For my final in this course, I have decided to try and completely create a neural network from scratch. My goal was pretty basic, recognize hand written numerical digits. For this task, one of the hardest aspects was finding some sort of a database of which to train the network off of. After a lot of research, I came across something known as the MNIST database – it is a completely free, open, database of 10s of thousands of handwritten digits with a resolution of 28x28.

From there, I knew the resolution I was going to base this off of, now I just needed to do some research on different types of networks, and which one was the right one for me to use. I eventually came across convolutional neural networks (CNNs). These are generally used for image recognition and are comprised of the input layer, the output layer, and hidden layers in between. To make this network as optimized as possible, you would need enough layers and neurons (one of the values inside of the layer) inside each layer in order to function, but without having too many variables slowing the network down. The values of the neurons are between 0 and 1.

The value of a neuron in a proceeding layer is determined by the sum of each of the neurons in the current layer being multiplied a completely random weight (to show how much of an impact that neuron gives to the current neuron) and a bias added at the very end to account for tendencies of that neuron. After that, a function is applied to that neuron due to the reason each neuron must be between 0 and 1. For this a logistic function (more specifically a sigmoid) was used; this meant the higher the number the closer to one, the lower, the closer to 0.

The neural network now looked a little something like this in layers: 784 – 196 – 49 – 10. This meant there were ~170,000 different variables. Now in order to save all of these, I made 6 different save files, 1 one for each bias array and weight matrix. There were also many classes so that different aspects could be used by different applications. Setup Basis was used in order to populate the variables with random digits to go off of. The neural network class housed many vital functions for the network – back-propagation, network run, and file read. The train application used all 3 of these functions to read the data, run the function using a completely random image, and back propagate through. In order to actually read the images in an efficient manner, the images had to be stored in an MNIST data format. I used to other classes that worked together to read the data and return it. There was also a user draw application that allowed the user to draw a number, read the data, and then calculate what the final answer was.

One of the issues that came up was layers shrinking too fast. This caused the same output or 2 outputs to come out, which is why I ended up with the neuron sets that I did. Another challenge was creating a back-propagation algorithm completely from scratch, it took a lot of research and trial and error. The algorithm I ended up with works in theory but because of the a certain statistical fact, there was an issue; in 1/10 cases, an output neuron should be one, but in 9/10 it should be zero. This caused all of the outputs to be very close to zero and be extremely similar, leaving the cost function to be left at around 0.99, and stay at that almost no matter what.