Problem 1.

Problem Statement. In this problem, we will implement a multi-layer feed-forward neural network and train it using the back-propagation algorithm and gradient descent with momentum. Our major focus will be on the network's behavior when there is only 1 hidden layer, and the performance is evaluated on the MNIST dataset using *error fraction* and confusion matrices.

System Description. The best performance for our network was observed for the following values of system variables:

Parameter	**Value**
# hidden layers	1
# neurons in hidden layers	[120] (array indicates the size of each hidden layer)
epochs	400
eta	0.001
gamma	0.4
hidden layer activation function	sigmoid
Initial weights criterion	random values from Standard Normal Distribution with numpy random seed set to 1234
loss function	squared error
optimizer	gradient descent with momentum
output layer activation function	softmax
output thresholds	NA

Results.

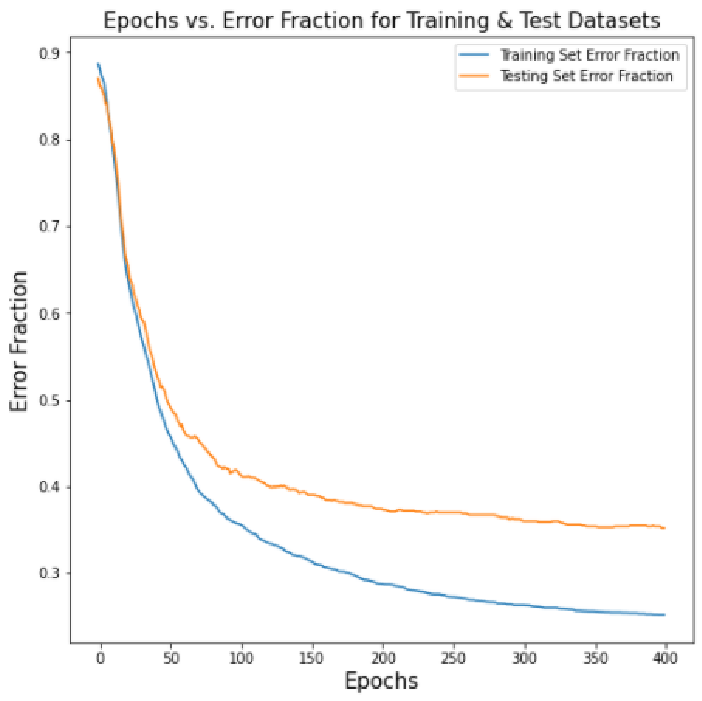
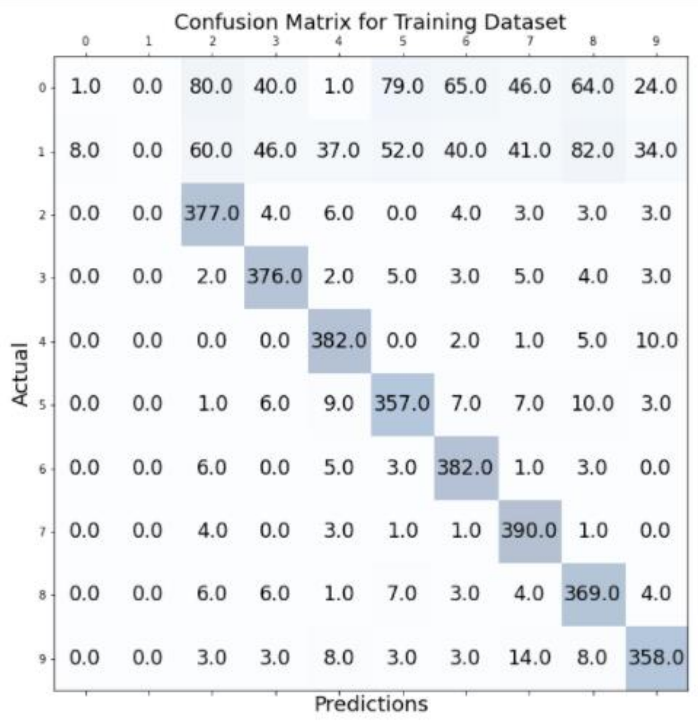
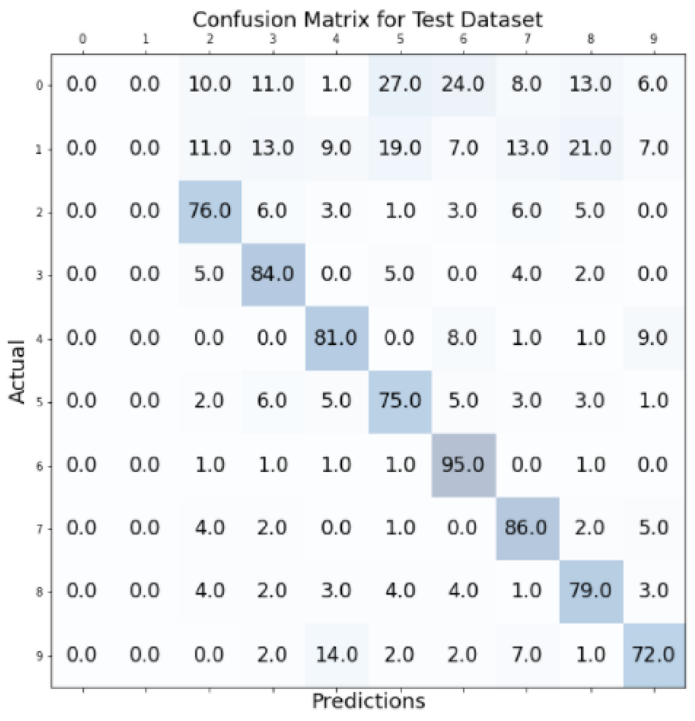


Figure 1.1: Epochs vs. Error Fraction for Training & Test Datasets


Confusion Matrix for Training Dataset

Actual \ Predictions	0	1	2	3	4	5	6	7	8	9
0	1.0	0.0	80.0	40.0	1.0	79.0	65.0	46.0	64.0	24.0
1	8.0	0.0	60.0	46.0	37.0	52.0	40.0	41.0	82.0	34.0
2	0.0	0.0	377.0	4.0	6.0	0.0	4.0	3.0	3.0	3.0
3	0.0	0.0	2.0	376.0	2.0	5.0	3.0	5.0	4.0	3.0
4	0.0	0.0	0.0	0.0	382.0	0.0	2.0	1.0	5.0	10.0
5	0.0	0.0	1.0	6.0	9.0	357.0	7.0	7.0	10.0	3.0
6	0.0	0.0	6.0	0.0	5.0	3.0	382.0	1.0	3.0	0.0
7	0.0	0.0	4.0	0.0	3.0	1.0	1.0	390.0	1.0	0.0
8	0.0	0.0	6.0	6.0	1.0	7.0	3.0	4.0	369.0	4.0
9	0.0	0.0	3.0	3.0	8.0	3.0	3.0	14.0	8.0	358.0

Figure 1.2: Confusion Matrix for the Training Dataset After Training


Confusion Matrix for Test Dataset

Actual \ Predictions	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	10.0	11.0	1.0	27.0	24.0	8.0	13.0	6.0
1	0.0	0.0	11.0	13.0	9.0	19.0	7.0	13.0	21.0	7.0
2	0.0	0.0	76.0	6.0	3.0	1.0	3.0	6.0	5.0	0.0
3	0.0	0.0	5.0	84.0	0.0	5.0	0.0	4.0	2.0	0.0
4	0.0	0.0	0.0	0.0	81.0	0.0	8.0	1.0	1.0	9.0
5	0.0	0.0	2.0	6.0	5.0	75.0	5.0	3.0	3.0	1.0
6	0.0	0.0	1.0	1.0	1.0	1.0	95.0	0.0	1.0	0.0
7	0.0	0.0	4.0	2.0	0.0	1.0	0.0	86.0	2.0	5.0
8	0.0	0.0	4.0	2.0	3.0	4.0	4.0	1.0	79.0	3.0
9	0.0	0.0	0.0	2.0	14.0	2.0	2.0	7.0	1.0	72.0

Figure 1.3: Confusion Matrix for the Test Dataset After Training

Analysis of Results. We started training our network using the standard values for momentum (0.4) and learning rate (0.001) and set the number of epochs to a fairly large value of 400. For a randomly generated set of initial parameters based on the Standard Normal Distribution, our network performed very well for the first 100 epochs and the learning has then slowed down afterwards. This explains the saturation in accuracy and error fractions. This could have been due to the sigmoid functions reaching their extreme values in the hidden layers. Talking about numbers, the error fraction on the training set before training was 0.88 and the accuracy was around 11%. Similarly, the error fraction on the test set for randomly assumed parameters was 0.87 and the accuracy was 13%. As the training progressed, the error fraction has gradually reduced while the accuracy has increased for both the training and test datasets. This behavior can be observed from *Figure 1.1.* The same figure also shows that our network has started overfitting the training data after 100 epochs and was unable to generalize the learning on the test data. There was a 10% difference in accuracy between training and test data sets after training. This explains the difference between graphs from *Figure 1.1.* Interestingly, the network has always failed to learn either 0, 1 or 1, 2 digits for different values of parameters and hyper-parameters. This can be seen from the confusion matrices from *Figure 1.2 & Figure 1.3.* In conclusion, our network has exhibited good learning behavior on the dataset available with us. However, the accuracy couldn't go past 70% on the test set which implies the necessity of using more than 1 hidden layers in our network for the MNIST dataset.