Immersive Reader

2022-2023 Spring Quarter

- Home
- Announcements
- Modules
- Grades**1212**
- People
- Student Course Evaluations **1**

Lab 4: Tuning your autograd training loop for Fashion-MNIST and CIFAR-10

Lab 4: Tuning your autograd training loop for Fashion-MNIST and CIFAR-10

Due: Wed Apr 12, 2023 11:00amDue: Wed Apr 12, 2023 11:00am4/12/2023

Ungraded, 95 Possible Points

95 Possible Points

Attempt

Attempt 2

SUBMITTED on Apr 12, 2023 8:14amSUBMITTED on Apr 12, 2023 8:14am4/12/2023

Attempt 2 Score: N/

Add Comment

Next Up: Review Feedback

Unlimited Attempts Allowed

Details

As with previous labs, please work this lab without using anything from torch.nn (and also not torch.softmax), using only basic PyTorch operators such as *, +, /, mean, sum, etc. Once you meet the requirements of the lab, you are welcome to try out convolutions "just for fun," but with the flattened datasets I provide, this could be a challenge, so please let me know if you want this opportunity this week.

In this lab, you will tune the hyperparameters of your Lab 3 network to train on the Fashion-MNIST and CIFAR-10 datasets. Training on Fashion-MNIST is 95% of the lab grade, and training on CIFAR-10 is 5% of the lab grade.

Strictly cumulative lab

Before you begin your lab, the code MUST train successfully on the simple linear dataset used in the Week 3 lab. Submit or resubmit a competed Lab 3 before beginning this lab.

Prelab: Debugging with Fashion-MNIST and softmax + cross-entropy

Change to your working directory and run the command

```
cp /data/cs3450/mlp_from_scratch/week2/client.py client_fmnist.py
```

to copy the client into your working directory. (Note the name to avoid overwriting your work from last week!)

Open `client_fmnist.py` as a Python Jupyter notebook.

Copy your batch-by-batch training loop from last week's lab into this week's lab. Ensure it still trains with the previous dataset.

Then replace the MSE loss with the the softmax activation function and cross-entropy loss computations and switch the dataset to Fashion-MNIST.

Train network enough to ensure that these new layers are bug-free and carry gradients to the rest of the network. (For example, debug until you get 15 or 20% training accuracy so the performance is better than random guessing and better than always picking the same output category.) Then proceed to the systematic training described in the next section.

Prelab demo: Achieve 15-20% accuracy on Fashion-MNIST. Show your final training *and test* set accuracies along with an epoch-by-epoch plot of these to the instructor.

Debugging tip: Print loss for each batch and print accuracy every ten thousand samples or so. Don't always train for a whole epoch. Try longer and shorter training runs based on your gut feelings. Every tenth run or so train for a whole epoch just to make sure you aren't missing something.

Predictions and records of network tuning

Once your prelab demo is complete, make all of the hyperparameters in your network named constants very near the start of your script (or variables read from the environment.) Ensure the notebook prints out the final test accuracy as well as the amount of GPU memory used and the wall-clock runtime when the program finishes running. It's a good idea to print the hyperparameters to standard out to, so that these are available in your log files if you forget which run corresponds to which sbatch file (assuming you are using sbatch).

Then, in Rosie's jupyter notebook, open the notebook and choose **File -> Save as…**. Save the file as `run_fmnist.py`. Then run one `srun` or `sbatch` command for each of ten experiments.

As you train your network, for each experiment, record the following:

- All the critical hyperparameters you used on that training:
  - **Your motivation for the change in this training run. Wherever possible, provide a mathematical justification for your hypothesis.** Sometimes we go by hunches when training neural networks. That's OK, and don't ignore a possibility just because you can't turn a hunch into a mathematical justification. But try to provide mathematical justification for your changes whenever you can think of one. **Each run must have at least a brief narrative explanation of what you decided to change for that run.**
  - number of layers, number of nodes in fully-connected layers, number of channels (feature maps), filter size, padding, etc. in each convolutional layer (convolutional layers are optional).
  - learning rate
  - batch size
  - regularization constant or a statement that regularization was not used
  - roughly how long you trained the network (either in samples seen or epochs) before canceling the training. (You are encourage to report accuracy every 10,000 samples or so and to cancel training early if it is clearly not going well)
  - **An estimate of what the GPU memory and compute time will be for this run.**
    - For your initial run, estimate the memory usage based on the number of learnable parameters, doubled to count both the parameters and the gradients. You do not need to estimate runtime for your first run.
    - For your subsequent runs, use your results of your previous runs and your own analysis to estimate runtime. You can continue to assume constant training time per epoch when predicting runtimes, although you may find that larger networks take longer to train. **But a detailed Big-O analysis of how training time scales with network size is NOT required.**
  - The actual GPU memory usage reported by NVIDIA-SMI during the run. (You can run this command from within VS Code's shell to see the GPU that you are using for that project.)
  - Report actual training time. You can
  - The test accuracy achieved in that training run. use system time within your Python cell to record this.

Summarize this information in a table.

Include narrative text describing your motivation for the changes in each run in the narrative of your report. Refer to each experiment by number from your narrative to clearly explain for each experiment how you changed your numbers. Don't attempt to fit your narrative into the table and go beyond summarizing all experiments by discussing specific motivations for each experiment.

You should perform at least **ten experiments for each training set**, including the ones that you cancel part-way through training. You must achieve an accuracy of at least 50% on Fashion-MNIST and 40% on CIFAR-10, but you should also try to achieve a higher accuracy if you reach this goal before completing the ten runs.

Once you complete all ten runs, open your `run_fminst.py` in a Jupyter notebook so you can plot training curves for your most successful experiment.

CIFAR-10

Once your fashion-MNIST experiments are complete, begin your CIFAR-10 experiments. Copy your `run_fmnist.py` script to `run_cifar.py` and updated it to use the CIFAR dataset.

(If your same file supports easily switching between the two datsets, you can use your `run_fmnist.py` file for your CIFAR experiments.)

Hints

The provided Python template code provides a mechanism for reporting on your GPU usage. However, it doesn't show you the amount of memory available on the GPU. To see this, use `nvidia-smi` on a node, use `squeue` to determine what node you are on. Then `ssh` into that node, e.g. node 7:

```
ssh dh-node7
```

Then from within the node run

```
nvidia-smi
```

To see who is running a process, grab the process ID (in this case 50870) and run:

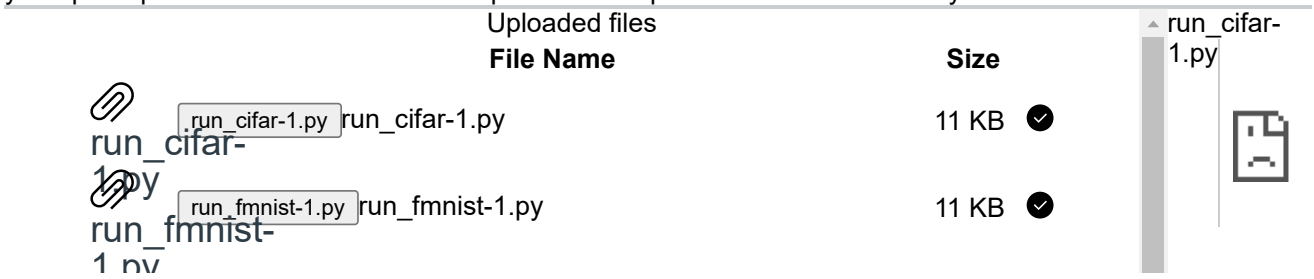
```
ps -Af | grep 50870
```

The username is on the far left of the output of this command.

Deliverables

- Your final `run_fmnist.py` and `run_cifar.py` scripts used for your last experiment. (You may make minor changes to the script as you work through the experiments.)
- A PDF report containing:
  - Your records of your ten test runs for each test set.
  - For each dataset, print the training accuracy after each epoch of training for your most accurate experiment and include a plot showing the training curve.

Upload your `run_fmnist.py`, `run_cifar.py` (if you do the CIFAR experiments in a different notebook), and your .pdf report file to canvas. Do NOT zip the files – upload each file individually.



File Name	Size
1e11fd18-...Lab_4.pdf 1e11fd18-1fdb-4db7-8b3d-5a5811950e50_Lab_4.pdf	118 KB

Previous Module Previous

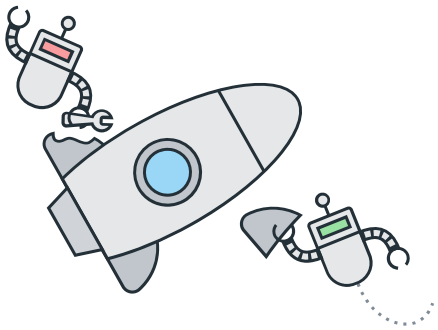
Links to an external site.

Try Again

Next Module Next

Links to an external site.

Sorry, Something Broke



Help us improve by telling us what happened

Report Issue