Classifying Digits using Neural Nets

Vivianne Dirven and Kwan Suppaiboonsuk

15 January 2016

1 Introduction

The assignment is to build a neural net that can classify different numbers of 0 to 9 in different styles/handwriting. We are building this net using only Matlab. As we are beginners in the field of neural nets, we chose to use the toolbox which is provided within Matlab. This toolbox offers different standardized nets using a specific algorithm.

A pattern recognition net will be used to find the patterns in the pixels representing the digits. The available data has labels, which results in the ability to use a supervised net. Meaning that target can be used to calculate an error and adapt itself to this error to create an as small as possible error to optimize the net, during the training phase.

2 Data

For this assignment, we worked with a testing data set that has an overall size of 60,000 samples. Each sample has 784 data points, each with a value between 0 and 255 (gray scale). The 784 data points of a sample represents a 28x28 image of a a number (0 to 9).

2.1 Data processing

We want the data in only 0's and 1's, otherwise the simple net that we have created would not be able to process it. In order to do this, we wrote a short and simple Matlab code to manipulate the data to contain only 0's and 1's. Since the data is in gray scale, the numbers are not composed of only just 0's and 255's, but also numbers in between. Therefore we thought it reasonable to make the cutoff at 100. Any value below 100 is set as 0, and any that is 100 and up is set as 1. For each sample, this data is still in the form of one string that is 784 values in length.

Also in order to decrease the running time of the net, not all 60,000 samples of data were used as input.

3 Construction and training of the net

The basis for the neural nets constructed is the pattern net algorithm. In constructing the eventual different nets four different properties were constantly tweaked: The structure of the net - the amount of layers and nodes the net comprises out of data to train the net with; the amount of data samples used; and the training method/algorithm used to train the net.

To optimize the training time, the learning gradient and the amount of validation errors occurring should be watched and thresholds set. The learning gradient shows how much the machine is still learning. If the value for this parameter is low, then the machine is not learning much anymore and training it is no longer useful nor efficient. The validation errors show when there is an error in the training, thus the machine is not learning correctly. When the learning gradient reaches the minimum of 1e-5 or the amount of validation errors reaches 6 the training will automatically stop. We decided to leave this threshold the way that the Matlab toolbox has originally set, since they are both reasonable values for when to stop training.

To evaluate the the function of the trained net an evaluation parameter was used. The evaluation parameter is the ROC: receiver operator characteristics. The ROC plots the correct positive results in respect to the amount of false positives. This is interesting information in relation to finding out whether the network used is over-fitted. The ROC plot we are most interested in is the Test ROC plot. When the test ROC has a fairly low rate of correct positives and the training ROC a relatively high rate of correct positives it can be assumed that over-fitting occurs. What we would like to see is the lines of the graph moving along the left vertical axis and the top of the plot (lines clustered in the upper left-hand corner of the plot). Anything above the grey diagonal line in the ROC plot is good.

3.1 Different nets

The different nets constructed have one or two hidden layers with each 28, 100, 300 or 784 nodes. The amount of data to train the net has varied between 99 and 9999 samples. Not the whole database has been used, since it would increase the training time to much and Matlab could not handle the amount of data.

In the training three different training methods have been used: train, trainrp, and trainscg. 'train' is the general Matlab neural network training algorithm and is a blackbox. 'trainrp' uses a resilient back-propagation algorithm. 'trainscg' is a scaled conjugate gradient algorithm. All the methods above are appropriate methods for training a pattern recognition network.

4 Results

In the results in the ROC plots below, the classes represent each digit (class 1 is number 0, class 2 is number 1, and so on). In the caption for each figure, the parameters changed can be found: sample size, amount of layers, amount of nodes in a layer, and training method/algorithm used.

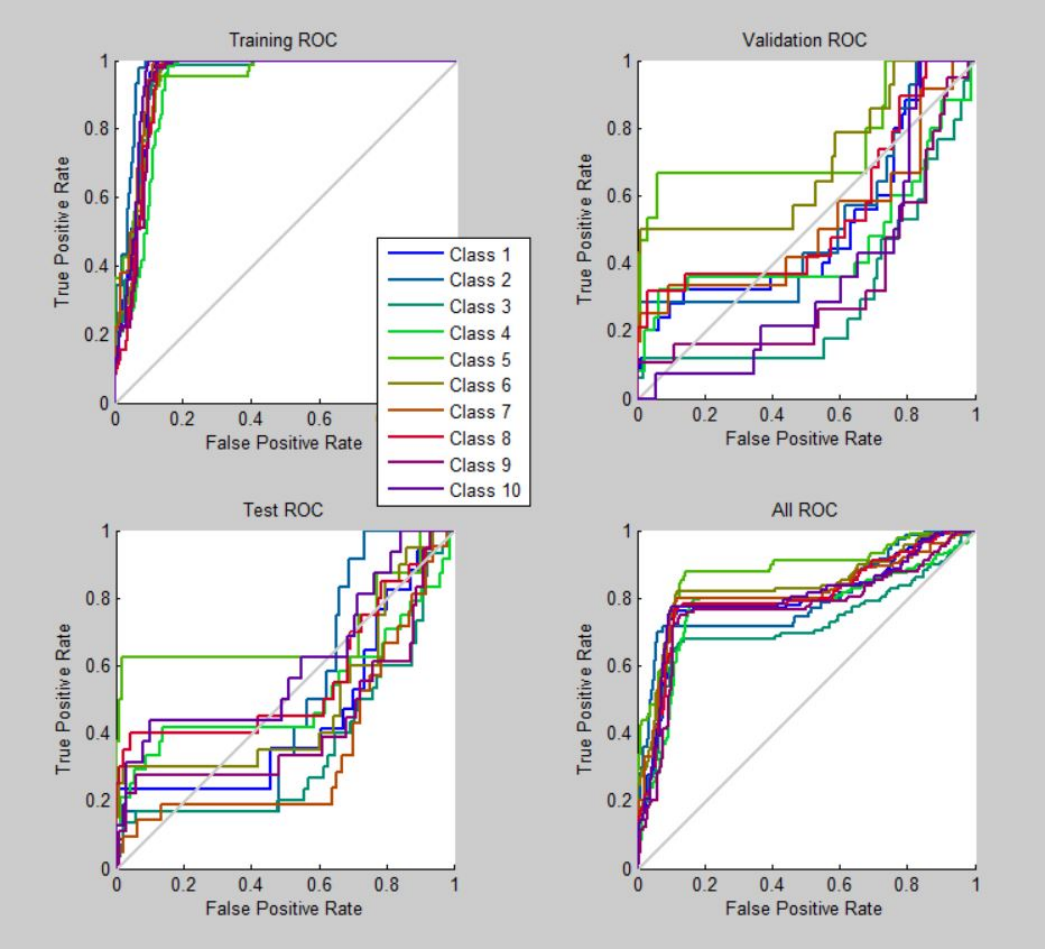

Figure 1: ROC Plots of 2,500 samples; layer 1: 784 nodes; layer 2: 784 nodes; training algorithm: train

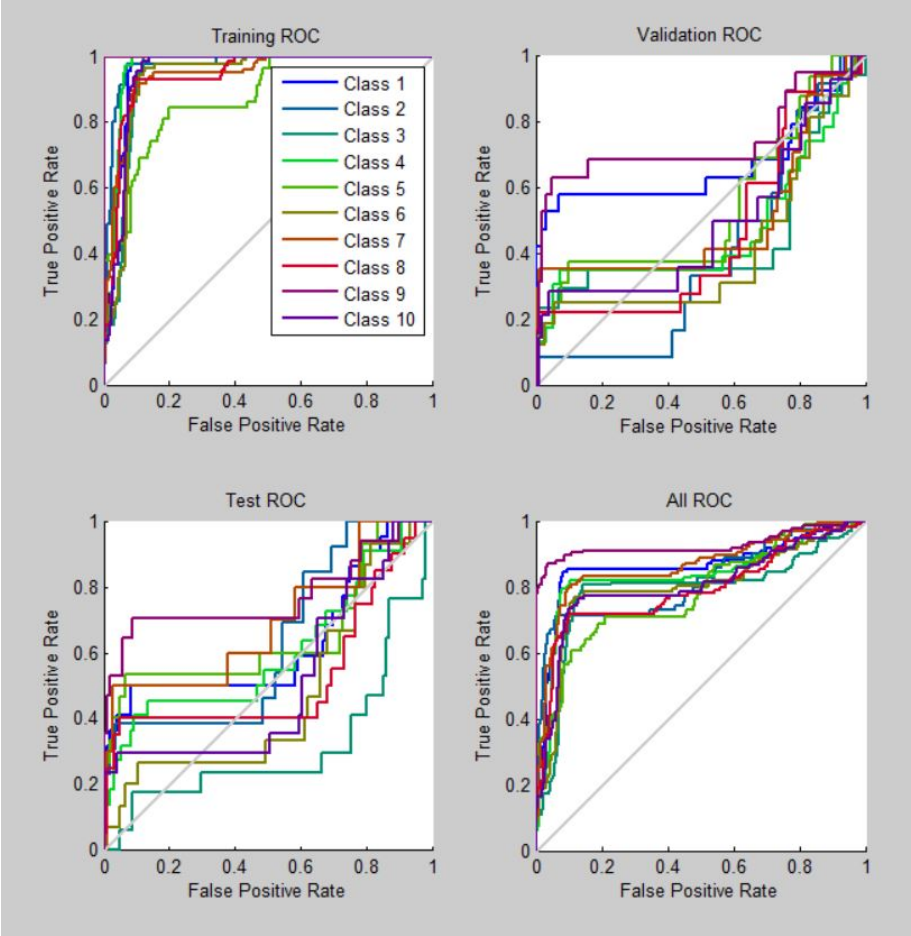

Figure 2: ROC Plots of 2,500 samples; layer 1: 784 nodes; layer 2: 300 nodes; training algorithm: train

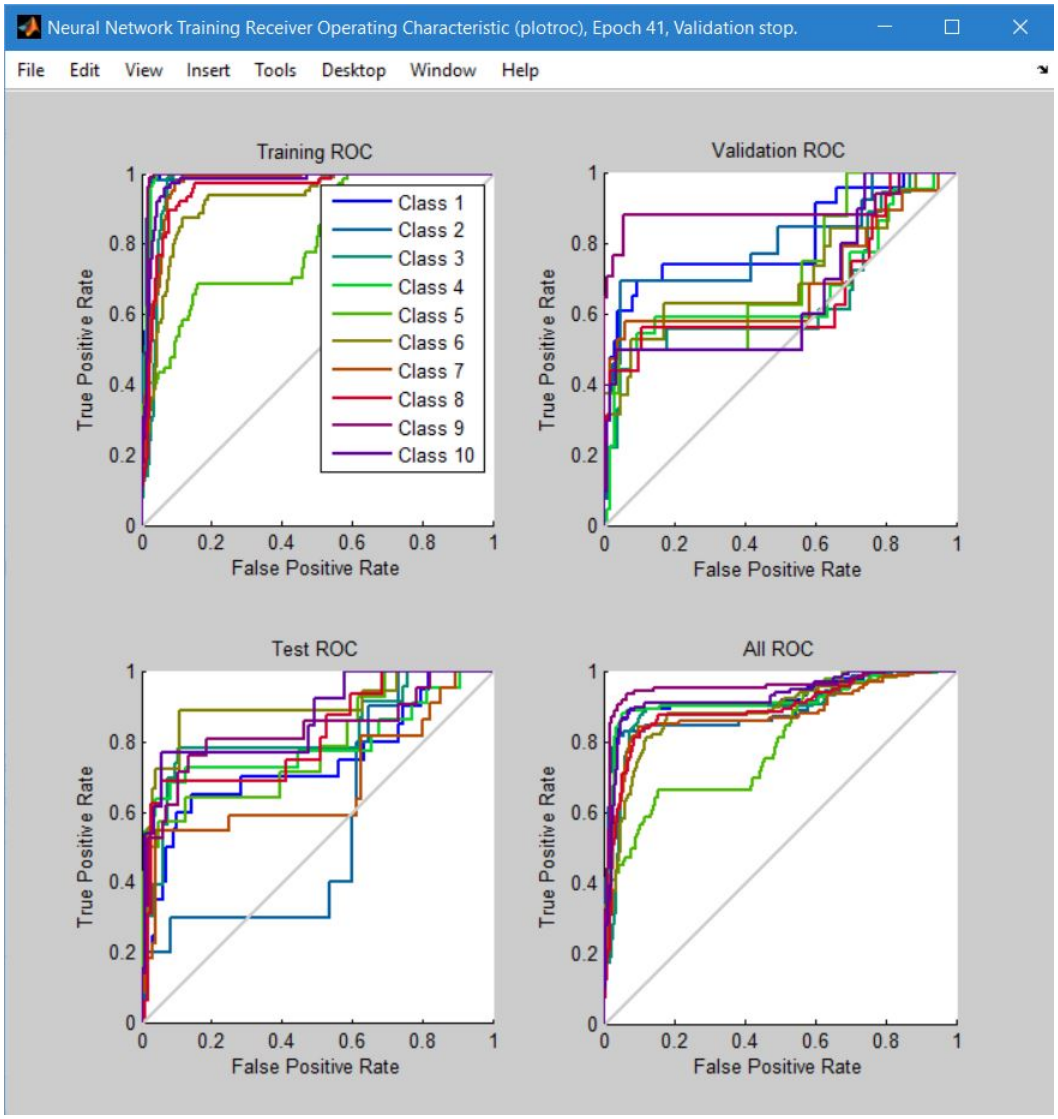

Figure 3: ROC Plots of 2,500 samples; layer 1: 784 nodes; layer 2: 28 nodes; training algorithm: train

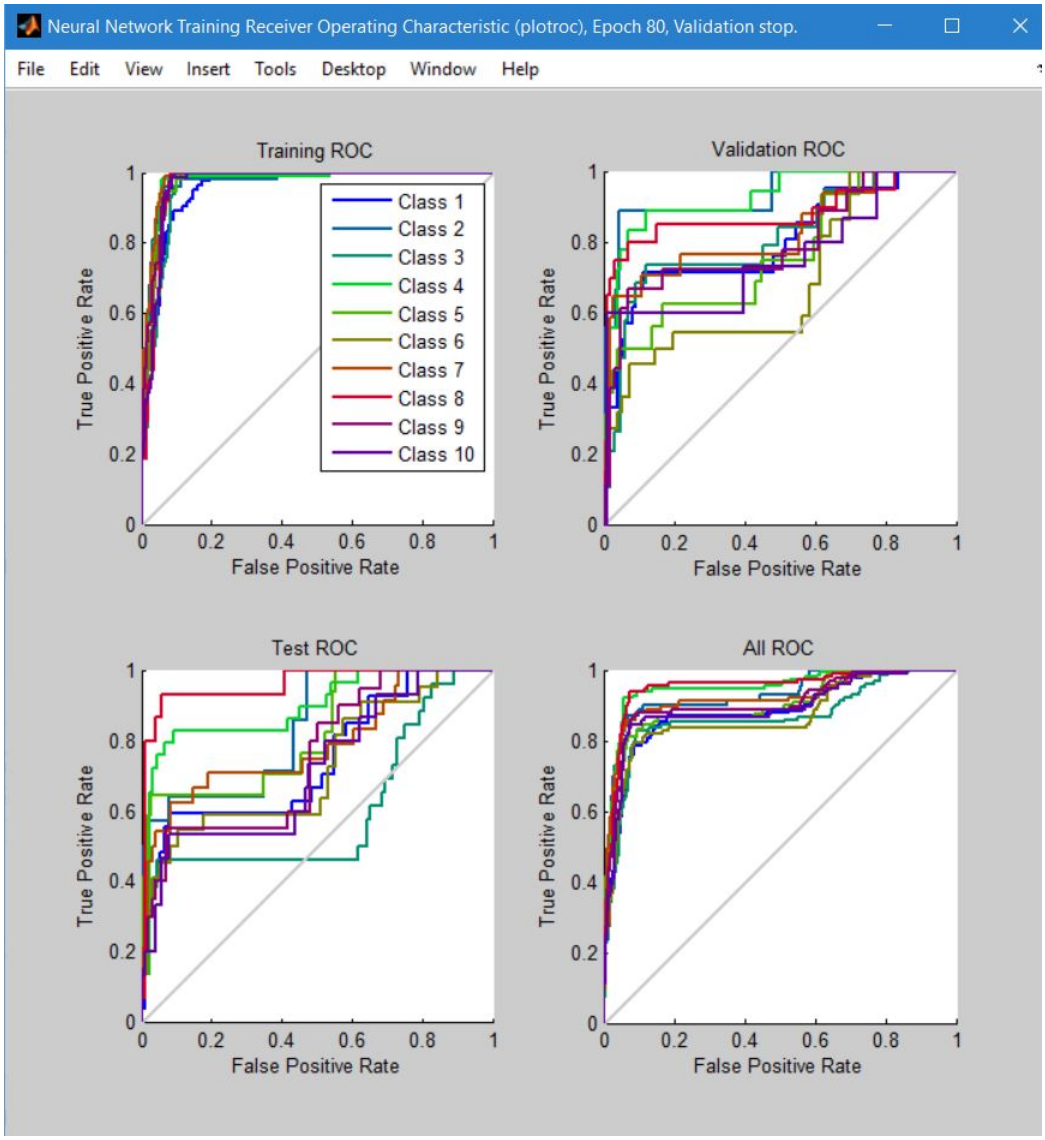

Figure 4: ROC Plots of 2,500 samples; layer 1: 784 nodes; layer 2: 16 nodes; training algorithm: train

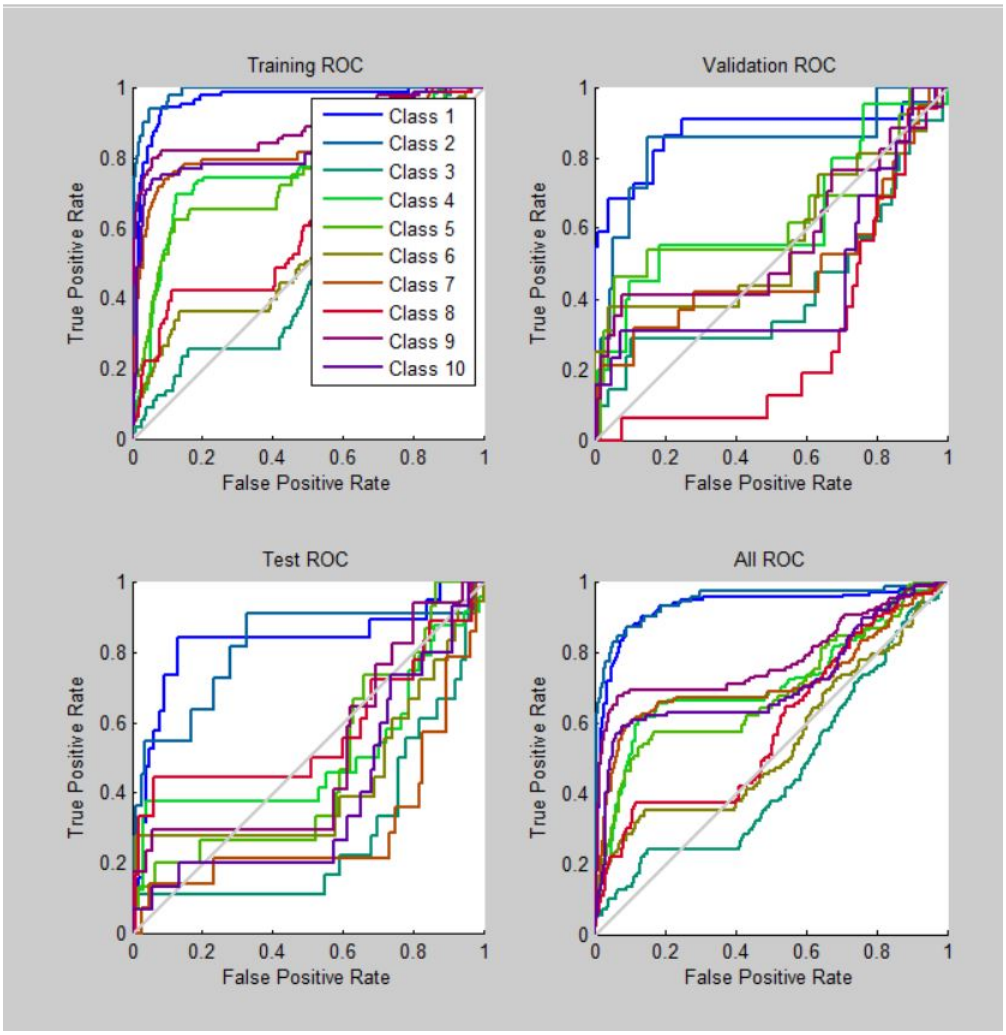

Figure 5: ROC Plots of 2,500 samples; layer 1: 784 nodes; layer 2: none; training algorithm: train

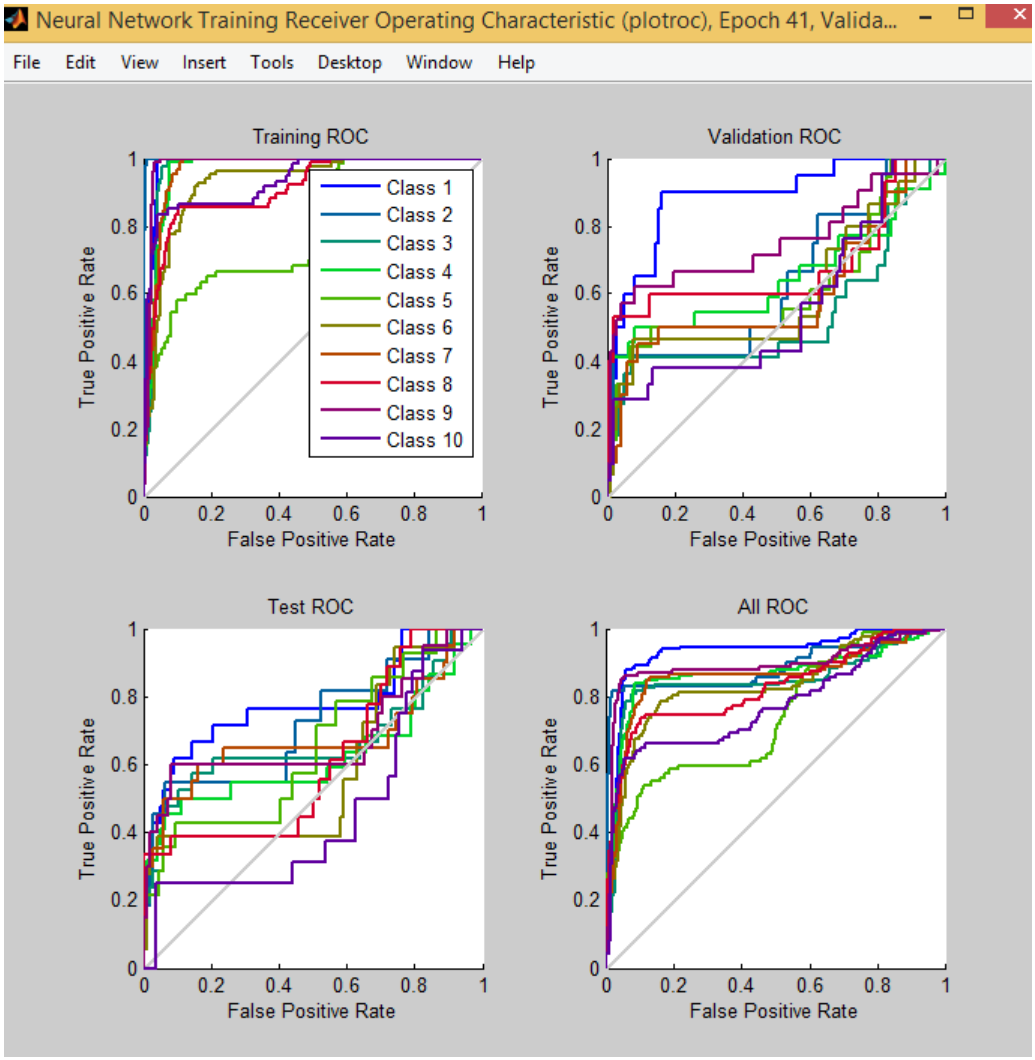

Figure 6: ROC Plots of 9999 samples; layer 1: 100 nodes; layer 2: 100; training algorithm: trainsgp

5 Analysis

The general trend in the different nets constructed is that the greater the data set the better, the higher the training ROC value and testing ROC value. The trainrp resilient back-propagation resulted independently of the amount of nodes used in fairly bad classifications; a high false positive score occurs with all trainings. The trainscg led to better results. The amount of nodes and hidden layers has a more various influence. When using two hidden layers and many nodes (784 per layer) the training ROC was very good, very little false positives and the test ROC was fairly bad, showing more false positives than correct positives for all classes. From which we concluded that a high amount of nodes leads to

over-fitting of the neural net. This relates to the high amount of weights at forehand which can specify to much towards the characteristics of the training data. However very little nodes also will not create an optimal net. Also, it seems as though having only one layer (figure 5) results in a worse test result than when having two layers.

Though not clearly pictured, the increase in data samples do not seem to always affect the result positively. The most optimal result created is when the amount of nodes in the first layer is fairly high (784) and in the second layer fairly low (28).

6 Outlook

There are a number of things which could be improved to create a better classifier.

6.1 Data preparation

In the current approach of the preparation of the data the distinction between black and white pixels was chosen arbitrarily. This could be investigated further by trying to test whether the net improves if this division is made around another number in the gray scale. Or look for possibilities to construct a net that can work with a more complex input than ones and zeros. Besides that the target data used, was arbitrarily chosen from the data file. Resulting in that the target data is not an ideal digit but a random digit, meaning that it can vary a lot from the average digit.

6.2 The net constructed

In relation to data files used the current net used is not an optimal net.The net constructed uses a string of numbers as input data. This results in a loss of information on spatial linkages between the pixels. To overcome this problem the net should be constructed in a different fashion. The net should internally have linkages between the nodes which are in the original matrix spatially close to each other. For instance pixels in the same column or adjusting columns but a different row. In this way the net would generate a more accurate result using the spatial arrangement of the pixels.

To build such a net, the NeuralNetworkToolbox would come short. As a less rigorous change the amount of layers could be increased and deep learning applied, which not necessarily would result in a better end result.