MNIST Machine Learning and Classification

Leif Wesche

3-10-2018

Abstract:

The MNIST data set was classified using a variety of neural networks. The goal was to classify hand written single digit numbers, ranging from 0 to 9. The MNIST data set consisted of 42,000 labeled training images and and 28,000 unlabelled test images. First, three networks with single hidden layers were tested. These hidden layers included a 2D convolution layer, a rectified linear unit layer, and a max pooling layer. Next, all three layers were used together in a single deep neural network to classify the data. The networks were trained and cross validated using the labeled training data before they were used to classify the unlabeled data. It was observed that the convolution layer was the most effective single layer scheme. The overall most accurate classification was obtained by combining all three of the layers into a single network.

I. Introduction and Overview

In this assignment, a simple neural network was used to recognize handwritten single digit numbers. The network was trained, validated, and used to identify numbers from the MNIST database of handwritten numbers. The training data consisted of 42,000 images of handwritten digits ranging from 0-9. This training data was used to train as well as validate each neural network. The MNIST test data consisted of an additional 28,000 handwritten digits. The test data was unlabeled, unlike the training data. The networks were trained and cross validated by using 90% of the training data to train the network to classify the remaining 10% of the training data data. Then accuracy of the networks was calculated accuracy by comparing the classifications with the labels, and the process was repeated five times for each network. The test data was then classified and uploaded to the MNIST database through the Keggle website to be scored.

II. Theoretical Background

Neural networks were constructed using Matlab's deep learning toolbox. The network architecture was constructed by concocting various layers into a martix. These layers included a image input layer, a variety of hidden layers, a fully connected layer, sofmax layer, and finally a classification layer. Matlab's "trainNetwork" command was then used with the layer architecture, the training images, training labels, and a variety of options.

The first of the hidden layers tested was a 2D convolution layer. The convolution method applies a filter to a small region in the image. In this case the filter size used was 5 pixel tall by 5 pixels long. The filter computes the dot product of the region and a weighting vector, then adds a bias factor. The filter moves horizontally and vertically along the image and performs this calculation. The weighting vectors and bias factors are calculated during the training phase. Based on the total result of this weighting calculation over the entirety of the image, the image is classified.

The second hidden layer used was the max pooling layer. The max pooling layer down sampled each image by dividing the image into small regions, then calculated which labeled category each region most closely matched. The images were classified based on how well the average region matched the corresponding regions in each category.

The rectified linear unit layer tested each element, or pixel value, in the input to see if it was greater than or less than an expected threshold. Based on how the pattern of the outcomes fit into the categories, the image was classified. The threshold value used to test each element was determined during the training phase.

The work flow used to train, cross validate, and use each network used began with the training data. The labeled training data provided was split into two subsets, a verification training subset, which consisted of random 90% of the training data and labels, and a verification test subset, which consisted of the remaining data and labels. The networks were trained on a the verification training sets five times, and each time the verification set was re-randomized. The networks were used to classify the corresponding verification test sets, which were composed of images that they were not trained on. The classification results were compared with the labels to compute the accuracy of each network. Overall, each of the four network architectures were verified by rebuilding and testing it 5 times, for a total of 20 verification tests. This method of cross validation was performed to ensure the accuracy of the networks before they were used to classify unlabeled data. Finally, the networks were rebuilt using the full set of training data and used to classify the test data.

III. Algorithm Implementation and Development

First the two data sets, both in the form of CSV files, were read into Matlab. The training data was imported as a 42,000x785 matrix, where the top row consisted of a numeric label and remaining rows consisted of black and white image pixel values. The labels were separated from the images, and the images were reshaped into 28x28 matrices. The training images were arranged into a single 4D 28x28x1x42000 matrix.

The original training data set was then separated into a subscale training set consisting of 90% of the images and their corresponding labels, and a validation set, consisting of 10% of the images and their corresponding labels. The images stored in both sets were randomly selected without repeating by using Matlab's "randperm" command to generate a vector consisting of non repeating random numbers from 1 to 42,000 to be used as indices. To validate the matrices, each Network configuration was trained and tested five times using unique randomly shuffled training and validation data sets.

The test data was originally imported as a single 28,000x784 matrix of images arranged as column vectors. Similarly to the training data, the images were reshaped to 28x28 squares, and stored in a single 4D 28x28x1x2800 matrix. These test images came with no labels.

Four networks were tested in total. Matlabs "trainNetwork" command was used to train a network based on the with the training images, labels, layers, and options as inputs. The options were set using Matlab's "trainingOptions" function. The options specified for each test included the maximum epochs, which was set to 20, the learning rate, which was set to 0.0001, and the optimization method, which was set to stochastic gradient descent.

All four of the networks tested used an input layer, some combination of hidden layers, a fully connected layer, softmax layer, and a final classification layer. The first network used a convolution layer as the first layer. The second network used a max pooling layer for the hidden layer. The third network used a rectified linear unit layer. The fourth network used the convolution layer, the max pooling layer, and the rectified linear unit layer in series. The exact configurations of the hidden layer portions of each network are shown in Table 1 below. In each test, the maximum epoch was set to 20, and the learning rate was set to 0.0001. These parameters were experimentally determined to work well for the MNIST data set.

Table 1: Neural network hidden layer composition.

Network number	**Hidden Layer**	**Settings**
Network 1	2D Convolution	Size=[5,5] Filter Number=20
Network 2	Max Pooling	Size=[2,2] Stride=[2,2]
Network 3	Rectified Linear Unit	n/a
Network 4	2D convolution	Size=[1,5] Filter Number=20
	Rectified Linear Unit	n/a
	Max Pooling	Size=[2,2] Stride=[2,2]

After each neural network was validated, a new network using the same architecture was constructed to be used on the test data. This new network was constructed using the entirety

of the test data and the same settings as the validation networks. This was done to provide further robustness to the network by training it on a slightly larger data set. The new network was used to classify the test data, and the results were submitted to the Kaggle website to be scored.

IV. Computational Results

Each of the four network architectures were constructed and validated five times before it was used to classify actual test data. The accuracy of each verification trial was recorded, and is shown in Figure 1 below.

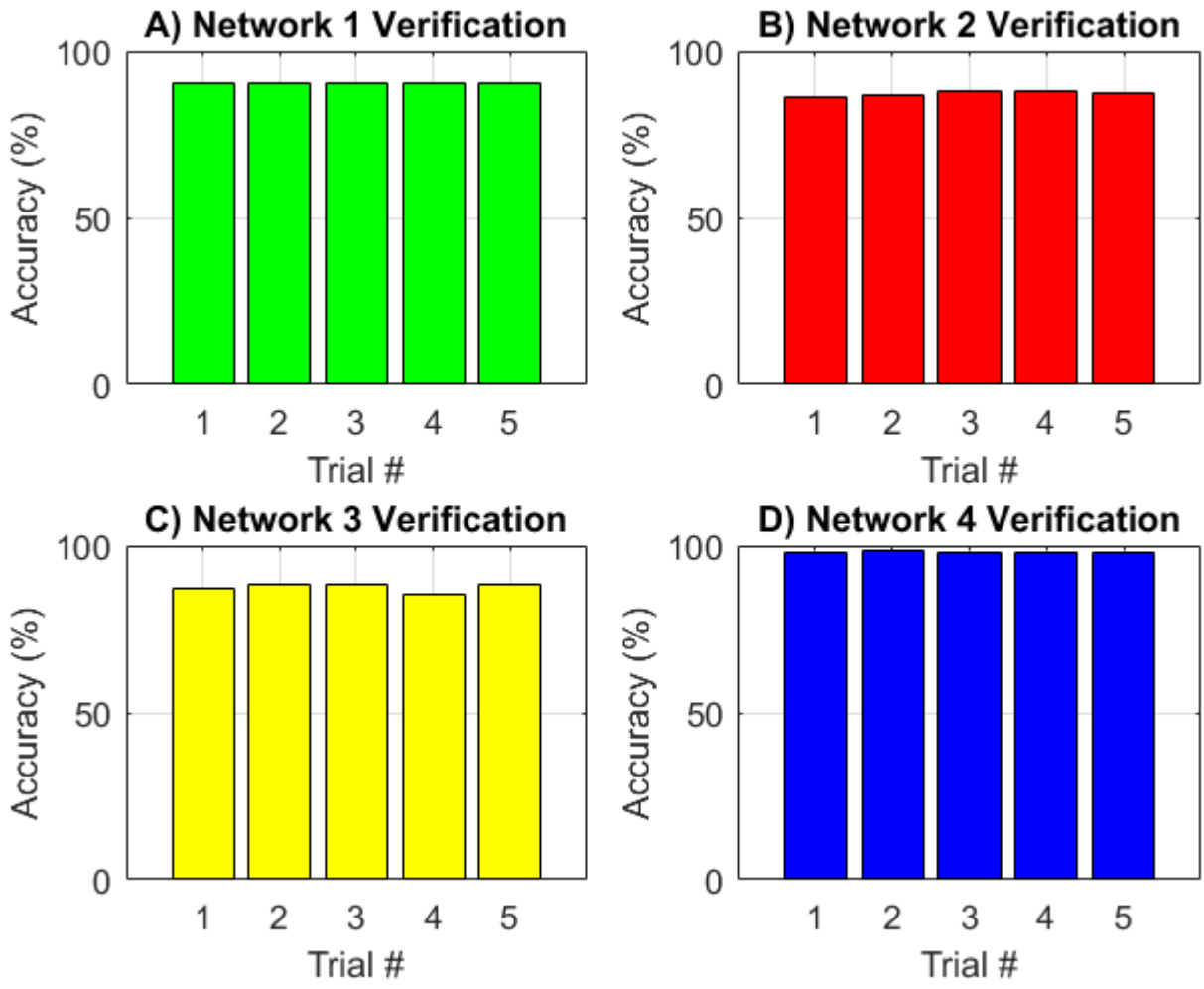

Figure 1: Network verification using training data subsets.

The first network architecture used a single 2D convolution layer, in addition to all other layers and settings that all networks had in common. It was constructed and tested five times

using random training and validation subsets. It was found to perform well, doing the best out of all the single hidden layer networks, with an average accuracy rate around 90% for each of the five validation tests. The validation results of network 1 are shown above in Figure 1A.

Networks 2 and 3 performed similarly in the verification tests. Network 2 used a single max pooling layer while network 3 used a single rectified linear unit layer. The scores from both networks were all in the range of 85-88% accuracy, which was slightly worse than network 1. The validation results of networks 2 and 3 are shown in Figures 1B an 1C above.

The final network used three hidden layers, and performed much better than the other three single layer networks. The network 4 hidden layer architecture used a convolution layer, followed by a rectified linear unit layer, and finally a max pooling layer. The average accuracy for each trial of network 4 was 98%. The verification results of network 4 are shown above in Figure 1D.

Out of all of the single hidden layer networks, network one performed the best, which used a single convolution layer. This results suggests for databases of this relatively small size, 2D convolution architecture seems to be most effective when using a single hidden layer network to classify image data. However, the network that employed multiple hidden layers, network 4, greatly outperformed all of the single layer networks. So while the convolution method worked well on its own, the rectified linear unit and max pooling methods were still found to be very useful additions to the network.

Finally, network 1 was reconstructed using the entire 42,000 image set of training data, and this network was used to classify the unlabeled test data. The same was done using the architecture of the networks 2, 3, and 4. The classification results were uploaded to the MNIST database hosted by the Keggle website and scored based on accuracy. The results are shown below in Figure 2.

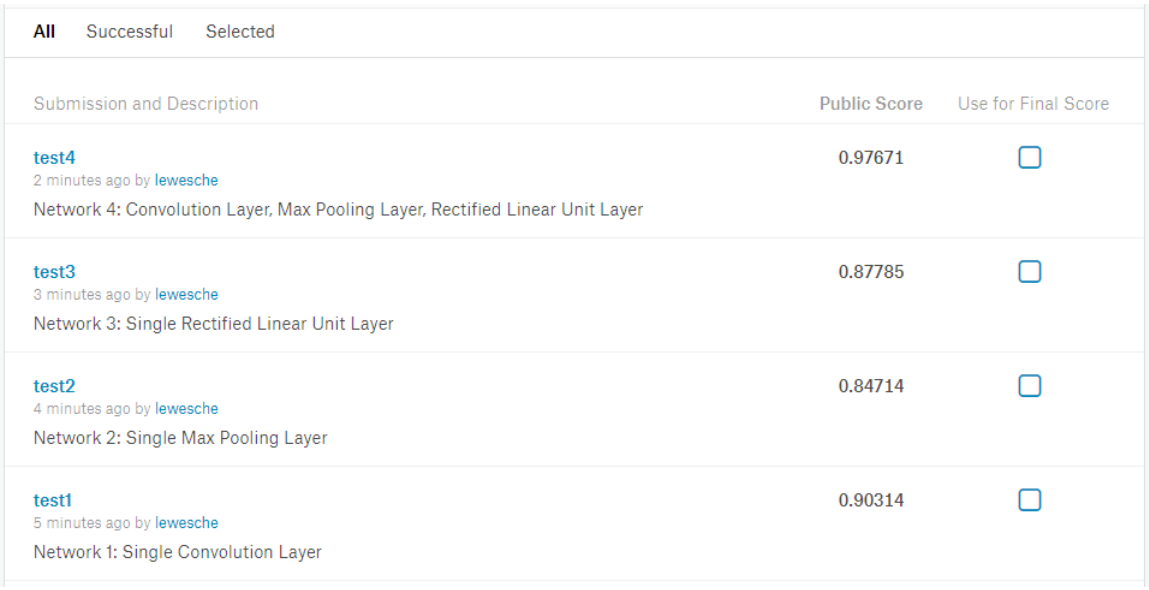

Figure 2: Keggle scoring of MNIST classification performed by each network.

As predicted, the overall best predictor was found to be network 4, which used multiple hidden layers. The second best was found to be network 1, which used a single convolution layer. The third best was found to be network 3 which used a rectified linear unit layer, followed by network 2 which used a max pooling layer. Out of all the single layer networks, network 1, which used a convolution layer, did the best. This result matched the results from the verification tests. Interestingly, networks 2 and 3 performed very similarly in the verification test, but networked 3 performed considerably better on the actual test data. This difference could suggest that the max pooling layer used in network 2 became over fitted to the training data, and thus scored slightly worse on the actual test data.

V. Summary and Conclusions

A variety of neural networks were used to identify handwritten numbers from the MNIST data set. The networks used three different layer architectures, including 2D convolution, rectified linear unit, and max pooling methods. These layers were also tested together in a single multi-layer network. The networks were trained and cross validated using labeled test data, and then used to classify unlabeled data. Out of all the single layer networks, the 2D convolution performed best on its own, with an accuracy rating of about 90%. However, highest accuracy was obtained when all three layers were used in conjunction in a single network, which classified about 98% of the data correctly.