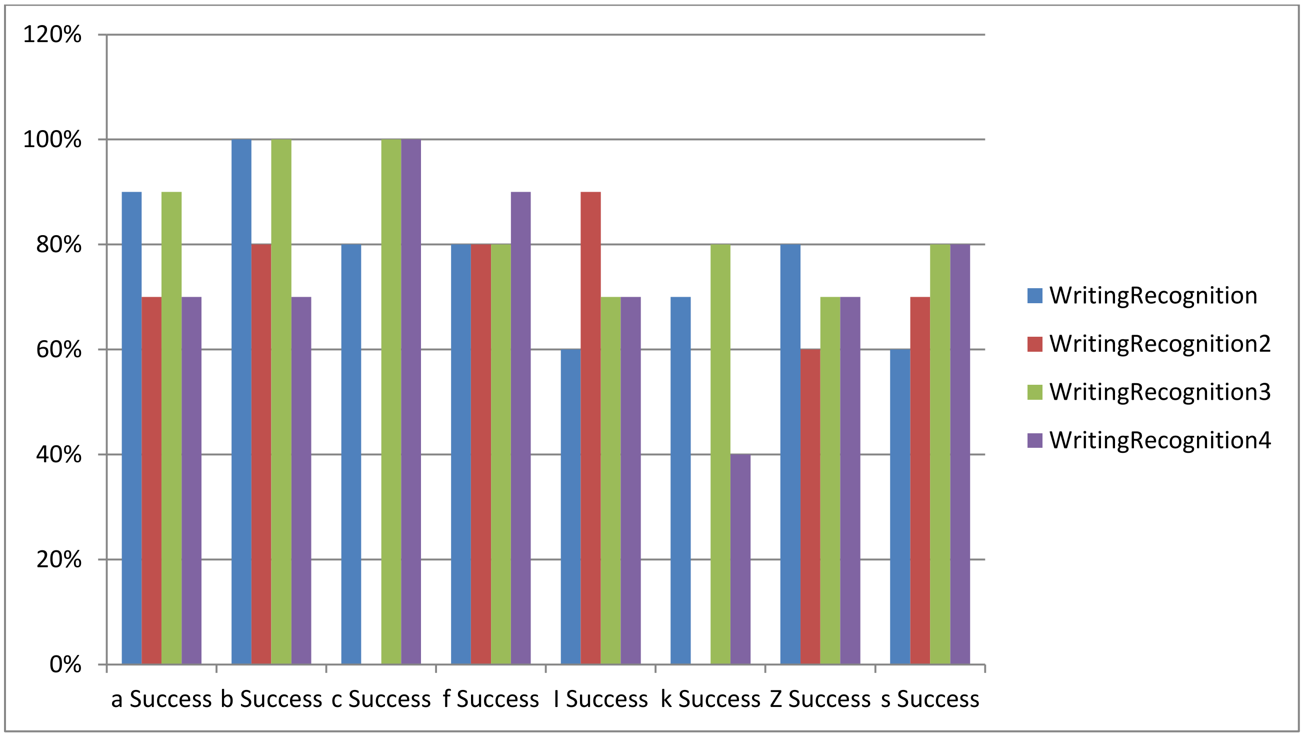
John Nickle

Handwriting Recognition

Our task was to create an AI that can recognize hand drawn symbols. Through extensive testing and training, I was able to create a variety of perceptrons that were able to recognize a respectable percentage of symbols. In all, my multilayer perceptron performed better than expected.

I created four subtly different multilayer perceptrons and compared the results. The symbols used were a, b, c, f, i, k, s, and z. After training each perceptron, each letter was tested individually in groups of 10. The results were as follows:



As you can see, overall the machines did not do too bad. However, there were a couple failed tests where WritingRecognition2's "c" and "k", were recognized 0%. Strangely enough, it did better than every other perceptron with the symbol "i", and did a fair job of competing with the others in every other symbol. The cause of this will be examined next.

Let’s take a look at each perceptron’s variables to explain why each is different in its abilities to recognize symbols. To start, all of the perceptrons were set up using 1600 input nodes, the same encoding and decoding, and 8 outputs. The only variables changed were the learning rate, amount of hidden nodes, and number of training iterations.
WritingRecognition had a learning rate of 0.05, 200 hidden nodes, and 10,000 training iterations.
WritingRecognition2 had a learning rate of 0.1, 50 hidden nodes, and 5,000 training iterations.
WritingRecognition3 had a learning rate of 0.1, 100 hidden nodes, and 10,000 training iterations.
WritingRecognition4 had a learning rate of 0.2, 200 hidden nodes, and 10,000 training iterations.
Examining the data, it seems that the amount of training iterations could have lead to the shortcomings of WritingRecognition2, along with a small number of hidden nodes.
WritingRecognition3 did the best overall with a learning rate of 0.1 and 100 hidden nodes. This seemed to be the sweet spot for my machine learning, as the results of further testing using an extreme number of hidden nodes (8 to 1,000) and a large learning rate (0.5) were disastrous.

10,000 training iterations seemed excessive at first, but it showed to be a great number in regards to accuracy and became the base of my first WritingRecognition. More iterations did not seem to improve the accuracy, however I did not test this extensively. A learning rate of 0.1 also appears to be a good number. Along with a decent amount of hidden nodes, I can see why a low learning rate would be beneficial in inching towards the ideal weights to produce the correct outputs.

Overall this project was a great success. It performed much better than I expected. Given more time I would try to devise a way to improve the inputs given to the perceptron in hopes that this would further improve results.