Matthew Hess

Red-id: 818299658

Assignment 5: Feed Forward Neural Network

**Model: Two Layered Relu:**

Best model performance with data:

Activation function used: Only Relu

Most accurate model Parameters:

Parameters

batch\_size = 800

hidden1 = 100

learning\_rate = 0.1

max\_steps = 2000

reg\_constant = 0.01

Test accuracy 0.929

confusion matrix

[[ 960 0 12 2 1 8 12 4 7 12]

[ 0 1113 7 1 5 2 3 12 8 9]

[ 1 2 923 20 4 4 5 23 6 1]

[ 2 2 17 935 1 37 1 3 24 13]

[ 0 1 15 1 912 6 10 7 10 32]

[ 4 0 1 17 0 794 12 0 14 6]

[ 9 4 12 3 11 17 912 0 16 1]

[ 1 2 11 12 2 3 1 951 8 8]

[ 3 11 30 13 3 14 2 4 866 3]

[ 0 0 4 6 43 7 0 24 15 924]]

recall\_array

[0.94302554 0.95948276 0.93326593 0.90338164 0.91750503 0.93632075

0.92588832 0.95195195 0.91253952 0.90322581]

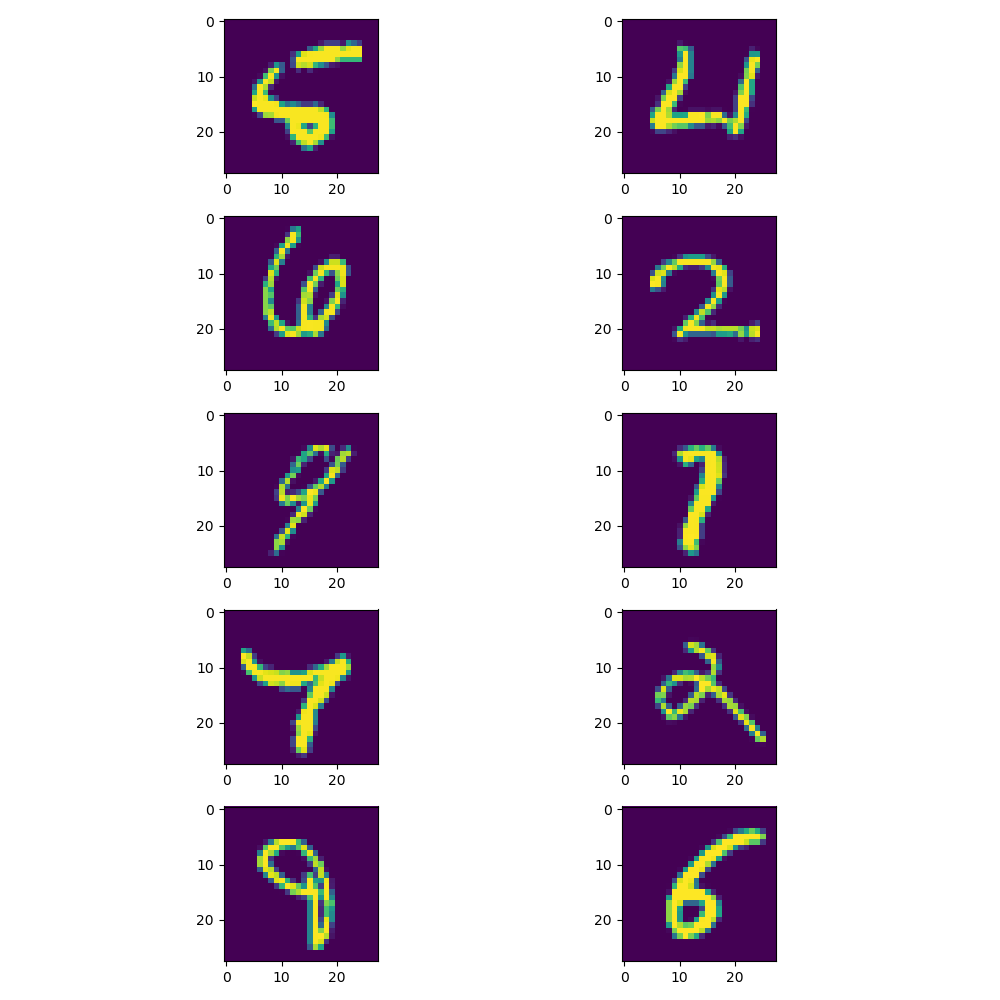
precision\_array

[0.97959184 0.98061674 0.89437984 0.92574257 0.9287169 0.89013453

0.9519833 0.92509728 0.88911704 0.91575818]

Total time: 40.2112

Images that were errors:



**Note:** The following observations were taken with respect to fixed indices of selection for each data set. This makes each observation respectable to each other. In addition the following observations were made with changes in single hyper-parameter with respect to the following.

Respectable Parameters:

Activation function hidden: relu

batch\_size = 400

hidden1 = 100

learning\_rate = 0.01

max\_steps = 2000

reg\_constant = 0.01

These parameters made the model have an accuracy of 85.85% and a completion time of 22 seconds.

After changing hyper parameters around I noticed that if my number of units in hidden layer was 500 it would take around 3 times longer than if the number of hidden units was lower than 100. I also observed that I had a more accurate model when I had 500 units appose to 100 by about 0.56 percent.

|  |  |  |
| --- | --- | --- |
| Number of neuron units | Accuracy | Time |
| 500 | 0.864 | 72 seconds |
| 100 | 0.8584 | 22 seconds |

With this being said, I don’t believe that the additional 400 neural units in the hidden layer was worth the additional computational demand and time.

I noticed that changing the learning rate from 0.01 to 0.1 made the model converge faster in addition to the accuracy increasing by 0.12 percent.

Changing the batch size from 400 to 800 severely improved the model’s accuracy. With the original parameters the accuracy was 85.8 percent and it jumped to 92.9 percent accuracy. The other significant change was that the time to complete training and evaluation took twice as long.

Decreasing the amount of steps for training from 2000 to 1000 increased the performance by 6.1% and decreased the time by 2 seconds.

Increasing the regularization constant from 0.01 to .1 decreased the accuracy by about 0.51 percent.

Changing the activation function to leaky-relu increased the accuracy by 3.89% but increased the time by 8 seconds.

Changing the activation function to sigmoid function decreased the accuracy by 7.69% as well as increased the time by 15.6 seconds.

With all of these changes to this specific model and chosen hyper-parameters to compare with I would conclude that increasing the batch size had the most dramatic increases in the model’s accuracy despite additional time that was needed for the model to complete training and also evaluation.

**Model: Three Layered Relu:**

Best model performance with data:

Activations functions used: only Relu

Parameters

batch\_size = 800

hidden1 = 100

hidden2 = 100

learning\_rate = 0.1

max\_steps = 2000

reg\_constant = 0.01

Test accuracy 0.9285

confusion matrix

[[ 960 0 10 2 1 10 12 4 6 14]

[ 0 1113 8 1 5 3 3 13 7 9]

[ 1 2 926 19 5 5 4 23 6 1]

[ 1 2 17 927 0 33 0 3 19 11]

[ 0 0 14 0 916 7 11 6 11 35]

[ 3 2 0 22 0 795 16 0 18 6]

[ 11 4 12 2 11 14 909 0 11 1]

[ 1 2 13 14 2 3 1 948 9 11]

[ 3 10 30 13 3 15 2 3 874 4]

[ 0 0 2 10 39 7 0 28 13 917]]

recall\_array

[0.9421001 0.95783133 0.93346774 0.91510365 0.916 0.92227378

0.93230769 0.94422311 0.91327064 0.90255906]

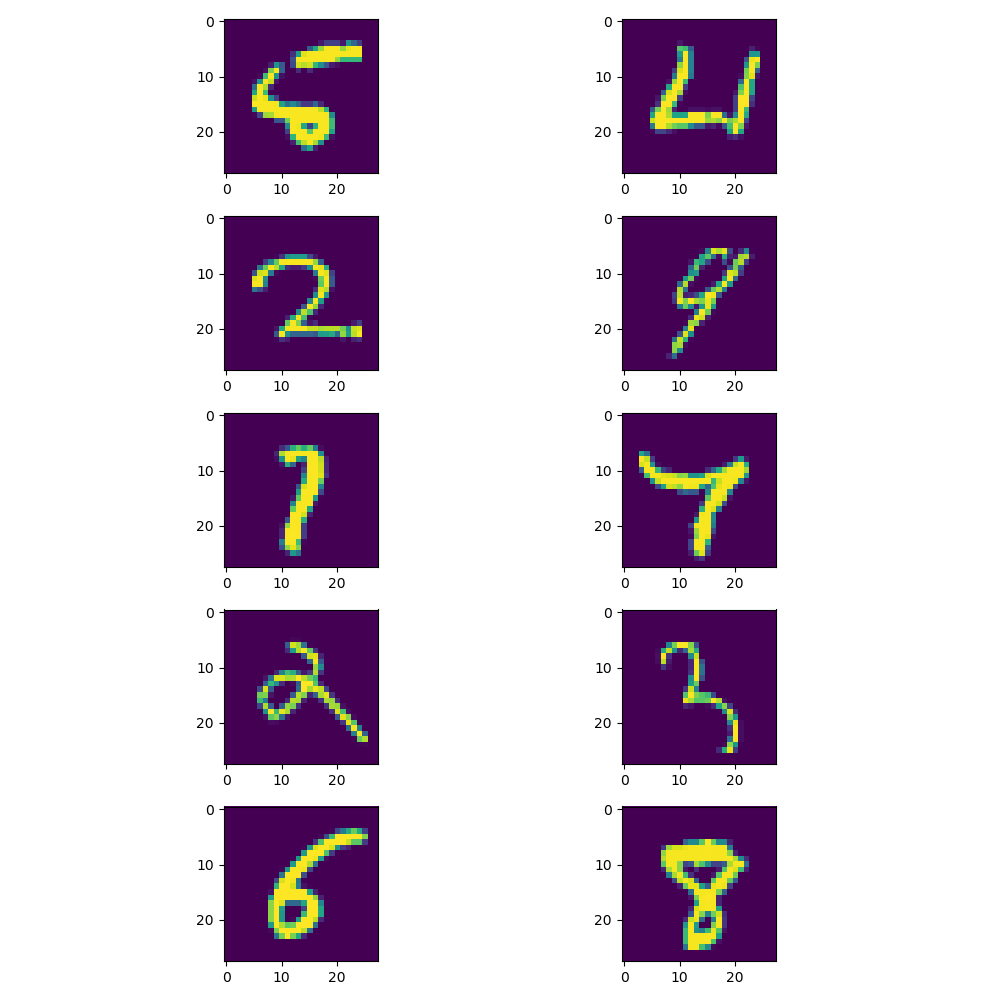
precision\_array

[0.97959184 0.98061674 0.89728682 0.91782178 0.93279022 0.89125561

0.94885177 0.92217899 0.8973306 0.90882061]

Total time: 89.0393

Images that were errors:



**Note**: The same precautions of fixing indices and comparing one hyper-parameter change was made at a time.

Respectable Parameters:

Activation function hidden 1: relu

Activation function hidden 2: relu

batch\_size = 400

hidden1 = 100

hidden2 = 100

learning\_rate = 0.01

max\_steps = 2000

reg\_constant = 0.01

These parameters made the model have an accuracy of 90.16% and a completion time of 25.94 seconds.

Changing the hidden1 and hidden2 from 100 neural units each to 500 improved the model accuracy by 0.13% but increased the model time by around 5 times. Which again wouldn’t be worth the additional time nor the additional computational power in my opinion.

Leaving the hidden1 to have 100 neural units and changing the hidden2 to be equal to 500 increased the performance by 2.69%, the time also increased by from 25.94 to 89.039 seconds.

Changing batch size from 400 to 800 increased the time by about 2 and decreased the performance by 1.6%.

Doubling the batch size from 400 to 800 decreased the performance by 0.36% and increased the time by 20 seconds as well.

Decreasing the number of steps from 2000 to 1000 decreased the time by around 8 seconds and decreased the performance by about 3%.

Keeping the first hidden layer activation function as relu but changing the second hidden layer activation function to a leaky-relu instead of a relu decreased the performance by 0.17% and increased the time by 16.2 seconds.

Changing the first hidden layer activation function to a leaky-relu and keeping the second hidden layer activation function as relu decreased performance by 0.3% and increased the time by 5.5 seconds.

Manipulating different number of hidden units in each layer seemed to change the performance by the most in this model with this specific data. I thought it was interesting that changing the number of first layer hidden neuron units had more significance than changing the number of second layer hidden neuron units. I also thought that the time increase when changing activation functions was interesting. I noticed that most of the models seemed to have problems predicting similar images.

In addition I though working with Tensorflow was challenging due to the fact that I had to learn a lot about the framework to complete this assignment though I think it was application for myself.

Please note that I computed a lot of different models with different hyper-parameters and didn’t want to repeatedly copy and paste my confusion matrices and other data, instead I changed the parameters with respect to a fixed set of parameters and noted how each change in hyper-parameters changed the overall performance and time. With this being said I still included the different activation function configurations in my code that is attached with this report. In addition, I have reported the results of the two layered sigmoid model on the following page.

**Model: Two Layered Sigmoid:**

Parameters:

batch\_size = 800

hidden1 = 100

hidden2 = 100

learning\_rate = 0.01

max\_steps = 2000

reg\_constant = 0.01

Test accuracy 0.8991

confusion matrix

[[ 955 0 14 4 1 16 18 4 6 11]

[ 0 1104 15 1 3 4 3 20 10 7]

[ 2 0 884 21 5 5 10 24 12 3]

[ 2 6 18 898 1 51 1 4 30 13]

[ 0 1 20 0 905 15 16 8 14 43]

[ 7 1 1 36 1 733 17 0 36 14]

[ 8 4 21 1 14 21 888 0 15 1]

[ 1 1 15 22 2 8 1 923 11 34]

[ 5 18 41 21 7 31 4 5 822 4]

[ 0 0 3 6 43 8 0 40 18 879]]

recall\_array

[0.92808552 0.94601542 0.91511387 0.87695312 0.88551859 0.86643026

0.91264132 0.90667976 0.85803758 0.88164493]

precision\_array

[0.9744898 0.97268722 0.85658915 0.88910891 0.92158859 0.82174888

0.92693111 0.89785992 0.84394251 0.87115956]

Total time: 49.91

