Computer Science 481 / 581

M5 Lab (10 points)
Due Sunday, February 7, 2021 at 10pm

Overview

In this lab you will implement a convolutional neural network (CNN) following a tutorial, then reimplement it in PyTorch Lightning, a useful library for rapid development of deep learning models with minimal boilerplate code, before finally crafting a better CNN of your choosing in PyTorch Lightning.

Instructions

Group Work Policy

All parts of this lab are to be done collaboratively and synchronously in your permanent Worksheet/Lab group: no divide-and-conquer and no free-loaders. The collaborative format is not an arbitrary decision: the labs are designed to draw your attention to key concepts, and learning from and teaching your peers about these concepts through discussion will reduce the risk of misconceptions and deepen your understanding.

Unless your group has managed to finish before the scheduled lab time, your full group should attend the lab session. Anyone absent from the lab session must be noted on your submission (details below). During the lab time, please call me into your breakout room if you are stuck or have questions while working. This lab, and those to come in future weeks, are unlikely to be completed by the end of the scheduled lab time. Please arrange to meet as a lab group outside of class time to finish up any work, potentially over multiple working sessions.

Github Repository

Your work will be submitted via your official Worksheet/Lab Group github repository, which can be found at

```
https://github.com/hutchteaching/202110_csci581_worksheets_group1
```

with the 1 in `group1` replaced by whatever your actual group number is. Any code for this lab should be developed under version control from the beginning.

Development Environment

No jupyter notebook needed for this lab. Assuming your environment has been set up, you simply need to activate your virtual environment:

```
source ~/.venv/bin/activate
```

Data

In this lab you will use MNIST, which we have used before, and CIFAR-10, a dataset of 32×32 pixel RGB color images, each belonging to one of 10 classes.

Writeup

For this lab you will produce a plaintext write up, answering some questions posed in this pdf. Please answer the following questions (label them `L.1`, `L.2`, etc.) in your writeup file: `m5_lab.txt` (please spell-check, e.g. with `aspell -c m5_lab.txt`):

L.1) List the lab group members who were present for all parts of this lab (including any follow-up after the class period).

L.2) List any lab group members who were not present for at least some parts of this lab (including any follow-up after class), and please clarify the parts they missed.

L.3) How much time it took your group to complete this assignment (in hours).

You will answer additional questions in Parts I and II of the lab.

Tasks to Complete

Part I: Implement a CNN

Skills: convolutional neural networks, torchvision

1. Implement each step of the following tutorial (up to but excluding the GPU compute part), accumulating your code as you write it in an `m5_part1.py` file.

   `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`

   We will deviate from the tutorial in two ways.

   (a) Instead of

   ```
   trainloader = torch.utils.data.DataLoader(trainset, batch_size=4,
                                             shuffle=True, num_workers=2)
   testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                          download=True, transform=transform)
   testloader = torch.utils.data.DataLoader(testset, batch_size=4,
                                            shuffle=False, num_workers=2)
   ```

   you should run

   ```
   from torch.utils.data import random_split
   trainset,valset = random_split(trainset,[45000,5000])

   trainloader = torch.utils.data.DataLoader(trainset, batch_size=4,
                                             shuffle=True, num_workers=2)
   valloader   = torch.utils.data.DataLoader(valset, batch_size=4,
                                            shuffle=False, num_workers=2)
   ```

And then refer to `valloader` instead of `testloader` later in the tutorial.

2. Call your model class `CNN` instead of `Net`.

Questions

In your `m5_lab.txt` writeup, answer the following questions:

L.4) In a sentence or two, what is `transform` doing in the tutorial?

L.5) What are `torch.save` and `torch.load` doing?

Part II: Implement a CNN in PyTorch Lightning

Skills: PyTorch Lightning

1. Pip install PyTorch Lightning (make sure you are in your venv):

   ```
   pip install pytorch-lightning
   ```

2. Carefully look over `m5_demo.py`, which demonstrates the use of PyTorch Lightning for a simple multinomial logistic regression model on the MNIST dataset. Answer the questions L.6-L.7 below.

3. Create a `m5_part2.py` file, starting with the `m5_demo.py`, and modify it so that it uses the same CNN architecture and CIFAR10 datasets you used in Part I. When you call `python3 m5_part2.py -h` you should see this output:

   ```
   usage: m5_demo.py [-h] [-lr LR] [-mb MB] [-epochs EPOCHS] [-opt OPT]

   optional arguments:
     -h, --help      show this help message and exit
     -lr LR          The learning rate (float) [default: 0.1]
     -mb MB          The minibatch size (int) [default: 4]
     -epochs EPOCHS  The number of training epochs (int) [default: 100]
     -opt OPT        The optimizer: "adadelta", "adagrad", "adam", "rmsprop",
                     "sgd" (string) [default: "adam"]
   ```

   Your `m5_part2.py` should support each of these arguments.

Questions

In your `m5_lab.txt` writeup, answer the following questions:

L.6) In a sentence or two each, describe the role of each of the following methods in the `MNISTDataModule`:

- `__init__`
- `prepare_data`

- `setup`
- `train_dataloader`
- `val_dataloader`
- `test_dataloader`

L.7) In a sentence or two each, describe the role of each of the following methods in the `MultiLogReg`:

- `__init__`
- `forward`
- `eval_batch`
- `training_step`
- `validation_step`
- `test_step`
- `configure_optimizers`

Part III: Optimizing the CNN

Skills: CNNs.

1. Create a `m5_part3.py` code that starts out identical to `m5_part2.py`. Then explore different model architectures (number of layers, number of filters, kernel size, etc.) to see if you can get measurably better performance on validation set. Submit your best model as `m5_part3.py`. You are welcome to draw inspiration from well-known CNNs (e.g. `AlexNet`, `vgg11`) but must implement the model yourself.

Questions

In your `m5_lab.txt` writeup, answer the following questions:

L.8) In a short paragraph, describe your architecture, your recommendation for the hyperparameters to use to get the best performance out of the model, and accuracy *on test* of the model that achieved the best accuracy *on validation* (i.e., we pick our model on val, and then evaluate once at the end on test).

Submission Instructions

There should only one submission per group. Your submission consists of your code (`m5_part1.py`, `m5_part2.py` and `m5_part3.py`) and your writeup (`m5_lab.txt`), all pushed to the root of your worksheet github repo. Make sure your final code and writeup are pushed is pushed by looking at the github web interface).

In writeup answers `L.1` and `L.2`, you must separately list all of your worksheet/lab group members that were present and those that were absent. If someone was only present for part of the lab, indicate that and provide a brief explanation in `L.2` about what parts were missed. Failure to disclose when someone was absent is a violation of the course academic honesty policy (violates proper attribution of work).