MNIST Digit Recognition Using Multi-Layer Neural Networks

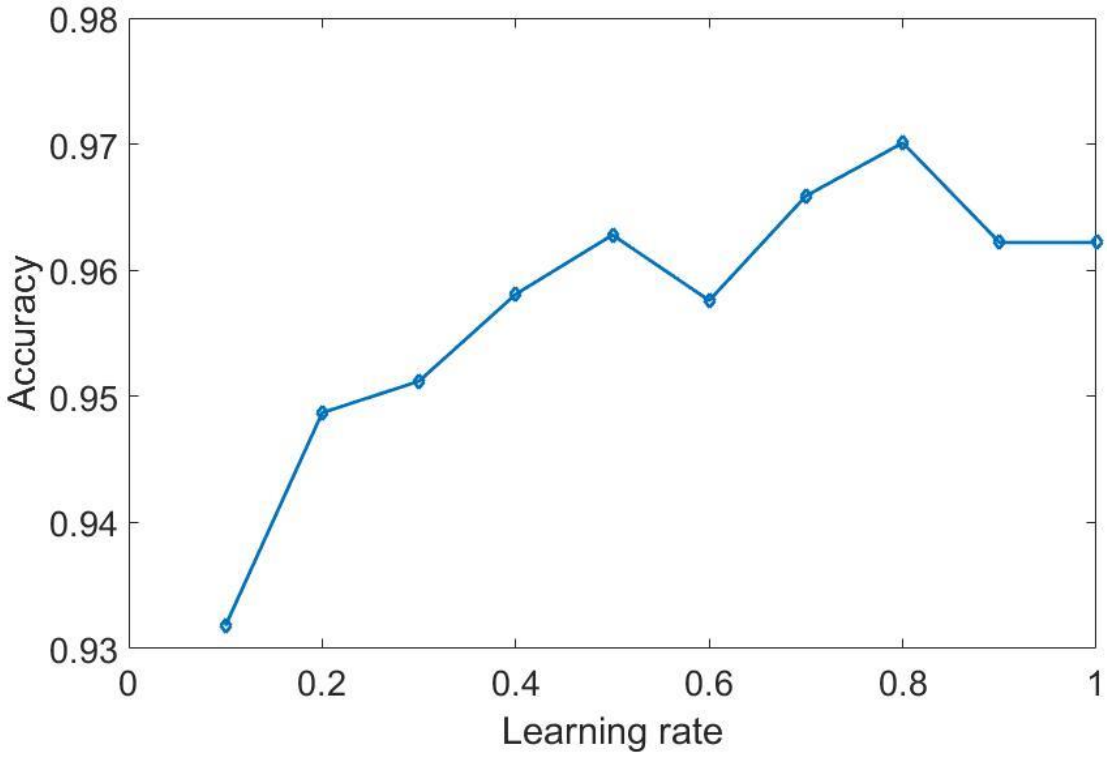
Zheng Lu

Performance evaluation (Tensorflow)

We test the performance of our neural network against 3 sets of hyper-parameters: learning rate, number of layers and number of neurons in each hidden layer.

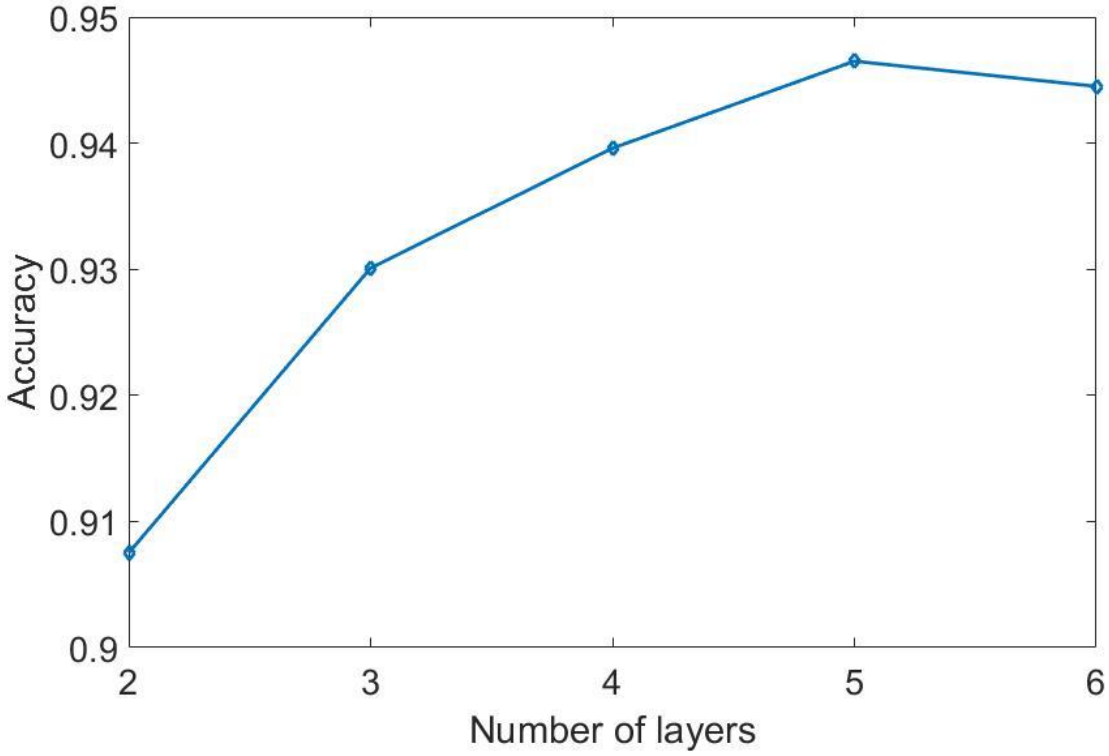
1. Learning rate

   First we test the relation of accuracy and learning rate. We vary the learning rate from 0.1 to 1.0. We firx number of layers at 3 (with 1 hidden layer) and neurons in each hidden layer at 88 which is the square root of sum of input and output size.



2. Number of layers

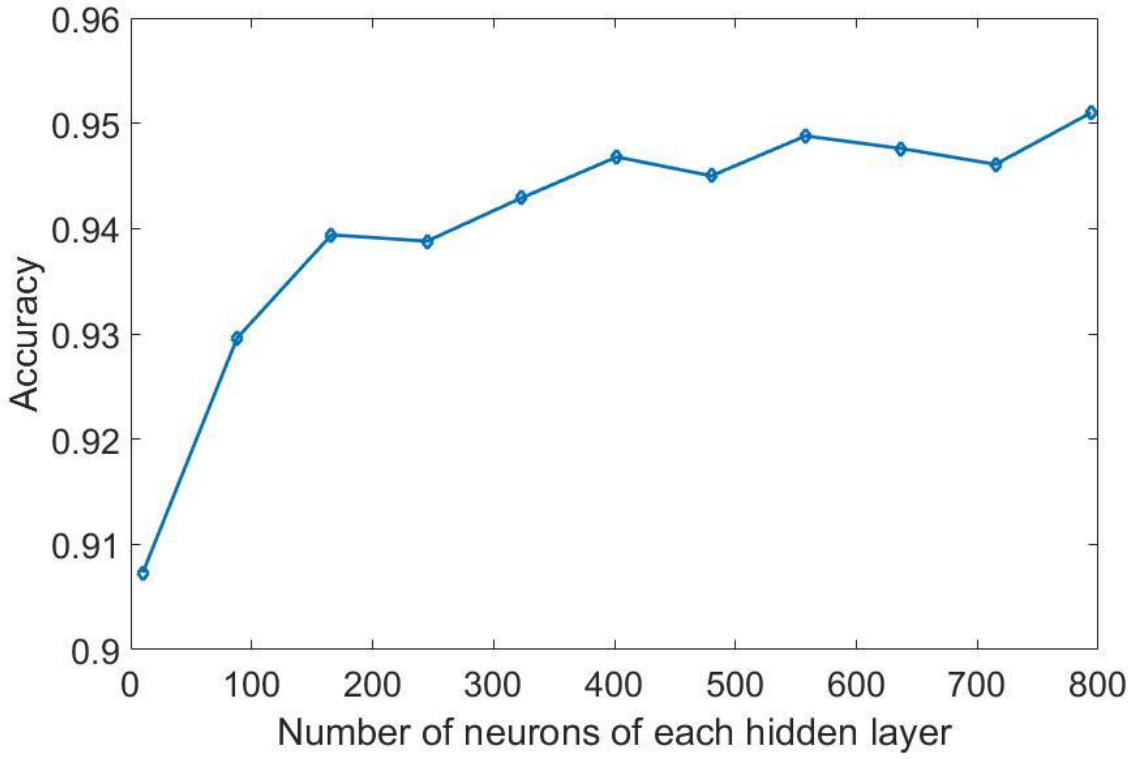
We then use different number of layers for our NN. We fix learning rate at 0.1 and set number of neurons in each hidden layer at 88.



We can see that increase the number of layers will create a more complex network thus increase the accuracy. However, as the number of layer are too high, the model may have overfitting problem.

3. Number of neurons in each hidden layer

   Finally, to test the performance of different number of neurons in each hidden layer, we fix learning rate at 0.1 and number of layers at 3. We use int(image_size * 0.1 * i) + n_class to calculate the number of neurons, where i increases from 0 to 10.

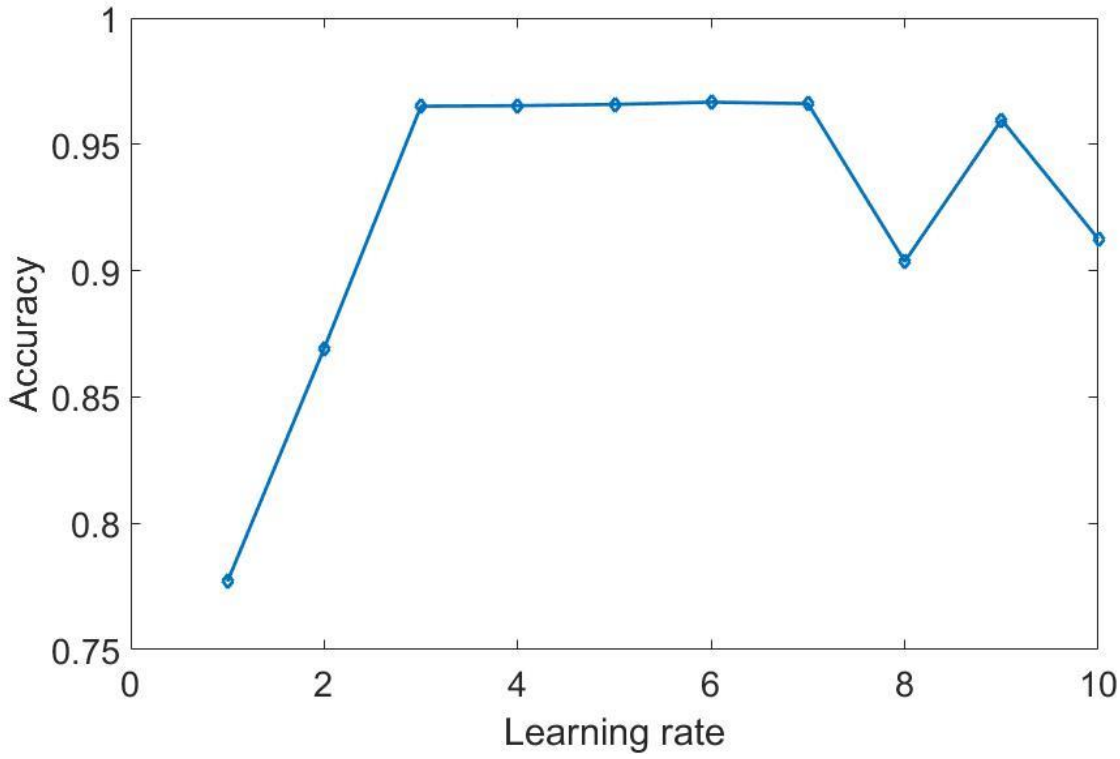


The result of this test seems chao. The only conclusion we can draw from it that as long as the number of neurons in hidden layers is not too high or too low, the NN will perform well.

Performance evaluation (Nelson)

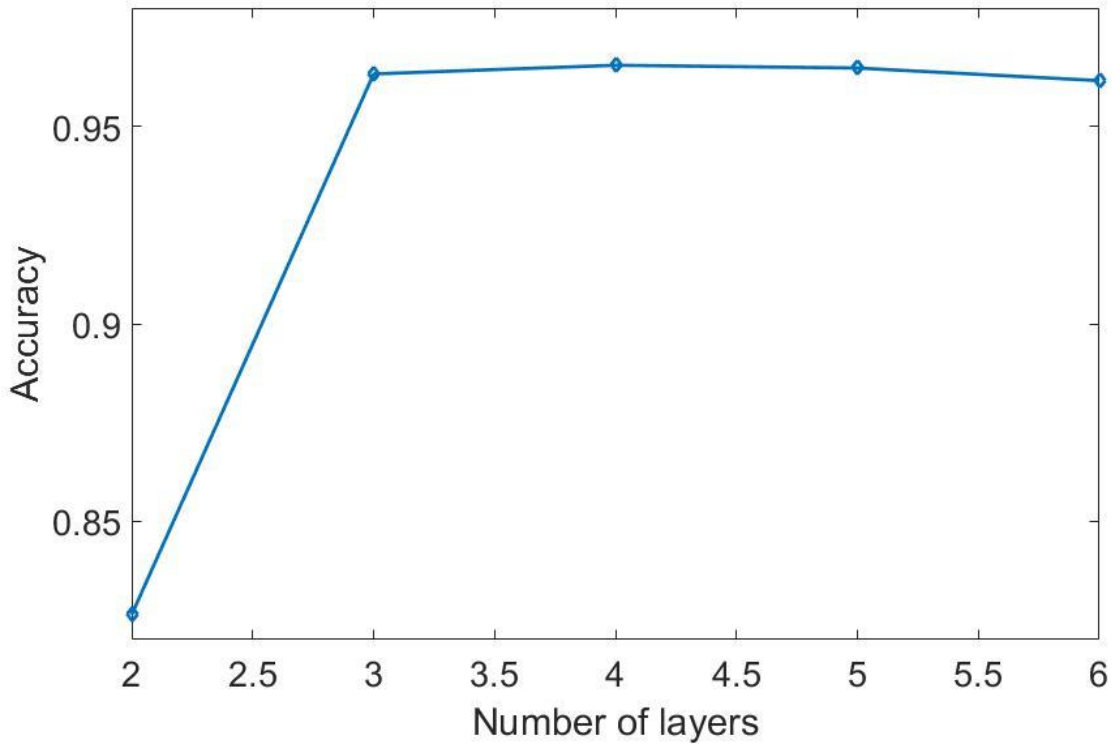
1. Learning rate

   First we test the relation of accuracy and learning rate. We vary the learning rate from 1 to 10. We firx number of layers at 3 (with 1 hidden layer) and neurons in each hidden layer at 88 which is the square root of sum of input and output size.



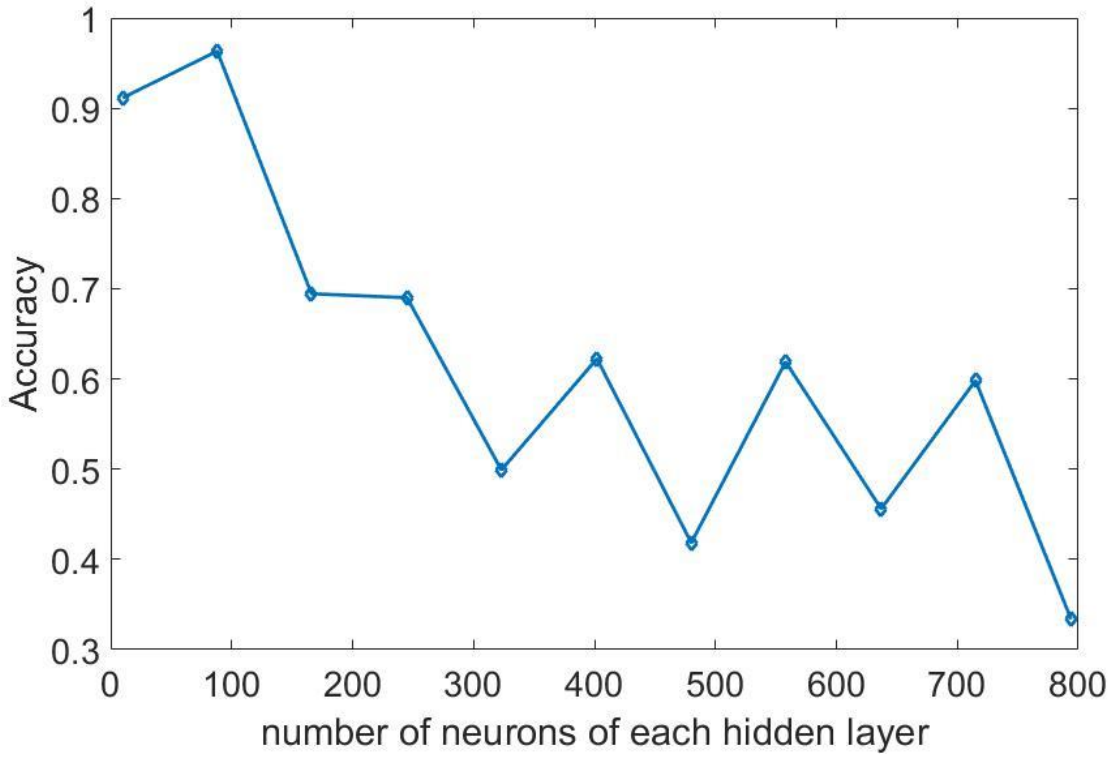
2. Number of layers

   We then use different number of layers for our NN. We fix learning rate at 1 and set number of neurons in each hidden layer at 88.



3. Number of neurons in each hidden layer

   Finally, to test the performance of different number of neurons in each hidden layer, we fix learning rate at 0.1 and number of layers at 3. We use int(image_size * 0.1 * i) + n_class to calculate the number of neurons, where i increases from 0 to 10.



Subject understanding:

1. Batch vs. online processing:

   Online processing use the most recent value in computations, so that the optimization process converges much faster. However, due to the extra dependencies at each layer, it does not fit well for parallel processing.

2. Gradient descent vs. stochastic gradient descent:

   Stochastic GD is simply a stochastic approximation of the GD to reduce the high computation cost introduced by large training sets.

3. Perceptron vs. sigmoid:

   Sigmoid is a continuous version of perceptron. The learning procedure of NN requires a

continuous function so that slight changes in input may not cause dramatic changes in output.

4. Feedforward vs. backpropagation:

   Feedforward is to use NN to get the classification results. Backpropagation is to use the classification errors to adjust the weights and biases of the network.