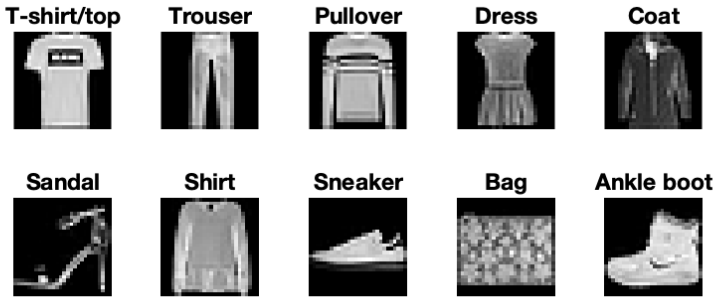
AMATH 482 Winter Quarter 2020

Homework 5: Neural Networks for Classifying Fashion MNIST

DUE: Friday, March 13, 2020

In class, we have built (or will build) a classifier for the popular MNIST dataset of handwritten digits. In this assignment, you will work with an analogous data set called Fashion-MNIST. Instead of handwritten digits, there are images of 10 different classes of fashion items. One image of each type is shown below.



You can load the data using the following commands:

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```

The Fashion-MNIST data has exactly the same structure as the MNIST data. There are 60,000 training images in the array `X_train_full` and 10,000 test images in the array `X_test`, each of which is 28×28. The labels are contained in the vectors `y_train_full` and `y_test`. The values in these vectors are numbers from 0 to 9, but they correspond to the 10 classes in the following way:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Before you begin, preprocess the data in the following way.

1. Remove 5,000 images from your training data to use as validation data. So you should end up with 55,000 training examples in an array `X_train` and 5,000 validation examples in an array `X_valid`.

2. The numbers in the arrays `X_train`, `X_valid`, and `X_test` are integers from 0 to 255. Convert them to floating point numbers between 0 and 1 by dividing each by 255.0.

Part I

Train a **fully-connected** neural network to classify the images in the Fashion-MNIST data set. You should try several different neural network architectures with different hyperparameters to try to achieve the best accuracy you can on the validation data. Some things you might try adjusting are:

(a) The depth of the network (i.e. the number of layers)

(b) The width of the layers

(c) The learning rate

(d) The regularization parameters

(e) The activation functions

(f) The optimizer

(g) Anything else you can think of or find online

Can you beat 90% accuracy on the validation data? What about 95%? Once you have experimented and found a network architecture and hyperparameter values that you think perform well, use that architecture and those hyperparameter values to train one final model. Check the accuracy of your final model on the test data and report your results. You should also include the confusion matrix for the test data. You should include in your report what things you tried while experimenting, but you only need to include plots or results for the final model.

Part II

Repeat the same procedure as Part I, but now use a **convolutional** neural network. In addition to the things listed in the previous part, you can now try adjusting:

(a) The number of filters for your convolutional layers

(b) The kernel sizes

(c) The strides

(d) The padding options

(e) The pool sizes for your pooling layers

Report your results and compare them to your fully-connected network from Part I.