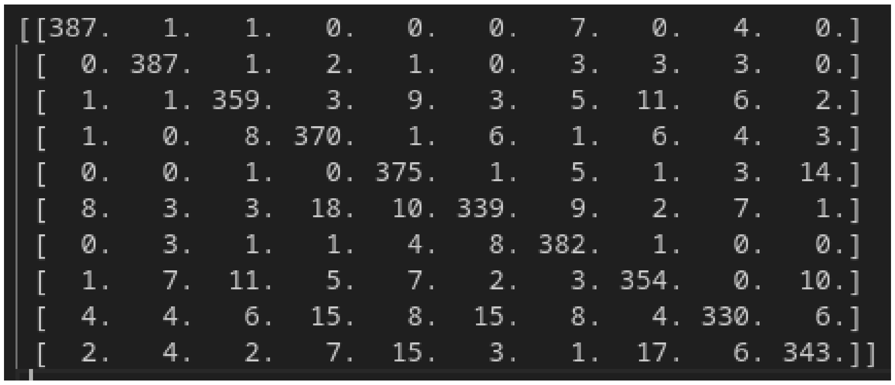
Homework 5

Chris Lee

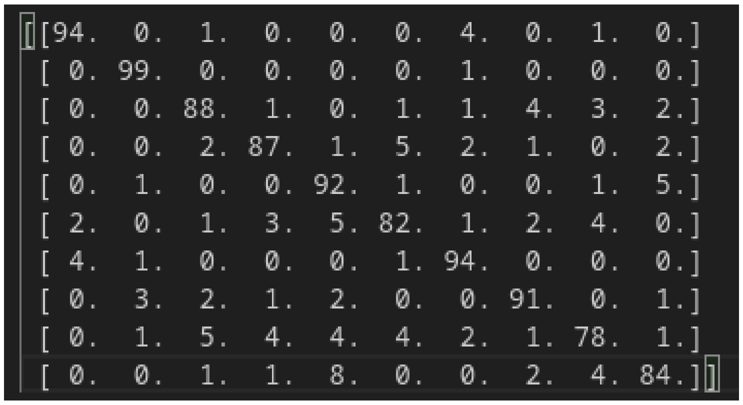
Problem 1

Results



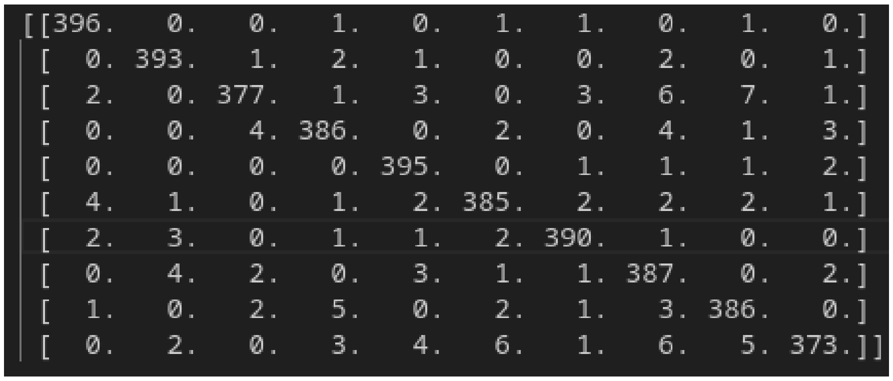
```
[[387.   1.   1.   0.   0.   0.   7.   0.   4.   0.]
 [  0. 387.   1.   2.   1.   0.   3.   3.   3.   0.]
 [  1.   1. 359.   3.   9.   3.   5.  11.   6.   2.]
 [  1.   0.   8. 370.   1.   6.   1.   6.   4.   3.]
 [  0.   0.   1.   0. 375.   1.   5.   1.   3.  14.]
 [  8.   3.   3.  18.  10. 339.   9.   2.   7.   1.]
 [  0.   3.   1.   1.   4.   8. 382.   1.   0.   0.]
 [  1.   7.  11.   5.   7.   2.   3. 354.   0.  10.]
 [  4.   4.   6.  15.   8.  15.   8.   4. 330.   6.]
 [  2.   4.   2.   7.  15.   3.   1.  17.   6. 343.]]
```

Figure 1.1. Case I confusion matrix for the entire training set.



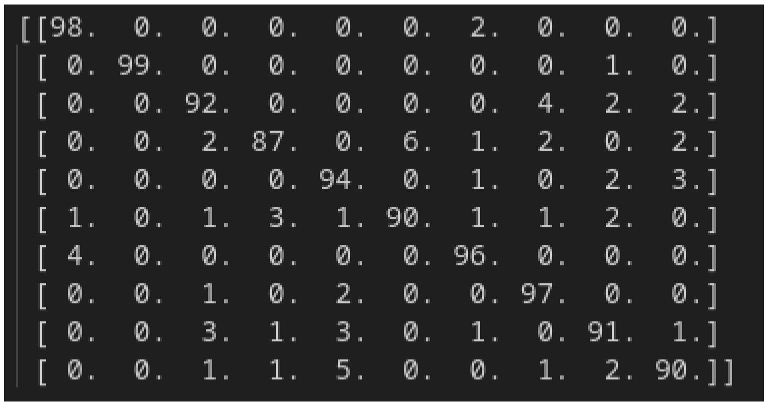
```
[[94.  0.  1.  0.  0.  0.  4.  0.  1.  0.]
 [ 0. 99.  0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0. 88.  1.  0.  1.  1.  4.  3.  2.]
 [ 0.  0.  2. 87.  1.  5.  2.  1.  0.  2.]
 [ 0.  1.  0.  0. 92.  1.  0.  0.  1.  5.]
 [ 2.  0.  1.  3.  5. 82.  1.  2.  4.  0.]
 [ 4.  1.  0.  0.  0.  1. 94.  0.  0.  0.]
 [ 0.  3.  2.  1.  2.  0.  0. 91.  0.  1.]
 [ 0.  1.  5.  4.  4.  4.  2.  1. 78.  1.]
 [ 0.  0.  1.  1.  8.  0.  0.  2.  4. 84.]]
```

Figure 1.2. Case I confusion matrix for the entire testing set.



```
[[396.   0.   0.   1.   0.   1.   1.   0.   1.   0.]
 [  0. 393.   1.   2.   1.   0.   0.   2.   0.   1.]
 [  2.   0. 377.   1.   3.   0.   3.   6.   7.   1.]
 [  0.   0.   4. 386.   0.   2.   0.   4.   1.   3.]
 [  0.   0.   0.   0. 395.   0.   1.   1.   1.   2.]
 [  4.   1.   0.   1.   2. 385.   2.   2.   2.   1.]
 [  2.   3.   0.   1.   1.   2. 390.   1.   0.   0.]
 [  0.   4.   2.   0.   3.   1.   1. 387.   0.   2.]
 [  1.   0.   2.   5.   0.   2.   1.   3. 386.   0.]
 [  0.   2.   0.   3.   4.   6.   1.   6.   5. 373.]]
```

Figure 1.3. Case II confusion matrix for the entire training set.



```
[[98.  0.  0.  0.  0.  0.  2.  0.  0.  0.]
 [ 0. 99.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0. 92.  0.  0.  0.  0.  4.  2.  2.]
 [ 0.  0.  2. 87.  0.  6.  1.  2.  0.  2.]
 [ 0.  0.  0.  0. 94.  0.  1.  0.  2.  3.]
 [ 1.  0.  1.  3.  1. 90.  1.  1.  2.  0.]
 [ 4.  0.  0.  0.  0.  0. 96.  0.  0.  0.]
 [ 0.  0.  1.  0.  2.  0.  0. 97.  0.  0.]
 [ 0.  0.  3.  1.  3.  0.  1.  0. 91.  1.]
 [ 0.  0.  1.  1.  5.  0.  0.  1.  2. 90.]]
```

Figure 1.4. Case II confusion matrix for the entire testing set.

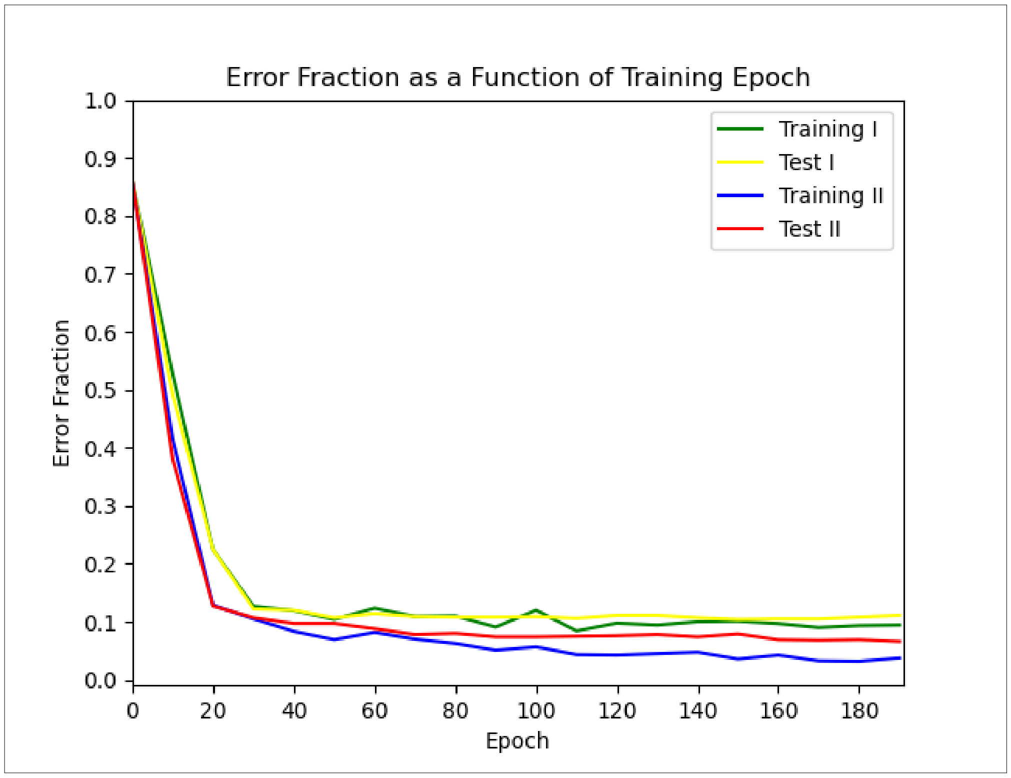


Figure 1.5. Error fraction as a function of training epoch for the Case II network.

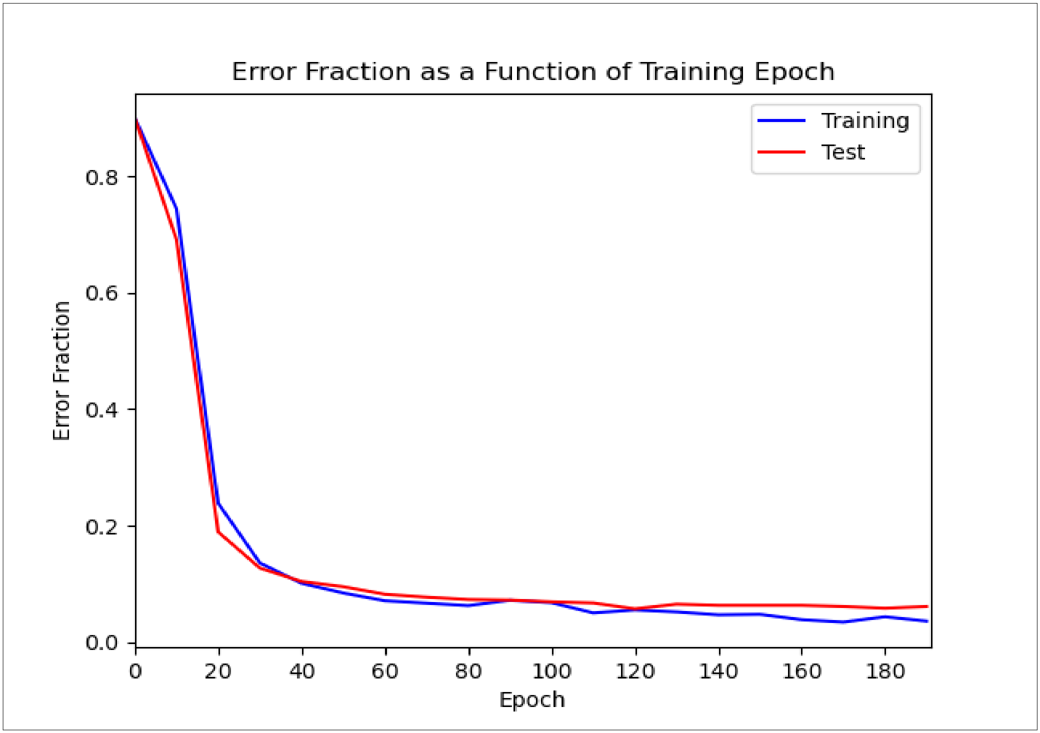


Figure 1.6. Error fraction plotted against training epochs for Homework 4 network.

Analysis of Results

The Case I network performed worse than the Case II network and the network from Homework 4 in all metrics. It had a worse overall error fraction and classified every digit worse than the other networks, as can be seen when comparing Figures 1.1 and 1.2 to Figures 1.3 and 1.4. The Case II network had inferior performance to the network from Homework 4.

The error fraction largely plateaus for the test set, while the training error fraction continues to drop, indicating that the Case II network seemed to begin overfitting on the training dataset even though it was using the same training process as the other networks. I think that Case I and Case II had worse performance than the homework 4 network because the autoencoder's hidden layer may not have represented the features as accurately as the homework 4 network's hidden layer.

Case II seems to have benefited from training the autoencoder. This is visible in Figure 1.5, where the error fractions for the Case II network are visibly both lower than the Case I network, although this still doesn't quite compare to Homework 4's network.

Training appears to have been faster when using the hidden weights from the autoencoder. The error fraction drops more rapidly for both Case I and Case II networks as shown in Figure 1.5 in comparison to Figure 1.6. It appears that training for both of these networks went faster because the hidden layer was already mostly trained.

Problem 2
Results

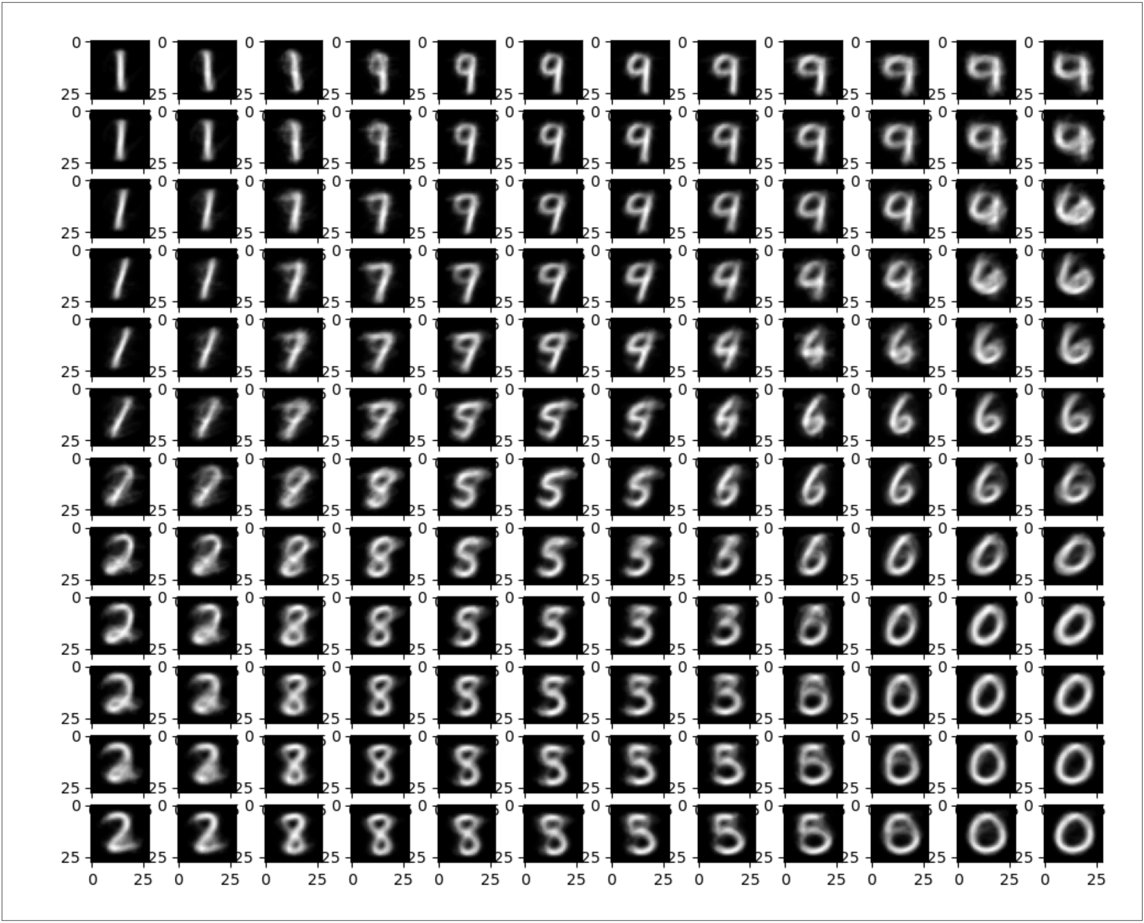


Figure 2.1. The self-organized feature map represented as an array of heatmaps.

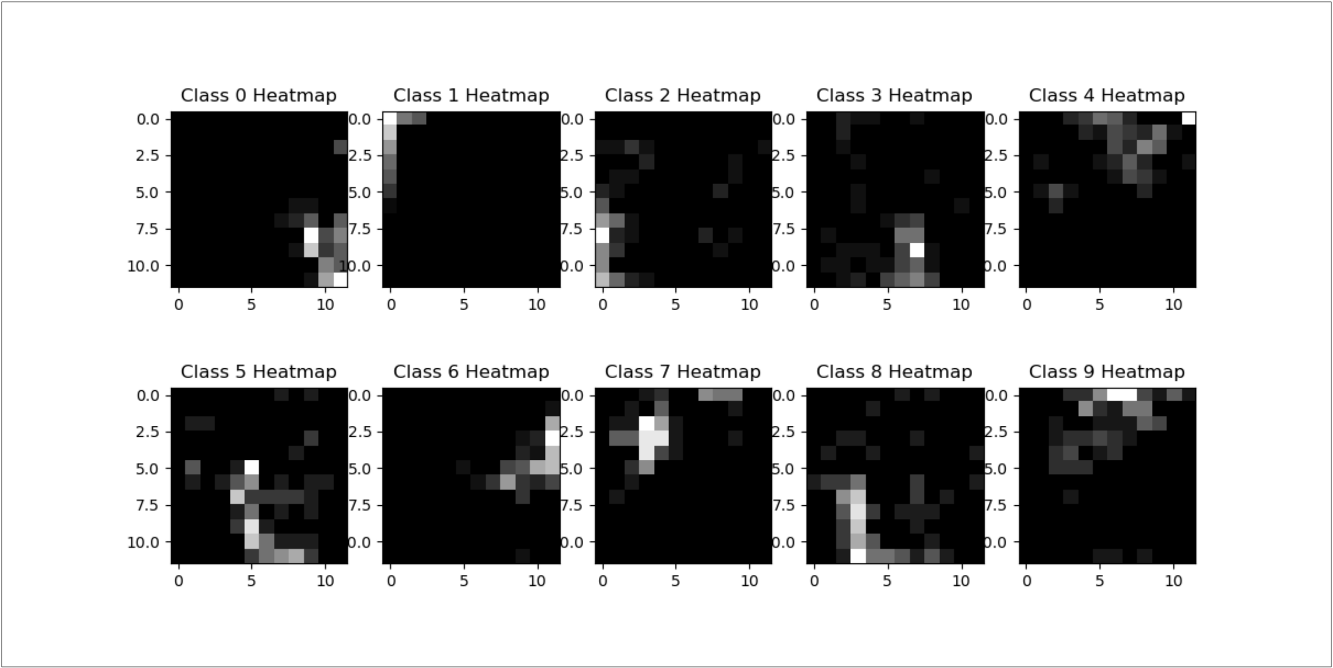


Figure 2.2. Heatmaps representing which neurons activate in response to each digit.

Analysis of Results
Figures 2.1 and 2.2 both show which digits are more similar and more different to each other. In the top left, the digit 1 dominates, and its neighbors are 7 and 9. In the bottom left, 2 dominates, and its neighbor is 8. In the bottom right, 0 dominates, and its neighbors are 5 and 6. In the top right, 4 appears and its neighbors are 9 and 6.

Problem 3
Results

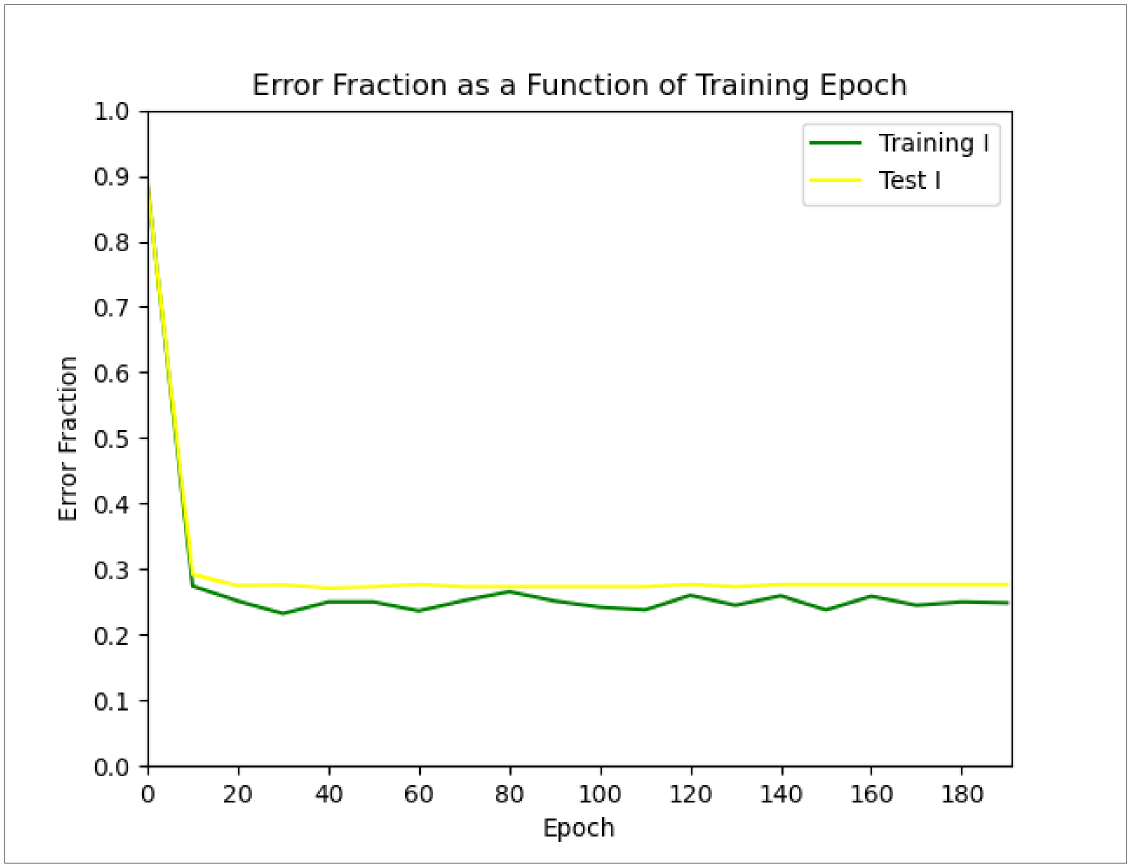
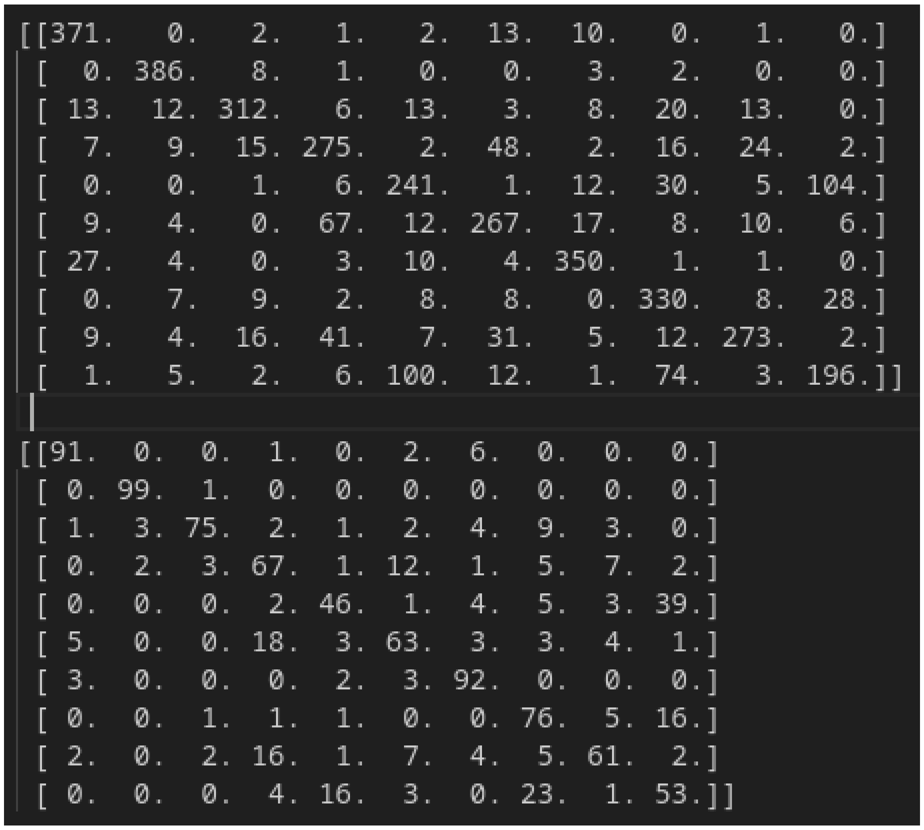


Figure 3.1. The error fraction of the network whose hidden layer is replaced with the self-organized feature map.



```
[[371.   0.   2.   1.   2.  13.  10.   0.   1.   0.]
 [  0. 386.   8.   1.   0.   0.   3.   2.   0.   0.]
 [ 13.  12. 312.   6.  13.   3.   8.  20.  13.   0.]
 [  7.   9.  15. 275.   2.  48.   2.  16.  24.   2.]
 [  0.   0.   1.   6. 241.   1.  12.  30.   5. 104.]
 [  9.   4.   0.  67.  12. 267.  17.   8.  10.   6.]
 [ 27.   4.   0.   3.  10.   4. 350.   1.   1.   0.]
 [  0.   7.   9.   2.   8.   8.   0. 330.   8.  28.]
 [  9.   4.  16.  41.   7.  31.   5.  12. 273.   2.]
 [  1.   5.   2.   6. 100.  12.   1.  74.   3. 196.]]

[[91.  0.  0.  1.  0.  2.  6.  0.  0.  0.]
 [ 0. 99.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  3. 75.  2.  1.  2.  4.  9.  3.  0.]
 [ 0.  2.  3. 67.  1. 12.  1.  5.  7.  2.]
 [ 0.  0.  0.  2. 46.  1.  4.  5.  3. 39.]
 [ 5.  0.  0. 18.  3. 63.  3.  3.  4.  1.]
 [ 3.  0.  0.  0.  2.  3. 92.  0.  0.  0.]
 [ 0.  0.  1.  1.  1.  0.  0. 76.  5. 16.]
 [ 2.  0.  2. 16.  1.  7.  4.  5. 61.  2.]
 [ 0.  0.  0.  4. 16.  3.  0. 23.  1. 53.]]
```

Figure 3.2. The confusion matrices on the training and test sets, respectively.

Analysis of Results
The usage of the self-organized feature map as a hidden layer yielded much worse results compared to using the auto-encoder or training a hidden layer. As shown in Figure 3.1, the error fraction bottomed out at 27% for the testing set, and around 25% for the training set. However, the specific error fractions for each digit has a much larger variance in comparison to the networks discussed previously. In Figure 3.2, it is obvious to see that digits 4, 5, 8, and 9 had much poorer performance than 0, 1, and 6.

The error fraction seems to be correlated with how much of the self-organized feature map the digits seemed to take up. As shown in Figure 2.1, the digits 0, 1, and 6 have well-defined regions, where digits like 4 especially already appear to be melded with the other digits even in the neurons where Figure 2.2 show that the poorly classified digits dominate.

Lastly, the computational effort involved to create this neural network was much greater than the other neural networks. This is likely because the processing of an input by the self-organized feature map is computationally expensive, because it involves getting the minimum output, which requires iterating over all 144 inputs. The training of the self-organized feature map itself was very expensive, as well. My training process was 250 epochs over all four thousand points-- less than 200 epochs yielded much worse self-organized feature maps. In comparison, the earlier networks used 190 epochs of training on a different random subset of 1200 points from the training set.