## Homework 3

### Problem #1

The objective of Problem #1 is to create a multi-layer feed-forward neural networks and training them with back- propagation including momentum that can handle any number of hidden layers.

### System Description:

The following hyperparameters were used for the multi-layer feed-forward neural network for this problem after trying several combinations: the optimal number of hidden neurons is 146, a learning rate of 0.001 is ideal with the number of epochs of 750, the value for the momentum is 1 for the reason of having bigger number of training examples, the chosen H and L thresholds are 0.75 and 0.25, respectively. Because probabilities are considered while making output predictions using the sigmoid function, the weights were initialized randomly. The weight initialization has a significant impact on the final output, and as a result, getting the appropriate output may need more than one run.

### Results:

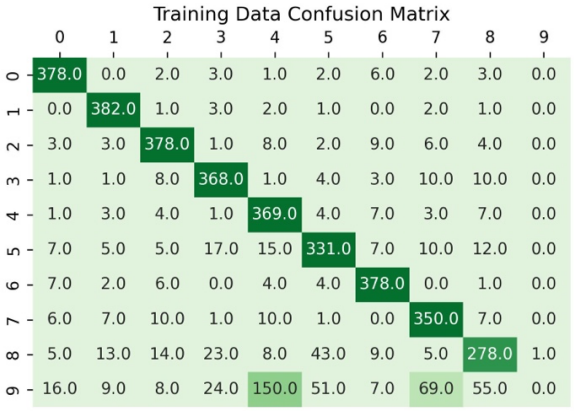


**Figure 1.1**: *10x10 confusion matrix for the training data*

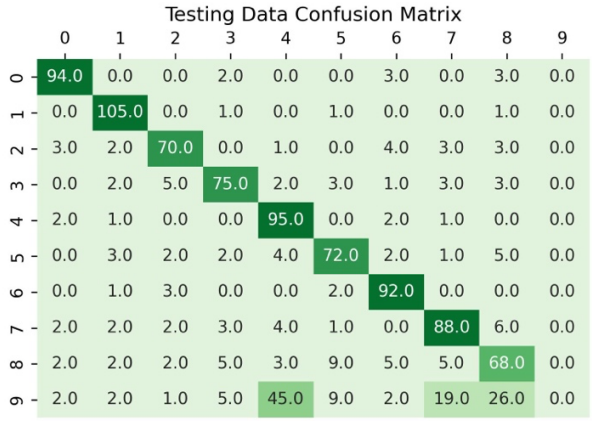


**Figure 1.2**: *10x10 confusion matrix for the testing data*

**Problem #1**

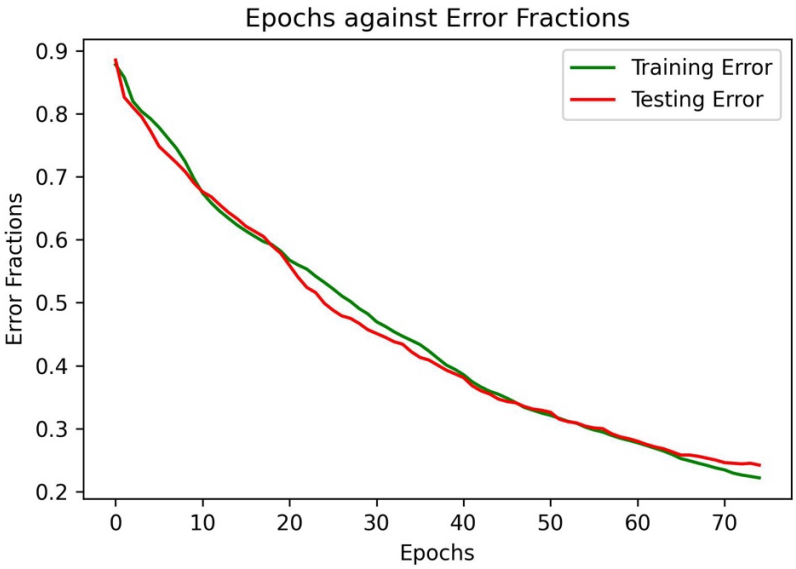


**Figure 1.3**: *Epochs against the training error fraction and the test error fraction*

**Analysis of Results:**

As a conclusion, we can state that the multi-layer feed-forward neural network can effectively and accurately identify the MNIST data when given the right set of hyperparameters. From the figure 1.3, it clearly states that the network stabilizes after the first training period, during which the error drop is significant in the first few epochs. The error fraction determined at the zero epoch, when the network generates output using random weights, represents the error fraction prior to training the network. Also, the training error is decreasing at a faster rate than the testing error. However, the testing error stabilizes before the training error does because the network absorbed the data's structure across several training epochs. The decrease that finally leads to stability shows that gradient descent performance was successful. The generated confusion matrix in figure 1.1 and figure 1.2 supports the subsequent finding. In the training set as opposed to the testing set, there is a larger ratio of properly identified instances to wrongly categorized examples. This observation is proof that the network was properly trained and capable of generalizing to previously undiscovered data. Due to the approximate division of the training and testing sets, it is possible that some numbers may have acquired more data than others, making that digit's classification accuracy significantly greater. The figures 1.1 and 1.2, where the network had difficulty classifying 0 and 9, provide support for the aforementioned finding. The diagonal of both matrices displays instances of digits from the training and testing sets that were properly classified.

**Problem #2**

The objective of Problem #2 is to train an auto-encoder neural network using the same training sets and test sets in Problem #1 and to attain a good set of features to represent them.

**System Description:**

The following hyperparameters were used for the auto-encoder neural network for this problem after trying several combinations: the same number of hidden neurons in problem #1 which is 146, a learning rate of 0.001 is ideal with the number of epochs of 1000, the value for the momentum is 1 for the reason of having bigger number of training examples. Because probabilities are considered while making output predictions using the sigmoid function, the weights were initialized randomly. The weight initialization has a significant impact on the final output, and as a result, getting the appropriate output may need more than one run. After a predetermined number of epochs for which obtaining a noticeably low training error, the training is considered complete.

**Results:**

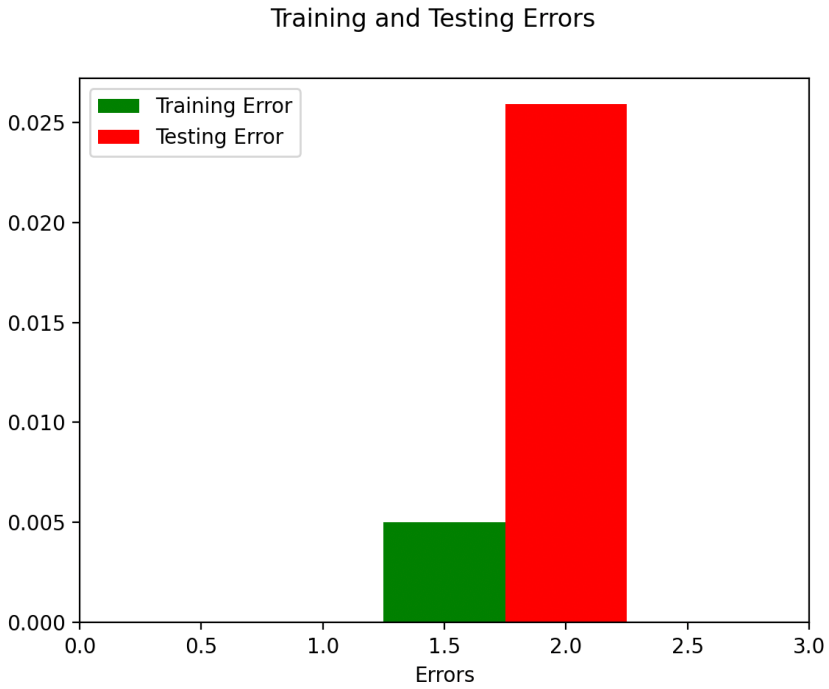


**Figure 2.1:** *Bar chart of the training error and testing error after completion of training*

**Problem #2**

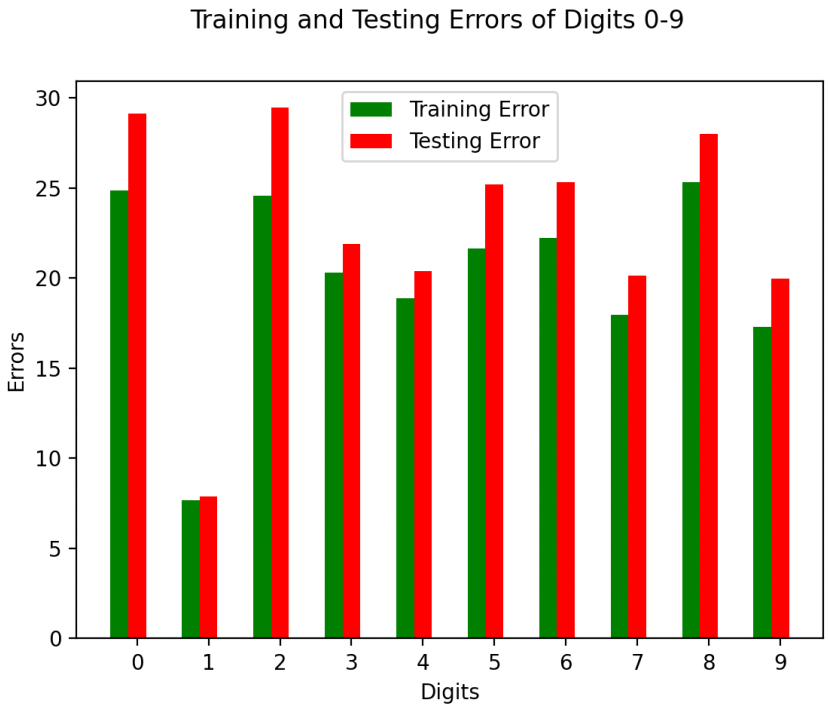


**Figure 2.2:** *Bar chart of the training error and testing error of digits 0-9*

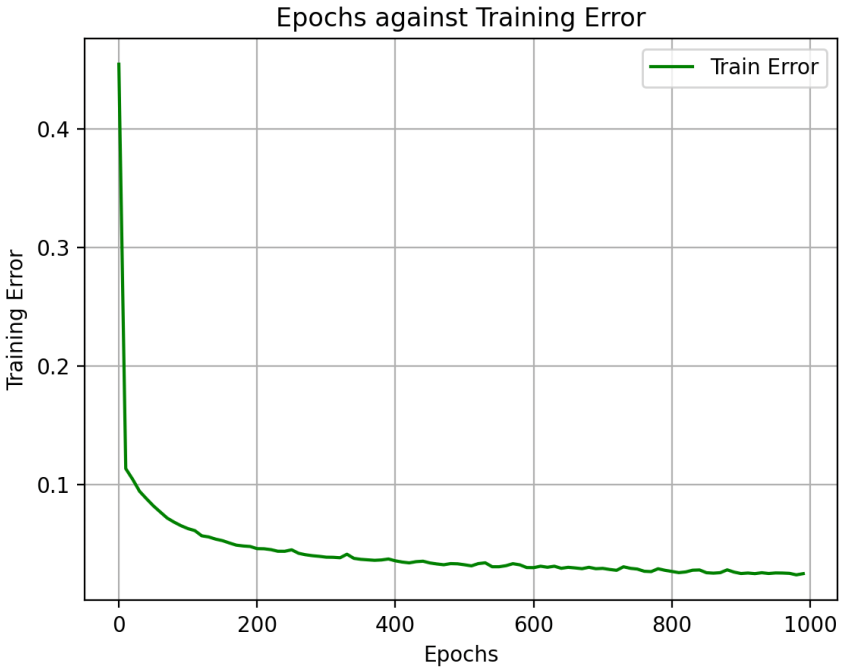


**Figure 2.3:** *Time series of the training error against epochs*

**Problem #2**

**Features:**

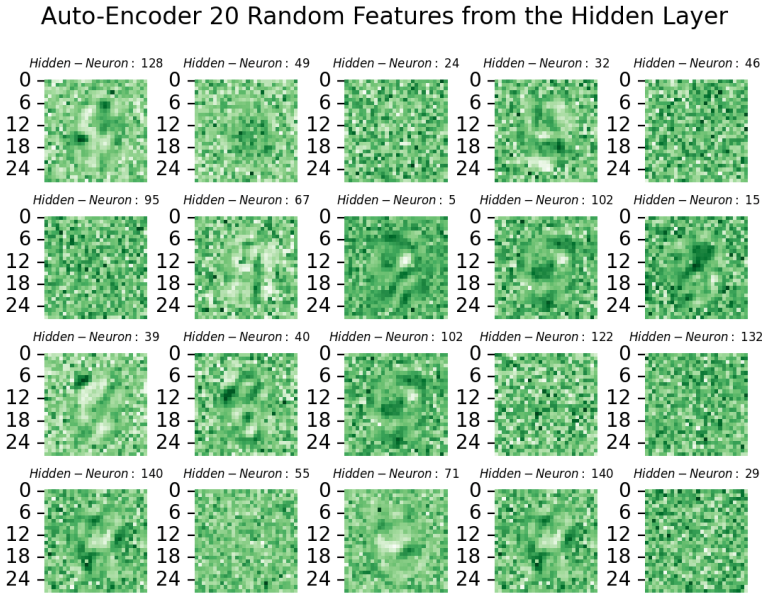


**Figure 2.4:** *Auto-encoder's 20 random features from the hidden layer*

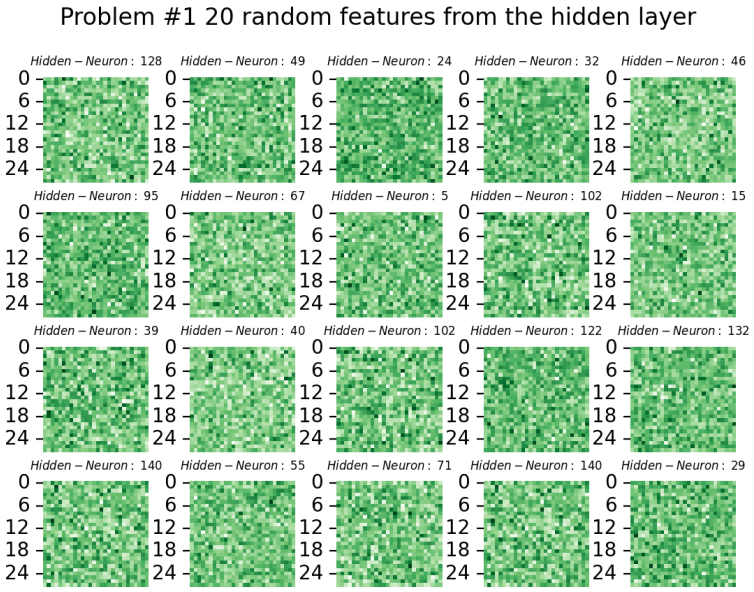


**Figure 2.5:** *Multi-layer feed-forward neural network's* (from problem #1) *20 random features from the hidden layer*

In comparison between Figure 2.4 which is an auto-encoder network and Figure 2.5 which is the network from problem #1, it is clearly shown that small portion of the numbers are visible in Figure 2.4 which implies that the hidden neurons became tuned to particular areas of the picture.

**Problem #2**

**Sample Output:**

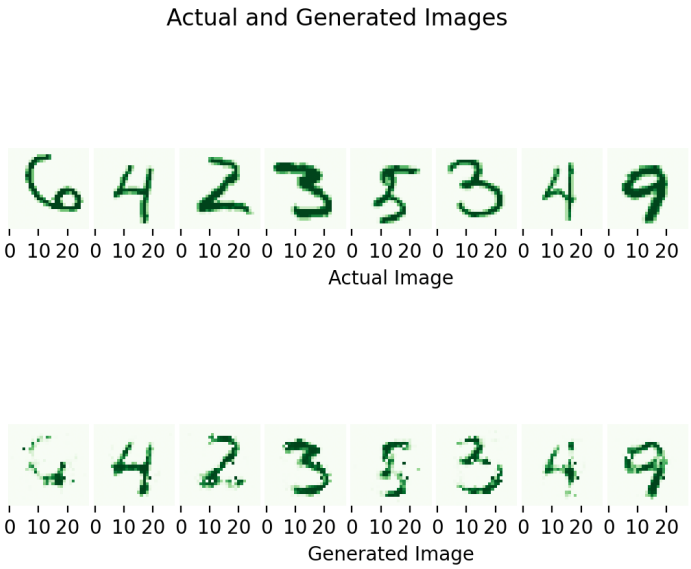


**Figure 2.5:** *An example output of 8 random test images and the associated auto-encoder-generated output images*

**Analysis of Results:**

The auto-encoder can effectively rebuild the input by learning hidden features. The input can be accurately reconstructed by the auto-encoder with an accuracy of about 98%. According to Figure 2.1, the training error is lower than the testing error, although both errors are smaller in comparison, showing that the network fits the data well. According to Figure 2.2, 1 is presumably easy to reconstruct because of its structure, but 0, 2, and 8 are probably difficult to reconstruct because of how similar their structures are. The training error steadily lowers with an increase in the number of epochs until stabilizing. This demonstrates the network's good fit and strong generalizability. Curve and straight lines are the main features that the network learned for this particular run, as we can see in Figure 2.4. The general structure of the properties learnt by the hidden neurons is seen in Figure 2.5. This is likely due to the fact that Figure 2.4 represents the hidden output of an auto encoder, which attempts to reconstruct the input provided, causing nearly all hidden neurons to tune to strike for a specific fragment, as opposed to neural networks, where the task of hidden neurons is classification, leading to the generic nature of weights.