Assignment 2: Backpropagation

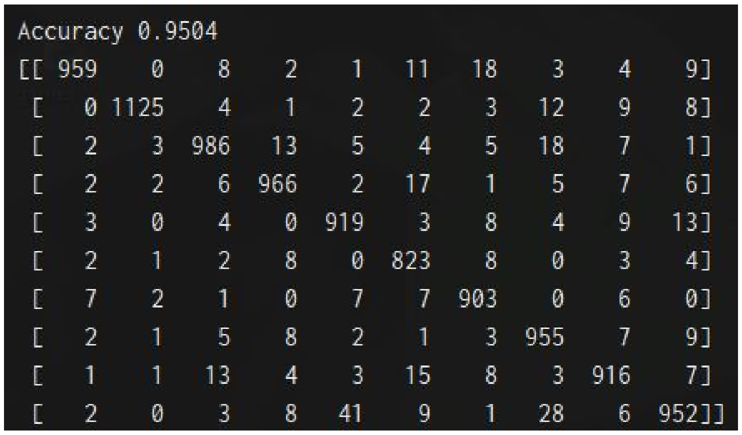
Duncan Clarke
20056561
COGS 400
October 21, 2020

1. Model 1

Confusion Matrix

		Actual Class									
		0	1	2	3	4	5	6	7	8	9
Predicted Class	0	959	0	8	2	1	11	18	3	4	9
	1	0	1125	4	1	2	2	3	12	9	8
	2	2	3	986	13	5	4	5	18	7	1
	3	2	2	6	966	2	17	1	5	7	6
	4	3	0	4	0	919	3	8	4	9	13
	5	2	1	2	8	0	823	8	0	3	4
	6	7	2	1	0	7	7	903	0	6	0
	7	2	1	5	8	2	1	3	955	7	9
	8	1	1	13	4	3	15	8	3	916	7
	9	2	0	3	8	41	9	1	28	6	952

Terminal output:



```
Accuracy 0.9504
[[ 959    0    8    2    1   11   18    3    4    9]
 [    0 1125    4    1    2    2    3   12    9    8]
 [    2    3  986   13    5    4    5   18    7    1]
 [    2    2    6  966    2   17    1    5    7    6]
 [    3    0    4    0  919    3    8    4    9   13]
 [    2    1    2    8    0  823    8    0    3    4]
 [    7    2    1    0    7    7  903    0    6    0]
 [    2    1    5    8    2    1    3  955    7    9]
 [    1    1   13    4    3   15    8    3  916    7]
 [    2    0    3    8   41    9    1   28    6  952]]
```

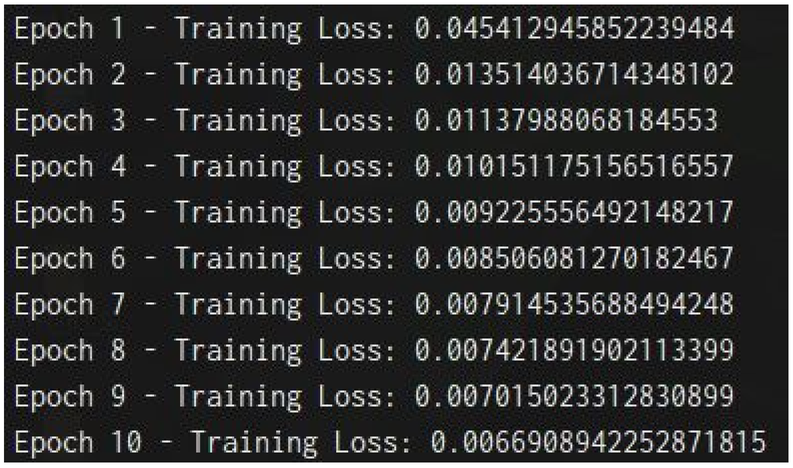
Precision and Recall

Number	TP	FP	FN	Precision	Recall
0	959	56	21	0.94482758620689655172413	0.97857142857142857142857142
1	1125	41	10	0.96483704974271012006861	0.99118942731277533039647577
2	986	58	46	0.94444444444444444444444	0.95542635658914728682170542
3	966	48	44	0.95266272189349112426035	0.95643564356435643564356435
4	919	44	63	0.95430944963655244029075	0.93584521384928716904276985
5	823	28	69	0.96709753231492361927144	0.92264573991031390134529147
6	903	30	55	0.96784565916398713826366	0.94258872651356993736951983
7	955	38	73	0.96173212487411883182275	0.92898832684824902723735408
8	916	55	58	0.94335736354273944387229	0.94045174537987679671457905
9	952	98	57	0.90666666666666666666666	0.94350842418235877106045589
			Average	0.95077805984865303806850	0.94956510327213632270602871

Results from Training and Testing Phases

Training

- Final Training Loss:
  0.0066908942252871815
  (Terminal output on the right)



```
Epoch 1 - Training Loss: 0.045412945852239484
Epoch 2 - Training Loss: 0.013514036714348102
Epoch 3 - Training Loss: 0.01137988068184553
Epoch 4 - Training Loss: 0.010151175156516557
Epoch 5 - Training Loss: 0.009225556492148217
Epoch 6 - Training Loss: 0.008506081270182467
Epoch 7 - Training Loss: 0.007914535688494248
Epoch 8 - Training Loss: 0.007421891902113399
Epoch 9 - Training Loss: 0.007015023312830899
Epoch 10 - Training Loss: 0.0066908942252871815
```

Testing

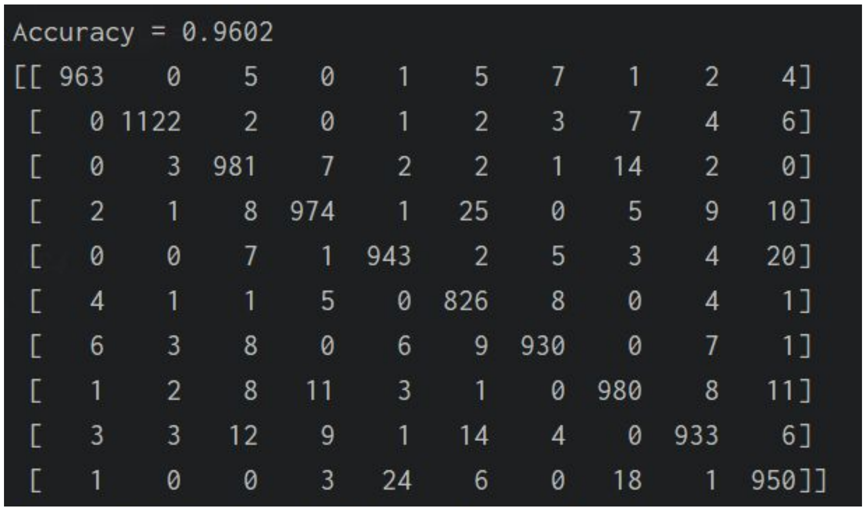
- Accuracy = 0.9504
  (Terminal output on previous page)

2. Model 2A

Confusion Matrix

		Actual Class									
		0	1	2	3	4	5	6	7	8	9
Predicted Class	0	963	0	5	0	1	5	7	1	2	4
	1	0	1122	2	0	1	2	3	7	4	6
	2	0	3	981	7	2	2	1	14	2	0
	3	2	1	8	974	1	25	0	5	9	10
	4	0	0	7	1	943	2	5	3	4	20
	5	4	1	1	5	0	826	8	0	4	1
	6	6	3	8	0	6	9	930	0	7	1
	7	1	2	8	11	3	1	0	980	8	11
	8	3	3	12	9	1	14	4	0	933	6
	9	1	0	0	3	24	6	0	18	1	950

Terminal output:



```
Accuracy = 0.9602
[[ 963    0    5    0    1    5    7    1    2    4]
 [   0 1122    2    0    1    2    3    7    4    6]
 [   0    3  981    7    2    2    1   14    2    0]
 [   2    1    8  974    1   25    0    5    9   10]
 [   0    0    7    1  943    2    5    3    4   20]
 [   4    1    1    5    0  826    8    0    4    1]
 [   6    3    8    0    6    9  930    0    7    1]
 [   1    2    8   11    3    1    0  980    8   11]
 [   3    3   12    9    1   14    4    0  933    6]
 [   1    0    0    3   24    6    0   18    1  950]]
```

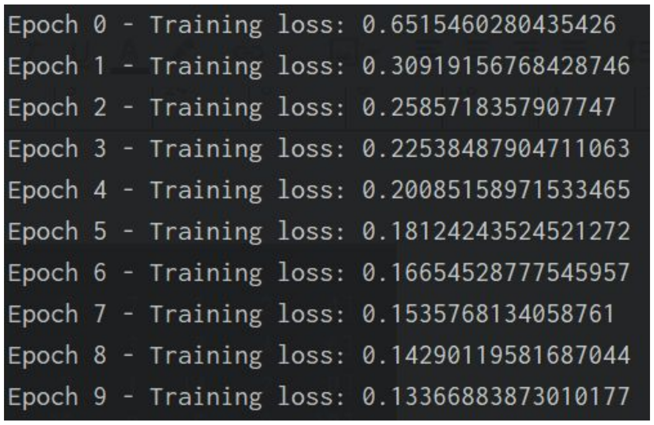
Precision and Recall

Number	TP	FP	FN	Precision	Recall
0	963	25	17	0.9746963562753036	0.9826530612244898
1	1122	25	13	0.978204010462075	0.9885462555066079
2	981	31	51	0.9693675889328063	0.9505813953488372
3	974	61	36	0.9410628019323671	0.9643564356435644
4	943	42	39	0.9573604060913706	0.9602851323828921
5	826	24	66	0.9717647058823529	0.9260089686098655
6	930	40	28	0.9587628865979381	0.9707724425887265
7	980	45	48	0.9560975609756098	0.9533073929961089
8	933	52	41	0.9472081218274112	0.9579055441478439
9	950	53	59	0.9471585244267198	0.9415262636273538
			Average	0.9601682963403954	0.959594289207629

Results from Training and Testing Phases

Training

- Final Training Loss: 0.13366883873010177
  (Terminal output on the right)



```
Epoch 0 - Training loss: 0.6515460280435426
Epoch 1 - Training loss: 0.30919156768428746
Epoch 2 - Training loss: 0.2585718357907747
Epoch 3 - Training loss: 0.22538487904711063
Epoch 4 - Training loss: 0.20085158971533465
Epoch 5 - Training loss: 0.18124243524521272
Epoch 6 - Training loss: 0.16654528777545957
Epoch 7 - Training loss: 0.1535768134058761
Epoch 8 - Training loss: 0.14290119581687044
Epoch 9 - Training loss: 0.13366883873010177
```

Testing

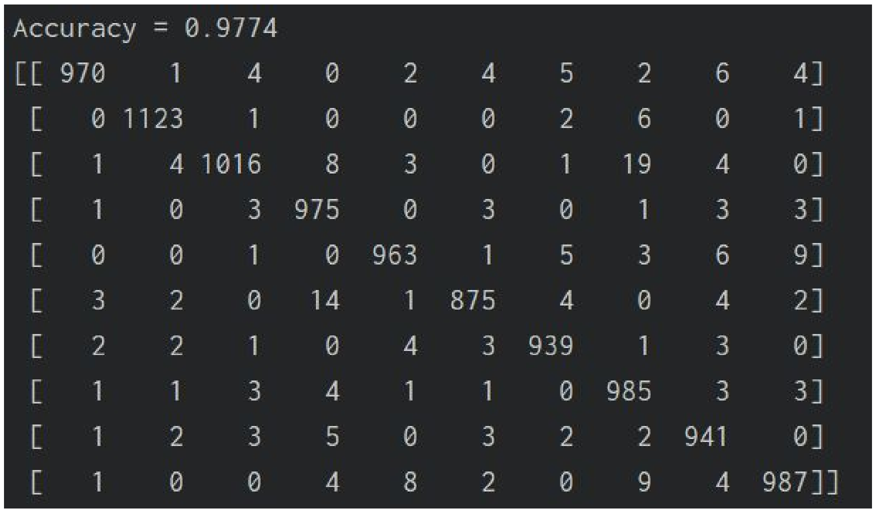
- Accuracy = 0.9602
  (Terminal output on previous page)

3. Model 2B

Confusion Matrix

		Actual Class									
		0	1	2	3	4	5	6	7	8	9
Predicted Class	0	970	1	4	0	2	4	5	2	6	4
	1	0	1123	1	0	0	0	2	5	0	1
	2	1	4	1016	8	3	0	1	19	4	0
	3	1	0	3	975	0	3	0	1	3	3
	4	0	0	1	0	963	1	5	3	6	9
	5	3	2	0	14	1	875	4	0	4	2
	6	2	2	1	0	4	3	939	1	3	0
	7	11	1	3	4	1	1	0	985	3	3
	8	1	2	3	5	0	3	2	2	941	0
	9	1	0	0	4	8	2	0	9	4	987

Terminal output:



```
Accuracy = 0.9774
[[ 970    1    4    0    2    4    5    2    6    4]
 [   0 1123    1    0    0    0    2    6    0    1]
 [   1    4 1016    8    3    0    1   19    4    0]
 [   1    0    3  975    0    3    0    1    3    3]
 [   0    0    1    0  963    1    5    3    6    9]
 [   3    2    0   14    1  875    4    0    4    2]
 [   2    2    1    0    4    3  939    1    3    0]
 [   1    1    3    4    1    1    0  985    3    3]
 [   1    2    3    5    0    3    2    2  941    0]
 [   1    0    0    4    8    2    0    9    4  987]]
```

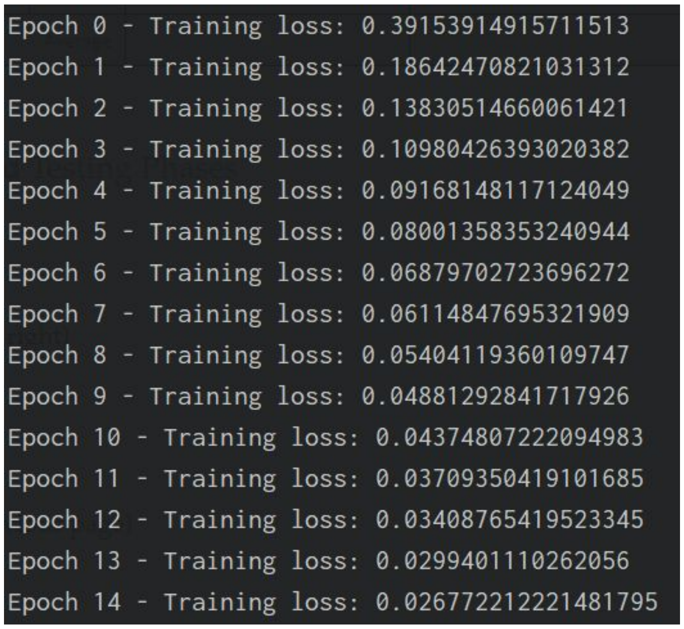
Precision and Recall

Number	TP	FP	FN	Precision	Recall
0	970	28	20	0.9719438877755511	0.9797979797979798
1	1123	9	12	0.9920494699646643	0.9894273127753304
2	1016	40	15	0.9621212121212121	0.98545101842871
3	975	14	35	0.9858442871587462	0.9653465346534653
4	963	25	19	0.9746963562753036	0.9806517311608961
5	875	30	17	0.9668508287292818	0.9809417040358744
6	939	16	19	0.9832460732984293	0.9801670146137787
7	985	27	42	0.9733201581027668	0.9591041869522882
8	941	18	33	0.9812304483837331	0.9661190965092402
9	987	28	22	0.9724137931034483	0.9781962338949455
			Average	0.9763716514913137	0.9765202812822509

Results from Training and Testing Phases

Training

- Final Training Loss: 0.026772212221481795
  (Terminal output on the right)



```
Epoch 0 - Training loss: 0.39153914915711513
Epoch 1 - Training loss: 0.18642470821031312
Epoch 2 - Training loss: 0.13830514660061421
Epoch 3 - Training loss: 0.10980426393020382
Epoch 4 - Training loss: 0.09168148117124049
Epoch 5 - Training loss: 0.08001358353240944
Epoch 6 - Training loss: 0.06879702723696272
Epoch 7 - Training loss: 0.06114847695321909
Epoch 8 - Training loss: 0.05404119360109747
Epoch 9 - Training loss: 0.04881292841717926
Epoch 10 - Training loss: 0.04374807222094983
Epoch 11 - Training loss: 0.03709350419101685
Epoch 12 - Training loss: 0.03408765419523345
Epoch 13 - Training loss: 0.0299401110262056
Epoch 14 - Training loss: 0.026772212221481795
```

Testing

- Accuracy = 0.9774
  (Terminal output on previous page)

4. Discussion

The first implemented model was built from scratch, outside of using the sklearn and pandas libraries in order to import the MNIST data and convert the desired outputs into one-hot vectors, respectively.

The structure of the neural network for this model involved 784 inputs, one for each pixel value in the MNIST data points, 64 hidden neurons in a single hidden layer, and 10 outputs, one for each possible classification. The model was trained over 10 epochs with a learning rate of 0.1 and a momentum of 0.9, because these parameters seemed to result in the lowest loss for this implementation, while keeping the training time reasonably low.

Initially, I was experiencing issues with this model, and the activation weights tended to be very large at every epoch (0.999+). This resulted in very low accuracy on the training data. The initialization of the weights was being done by randomly generating the initial weights using Numpy. After some experimenting, multiplying the initial weights by a constant value of 0.1 fixed the issue entirely. This allowed the model to randomly generate the weights, which is far more effective than initializing them all to 0, while also keeping the weight values reasonably small at initialization.

The second implemented model was done using the PyTorch machine learning library. An identical structure was used for this model as for the previous, with 64 hidden neurons in a single hidden layer, 10 epochs, a learning rate of 0.1, and a momentum of 0.9. This model achieved a higher accuracy than the first, despite having identical parameters. I can assume this is caused by the more optimized implementation of the model that comes with using a pre-built library rather than constructing the model from scratch.

The third and final implemented model was also done using PyTorch, and is almost identical from the second model aside from some tweaked parameters. This model used 128 hidden neurons rather than the 64 used in the previous model. 64 hidden neurons was originally used in the first model (and copied to the second) as to reduce the runtime of each epoch. Since PyTorch seems to run much more efficiently, runtime was not as much of an issue, so 128 hidden neurons were used. Additionally, the amount of epochs was modified to 15, and the learning rate was changed to 0.05 since these parameters seemed to result in higher accuracy on the test set.

The higher accuracy in the third model could be attributed to the increased amount of hidden neurons and lower learning rate. If the learning rate was significantly decreased below 0.05, there would likely be a failure to train. Similarly, if there were far more hidden neurons, the model might run into an issue of overfitting the data, which creates an inability to generalize to the test set and would result in low accuracy.