Assignment 2: Policy Gradient

Andrew ID: `rickyy`
Collaborators: `danielya`
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

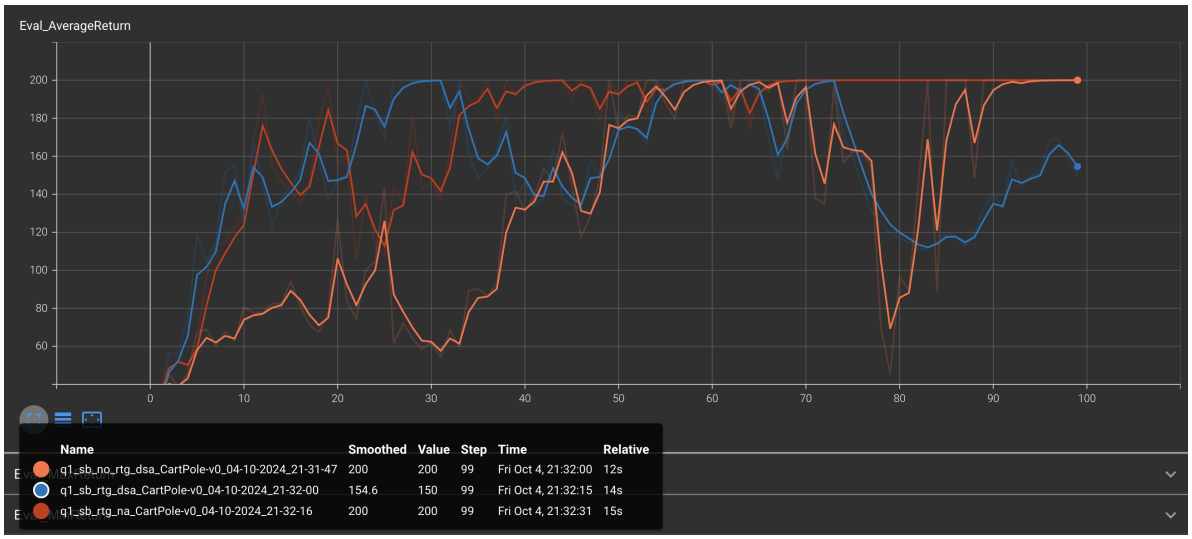
Q5.1.1

```
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

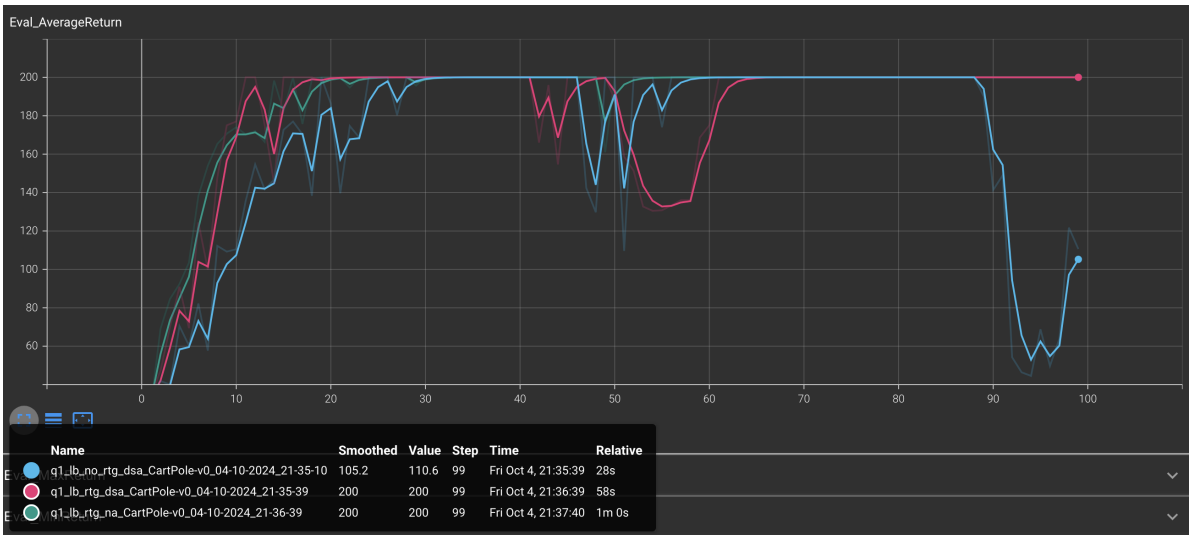
5.1.2.1 Small batch – [5 points]

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

The reward-to-go value estimator performs better in both large-batch and small-batch experiments. It only considers the cumulated rewards after this particular time step, which reduces the variance. Therefore the policy converges faster and more robustly.

5.1.3.2 Advantage standardization – [5 points]

Q5.1.3.2

For small batch, the advantage standardization helps. From my experiments, the one uses advantage standardization (red) converges while the other (blue) does not. However, for large batch, the improvements are not significant, both experiments with/without advantage standardization converges.

5.1.3.3 Batch size – [5 points]

Q5.1.3.3

In generally, batch size helps. From my experiments, we can see that tests with larger batch size converge faster and more stable.

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 1000 -lr 0.02 -rtg \
    --exp_name q2_b1000_r0.02
```

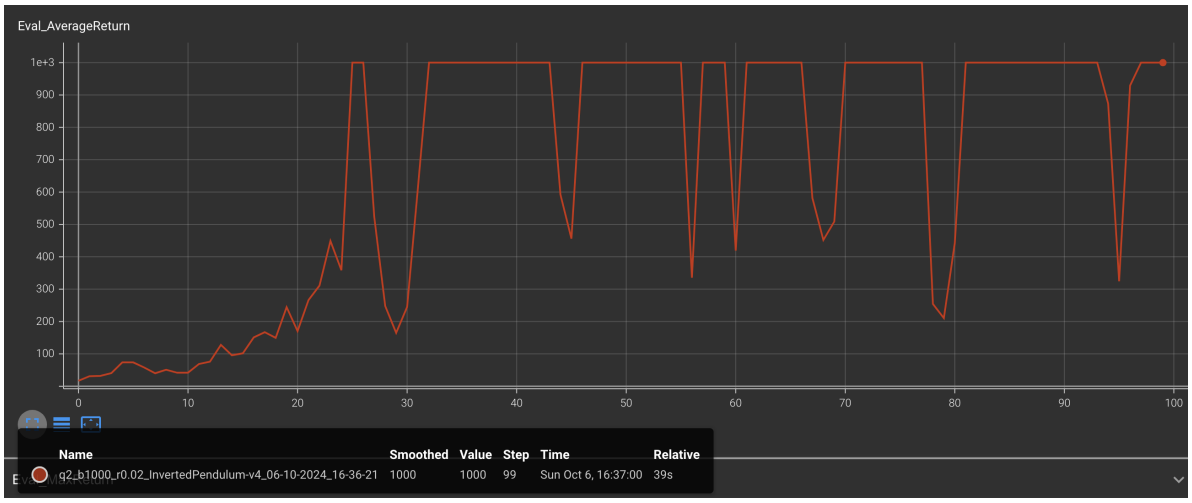
5.2.2 smallest b* and largest r* (same run) – [5 points]

Q5.2.2

b*: 1000, **r***: 0.02

5.2.3 Plot – [5 points]

Q5.2.3



7 More Complex Experiments

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

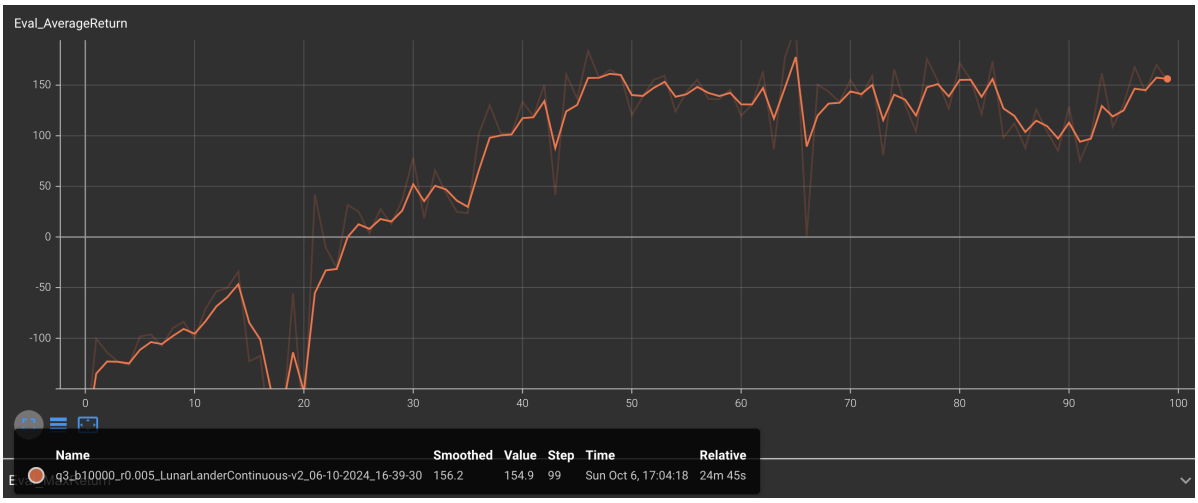
7.1.1 Configurations

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]

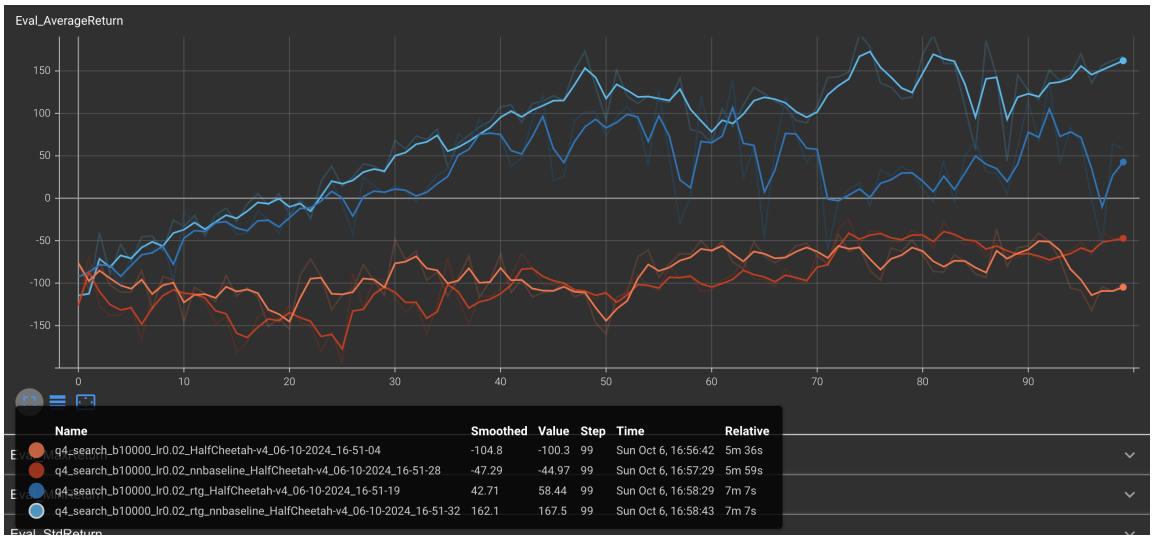
7.2.1 Configurations

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



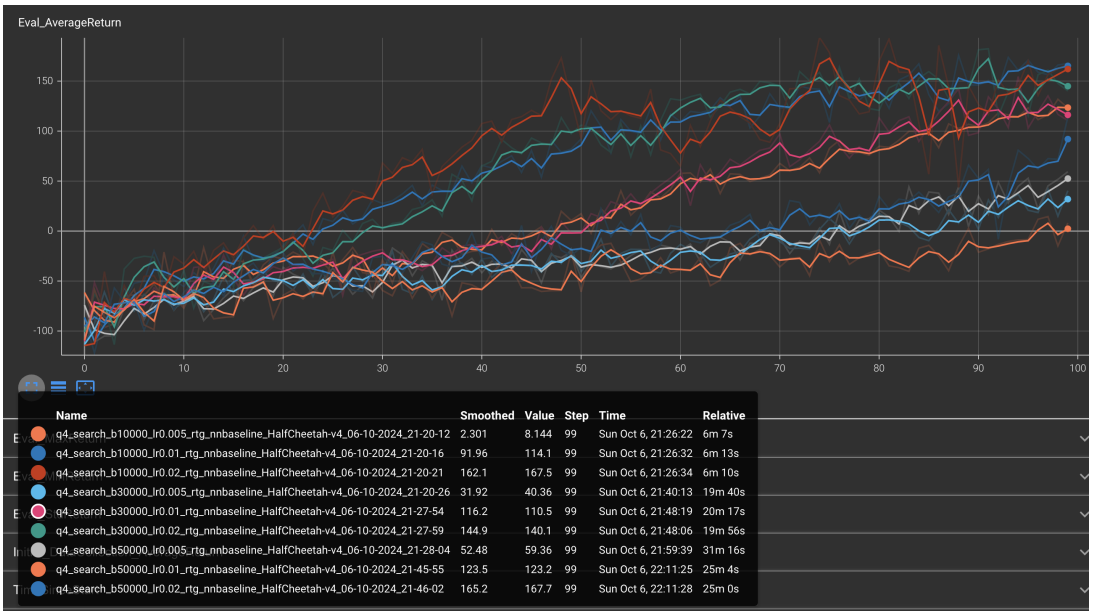
7.2.3 (Optional) Optimal b* and r* – [3 points]

Q7.2.3

b*: 10000, **r***: 0.02

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

Larger batch sizes make the training process more stable and less noisy. However, in my experiment, batch size 10000 reaches slightly higher performance than 30000 and 50000. That is probably because the larger the batch size, the longer the time needed for the model to converge. For the learning rate, 0.02 outperforms other settings. The reason might be that the model would converge faster for a larger learning rate, which makes the overall performances better than the lower learning rates.

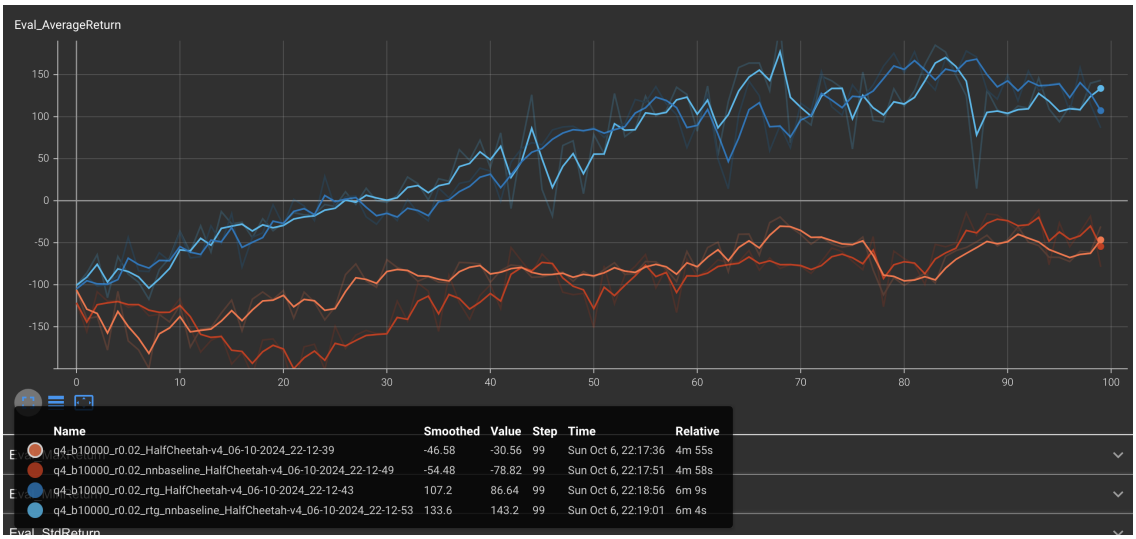
7.2.6 (Optional) Configurations with optimal b* and r* – [3 points]

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_b10000_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 10000 -lr 0.02 -rtg \
    --exp_name q4_b10000_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 10000 -lr 0.02 --nn_baseline \
    --exp_name q4_b10000_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 10000 -lr 0.02 -rtg --nn_baseline \
    --exp_name q4_b10000_r0.02_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]

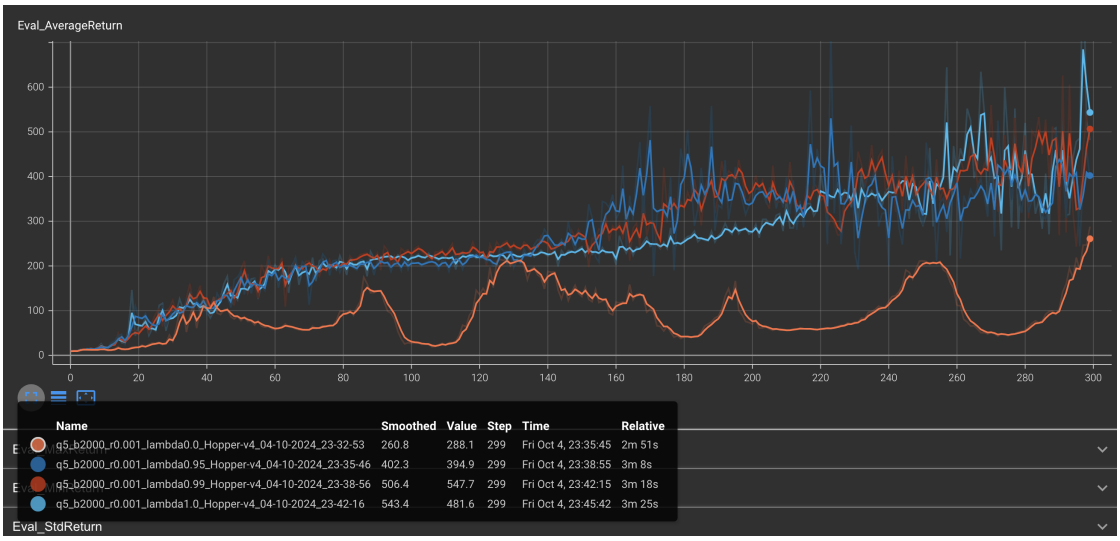
8.1.1 Configurations

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

For $\lambda = 0$, we can see that the model does not learn well and not converging because $\lambda = 0$ means that we are not utilizing the baseline. Generally speaking, as λ increases, we can see that the model is more stable and the performance is better.

9 Bonus! (optional)

9.1 Parallelization – [15 points]

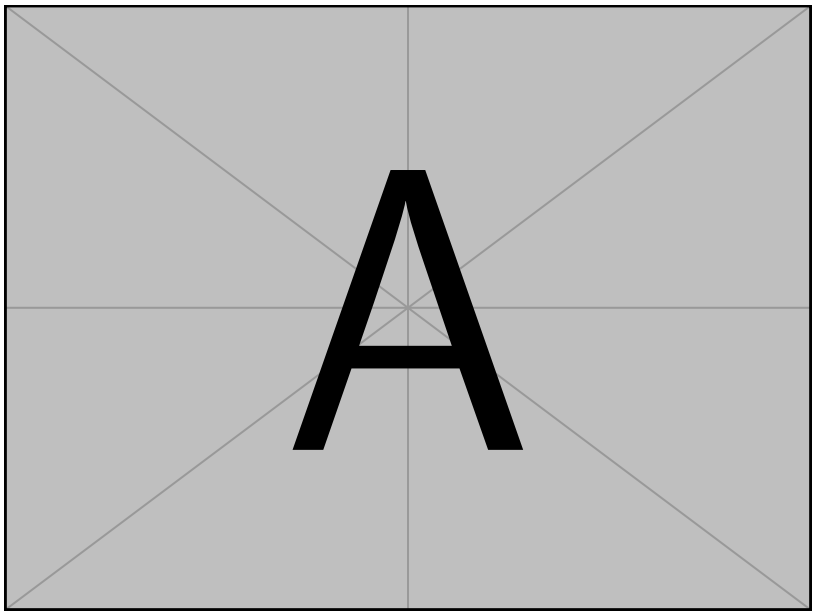
Q9.1

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 \
```