Assignment 2: Policy Gradient

Andrew ID: `paramjib`
Collaborators:
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

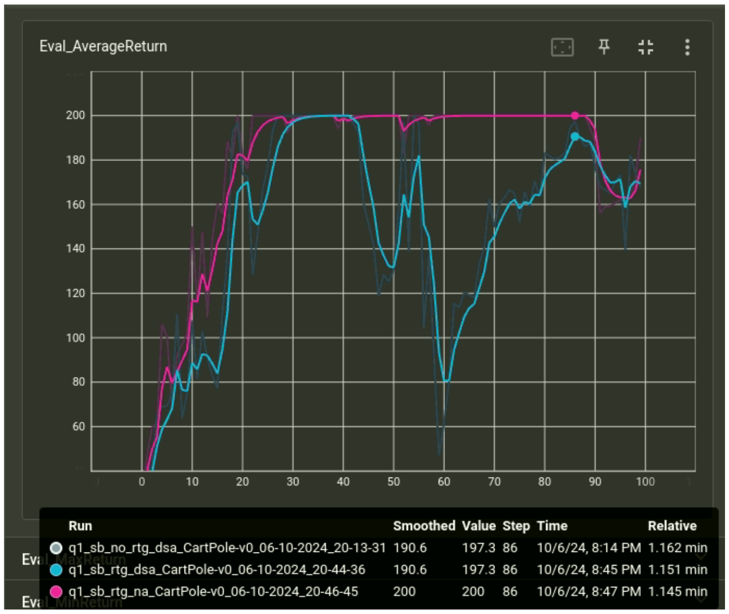
Q5.1.1

```
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]

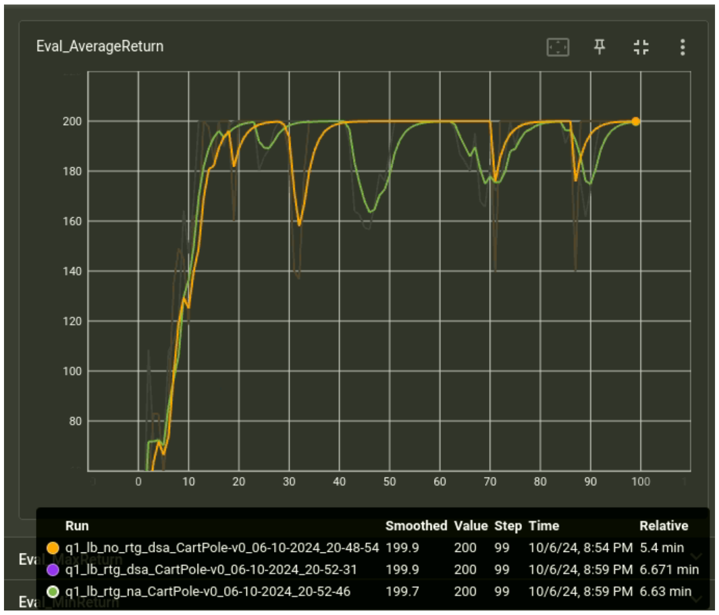
Q5.1.2.1



	Run	Smoothed	Value	Step	Time	Relative
●	q1_sb_no_rtg_dsa_CartPole-v0_06-10-2024_20-13-31	190.6	197.3	86	10/6/24, 8:14 PM	1.162 min
●	q1_sb_rtg_dsa_CartPole-v0_06-10-2024_20-44-36	190.6	197.3	86	10/6/24, 8:45 PM	1.151 min
●	q1_sb_rtg_na_CartPole-v0_06-10-2024_20-46-45	200	200	86	10/6/24, 8:47 PM	1.145 min

5.1.2.2 Large batch – [5 points]

Q5.1.2.2



	Run	Smoothed	Value	Step	Time	Relative
●	q1_lb_no_rtg_dsa_CartPole-v0_06-10-2024_20-48-54	199.9	200	99	10/6/24, 8:54 PM	5.4 min
●	q1_lb_rtg_dsa_CartPole-v0_06-10-2024_20-52-31	199.9	200	99	10/6/24, 8:59 PM	6.671 min
●	q1_lb_rtg_na_CartPole-v0_06-10-2024_20-52-46	199.7	200	99	10/6/24, 8:59 PM	6.63 min

5.1.3 Analysis

5.1.3.1 Value estimator – [5 points]

Q5.1.3.1

From all the above experiments, in both small and large batches, we can see that the experiments with reward-to-go value estimator had better results. This is because rtg only calculates the sum of rewards for the remainder of the trajectory from the current state, which tends to be a better estimate than calculating the sum from the beginning.

5.1.3.2 Advantage standardization – [5 points]

Q5.1.3.2

In both the small and large batch size experiments, advantage standardization improves the average return output. Normalization makes the episodes comparable to each other once they are standardized, and it helps reduce variance and increases learning stability.

5.1.3.3 Batch size – [5 points]

Q5.1.3.3

The batch size had a huge impact on the average return. In the runs with a large batch size, the best result of 200 was reached much quicker. The larger set of data points gives better generalization due to the diversity in the examples sampled, which in turn gives better gradient updates every time.

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

5.2.1 Configurations – [5 points]

Q5.2.1

```
python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  10000 -lr 0.02 -rtg --exp_name q2_b10000_r0.02


python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  10000 -lr 0.01 -rtg --exp_name q2_b10000_r0.01

python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  10000 -lr 0.03 -rtg --exp_name q2_b10000_r0.03

python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  5000 -lr 0.01 -rtg --exp_name q2_b5000_r0.01

python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  5000 -lr 0.02 -rtg --exp_name q2_b5000_r0.02

python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  15000 -lr 0.02 -rtg --exp_name q2_b15000_r0.02

python rob831/scripts/run_hw2.py --env_name InvertedPendulum-v4 --ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b
↪  15000 -lr 0.01 -rtg --exp_name q2_b15000_r0.01
```

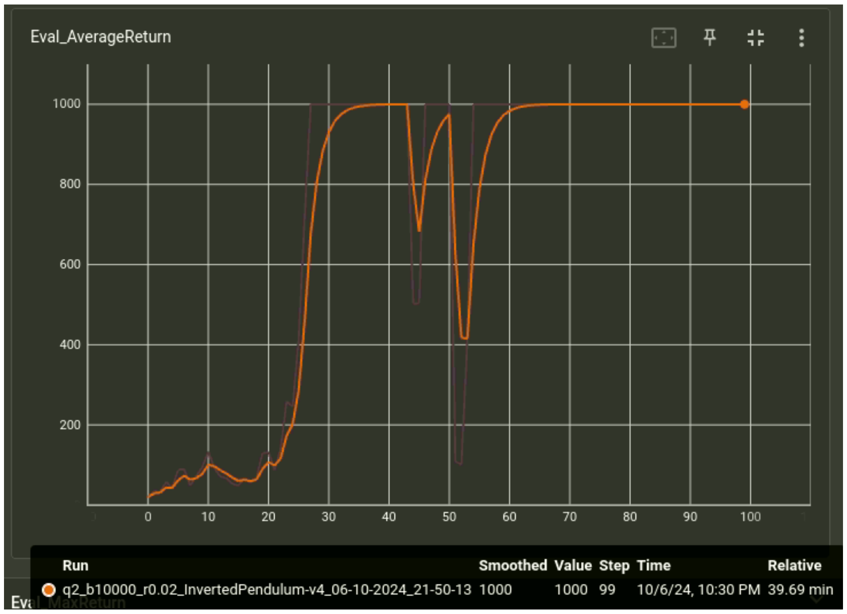
5.2.2 smallest b* and largest r* (same run) – [5 points]

Q5.2.2

I experimented with batch sizes 5000, 1000, 15000. And Learning rates of 0.01, 0.02, 0.03. Starting off with 5000, 0.01 gave good results. increasing the learning rate however caused the return to reach the max, faster, and drop with 10000, and 0.01, the results took too long to reach the max, and weren't steady. With 0.02, they were perfect, and stayed steady at the max. The result was better than with 5000, 0.01. With 0.03, the values fluctuated again after reaching the max. With 15000, the results were too unstable. Hence, the 10000 batch size and 0.02 learning rate yielded the best results.

5.2.3 Plot – [5 points]

Q5.2.3



Run	Smoothed	Value	Step	Time	Relative
● q2_b10000_r0.02_InvertedPendulum-v4_06-10-2024_21-50-13	1000	1000	99	10/6/24, 10:30 PM	39.69 min

7 More Complex Experiments

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

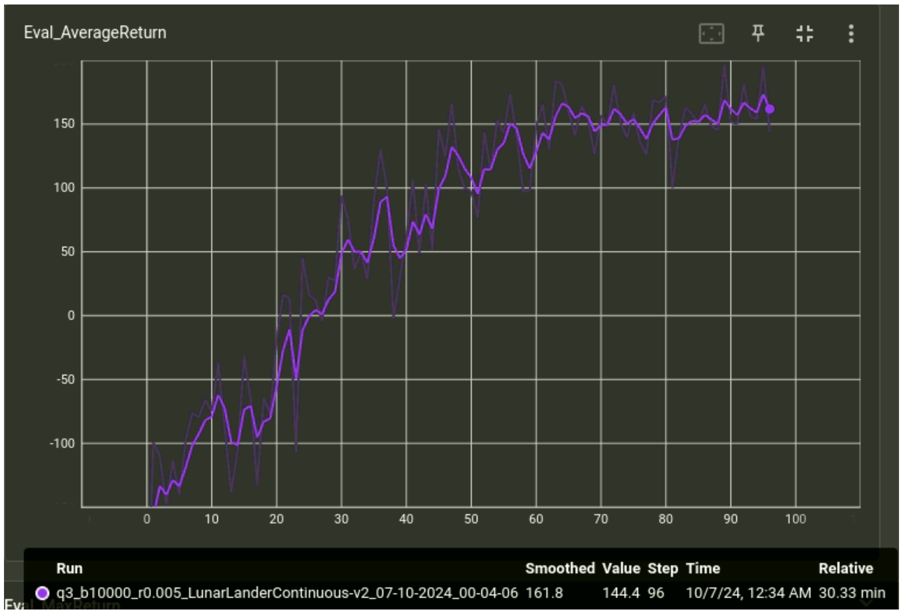
7.1.1 Configurations

Q7.1.1

```
python rob831/scripts/run_hw2.py --env_name LunarLanderContinuous-v2 --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



Run	Smoothed	Value	Step	Time	Relative
● q3_b10000_r0.005_LunarLanderContinuous-v2_07-10-2024_00-04-06	161.8	144.4	96	10/7/24, 12:34 AM	30.33 min

7.2 Experiment 4 (HalfCheetah) – [30 points]

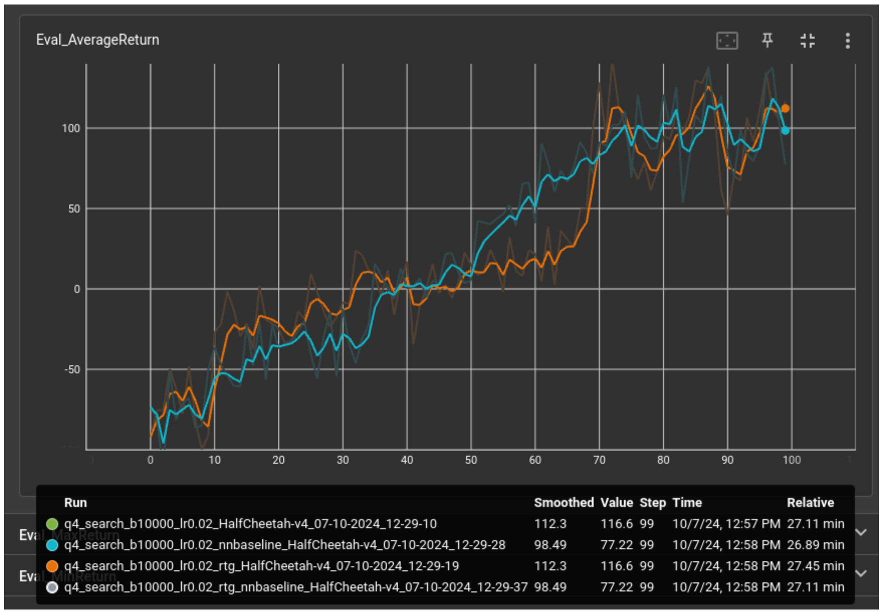
7.2.1 Configurations

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



	Run	Smoothed	Value	Step	Time	Relative
●	q4_search_b10000_lr0.02_HalfCheetah-v4_07-10-2024_12-29-10	112.3	116.6	99	10/7/24, 12:57 PM	27.11 min
●	q4_search_b10000_lr0.02_nnbaseline_HalfCheetah-v4_07-10-2024_12-29-28	98.49	77.22	99	10/7/24, 12:58 PM	26.89 min
●	q4_search_b10000_lr0.02_rtg_HalfCheetah-v4_07-10-2024_12-29-19	112.3	116.6	99	10/7/24, 12:58 PM	27.45 min
●	q4_search_b10000_lr0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_12-29-37	98.49	77.22	99	10/7/24, 12:58 PM	27.11 min

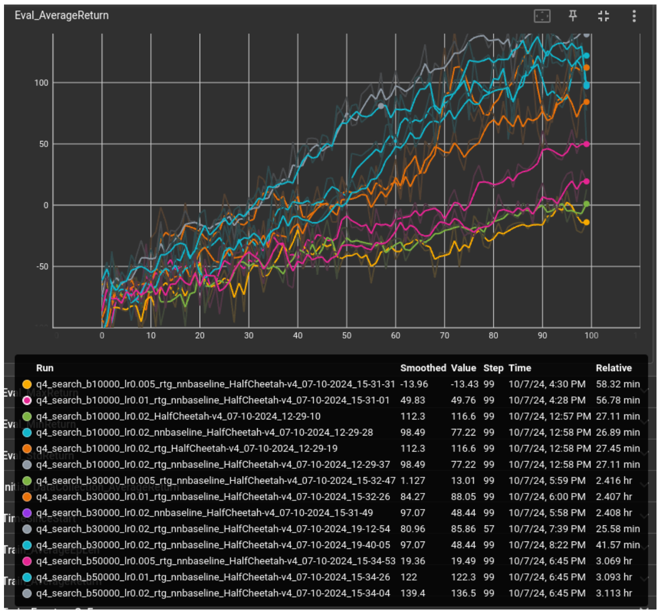
7.2.3 (Optional) Optimal b* and r* – [3 points]

Q7.2.3

My optimal batch size is 50000 and optimal learning rate is 0.02.

7.2.4 (Optional) Plot – [10 points]

Q7.2.4



	Run	Smoothed	Value	Step	Time	Relative
●	q4_search_b10000_lr0.005_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-31-31	-13.96	-13.43	99	10/7/24, 4:30 PM	58.32 min
●	q4_search_b10000_lr0.01_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-31-01	49.83	49.76	99	10/7/24, 4:28 PM	56.78 min
●	q4_search_b10000_lr0.02_HalfCheetah-v4_07-10-2024_12-29-10	112.3	116.6	99	10/7/24, 12:57 PM	27.11 min
●	q4_search_b10000_lr0.02_nnbaseline_HalfCheetah-v4_07-10-2024_12-29-28	98.49	77.22	99	10/7/24, 12:58 PM	26.89 min
●	q4_search_b10000_lr0.02_rtg_HalfCheetah-v4_07-10-2024_12-29-19	112.3	116.6	99	10/7/24, 12:58 PM	27.45 min
●	q4_search_b10000_lr0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_12-29-37	98.49	77.22	99	10/7/24, 12:58 PM	27.11 min
●	q4_search_b30000_lr0.005_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-32-47	1.127	13.01	99	10/7/24, 5:59 PM	2.416 hr
●	q4_search_b30000_lr0.01_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-32-26	84.27	88.05	99	10/7/24, 6:00 PM	2.407 hr
●	q4_search_b30000_lr0.02_nnbaseline_HalfCheetah-v4_07-10-2024_15-31-49	97.07	48.44	99	10/7/24, 5:58 PM	2.408 hr
●	q4_search_b30000_lr0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_19-12-54	80.96	85.86	57	10/7/24, 7:39 PM	25.58 min
●	q4_search_b30000_lr0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_19-40-05	97.07	48.44	99	10/7/24, 8:22 PM	41.57 min
●	q4_search_b50000_lr0.005_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-34-53	19.36	19.49	99	10/7/24, 6:45 PM	3.069 hr
●	q4_search_b50000_lr0.01_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-34-26	122	122.3	99	10/7/24, 6:45 PM	3.093 hr
●	q4_search_b50000_lr0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_15-34-04	139.4	136.5	99	10/7/24, 6:45 PM	3.113 hr

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

Q7.2.5

We know that having a larger set of datapoints gives better generalization since there is a diverse set of examples sampled. Here, the same was visible. The performance increases as the batch size increases from 10000 to 50000. The performance for 30000 and 50000 was similar. Learning rate dictates how much the weights are updated. With the learning rate as 0.02, the higher learning rate helps the model converge to the optimal result faster.

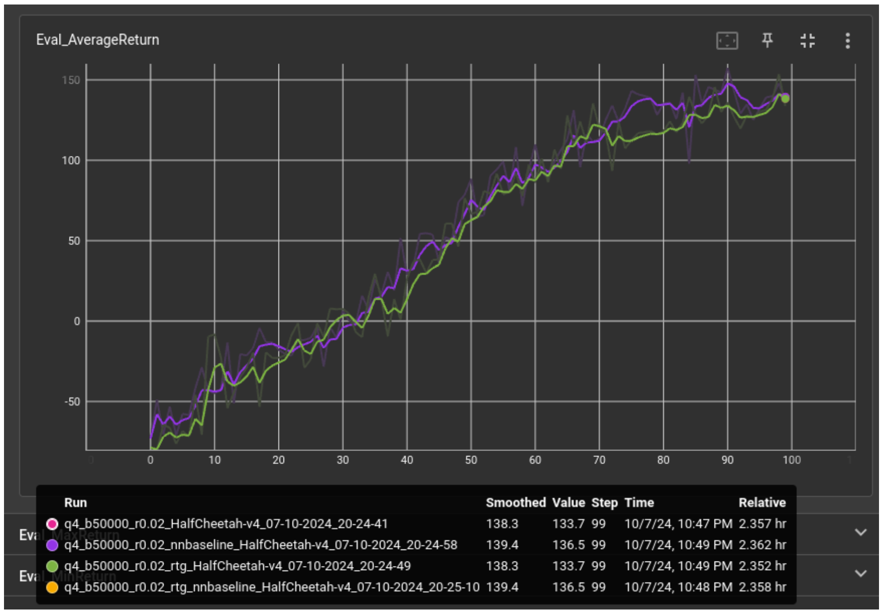
7.2.6 (Optional) Configurations with optimal b* and r* – [3 points]

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 50000 -lr
↪  0.02 --exp_name q4_b50000_r0.02

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr
↪  0.02 -rtg --exp_name q4_b50000_r0.02_rtg

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr
↪  0.02 --nn_baseline --exp_name q4_b50000_r0.02_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 --discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr
↪  0.02 -rtg --nn_baseline --exp_name q4_b50000_r0.02_rtg_nnbaseline
```

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

Q7.2.7



	Run	Smoothed	Value	Step	Time	Relative
●	q4_b50000_r0.02_HalfCheetah-v4_07-10-2024_20-24-41	138.3	133.7	99	10/7/24, 10:47 PM	2.357 hr
●	q4_b50000_r0.02_nnbaseline_HalfCheetah-v4_07-10-2024_20-24-58	139.4	136.5	99	10/7/24, 10:49 PM	2.362 hr
●	q4_b50000_r0.02_rtg_HalfCheetah-v4_07-10-2024_20-24-49	138.3	133.7	99	10/7/24, 10:49 PM	2.352 hr
●	q4_b50000_r0.02_rtg_nnbaseline_HalfCheetah-v4_07-10-2024_20-25-10	139.4	136.5	99	10/7/24, 10:48 PM	2.358 hr

8 Implementing Generalized Advantage Estimation

8.1 Experiment 5 (Hopper) – [20 points]

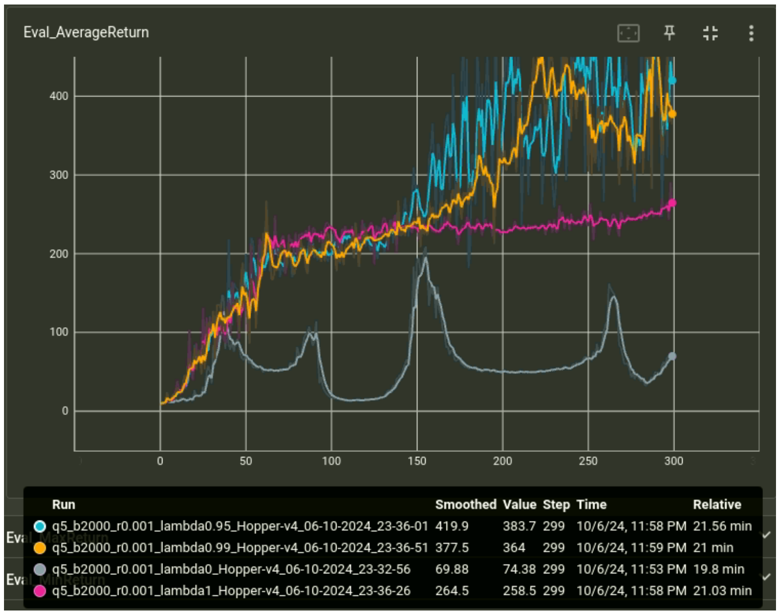
8.1.1 Configurations

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



	Run	Smoothed	Value	Step	Time	Relative
●	q5_b2000_r0.001_lambda0.95_Hopper-v4_06-10-2024_23-36-01	419.9	383.7	299	10/6/24, 11:58 PM	21.56 min
●	q5_b2000_r0.001_lambda0.99_Hopper-v4_06-10-2024_23-36-51	377.5	364	299	10/6/24, 11:59 PM	21 min
●	q5_b2000_r0.001_lambda0_Hopper-v4_06-10-2024_23-32-56	69.88	74.38	299	10/6/24, 11:53 PM	19.8 min
●	q5_b2000_r0.001_lambda1_Hopper-v4_06-10-2024_23-36-26	264.5	258.5	299	10/6/24, 11:58 PM	21.03 min

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

Q8.1.3

As stated in the assignment question, higher values of lambda emphasize advantage estimates with larger values of n. Hence, choosing a larger lambda increases the variance and oscillations in the average return, but decrease bias while getting the output to the desired. With lower lambdas, the estimates would be more stable but biased. Hence, lambdas need to be chosen while balancing the tradeoff between bias and fluctuations.

9 Bonus! (optional)

9.1 Parallelization – [15 points]

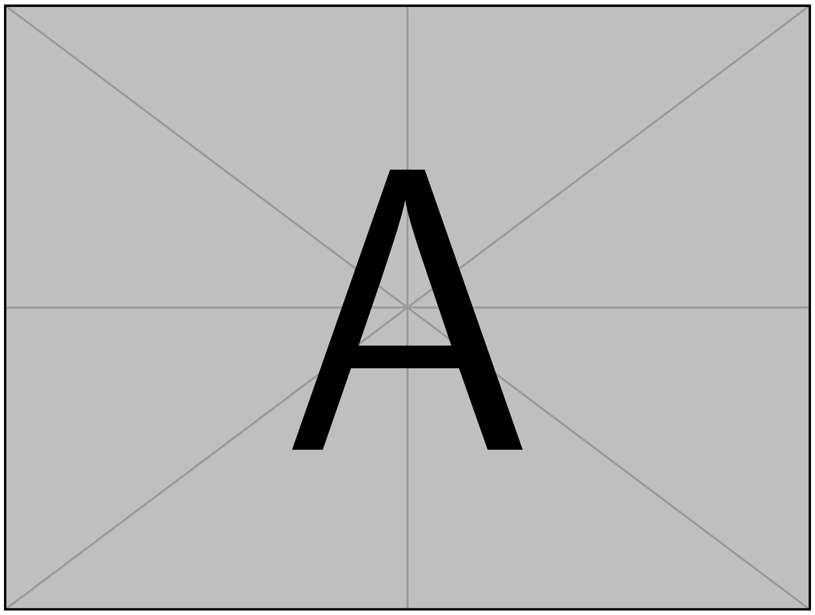
Q9.1

Difference in training time:

```
python rob831/scripts/run_hw2.py \
```

9.2 Multiple gradient steps – [5 points]

Q9.1



```
python rob831/scripts/run_hw2.py \
```