Implementation of CNN on MNIST

Shubham Miglani
Department of ECE
North Carolina State University
smiglan@ncsu.edu

I. Introduction

The aim of the homework was to implement CNN using a deep learning framework (TensorFlow) for MNIST Data set. Using the validation data, the hyperparameters were tuned. The learning curves were recorded for loss and accuracy for the final network. Finally, the network was tested on the test data.

II. Method

A. Dataset

The training images were split into training set (50000) and validation set (10000) to implement one-fold cross-validation.

B. Procedure

The Procedure involved first reshaping the datasets and normalizing them between 0 and 1. The optimizer used was Adam, the loss was sparse categorical cross-entropy and the models were trained for 30 epochs. A script was created to run various combinations of hyperparameters and record the data. The training was done in Google Colab to utilize the GPU resources and make the training faster. The activation function for the final layer was SoftMax.

III. Hyperparameter Tuning

Various Hyperparameters were varied and their effects were studied. As training takes time, these hyperparameters were varied individually and selected based on the highest validation accuracy and the least loss. The set of hyper-parameters that were studied are Number of neurons in the dense layer, learning rate, Feature map size, kernel size, dropout probability, convolutional layers, activation function. The default values of the hyperparameters were as follows:

Table 1
Default Hyperparameter Values

Hyperparameter	Default value
Neuros in Dense Layer	128
Learning Rate	0.001
Feature Map Size	32
Kernel Size	3
Number of Convolution Layers	3
Dropout Probability	0.4
Batch Size	32
Activation Function	Relu

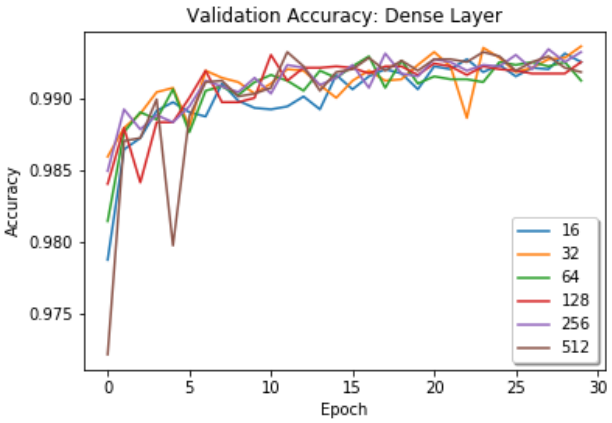
A. Number of neurons in the hidden layer

Firstly, the number of neurons in the dense layer were varied as shown in Table2. The rest of the parameters were kept the same as the default configuration.

Table 2
Validation Accuracy for change in neurons in dense layer

Number of neurons	Validation accuracy
16	0.9926
32	0.9937
64	0.9913
128	0.9926
256	0.9933
512	0.9919

The learning curve for the validation set is as shown:



Based on table 2, the layer with the highest validation accuracy is the layer with 32 neurons. But as can be seen in the above figure that although the validation accuracy for 32 neurons is highest, there is more fluctuation in it. As compared to it, the layer with 128 neurons seems more stable and reduces overfitting hence 128 is selected. Also, it is seen that when the number of neurons in the dense layer increases, the accuracy in the starting epochs increases but the validation accuracy at the end is close to each other.

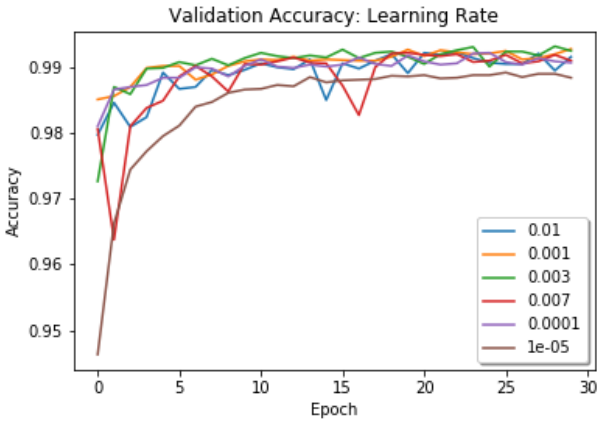
B. Learning rate for Adam Optimizer

The learning rate for Adam was varied keeping the other parameters as the default configuration

Table 3
Validation Accuracy for change in Learning Rate

Learning Rate	Validation accuracy
0.01	0.9917
0.001	0.9929
0.003	0.9926
0.007	0.9911
0.0001	0.9909
0.000001	0.9885

As can be seen from Table 3, when the learning rate decreases, the validation accuracy increases up to a point after which it starts decreasing. The learning curve for the validation set is as shown.



Based on this, 0.001 was selected as the learning rate.

C. *Feature Maps*

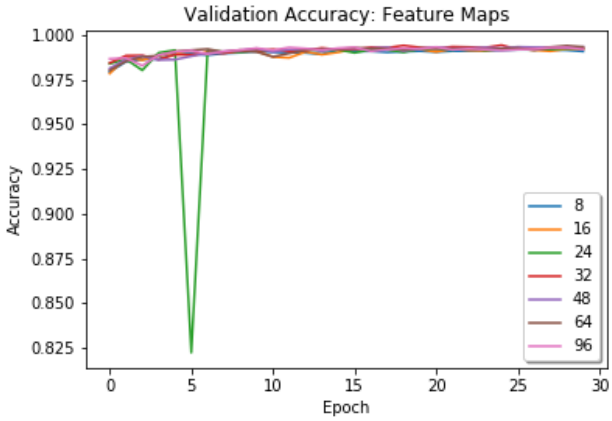
Then the number of feature maps in the convolutional layer were varied as shown in Table 4.

Table 4
Validation Accuracy for change in Feature maps

Maps	Validation accuracy
8	0.9909
16	0.9921
24	0.9924
32	0.9927
48	**0.9934**
64	**0.9934**
96	0.9920

As can be seen, the validation accuracy improves for an increase in map size up to 64 after which it decreases. Although the final validation accuracy for both 48 and 64 is the same, 64 is selected as it had the least overfitting on training data.

The learning curve for the validation set is as shown.



D. *Kernel Size*

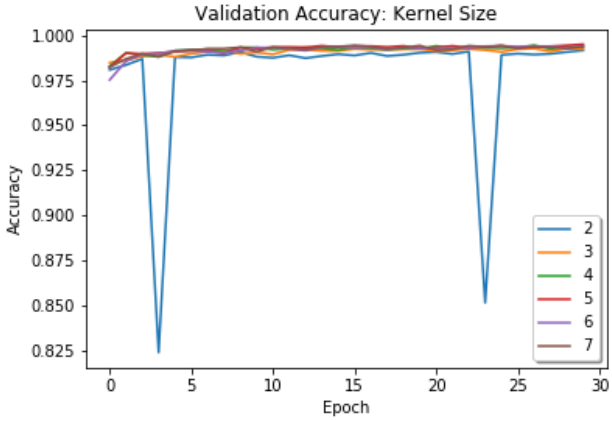
Then the kernel size in the convolutional layer was varied as shown in Table 5.

Table 5
Validation Accuracy for change in Kernel Size

Kernel Size	Validation accuracy
2	0.9916
3	0.99
4	0.9924
5	0.9927
6	0.9934
7	**0.9934**

Based on the above data, kernel size of 7 was selected. Also, it could be seen that with an increase in kernel size, the validation accuracy was increasing.

The learning curve for the validation set is as shown.



E. *Dropout*

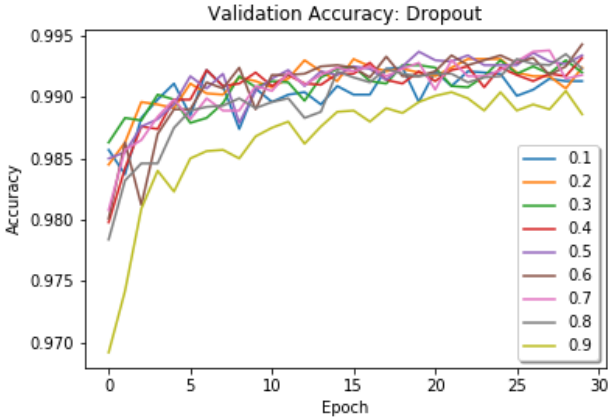
Then the dropout probability was varied as shown in Table 6.

Table 6
Validation Accuracy for change in Dropout

Dropout	Validation accuracy
0.1	0.9913
0.2	0.9924
0.3	0.992
0.4	0.9932
0.5	0.9934
0.6	**0.9943**
0.7	0.9918
0.8	0.9923
0.9	0.9886

As can be seen, the validation accuracy for Dropout 0.6 is highest and hence is selected. Also, it was seen that when the dropout was increased, the training accuracy decreased thereby preventing overfitting.

The learning curve for the validation set is as shown.



F. *Number of Convolutional Layers*

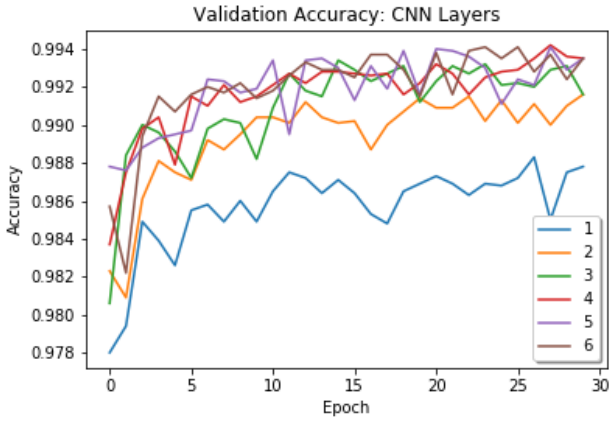
Then the number of convolutional layers were varied as shown in Table 4.

Table 7
Validation Accuracy for change in the number of convolutional layers

Number of Convolution layers	Validation accuracy
1	0.9878
2	0.9916
3	0.9916
4	**0.9935**
5	0.9935
6	0.9935

Based on this, 4 number of convolutional layers were chosen for the network. Although 4,5 and 6 have sale validation loss, 4 is selected as it leads to low complexity and time for training.

The learning curve for the validation set is as shown.



G. *Batch Size*

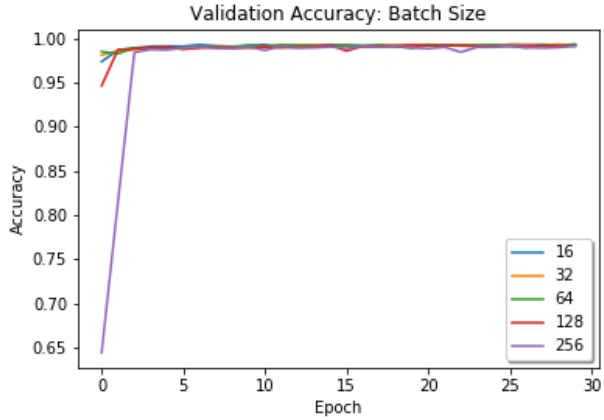
Then the batch size was varied as shown in Table 4.

Table 8
Validation Accuracy for change in Batch Size

Batch Size	Validation accuracy
16	0.9933
32	0.9918
64	**0.9930**
128	0.9915
256	0.9912

Based on this, although the batch size of 16 provides the highest validation accuracy, 64 was chosen due to computation constraints and it provides a similar result

The learning curve for the validation set is as shown.



It can also be observed that when the batch size increases, the starting accuracy is very low but after certain epochs it comes close to the accuracy with lower batch size.

IV. FINAL NETWORK STRUCTURE AND PARAMETERS

Based on hyperparameter tuning, the final network is as:

Table 9
Final Network Structure

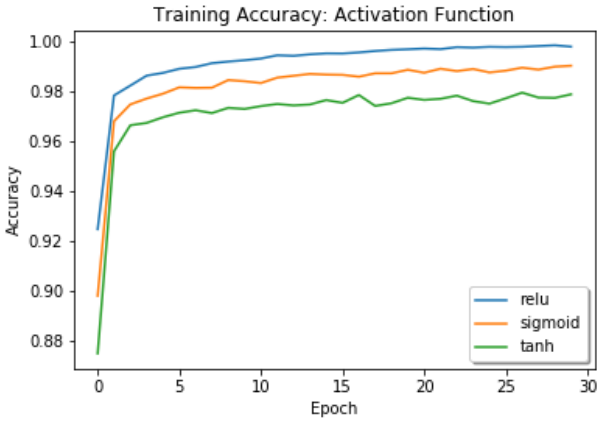
Layer type	Output Shape
Conv2D	(None,22,22,64)
Batch Normalization	(None,22,22,64)
Conv2D	(None,16,16,64)
Batch Normalization	(None,16,16,64)
Conv2D	(None,10,10,64)
Batch Normalization	(None,4,4,64)
Conv2D	(None,4,4,64)
Batch Normalization	(None,4,4,64)
Dropout	(None,4,4,64)
Flatten	(None,1024)
Dense	(None,128)
Batch Normalization	(None,128)
Dropout	(None,128)
Dense	(None,10)

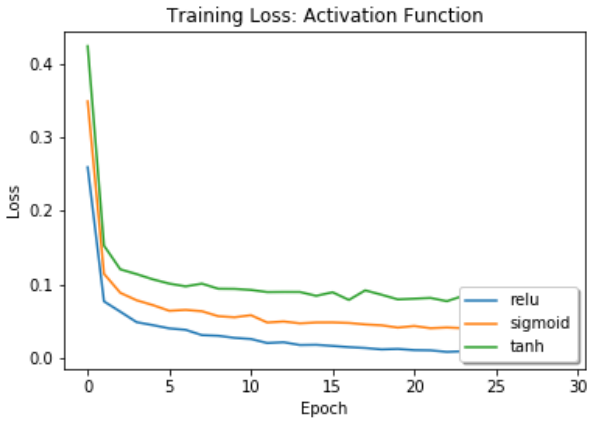
The final values of hyperparameters are:

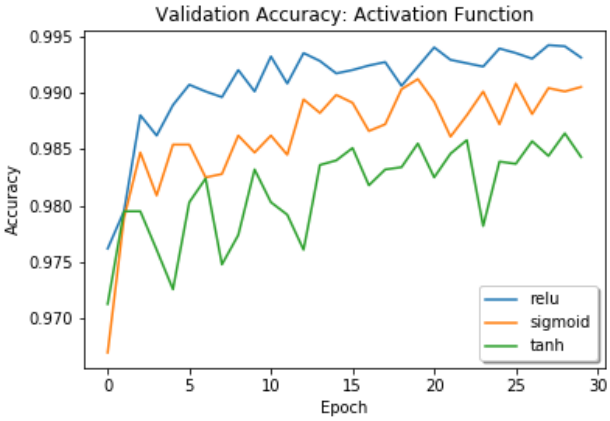
Table 10
Final Hyperparameter Values

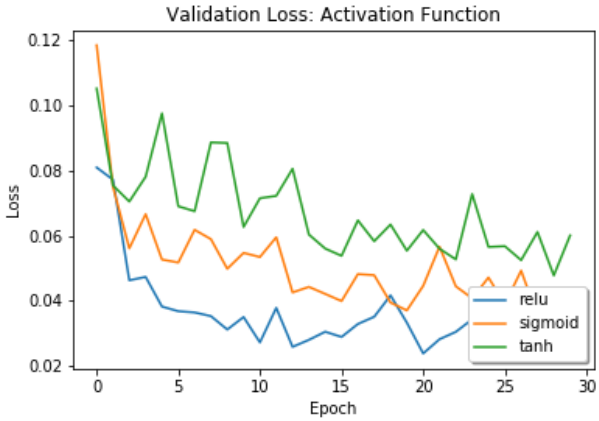
Hyperparameter	Default value
Neuros in Dense Layer	128
Learning Rate	0.001
Feature Map Size	64
Kernel Size	7
Number of Convolution Layers	4
Dropout Probability	0.6
Activation Function	Relu
Batch Size	64

The learning curves for different activation functions are:









Also, the final values of Validation accuracy and loss for different activations are in the table below. As can be seen, Relu performs the best with the least loss and highest validation accuracy, after that sigmoid and after that Tanh.

Table 11
Validation accuracy for change in Activation function

Activation Function	Validation accuracy	Validation Loss
Relu	0.9931	0.0345
Sigmoid	0.9907	0.0342
Tanh	0.9848	0.0546

The final model trained was tested on the test data and gave an accuracy of **99.36%** and the final loss was **0.0302**

V. CONCLUSION

In this homework, CNN was implemented using TensorFlow for MNIST classification. One-fold cross-validation was used by dividing the original training set (60000) to validation set (10000) and a new training set (50000). The google colab environment was used for training which made the training much faster. Hyperparameter tuning led to an understanding of various hyperparameters and their effects. Finally, the loss curves for training and validation data for different activation functions were plotted and showed the effectiveness of Relu function. Finally, the model was tested on the test data to give an accuracy of 99.36%.