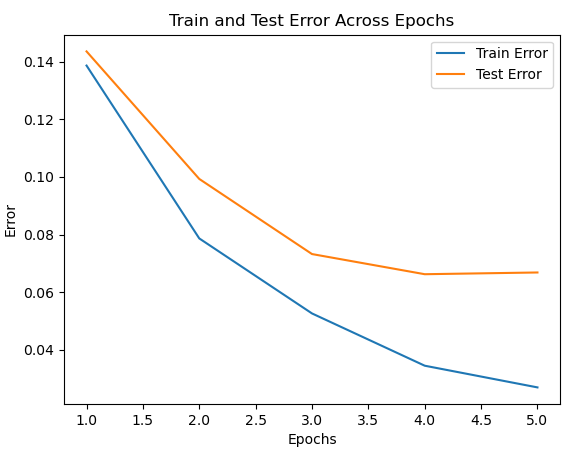
**Applied Deep Learning - Homework 3**

**Task 1**

The following graph shows the test and train errors after each epoch our model did while training. The model consistent of two fully connected linear layers with the ReLU function.

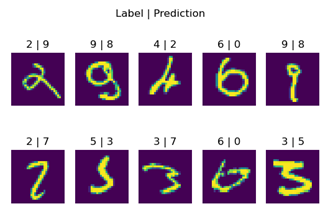


Test error of final network on the 10000 test images: 0.0668.

Accuracy of: 97.97%

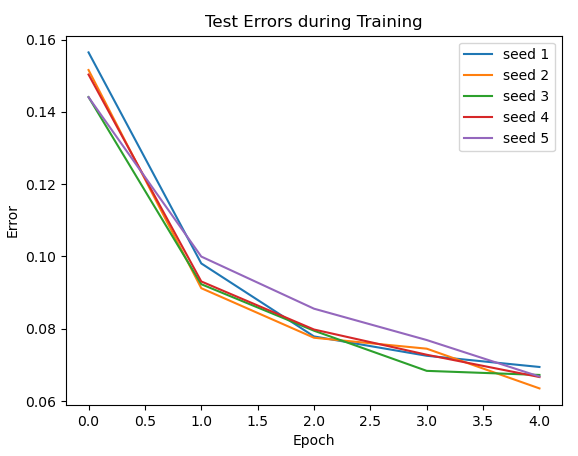
Both errors are decreased with each epoch and the decline is slowing down the longer the model trains. We can also see the train error is lower than the test error as expected.

The following are images that our final model has misclassified. Above each image we have the actual Label, followed by the misclassification the model did.



We can see that the images the model failed on are somewhat hard to label even as a human and for almost all of them we can see the similarity between the image and the prediction the model gave.

**Task 2**  
the following graph shows the same earlier model’s test error per epoch, for each seed between 1-5.



Mean final Error is 0.0667.

Standard deviation of final Errors is 0.00212.

The final errors exhibit very low standard deviation, and the graph illustrates that even though the different graphs aren’t exactly the same, the errors have comparable patterns among various seed values.

Judging by the variance and the graph provided, it appears that the model's performance is relatively stable and not too sensitive to the initial seed number chosen. To definitely say the model is robust to the choice of a seed number, more tests with more seed numbers should be conducted, but based on these results we would say the model is robust.

**Task 3**

We tried using 5 different seeds (numbers 1-5).

Min validation error found with seed 1, after the 5th epoch.

Min validation error: 0.0796.

Corresponding test error: 0.0739.

Table with results for each seed-epoch combination found at the end of the report under **Task 3 – full results**.

It’s worth mentioning, in the full table we can see although there is a clear correlation between the validation and test errors, the min validation-error doesn’t necessarily correspond to the min test-error.   
But since we shouldn’t use our test data for hyper-parameters validation to avoid overfitting our test data, we still choose the seed-epoch combination that yields the min **validation-error**.

**Task 4**

The following table show the test error corresponds to the best validation error of every hyper-parameters combination.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hidden Size** | **Batch Size** | **Learning Rate** | **Test Error** |
| 100 | 100 | 0.01 | 0.1039 |
| 100 | 100 | 0.001 | 0.1093 |
| 100 | 100 | 0.0001 | 0.289 |
| 100 | 1000 | 0.01 | 0.1099 |
| 100 | 1000 | 0.001 | 0.229 |
| 100 | 1000 | 0.0001 | 0.8266 |
| 500 | 100 | 0.01 | 0.1174 |
| 500 | 100 | 0.001 | 0.0739 |
| 500 | 100 | 0.0001 | 0.2088 |
| 500 | 1000 | 0.01 | 0.0686 |
| 500 | 1000 | 0.001 | 0.1555 |
| 500 | 1000 | 0.0001 | 0.4456 |

The combination that yields the best results is:   
Hidden Size = 500 , Batch Size = 1000 and Learning rate = 0.01.

We checked as a thumb rule what was the average test error for every hyper-parameter option:  
(the test errors we averaged are the same one displayed in the table)

|  |  |
| --- | --- |
| **Hidden size** | **Average test error** |
| 100 | 0.2780 |
| 500 | 0.1783 |

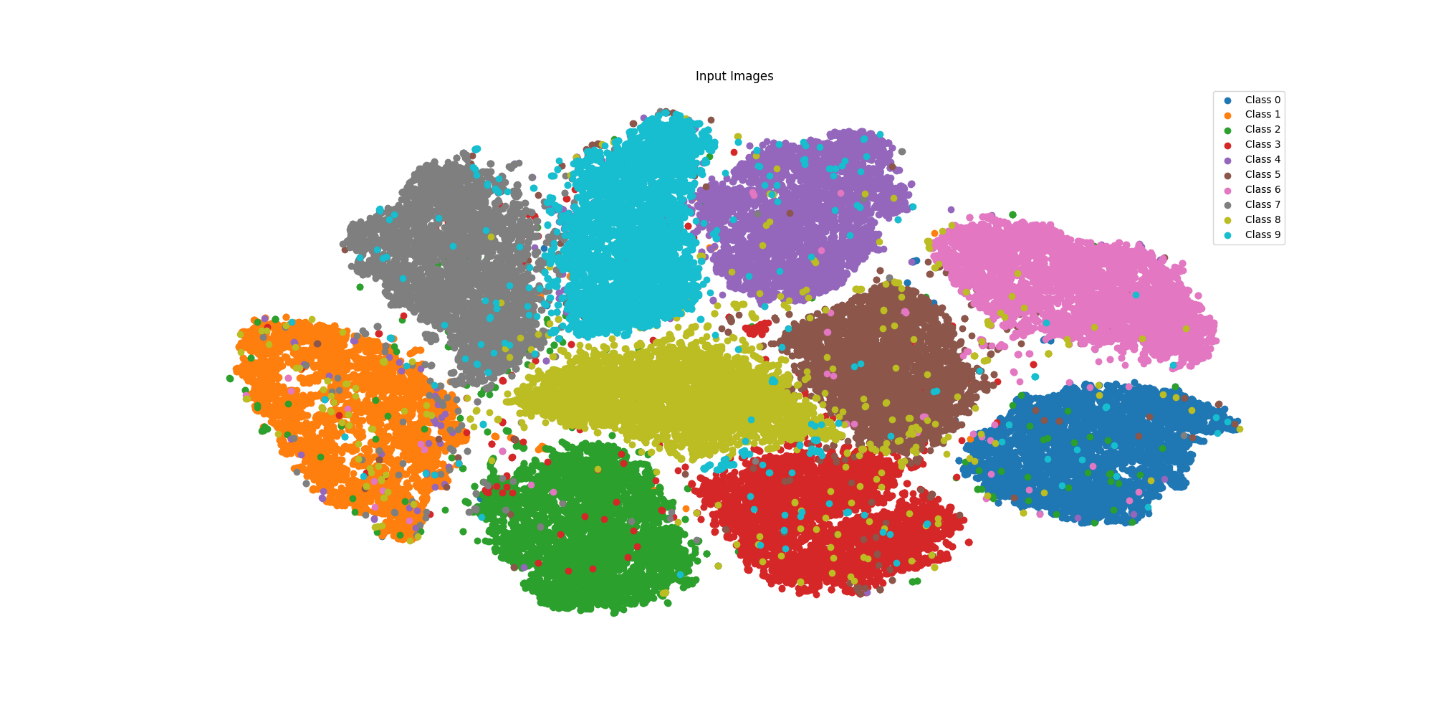
|  |  |
| --- | --- |
| **Batch size** | **Average test error** |
| 100 | 0.1504 |
| 1000 | 0.3059 |

|  |  |
| --- | --- |
| **Learning Rate** | **Average test error** |
| 0.01 | 0.1000 |
| 0.001 | 0.1419 |
| 0.0001 | 0.4425 |

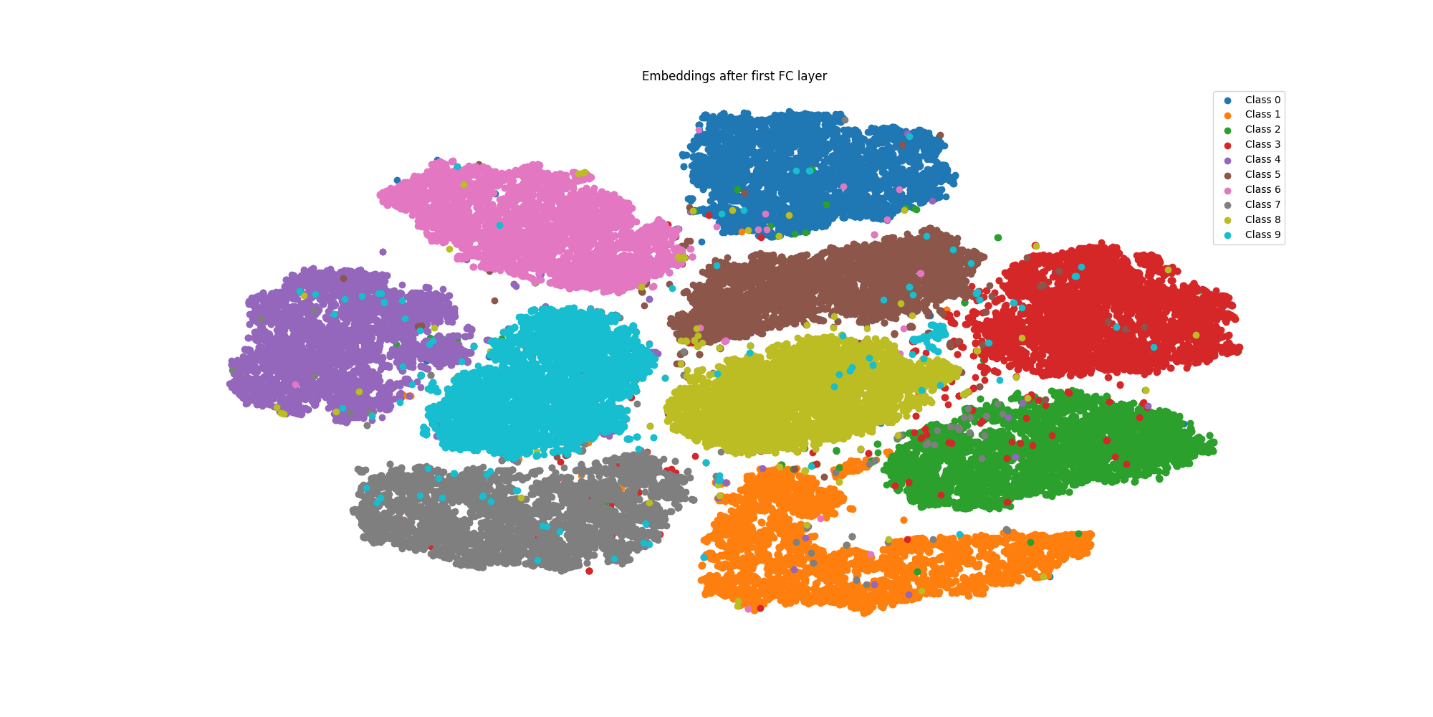
We can see that although it looks like batch size = 100 produce smaller test error by average, when combining it with hidden size and learning rate params it actually produce better results,   
justifying our use of grid search.

**Task 5**

The following plot displays the tSNE results when applied on the input:



The following plot displays the tSNE results of applying the first layer of our NN model on the input:



In general, both plots got good samples separation-by-label into distinct groups.

Therefore, we assume that other algorithms based on distance between samples may also produce relatively good results. Looks like classic ML models may also be able to produce good results here (maybe even K-Nearest-Neighbors). Of course, this is just a hypothesis that will need to be tested.

Moreover, we see that groups that were close in the first plot stay close in the second plot as well.   
For example, groups 3,5,8. Since these digits have some common “features” it makes sense that they will be shown close both in the original image and the result of the first NN layer.   
On the other hand, from qualitative point of view, the separation between these groups in the second figure is more distinct compared to the first figure.

This difference implies that after the first layer of neurons, each sample got "closer" to the other samples in its group and moved away from samples that were originally close to it from other groups.  
So we may say the layer of neurons from the model converted the original images to samples that "better-represents" the label while reducing the dimension from 784 dimensions to 500.

Another qualitative observation we found is that many images with the label “8” were originally close to the images with the labels “1”,”3”,”4”,”5” (according to the tSNE), while after the first NN-layer the amount of samples labeled “8” that are closer to samples with other labels decreased, meaning the 500-feature representation of NN was better representing the differences between images with “8” labels and images with “1”,”3”,”4”,”5” labels.

**Task 3 - Full results:**

|  |  |  |
| --- | --- | --- |
| **Epoch #** | **Validation Error** | **Test Error** |
| **Seed 1** | | |
| **Epoch 1** | 0.1687 | 0.1643 |
| **Epoch 2** | 0.1154 | 0.1024 |
| **Epoch 3** | 0.0984 | 0.0931 |
| **Epoch 4** | 0.0894 | 0.0837 |
| **Epoch 5** | 0.0796 | 0.0739 |
| **Seed 2** | | |
| **Epoch 1** | 0.1959 | 0.1736 |
| **Epoch 2** | 0.1224 | 0.1082 |
| **Epoch 3** | 0.1122 | 0.1010 |
| **Epoch 4** | 0.0893 | 0.0772 |
| **Epoch 5** | 0.0846 | 0.0762 |
| **Seed 3** | | |
| **Epoch 1** | 0.1904 | 0.1688 |
| **Epoch 2** | 0.1261 | 0.1084 |
| **Epoch 3** | 0.1031 | 0.0869 |
| **Epoch 4** | 0.0971 | 0.0804 |
| **Epoch 5** | 0.0887 | 0.0744 |
| **Seed 4** | | |
| **Epoch 1** | 0.1803 | 0.1597 |
| **Epoch 2** | 0.1244 | 0.1110 |
| **Epoch 3** | 0.1122 | 0.0982 |
| **Epoch 4** | 0.0984 | 0.0828 |
| **Epoch 5** | 0.0865 | 0.0710 |
| **Seed 5** | | |
| **Epoch 1** | 0.1820 | 0.1603 |
| **Epoch 2** | 0.1247 | 0.1081 |
| **Epoch 3** | 0.1044 | 0.0857 |
| **Epoch 4** | 0.0892 | 0.0757 |
| **Epoch 5** | 0.0809 | 0.0703 |