Homework 3
APPM 4720/5720 Scientific Machine Learning, Fall 2024

Due date: Friday, Sep. 13 '24, before midnight, via Gradescope

Instructor: Prof. Becker
Revision date: 9/10/2024

Theme: Training neural nets

Instructions Collaboration with your fellow students is OK and in fact recommended, although direct copying is not allowed. The internet **is allowed for basic tasks** (e.g., looking up definitions on wikipedia, looking at documentation, looking at basic tutorials) but it is not permissible to search for solutions to the exact problem or to *post* requests for help on forums such as `http://math.stackexchange.com/`.

Problem 1: Create and train a neural net on the 50k training data in CIFAR10, as you did last week, but with a few new twists described below:

a) Split the data into training/validation/test. You can choose the split yourself, but describe in your homework $\boxed{\text{why you chose it and how you implemented it}}$.

b) In addition to the neural net architecture you used last week, create a variant of this architecture (at your discretion), e.g., take away layers ("ablation") or add layers, and/or change sizes, and/or change activations. $\boxed{\text{Turn in a PDF of your source code}}$, **nicely formatted** (no screenshots!).

c) For both the old and new architectures, plot the training and validation errors for every iteration. $\boxed{\text{Turn in the plot}}$ and $\boxed{\text{comment on whether you think your training has converged}}$, and $\boxed{\text{if you are over- or under-fitting}}$.

d) $\boxed{\text{Report the accuracy of both models on the testing set}}$, and comment on $\boxed{\text{if this was expected}}$ given performance on training and validation.

e) Show some basic $\boxed{\text{profiler}}$ output from running a profiler on your code (on either architecture, not both), which should show which parts of your code are the slowest to run.

Suggestions You are welcome to do this assignment in any language (Python or Julia suggested) and any framework (PyTorch, Tensorflow, Jax, Lux, etc.). However, I suggest (and class demos and homework solutions will reflect this) that you use PyTorch Lightning. This is a higher-level interface to PyTorch, and it's not always the right tool, but it does have some nice benefits:

1. Recording ("**logging**") error metrics is standardized, and you can also easily **plot** the results via TensorBoard. See lightning.ai: Track and Visualize Experiments (Start with the "Basic" demo which is probably sufficient for this homework; it also describes how to use TensorBoard with Colab and Jupyter)
   - This goes beyond just making plots. Not only does it make plotting nice (easy to overlay many different runs that used different parameters) and have handy features like smoothing, but it also saves the log data to files. These log files are a bit like the equivalent of a scientist's lab notebook.

- See Cooper's PyTorch Lightning project template for an example of how to use 'json' experiment files. This is a nice way to record the hyperparameters used for a run, and works well on the CURC cluster.

2. It also supports **profiling**. See lightning.ai: Find bottlenecks in your code (again, the "Basic" demo is sufficient for now)

3. It is also designed to be portable and high-level, so that you can just flip a single switch to train on the GPU instead of CPU.