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 . 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 . 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.

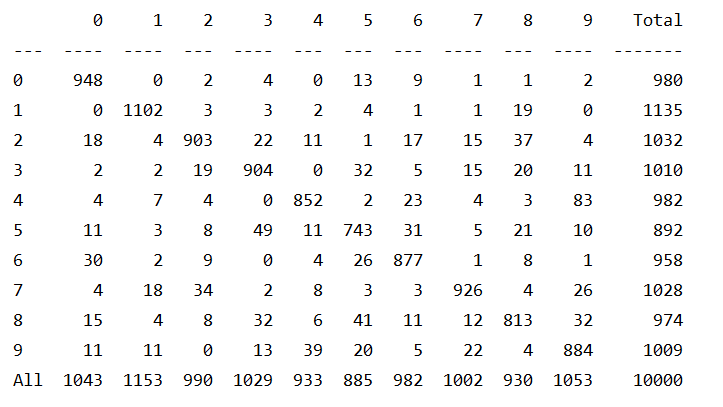


Figure . Confusion Matrix of Model 1 Neural Net’s test data with 70 nodes in the hidden layer, bias of 1, and learning rate of 0.01.

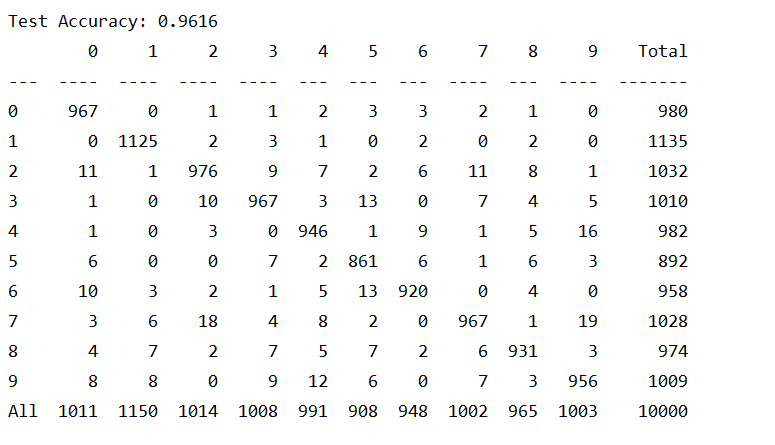


Figure . Confusion Matrix of Model 1 Neural Net’s test data with 70 nodes in the hidden layer, bias of 1, and learning rate of 0.01, with normalized data.

For Model 1, Figure 2:

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

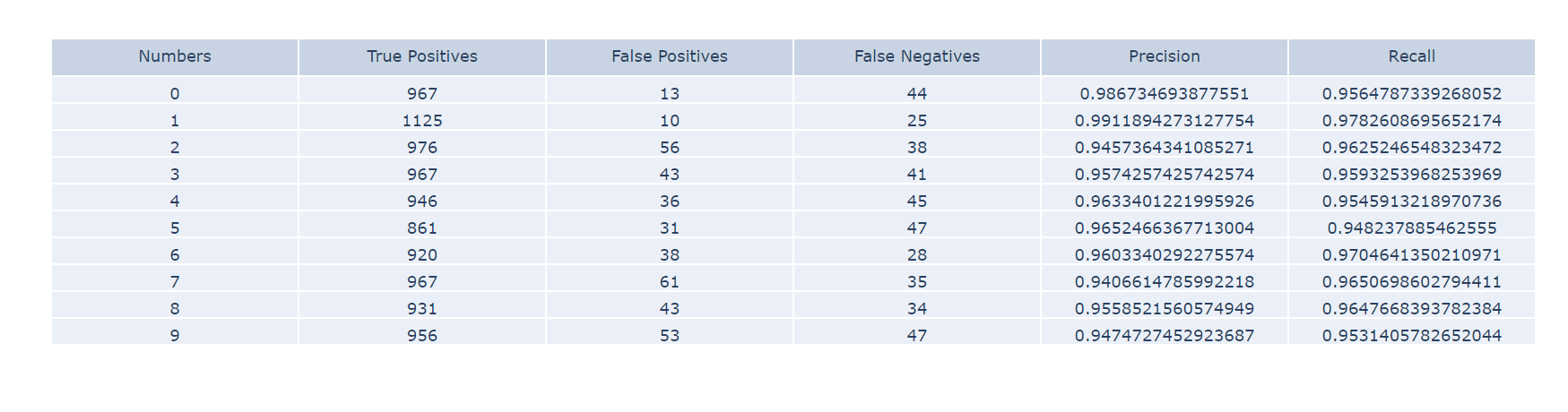
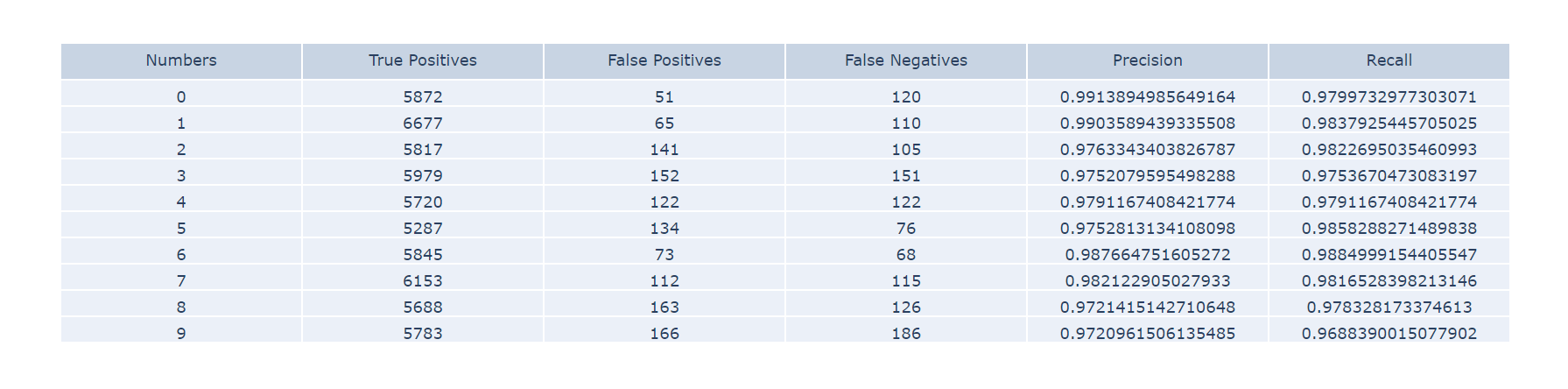
Figure 3. The calculated Precision and Recall values based on the table in Figure 2. Overall precision: 0.96, overall recall: 0.96

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 . 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 . 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 |

Calendar

Description automatically generated

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.

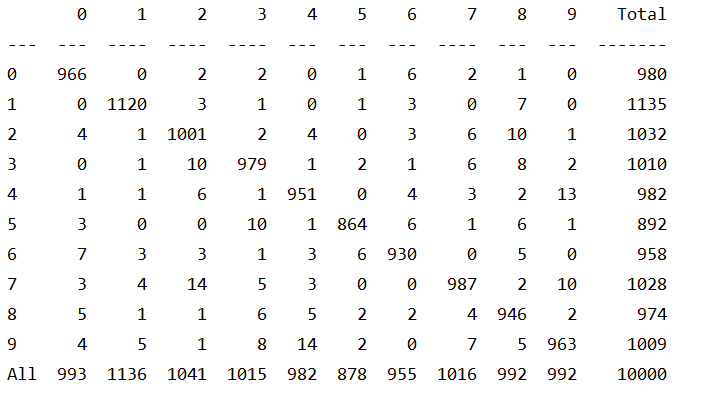


Figure 6. The tabular data corresponding to the heatmap in Figure 5.

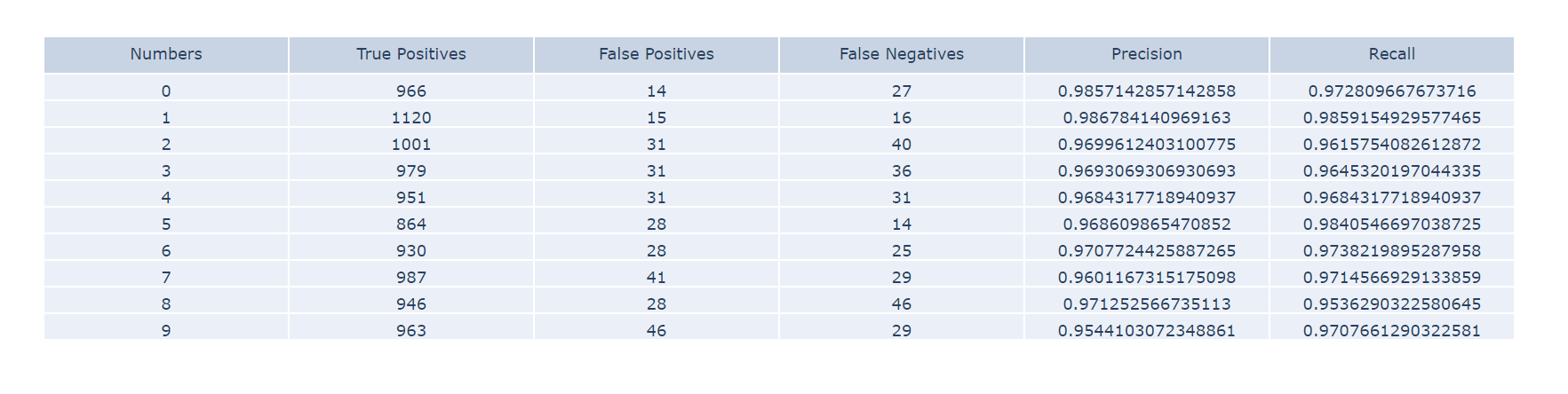
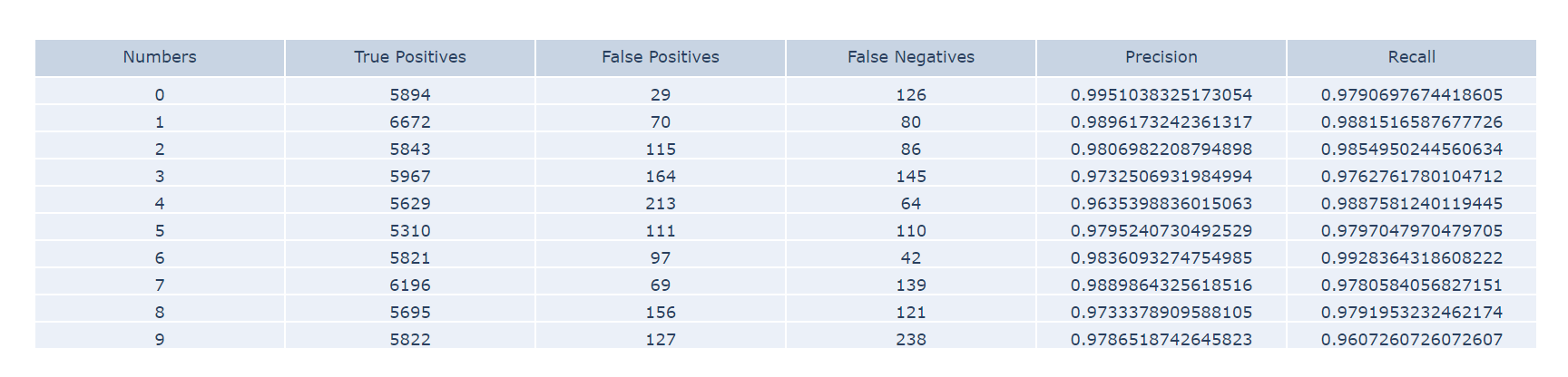
 Figure 7. The calculated Precision and Recall values based on the table in Figure 6. Overall precision: 0.97, overall recall: 0.97

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 . 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.