CISC 874

Assignment 1 – Backpropagation

Model 1

To start, I collected data on networks with different numbers of nodes in the hidden layer. The results are shown in Table 1 and Table 2. Table 1 contains the results of using unnormalized data with a biases of 0.5. Table 2 takes the best results from Table 1, and uses a bias of 1, which was shown to give better results, and compares the normalized and unnormalized data. The normalized data performed significantly better than the unnormalized data.

Table 1. Results of backpropagation with different numbers of nodes in the hidden layer.

Model 1 Results with Bias 0.5		
Hidden Nodes (bias = 0.5)	Accuracy on Test Data	Min Training Error
20	82%	0.14
30	85%	0.11
35	86%	0.13
40	86%	0.11
50	87%	0.11
60	88%	0.10
70	88%	0.10
80	88%	0.10
90	88%	0.10
100	81%	0.13

Table 2. Results of unnormalized and normalized data being trained on the same network.

Model 1 Results with Bias 1		
Hidden Nodes	Accuracy on Test Data	Min Training Error
60	89%	0.11
70	89%	0.09
60 (Normalized)	96%	0.01
70 (Normalized)	97%	0.01

While the accuracy shown was only on the test data, the final accuracy of the training data was very similar, typically around 1% better, with the exception of the 100 hidden nodes run, which had a much lower test accuracy, likely due to overtraining.

Figure 1 shows a confusion matrix in table form with actual numbers. The diagonal shows the true positives, as the predicted value and actual values matched up. The y-axis corresponds to the actual values, and the x-axis corresponds to the predicted values. Figure 2 shows a confusion matrix corresponding to the normalized data. All the following tables are for when 70 hidden nodes were used.

	0	1	2	3	4	5	6	7	8	9	Total
0	948	0	2	4	0	13	9	1	1	2	980
1	0	1102	3	3	2	4	1	1	19	0	1135
2	18	4	903	22	11	1	17	15	37	4	1032
3	2	2	19	904	0	32	5	15	20	11	1010
4	4	7	4	0	852	2	23	4	3	83	982
5	11	3	8	49	11	743	31	5	21	10	892
6	30	2	9	0	4	26	877	1	8	1	958
7	4	18	34	2	8	3	3	926	4	26	1028
8	15	4	8	32	6	41	11	12	813	32	974
9	11	11	0	13	39	20	5	22	4	884	1009
All	1043	1153	990	1029	933	885	982	1002	930	1053	10000

Figure 1. Confusion Matrix of Model 1 Neural Net's test data with 70 nodes in the hidden layer, bias of 1, and learning rate of 0.01.

Test Accuracy: 0.9616

	0	1	2	3	4	5	6	7	8	9	Total
0	967	0	1	1	2	3	3	2	1	0	980
1	0	1125	2	3	1	0	2	0	2	0	1135
2	11	1	976	9	7	2	6	11	8	1	1032
3	1	0	10	967	3	13	0	7	4	5	1010
4	1	0	3	0	946	1	9	1	5	16	982
5	6	0	0	7	2	861	6	1	6	3	892
6	10	3	2	1	5	13	920	0	4	0	958
7	3	6	18	4	8	2	0	967	1	19	1028
8	4	7	2	7	5	7	2	6	931	3	974
9	8	8	0	9	12	6	0	7	3	956	1009
All	1011	1150	1014	1008	991	908	948	1002	965	1003	10000

Figure 2. Confusion Matrix of Model 1 Neural Net's test data with 70 nodes in the hidden layer, bias of 1, and learning rate of 0.01, with normalized data.

For Model 1, Figure 2:

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

Numbers	True Positives	False Positives	False Negatives	Precision	Recall
0	967	13	44	0.986734693877551	0.9564787339268052
1	1125	10	25	0.9911894273127754	0.9782608695652174
2	976	56	38	0.9457364341085271	0.9625246548323472
3	967	43	41	0.9574257425742574	0.9593253968253969
4	946	36	45	0.9633401221995926	0.9545913218970736
5	861	31	47	0.9652466367713004	0.948237885462555
6	920	38	28	0.9603340292275574	0.9704641350210971
7	967	61	35	0.9406614785992218	0.9650698602794411
8	931	43	34	0.9558521560574949	0.9647668393782384
9	956	53	47	0.9474727452923687	0.9531405782652044

Figure 3. The calculated Precision and Recall values based on the table in Figure 2. Overall precision: 0.96, overall recall: 0.96

Numbers	True Positives	False Positives	False Negatives	Precision	Recall
0	5872	51	120	0.9913894985649164	0.9799732977303071
1	6677	65	110	0.9903589439335508	0.9837925445705025
2	5817	141	105	0.9763343403826787	0.9822695035460993
3	5979	152	151	0.9752079595498288	0.9753670473083197
4	5720	122	122	0.9791167408421774	0.9791167408421774
5	5287	134	76	0.9752813134108098	0.9858288271489838
6	5845	73	68	0.987664751605272	0.9884999154405547
7	6153	112	115	0.982122905027933	0.9816528398213146
8	5688	163	126	0.9721415142710648	0.978328173374613
9	5783	166	186	0.9720961506135485	0.9688390015077902

Figure 4. The calculated Precision and Recall values of the training data after the last epoch using the best weights. Overall precision: 0.98, overall recall: 0.98

Model 2

Table 3. Results of training and testing unnormalized data on a network built using Keras. Parameters were the default values excepting the number of nodes in the hidden layer

Model 2 Results (Data Unnormalized)		
Hidden Nodes (bias = 1)	Accuracy on Test Data	Test Error
20	91%	0.030
30	91%	0.029
40	93%	0.023
50	93%	0.024
60	94%	0.021
70	94%	0.021
80	94%	0.020
90	94%	0.021
100	93%	0.022
110	94%	0.019
120	94%	0.028

Table 4. Results of training and testing normalized data on a network built using Keras. Parameters were the default values excepting the number of nodes in the hidden layer

Model 2 Results (Data Normalized)		
Hidden Nodes (bias = 1)	Accuracy on Test Data	Test Error
20	95%	0.017
30	96%	0.013
40	97%	0.011
50	97%	0.010
60	97%	0.010
70	97%	0.009
80	97%	0.009
90	97%	0.009
100	97%	0.008
110	97%	0.009
120	97%	0.008

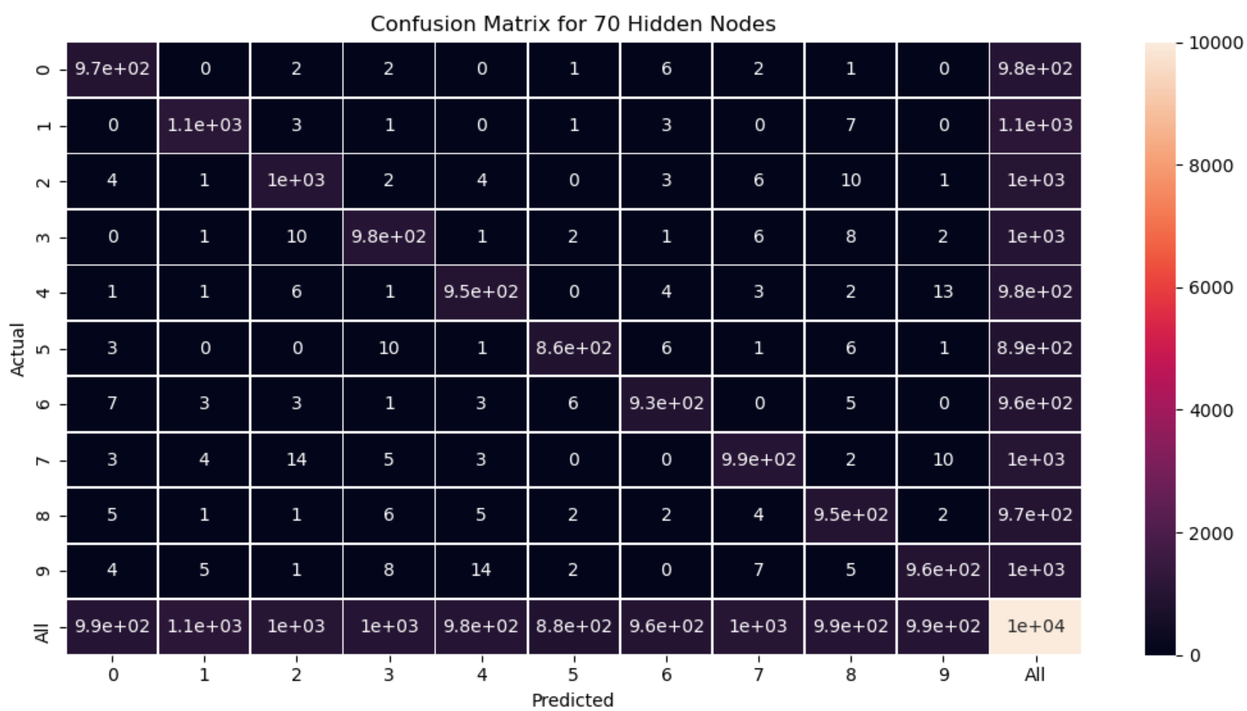

Figure 5. Confusion Matrix of Model 2 Neural Net with 70 nodes in the hidden layer, and all default options for the Adam optimizer, but with normalized data.

	0	1	2	3	4	5	6	7	8	9	Total
0	966	0	2	2	0	1	6	2	1	0	980
1	0	1120	3	1	0	1	3	0	7	0	1135
2	4	1	1001	2	4	0	3	6	10	1	1032
3	0	1	10	979	1	2	1	6	8	2	1010
4	1	1	6	1	951	0	4	3	2	13	982
5	3	0	0	10	1	864	6	1	6	1	892
6	7	3	3	1	3	6	930	0	5	0	958
7	3	4	14	5	3	0	0	987	2	10	1028
8	5	1	1	6	5	2	2	4	946	2	974
9	4	5	1	8	14	2	0	7	5	963	1009
All	993	1136	1041	1015	982	878	955	1016	992	992	10000

Figure 6. The tabular data corresponding to the heatmap in Figure 5.

Numbers	True Positives	False Positives	False Negatives	Precision	Recall
0	966	14	27	0.9857142857142858	0.972809667673716
1	1120	15	16	0.986784140969163	0.9859154929577465
2	1001	31	40	0.9699612403100775	0.9615754082612872
3	979	31	36	0.9693069306930693	0.9645320197044335
4	951	31	31	0.9684317718940937	0.9684317718940937
5	864	28	14	0.968609865470852	0.9840546697038725
6	930	28	25	0.9707724425887265	0.9738219895287958
7	987	41	29	0.9601167315175098	0.9714566929133859
8	946	28	46	0.971252566735113	0.9536290322580645
9	963	46	29	0.9544103072348861	0.9707661290322581

Figure 7. The calculated Precision and Recall values based on the table in Figure 6. Overall precision: 0.97, overall recall: 0.97

Numbers	True Positives	False Positives	False Negatives	Precision	Recall
0	5894	29	126	0.9951038325173054	0.9790697674418605
1	6672	70	80	0.9896173242361317	0.9881516587677726
2	5843	115	86	0.9806982208794898	0.9854950244560634
3	5967	164	145	0.9732506931984994	0.9762761780104712
4	5629	213	64	0.9635398836015063	0.9887581240119445
5	5310	111	110	0.9795240730492529	0.9797047970479705
6	5821	97	42	0.983609327475498 5	0.9928364318608222
7	6196	69	139	0.9889864325618516	0.9780584056827151
8	5695	156	121	0.9733378909588105	0.9791953232462174
9	5822	127	238	0.9786518742645823	0.9607260726072607

Figure 8. The calculated Precision and Recall values of the training data on the last epoch. Overall precision: 0.98, overall recall: 0.98

Momentum

The normalized data performed substantially better, so I only used normalized data to test the momentum.

Table 5. Results of normalized data on Model 1 with momentum.

Hidden Nodes	Accuracy on Test Data	Test Error
30	95%	0.03
50	96%	0.02
70	98%	0.02
90	97%	0.02

While the scores for momentum were about the same as the runs without (Table 2) the convergence seemed to be a bit faster, requiring about 20 epochs for the 70 and 90 hidden layer nodes, as opposed to 30 epochs.

Discussion:

As we can see from the results, normalizing the data vastly improves the accuracy of the neural networks. This makes sense, as the features stay mostly the same (in terms of location, stroke, shape), while reducing the impact of the inputs when it comes to pixel magnitude. Reducing the values to be within the range of 0 to 1 instead of 0 to 255 means that changes are smaller, so that abnormal features don't wildly distort the gradients.

I also initially ran the code with a bias of 0.5, and then with a bias of 1. The improvement was minimal, so I stuck with a bias of 1 without trying to test for different biases, since I got greater improvements by modifying other things.

The number of hidden nodes was one of the things I modified, and I tested a broad range of numbers. From 20 to as high as 120 in some cases, the difference in improvement tended to be small but still significant with every additional 10 hidden nodes up until around 100. At around 100 hidden nodes, while the training accuracy was still high, the testing accuracy tended to be the same or worse, so at this point I felt it was beginning to overtrain. For the Keras network this occurred at around 120 nodes instead. The number of input nodes and output nodes was decided based on the data, since the network needed an input node for every pixel in the image, and the output was to be 1 of 10 numbers. A learning rate of 0.1 was chosen for both models as it seemed to produce good results, and increasing the number by a little didn't have much impact on the scores. Increasing the learning rate by a lot worsened the scores, often resulting in fast by inaccurate results. Decreasing the number didn't improve the accuracy by much, but did seem to lengthen the time. This was the case for both models.

The highest test accuracy achieved without momentum was 97%, and this was achieved by both models. However, the Keras model was able to achieve this accuracy with fewer hidden layer nodes, requiring only 40 nodes, while the model built from scratch required 70. With the unnormalized data it can be seen that the Keras model had much higher accuracy than the model built from scratch, as the former attained accuracies in the 90s, while the latter only attained accuracies in the 80s.

Adding momentum improved accuracy to 98%, which is likely the highest the current models can get with this dataset. Setting a = 0.5 or higher resulted in 98% accuracies on the test data set, setting it much lower didn't always.