Assignment 2: Policy Gradient

Andrew ID: guangzhl
Collaborators: weihaoz
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

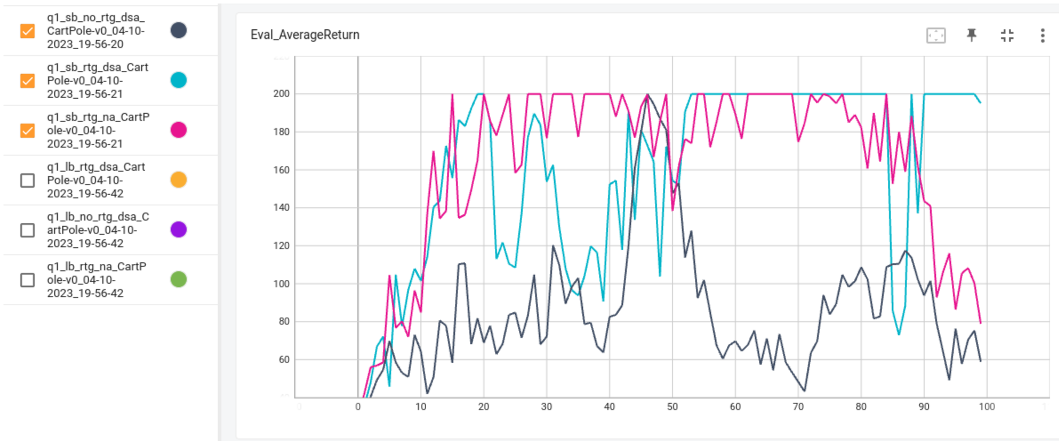
Q5.1.1 with exp1.sh

```bash
#!/bin/bash

# Experiment 1: q1_sb_no_rtg_dsa
echo "======================" && \
echo "Running experiment: q1_sb_no_rtg_dsa" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 1000 -dsa" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -dsa --exp_name q1_sb_no_rtg_dsa &

# Experiment 2: q1_sb_rtg_dsa
echo "======================" && \
echo "Running experiment: q1_sb_rtg_dsa" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 1000 -rtg -dsa" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg -dsa --exp_name q1_sb_rtg_dsa &

# Experiment 3: q1_sb_rtg_na
echo "======================" && \
echo "Running experiment: q1_sb_rtg_na" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 1000 -rtg" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 -rtg --exp_name q1_sb_rtg_na

# Experiment 4: q1_lb_no_rtg_dsa
echo "======================" && \
echo "Running experiment: q1_lb_no_rtg_dsa" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 5000 -dsa" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -dsa --exp_name q1_lb_no_rtg_dsa &

# Experiment 5: q1_lb_rtg_dsa
echo "======================" && \
echo "Running experiment: q1_lb_rtg_dsa" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 5000 -rtg -dsa" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -rtg -dsa --exp_name q1_lb_rtg_dsa &

# Experiment 6: q1_lb_rtg_na
echo "======================" && \
echo "Running experiment: q1_lb_rtg_na" && \
echo "Settings: --env_name CartPole-v0 -n 100 -b 5000 -rtg" && \
python rob831/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 -rtg --exp_name q1_lb_rtg_na

# echo "======================"
# echo "All experiments completed!"
```

5.1.2 Plots

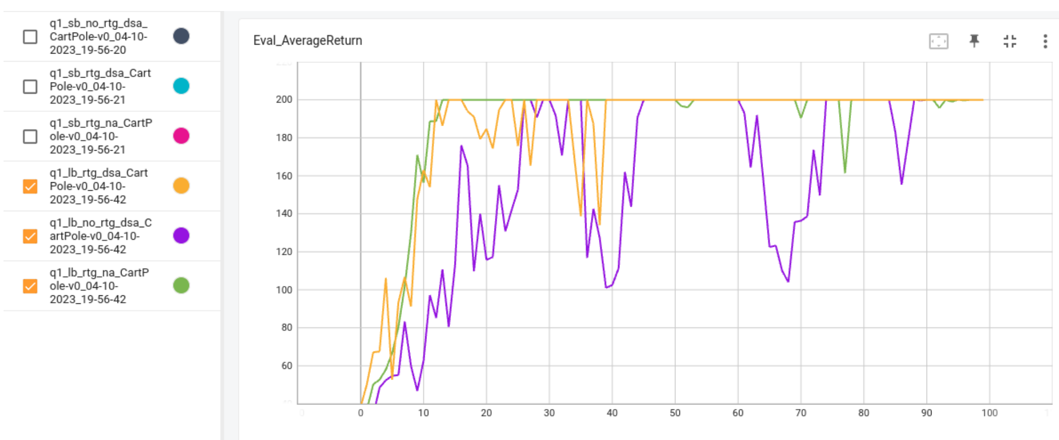
5.1.2.1 Small batch – [5 points]

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 one using reward-to-go is better and produces stabler results because of two reasons. First, reward-to-go makes physical sense to evaluate the subsequent rewards after an action is executed. Second, mathematically, this reduces misguidance/variance for the policy to learn better.

5.1.3.2 Advantage standardization – [5 points]

Q5.1.3.2

The advantage standardization perceivably makes the evaluation average reward stabler and higher compared to other settings.

5.1.3.3 Batch size – [5 points]

Q5.1.3.1

From the result, it is shown that the larger the batch size, the quicker to reach to higher rewards given same amount of iterations.

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

5.2.1 Configurations – [5 points]

Q5.2.1 with exp2.sh

```
#!/bin/bash

# Define arrays of values you want to try for b* and r*

# Reach training goal of 1000 within 100 iterations but not stablized
# batch_sizes=(900)
# learning_rates=(0.06)

# Approaching to the perform to the "B" level in piazza
batch_sizes=(20000)
learning_rates=(0.01)

# Counter for simultaneous processes
count=0

# Loop over all combinations of batch size and learning rate
for b in "${batch_sizes[@]}"; do
    for r in "${learning_rates[@]}"; do
        # Run the command in the background
        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 $r -rtg \
        --exp_name q2_b${b}_r${r} &

        # Increment the counter
        ((count++))

        # If 3 processes are running, wait for them to finish
        if ((count % 3 == 0)); then
            wait
        fi
    done
done

# Wait for any remaining processes to finish
wait
echo "Experiment 2 is done!"
```

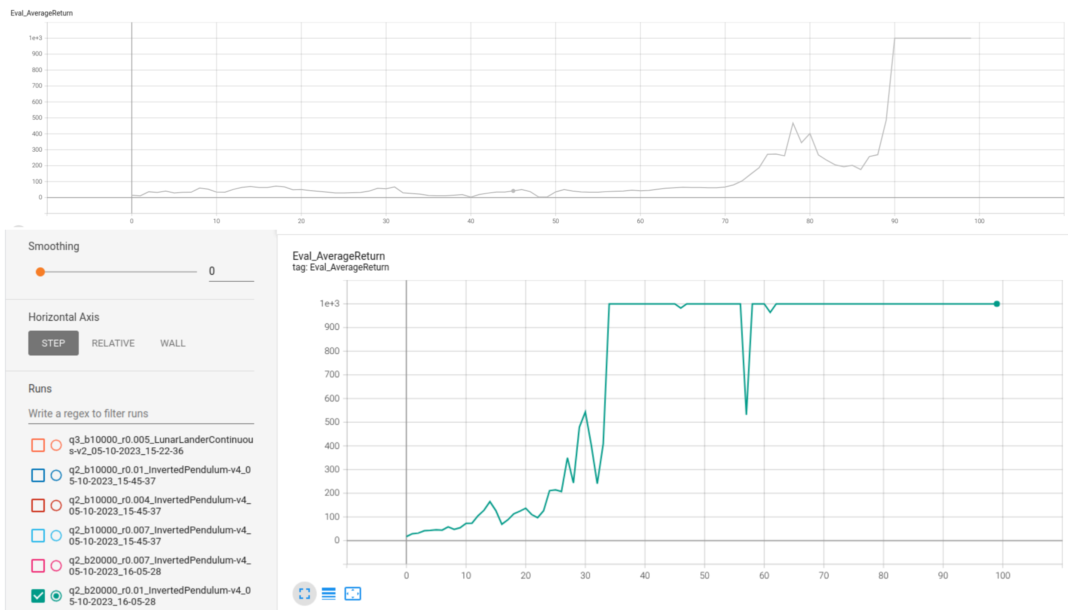
5.2.2 smallest b* and largest r* (same run) – [5 points]

Q5.2.2

Reach training goal of 1000 within 100 iterations but not stabilized
batch sizes = 900
learning rates= 0.06
Approaching to the perform to the ”B” level in piazza
batch sizes = 20000
learning rates = 0.01

5.2.3 Plot – [5 points]

Q5.2.3



7 More Complex Experiments

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

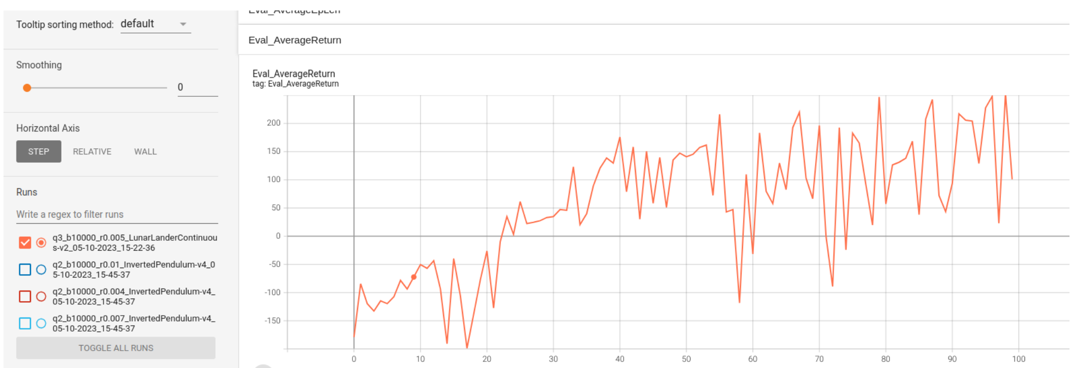
7.1.1 Configurations

Q7.1.1 with exp3.sh

```
python rob831/scripts/run_hw2.py \
    --env_name LunarLanderContinuous-v4 --ep_len 1000
    --discount 0.99 -n 100 -l 2 -s 64 -b 40000 -lr 0.005 \
    --reward_to_go --nn_baseline --exp_name q3_b40000_r0.005
```

7.1.2 Plot – [10 points]

Q7.1.2



7.2 Experiment 4 (HalfCheetah) – [30 points]

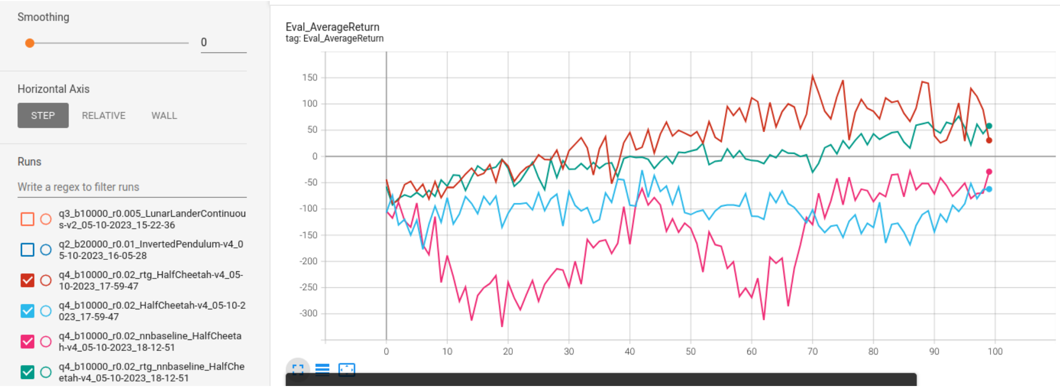
7.2.1 Configurations

Q7.2.1 with exp4.sh

```
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
```

7.2.2 Plot – [10 points]

Q7.2.2 with exp4.sh



As instructed on piazza, the average return of the best setup is bigger than 50.

7.2.3 Optimal b* and r* – [3 points]

Q7.2.3

This linear search for optimal parameters are skipped per instructions on piazza.

7.2.4 Describe how b* and r* affect task performance – [7 points]

Q7.2.4

In general the bigger the batch size, the better training performance per iteration. However, for learning rate, if it is too small, the policy learns really slow; if it is too big, the training would become unstable and resulting policy would have undefined behavior.

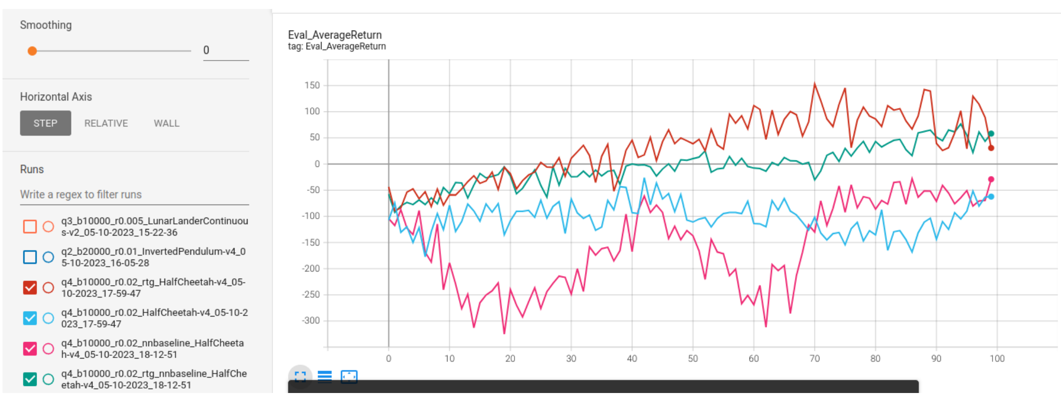
7.2.5 Configurations with optimal b* and r* – [3 points]

Q7.2.5 with exp5.sh

```
python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b <b*> -lr <r*> \
    --exp_name q4_b<b*>_r<r*>

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b <b*> -lr <r*> -rtg \
    --exp_name q4_b<b*>_r<r*>_rtg

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b <b*> -lr <r*> --nn_baseline \
    --exp_name q4_b<b*>_r<r*>_nnbaseline

python rob831/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
    --discount 0.95 -n 100 -l 2 -s 32 -b <b*> -lr <r*> -rtg --nn_baseline \
    --exp_name q4_b<b*>_r<r*>_rtg_nnbaseline
```

7.2.6 Plot for four runs with optimal b* and r* – [7 points]

Q7.2.6



As instructed on piazza, the average return of the best setup is bigger than 50.

8 Implementing Generalized Advantage Estimation

8.1 Experiment 5 (Hopper) – [20 points]

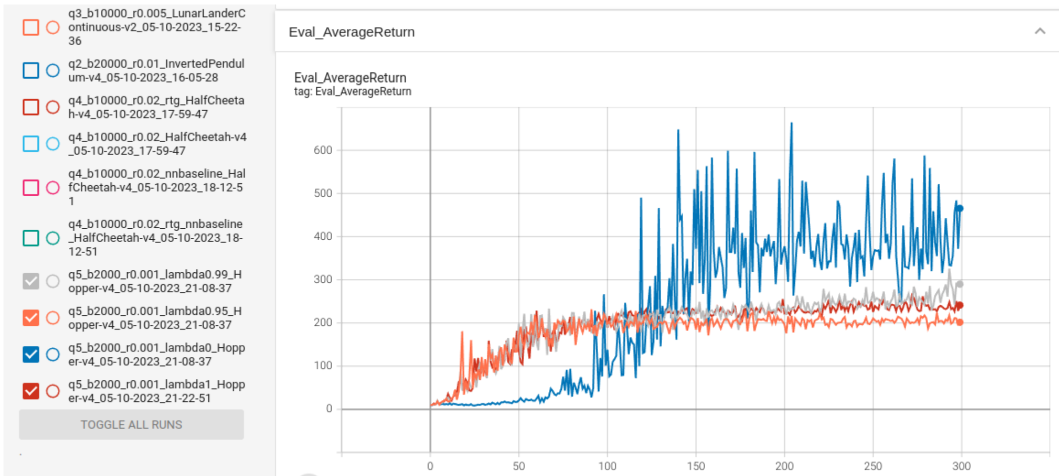
8.1.1 Configurations

Q8.1.1 with exp5.sh

```
#!/bin/bash

# Define the lambda values
lambdas=(0 0.95 0.99 1)

# Counter for simultaneous processes
count=0

# Loop over all lambda values
for lambda in "${lambdas[@]}"; do
    # Run the command in the background
    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 $lambda \
    --exp_name q5_b2000_r0.001_lambda$lambda &

    # Increment the counter
    ((count++))

    # If 3 processes are running, wait for them to finish
    if ((count % 3 == 0)); then
        wait
    fi
done

# Wait for any remaining processes to finish
wait
echo "Experiment 5 is done!"
```

8.1.2 Plot – [13 points]

Q8.1.2



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

Q8.1.3

From the result, when λ is 1 or 0.99 or 0.95, they are or at least very close to the vanilla neural network baseline estimator with summation of the discounted trajectory rewards upfront. This possesses less bias but introduces strong variance in the data collected. On the other hand, when λ is 0, the advantage at each step is the temporal difference where the upcoming reward is obtained from a value function. This has low variance but high bias. In the training evaluation result, it is somewhat explainable with the aforementioned: the temporal difference end of the spectrum has huge variance while the others are a lot stabler; the state-dependent value function learns slowly due to an initial lack of data but is able to enable the temporal difference setup (λ=0) to have better average return after learned a decent mapping from state to future returns.

9 Bonus! (optional)

9.1 Parallelization – [15 points]

Q9.1 not finished but with some progress!!!

Difference in training time:

```python
def worker(self, start, end, collect_policy, eval_policy, initial_expertdata, relabel_with_expert,
↪  start_relabel_with_expert, expert_policy, total_envsteps):
    for itr in range(start, end):
        print("\n\n********** Iteration %i ************"%itr)

        # decide if videos should be rendered/logged at this iteration
        if itr % self.params['video_log_freq'] == 0 and self.params['video_log_freq'] != -1:
            self.log_video = True
        else:
            self.log_video = False

        # decide if metrics should be logged
        if self.params['scalar_log_freq'] == -1:
            self.log_metrics = False
        elif itr % self.params['scalar_log_freq'] == 0:
            self.log_metrics = True
        else:
            self.log_metrics = False

        # collect trajectories, to be used for training
        training_returns = self.collect_training_trajectories(itr,
                            initial_expertdata, collect_policy,
                            self.params['batch_size'])
        paths, envsteps_this_batch, train_video_paths = training_returns
        with total_envsteps.get_lock():
            total_envsteps.value += envsteps_this_batch

        # add collected data to replay buffer
        self.agent.add_to_replay_buffer(paths)

        # train agent (using sampled data from replay buffer)
        train_logs = self.train_agent()

        # log/save
        if self.log_video or self.log_metrics:
            # perform logging
            print('\nBeginning logging procedure...')
            self.perform_logging(itr, paths, eval_policy, train_video_paths, train_logs)

            if self.params['save_params']:
                self.agent.save('{}/agent_itr_{}.pt'.format(self.params['logdir'], itr))
```

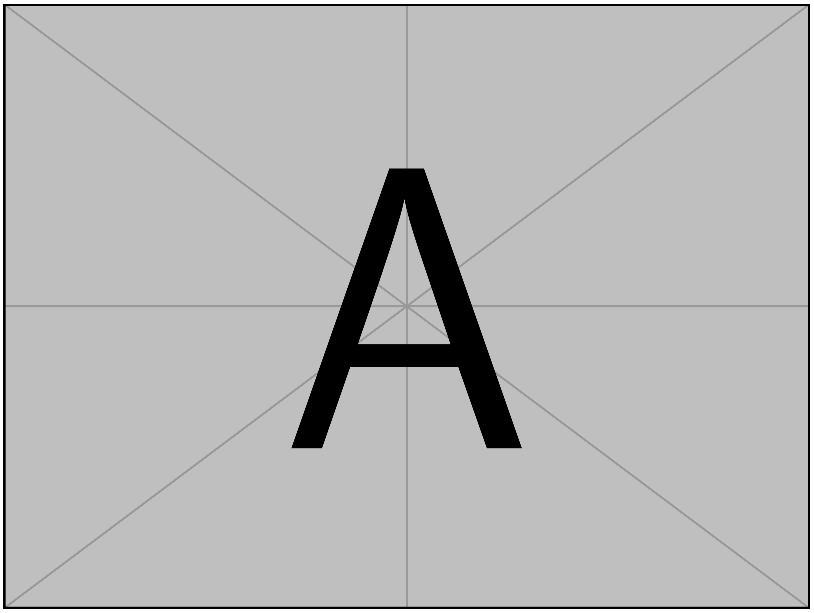
Q9.1 conti' not finished but with some progress!!!

Difference in training time:

```
def run_training_loop(self, n_iter, collect_policy, eval_policy, initial_expertdata=None,
↪   relabel_with_expert=False, start_relabel_with_expert=1, expert_policy=None):
    """
    :param n_iter:  number of (dagger) iterations
    :param collect_policy:
    :param eval_policy:
    :param initial_expertdata:
    :param relabel_with_expert:  whether to perform dagger
    :param start_relabel_with_expert: iteration at which to start relabel with expert
    :param expert_policy:
    """

    # init vars at beginning of training
    self.total_envsteps = mp.Value('i', 0, lock=True)  # multiprocessing Value to share data across processes
    self.start_time = time.time()

    # Number of workers (logical cores)
    num_workers = mp.cpu_count()

    # Split iterations across workers
    iter_per_worker = n_iter // num_workers

    processes = []
    for i in range(num_workers):
        start_iter = i * iter_per_worker
        end_iter = (i + 1) * iter_per_worker if i != num_workers - 1 else n_iter

        p = mp.Process(target=self.worker, args=(start_iter, end_iter, collect_policy, eval_policy,
        ↪   initial_expertdata, relabel_with_expert, start_relabel_with_expert, expert_policy,
        ↪   self.total_envsteps))
        p.start()
        processes.append(p)

    for p in processes:
        p.join()

    self.total_envsteps = self.total_envsteps.value  # Convert back to regular int after all processes have
    ↪   finished
```

9.2 Multiple gradient steps – [5 points]

Q9.1



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