Neural Networks for Digit Recognition

Richardo Hopkins, Luis Rosias, Amy Sham

Abstract

We studied the effectiveness of single-layer feedforward neural networks, commonly known as "perceptrons," in recognizing digits represented in three input formats. We created a perceptron with three different input representations and three different output representations, each with differently-sized layers. The goal of this project was to determine the optimal input/output pairing.

1 Introduction

In computer science, neural networks are mathematical models inspired by biological neural networks. Neurons are modeled by nodes, which are simple input-output units. These nodes process information as a connected network.

1.1 Perceptrons

Here, the perceptron, or the single-layer feedforward neural network consists of a layer of input nodes and a layer of output nodes, where every input node is connected to every output node. Each connection is associated with a weight. Each output node receives, as an input, the dot product (a weighted sum) of the values of its connected input nodes and their corresponding weights. The dot product is then fed into an activation function, which determines the final output value. The network then compares the final output value with the expected output value and adjusts the weights according to a learning rule.

1.2 Learning

In general, weights are adjusted according to this learning rule:

$$W_j = W_j + (A \times I_j \times Err \times g'(in_i))$$

Where W_j : weight to node j
A : learning rate
I_j : input node value to node j
Err : difference between output value and expected value
in_j : weighted sum of inputs to node j
g' : derivative of the activation function.

The weights are adjusted for all connections between input and output nodes for a given number of training cycles over an entire training set, also known as an "epoch."

2 Testing Methodology

For simplicity's sake, we implemented a single-layer perceptron (with no hidden layers). To test our perceptron's performance, we compared each input/output pair's final output value ("classification") to the expected output value.

We implemented three different input representations: (1) a 32×32 bitmap of 0's and 1's representing the digit's image ("bitmap" input); (2) an 8×8 down-sampled image of the digit ("real" input); and (3) a normalized version of the 8×8 down-sampled image ("integer" input). We implemented three different output representations: (1) a single node, whose classification is its output value is multiplied by 10 and rounded to the nearest integer ("simple" output); (2) four nodes, whose rounded values are interpreted as the binary representation of the classification ("binary" output); and (3) 10 nodes, each representing a digit, where the classification is taken as the digit corresponding to the node with the maximum value ("max" output).

For each input representation, we fed a different set of files into the perceptron (see above). Each set of files contained the

representation of an arbitrary set of digits and their expected values.

We tested each of the nine input/output representation pairs by tracking the (1) percentage of correct output classifications and (2) the mean squared error ("MSE") while adjusting (1) the number of training epochs: 1, 10, 25, and 50; and (2) the learning rate: 0.01, 0.05, 0.1, 0.5, 1.0. As default parameters, we set the size of the training epoch to 25 and the learning rate to .5.

To account for variability between samples, we used three perceptrons to produce the data for the pairings using the binary and max output representations and 50 perceptrons to produce the data for the pairings using the simple output representation. In all cases, we averaged the outputs over the number of perceptrons used. We hard-coded all of this averaging into the implementations. Future testers can adjust the number of perceptron objects used by changing a very small piece of the code in NeuralNetTester.

3 Experiments and Results

We used the percentage of correct output classifications as the primary metric for determining performance.

Our experiments yielded conclusive results. Certain input/output pairings emerged with consistently high performance, while other pairings emerged with consistently low performance. Output representation played a much more significant role in these differences in performance than did input representation, regardless of epoch size and learning rate. Therefore, the influential role of the output representations separated the performance percentages into three groups, each based on a different type of output representation (Figure 1, Figure 2).

The max output representation was by far the most successful, with correct output percentages ranging from 87% to 94% across all experiments. The binary output representation came at a far second, with correct output percentages averaging around the mid- to upper-60th percentile. The simple output representation was comparatively abysmal, as its correct output representations averaged well below the 10th percentile (with a few outliers above, none of which exceeded 15%).

Patterns with regard to the input representations were less noticeable, but we can still draw some conclusions from the data. Figure 1 shows that, among the pairings using max output representation, the real input representation produced the best results. The bitmap input representation closely followed, while the integer input representation consistently performed below the other two input representation types.

For the binary output representation, this pattern changes. The bitmap input representation performs best overall, and the real and the integer input representations vary, with no clear optimal configuration, in performance across either size or learning rate.

For the simple output representation, performance varied greatly across all pairings, with no clear pattern emerging across either epoch size or learning rate based on the percentage correct graphs.

3.1 Epoch Size

A marked increase occurred in the correct output percentages across all input/output representation pairings when the number of training epochs increased from 1 to 10. Thereafter, improvement leveled off; in fact, for most pairings, performance decreased slightly from 50 training epochs to 100 training epochs (Figure 1). This decrease is likely due to overtraining.

The exceptions to this trajectory were the input/output pairings using the down-sampled input representations and the simple output representation (i.e., two pairings). Their output percentages were fairly random despite increases in epoch size. The MSE data showed this result as well; the simple output representations had high (but stable) MSE

values, which indicates high variability in the results despite increases in the epoch size (Figure 3). Figure 1 does not illustrate this high variability, as the data it shows is an average over 50 test runs; however, we can conclude that no change in correct output percentage occurs as a result of epoch size increases. This result is consistent with the MSE findings.

3.2 Learning Rate

Overall, changing the learning rate did not significantly affect performance. In most cases, performance fluctuated slightly (within the lower 90^{th} percentile for the input/output pairings using max output representation and within the upper 60^{th} and lower 70^{th} percentiles for the pairings using binary output representation) as learning rates increased (Figure 2).

For the pairings on which changing the learning rate did affect performance, the correct output percentages were highest with the lowest learning rates: (.01 and .05). These changes were most apparent with the input/output pairings using the simple output representation (Figure 2). When paired with the simple output representation, the bitmap and real input representations performed best with the lowest learning rate and decreased as the learning rate increased. Performance from the integer input representation remained the same across all learning rates.

4 Conclusion

The results strongly indicate that output representations played a far more significant role in determining the perceptron's performance.

Input/output pairings using max output representation performed best, followed by those using binary representation, and then followed by those using simple representation, with large gaps separating each representation type.

The input representation's effect on performance varied depending on the pairing, but the overall effect of changing the input representation type was far less pronounced than that of changing the output representation type.

Increasing the number of training epochs used decreased the rate at which performance increased; improvement slowed down at an exponential rate.

Except for extreme cases, changing the learning rate did not affect performance. The lowest learning rates produced the best correct output percentages. In most cases, each input/output pair had an optimal learning rate within the range of learning rates we tested; the true optimal learning rate for each input/output pair likely lies within the range of learning rates.

4.1 Best Pairing?

The pairing of the real input representation and the max output representation consistently produced the best results across all epoch sizes and learning rates; however, we would not consider it a downgrade to use any of the other pairings with max output representation.

Figure 1: (learning rate = 0.5)

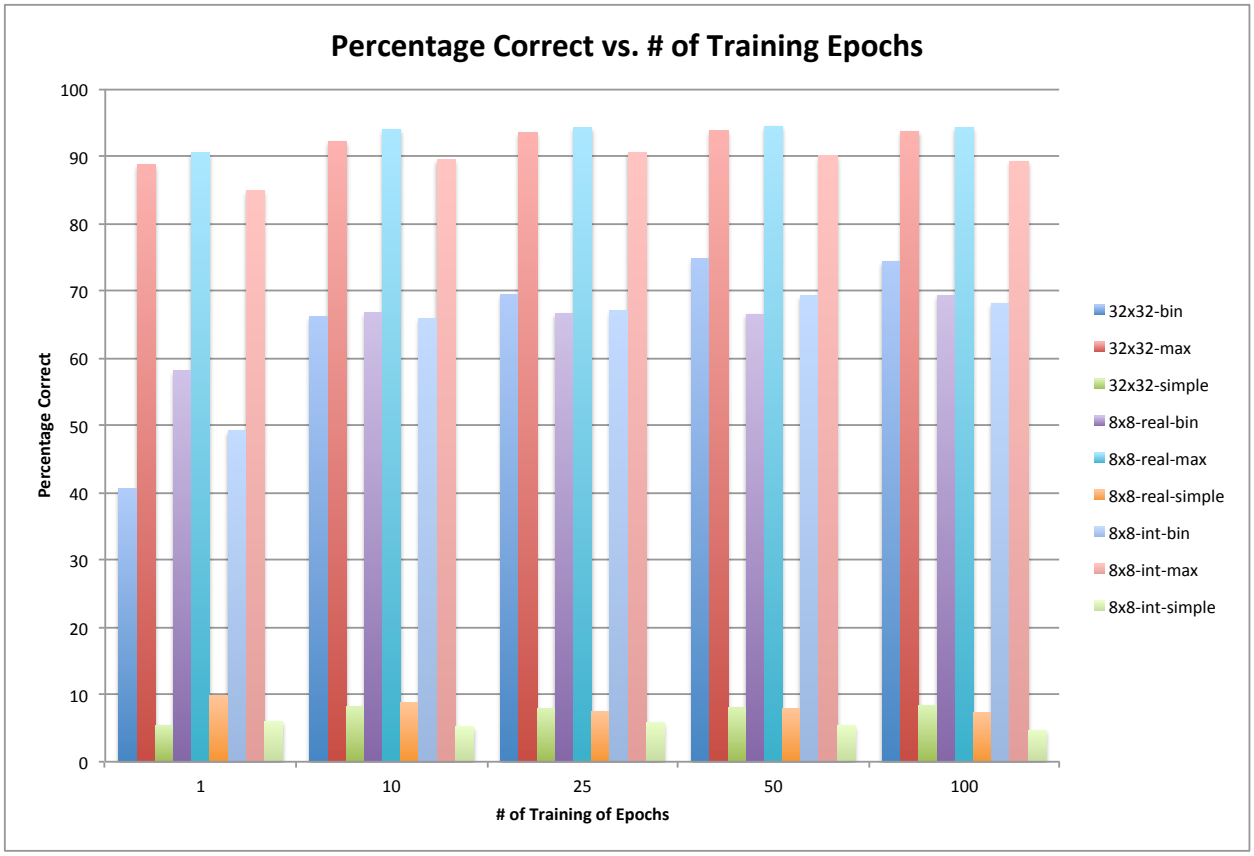

Figure 2: (epoch size = 25)

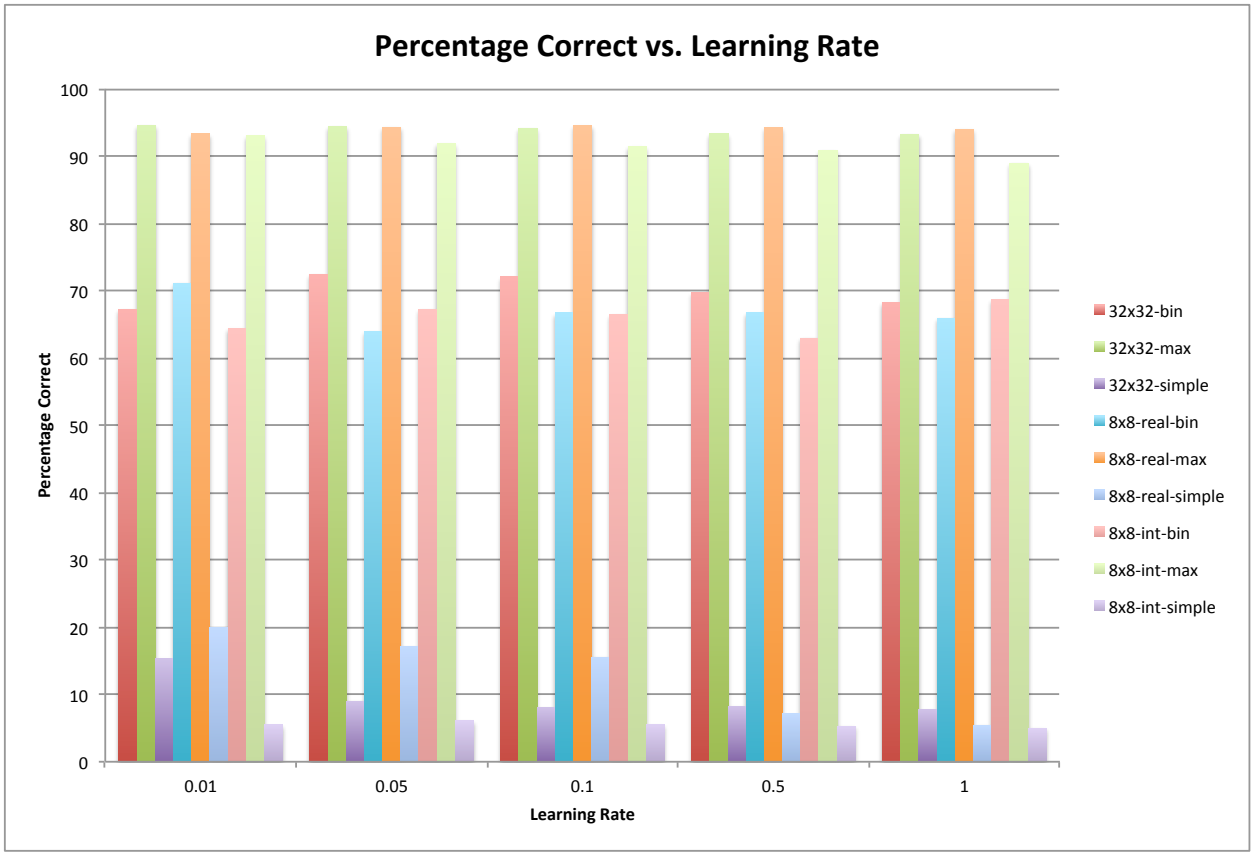

Figure 3: (learning rate = 0.5)

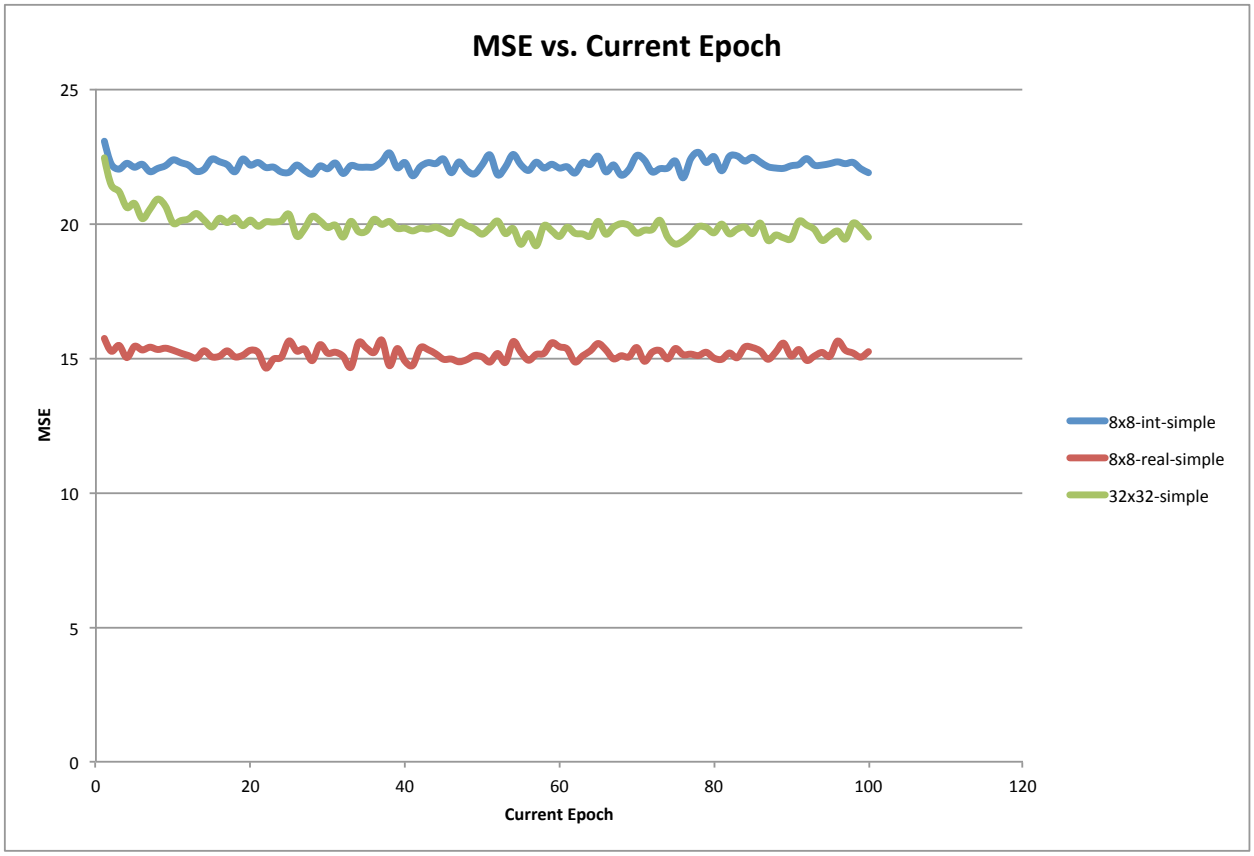

Figure 4: (learning rate = 0.5)