**Ryan Connolly**

**Problem 1**

**Results**

|  |  |
| --- | --- |
| **A picture containing text  Description automatically generatedFigure 1.1.1:** confusion matrix for the training set results using the autoencoder’s trained weights, with no backpropagation. Rows represent actual number in the image, columns represent number classified by the network. | **A picture containing text  Description automatically generated *Figure 1.1.2:*** *confusion matrix for the training set results using the autoencoder’s trained weights, using backpropagation. Rows represent actual number in the image, columns represent number classified by the network.* |

|  |  |
| --- | --- |
| **A picture containing text  Description automatically generated Figure 1.2.1:** confusion matrix for the test set results using the autoencoder’s trained weights, with no backpropagation. Rows represent actual number in the image, columns represent number classified by the network. | **A picture containing text  Description automatically generated *Figure 1.2.1:*** *confusion matrix for the test set results using the autoencoder’s trained weights, using backpropagation. Rows represent actual number in the image, columns represent number classified by the network.* |

|  |
| --- |
| **Chart, bar chart  Description automatically generated**  ***Figure 1.3:*** *a bar graph of the error fractions of the training set  (darker shades) and test set (lighter shades) with backpropagation  (in blue) and without backpropagation (in green).* |
| **Chart, line chart  Description automatically generated**  ***Figure 1.4:*** *a time series plot of the error fraction as a  function of epoch number. The orange line is the error fraction  of the training set without using backpropagation, and the  red line is with backpropagation.* |

**Analysis of Results**

Figures 1.1.1, 1.1.2, 1.2.1, and 1.2.2, as described above, show the quantified results of the classification outputs of the neural network’s different runs. Figures 1.1.1 and 1.1.2 show the results of the training set with the autoencoder’s weights for the input-to-hidden layer, with 1.1.1 only training the hidden-to-outer layer’s weights while 1.2.1 trains both through backpropagation. Right away, it is noticeable that Figure 1.1.2 has higher numbers in the top-left to bottom-right diagonal cells, which hold the number of correct outputs for the corresponding number. This is evidence that backpropagating the autoencoder’s weights trains the network more efficiently and produces more accurate results. This conclusion is supported by the rest of problem 1’s figures, such as Figures 1.2.1 and 1.2.2 which show the confusion matrices of the two alternatively trained sets of weights when applied to the test set. Figure 1.2.2, for the backpropagating run, also has higher numbers in the diagonal cells.

Figure 1.3 shows the error fraction for the same two cases for each two sets. It is shown that the backpropagating network has less than half the overall error fraction for the training set as the LMS-only network does for the training set, and the former’s test set error fraction is about two thirds of the latter’s. There are only two numbers that either LMS-only set scored lower than a backpropagating set: 0, where the backpropagating test set was higher than the LMS-only test set and almost as high as the LMS-only training set, and 4, with a similar situation but much higher error fraction values. Figure 1.3 shows that while the backpropagating network is certainly more accurate than the LMS-only network, it does not always far exceed its competitor’s performance. However, it is worth noting that the backpropagating network’s training set was the only run to achieve an error fraction of zero, for the number 1. This is significant, though it should also be observed that the corresponding backpropagating test set scored around a 0.04, so this perfect accuracy is contained to the training set rather than being across the board.

Figure 1.4 is the graph that most clearly illustrates the difference in speed between the two networks. It plots the error fraction of each network’s training set over 41 epochs, the same number of epochs that it took the training error fraction for my network from HW3 to reach my designated threshold of ≤ 0.1. My final error fraction for HW3’s network was 0.0913 at the 41st epoch, and my finals for this homework were 0.2290 for the LMS-only network and 0.079646 for the backpropagating network. The BP network first made it below HW3’s threshold on epoch 32 with an error fraction of 0.092882, and then it oscillated back and forth over the 0.1 mark three times before spending the last three epochs decreasing by about 0.01, ending at 0.079646. The LMS-only network never reached the threshold, though it does appear to still be decreasing its error fraction with each epoch, if only slightly. It is unclear if it would eventually reach the threshold, but it seems unlikely. The BP network, however, reaches the threshold even faster than the HW3 network did, and it is not clear if it has reached its maximum accuracy either. Through observing Figure 1.4 along with the others, it can be concluded that the BP network with the pre-loaded autoencoder weights is both more efficient and accurate than both the LMS-only network with the pre-loaded autoencoder weights as well as the HW3 network. Additionally, the HW3 network is more efficient than the LMS-only HW4 network, supporting the idea that backpropagation in itself is more beneficial than having pre-generated autoencoder weights that are not trained any further. (However, it should also be noted that my HW3 autoencoder was not the best, so this last conclusion may not hold up under other circumstances.)

**Problem 2**

**Results**

**Chart, bar chart

Description automatically generated**

**Chart, bar chart

Description automatically generated**

**Sample Outputs**

**Figure 2.4: Original Images Compared to Reconstruction**

**Text

Description automatically generated**

**Analysis of Results**

There wasn’t much variation in my results from digit to digit in the test set of numbers 5 through 9, although 8 was the lowest if not by much. I believe that this is because 8 is quite similar to many other numbers in appearance. It is made up of two circles, like zero is made up of one large oval in the same shape. It has two loops, in the top and bottom half, so it shares one loop in either half with many other numbers (5, 3, 6, etc). Therefore, it is the most easily affiliated with them.