# Convolutional Neural Network for classification in MNIST dataset


Aditya Govil
*Department of Computer Science*
*North Carolina State University*
Raleigh, USA
agovil@ncsu.edu

Priyankha Bhalasubbramanian
*Department of Computer Science*
*North Carolina State University*
Raleigh, USA
pbhalas@ncsu.edu

Harsh Agrawal
*Department of Computer Science*
*North Carolina State University*
Raleigh, USA
hagrawa2@ncsu.edu


## I. Introduction

This document is describing the hyper-parameter tuning for a convolutional neural network which is used to classify digits in a MNIST dataset.

## II. Dataset

The MNIST database of handwritten digits, is used in this project. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

## III. Final Structure of Network and other hyperparameters

We used a Convolution Neural network with 2 layers. The first layer has input images of size 28X28X1 with 28 filters, kernel size of 3, stride of 1 and reLU activation function. After this, it has max pool layer of 2X2 filter with no stride ( means there is no down scaling ). At the end there is a flatten layer to convert the 3D data into 1D data

Deep Neural Network has 3 layers. The input layer has values in the range of 0 and 1 because of reLU activation function in CNN. The first hidden layer has 128 nodes with dropout and reLU as activation function. The output layer has 10 nodes and sigmoid activation function.

| | |
|---|---|
| Output size of 2D Convolution layer | 26X26X28 |
| Output size of Max Pooling layer | 26X26X28 |
| Number of neurons in input layer ( O/P of flatten layer ) | 18928 |
| Number of neurons in hidden layer | 128 |
| Number of neurons in output layer | 10 |
| Batch Size | 128 |
| Learning Rate | 0.001 |

## IV. Learning curve (loss and accuracy) on both training and validation set

The learning curves for training and validation have been plotted for each combinations of hyper parameters in grid search.

We fixed the learning rate parameter to best value which was 0.001 and varied the batch size to find the best combination. We plotted the accuracy and error graphs for both testing and validation data-set. Results are in figure 1.

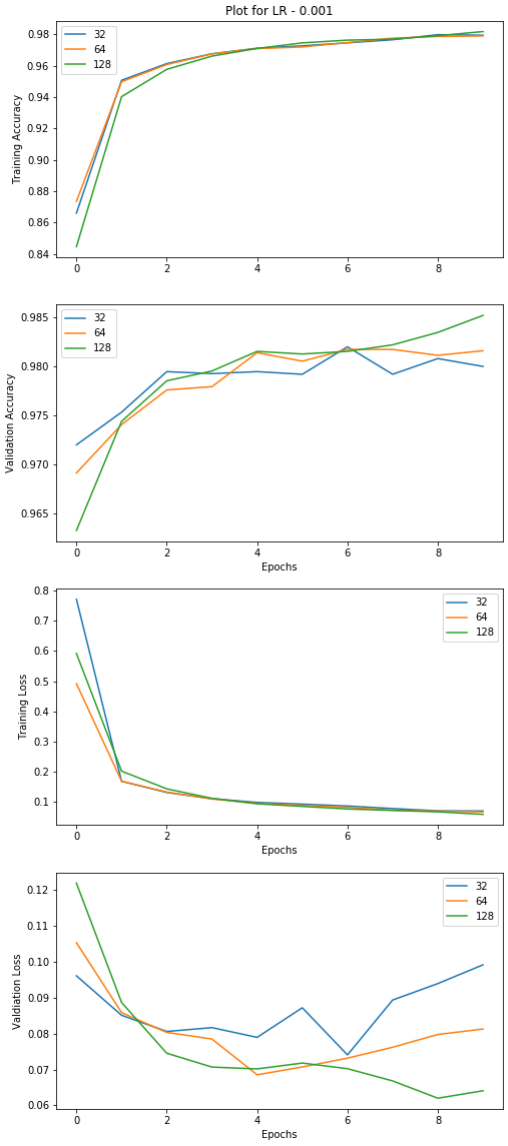


Fig. 1. Learning curves for different batch sizes for training and validation data

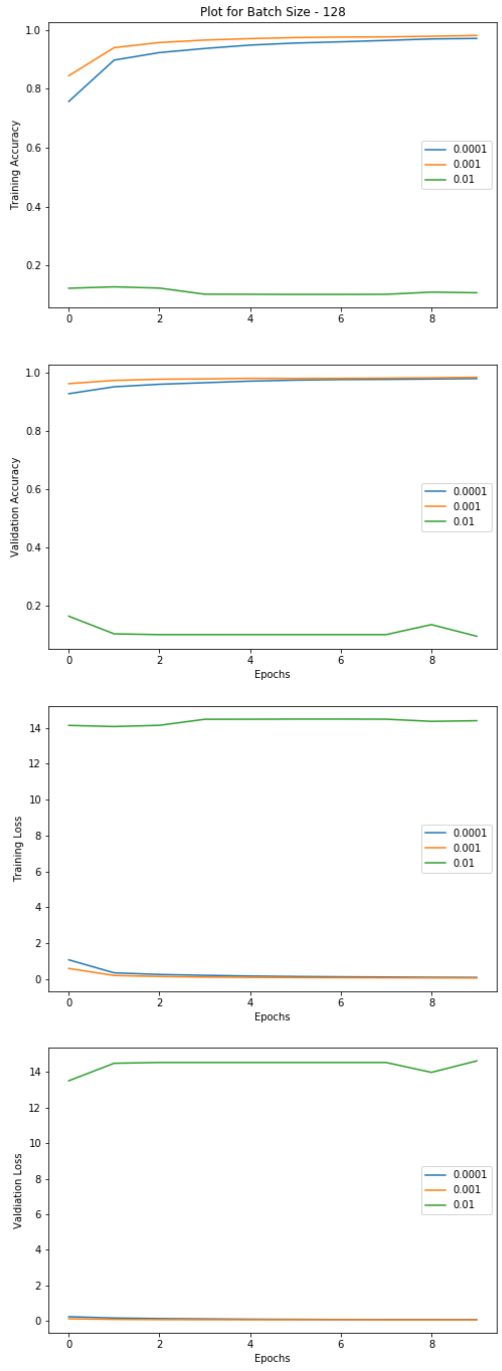


Fig. 2. Learning curves for different learning rate for training and validation data

After, this we fixed the batch size parameter to best value which was 128 and varied the learning rate to find the best combination. We plotted the accuracy and error graphs for both testing and validation data-set. Results are in figure 2.

Through this, the best combination or our hyper-parameters was 0.001 learning rate and 128 batch size. To further analyze the performance we have recorded the accuracy and error on training and validation data-set for different activation functions. Results are in figure 3, 4, 5 and 6.

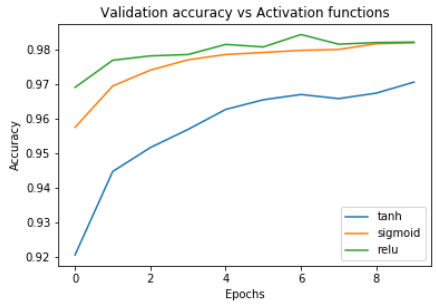


Fig. 3. Validation accuracy for different activation functions across epoch

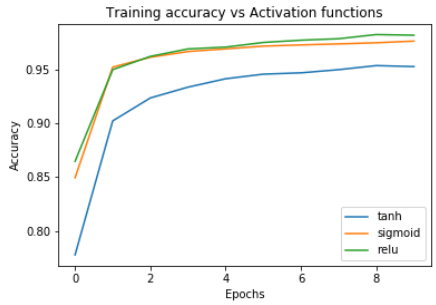


Fig. 4. Training accuracy for different activation functions across epoch

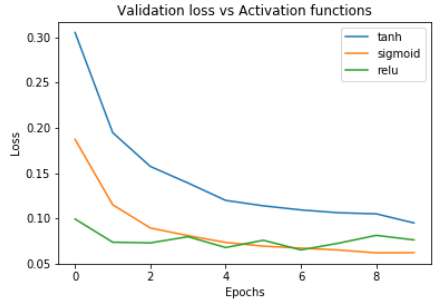


Fig. 5. Validation loss for different activation functions across epoch

## V. Accuracy on Testing Set

We trained the model on the entire training data (without splitting it to validation data) using the best hyper-parameters values for learning rate (0.001) and batch size (128). The trained model was evaluated on test data and a test accuracy of **98.5%** was obtained.

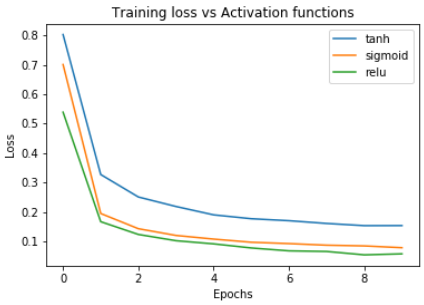


Fig. 6. Training loss for different activation functions across epoch

## VI. Acknowledgment

We would like to acknowledge our professor, Edgar Lobaton and our TA Qian GE and Turner Richmond for their guidance.

## References

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, “Deep Learning“