Deep Learning: Homework #1

In this homework, we will use Google's Tensorflow (TF) to create deep neural networks (DNNs), in order to classify handwritten digits from the MNIST dataset.

Note: we use a low-level TF 1.x coding style for this homework, defining all learning parameters. Do not use other TF functions which create layers, for example, tf.layers functions.

Task #1 (Code Completion)

In this task, you need to understand the provided python source code (dnn_mnist.py). In particular, there are some missing parts of the code marked with "???", for example:

```
'h2': tf.Variable(tf.random_normal([???, ???])),
```

Fill-in these missing parts of the code, so that it will perform 10-classes multi-class classification on the MNIST digits.

Task #2 (A function to create DNNs with different numbers of hidden layers)

In the source code (dnn_mnist.py), there is a function: multilayer_perceptron(x, weights, biases). The current version of the function creates a DNN with two hidden layers. Modify this function, so that the function can create a DNN with an arbitrary number of hidden layers. You control the desired number of hidden layers by the number of elements in the lists, *weights* and *biases*.

Task #3 (Test accuracy vs. DNN architecture)

The last line of the code reports classification accuracy of the created DNN with regard to the test dataset. Report and compare the test accuracy values of the following DNNs with different architecture:

1) DNN with two hidden layers, each layer has 256 units (as in the given code)
2) DNN with three hidden layers, each layer has 128 units
3) DNN with four hidden layers, each layer has 64 units

Make a short discussion about the results, considering i) test accuracy, ii) number of hidden layers, and iii) the total number of learning parameters.

Note that we should use a validation set, not a test set, for model selection. However in this homework, for simplicity, we use the provided test set for finding the best architecture.

Task #4 (Tuning hyper-parameters)

There are many other nobs we can tune, such as learning_rate, n_epochs, batch_size, in addition to the number of hidden layers and the number of units in hidden layers. Try several settings of your own choices and report the best architecture with the highest test accuracy. Also report the best test accuracy value.

Submit a report, containing the following:

1. Task #1: completed python code, printed out (font size 9)
2. Task #2: the function definition code, printed out (font size 9)
3. Task #3:
    a. test accuracy values for the three cases
    b. a short discussion (see instruction above)
4. Task #4: hyper-parameters of the best architecture
    a. no. of hidden layers, no. of units in each layer
    b. learning_rate, n_epoch, & batch_size
    c. best test accuracy value

Checklist:

1. Don’t forget to write down your name and student ID
2. The report MUST be written in English.
3. Use word processor. Hand-written reports will NOT be accepted.
4. Discussion with other students are encouraged, but you MUST make your own answers. Violators will get strong penalties.