Homework 5 – Deep Neural Networks (CS/DS 541, Whitehill, Spring 2021)

You may complete this homework either individually or in teams up to 3 people.

1 Implementing a Neural Network with TensorFlow [15 points]

In this problem you will train a neural network to classify images from the Fashion MNIST dataset. Instead of implementing forward- and backward-propagation yourself using numeric Python, you will rely on the highly optimized TensorFlow package, which implements symbolic differentiation and other nifty features. Your tasks are to do the following:

1. Either install TensorFlow on your own machine (see `https://www.tensorflow.org/install/` for instructions), or use Google Colab.

2. Follow the instructions in the following Medium blogpost (by Margaret Maynard-Reid): `https://medium.com/tensorflow/hello-deep-learning-fashion-mnist-with-keras-50fcff8cd74a` to train a convolutional neural network (CNN). Train a neural network to achieve a **test** accuracy of at least 90.0%. Feel free to vary the number of convolutional and pooling layers as well as their associated hyperparameters (padding, stride, etc.). Create a screenshot showing at least 5 gradient updates on the training set, along with a screenshot of your final accuracy on the test set.

2 Representing a Convolutional Neural Network as a Fully-Connected Neural Network [30 points]

1. Implement **exactly** the following architecture using TensorFlow:

   - Convolution layer with 64 filters, each 3x3, stride of 1 (i.e., apply the filter at all pixel locations), no padding.
   - Max pool with a pooling width of 2x2, stride of 2, no padding.
   - ReLU.
   - Flatten the 64 feature maps into one long vector.
   - Fully-connected layer to map into a 1024-dimensional vector.
   - ReLU.
   - Fully-connected layer to map into a 10-dimensional vector.
   - Softmax.

   Then train the network for a few epochs (the exact accuracy doesn't matter for this problem).

2. Extract the weights from the model after training it. See `model.summary()` and `model.trainable_variables`, where `model` is the TensorFlow model you trained.

3. Expand on your homework 4 submission (just the forward-propagation part, which is straightforward even if your homework 4 submission was incorrect) to implement max-pooling. (It's fine to use a for-loop to iterate over all the different 2x2 regions for the pooling step.) You should **not** write or use a dedicated convolution function on this step. Moreover, you do **not** need to compute the gradients for this network.

4. Convert the convolution filter weights that you extracted from the TensorFlow-based network into the weights of a **fully-connected layer** (i.e., standard matrix multiplication, like we did in two exercises during class). **Make sure to include the code you wrote to convert the weights**

from TensorFlow's model to your homework 4-style network. One thing you will have to do is to figure out the number of neurons in each hidden layer, based on the padding, stride, # filters, etc., of the previous pooling or convolution step. (`model.summary()` will help with this.)

5. Pick an arbitrary Fashion MNIST image, classify it, and show that you get the exact same predictive distribution (output of softmax) using TensorFlow as you do using your own fully-connected neural network you built from scratch. Include a screenshot in your PDF to verify this.

In addition to your Python code (`homework5_WPIUSERNAME1.py` or `homework5_WPIUSERNAME1_WPIUSERNAME2.py` for teams), create a PDF file (`homework5_WPIUSERNAME1.pdf` or `homework5_WPIUSERNAME1_WPIUSERNAME2.pdf` for teams) containing the screenshots described above. **Please submit both the PDF and Python files in a single Zip file.**