COMS 6998 - High Performance Machine Learning
Homework Assignment 2
Parijat Dube and Kaoutar El Maghraoui

Due Date: Feb 26, 2023   Spring 2023   Max Points: 50

Instructions:

This lab is intended to be performed **individually**, great care will be taken in verifying that students are authors of their own submission and that the exam is different from previous courses.

Theoretical questions are identified by Q<number> while coding exercises are identified by C<number>.

In this part of the lab we create a Neural Network in PyTorch to classify a dataset of images. The dataset we are going to use is CIFAR10, which contains 50K 32×32 color images. The model we are going to build is ResNet-18, as described in **Deep Residual Learning for Image Recognition**. The refence code is at **pytorch-cifar**.

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

1. Random cropping, with size 32×32 and padding 4

2. Random horizontal flipping with a probability 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 minibatch size of 100 and 3 IO processes (i.e., *num_workers=2*).

Create a ResNet-18 model as defined in **Deep Residual Learning for Image Recognition**.

Specifically, The first convolutional layer should have 3 input channels, 64 output channels, 3×3 kernel, with *stride=1* and *padding=1*.

Followed by 8 basic blocks in 4 sub groups (i.e. 2 basic blocks in each subgroup):

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

- The second sub-group contains convolutional layer with 128 output channels, 3×3 kernel, *stride=2, padding=1*.

- The third sub-group contains convolutional layer with 256 output channels, 3×3 kernel, *stride=2, padding=1*.

- The forth sub-group contains convolutional layer with 512 output channels, 3×3 kernel, *stride=2, padding=1*.

- 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 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 run 5 epochs with a complete training phase on all the minibatches 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 dataloader 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:
(C2.1) Data-loading time for each epoch
(C2.2) Training (i.e., mini-batch calculation) time for each epoch
(C2.3) Total running time for each epoch Run 5 epochs.

C3: I/O optimization for code in C2   5 points

(C3.1) 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.
(C3.2) 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 is 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 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 2 points

How many convolutional layers are in the ResNet-18 model?

Q2 2 points

What is the input dimension of the last linear layer?

Q3 4 points

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

Q4 2 points

Same question as Q3, except now using Adam (only the answer is required, not the code).

Appendix - Submission Instructions

Please submit a *.targz* archive 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 exercise, commenting all the lines needed for the exercise 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/file name specification is -1 point. Failing to have programs executed in 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**.