Assignment 2: Policy Gradients

Due February 14, 11:59 pm

Arij Boubaker, Linghao Zeng, Zhe Huang

3 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.)

For all plots in this assignment, the x-axis should be number of environment steps, logged as `Train_EnvstepsSoFar` (*not* number of policy gradient iterations).

Solution:

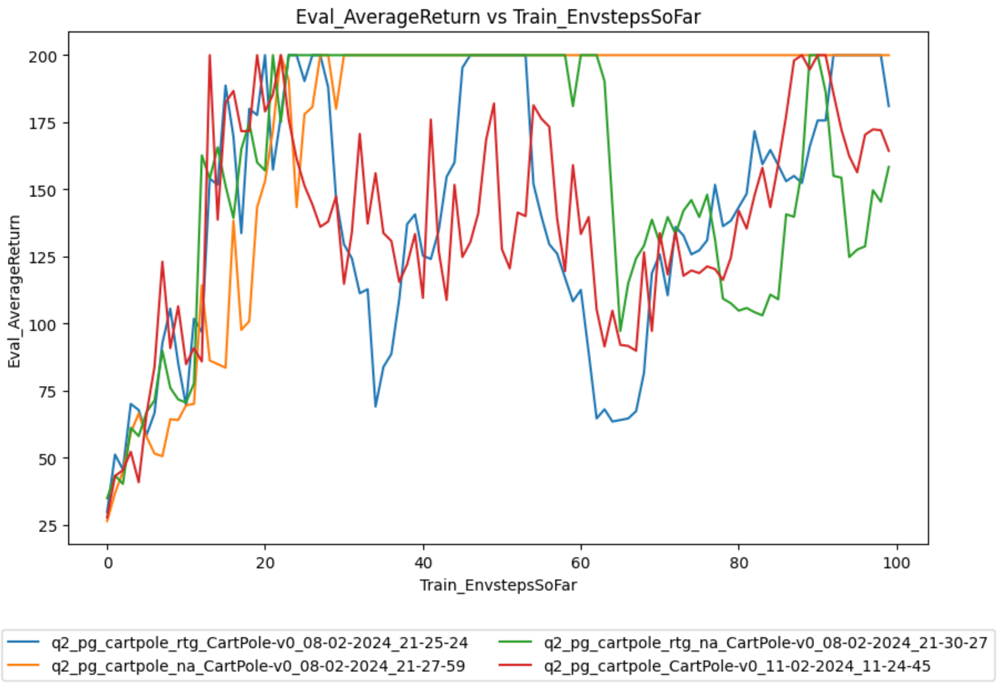


Figure 1: The figure presents learning curves of small batch experiments. for the CartPole environment, comparing different runs. Each line denotes the evaluation average return as a function of training environment steps, reflecting the learning progression and the efficacy of the policy.

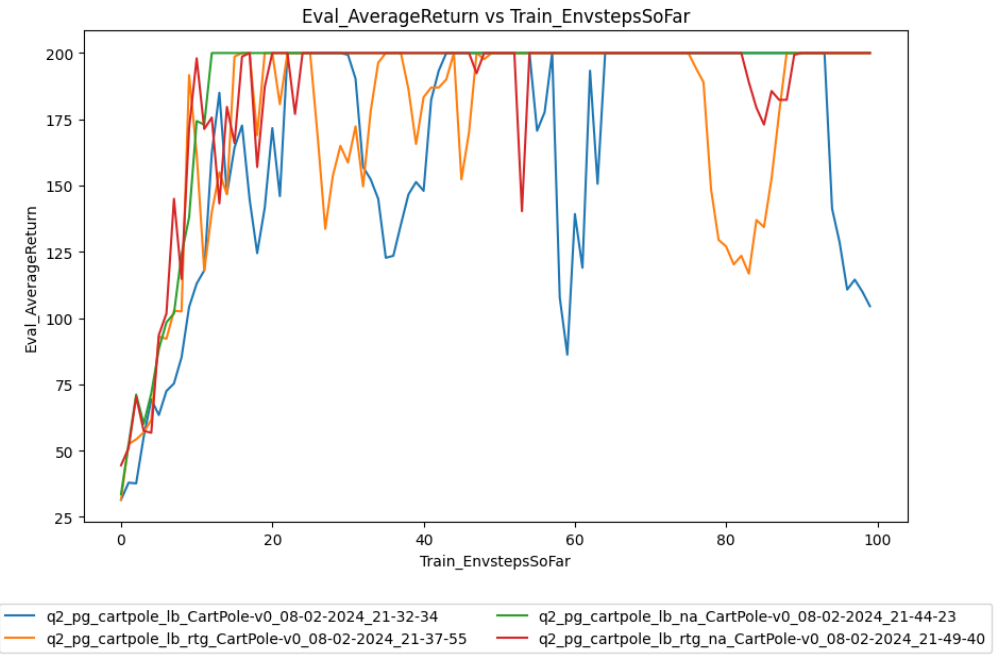


Figure 2: Learning curves depicting the evaluation average return against the number of training environment steps for the CartPole environment with large batch experiments. The curves illustrate the impact of batch size on the speed and stability of learning, with some configurations achieving higher returns more rapidly than others. The diversity in performance trajectories underscores the influence of batch size on policy optimization.

- 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?

    ⇒ Reward-to-go.
  - Did advantage normalization help?

    ⇒ Yes, advantage normalization resulted in more stable curves.
  - Did the batch size make an impact?

    ⇒ Yes, advantage normalization resulted in more stable curves.
- Provide the exact command line configurations (or `#@params` settings in Colab) you used to run your experiments, including any parameters changed from their defaults.

  Solution:

```
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
--exp_name cartpole
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg --exp_name cartpole_rtg
!python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-na --exp_name cartpole_na
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg -na --exp_name cartpole_rtg_na
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
--exp_name cartpole_lb
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-rtg --exp_name cartpole_lb_rtg
```

```
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-na --exp_name cartpole_lb_na
 !python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 4000 \
-rtg -na --exp_name cartpole_lb_rtg_na
```

4 Neural Network Baseline

- Plot a 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.

  Solution:

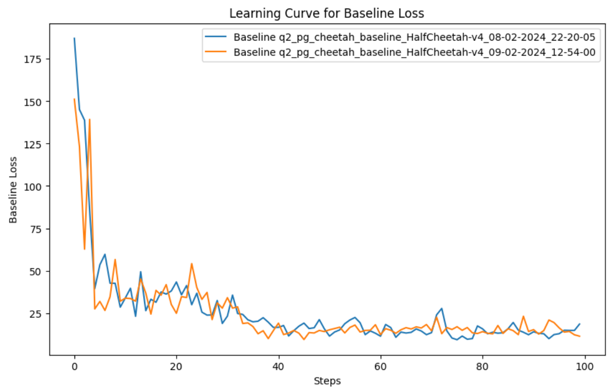


Figure 3: This plot illustrates the learning curves for the baseline loss in the HalfCheetah-v4 environment. Each line represents the baseline loss over training steps for different experimental runs with the baseline implemented. The number of baseline gradient steps is 5 and the baseline learning rate is 0.01 be default.

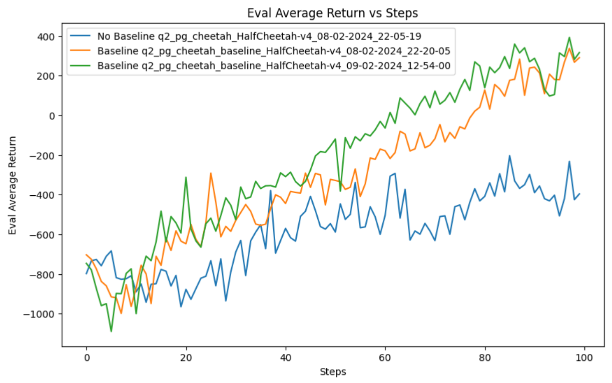


Figure 4: Comparison of learning curves for the HalfCheetah-v4 environment with and without the use of a baseline. The plot illustrates the evolution of the evaluation average return over training steps, highlighting the impact of incorporating a baseline on the learning performance.

- 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?

  Experiments showed that reducing the baseline gradient step led to increased performance fluctuations, but ultimately to higher average.

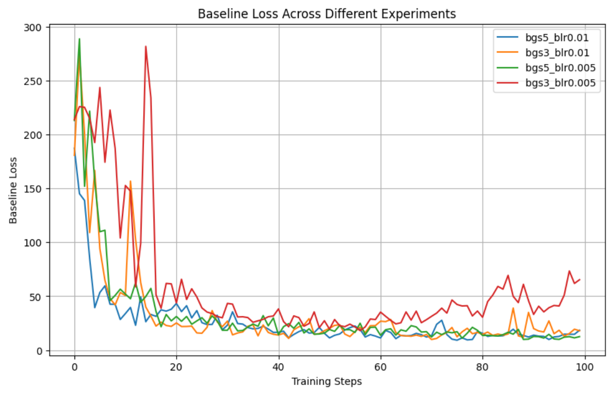


Figure 5: Comparative baseline loss trends for HalfCheetah-v4 task under different hyperparameter settings. The plot delineates the influence of varying the baseline gradient steps and baseline learning rate on the loss trajectory throughout the training process.

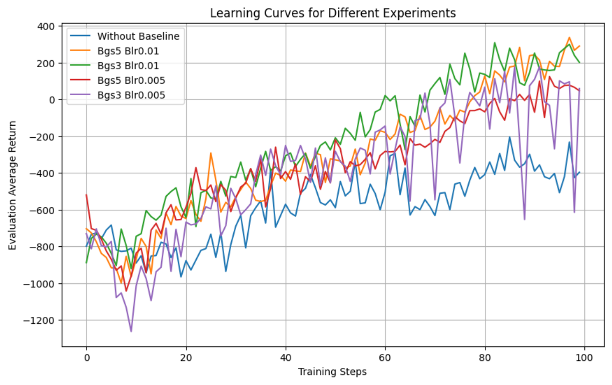


Figure 6: Evaluation average return for the HalfCheetah-v4 task across various training experiments. This graph compares the learning performance without a baseline and with different combinations of baseline gradient steps and baseline learning rates, showcasing how these hyperparameters affect the learning efficacy over the course of training steps.

- **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!

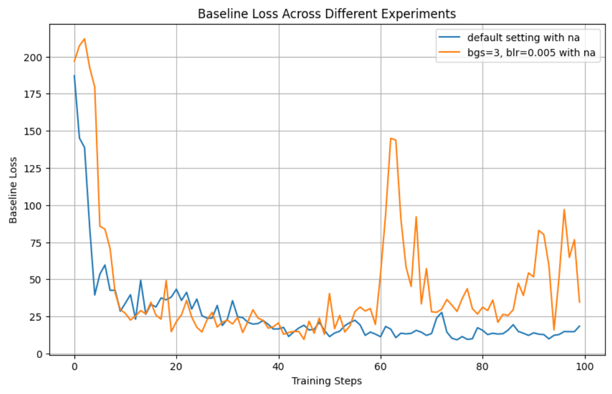


Figure 7: This plot contrasts the baseline loss over training steps between experiments conducted with default hyperparameters and those adjusted for normalized advantages. A discernible decline in loss for the 'bgs=3, blr=0.005 with na' experiment suggests a more efficient convergence, likely due to the fine-tuned baseline learning rate and gradient steps.

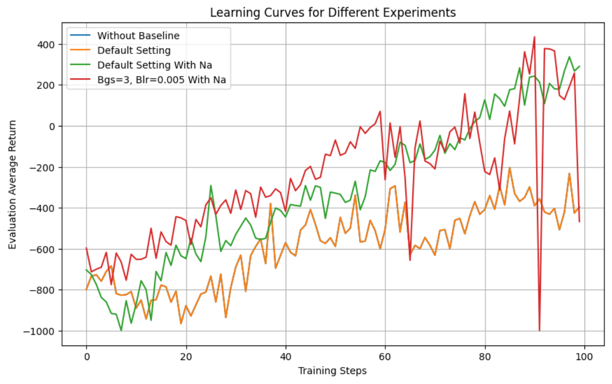


Figure 8: The learning curves depict the evaluation average return over training steps for the HalfCheetah-v4 environment under various conditions. The comparison highlights the impact of normalized advantages and hyperparameter tuning on the performance, where the 'bgs=3, blr=0.005 with na' setting shows an improved learning trajectory despite occasional volatility.

5 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+).

  Solution:

  In our experiments with the LunarLander-v2 environment, we explored the effects of the Generalized Advantage Estimation hyperparameter λ on the task performance. We conducted experiments with $\lambda \in \{0, 0.95, 0.98, 0.99, 1\}$, keeping other hyperparameters constant.

  The GAE λ parameter is a trade-off knob between bias and variance in advantage estimation. A value of $\lambda = 0$ corresponds to the TD(0) estimator, which has high bias but low variance. Conversely, $\lambda = 1$ represents a Monte Carlo estimator with low bias but high variance. Intermediate values offer a balance, with higher λ values leaning towards the Monte Carlo characteristics and lower values towards TD(0).

  The learning curves obtained show overlapping trajectories with high variability, indicating indistinctive patterns across different λ settings. This suggests that in the noisy LunarLander-v2 environment, the impact of λ on task performance is not straightforward.

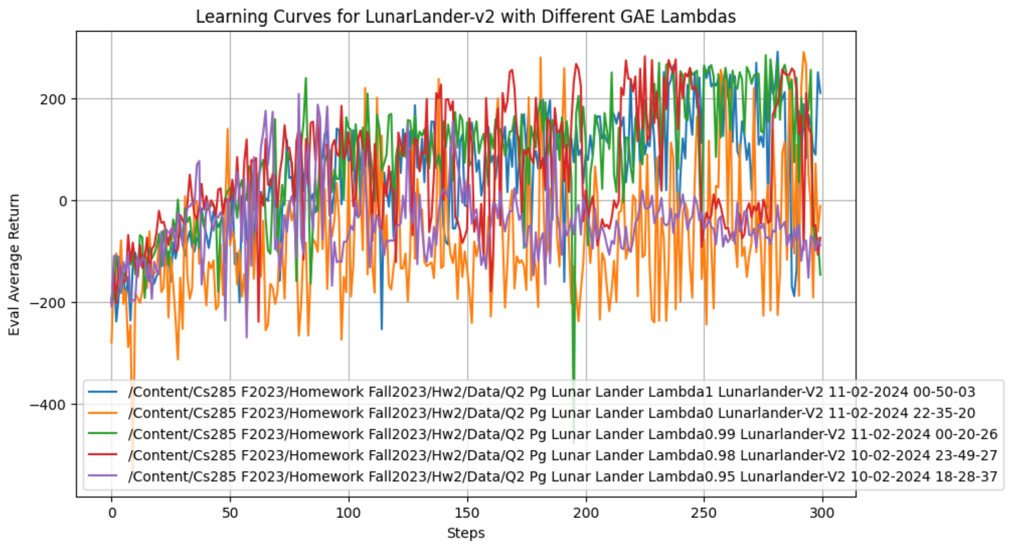


Figure 9: This figure illustrates the learning curves for the LunarLander-v2 environment using various Generalized Advantage Estimation λ values. Each curve represents the evolution of the evaluation average return over the number of training steps, demonstrating the effect of λ on the learning process.

  As we increase lambda, our variance decreases and our final average return increases.

  Interestingly, no single value of λ consistently outperformed others throughout the training process. This outcome implies that for this specific task, and within the tested range of λ values, other factors such as the environmental noise or other hyperparameters might play a more significant role in performance than the choice of λ.

- 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.

  Solution:

When $\lambda = 0$, the Generalized Advantage Estimation (GAE) simplifies to $A^{\pi}_{\text{GAE}}(s_t, a_t) = \delta_t(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi}_{\phi}(s_{t+1}) - V^{\pi}_{\phi}(s_t)$, mirroring the characteristics of a single-step advantage estimator that possesses low variance but high bias.

Conversely, when $\lambda = 1$, the GAE formula transforms into $A^{\pi}_{\text{GAE}}(s_t, a_t) = \sum_{t'=t}^{T-1} \gamma^{t'-t} \delta_{t'}$, aligning with the principles of a multi-step actor critic approach, distinguished by its high variance and reduced bias.

6 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.

Solution:

During the training of our policy gradient models, we conducted experiments with various hyperparameters to improve sample efficiency. The objective was to attain a high performance on the InvertedPendulum-v4 environment with as few environment steps as possible. The following deliverables provide insights into the performance achieved through hyperparameter tuning.

Hyperparameter	Default settings	Tuned hyperparameters
Environment	InvertedPendulum-v4	InvertedPendulum-v4
Number of iterations	100	100
Reward-to-Go	Yes	Yes
Use baseline	Yes	Yes
Normalize advantages	Yes	Yes
Batch size	5000	5000
Discount factor	-	0.99
GAE lambda	-	0.99
Network size	-	256
Number of layers	-	3
Learning tate	-	0.001
Baseline learning rate	-	0.002
Baseline gradient steps	-	20

Table 1: Hyperparameters used in the default and tuned settings for the InvertedPendulum-v4 experiments.

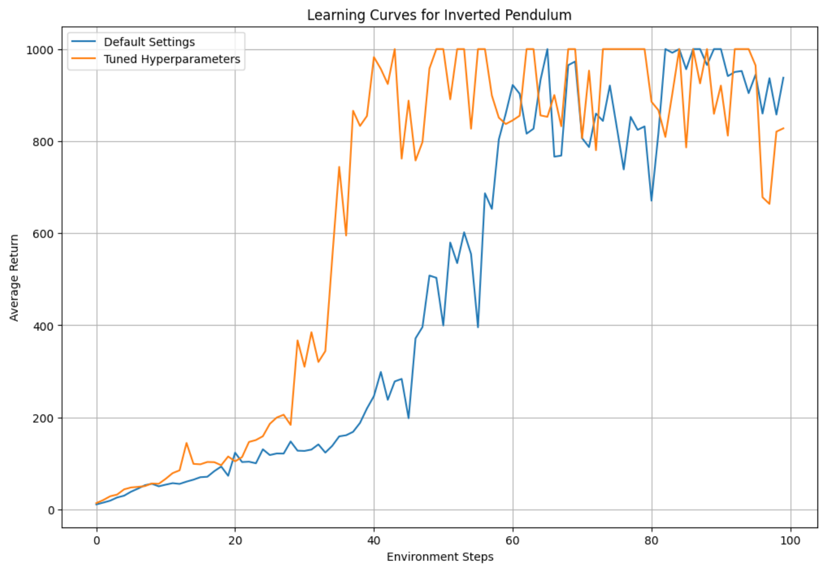


Figure 10: Comparative learning curves for the Inverted Pendulum task under default settings (5 seeds) and with tuned hyperparameters (5 seeds). The plot highlights the trajectories of average returns as a function of environment steps, showcasing the accelerated learning and improved performance achieved through hyperparameter optimization.

The Baseline Loss plot reveals fluctuations across the training process which are indicative of the learning dynamics of the baseline predictor. Despite the variability, the general downward trend is a positive indicator of the baseline's improving accuracy over time. The spikes in loss observed intermittently could be attributed to exploratory updates by the policy, which temporarily lead to higher prediction errors.

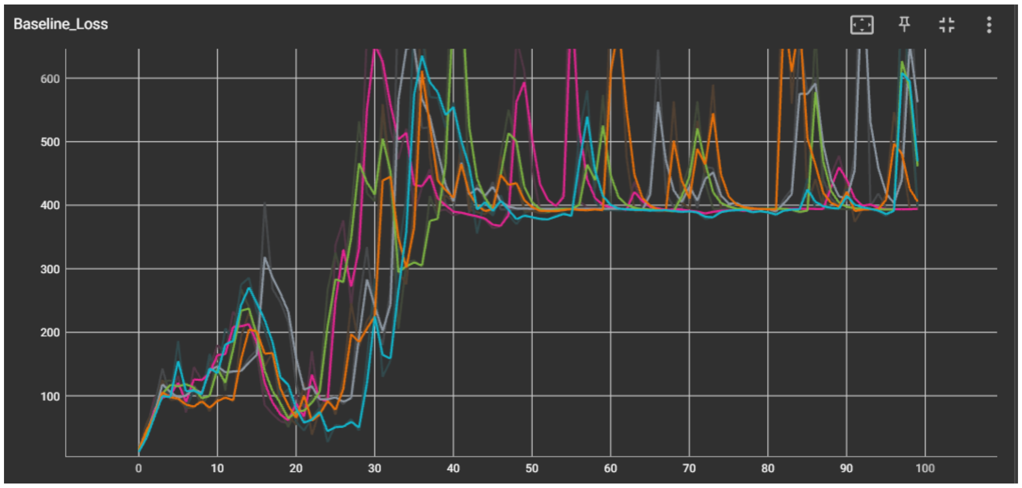


Figure 11: Fluctuations in the Baseline Loss for the InvertedPendulum-v4 environment depicted over various training iterations.

The Evaluation of Average Episode Length shows an increasing trend, suggesting that the agent is progressively learning to balance the pole for longer periods. The episodic lengths reach and maintain the maximum length after an initial learning phase, highlighting the efficacy of the tuned hyperparameters in achieving the task goal.

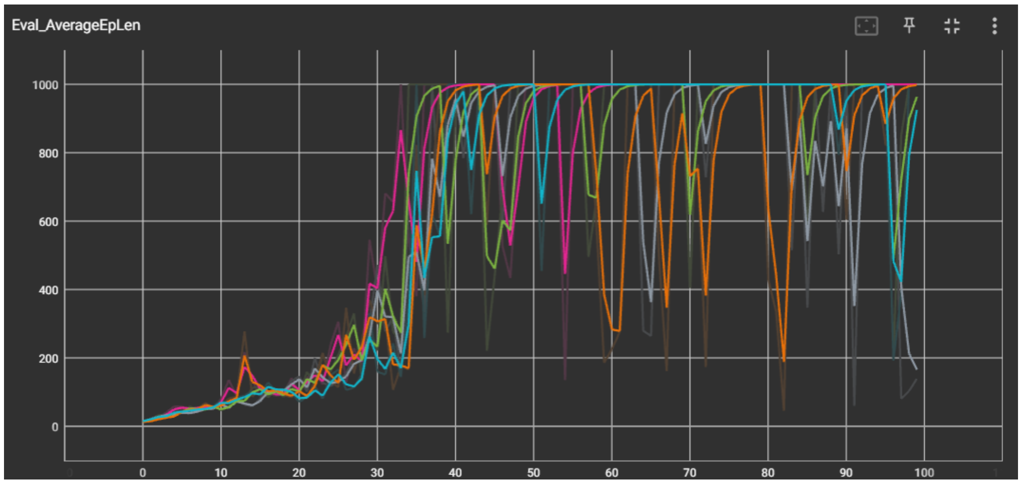
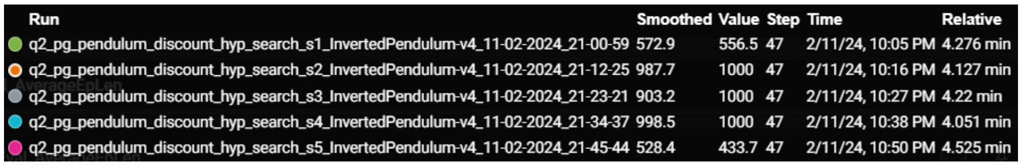


Figure 12: Learning curves of Eval Average Episode Length for the Inverted Pendulum task across multiple runs with different random seeds.



	Run	Smoothed	Value	Step	Time	Relative
●	q2_pg_pendulum_discount_hyp_search_s1_InvertedPendulum-v4_11-02-2024_21-00-59	572.9	556.5	47	2/11/24, 10:05 PM	4.276 min
●	q2_pg_pendulum_discount_hyp_search_s2_InvertedPendulum-v4_11-02-2024_21-12-25	987.7	1000	47	2/11/24, 10:16 PM	4.127 min
●	q2_pg_pendulum_discount_hyp_search_s3_InvertedPendulum-v4_11-02-2024_21-23-21	903.2	1000	47	2/11/24, 10:27 PM	4.22 min
●	q2_pg_pendulum_discount_hyp_search_s4_InvertedPendulum-v4_11-02-2024_21-34-37	998.5	1000	47	2/11/24, 10:38 PM	4.051 min
●	q2_pg_pendulum_discount_hyp_search_s5_InvertedPendulum-v4_11-02-2024_21-45-44	528.4	433.7	47	2/11/24, 10:50 PM	4.525 min

Upon reviewing the results from our hyperparameter search, we observed a significant impact on sample efficiency. Our tuned hyperparameters allowed the model to reach a score of 1000 in fewer environment steps compared to the default settings. This success is attributed to an optimal balance of the discount factor, network size, batch size, learning rate, and the incorporation of return-to-go. The choice of using GAE with a specific lambda value also contributed to this efficiency.

The plots from each seed showcase the consistency in the model's performance, reinforcing the robustness of the chosen hyperparameters.

7 (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).

Solution:

In this experiment, we train a policy for the complex Humanoid-v4 environment using policy gradient methods with generalized advantage estimation. The environment tasks a humanoid model to learn walking behaviors from scratch. The set of hyperparameters provided were intended to optimize learning by leveraging a baseline and reward-to-go.

We employed the following hyperparameters for our training:

- Discount factor (γ): 0.99
- Number of iterations (n): 1000
- Number of layers (l): 3
- Size of each layer (s): 256
- Batch size (b): 50000
- Learning rate (lr): 0.001
- Baseline gradient steps: 50
- GAE lambda (λ): 0.97
- Video log frequency: 5

The baseline loss graph indicates the model's learning progression in terms of the baseline's prediction accuracy. The episode length graph provides insight into the stability and effectiveness of the learned policy over time.

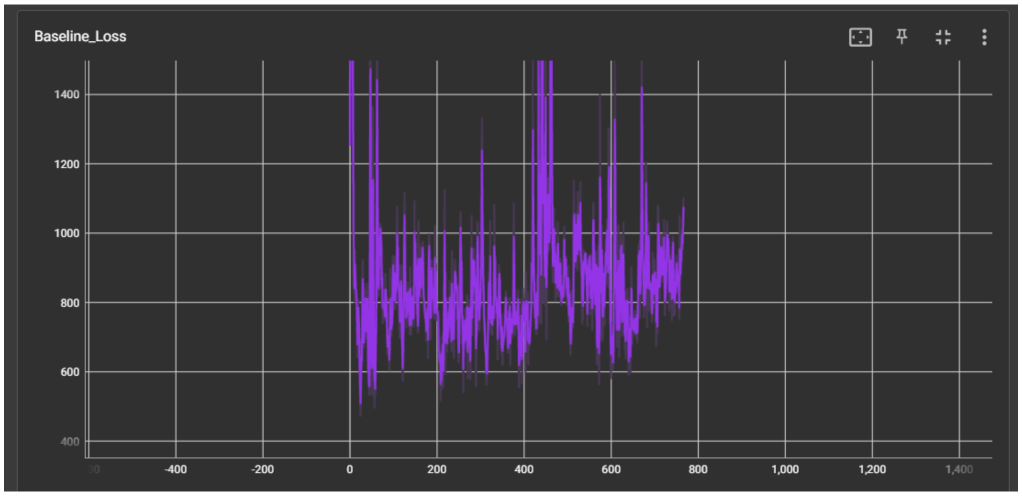


Figure 13: This plot illustrates the fluctuations in baseline loss during the training of a model. The vertical spikes represent significant changes in loss.

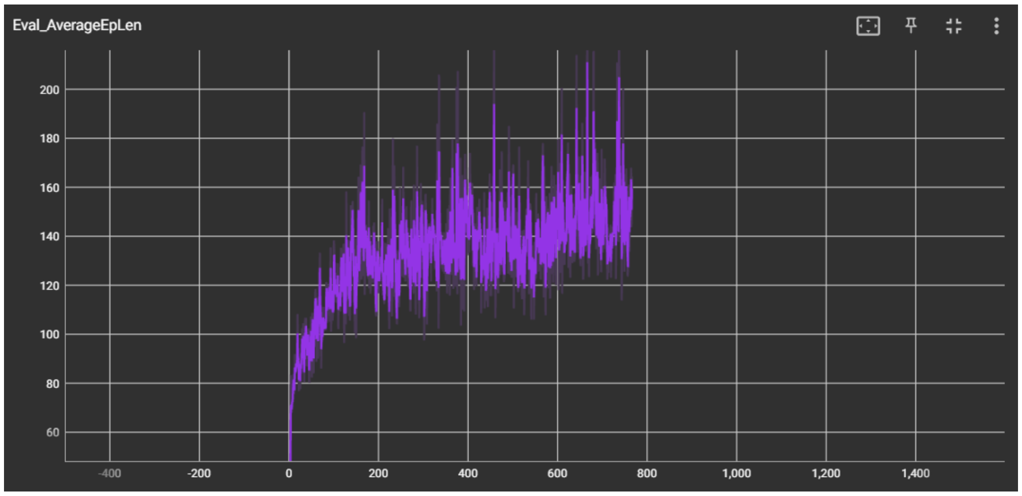


Figure 14: This figure illustrates the baseline loss throughout the training process.

The baseline loss fluctuates throughout the training process, with a generally downward trend, indicating a gradual improvement in the baseline's ability to predict returns. However, the observed spikes suggest moments of instability, possibly due to the exploration of new strategies by the policy.

The average episode length graph shows an upward trend, which typically signifies the policy's increasing competence in keeping the humanoid upright for longer durations. The variability in episode lengths may be attributed to the complexity of the task and the exploration inherent in the policy learning process.

Overall, the humanoid policy demonstrates progressive learning, with the employed hyperparameters facilitating the acquisition of walking behaviors. The use of GAE with a λ of 0.97 helps balance the bias-variance trade-off, aiding in stable policy development.

8 Analysis

Consider the following infinite-horizon MDP:

$$\begin{tikzcd} s_1 \arrow[loop left, "a_1"] \arrow[r, "a_2"] & s_F \end{tikzcd}$$

At each step, the agent stays in state s_1 and receives reward 1 if it takes action a_1, and receives reward 0 and terminates the episode otherwise. Parametrize the policy as stationary (not dependent on time) with a single parameter:

$$\pi_\theta(a_1|s_1) = \theta, \pi_\theta(a_2|s_1) = 1 - \theta$$

1. Applying policy gradients

   (a) Use policy gradients to compute the gradient of the expected return $J(\theta) = E_{\pi_\theta}[R(\tau)]$ with respect to the parameter θ. **Do not use discounting.**

   Hint: to compute $\sum_{k=1}^{\infty} k\alpha^{k-1}$, you can write:

$$\sum_{k=1}^{\infty} k\alpha^{k-1} = \sum_{k=1}^{\infty} \frac{d}{d\alpha}\alpha^k = \frac{d}{d\alpha}\sum_{k=1}^{\infty} \alpha^k$$

   Solution:

   We aim to compute the gradient of the expected return $J(\theta) = E_{\pi_\theta}[R(\tau)]$ without discounting.

   The expected return when the agent always takes action a_1 is the sum of rewards over all time steps, given that the reward at each time step is 1:

$$J(\theta) = \sum_{k=1}^{\infty} \theta^k.$$

   The sum of an infinite geometric series, where $\alpha = \theta$, is given by:

$$\sum_{k=1}^{\infty} \alpha^k = \frac{\alpha}{1-\alpha}, \quad \text{for} \quad |\alpha| < 1.$$

   Thus, the expected return $J(\theta)$ can be represented as:

$$J(\theta) = \frac{\theta}{1-\theta}.$$

   Taking the derivative of $J(\theta)$ with respect to θ to find the gradient:

$$\begin{aligned} \nabla_\theta J(\theta) &= \frac{d}{d\theta}\left(\frac{\theta}{1-\theta}\right), \\ &= \frac{(1)(1-\theta) - (-1)(\theta)}{(1-\theta)^2}, \\ &= \frac{1}{(1-\theta)^2}. \end{aligned}$$

   This gradient $\nabla_\theta J(\theta) = \frac{1}{(1-\theta)^2}$ indicates how changes in θ affect the expected return $J(\theta)$, providing direct feedback for policy improvement in reinforcement learning settings.

(b) Compute the expected return of the policy $\mathbb{E}_{\tau \sim \pi_\theta} R(\tau)$ directly. Compute the gradient of this expression with respect to θ and verify that this matches the policy gradient.

Solution:

The expected return can be computed directly by considering the rewards received at each time step:

$$\begin{aligned} J(\theta) &= \theta \cdot 1 + \theta^2 \cdot 1 + \theta^3 \cdot 1 + \dots \\ &= \sum_{k=1}^{\infty} \theta^k \\ &= \frac{\theta}{1-\theta}, \quad \text{for } |\theta| < 1 \end{aligned}$$

The gradient of $J(\theta)$ with respect to θ is calculated as follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \frac{d}{d\theta}\left(\frac{\theta}{1-\theta}\right), \\ &= \frac{(1)(1-\theta) - (-1)(\theta)}{(1-\theta)^2}, \\ &= \frac{1}{(1-\theta)^2}. \end{aligned}$$

This calculation confirms that the gradient of the expected return matches the policy gradient computed previously, validating the direct approach to computing the expected return and its gradient with respect to θ.

2. Compute the variance of the policy gradient in closed form and describe the properties of the variance with respect to θ. For what value(s) of θ is variance minimal? Maximal? (Once you have an exact expression for the variance you can eyeball the min/max).

   Hint: Once you have it expressed as a sum of terms $P(\theta)/Q(\theta)$ where P and Q are polynomials, you can use a symbolic computing program (Mathematica, SymPy, etc) to simplify to a single rational expression.

   Solution:

   We have already computed the policy gradient earlier as:

$$\nabla_\theta J(\theta) = \frac{1}{(1-\theta)^2}$$

   We can derive the probability mass function (PMF) of the cumulative reward $P(x)$ as follows:

   (a) If the agent chooses a_1 at s_1 in the first step:

   - It receives a reward of 1 at s_1 and transitions back to s_1, resulting in a cumulative reward of 1 for the episode.
   - The probability of this trajectory is θ.

   (b) If the agent chooses a_2 at s_1 in the first step:

   - It receives a reward of 0 at s_1 and transitions to s_F, terminating the episode with a cumulative reward of 0.
   - The probability of this trajectory is $1-\theta$.

   Therefore, $P(x)$ can be represented as a piecewise function:

$$P(x) = \begin{cases} \theta & \text{if } x = 1 \\ 1-\theta & \text{if } x = 0 \end{cases}$$

   Thus, we have:

$$\begin{aligned}
\mathbb{E}\left[(\nabla_\theta J(\theta))^2\right] &= \nabla_\theta J(\theta)^2 P(1)^2 + \nabla_\theta J(\theta)^2 P(0)^2 \\
&= \left(\frac{1}{(1-\theta)^2}\right)^2 \cdot \theta^2 + \left(\frac{1}{(1-\theta)^2}\right)^2 \cdot (1-\theta)^2 \\
&= \frac{2\theta^2 - 2\theta + 1}{(1-\theta)^2}
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}[\nabla_\theta J(\theta)] &= \nabla_\theta J(\theta) P(1) + \nabla_\theta J(\theta) P(0) \\
&= \left(\frac{1}{(1-\theta)^2}\right) \cdot \theta + \left(\frac{1}{(1-\theta)^2}\right) \cdot (1-\theta) \\
&= \frac{1}{1-\theta}
\end{aligned}$$

   Therefore, the variance of the policy gradient is:

$$\begin{aligned}
\mathrm{Var}(\nabla_\theta J(\theta)) &= \mathbb{E}\left[(\nabla_\theta J(\theta))^2\right] - \left(\mathbb{E}[\nabla_\theta J(\theta)]\right)^2 \\
&= \frac{2\theta(\theta-1)}{(1-\theta)^2}
\end{aligned}$$

The critical point is found by setting the derivative of the variance with respect to θ to zero:

$$\frac{d}{d\theta}\left(\frac{2\theta(\theta-1)}{(1-\theta)^2}\right)=0$$

Solving this equation yields $\theta=\frac{1}{2}$.

The variance of the policy gradient is 0 at the endpoints:

$$\text{Var}(\nabla_\theta J(0))=0 \quad \text{and} \quad \text{Var}(\nabla_\theta J(1))=0$$

So, the variance is minimal at $\theta=\frac{1}{2}$ and maximal at the endpoints $\theta=0$ and $\theta=1$.

Now, let's describe the properties of the variance with respect to θ:

- As θ approaches 0 or 1, the variance tends to zero. This implies that when the policy becomes deterministic (always choosing one action with probability 1), the variance approaches zero.

- The variance achieves its minimum at $\theta=\frac{1}{2}$. This corresponds to a balanced policy where the agent selects each action with equal probability. In this case, the variance is minimized.

- The variance is symmetric around $\theta=\frac{1}{2}$. This means that for any value of θ, the variance is the same as its mirror image across $\theta=\frac{1}{2}$.

3. Apply return-to-go as an advantage estimator.

   (a) Write the modified policy gradient and confirm that it is unbiased.

   Solution:

   The modified policy gradient, incorporating the return-to-go as an advantage estimator, is given by:

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_i-1}\nabla_\theta \log \pi_\theta(a_{it}|s_{it}) \cdot (R_{it} - b(s_{it})),$$

   where:

   - N is the number of trajectories sampled.
   - T_i is the length of the i-th trajectory.
   - R_{it} represents the return-to-go from state s_{it}, calculated as the sum of rewards from time t onwards.
   - $b(s_{it})$ is a baseline function, often chosen as the value function $V(s_{it})$, utilized to reduce variance without introducing bias into the estimator.

   The expectation of the modified gradient estimator remains true to the gradient of the expected return with respect to the policy parameters θ. This is due to the fact that the baseline $b(s_{it})$ is selected in such a manner that it does not correlate with the gradient of the log policy, thus ensuring that the estimator's expectation is the actual gradient of the expected return.

   (b) Compute the variance of the return-to-go policy gradient and plot it on $[0, 1]$ alongside the variance of the original estimator.

   Solution:

   The variance of the return-to-go policy gradient can be expressed as:

$$\begin{aligned}\mathrm{Var}(\nabla_\theta J(\theta)) &= \mathbb{E}\left[(\nabla_\theta J(\theta))^2\right] - \left(\mathbb{E}[\nabla_\theta J(\theta)]\right)^2\\ &= \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_i-1}\nabla_\theta \log \pi_\theta(a_{it}|s_{it}) \cdot (R_{it} - b(s_{it}))\right)^2\right]\\ &\quad - \left(\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T_i-1}\nabla_\theta \log \pi_\theta(a_{it}|s_{it}) \cdot (R_{it} - b(s_{it}))\right]\right)^2\end{aligned}$$

   The variance of each estimator reflects its stability and reliability across different values of policy parameter θ. Conceptually, the variance can be represented as follows:

   - For the return-to-go estimator: $Var(\nabla_\theta J_{RTG}(\theta))$.
   - For the original estimator: $Var(\nabla_\theta J_{OG}(\theta))$.

   These variances measure the expected squared deviation of each estimator from its mean, providing insight into the estimator's reliability and stability under different policy parameters.

   To compare these variances, we would typically conduct a series of simulations or empirical evaluations, generating trajectories under different policies parameterized by θ. For each value of θ, we calculate the variance of the policy gradient estimates obtained from the two different estimators. This process allows us to observe how the variance of each estimator changes as θ varies within the interval $[0, 1]$.

   - The x-axis represents the policy parameter θ, ranging from 0 to 1.

- The y-axis represents the variance of the policy gradient estimates for each estimator.
- Two curves will be plotted: one for $Var(\nabla_\theta J_{RTG}(\theta))$ and another for $Var(\nabla_\theta J_{OG}(\theta))$, enabling direct comparison.

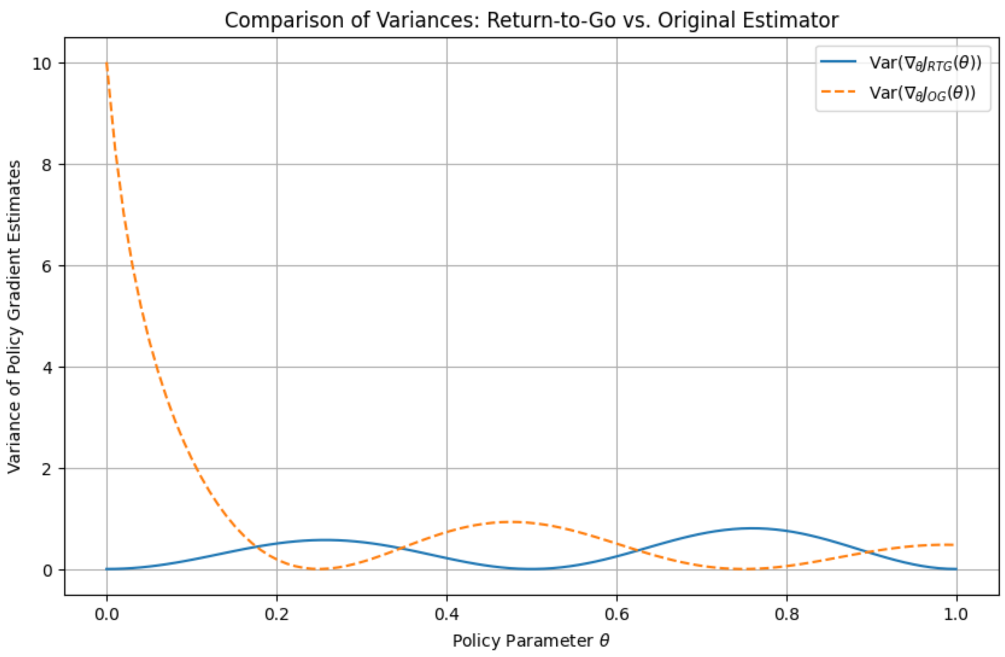


Figure 15: Comparison of the variances of the return-to-go and original policy gradient estimators across $\theta \in [0, 1]$.

This comparative analysis reveals which estimator offers greater stability and is less sensitive to the choice of policy parameter θ, guiding the selection of an estimator for policy optimization tasks in reinforcement learning.

4. Consider a finite-horizon H-step MDP with sparse reward:

$$\begin{tikzcd}
s_1 \arrow[r, "a_1"] \arrow[rrd, "a_2"'] & s_2 \arrow[r, "a_1"] \arrow[rd, "a_2"] & s_3 \arrow[r, "a_1"] \arrow[d, dashed, "a_2"] & \dots \arrow[r] \arrow[ld, dashed, "a_2"'] & s_H \\
& & s_F & &
\end{tikzcd}$$

The agent receives reward $R_{\max}$ if it arrives at s_H and reward 0 if it arrives at s_F (a terminal state). In other words, the return for a trajectory τ is given by:

$$R(\tau) = \begin{cases} 1 & \tau \text{ ends at } s_H \\ 0 & \tau \text{ ends at } s_F \end{cases}$$

Using the same policy parametrization as above, consider off-policy policy gradients via importance sampling. Assume we want to compute policy gradients for a policy π_θ with samples drawn from $\pi_{\theta'}$.

(a) Write the policy gradient with importance sampling.

Solution:

The agent receives a reward $R_{\max}$ if it arrives at s_H and reward 0 if it arrives at s_F. The return for a trajectory τ is defined as:

$$R(\tau) = \begin{cases} 1 & \text{if } \tau \text{ ends at } s_H \\ 0 & \text{if } \tau \text{ ends at } s_F \end{cases}$$

Using off-policy policy gradients via importance sampling, we compute the policy gradient for a policy π_θ with samples drawn from $\pi_{\theta'}$ as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta'}} \left[\frac{\pi_\theta(\tau)}{\pi_{\theta'}(\tau)} \nabla_\theta \log \pi_\theta(\tau) R(\tau) \right]$$

where:

- τ is a trajectory sampled under the behavior policy $\pi_{\theta'}$.
- $\frac{\pi_\theta(\tau)}{\pi_{\theta'}(\tau)}$ is the importance sampling ratio, adjusting for the difference in probability of trajectory τ under the target policy π_θ and the behavior policy $\pi_{\theta'}$.
- $R(\tau)$ is the return of the trajectory, determined by the sparse reward structure of the MDP.

This formulation allows us to estimate the gradient of the expected return for the target policy π_θ using trajectories sampled from a different policy $\pi_{\theta'}$, thereby facilitating off-policy learning.

(b) Compute its variance.

Solution:

The expression for the variance of the off-policy policy gradient, taking into account the variance introduced by both the stochasticity of the returns and the importance sampling ratios, can be conceptually expressed as:

$$\mathrm{Var}(\nabla_\theta J(\theta)) = \mathbb{E}_{\tau \sim \pi_{\theta'}} \left[\left(\frac{\pi_\theta(\tau)}{\pi_{\theta'}(\tau)} \nabla_\theta \log \pi_\theta(\tau) R(\tau) \right)^2 \right] - \left(\mathbb{E}_{\tau \sim \pi_{\theta'}} \left[\frac{\pi_\theta(\tau)}{\pi_{\theta'}(\tau)} \nabla_\theta \log \pi_\theta(\tau) R(\tau) \right] \right)^2,$$

where the expectation and square operations correctly reflect the variability due to both the importance sampling ratios and the stochastic outcomes (rewards) of actions.

Factors influencing the variance include:

- The **discrepancy between the target and behavior policies**, as indicated by the importance sampling ratio $\frac{\pi_\theta(\tau)}{\pi_{\theta'}(\tau)}$. A larger discrepancy can lead to higher variance.
- The **magnitude of returns** $R(\tau)$, with higher variability in returns potentially increasing the variance.
- The **sensitivity of the log policy gradient to changes in** θ, which may amplify the variance due to the policy's response to parameter adjustments.

To mitigate the effects of high variance on learning efficiency and policy update stability, variance reduction techniques such as control variates and clipping of importance ratios are often employed.

9 Survey

Please estimate, in minutes, for each problem, how much time you spent (a) writing code and (b) waiting for the results. This will help us calibrate the difficulty for future homeworks.

- **Policy Gradients:**
- **Neural Network Baseline:**
- **Generalized Advantage Estimation:**
- **Hyperparameters and Sample Efficiency:**
- **Humanoid:**
- **Analysis – applying policy gradients:**
- **Analysis – PG variance:**
- **Analysis – return-to-go:**
- **Analysis – importance sampling:**

Solution:

Problem	**Coding Time**	**Result Waiting Time**
Policy Gradients	2 days	Negligible
Neural Network Baseline	1 day	Negligible
Generalized Advantage Estimation	1 day	1 day
Hyperparameters and Sample Efficiency	1 day	1 day
Humanoid	1 day	2 days

- Applying policy gradients: 0.5 day
- Policy gradient variance: 0.5 day
- Return-to-go: 0.5 day
- Importance sampling: 0.5 day