Assignment 2: Policy Gradients

Due September 25, 11:59 pm

4 Policy Gradients

- Create two graphs:
  - In the first graph, compare the learning curves (average return vs. number of environment steps) for the experiments prefixed with `cartpole`. (The small batch experiments.)
  - In the second graph, compare the learning curves for the experiments prefixed with `cartpole_lb`. (The large batch experiments.)

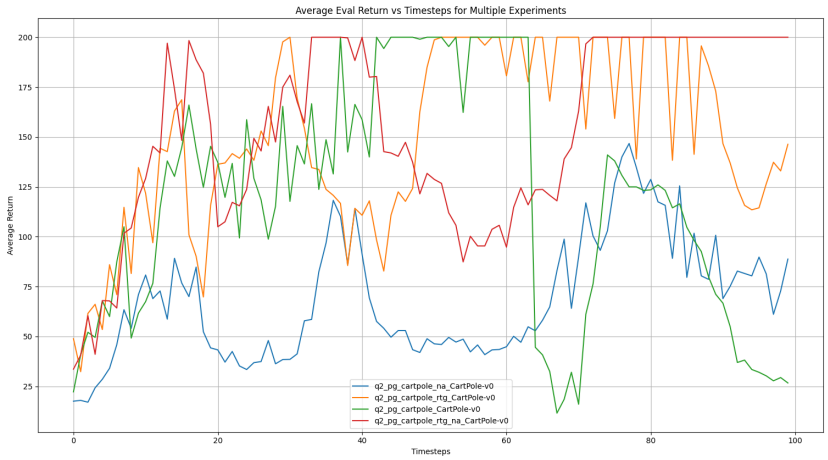

Figure 1: Learning curves for small batch experiments prefixed with `cartpole`.

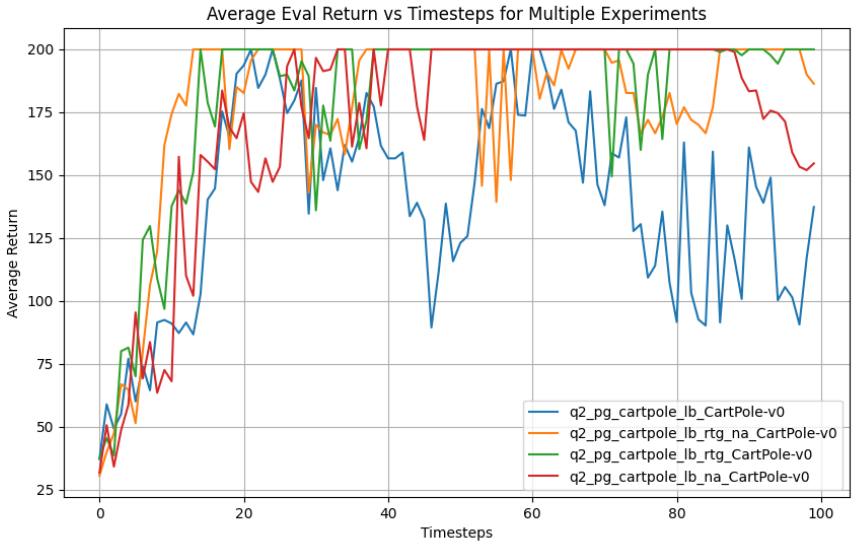

Figure 2: Learning curves for large batch experiments prefixed with `cartpole_lb`.

- Answer the following questions briefly:
  - Which value estimator has better performance without advantage normalization: the trajectory-centric one, or the one using reward-to-go?
    Answer: The value estimator using reward-to-go generally has better performance without advantage normalization because it provides a more accurate estimate of future rewards, leading to more effective policy updates. The overall variance is reduced when we consider only the rewards-to-go, which increases performance.
  - Did advantage normalization help?
    Answer: Yes, advantage normalization helped by reducing the variance of the policy gradient estimates, leading to more stable and faster learning. We see from the plots that the reward hits the maximum much quicker than in the case without the normalization.
  - Did the batch size make an impact?
    Answer: Yes, the batch size made a significant impact. Larger batch sizes tend to reduce the variance of the gradient estimates, leading to more stable learning, though they also increase the computational cost per update. We see from the plots that the reward reaches 200 much quicker, and the average returns are also higher when we use larger batches.
- Provide the exact command line configurations (or `#@params` settings in Colab) you used to run your experiments, including any parameters changed from their defaults.

SNO	Experiment Name	Iterations	Batch Size	RTG	Normalize Advantages
1	cartpole	100	1000	No	No
2	cartpole_rtg	100	1000	Yes	No
3	cartpole_na	100	1000	No	Yes
4	cartpole_rtg_na	100	1000	Yes	Yes
5	cartpole_lb	100	4000	No	No
6	cartpole_lb_rtg	100	4000	Yes	No
7	cartpole_lb_na	100	4000	No	Yes
8	cartpole_lb_rtg_na	100	4000	Yes	Yes

Table 1: Experiment configurations and parameters.

In addition, the following parameters were fixed across all experiments:
- Discount Rate: 1
- Learning Rate: 0.005
- Number of Layers: 2
- Layer Size: 64

5 Neural Network Baseline

- Plot a learning curve for the baseline loss.

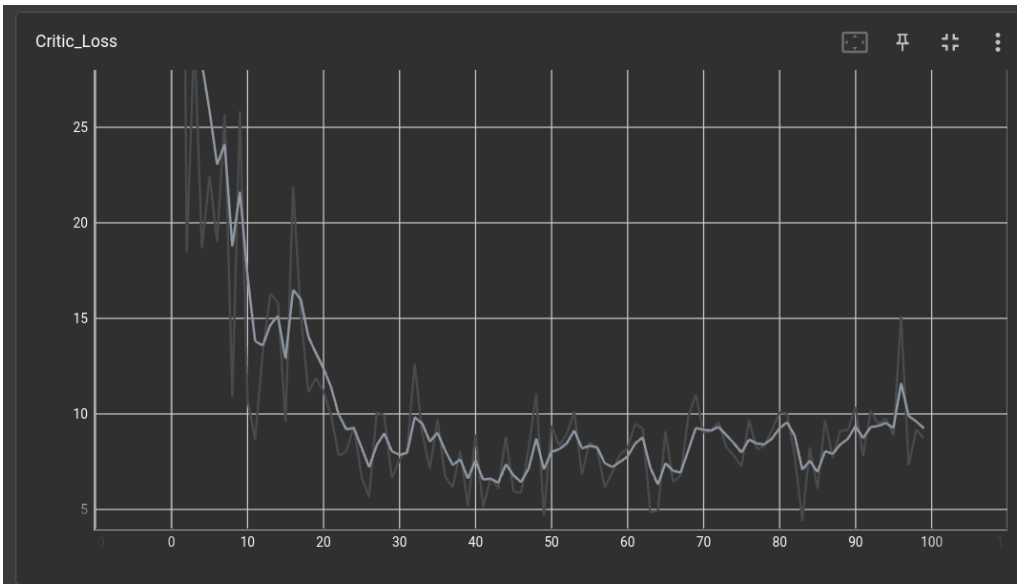

Figure 3: Learning curve for the baseline loss.

- Plot a learning curve for the eval return. You should expect to achieve an average return over 300 for the baselined version.

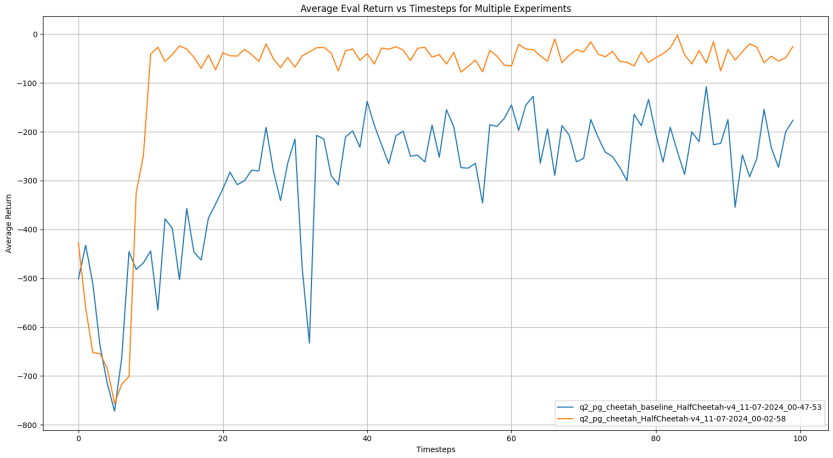

Figure 4: Learning curve for the eval return with baseline.

- Run another experiment with a decreased number of baseline gradient steps (`-bgs`) and/or baseline learning rate (`-blr`). How does this affect (a) the baseline learning curve and (b) the performance of the policy?
  One thing that seemed problematic with this experiment was that it did not converge. I tried implementing various tricks such as gradient clipping, xavier initialization of the neural networks, different learning rates, optimizers and also varying network sizes. I also tried normalizing advantages. It initially

seemed to be a problem with all continuous environments. Upon introducing an entropy component to the policy loss, I was able to fix it for the simpler continous environment such as Inverted Pendulum. However, this one did not get resolved. I have attached the plots of the best performing experiments here nonetheless.

- **Optional:** Add `-na` back to see how much it improves things. Also, set `video_log_freq 10`, then open TensorBoard and go to the "Images" tab to see some videos of your HalfCheetah walking along!

6 Generalized Advantage Estimation

- Provide a single plot with the learning curves for the `LunarLander-v2` experiments that you tried. Describe in words how λ affected task performance. The run with the best performance should achieve an average score close to 200 (180+).

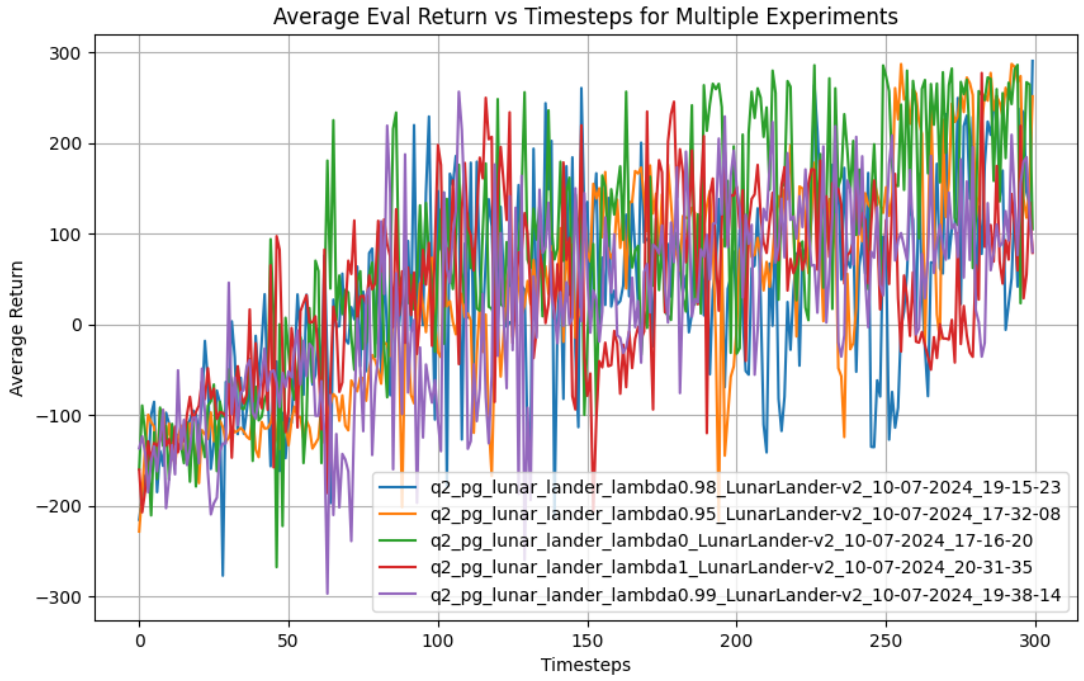

Figure 5: Learning curves for `LunarLander-v2` with different λ values.

The plot in Figure 5 shows the learning curves for the `LunarLander-v2` environment with different values of λ. From the plot, we observe that while λ values of 0.99 and 0.98 achieve the best performance with average scores close to 200, $\lambda = 1$ performs worse. This suggests that using a slightly lower value than 1 helps stabilize learning and improve performance by reducing the variance introduced by using the full trajectory of rewards.

- Consider the parameter λ. What does $\lambda = 0$ correspond to? What about $\lambda = 1$? Relate this to the task performance in `LunarLander-v2` in one or two sentences.

  The parameter λ in Generalized Advantage Estimation (GAE) controls the bias-variance trade-off in the advantage estimates. A value of $\lambda = 0$ corresponds to using only the immediate rewards (high bias, low variance), while $\lambda = 1$ corresponds to using the full trajectory of rewards (low bias, high variance). In the `LunarLander-v2` environment, values of λ close to but slightly less than 1 (such as 0.99 and 0.98) provide the best task performance by balancing the trade-off, whereas $\lambda = 1$ introduces too much variance, resulting in worse performance.

7 Hyperparameter Tuning

1. Provide a set of hyperparameters that achieve high return on `InvertedPendulum-v4` in as few environment steps as possible.
2. Show learning curves for the average returns with your hyperparameters and with the default settings, with environment steps on the x-axis. Returns should be averaged over 5 seeds.

- **Environment Name:** InvertedPendulum-v4
- **Number of Iterations:** 100
- **Experiment Name:** pendulum_default_s{seed}
- **Use Reward to Go:** True
- **Normalize Advantages:** True
- **Batch Size:** 5000
- **GAE Lambda:** 0.98
- **Learning Rate:** 0.005

The above configuration was able to attain a reward of 1000 within 48 iterations.

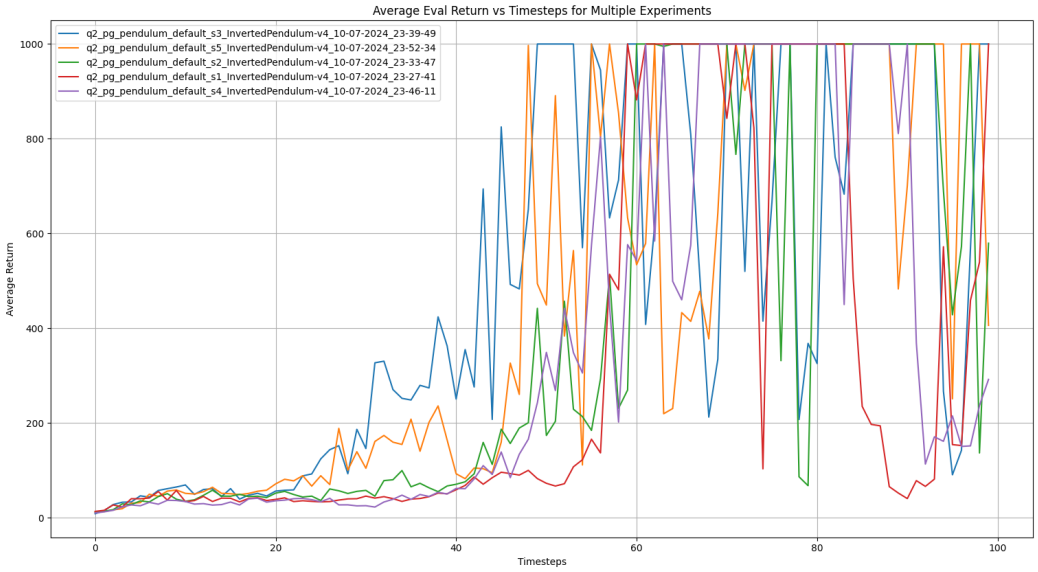

Figure 6: Learning curve with default hyperparameters.

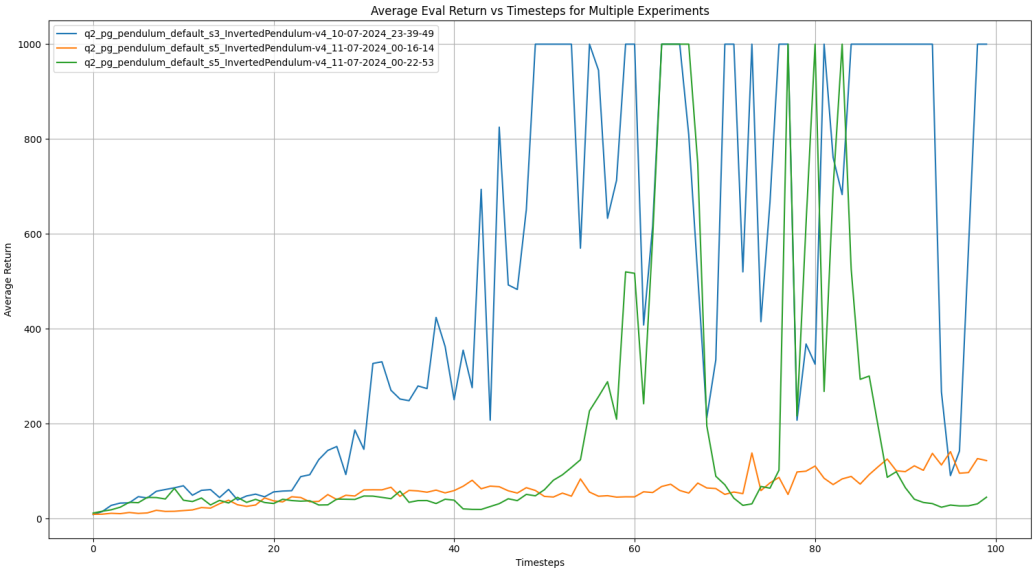

Figure 7: Learning curve with tuned hyperparameters.

8 (Extra Credit) Humanoid

1. Plot a learning curve for the Humanoid-v4 environment. You should expect to achieve an average return of at least 600 by the end of training. Discuss what changes, if any, you made to complete this problem (for example: optimizations to the original code, hyperparameter changes, algorithmic changes).