SIMON D. LEVY

Computer Science Department, Washington and Lee University

CSCI 315 Assignment #5

Due on Github 11:59PM Friday 24 March

Goal

The goal of this assignment is to get some practice working with convolutional neural networks (CNNs) in PyTorch. As before, we will begin by modifying existing code from the author, instead of attempting to write the whole program from scratch.

Get it working

As in the previous assignment, look over the code from the author's repository. Then begin copy/pasting piece-by-piece into a single Python script **cnn.py** that you can run in IDLE, command-line, or your favorite IDE. It should only take a few minutes to copy/paste the code up to the point where you can run the first network and get a final accuracy of around 55.6. Looking at my code I that point, I found there was a bit that was unused, so I removed it.

Train, pickle

Once you can run the code, repeat what we did in the previous assignment: create two separate scripts **cnn.py** (class definition) and **cnn_train.py** (training). This time you'll have to to add the pickling code, since it's not already in the code from the author's repository. While you're at it, copy the line of code from the previous assignment that reports which device (cpu or gpu) the program is using to train the network.

Test

Considering all the work we did getting **mlp_test.py** to work in the last assignment (including the confusion matrix), I found it easier to use that script as the basis for my third new script, **cnn_test.py**. Although the initial copy/paste/modify was easy, I immediately ran into some problems with the `forward()` method. (If you didn't have problems, feel to skip the rest of this paragraph!) Following my practice of *when in doubt, print out it*, I discovered the usual suspect I've mentioned in class: a mismatch in the shape of the tensor I was passing into the network versus what the network expected. I thought about doing a `reshape()` call on the data tensor, but quickly realized that a simpler solution involved eliminating, rather than adding, a line of code.

CPU/GPU Rematch!

Use your **cnn_train.py** script to repeat the timing experiment from the previous assignment and report your results (time for training on CPU vs. GPU for same number of epochs). In addition to noticing a significant time difference (expected), I found that the GPU vs. CPU agreed on the loss, but differed noticeably in accuracy . This puzzled me a bit, because the point of calling `torch.random.seed(0)` at the top is to get the same results every time, to help with debugging. In your writeup, note the CPU-vs-GPU time differences, and see what you can find online about the differences in accuracy.

But is it worth it?

Of course, a fancy neural net (CNN) with GPU speedup isn't really worth much if it doesn't beat the results (around 91% test accuracy) that we were able to get with the simpler networks (SLP, MLP) in our previous assignment. To test the advantage of the CNN, let's step up our game to a much bigger number of training epochs, say, 500. So to finish up, see what kind of accuracy you can get with your three networks (SLP, MLP, CNN) after such a large number of training epochs — making sure to use the same learning rate for all three. (I recommend bringing along some other work to do while you're waiting!) Report the accuracy for each of the three networks in your writeup.

What to turn in to Github

As before, your main turnin will be a PDF writeup of your results, but be sure to upload your three CNN scripts to for possible future use.