Project 2 - Convolutional Neural Network for MNIST classification

1st Anis Basak
Computer Science
North Carolina State University
Raleigh, United States
abasak3@ncsu.edu
200261580

2nd Vishal Shitole
Computer Engineering
North Carolina State University
Raleigh, United States
vgshitol@ncsu.edu
200258381

3rd FNU Vivek
Computer Science
North Carolina State University
Raleigh, United States
vvivek@ncsu.edu
200261356

I. Structure of Network

TABLE I
NETWORK IN TABULAR FORMAT - INITIAL NETWORK

Input Layer	Data from MNIST Input shape = (28, 28, 1)
Convolutional Layer	Number of Filters = 8
Pooling Layer	Pool size = (2, 2)
Convolutional Layer	Number of Filters = 16
Pooling Layer	Pool size = (2, 2)
Flatten	
Dense	20 nodes
Dense	10 nodes
Optimizer	Adam

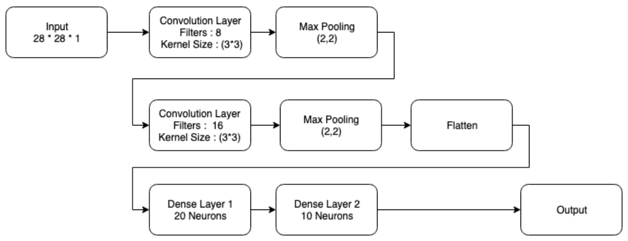

Fig. 1. Final Structure

II. The range of each hyper-parameters chosen for the experiment

The following are the hyper-parameters chosen for tuning:

- Kernel Size
- Epochs
- Batch Size
- Learning Rate

We have done a grid-search on the parameters which have given us the best value combination for :

- Kernel Size - (3, 3)
- Epochs - 15
- Batch Size - 128
- Learning Rate - 0.001

We have varied the all the parameters individually within a range and during this time all the other hyper-parameters have been kept to their best value that has been mentioned here. Activation function has been chosen as default i.e. Linear for tuning the hyperparameter and initial tuning. We will later apply the model with different activation functions and observe the performance.

Range of the parameters:

- Kernel Size: [2, 3, 4]
- Epochs: [5, 10, 15, 20]
- Batch Size: [16, 32, 64, 128, 256, 512, 1024]
- Learning Rate: [0.0001, 0.0005, 0.001, 0.002]

After tuning it on the above hyper-parameters, we have tested the varied the network with 3 activation functions *tanh*, *relu* and *sigmoid*. We have chosen the best activation function to test on our validation set.

III. Learning params for Tuning DNN Model

Kernel Size
From the graph we have best kernel size: 3 (Fig. 2)

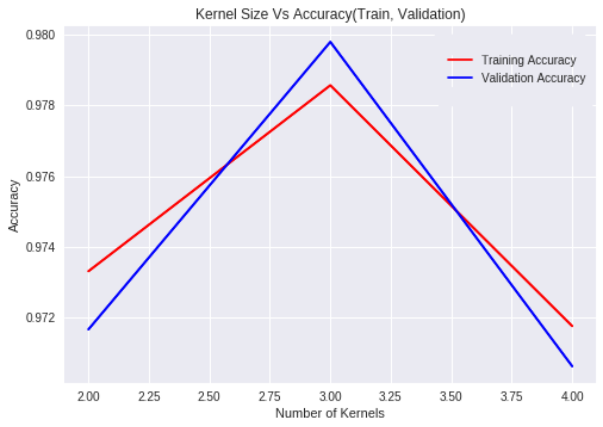

Fig. 2. Kernel Size

Epochs
From the graph we have best number of epochs: 15 (Fig. 3)

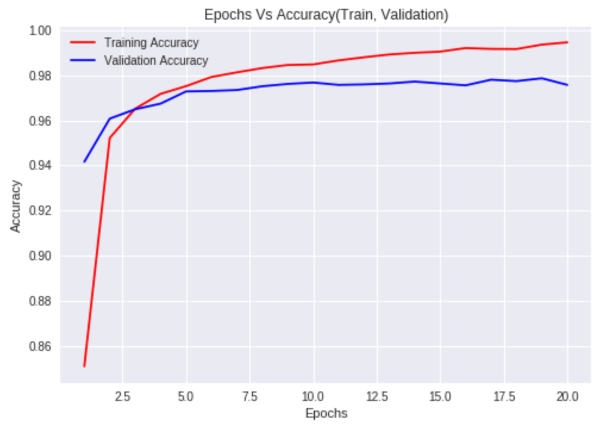

Fig. 3. Epochs

Batch Size
From the graph we have best batch size: 128 (Fig. 4

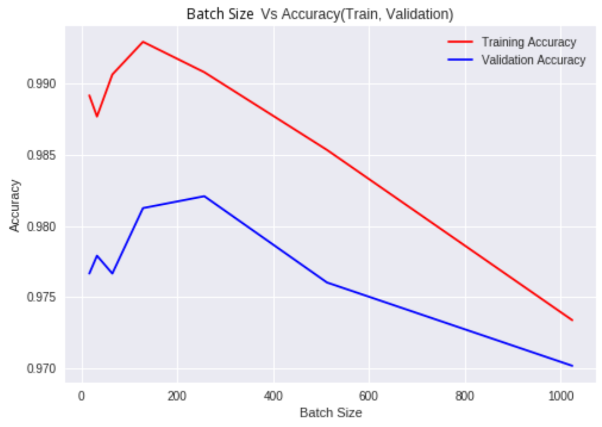

Fig. 4. Batch Size

Learning Rate
From the graph we have best batch size: 0.001 (Fig. 5)

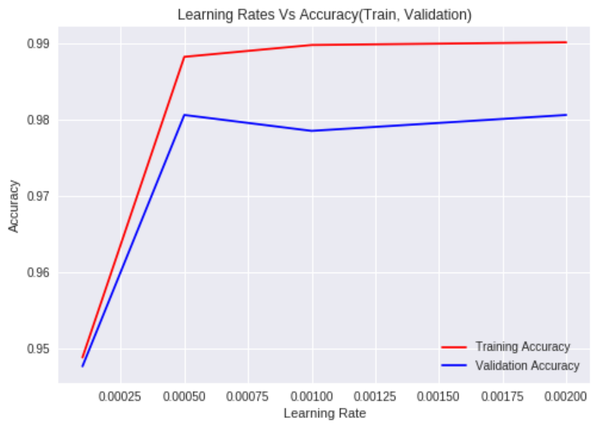

Fig. 5. Learning Rate

There may seem some discrepancies between the best value in the graphs and the best value given in by the grid-search. We have trained on randomly chosen training sets. Since we have done the training separately on 2 models this discrepancy is showing. That is, the models for training we not saved for both best value graph and grid-search, so there is bit of mismatch in the graphs and grid-search.

We have varied the activation functions based on the best values on the grid-search. The activation functions considered for the graphs are tanh, sigmoid, relu.

We find the best accuracy is obtained for *tanh* activation.

Final structure of the network
Epoch: 15
Batch Size: 128

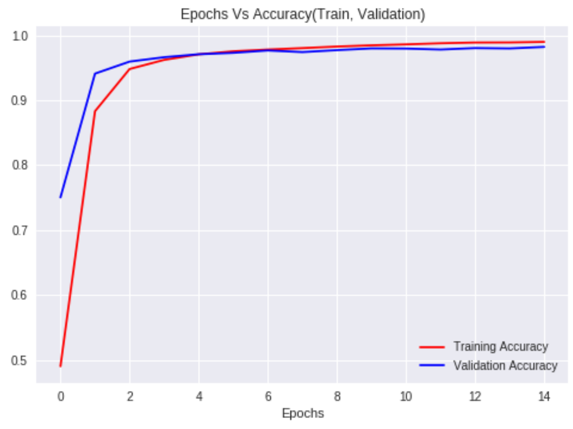

Fig. 6. Activation Relu Accuracy - y axis denotes accuracy

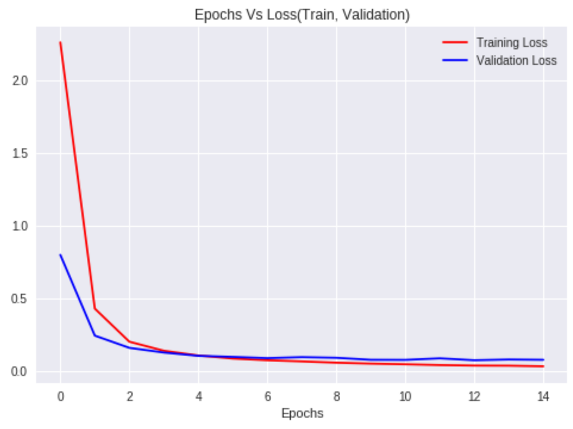

Fig. 7. Activation Relu Loss - y axis denotes loss

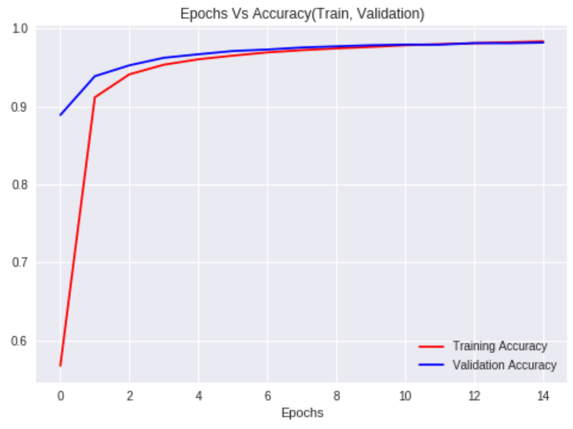

Fig. 8. Activation Sigmoid Accuracy - y axis denotes accuracy

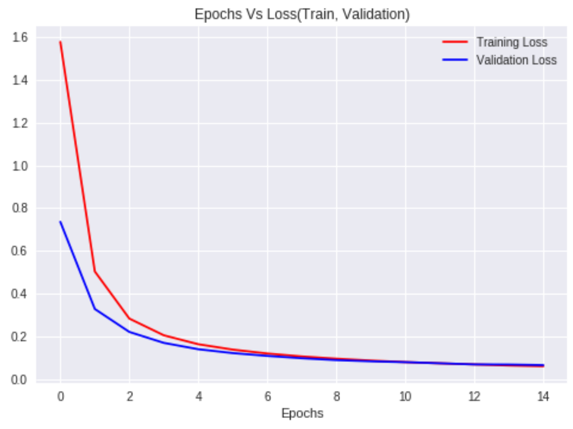

Fig. 9. Activation Sigmoid Loss - y axis denotes loss

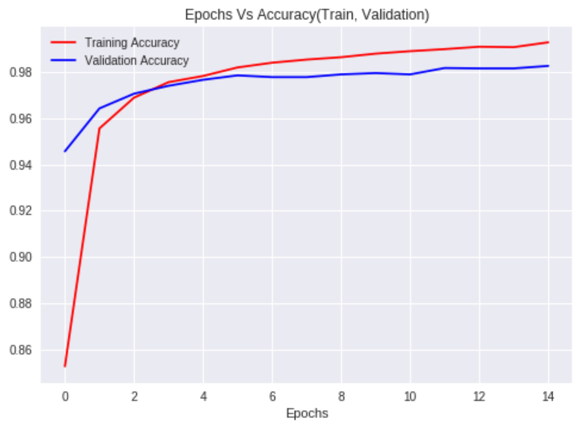

Fig. 10. Activation Tanh Accuray - y axis denotes accuracy

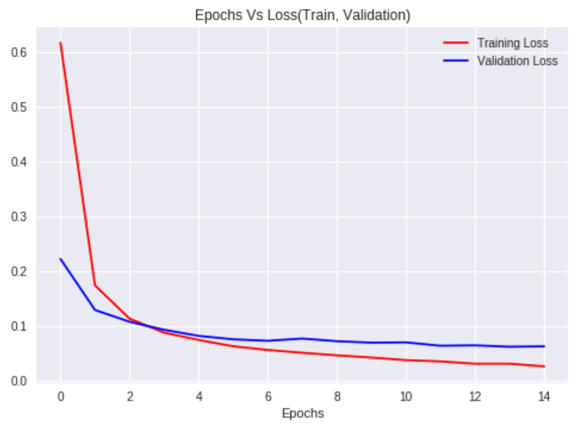

Fig. 11. Activation Tanh Loss - y axis denotes loss

IV. Accuracy on testing set

Before optimization:
Training accuracy : 0.9937
Testing accuracy: 0.9773

After Optimization:
Training accuracy : 0.9932
Testing accuracy: 0.9825

TABLE II
NETWORK IN TABULAR FORMAT - AFTER HYPER-PARAMETER TUNING

Input Layer	Data from MNIST digits set	
Convolutional Layer	Input shape = (28, 28, 1)	Activation: tanh, Kernel Size: 3, Number of Filters: 8
Pooling Layer	Pool size = (2, 2)	
Convolutional Layer		Activation: tanh, Kernel Size: 3, Number of Filters: 16
Pooling Layer	Pool size = (2, 2)	
Flatten		
Dense	20 nodes	Activation: tanh
Dense	10 nodes	Activation: softmax
Optimizer	Adam	Learning Rate: 0.001

V. ANALYSIS AND DISCUSSION

From the figures for activation versus loss and accuracy (Fig. 6 to Fig. 11), the loss decreases the fastest for *relu*, while sigmoid is slowest to converge. This is evident from the respective shape of the function and also from the graphs. Almost, similar accuracy is achieved by all three activation functions. We could have selected *relu* as our final activation function to decrease the time for training. However, as the training time is not large, we resort to the activation that gives the best accuracy here i.e. *tanh*
We selected only some of the many hyper-parameters that can be tuned. But we did select the hyper parameters that have major impact, like batch size and epochs and learning rate. Specifically, grid search with cross validation (2-fold) was applied as it is better to test the combinations of the hyper-parameters, rather than fixing one and tuning the other.