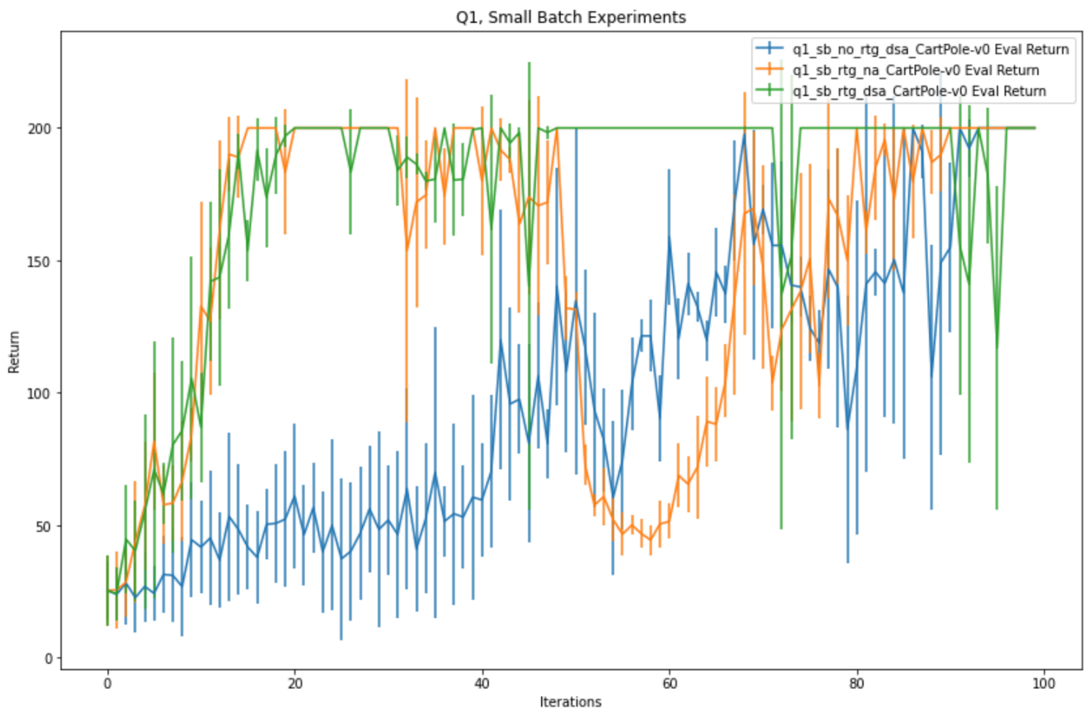
Aryan Luthra

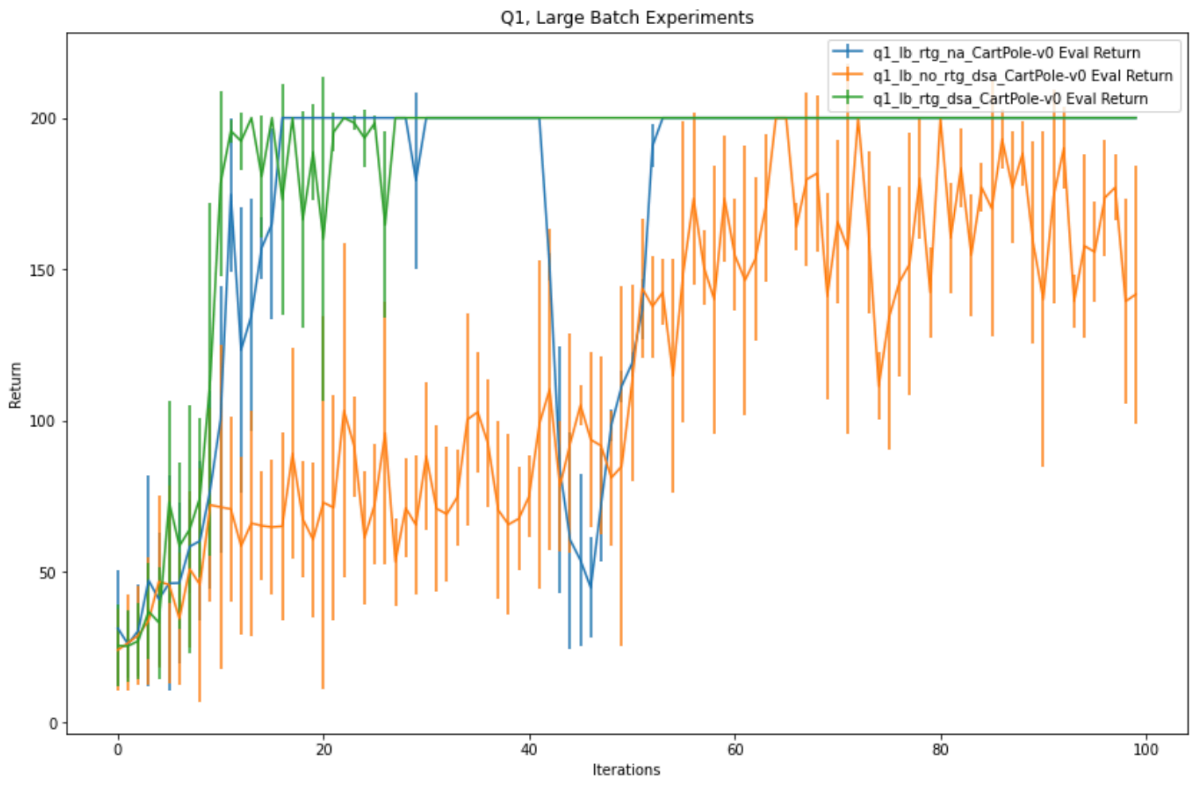
CS 285 HW2 Report

1. Experiment 1

    1. Small Batch results

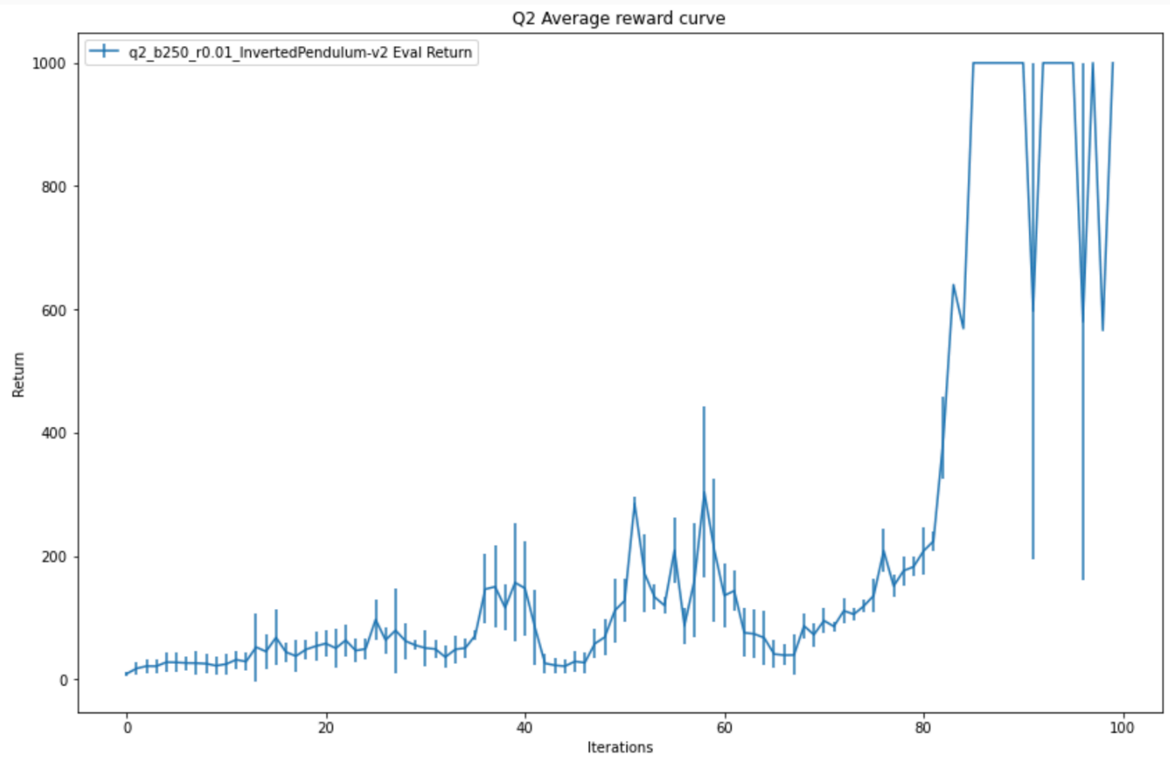


    2. Large batch Results



3. Reward to go was better
4. Advantage standardization did help in that the policy converged faster
5. Batch size also helped the policy coverage faster

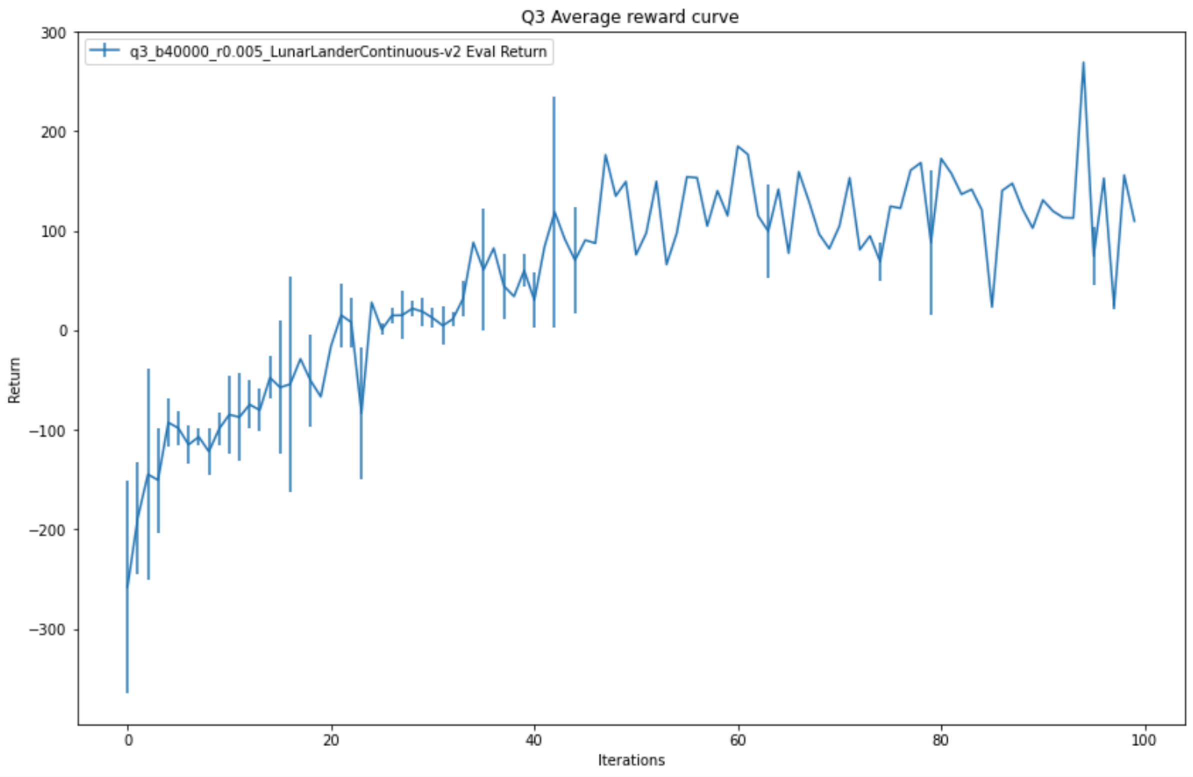
2. Experiment 2
    1. Learning curve



    2. I used the following command

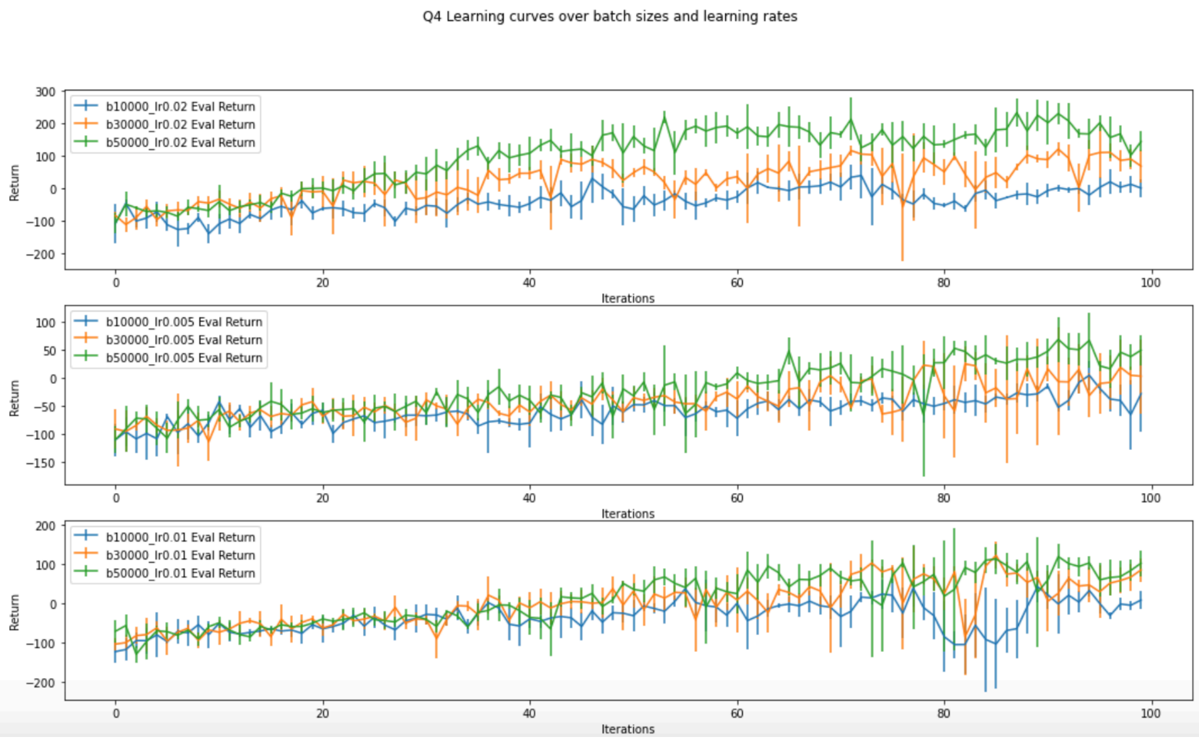
```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v2 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 250 -lr 0.01 -rtg \
--exp_name q2_b250_r0.01
```

3. Experiment 3

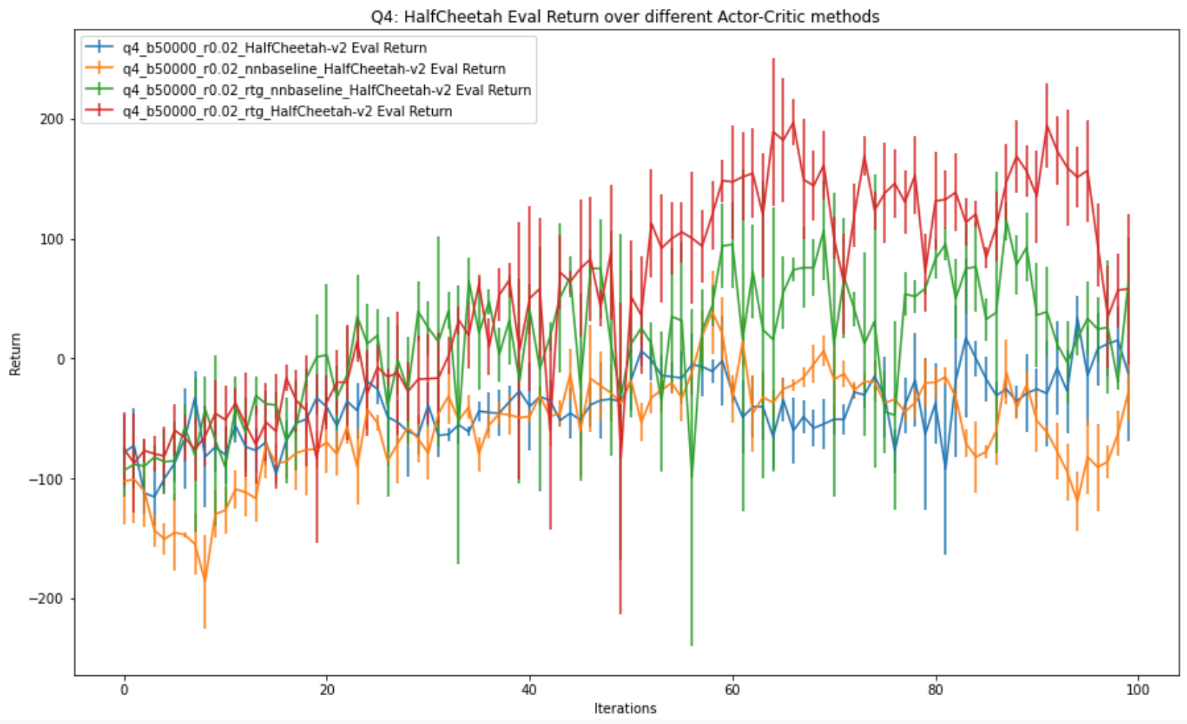


4. Experiment 4

    1. Part a: Decided to do 3 plots because 1 plot was way to messy



2. Part b: The best configuration was batch size: 50000 and learning-rate: 0.02



5. Bonus

Bonus

I parrallelized the trajectory sampling method using ray, which is a multtiprocessing library devoloped here at Berkeley which is super fast and incredibly easy to use.

To test this I ran the following command, with a massive batch size of 50,000, on both serial and parralel versions of the code.

```
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 50000 -rtg
```

The serial training loop took 1796.4s. After parrallelization, the training loop clocked in at 285.9s

This is a 6 times speed up, and my ray setup was only permitted to use 7 cores

Note: The results due to multiprocessing may differ as the samples collected may be in a different order every time you run the code. This is because sample collection order (which samples are finished collecting first), when done on multiple cores results in dependent on inherently random processes such as the number of cores, and even the temperature of each core.