1 show that the patterns in the confusion matrices were not entirely consistent with each other. The four most accurate digits in the training set were 0, 1, 4, and 6. The four most accurate digits in the test set were 0, 1, 2, and 6. The four least accurate digits in the training set were 2, 3, 5, and 9. The four least accurate digits in the test set were 3, 4, 5, and 9. The proportions of accuracy were not that similar. I can't think of a good explanation for the inconsistencies in the accuracy proportions of the training and test confusion matrices, besides the dataset not being large enough to discount outliers. However, the most accurate digits and least accurate digits are largely consistent, and it could be explained by that the most accurate digits having the most unique features, and the least accurate digits having the least unique features.

Problem 2
System Description
I chose 125 hidden neurons, a learning rate of 0.03, and a momentum coefficient of 0.5. I stopped training when the average loss reached 1. My neurons used the sigmoid activation function, and I used Xavier initialization for all of the neurons. Lastly, I trained on the entire training dataset for each epoch.

Results

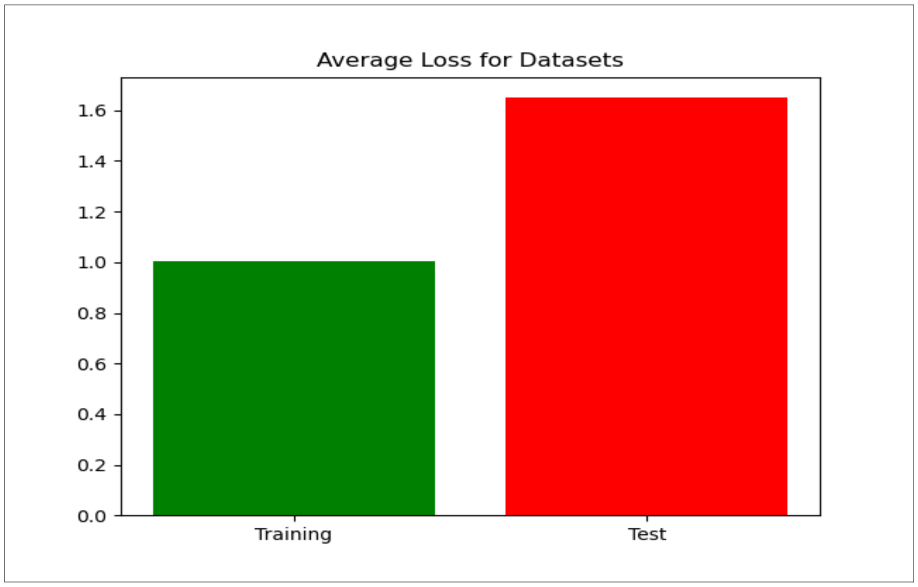


Figure 2.1. The performance of the final network in the training and test sets, quantified by average loss.

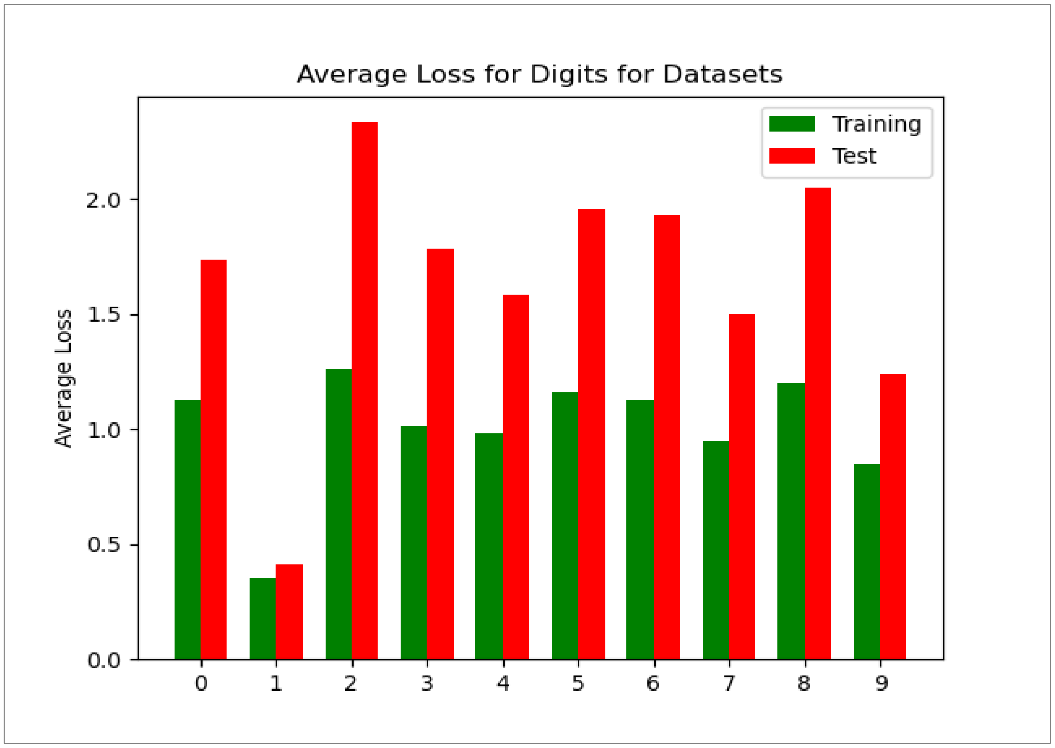


Figure 2.2. The performance of the final network for each digit in the training and test sets, quantified by average loss.

Features

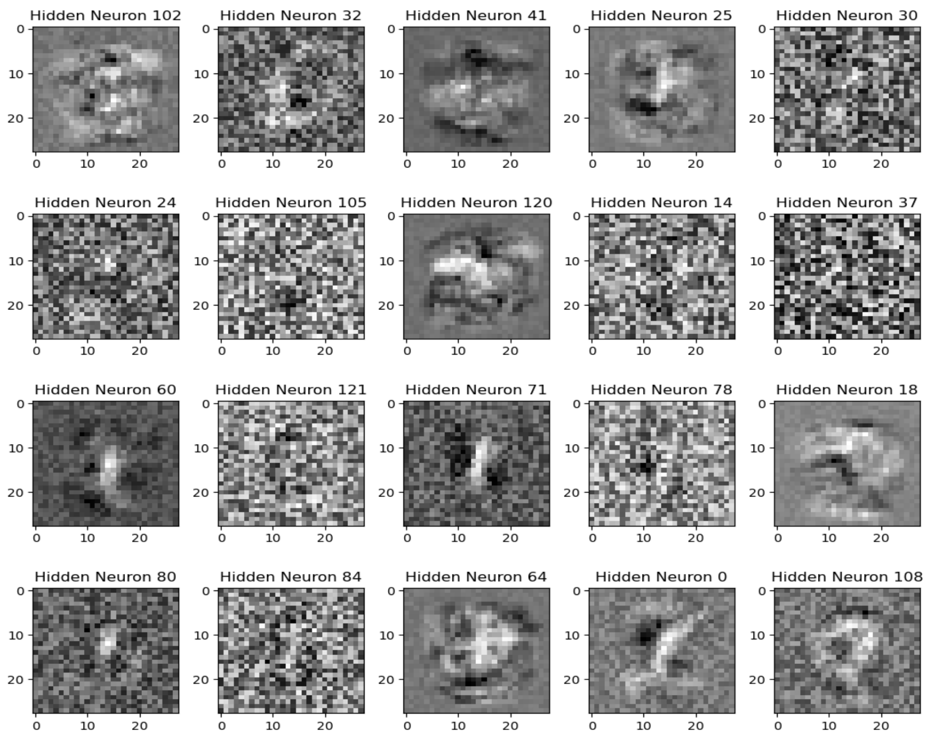


Figure 2.3. A sample of twenty hidden neurons in the classifier.

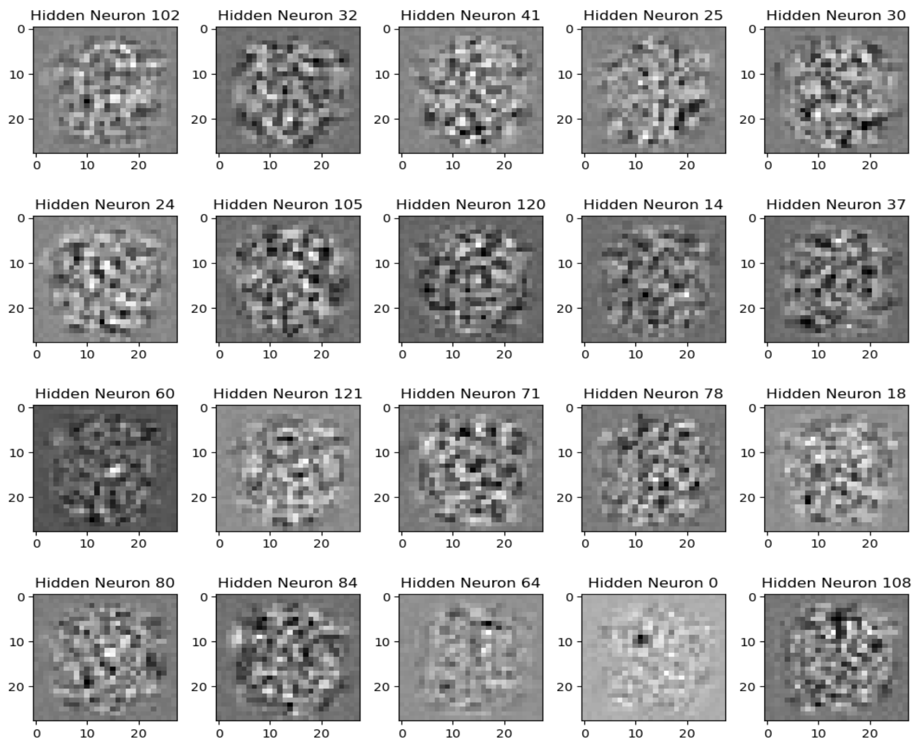


Figure 2.4. The same sample of hidden neurons in the autoencoder.

Sample Outputs

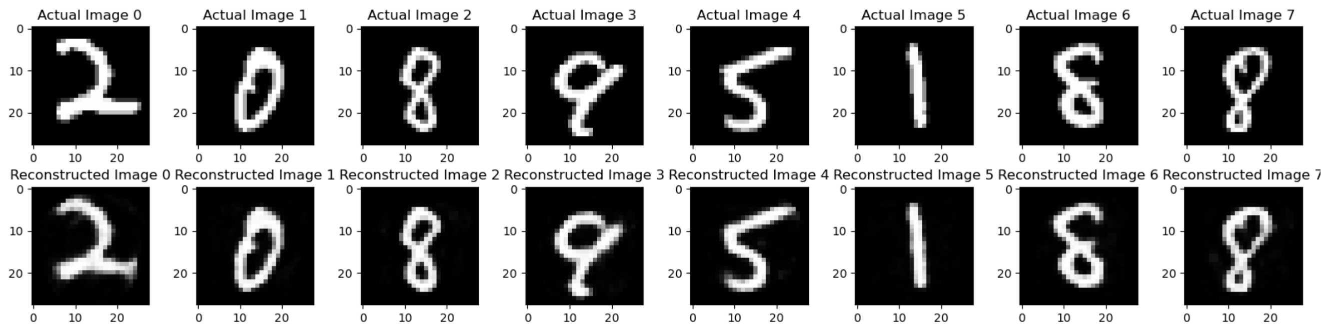


Figure 2.5. Eight input images and their respective outputs after being processed in the auto-encoder network.

Analysis of Results

I stopped training when my average loss for the training set reached 1. During the training, I used all 4000 data points and a larger learning rate. I was surprised to see how quickly the drop in loss between each epoch would plateau. In the final results for average loss as shown in Figure 2.1, it appears that some degree of overfitting has occurred, even though the testing loss never started rising during the training process. This reflects the final results of the classifier network shown in Figure 1.1, and I think there may be some nontrivial differences in the training and testing sets somehow.

The training and testing sets digit losses shown in Figure 2.2 show that the losses are largely proportional for each digit between the training and test sets. This shows that the auto-encoder is not failing to recreate any specific digit for either set, and I think this further confirms my idea that there's a nontrivial difference in my training and test sets.

The plotted hidden neurons for the classifier network and the autoencoder network, as shown in Figures 2.3 and 2.4, respectively don't seem to share much in common. For the classifier's hidden neurons, there is a large variance of noise. However, in all of them, it is still possible to at least see an outline of a feature, which is usually just a single line or even dot. The plots suggest that the neural network has found a set of lines, dots, and even spaces of inhibition whose linear combinations can accurately determine what an input image's digit is.

For the auto-encoder, the noise is concentrated in a circle around the middle. This is probably because all of the images have non-zero pixel data concentrated in this area. Secondly, the data plotted is incomprehensible; it's impossible to make out any features. This is a little surprising to me because I thought that image reconstruction would still involve the same "features" used to classify the digits.

The sample outputs show that even though the testing loss was relatively much larger to the training loss, the final network is still recreating the input images relatively accurately. However, there are some interesting changes- In Figure 2.5, the input and outputs for images 4 and 7 show that it "smooths" out the inputs when creating the outputs. Image 7 has a line that protrudes into the inside of the upper loop, but this is largely erased in the reconstruction. I think that even though the hidden neurons aren't plotting comprehensible features, it's still somehow capturing the "archetype" of each digit.