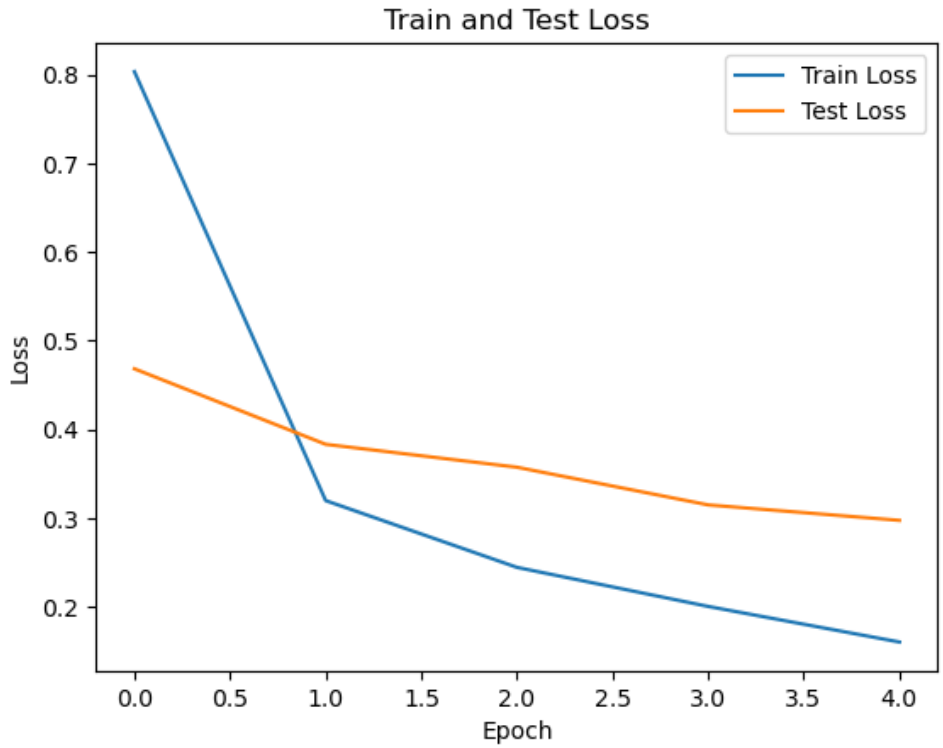
Assignment 3

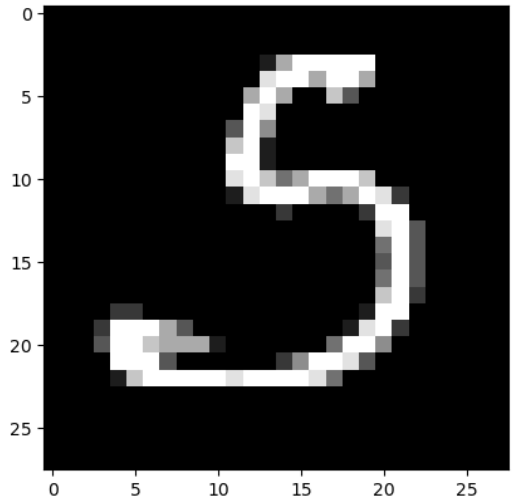
Task 1

```
Epoch [1/5], Train Loss: 0.8032, Test Loss: 0.4681
Epoch [2/5], Train Loss: 0.3195, Test Loss: 0.3830
Epoch [3/5], Train Loss: 0.2442, Test Loss: 0.3571
Epoch [4/5], Train Loss: 0.2002, Test Loss: 0.3146
Epoch [5/5], Train Loss: 0.1600, Test Loss: 0.2972
```

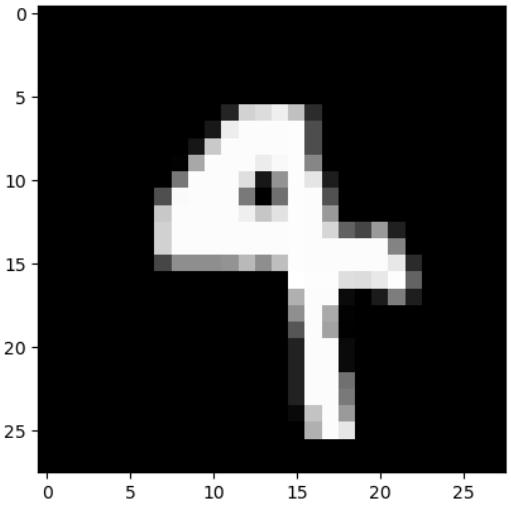


Accuracy of the network on the 10000 test images: 91 %
Test error (loss) for trained model 0.29720718214909236

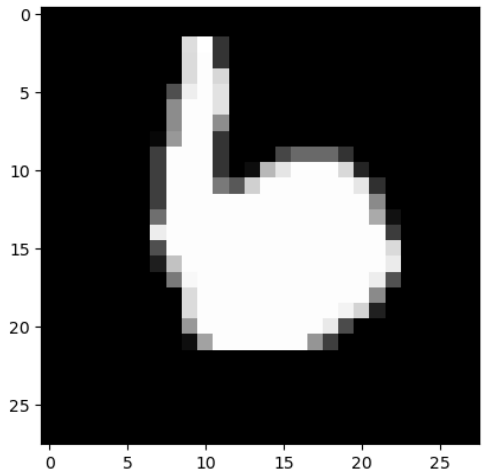
Some misclassified images



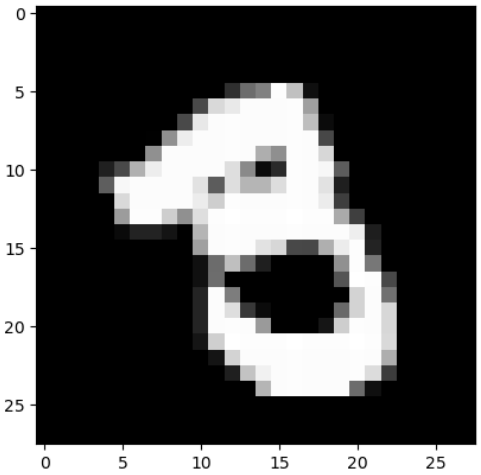
Predicted: 3
Actual: 5



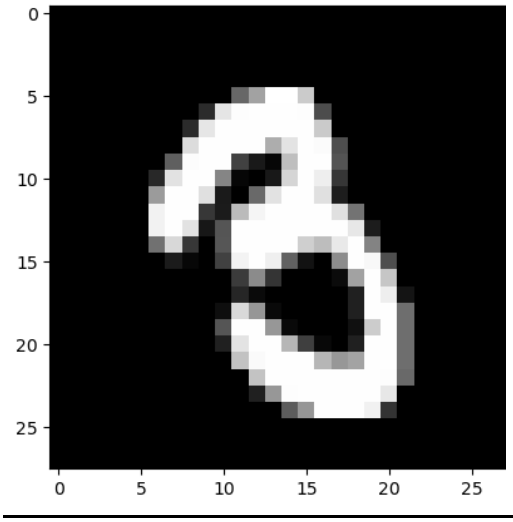
Predicted: 9
Actual: 4



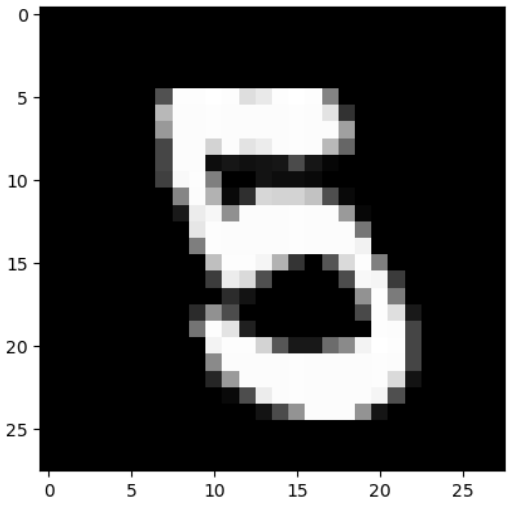
Predicted: 4
Actual: 6



Predicted: 8
Actual: 3

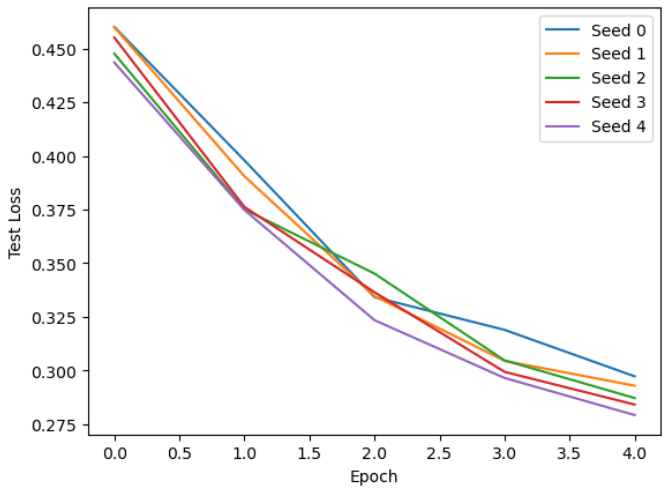


Predicted: 8
Actual: 3



Predicted: 8
Actual: 5

Task 2



Our Results:

Test error mean: 0.2882, **Test error std**: 0.0064

Based on the variance of the final test errors for each of the seeds we computed (Variance is the square of std).
As we can see, the standard deviation is low in comparison to the mean, also, in the plot we got we can observe that on all 5 independent runs (for which we trained are model) the graphs representing the test errors are similar to each other.
So, we can conclude that our model is indeed robust to the choice of the seed number.

Task 3

$(\boldsymbol{e}^*_{\boldsymbol{te}}, \boldsymbol{e}^*_{\boldsymbol{va}})$ such that e^*_{va} is the minimum throughout the training:
[0.10306332632899284, 0.2823956326271097]

Task 4

For the following hyperparameters values:

Hidden sizes = [400, 500, 600]

Batch sizes = [100, 200, 300]

Learning rates = [0.01, 0.001]

We have produced the following table:

Batch Size	Hidden Size	Learning Rate	Best validation loss	Test error for same epoch
100	400	0.01	0.12778663	0.285383652
100	400	0.001	0.131919164	0.287545425
100	500	0.01	0.127366383	0.297997185
100	500	0.001	0.128317158	0.292207769
100	600	0.01	0.131498893	0.292669085
100	600	0.001	0.147854576	0.308337054
200	400	0.01	0.17625069	0.328219447
200	400	0.001	0.190445307	0.338780781
200	500	0.01	0.191741601	0.340231987
200	500	0.001	0.181920633	0.32545496
200	600	0.01	0.195835623	0.332954895
200	600	0.001	0.173691687	0.322528313
300	400	0.01	0.218510404	0.356127454
300	400	0.001	0.241830673	0.362268367
300	500	0.01	0.216744442	0.351284007
300	500	0.001	0.231768161	0.36031138
300	600	0.01	0.24644623	0.363260362
300	600	0.001	0.225014199	0.365310583

The highlighted combinations are the ones with the best performance, which is:

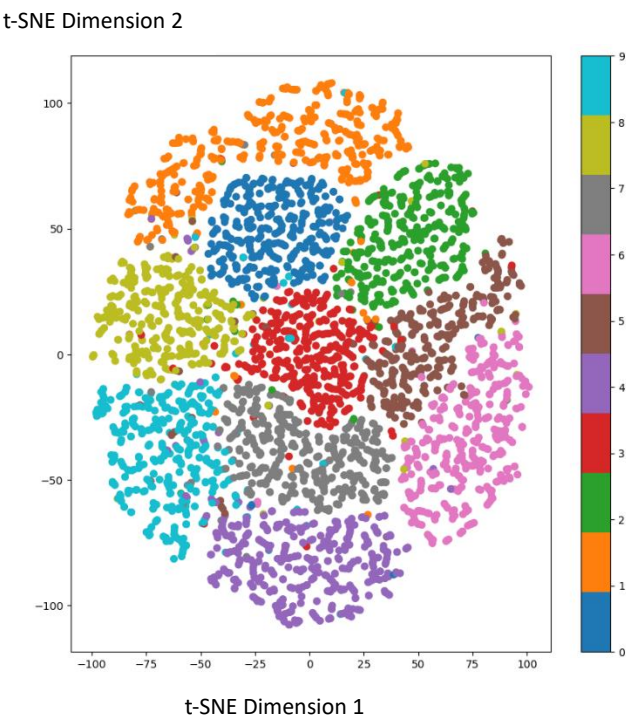
For lowest test loss:

Batch size = 100
Hidden size (of the intermediate layer) = 400
Learning rate = 0.01

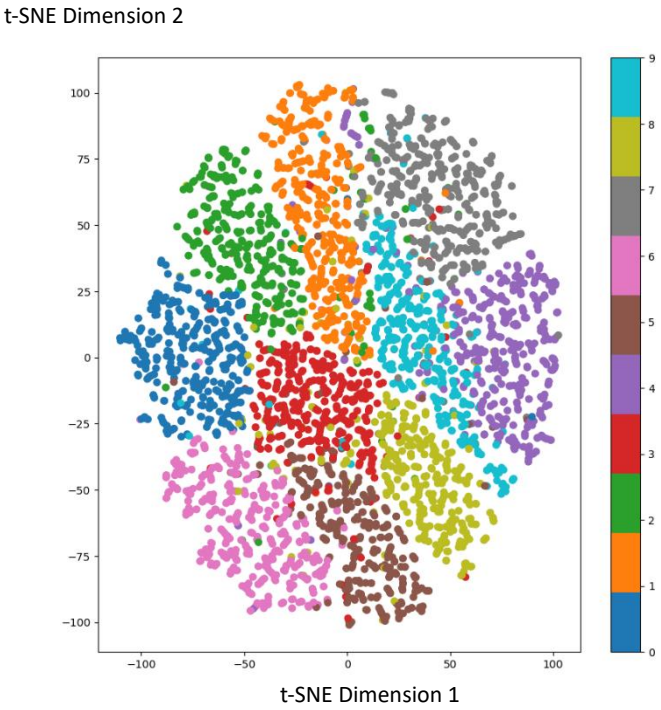
for lowest validation loss:
Batch size = 100
Hidden size (of the intermediate layer) = 500
Learning rate = 0.01

Task 5

Z Graph



X Graph



As we can observe in these plots, the z_i points are slightly more spread (we can see that in digits 1,6,9) and the "clusters" of the digits are a bit more separable (4 and 9, 3 and 8).

After applying the first layer on the input (and getting z_i from x_i), the $z_i s$ were more separable which means the learned model indeed made a "step" in the right direction of its classification task. By that, we can assume that the next layer will also classify some digits to their correct label even more and lower the test loss.
So, we can say that the learned model does indeed have parameters (weights) (that it obtained during the learning process) and operating function (ReLU in our case) that are compatible with the classification task.