Homework 4

Problem #1

Rather than setting the weights from the inputs to the hidden neurons at random, set them to the final weights from the input to the hidden layer from problem #2 in homework 3.

Results:

Case I:

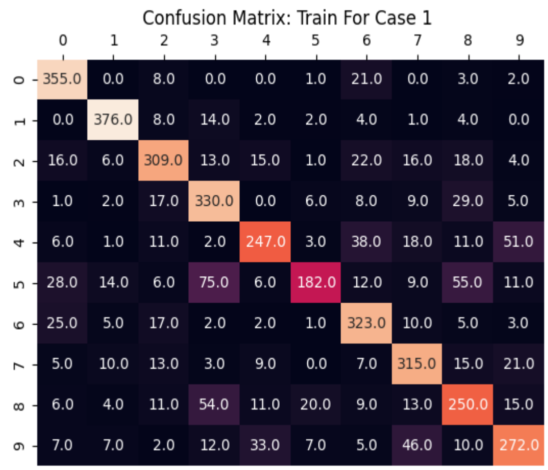


Figure 1.1: *10x10 confusion matrix for the training data*

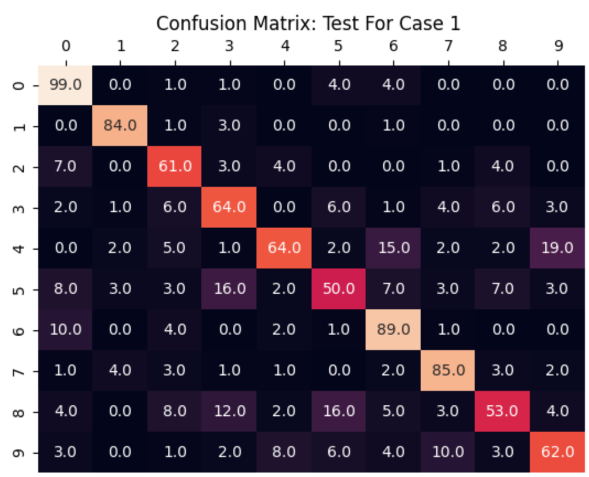


Figure 1.2: *10x10 confusion matrix for the testing data*

Problem #1

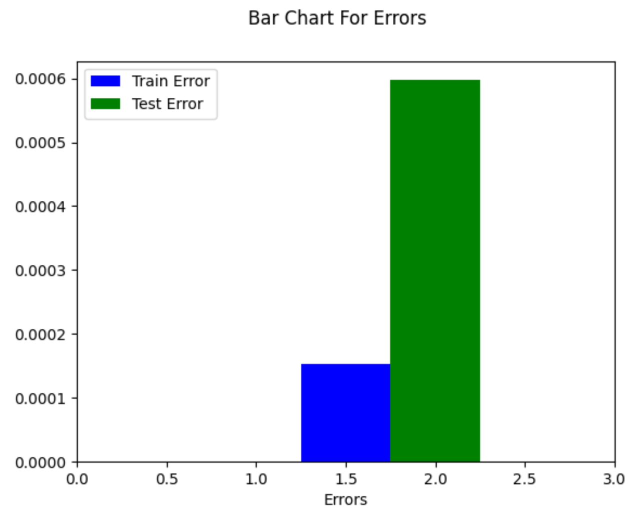


Figure 1.3: *Bar chart of the training error and testing error after completion of training*

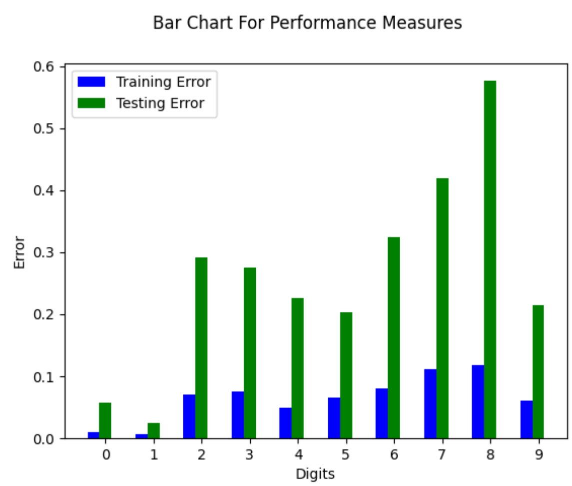


Figure 1.4: *Bar chart of the training error and testing error of digits 0-9*

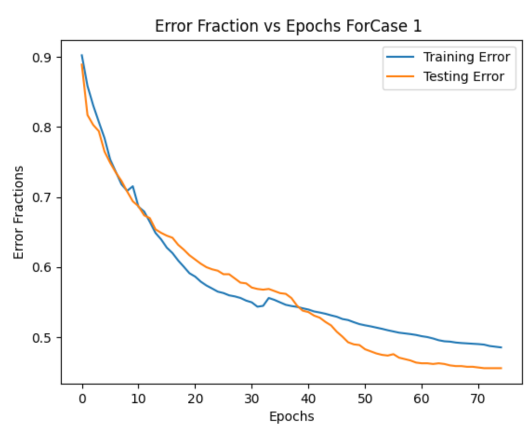


Figure 1.5: *Epochs against the training error fraction and the test error fraction*

Problem #1

Case II:

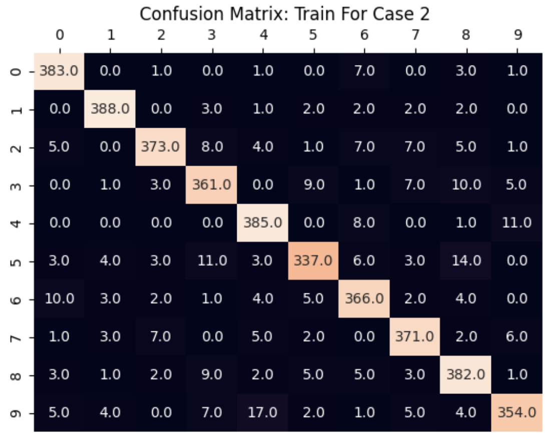


Figure 1.6: *10x10 confusion matrix for the training data*

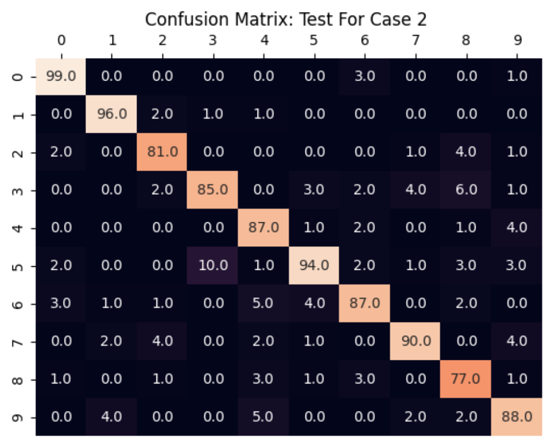


Figure 1.7: *10x10 confusion matrix for the testing data*

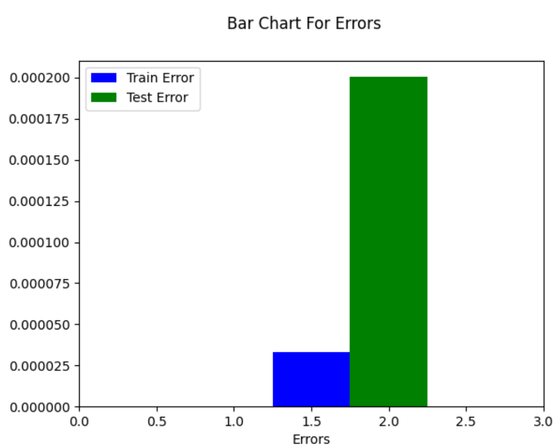


Figure 1.8: *Bar chart of the training error and testing error after completion of training*

Problem #1

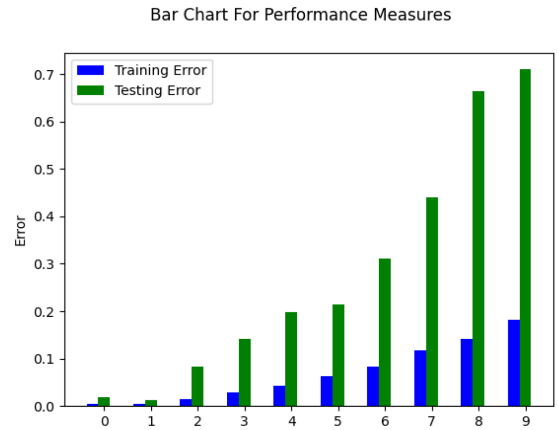


Figure 1.9: *Bar chart of the training error and testing error of digits 0-9*

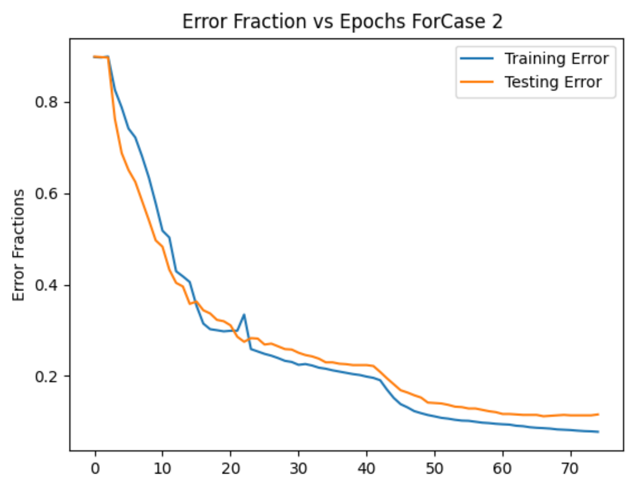


Figure 1.10: *Epochs against the training error fraction and the test error fraction*

Analysis of Results:

The graphs that were produced indicate that the network performs better in the case II than it does in the case I. It is clear from figure 1.3 and figure 1.8 that the case II's training error is significantly lower than case I. One explanation for this is because the autoencoder has indeed acquired the features from the provided images and may use them to its advantage during the classification. The network performance in case II is improved by training the previously identified feature weights. When the weights from the autoencoder were initialized against random initialization in homework 3, the training time during the same number of epochs significantly decreased. This may be because using the previously learnt features made it simple to determine the best location for the separation boundary. Both the network in case I and the network in homework 3 perform worse than the network in case II, when both weights are updated.

Problem #2
Train the auto-encoder network in Problem 2 in homework 3 by training the network using digits 0, 1, 2, 3 and 4 and test it on digits 5, 6, 7, 8 and 9.

System Description:

After experimenting with various combinations, the respective hyperparameters were used for the auto-encoder neural network to solve this issue: the number of hidden neurons is 150; a learning rate of 0.001 is recommended with 1000 epochs; and a momentum value of 1 due to the larger number of training examples. The weights were initialized randomized since probabilities are considered when using the sigmoid function to predict output. The weight initialization directly impacts the output, so it might take more than one run to obtain the desired output. The training is deemed finished once a predefined number of epochs have been completed with a training error that is noticeably low.

Results:

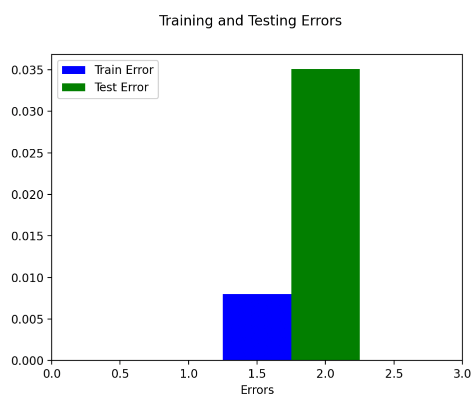


Figure 2.1: *Bar chart of the training error and 0-to-4 testing error after completion*

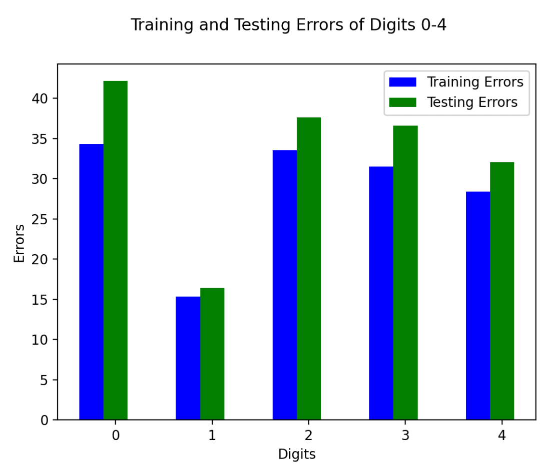


Figure 2.2: *Bar chart of the training error and testing error of digits 0-4*

Problem #2

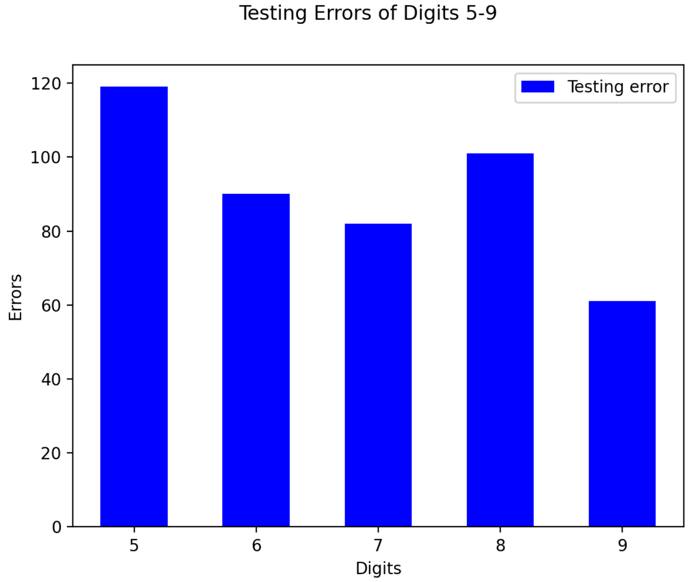
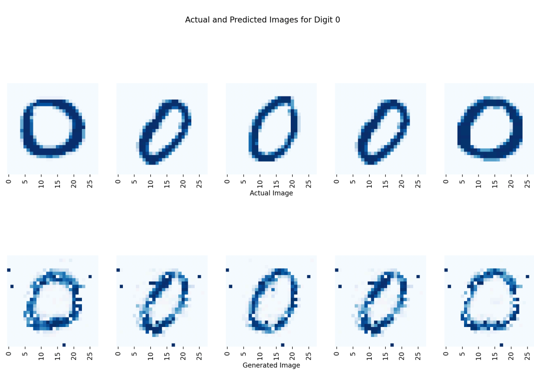
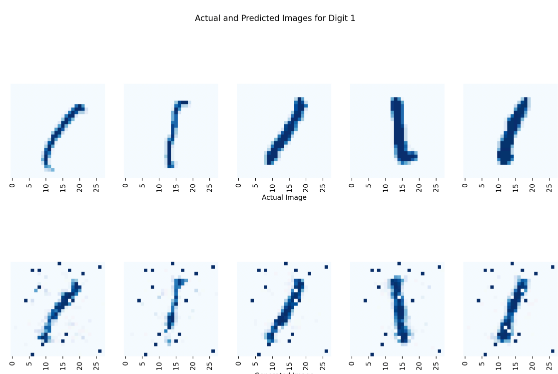
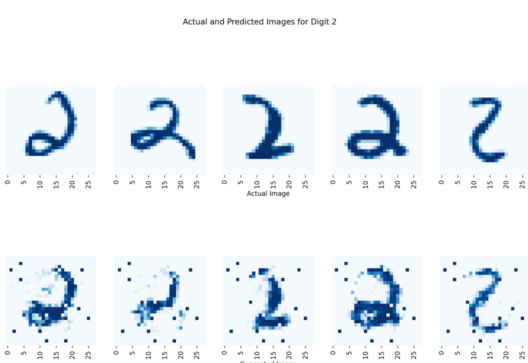


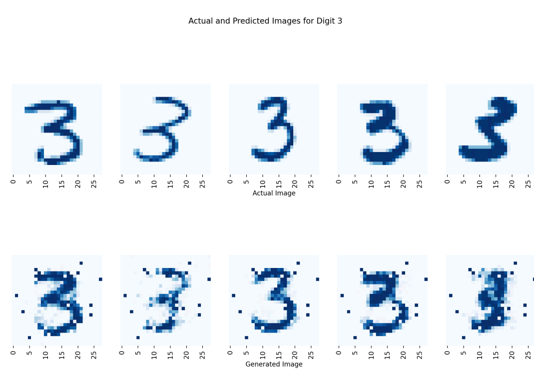
Figure 2.3: *Loss values for the 5-to-9 digits*

Sample Output:









Problem #2

Figure 2.4: *Example output of 5 random actual images and their corresponding generated images from digit 0-9*

Analysis of Results:

The input can be successfully rebuilt by the auto-encoder. For the digits 0-4, the auto-encoder can reconstruct the input with an accuracy of about 98%. Figure 2.1 illustrates that although both errors are smaller in comparison, the training error is lower than the testing error, demonstrating that the network fits the data well. Figure 2.2 shows that 1 should be easy to reconstruct due to its structure, whereas 0 and 2 should be challenging due to their similar structures. Figure 2.3 shows that while the structure of 9 is presumably easy to reconstruct because it is like the structure of the training sets of 0–4, the structures of 5 and 8 are probably more challenging to reconstruct because they

Problem #2

are not like the training sets of 0–4. As the number of epochs rises, the training error gradually decreases until it stabilizes. This demonstrates the network's strong generalizability and good fit. The reconstructed images in Figure 2.4 lend visual support to this observation.