Project 3 - Covolutional Neural Network for MNIST classification

1st Nirmal Elamon
Dept of ECE
North Carolina State University
Raleigh, USA
nelamon@ncsu.edu

2nd Ojas Barve
Dept of ECE
North Carolina State University
Raleigh, USA
ovbarve@ncsu.edu

3rd Kalyan Ghosh
Dept of ECE
North Carolina State University
Raleigh, USA
kghosh@ncsu.edu

***Abstract*—This document is the complete report for the Project-3 (CNN for MNIST classification) as a part of the course ECE 542 (Neural Networks & Deep Learning) during the Fall 2018 semester.**

I. Introduction

This document contains the following subsections :
1. Dataset & Deep Learning Framework
2. Structure of the network
3. Range of each hyper parameters
4. Visualization of cross-validation for hyper parameters
5. Learning curves (loss and accuracy)
6. Accuracy on testing set
7. Analysis of our experiment
8. Accuracy on testing set

II. Dataset & Framework

In our project we have used the MNIST dataset for our training/validation/testing purposes.The training,validation and testing set split that we have used in our project are described below.

- The total number of images present in the MNIST dataset is 70000.
- The dataset is split into sizes of 60000 and 10000 as the training set and the testing set respectively.
- The training set is further split into sizes of 50000 and 10000 as the training set and the validation set.
- For this project, we have used the Keras Deep Learning Library.Keras is a high-level neural networks API, written in Python and capable of running on top of Tensor Flow backend.Keras is a userfriendly, scalable and modular library and runs seamlessly on CPU & GPU.

III. Structure of Network

- The input to the CNN is a 1x28x28x50000 vector. Here 28x28 is the dimension of a single MNIST image.
- The input image is then convolved with 32 filter kernels of size 3x3 to generate 32 feature maps.In our project we have used RELU non-linearity.
- The 1st convolution layer is again passed through another convolution layer with kernel size of 3x3 to generate 64 feature maps.

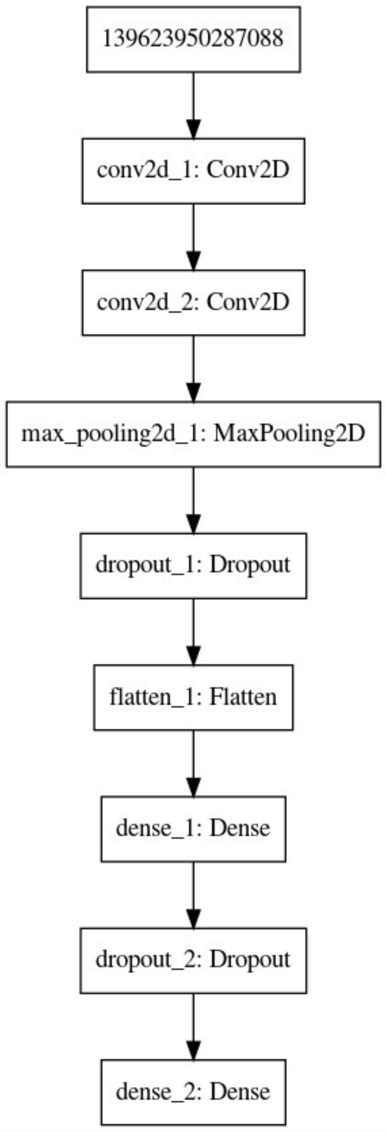


Fig. 1. NETWORK STRUCTURE

- Max Pooling is then applied on the feature maps to reduce the spatial dimensionality. Here the pooling size is taken as 2x2.
- Then Random Dropout is applied with probability of 0.25.
- The feature maps are then flattened and a densely connected layer of dimension 128 is added with RELU non-linearity.
- The 1st densely connected layer is then passed through a Random Dropout layer with probability of 0.50
- Then a densely connected Fully Connected Layer of dimension 128 is added which ultimately is fed into the 10 class output layer with Softmax classification.

IV. Range of each hyper parameters

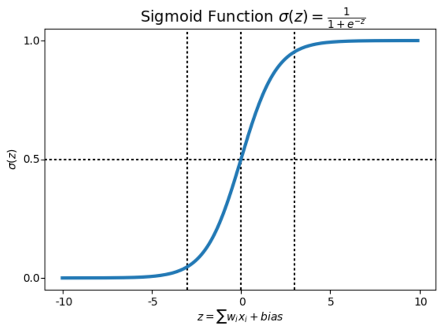
- In this project as mentioned, we have selected "Learning Rate" & "Batch Size" as the set of hyper parameters.Learning Rate is the smallest step that the optimization procedure take towards the gradient of steepest descent.Batch Size is the number of training examples present in a single batch. Hyper parameters are the parameters which are not learned by the network. The best values of these parameters are calculated experimentally for a particular task.
- Then one-fold cross validation techniques is used to select the hyper parameters.Cross validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.
- Grid Search technique is then used to select the optimal set of hyper parameters for this model.Grid Search is simply an exhaustive searching through a manually specified subset of the hyper parameter space of a learning algorithm.
- The hyper parameter selection technique is described as below.
- The range of values of "Learning Rate" that we used are as follows : [0.01,0.001,0.0001]
- The range of values of "Batch Size" that we used are as follows : [32, 64, 128]
- We have used Stochastic Gradient Descent as our optimization algorithm with Initial Learning Rate as 1, Learning Decay of 1e-6 and Momentum of 0.9.
- We then train our model for 5 epochs and calculate the Training/Validation loss and accuracy for all the combinations of Learning Rate and Batch Sizes.

A. Activation functions

After getting the optimal values for learning rate and bath size through hyper parameter optimization, the next step is to use these optimal values for finding the activation functions. We tested on three different activation functions which are: Sigmoid, Tanh and RELU The results are summarized below:

- Sigmoid

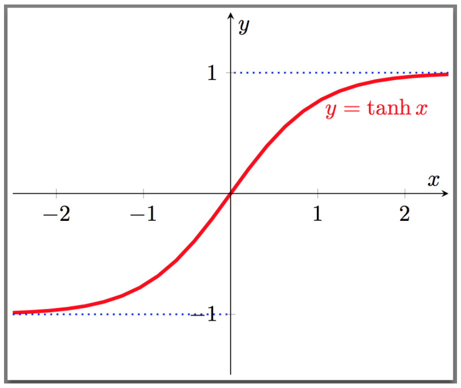
$$S(x) = 1/(1 + e^{-x}) \tag{1}$$



  - Disadvantages
    1) Activation saturates at 0 or 1 with a gradient = 0. Due to this, no signal will be there to update weights. This it cannot learn properly. Towards either end of sigmoid function, y values tend to respond very less to changes in x.
    2) Sigmoid outputs are not zero centered. If the data coming into a neuron is always positive, gradient of the weight will become either all positive or all negative depending on gradient of the whole expression. This can introduce undesirable zig-zag dynamics in gradient updates for the weights.

- Tanh

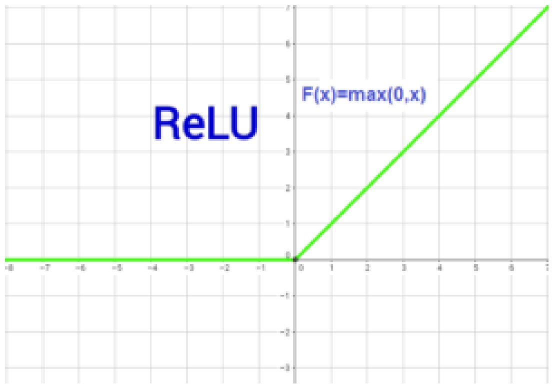
$$S(x) = 2/(1 + e^{-2x}) \tag{2}$$



  - Advantage over Sigmoid
    Outputs are zero centered
  - Disadvantage
    Activation saturates at 0 or 1 with a gradient = 0.

- RELU

$$S(x) = max(0, x) \tag{3}$$

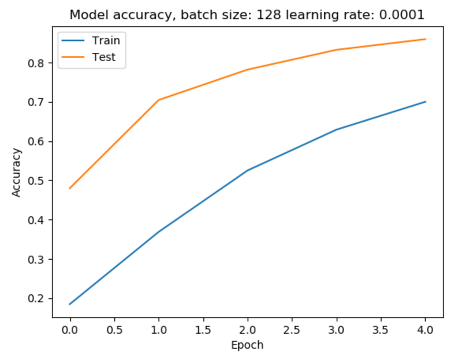


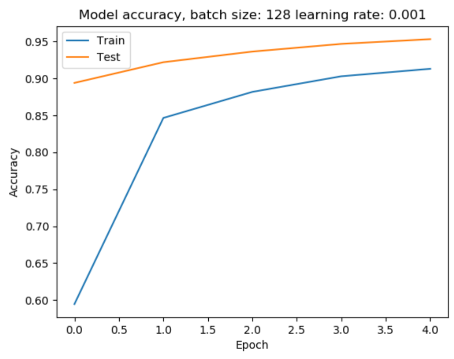
- Advantages
  1) Accelerates convergence
  2) Less computationally expensive
  3) Sparsity

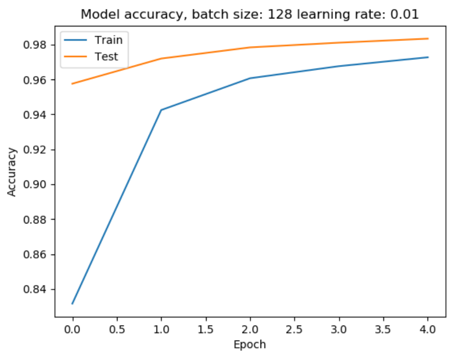
V. LEARNING CURVES

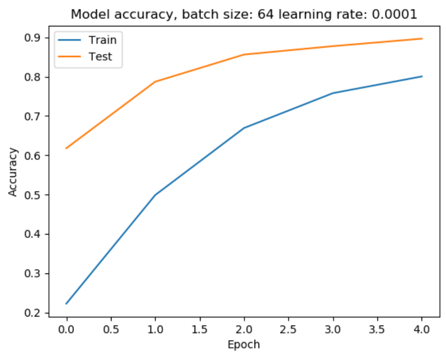
Learning Curves refer to plots of the prediction accuracy/error vs. the training set/testing set which gives us a visual indication of how well is the model generalizing to new unseen data.

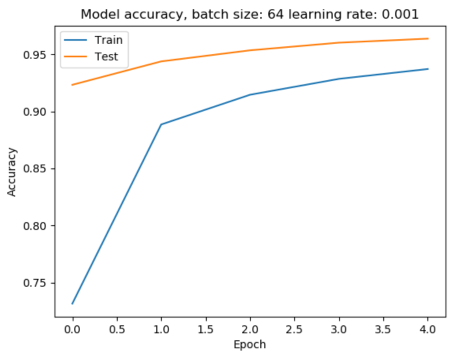
- Model Accuracy:
  Plots of Model Accuracy for various combinations of Learning Rate & Batch Size.

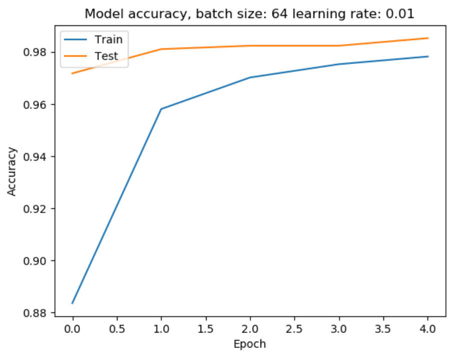


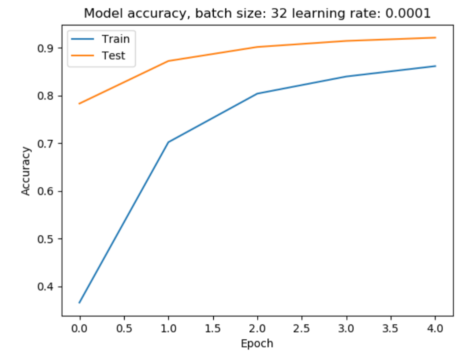


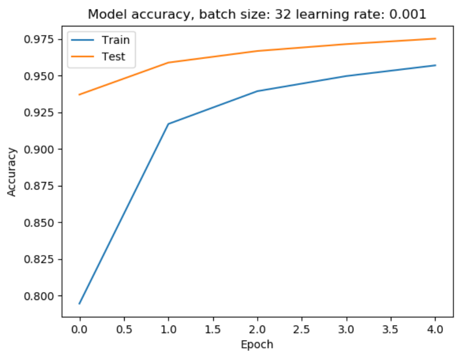


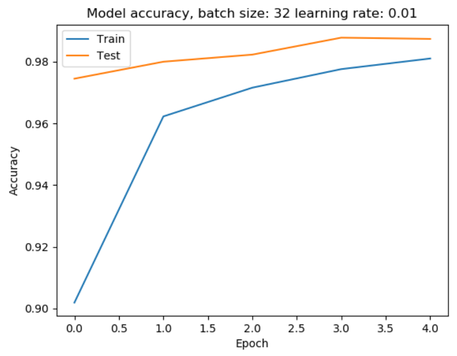




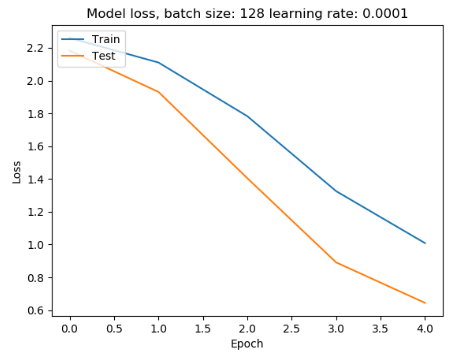


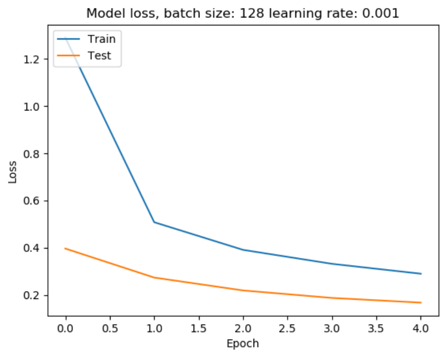


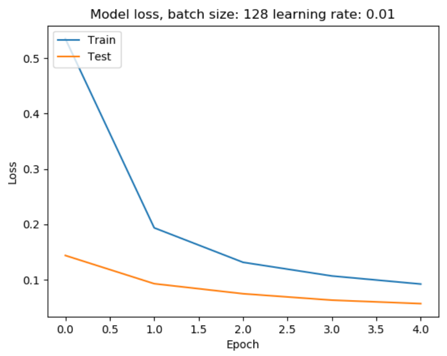


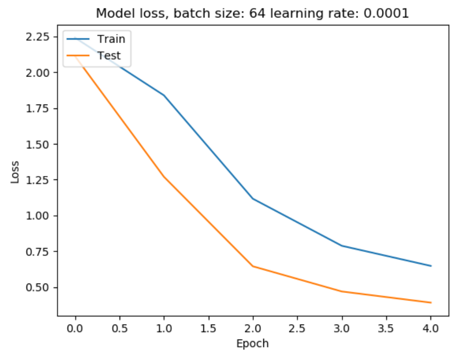


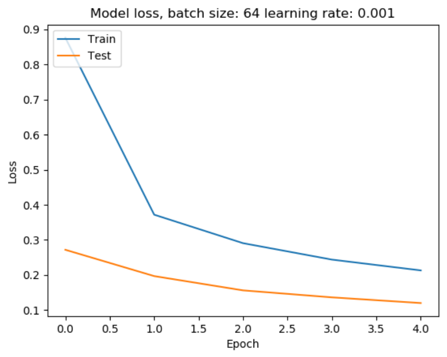
- Model Loss:
  Plots of Model Loss for various combinations of Learning Rate & Batch Size.

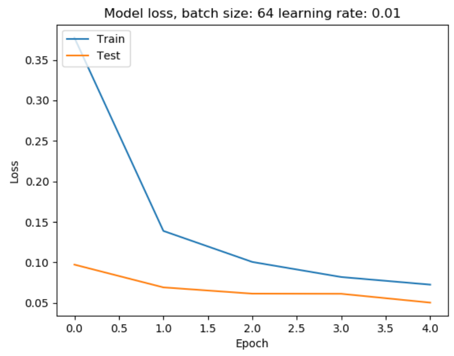


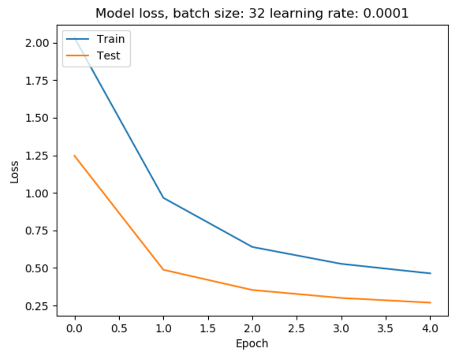


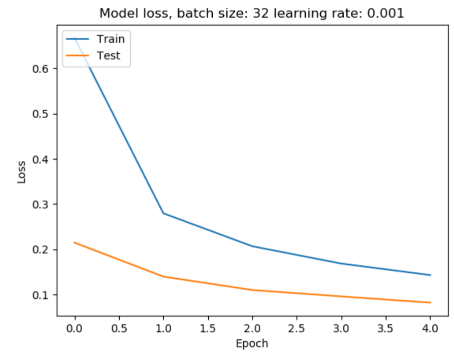


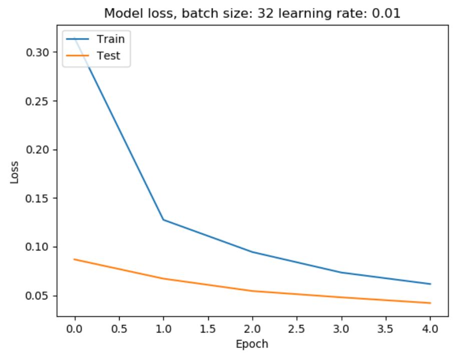










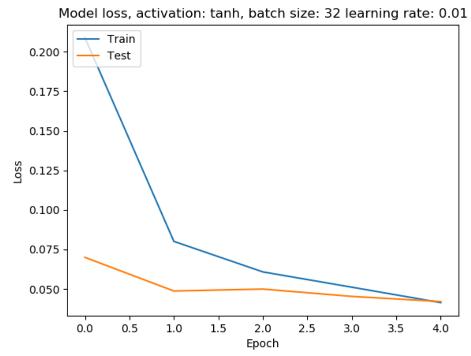


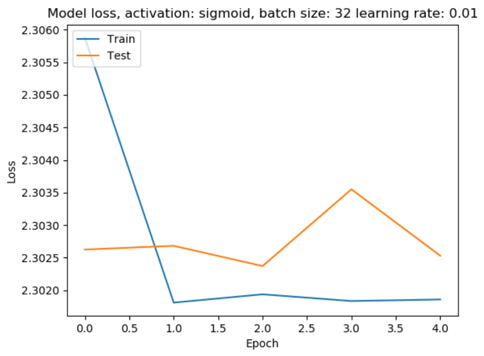
A. Activation Function

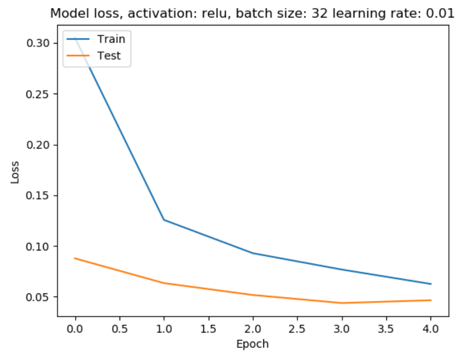
From the hyper parameter tuning analysis, a learning rate = 0.01 and batch size = 32 was obtained as the optimal value with the minimum validation loss of 0.0369 and validation accuracy of 98.79% compared to the other combinations of learning rates and batch sizes.

Using these values, the model was again trained on 3 different activation functions: Sigmoid, Tanh and ReLU. The following graphs summarizes the analysis:

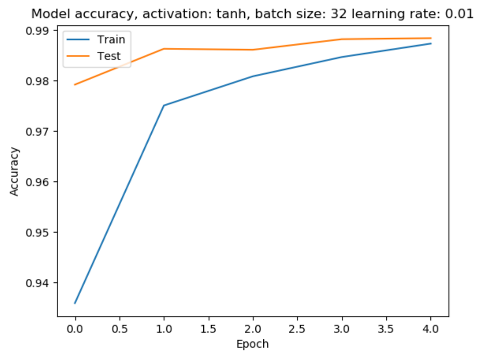
- Model Loss:
  Model Loss for different activation functions.

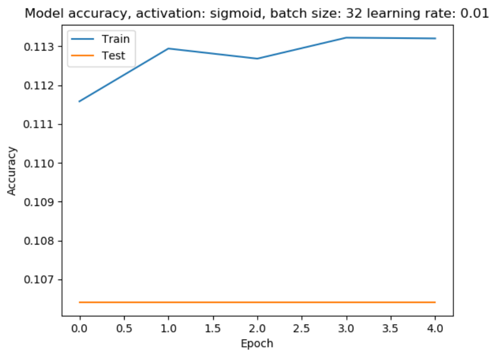


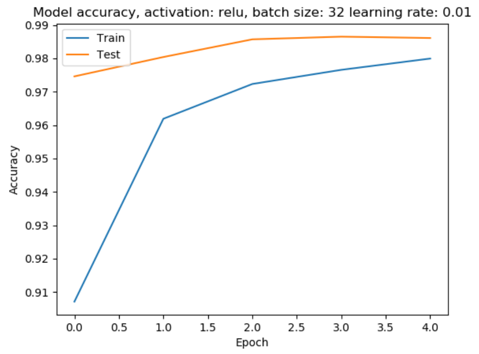




- Model Accuracy:
  Model Accuracy for different activation functions.







B. *Accuracy on testing set*

The following are the optimal hyper parameters that was obtained after proper hyper parameter tuning:

Batch size = 32

Learning rate = 0.01

Activation function = Tanh

Using these hyper parameters, the model was again trained on training + validation data which was tested on the testing data to obtain the following measures:

Test accuracy = 98.72%

Test loss = 0.04035

VI. Justification of hyper-parameters

The Network was evaluated for three hyper parameters namely learning rate, batch size and activation functions at hidden layers. A grid search was performed on batch size and learning rate as the two input hyper parameters. We tested for the permutations of the following namely learning rate as 0.1, 0.01, 0.001, 0.0001, 0.00001 and batch size as 32, 64, 128. After running our grid search algorithm we reached the conclusion that the network gave the best outputs for the batch size of 32 and learning rate of 0.01 for Relu activation functions. Next up we tested the network using the previously obtained values of hyper parameters namely learning rate - 0.01 and batch size 32 on different activation functions. We tried 3 different activation functions namely Relu, Sigmoid and Tanh. We obtained the best test results for the Tanh activation function. We conclude that the results obtained on the said hyperparameters were better than other permutations that we experimented as the Tanh activation has a steeper learning rate than Relu and converges faster in our scenario as we used only 5 epochs to train our network and also the learning rate being used i.e. 0.01 is on the higher side. Also due to a smaller size of the network i.e. the network being comparatively shallow we did not observe any problem of exploding gradients with Tanh as an activation function. We can obtain a better performance with Relu as an activation by increasing the number of epochs as it will take more time to converge to optimal values as it introduces sparsity into the model as ca be seen by below output.

The output for 12 epochs for Relu activation:

Epoch 12/12

loss: 0.0255 - acc: 0.9913 - val$_loss : 0.0268 - val_acc : 0.9906$

$Testloss$: 0.0267825192

Clearly we can see that we get better performance for Relu activation with higher epochs on the same conditions but the increase is marginal at the cost of resources utilized in form of time and number of epochs. Sow we can get a comparable performance with Tanh activation for just 5 epochs without losing much on accuracy as shown below.

Final Log For combination of with activation as tanh and batch size:32 and learning rate:0.01

Epochs 5

Test loss:0.04035723799202824

Test accuracy:0.9872