Convolutional Neural Network for MNIST Classification

1st Nazneen Kotwal
Dept. of Electrical Engineering
North Carolina State University
Raleigh, USA
nkotwal@ncsu.edu

2nd Priya Diwakar
Dept. of Electrical Engineering
North Carolina State University
Raleigh, USA
psdiwaka@ncsu.edu

3rd Sharvari Deshpande
Dept. of Electrical Engineering
North Carolina State University
Raleigh, USA
shdeshpa@ncsu.edu

***Abstract*—Design and implementation of a Convolutional Neural Network for MNIST Classification.**

***Index Terms*—CNN, MNIST, hyperparameter, activation functions, cross-validation**

I. Introduction

The goal of this project is to develop a convolutional neural network for MNIST classification. A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers followed by one or more fully connected layers as in a standard multilayer neural network. In this project, a CNN is used for MNIST classification. The hyperparameter values of the model are varied till optimum values are obtained. These values will give the best performance for the given model.

II. Network Structure

A. Architecture of the Network

We started with the MNIST 28X28 grey scale images of digits (source: http://yann.lecun.com/exdb/mnist/). We then create 32, 33 convolutional filters with ReLU (Rectified Linear Unit) node activations. After this, we still have a height and width of 28 nodes. We then perform down-sampling by applying a 22 max pooling operation with a stride of 1. Layer 2 consists of the same structure, but now with 64 filters channels and another stride-1 max pooling down-sample. We then flatten the output to get a fully connected layer with 128 nodes. These layers will use ReLU node activations. Finally, we use a softmax classification layer to output the 10 digit probabilities.

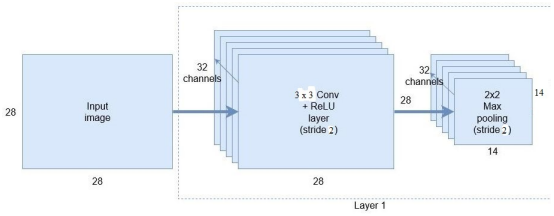

Fig. 1. Layer-1 Architecture

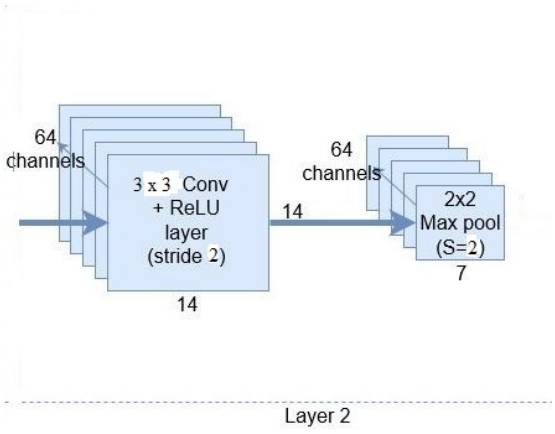

Fig. 2. Layer- 2 Architecture

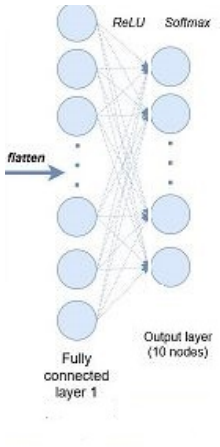

Fig. 3. Final Layer Architecture

III. Hyperparameter Selection

The hyperparameters that were tuned in this project were the learning rate, batch size and activation function. Learning rate is defined in the context of optimization, and minimizing the loss function of a neural network.An optimum learning rate for the model needs to be found since a high learning rate will result in overshooting which can lead to the model never

converging to the global minima and a low learning rate will take a long period of time for training. Hence, optimizing this parameter is vital. The second parameter the team optimized was the batch size. Batch size refers to the number of training examples utilized in one iteration. The activation function was also varied to obtain training, validation loss and accuracy each.

A. Range of Hyperparameters

The range of hyperparameters that were used for learning rate were 0.001,0.003,0.01, and 0.03. For each of these learning rates, training, testing and validation accuracy and loss were calculated for batch sizes 64,128 and 256. The training accuracy and loss, validation accuracy and loss were also obtained for different activation functions. The activation functions applied were ReLU, tanh and sigmoid.

B. Cross-Validation Visualization

For hyperparameter tuning, the data is split as shown in Fig. 4.

```
train: 0.57% | validation: 0.28% | test 0.14%
```

Fig. 4. Cross Validation Visualization

C. Hyperparameter Tuning

The learning rates taken into consideration for this model were 0.001,0.003,0.01,0.03.For each of these learning rates, training and validation accuracy and, training and validation loss were observed. It was observed that the training accuracy significantly decreased with increasing learning rate. This was true for validation accuracy as well. Both the training and validation losses increased with an increase in learning rate.

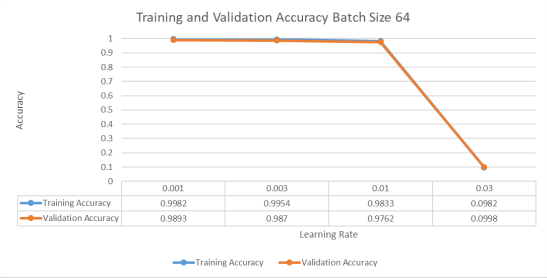

Fig. 5. Accuracy visualization for batch size 64

IV. Results

A. Learning Curve for Activation functions

1) Accuracy: As observed from figure 11, figure 12 and figure 13, the training accuracy is maximum for the activation function tanH. The validation accuracy is maximum for activation function ReLU.

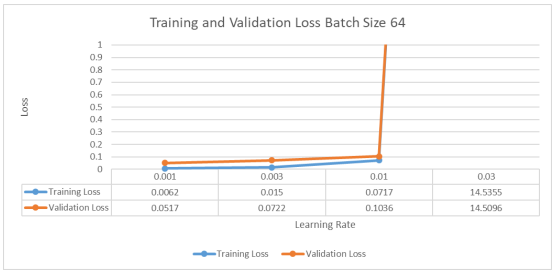

Fig. 6. Loss visualization for batch size 64

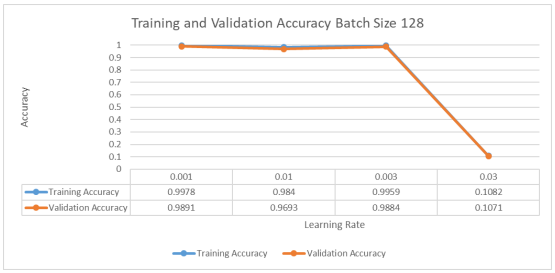

Fig. 7. Accuracy visualization for batch size 128

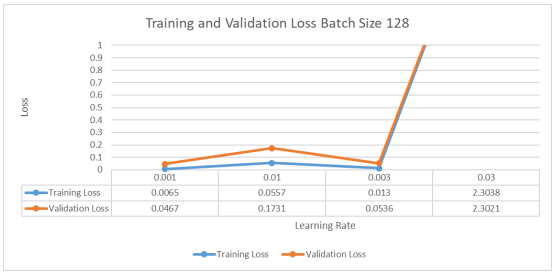

Fig. 8. Loss visualization for batch size 128

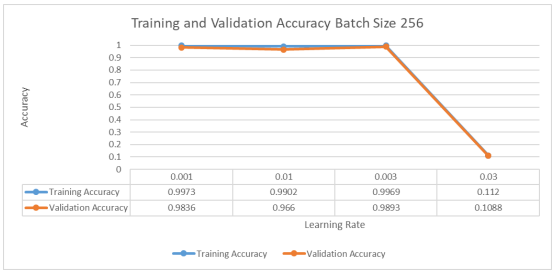

Fig. 9. Accuracy visualization for batch size 256

2) Loss: As observed from figures 14,15 and 16, the training loss for activation function tanH is the least and validation loss is the least for ReLU activation function.

B. Testing Set Accuracy

The test set accuracy that was obtained on this model for learning rate=0.001 and batch size=128 was 99.31 percent.

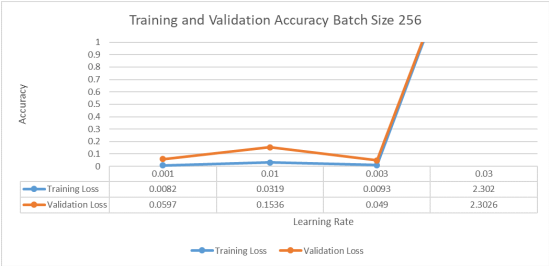

Fig. 10. Loss visualization for batch size 256

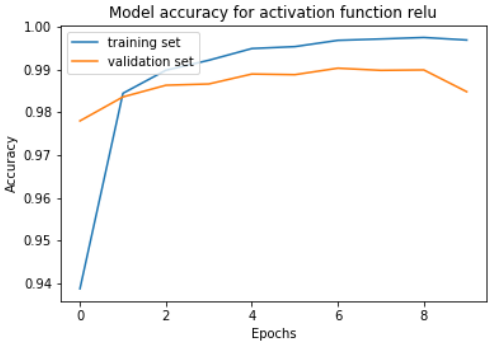

Fig. 11. Model accuracy for activation function ReLU

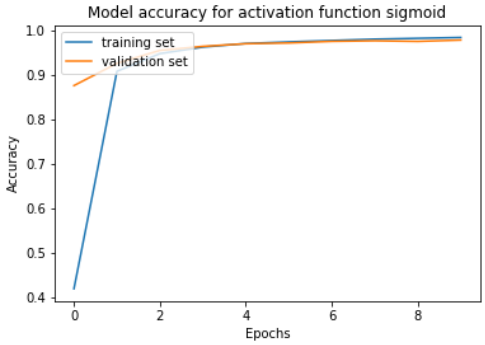

Fig. 12. Model accuracy for activation function Sigmoid

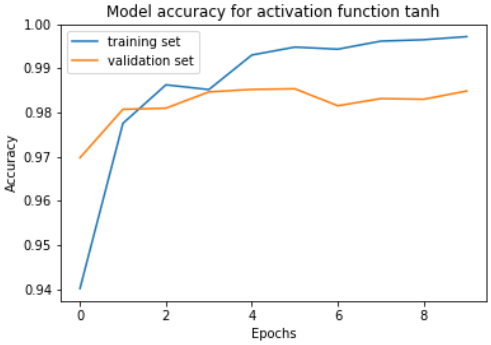

Fig. 13. Model accuracy for activation function tanH

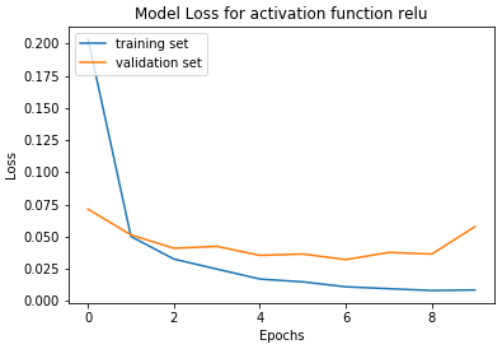

Fig. 14. Model loss for activation function ReLU

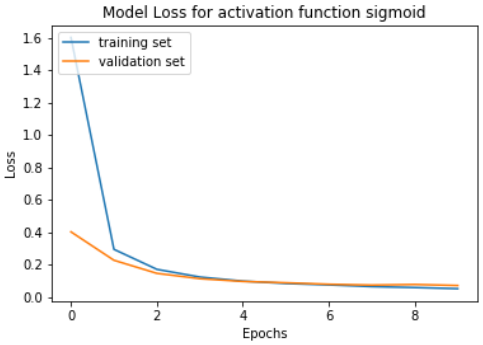

Fig. 15. Model loss for activation function Sigmoid

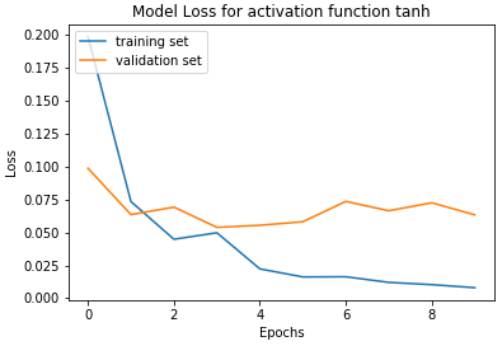

Fig. 16. Model loss for activation function tanH

V. Conclusion

For this model, maximum accuracy was obtained for a learning rate of 0.001, batch size of 128, and activation function ReLU. For these values, the cost function is minimized.

References

[1] Goodfellow-et-al-2016, Deep Learning, MIT Press, 2016.
[2] keras-team, keras-team/keras, GitHub, 07-Dec-2017. [Online]. Available: https://github.com/keras-team/keras/blob/master/examples. [Accessed: 01-Oct-2018].