Assignment 2: Policy Gradient

Andrew ID: `yulongl`
Collaborators: `None`
NOTE: Please do **NOT** change the sizes of the answer blocks or plots.

5 Small-Scale Experiments

5.1 Experiment 1 (Cartpole) – [25 points total]

5.1.1 Configurations

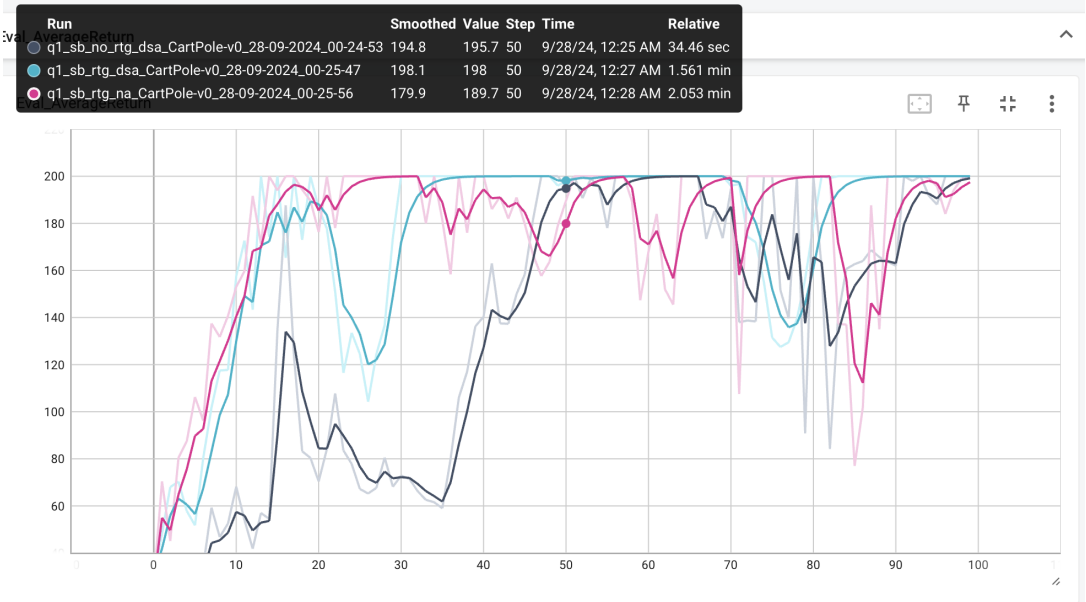
Q5.1.1

```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -dsa --exp_name q1_sb_no_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -rtg -dsa --exp_name q1_sb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
    -rtg --exp_name q1_sb_rtg_na

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -dsa --exp_name q1_lb_no_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -rtg -dsa --exp_name q1_lb_rtg_dsa

python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
    -rtg --exp_name q1_lb_rtg_na
```

5.1.2 Plots

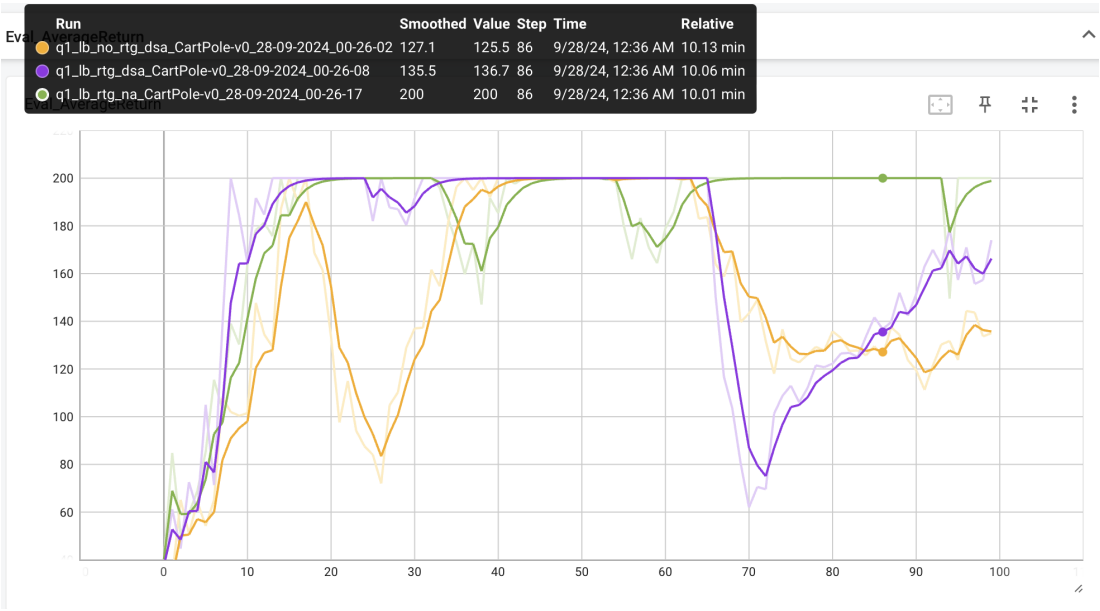
5.1.2.1 Small batch – [5 points]

Q5.1.2.1



5.1.2.2 Large batch – [5 points]

Q5.1.2.2



5.1.3 Analysis

5.1.3.1 Value estimator – [5 points]

Q5.1.3.1

Without advantage standardization, the reward to go value estimator performs better and more stable than the trajectory-centric value estimator for both the small batch and the large batch.

5.1.3.2 Advantage standardization – [5 points]

Q5.1.3.2

Based on my experiments, it is unclear whether advantage standardization is helpful for the Cartpole task. For both small batch and large batch, the performance with advantage standardization has higher variance.

5.1.3.3 Batch size – [5 points]

Q5.1.3.3

Based on my experiments, it is unclear whether batch size is helpful for the Cartpole task. For both small batch and large batch, the return s converge to around 200 at around 15 iterations. Large batch even seems to have higher variance. However, this might due to the randomness in the experiments.

5.2 Experiment 2 (InvertedPendulum) – [15 points total]

5.2.1 Configurations – [5 points]

Q5.2.1

```
for b in 1000 5000 10000
do
  for lr in 0.005 0.01 0.02 0.03
  do
    CUDA_VISIBLE_DEVICES=1 python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 \
    --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b $b -lr $lr -rtg \
    --exp_name q2_b${b}_r${lr}
  done
done
```

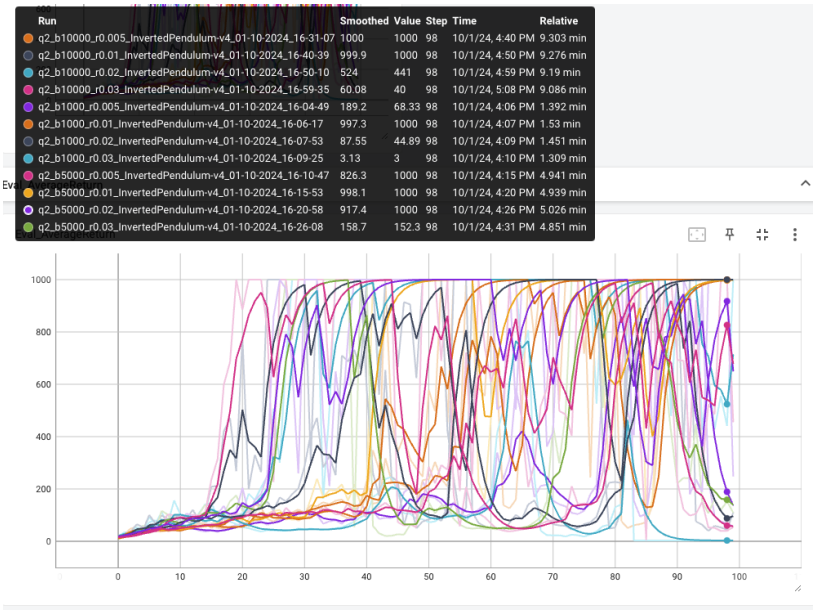
5.2.2 smallest b* and largest r* (same run) – [5 points]

Q5.2.2

b* = 1000, r* = 0.01

5.2.3 Plot – [5 points]

Q5.2.3



7 More Complex Experiments

7.1 Experiment 3 (LunarLander) – [10 points total]

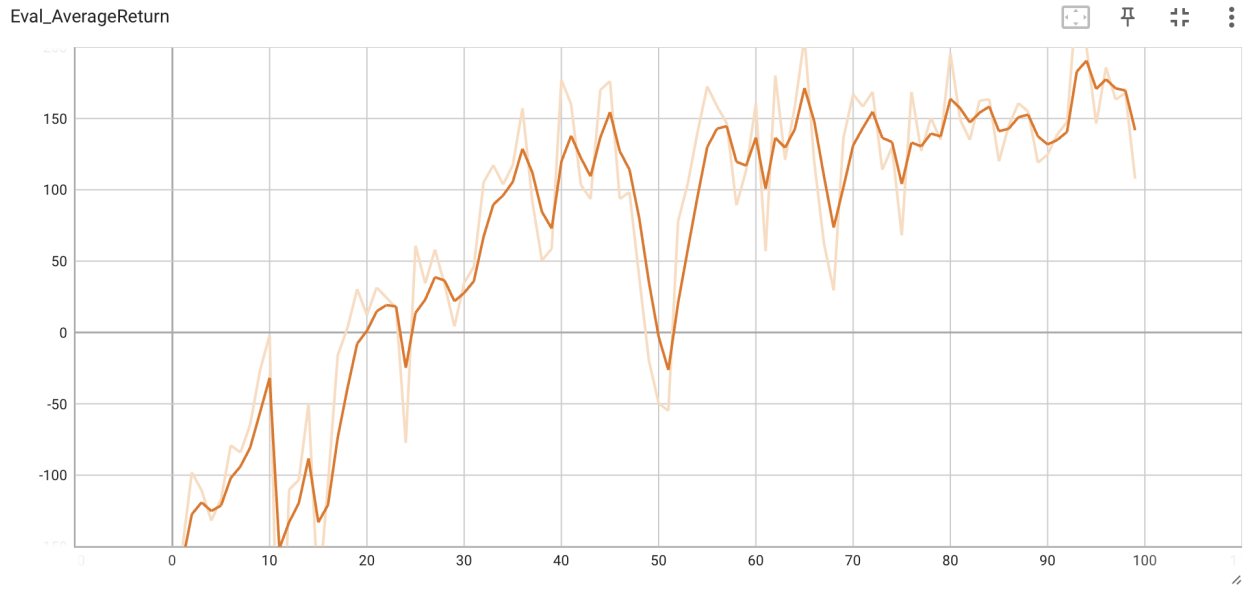
7.1.1 Configurations

Q7.1.1

```
python rob831/scripts/run_hw2.py \
    --env_name LunarLanderContinuous-v4 --ep_len 1000
    --discount 0.99 -n 100 -l 2 -s 64 -b 10000 -lr 0.005 \
    --reward_to_go --nn_baseline --exp_name q3_b10000_r0.005
```

7.1.2 Plot – [10 points]

Q7.1.2



7.2 Experiment 4 (HalfCheetah) – [30 points]

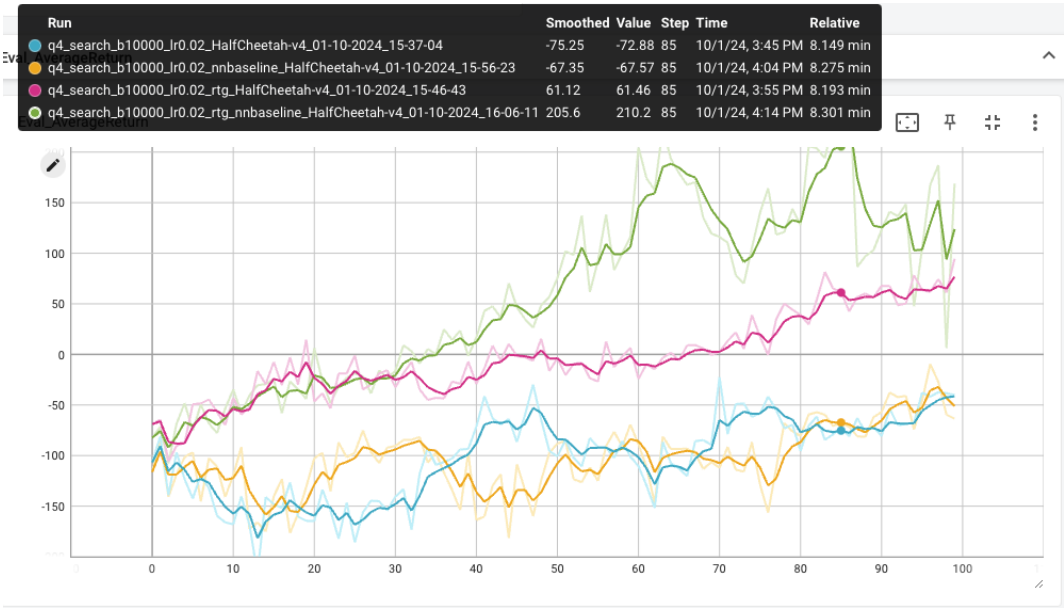
7.2.1 Configurations

Q7.2.1

```
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 \
    --exp_name q4_search_b10000_lr0.02
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg \
    --exp_name q4_search_b10000_lr0.02_rtg
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 --nn_baseline \
    --exp_name q4_search_b10000_lr0.02_nnbaseline
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn_baseline \
    --exp_name q4_search_b10000_lr0.02_rtg_nnbaseline
```

7.2.2 Plot – [10 points]

Q7.2.2



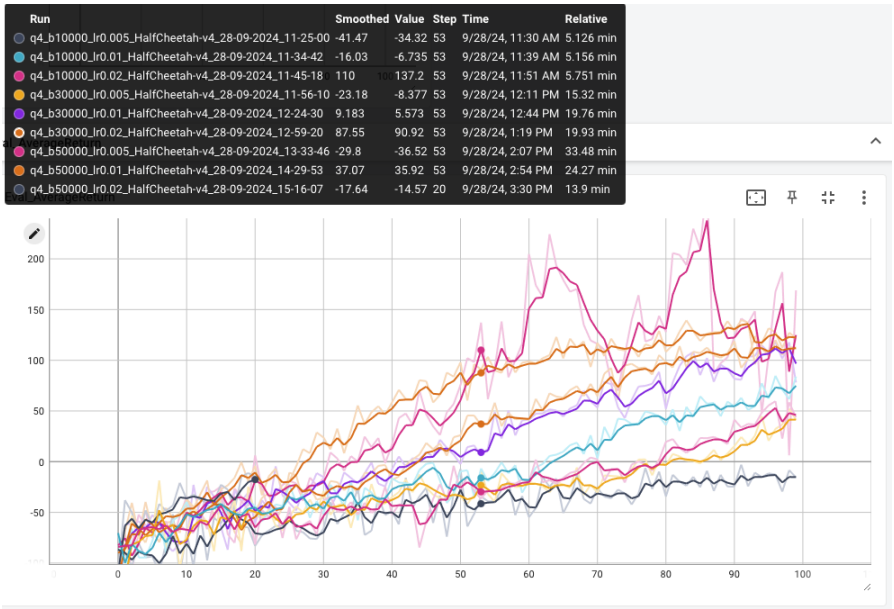
7.2.3 (Optional) Optimal b* and r* – [3 points]

Q7.2.3

I found that $b^* = 10000$ and $r^* = 0.02$ for the HalfCheetah task.

7.2.4 (Optional) Plot – [10 points]

Q7.2.4



7.2.5 (Optional) Describe how b* and r* affect task performance – [7 points]

Q7.2.5

With larger b*, the training is more stable and variance is reduced. With larger r*, the agent learns faster. In fact, with b*=30000 and r*=0.02, the agents achieves slightly inferior performance to b*=10000 and r*=0.02, but the training is more stable. With more iterations, the performance of b*=30000 and r*=0.02 is likely to be better. However, for 100 iterations, the performance of b*=10000 and r*=0.02 is the most optimal.

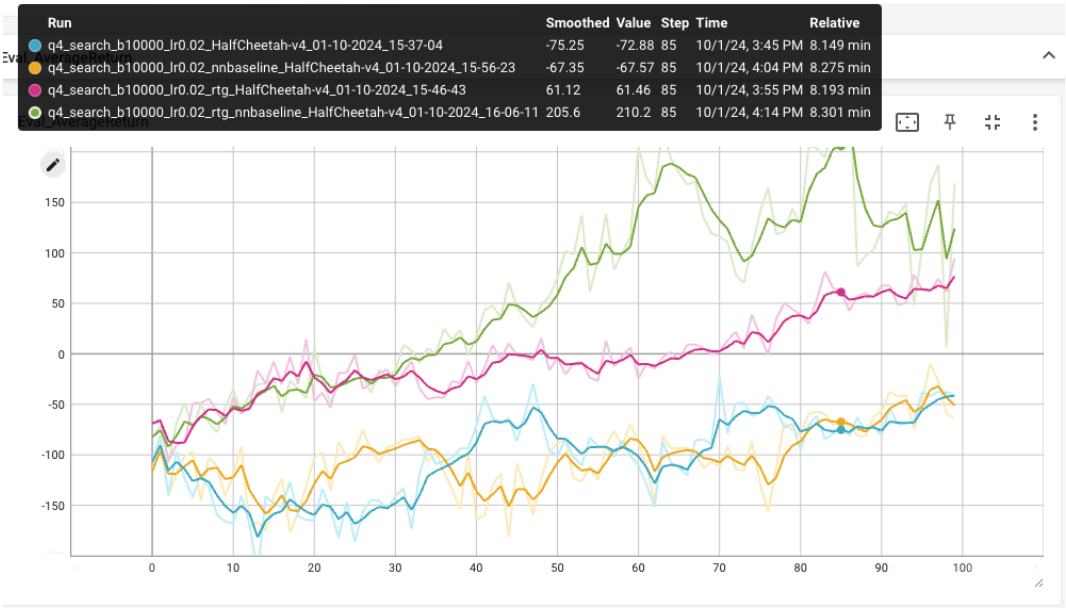
7.2.6 (Optional) Configurations with optimal b* and r* – [3 points]

Q7.2.6

```
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 \
    --exp_name q4_search_b10000_lr0.02
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg \
    --exp_name q4_search_b10000_lr0.02_rtg
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 --nn_baseline \
    --exp_name q4_search_b10000_lr0.02_nnbaseline
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn_baseline \
    --exp_name q4_search_b10000_lr0.02_rtg_nnbaseline
```

7.2.7 (Optional) Plot for four runs with optimal b* and r* – [7 points]

Q7.2.7



8 Implementing Generalized Advantage Estimation

8.1 Experiment 5 (Hopper) – [20 points]

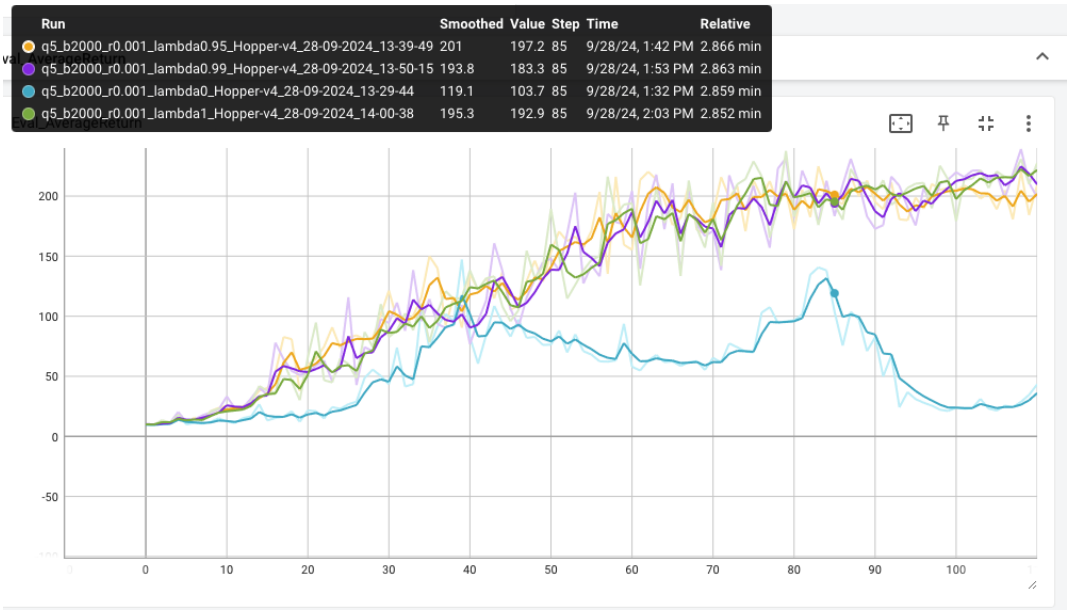
8.1.1 Configurations

Q8.1.1

```
# λ ∈ [0, 0.95, 0.99, 1]
python rob831/scripts/run_hw2.py \
    --env_name Hopper-v4 --ep_len 1000
    --discount 0.99 -n 300 -l 2 -s 32 -b 2000 -lr 0.001 \
    --reward_to_go --nn_baseline --action_noise_std 0.5 --gae_lambda <λ> \
    --exp_name q5_b2000_r0.001_lambda<λ>
```

8.1.2 Plot – [13 points]

Q8.1.2



8.1.3 Describe how λ affects task performance – [7 points]

Q8.1.3

With $\lambda = 0$, the performance is the worst, and performance is similar for $\lambda = 0.95, 0.99, 1$. This shows that a baseline is needed to reduce variance and improve the performance.

9 Bonus! (optional)

9.1 Parallelization – [15 points]

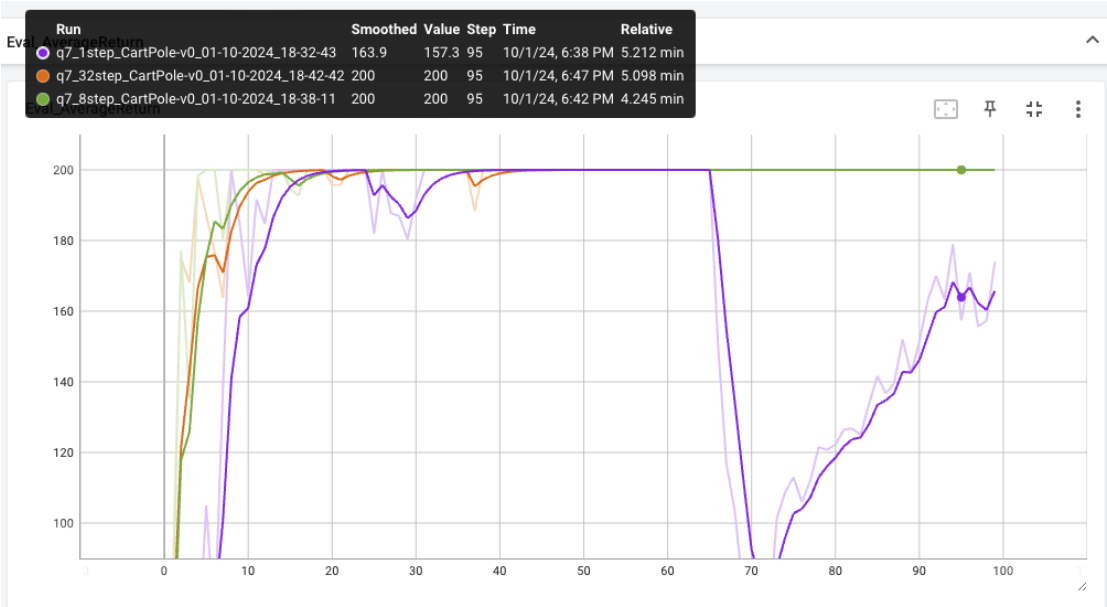
Q9.1

Difference in training time: 5m25s vs 1m12s (8 threads)

```
time python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg -dsa --num_threads 1 --exp_name
↪  q6_1_thread;
time python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg -dsa --num_threads 8 --exp_name
↪  q6_8_threads;
```

9.2 Multiple gradient steps – [5 points]

Q9.1




```
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -rtg -dsa --num_agent_train_steps_per_iter 1
↪  --exp_name q7_1step
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -rtg -dsa --num_agent_train_steps_per_iter 8
↪  --exp_name q7_8step
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -rtg -dsa --num_agent_train_steps_per_iter 32
↪  --exp_name q7_32step
```