Project 2: Artificial Neural Network

General Approach

The goal of the project was to create a back propagate neural network to recognize handwritten numbers from zero to nine. My network uses a linear activation function for the input nodes, and a sigmoid for the rest. Each layer besides the input layer contains a bias node.

Methods

I used C++ and it's STL for my network.

Input

The csv parser reads the training and test data by using a custom locale which marks commas as whitespace. Using this parsing technique greatly simplifies the code for reading handwriting samples, improving readability.

Structure

The Network class choreographs sequences of Layer events and processes the max of the output nodes. The Layer class does most of the network initialization as well as normalizing inputs and outputs. The Cluster class represents a neuron and all its forward and backward edges. It processes forward feeding, activation, and backwards propagation. The Edge class is a simple container with origin and destination pointers, and a weight.

Sticking Points

Initially, I had some trouble with copy constructors and destructors, but this obstacle was passed soon. Later, I was stuck with my network sticking at exactly 10% accuracy for some time. To fix this error, I found various smaller bugs which offered no help on the current problem. However, these minor bugfixes helped the project in the end. After struggling with this problem, I finally discovered I was missing a minus sign in my Sigmoid function. After fixing the typo, my network achieved 92% accuracy immediately.

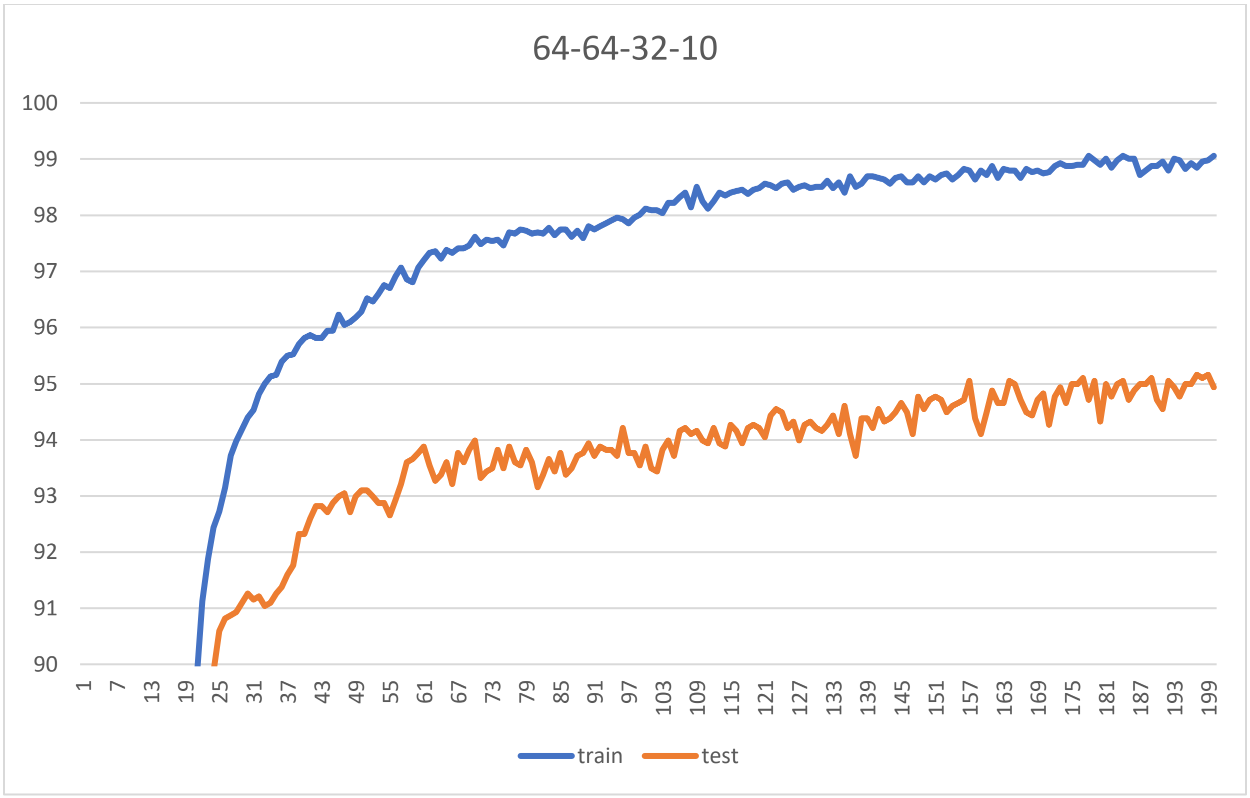
Experiments

These experiments detail the dimensions of a given network by [Input]-[Hidden1]-[Hidden2]-[Output].

All my experiments run to 200 Epochs at a 15% learning rate unless otherwise stated.

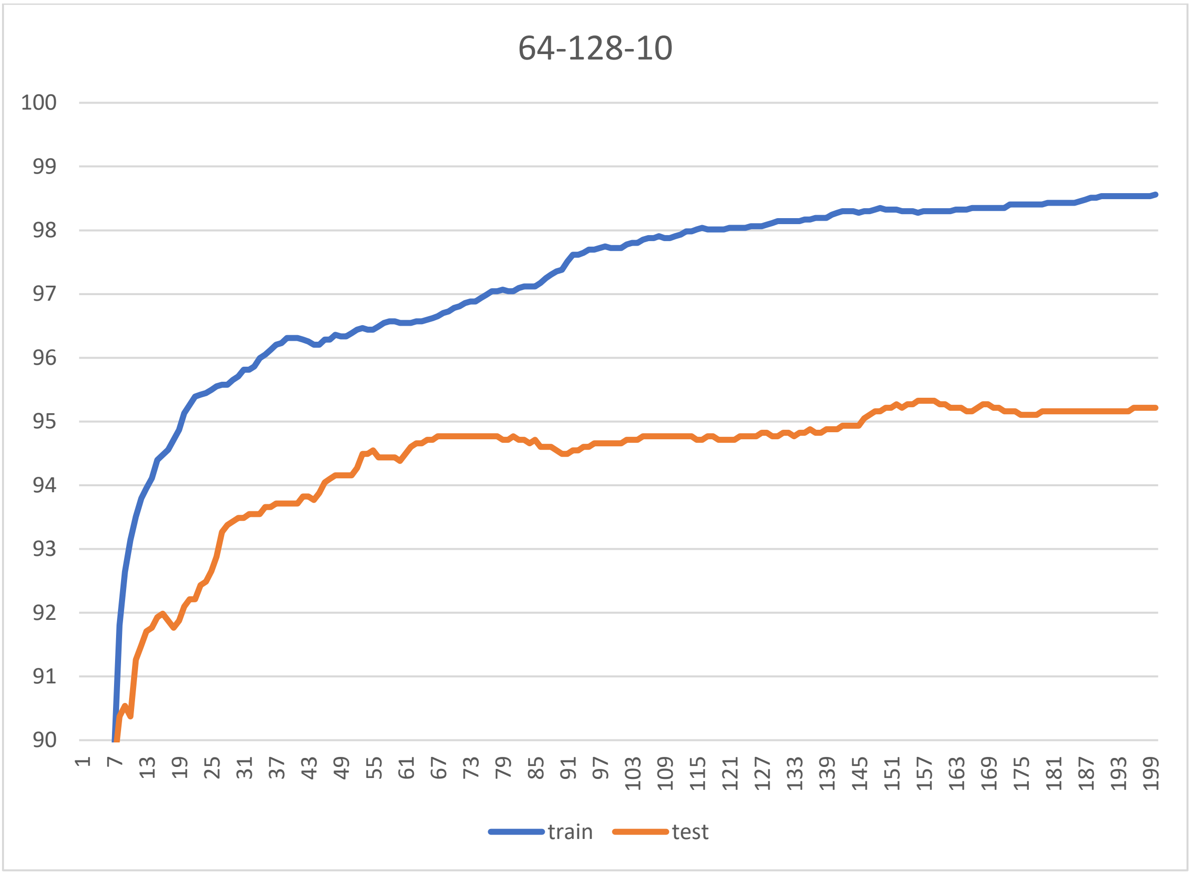
64-64-32-10

The first experiment was a network of size 64-64-32-10. I decided to start with two hidden layers that gradually decreased in size. After 20 epochs, this network crossed the 90% train accuracy threshold. Shortly after epoch 25, the test accuracy also crosses the 90% threshold, marking a delay of 5 epochs. The training set idles at 99% accuracy while the test set idles at 95% accuracy. The gap between these accuracies is 4%. The training set exhibited local variances of around 0.5% and the test set exhibited local variances of around 1%.



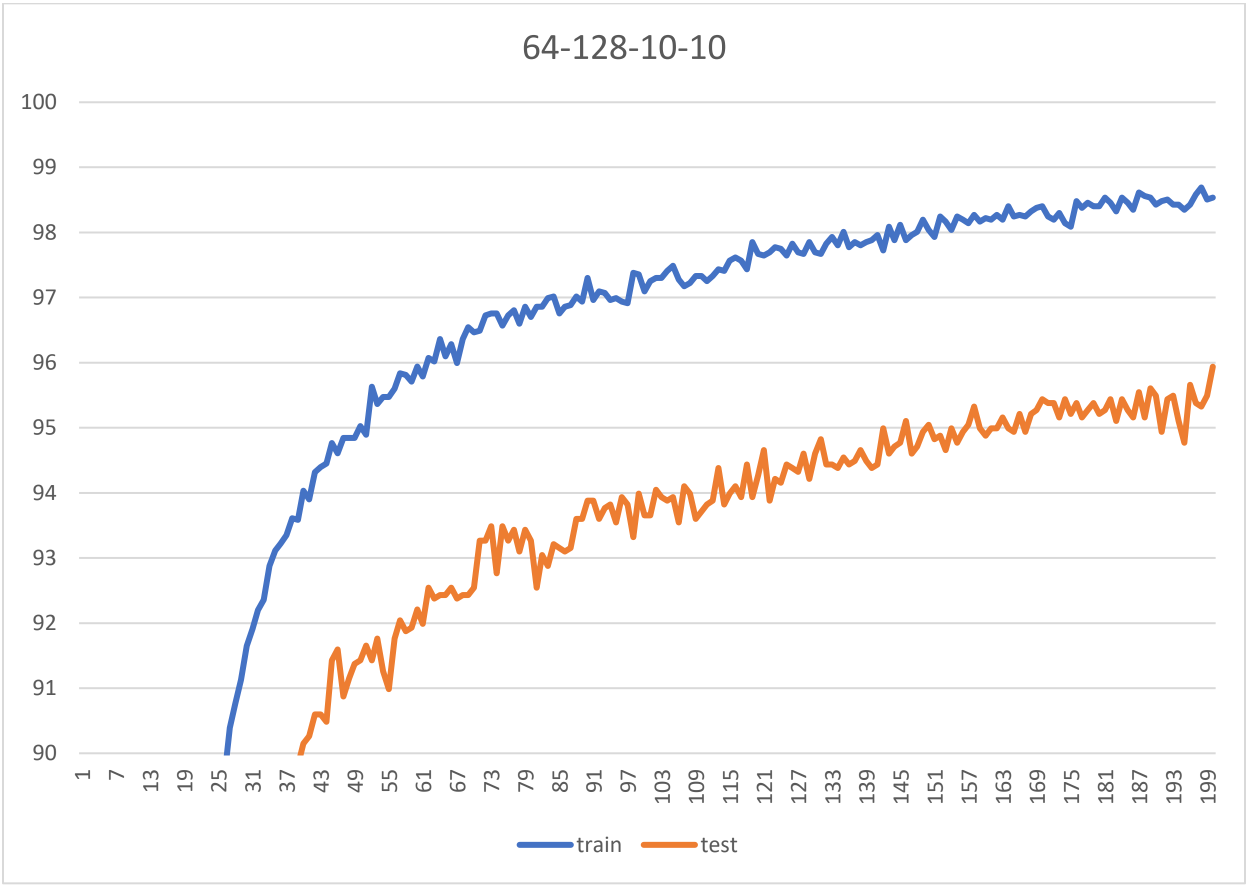
64-128-10

The second experiment was a network of size 64-128-10. I decided to try growing a layer's size beyond the input layer size. After 6 epochs, this network crossed the 90% train accuracy threshold. Shortly after epoch 7, the test accuracy also crosses the 90% threshold, marking a delay of 1 epoch. The training set idles at 98.6% accuracy while the test set idles at 95.3% accuracy. The gap between these accuracies is 3.3%. The training set exhibited local variances of around 0.1% and the test set exhibited local variances of around 0.2%.



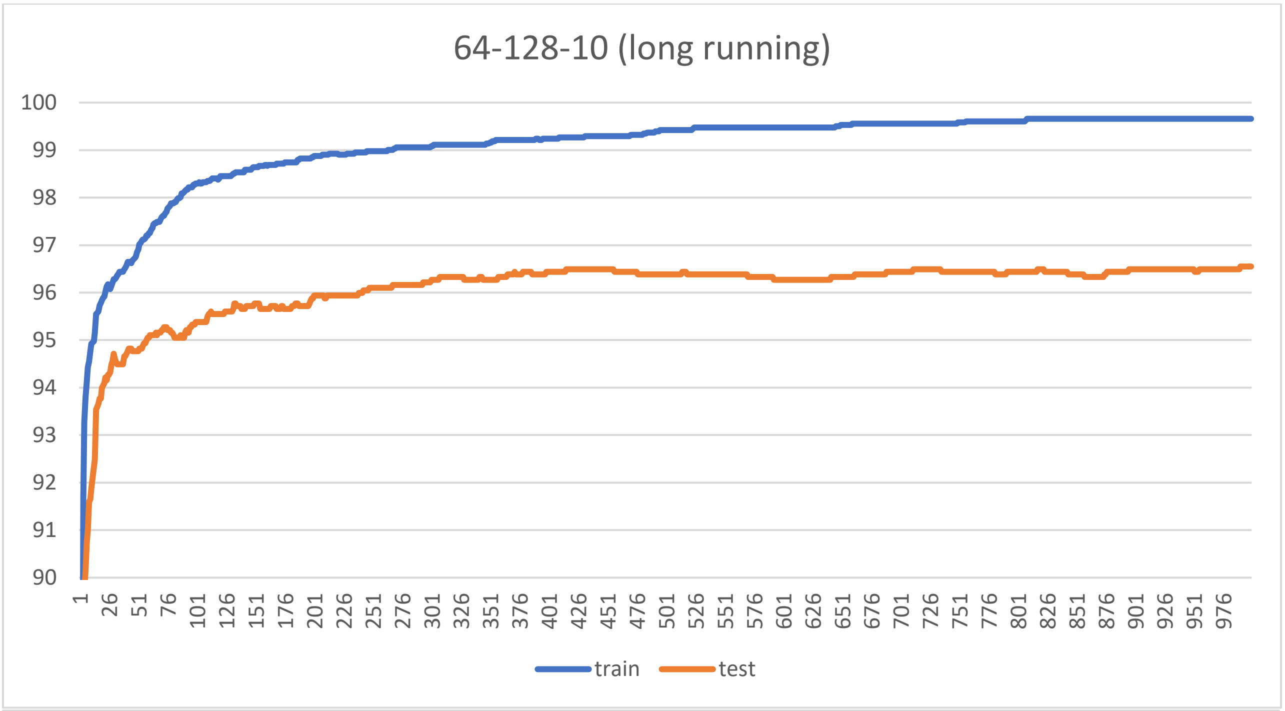
64-128-10-10

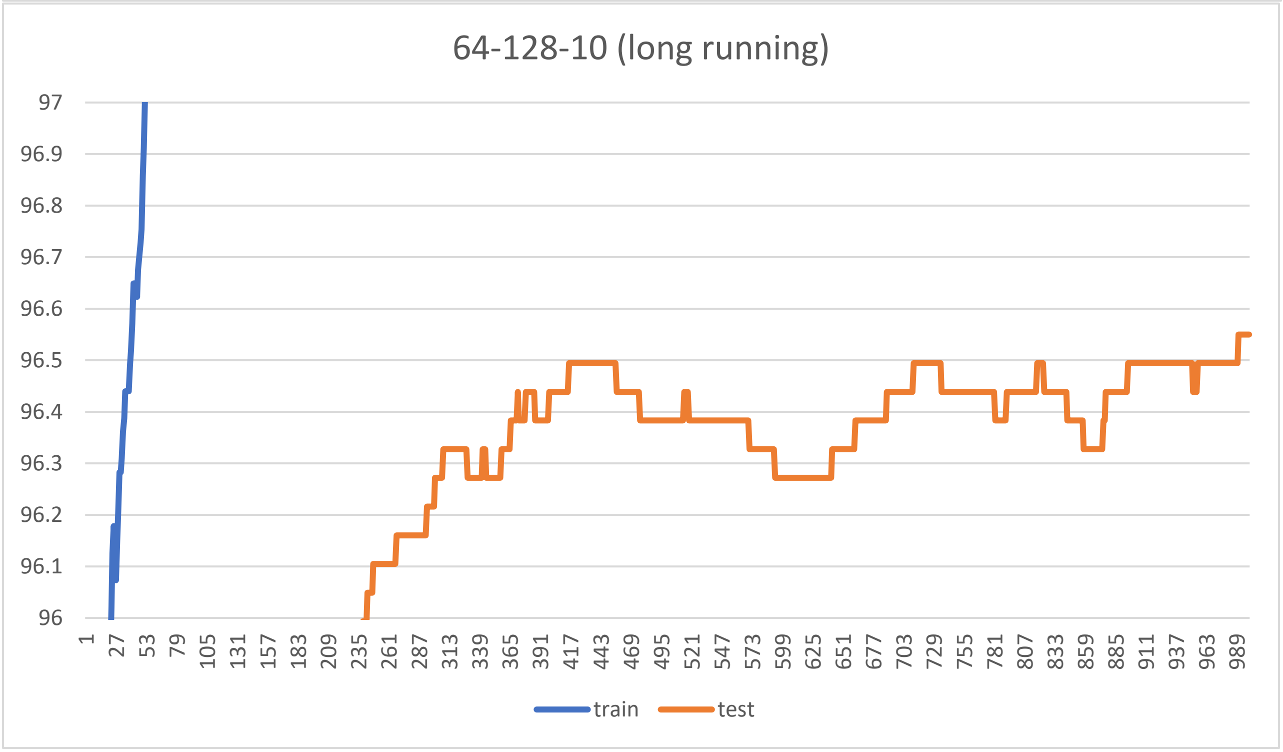
The second experiment was a network of size 64-128-10. I decided to insert an extra layer with a size identical to the output layer. My hope was for this to decrease variances. After 27 epochs, this network crossed the 90% train accuracy threshold. Shortly after epoch 38, the test accuracy also crosses the 90% threshold, marking a delay of 11 epochs. The training set idles at 98.3% accuracy while the test set idles at 95.2% accuracy. The gap between these accuracies is 3.1%. The training set exhibited local variances of around 0.5% and the test set exhibited local variances of around 1.1%.



Long running 64-128-10 with variable learning rate

After seeing the success of the 64-128-10, I decided to run it until my patience ran out. I went to about 1,000 epochs. To increase the learning value of each epoch, I kept the learning rate high for low accuracies on the training set, but lowered the learning rate as the accuracy increased. Given enough time, the network continues increasing the max accuracy. At 1000 epoch, the training set reached 99.6% accuracy and the test set reached 96.6% while maintaining a gap of 3%.





Analysis

Other than the training and test accuracy, I used various measurements to determine a network's effectiveness. I used the number of epochs for the test accuracy to reach 90% as the training rate. The number of epochs between the training accuracy and test accuracy reaching 90% measures the generalization delay. After both accuracies plateau, their difference is used to evaluate generalizability of the network. Additionally, local variances are measured to determine the precision of the network.

According to my experiments I have found that using fewer layers leads to more precision and generalizability, while using larger layers leads to more training rate and accuracy. The 64-128-10 achieves the best balance of these metrics. When used with a variable learn rate and a longer run cycle it's generalizability surpasses the 64-128-10-10 network while maintaining its high precision.