The MNIST (Modified National Institute of Standards and Technology) dataset contains 70,000 small images of handwritten digits. Each digit image is labeled with the digit it represents. The dataset is used for machine learning classification algorithms, and this assignment will evaluate the classification of the digits and the impact of Principal Component Analysis (PCA) on the data.

The dataset was downloaded, and the data variables and target were saved to X and y objects, respectively. The X object contains 70,000 handwritten digits with 784 dimensions or features (28 x 28 images). The y object is the true value of the handwritten digit. The data was split so there were 60,000 observations in the training set and 10,000 observations in the test set. The X data was processed to convert the data type, reshape the data, and normalize it by dividing by 255. The y data was processed to convert the data type.

The MNIST data sets were examined in TensorFlow via three different ways. First, the data was classified via plain TensorFlow in a 2x2 completely crossed design benchmark experiment. This was a simple comparison of neural networks which examined the accuracy between two and five hidden layers. For each layer the number of nodes was evaluated at 10 and 20 nodes each. The processing time and accuracy for both the training and test data sets were recorded and compared. Then a TensorFlow premade estimator called a DNN Classifier was used to create another model. This is a classifier for TensorFlow deep neural network (DNN) models. This model contained two hidden layers with 128 and 32 nodes, respectively. Finally, Keras was used in TensorFlow to create two models. One model contained two densely-connected neural network hidden layers with 128 and 32 nodes, respectively. The second model contained two densely-connected neural network hidden layers with 20 nodes each. In both instances, the first hidden layer was followed by a rectifying linear unit (relu) activation function, and the second hidden layer was followed by a softmax activation function. The first Keras model's accuracy and loss were plotted in matplotlib. The second Keras model used a confusion matrix to compare and visualize the true values against the predicted values, and the model’s accuracy and loss were visualized utilizing TensorBoard.

In the benchmark experiment, the two models with five layers had better training accuracy compared to test accuracy, which indicated that the models were slightly overfit to the training data. The models with five layers also took longer to train and evaluate compared to the models with only two layers. The models with two layers had similar test accuracy, but the one with 20 nodes versus 10 nodes took a few seconds longer to finish.

The DNN Classifier model achieved a test accuracy of 97.5%. This model used two layers with 128 nodes and 32 nodes, respectively. The additional nodes compared to the models in the benchmark studies boosted the accuracy, but this resulted in a computational time nearly three times longer (142 seconds).

Using a Keras model of two layers with 128 and 32 nodes resulted in similar accuracy (97.7%) but a longer computational time (256 seconds) compared to the DNN Classifer. The Keras model with two layers and 20 nodes each provided an accuracy of 96.1% and a computational time of 130 seconds. The confusion matrix demonstrated where the most frequent errors occurred, such as mistaking 9s for 4s or 7s for 2s.

From the benchmark study, it can be concluded that additional layers, which result in longer computational times, does not result in higher test accuracy scores. It is faster and just as accurate to use two layers with only 20 nodes. The models with two layers and 128 and 32 nodes achieved higher accuracy scores compared to those achieved in the benchmark study but had longer computational times. The decision on the number of layers and nodes will depend on whether computational time or accuracy is more important for the problem at hand.