CS 4803/8803 Efficient ML: Homework Project 0

Getting Started with PyTorch

Goals of this lab:

1. Learn the basics of PyTorch for implementing deep neural networks (DNNs)
2. Become familiar with hyperparameter tuning for DNN training
3. Understand efficient DNN designs and learn how to quantify DNN efficiency

Submission deadline on Canvas: Jan. 29, 2024

[20%] Task 1: Set up the coding environment and try an official Pytorch example

- *Step 1*: Learn the basics of Pytorch
  - Material: PyTorch Fundamentals from Microsoft (you can also learn from other sources)
- *Step 2*: Set up the Conda environment, used for managing Python packages required by your code, by following the official guideline
- *Step 3*: Execute an official PyTorch example that trains a small DNN for MNIST classification

- *Deliverables*: Submit your code and report the initial accuracy on MNIST you achieve

[50%] Task 2: Make your classifier more accurate

- *Step 1*: Update your DNN to a ResNet-18 (i.e., an 18-layer ResNet, hundreds of ResNet implementations are available online)
- *Step 2*: Tune the learning schedule and optimizer choices
  - Please survey the commonly used learning rate schedules and optimizers by yourself
- *Step 3*: Change your dataset to CIFAR-10 (which can be automatically downloaded & preprocessed by PyTorch) and re-tune the aforementioned hyperparameters

- *Deliverables*: Your code and the answers to the following questions
  - Q1: On MNIST, which impacts your final accuracy most: changes in your networks, learning rate schedules, or optimizers?
  - Q2: On CIFAR-10, what observations do you have when tuning the learning rate schedules and optimizers? Please detail your experience regarding hyperparameter tuning as much as possible. For instance,

compared to MNIST, are the required learning rates lower or higher? Will the choice of optimizers make a larger difference?

[30%] Task 3: Make your classifier more efficient

- *Step 1*: Replace the building blocks of ResNet-18 with more efficient blocks composed of depthwise convolutions from MobileNet
  - Hundreds of MobileNet implementations are available online
  - You have the freedom to modify the network based on your expertise as long as you can improve its accuracy-efficiency trade-offs
- *Step 2*: Train your modified networks on CIFAR-10
  - Feel free to tune the hyperparameters, especially the learning rates, similar to what you did in **Task 2**
- *Step 3*: Quantify the efficiency of ResNet-18 and your modified network using both theoretical metrics and latency measured on real devices
  - Theoretical metrics: Number of parameters / Floating-point operations (FLOPs)
  - Real-device measurement: Measure the latency of ResNet-18 and your modified network on the device where you execute your networks
    - Reference: https://deci.ai/blog/measure-inference-time-deep-neural-networks/
    - It would be great if you could measure the latency on common GPUs like RTX 2080Ti

- *Deliverables*: Your code and the answers to the following questions
  - Q1: How do you determine the strategy for replacing the building blocks in ResNet-18 with MobileNet blocks?
  - Q2: To what extent can you advance the accuracy-efficiency trade-off, meaning, what is the accuracy drop (or improvement) and what are the efficiency gains?
  - Q3: Are the efficiency savings in terms of theoretical metrics proportional to your real-device measurements? Could you analyze why?