NORTHWESTERN UNIVERSITY | SCHOOL OF PROFESSIONAL STUDIES

MNIST Dataset

Ranaa Ansari, Daniel Arenson, Yining Feng, Han Nguyen

Northwestern University

MSDS-422 Assignment 6: Neural Networks

Summary and problem definition for management

Assignment 6 focuses on applying neural network learning methods to a multi-classification of handwritten digits in the MNIST dataset. A variety of Deep Neural Network (DNN) models were built to assign a digit that is equal to the handwritten one; as there were slight differences among handwritten digits, the challenge was for the models to accurately distinguish between each digit within a reasonable runtime. Ultimately, the end goal was to test different combinations of hidden layers and nodes per layer to benchmark performance, and provide a recommendation on which neural network typology and hyperparameter settings are most reliable.

Research Design

The steps for the analysis are shown as below:

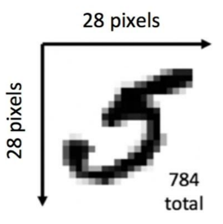
(1) Build a Deep Neural Network model using TensorFlow

(2) Train and test various designs of the model using ‘hidden layers’

(3) Time the model build and record the train/test accuracy

(4) Compare the model timing, precision and recall scores

The data set for this assignment is the same as the previous one. The MNIST data set consists of 70,000 handwritten digits, and is a popular example of training machine learning models. Each row represents one hand written example using 785 variables. The first column is the ‘response’ variable, which is the actual value to test the predicted estimate against. The remaining 784 columns are the integer grey scale values of each pixel in a 28 x 28 pixel square. For example, the plot on the left is a binary plot showing a row of data that has a y value of ‘5’. The first column of data is the actual value – for training & testing. The subsequent 784 columns of data are the grayscale values for each of the 28x28 pixels representing the digit.



An example of a plotted row of data (784 pixels)

The next step in the process is to split the data by training & testing data sets. According to the instructions, we are told to use 60,000 rows for training, and 10,000 for testing. We then shuffle the training data set by the index to randomize the data. Subsequently, we begin the experiment by creating multiple deep neural networks using TensorFlow. For model evaluation, we compare performance metrics including each model’s runtime and accuracy scores of training & testing data sets.

Programming Work

The programming dataset MINST was retrieved using *tf.keras.datasets*. The MNIST database contains 60,000 training images and 10,000 testing images. The entire dataset was used to create a multi classifier. Eight models, each of which comprised different combinations of hidden layers and nodes per layer were created using Tensorflow.

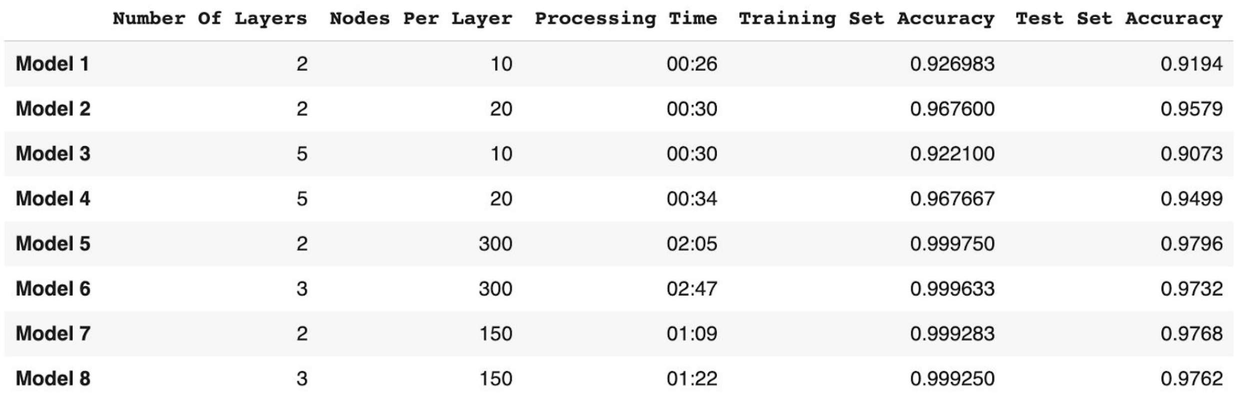
Each model that was developed first had the data trained to convert the features into continuous variables, representing a real value with numeric features. This was programmed using *tf.feature_column.number_column.* Then the programmed featured columns were specified with a certain amount of hidden layers and number of classes for the output layer. Ultimately, building a feedforward multilayer neural network that would train the dataset on the labeled data and perform classification on similar unlabeled data. This was applied with *tf.estimator.DNNClassifier.*

Since the *MNIST* dataset was programmed earlier in numpy format, we used *tf.estimator.inputs.numpy_input_fn* to pass in X as our dictionary and Y as our label. To monitor each model with varying layers, but maintaining similar hyperparameters, we kept the number of epochs, batch size, shuffle=True to be uniform.

Each train and test model was then further evaluated with an accuracy score (applied with *.evaluate*) and processing time.

Results and Recommendation

A total of 8 experiments were conducted to test neural network structures. 2 to 5 layers were tested along with up to 300 nodes per layer for some models. Models 1 to 4 had lower processing time when compared to models 5 to 8 because it had fewer nodes per layer in the experiments. Models 5 to 8 had a higher test set accuracy as a result of more nodes per layer in the experiment. Looking at the dataset, one can assume that model 7 is the winner with the highest test set accuracy score. However, model 2 had the lowest processing time. There is not a single winner for this experiment; instead, it shows two models that can be recommended based on the scenario implementation.



	Number Of Layers	Nodes Per Layer	Processing Time	Training Set Accuracy	Test Set Accuracy
Model 1	2	10	00:26	0.926983	0.9194
Model 2	2	20	00:30	0.967600	0.9579
Model 3	5	10	00:30	0.922100	0.9073
Model 4	5	20	00:34	0.967667	0.9499
Model 5	2	300	02:05	0.999750	0.9796
Model 6	3	300	02:47	0.999633	0.9732
Model 7	2	150	01:09	0.999283	0.9768
Model 8	3	150	01:22	0.999250	0.9762

Reference

Heitman, E. (2019, June 8). Beginner's Guide to Building Neural Networks in TensorFlow. Retrieved May 16, 2020, from https://towardsdatascience.com/beginners-guide-to -building-neural-networks-in-tensorflow-dab7a09b941d