Convolutional Neural Network Design Using MNIST Dataset

Daniel Mock
ECE 542
Neural Networks

***Abstract*—The purpose of this homework is to create a Convolutional Neural Network (CNN) based on the MNIST dataset. The MNIST dataset are a group of pictures of hand-written integers. This dataset is regularly used in Machine Learning to verify the accuracy of a neural network.**

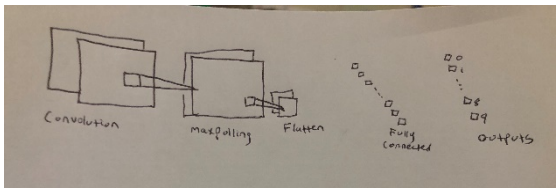
I. Introduction

This homework uses the MNIST dataset to train and test a convolutional neural network created using Tensorflow API. This API is a set of tools aimed at building, training, and testing machine learning (ML) models. In particular, this homework focuses on creation of a convolutional neural network (CNN). CNN's are great at classifying sets of data, which is the reason for using the MNIST dataset. The labels will be integers 0-9.

II. Design and Structure of Network

A basic CNN was designed using Tensorflow. There was one convolutional layer used in this design. The model is built by creating a convolution layer comprised of a window size of 3x3, and generating filters from the convolutions. The outputs of the convolution layer are applied with a dropout. Next, the outputs are flattened, and sent through an activation function. The hyperparameters used in this design include: dropout, batch size, and learning rate.

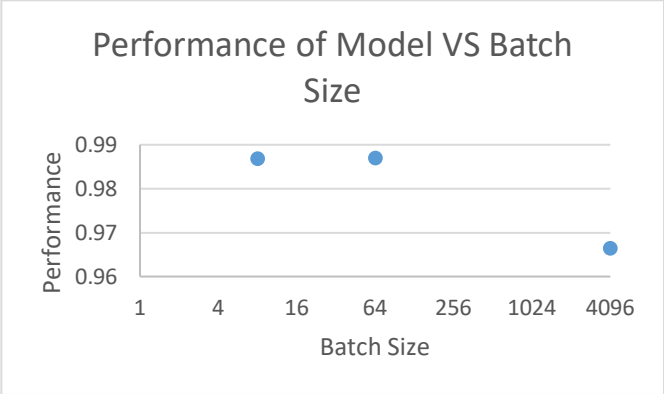
The following image clearly shows the structure of the CNN used for classification of the MNIST dataset:

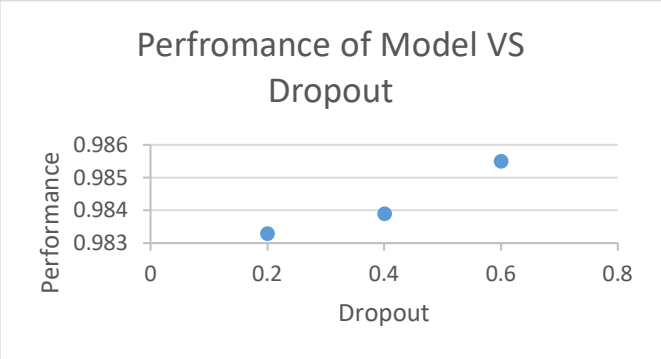


Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
batch_normalization (BatchNo	(None, 28, 28, 32)	128
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 256)	1605888
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570

III. Hyperparameter Performance

The performance of dropout and batch size were measured to determine the best value to use for each. For dropout, .2, .4, & .6 were tested, and for batch size, 8, 64, & 4096 were tested. The following graphs represent these tests:

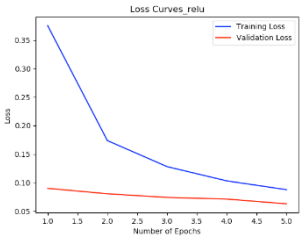


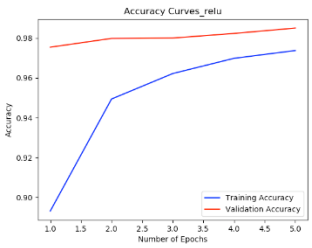


As you can see in the images above, the best performance relative to the batch size occurred when a batch size of 64 was used. Also, a dropout of .6 yielded the highest performance.

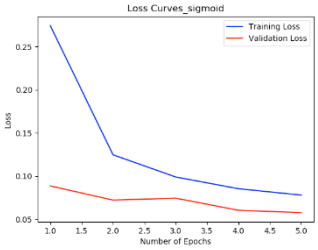
IV. Learning Curve

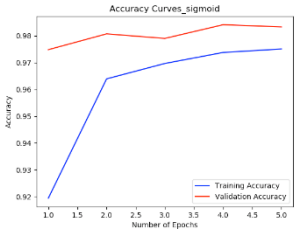
After the optimal hyperparameters were selected, the testing loss and testing accuracy were measured and plotted after each epoch using different activation functions. The activation functions used are sigmoid, relu, and softmax. The graphs are illustrated below:



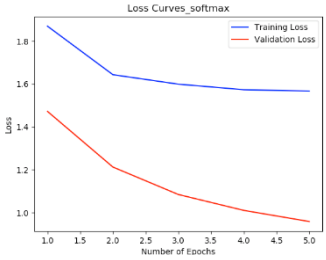


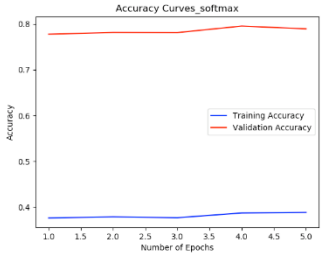
In the above graphs, the Relu activation function was used when collecting the testing accuracy and testing loss. The final testing loss recorded was .0486 and the final testing accuracy recorded was .9861.





In the above graphs, the Sigmoid activation function was used when collecting the testing accuracy and testing loss. The final testing loss recorded was .0547 and the final testing accuracy recorded was .9827.





In the above graphs, the Softmax activation function was used when collecting the testing accuracy and testing loss. The final testing loss recorded was .9626 and the final testing accuracy recorded was .7864.

The softmax activation function gave the worst accuracy out of the three. I believe this is due to only using 5 epochs in the training. If more epochs were used, I believe the softmax would be more comparable to the other models.

V. Analysis

Out of all the different hyperparmeters and activation functions used, I believe the best set up is to have a batch size of 64, a dropout of .6, and a relu activation function. This gives a testing accuracy of .9861, and a validation accuracy of .9869.

VI. Summary

Overall, this was a great assignment to understand how CNN works and to get hands on experience with Tensorflow. Tensorflow is a great tool to easily create neural networks using their API. It was nice to see the changes in performance as I changed the hyperparameters and the activation functions used.