Introduction to High Performance Machine Learning
ECE-GY 9143

Lab 2

This lab is intended to be performed individually, great care will be taken in verifying that students are authors of their own submissions.

Exercises need to be executed on the NYU HPC Greene cluster in the standard compute nodes. Theoretical questions are identified by **Q<number>** while coding exercises are identified by **C<number>**.

Grading rules

• Total is 50 points

In this part of the lab, we would create a Neural Network using PyTorch to classify a dataset of images. The dataset we are going to use is CIFAR10, which contains 50K 32x32 color images. The model we are going to build is ResNet-18, as described in https://arxiv.org/abs/1512.03385. The reference code is at https://github.com/kuangliu/pytorch-cifar.

Create a PyTorch program with a DataLoader that loads the images and their labels from the torchvision CIFAR10 dataset. Import CIFAR10 dataset for the torchvision package, with the following sequence of transformations:

1. Random cropping, with size 32x32 and padding 4
2. Random horizontal flipping with a probability of 0.5
3. Normalize each image's RGB channel with mean(0.4914, 0.4822, 0.4465) and variance (0.2023, 0.1994, 0.2010)

The DataLoader for the training set uses a minibatch size of 128 and 3 IO processes (i.e., num_workers=2).
The DataLoader for the testing set uses a minibatch size of 100 and 3 IO processes (i.e., num_workers =2).

Create a ResNet-18 model as defined in https://arxiv.org/abs/1512.03385. Specifically, The first convolutional layer should have 3 input channels, 64 output channels, 3x3 kernel, with stride=1 and padding=1. Followed by 8 basic blocks in 4 subgroups (i.e. 2 basic blocks in each subgroup):

1. The first sub-group contains a convolutional layer with 64 output channels, 3x3 kernel, stride=1, padding=1.

2. The second sub-group contains a convolutional layer with 128 output channels, 3x3 kernel, stride=2, padding=1.
3. The third sub-group contains a convolutional layer with 256 output channels, 3x3 kernel, stride=2, padding=1.
4. The fourth sub-group contains a convolutional layer with 512 output channels, 3x3 kernel, stride=2, padding=1.
5. The final linear layer is of 10 output classes.

For all convolutional layers, use RELU activation functions, and use batch normal layers to avoid covariant shift. Since batch-norm layers regularize the training, set the bias to 0 for all the convolutional layers. Use SGD optimizers with 0.1 as the learning rate, momentum 0.9, weight decay 5e-4. The loss function is cross-entropy.

C1 Training in PyTorch [10 points]

Create a main function that creates the DataLoaders for the training set and the neural network, then runs 5 epochs with a complete training phase on all the mini-batches of the training set. Write the code as device-agnostic, use the ArgumentParser to be able to read parameters from input, such as the use of Cuda, the data_path, the number of data loader workers, and the optimizer (as string, eg: ‘sgd’).

For each minibatch calculate the training loss value, the top-1 training accuracy of the predictions, measured on training data. Note: (i) No need to write your own ResNet module, you can re-use as much code as you want from the above-mentioned GitHub (ii) Typically, we would like to examine test accuracy as well, however, it is sufficient to just measure training loss and training accuracy for this assignment. (iii) You don’t need to submit any outputs for C1. You’ll only need to submit the relevant code for this question. C2-C7 will be based on the code you wrote for C1.

C2: Time measurement of code in C1 [5 points]

Report the running time (by using time.perf_counter() or other timers you find comfortable with) for the following sections of the code:

1. Data-loading time for each epoch
2. Training (i.e., mini-batch calculation) time for each epoch
3. Total running time for each epoch

Run 5 epochs.

C3: I/O optimization starting from code in C2 [5 points]

1. Report the total time spent for the Dataloader varying the number of workers starting from zero and increment the number of workers by 4 (0,4,8,12,16...) until the I/O time doesn’t decrease anymore.
2. Report how many workers are needed for best runtime performance.

C4: Profiling starting from code in C3 [5 points]

Compare data-loading and computing time for runs using 1 worker and the number of workers needed for best performance found in C3 and explain (in a few words) the differences if there are any.

C5: Training in GPUs vs CPUs [5 points]

Report the average running time over 5 epochs using the GPU vs using the CPU (using the number of I/O workers found in C3.2)

C6: Experimenting with different optimizers [5 points]

Run 5 epochs with the GPU-enabled code and the optimal number of I/O workers. For each epoch, report the average training time, training loss, and top-1 training accuracy using these Optimizers: SGD, SGD with Nesterov, Adagrad, Adadelta, and Adam. Note please use the same default hyper-parameters: learning rate 0.1, weight decay 5e-4, and momentum 0.9 (when it applies) for all these optimizers.

C7: Experimenting without Batch Norm layer [5 points]

With the GPU-enabled code and the optimal number of workers, report the average training loss, top-1 training accuracy for 5 epochs with the default SGD optimizer and its hyper-parameters but without batch norm layers.

Q1:

How many convolutional layers are in the ResNet-18 model? [2 points]

Q2:

What is the input dimension of the last linear layer? [2 points]

Q3:

How many trainable parameters and how many gradients are in the ResNet-18 model that you build (please show both the answer and the code that you use to count them) when using the SGD optimizer? [4 points]

Q4:

The same question as Q3, except now use Adam (only the answer is required, not the code). [2 points]

Appendix – Submission instructions

Submission through Brightspace.
Please submit a zip with file name <your-netID>.zip with a folder named as your netID (example: am9031/) containing the following files:
• A file named lab2.py containing all the exercises (please also include the model file if you use separate python files). Insert in this file the code used for all the exercises, commenting all the lines needed for the exercises C2-C7 and Q3.
• A file named lab2.pdf with a report of the outputs requested in C2-C7 and Q1-Q4.
• Failing to follow the right directory/filename specification is -1 point. Failing to have programs executed in the sequence is -1 point.

Appendix – How to Run Experiments

All jobs that do not require a GPU need to be executed on the CPU-only nodes. There are only a few GPU nodes.