Group Policy Gradient

Junhua Chen[†]
University of Cambridge, United Kingdom
jc2318@cam.ac.uk

Zixi Zhang[†]
University of Cambridge, United Kingdom
zz458@cam.ac.uk

Hantao Zhong
University of Cambridge, United Kingdom
hz445@cam.ac.uk

Rika Antonova
University of Cambridge, United Kingdom
ra702@cam.ac.uk

Abstract: We introduce Group Policy Gradient (GPG), a family of *critic-free* policy-gradient estimators for general MDPs. Inspired by the success of GRPO's approach in Reinforcement Learning from Human Feedback (RLHF), GPG replaces a learned value function with a group-based Monte Carlo advantage estimator, removing the memory, compute, and hyperparameter costs of training a critic while preserving PPO's clipped-objective structure. We prove the consistency of the GPG estimator, analyze the bias-variance tradeoffs, and demonstrate empirically that GPG matches or outperforms PPO on standard benchmarks. GPG makes better use of parallel simulations, which, together with its critic-free design, results in more efficient use of computational resources than PPO.

1 Introduction

Reinforcement learning (RL) trains agents to maximize cumulative rewards through interaction with an environment [1]. Policy gradient methods [2] combined with deep networks excel in domains from game playing [3] to continuous control [4] and generative modeling [5]. Proximal Policy Optimization (PPO) [6], with its clipped objective for stable updates, is now a default choice in deep RL and a core method in Reinforcement Learning from Human Feedback (RLHF) [7] for fine-tuning language models from human preferences.

Until recently, PPO dominated RLHF, leveraging a learned value function (critic) to reduce variance in gradient estimates and improve learning efficiency. However, critics add computational and memory overhead and are prone to approximation errors. Group Relative Policy Optimization (GRPO) [8] addresses these issues by estimating advantages through a group-based Monte Carlo approach, removing the need for a value network. This critic-free design enabled large language models to match or exceed prior RLHF performance, particularly on mathematical reasoning, while substantially reducing memory and computational cost [9].

Building on the strengths and resource-saving benefits of critic-free, group-based policy gradients, as well as their success in RLHF, we extend these methods to the broader realm of general RL. Our contributions are as follows:

- We generalize the GRPO framework and introduce a new critic-free policy gradient algorithm for general Markov Decision Processes (MDPs), which we call Group Policy Gradient (GPG). Like GRPO, GPG modifies only the advantage estimation step while preserving the core structure and benefits of PPO.
- We prove the consistency of our resulting policy gradient estimator under mild assumptions, showing that it converges in the large-group-size limit.

[†]These authors contributed equally to this work

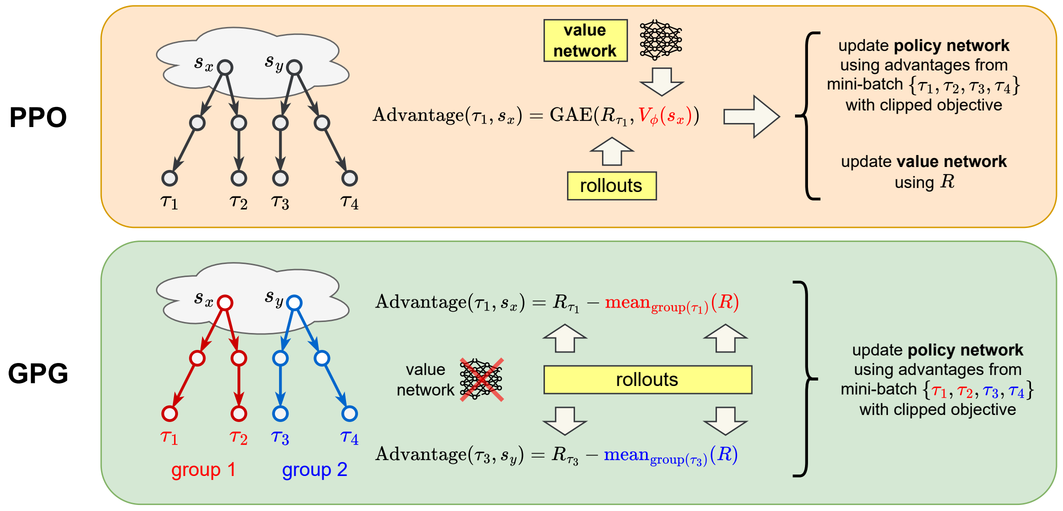
PPO
s_x
s_y
τ_1
τ_2
τ_3
τ_4
value network
$\text{Advantage}(\tau_1, s_x) = \text{GAE}(R_{\tau_1}, V_\phi(s_x))$
rollouts
update policy network using advantages from mini-batch $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ with clipped objective
update value network using R
GPG
group 1
group 2
$\text{Advantage}(\tau_1, s_x) = R_{\tau_1} - \text{mean}_{\text{group}(\tau_1)}(R)$
value network
rollouts
$\text{Advantage}(\tau_3, s_y) = R_{\tau_3} - \text{mean}_{\text{group}(\tau_3)}(R)$
update policy network using advantages from mini-batch $\{\tau_1, \tau_2, \tau_3, \tau_4\}$ with clipped objective

Figure 1: PPO (top) estimates the advantage function using Generalized Advantage Estimation (GAE) with the aid of a learnt value function. In contrast, GPG (bottom) utilizes group-averaged rewards to reduce policy gradient variance. GPG avoids learning a value function and makes greater use of the information in parallel simulations, thereby making better use of computational resources.

- We empirically evaluate GPG on various Gymnasium environments, validating its effectiveness with standard RL benchmarks. We perform ablation studies on design choices and discuss the practical trade-offs of our method.

2 Background

2.1 Notations and Conventions

We largely follow the convention of the Generalized Advantage Estimation paper [4]: We consider an undiscounted formulation of the policy optimization problem. The initial state s_0 is sampled from the distribution ρ_0. A trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$ is generated by sampling actions according to the policy $a_t \sim \pi(a_t|s_t)$ and sampling the states according to the dynamics $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$, until a terminal (absorbing) state is reached. A reward $r_t = r(s_t, a_t, s_{t+1})$ is received at each timestep. The goal is to maximize the expected total reward $\sum_{t=0}^{\infty} r_t$, which is assumed to be finite for all policies. Note that we are not using a discount as part of the problem specification; it will appear below as an algorithm parameter that adjusts a bias-variance tradeoff. But the discounted problem (maximizing $\sum_{t=0}^{\infty} \gamma^t r_t$) can be handled as an instance of the undiscounted problem in which we absorb the discount factor into the reward function, making it time-dependent.

We also utilize standard definitions for Value function and the Q function:

- $V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r_t \mid S_0 = s\right]$
- $Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} r_t \mid S_0 = s, A_0 = a\right]$

For a sampled trajectory τ, we also define the time t γ-discounted total return $R_t^\gamma(\tau) = \sum_{s \geq t} \gamma^{s-t} r_s$

2.2 Advantage Estimation and Baselines

A crucial result, first observed in [2] and generalized in [4], in policy-gradient algorithms, describes a family of policy-gradient estimators and is given by the following proposition; See [4] for the proof.

Prop 1. *For independent trajectories $\tau_{1:N}$ sampled under policy π_θ, any estimator of the form*

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}) \hat{A}_t^{(i)} \tag{1}$$

where $\hat{A}_t^{(i)} = q(s_{t:\infty}^{(i)}, a_{t:\infty}^{(i)}, r_{t:\infty}^{(i)}) - b(s_{1:t}^{(i)}, a_{1:(t-1)}^{(i)})$*, where* q *is any function* q *that satisfying* $\mathbb{E}[q(s_{t:\infty}^{(i)}, a_{t:\infty}^{(i)})|s_t, a_t] = Q(s_t, a_t)$ *and* b *is any function leads to an unbiased estimator for* $\nabla_\theta \eta$*.*

The quantity $\hat{A}$ is often referred to by *advantage estimator*, and b by *baseline*. A judicious choice of q and b leads to lower variance estimates, with $b = V^{\pi_\theta}$ the on-policy value function known to give near-optimal [4] estimates from a variance perspective. However, with V^{π_θ} intractable, it is common in algorithms such as PPO and TRPO to learn an approximate value function $\hat{V}_\phi$ (usually represented as a neural network, trained concurrently using gradient descent) alongside π_θ to reduce variance in advantage estimation. Well-tuned advantage estimators like Generalized Advantage Estimation [4] are critical to the success of algorithms like PPO.

2.3 Proximal Policy Optimization

Among policy gradient algorithms, PPO [6] improves training stability by constraining policy updates, using a relatively cheap clipped surrogate objective:

$$\mathcal{L} = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{old}}}} \left[\min \left\{ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t,\ \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right\} \right] \tag{2}$$

where $\hat{A}_t$ is commonly computed using truncated GAE:

$$\hat{A}_t = \sum_{s=t}^{T} (\gamma\lambda)^{s-t} \delta_s, \qquad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{3}$$

PPO is simple and delivers excellent performance on a wide range of tasks, but requires learning a value network, which is often costly and introduces additional complexity [6, 10, 11].

2.4 Group Relative Policy Optimization

GRPO [8] replaces the learned critic in PPO with a *group*-based, locally estimated baseline. In the RLHF-style *Outcome-Supervision* setting (one scalar reward $r^{(i)}$ per trajectory), GRPO sets advantages for each trajectory in a batch of size N to the group-normalized reward:

$$\hat{A}_t^{(i)} = \frac{r^{(i)} - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \quad t = 1, \ldots, T, \tag{4}$$

(or the analogous cumulative form for *Process Supervision*). Equivalently, GRPO applies REINFORCE with a locally estimated baseline, avoiding a learned value network. This critic-free design drives efficiency gains in LLM fine-tuning [8, 9], but has so far been explored mainly in RLHF settings rather than general MDPs.

2.5 Other Group-based RL Algorithms

Following GRPO's success, several RL algorithms adopting group-based advantage estimation have emerged. Sane [12] propose Hybrid-GRPO, an adaptation of group estimation to general RL environments, though without clear experimental validation. The recently introduced REINFORCE++ [13] is effectively PPO with advantage normalization and $\lambda_{\text{GAE}} = 1$, equivalent to replacing GAE with discounted returns and thus eliminating the critic. While effective in RLHF, it has not been evaluated in general RL settings. GiGPO [14] extends GRPO by normalizing advantages across episodes and states from different trajectories within a group, showing strong results on LLM agent benchmarks but not on broader RL tasks.

3 Group Policy Gradient

In this section, we first introduce a method that computes advantage estimates for each state within a group of sampled trajectories (Algorithm 2). These estimates are then used by the PPO algorithm to update the policy, forming the GPG Algorithm (Algorithm 1). We then explore the design space of GPG methods and present a theoretical result on Group Policy Gradient Estimation.

Algorithm 1 GPG Update

```
1: Initialize policy $\pi_\theta$
2: for each iteration do
3:     Set $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
4:     Collect trajectories group $\tau_{1:N}$ under $\pi_\theta$
5:     Compute advantage estimates $\hat{A}_t^n$ c.f Algorithm 2
6:     for several epochs do
7:         Optimize PPO loss $\mathcal{L}_{PPO}$ by gradient descent on mini-batch estimates
8:     end for
9:     Update policy $\pi_\theta$ via gradient ascent
10: end for
```

Algorithm 2 GPG Advantage Estimation

```
Require: Trajectories $\tau_{1:N}$, Binning function $f$
1: Compute returns $R_t^n$ for all $t$ and $n = 1 \dots N$
2: Initialize Empty Bins $B$
3: for $n = 1 \dots N$ do
4:     for $t = 1 \dots$ do
5:         If $f(s_t^n, t) \neq f(s_i^n, i)$, $i = 1 \dots t-1$, insert $R_t^n$ to $B[f(s_t^{(n)}, t)]$
6:     end for
7: end for
8: Set $\hat{A}_t^{(n)} = R_t^{(n)} - \text{mean}(B[f(s_t^{(n)}, t)])$ for all $t, n$
```

3.1 The GPG Method

GPG departs from PPO and GRPO only in how it estimates advantages, making it simple to implement and scale. It generalizes GRPO's group-based variance-reduction technique, using a broader formulation that applies to any RL setting. Unless noted otherwise, we denote $R_t^{(i)} = R_t^\gamma(\tau_i)$ the time t discounted returns for the ith trajectory of the group.

For GPG, we first introduce the concept of a *binning* function $f : \mathcal{S} \to \mathcal{B}$ where $\mathcal{B}$ is a countable set of bins and $\mathcal{S}$ is the *timestep-aware* state space. We use this to divide the set of states into a set of bins, with $f(s)$ being the bin state s is in. This in turn lets us define the *bin value function* $b(s) = \mathbb{E}_{s' \sim \pi_\theta | f(s') = f(s)}[V(s)]$ which corresponds to a state-likelihood weighted average of state-value functions of the bin each state is in. Inspired by Prop 1, we will use estimates of $b(s)$ as our policy-gradient baseline.

Formally, given independent trajectories $\tau_{1:N}$ sampled for a *group*, we estimate advantages using

$$\hat{A}_t^{(i)} = R_t^{(i)} - \widehat{b}_N(s_t^{(i)}) \tag{5}$$

where $\widehat{b}_N(s)$ is an estimate of the (on-policy) value function at s from the group trajectories. We take the estimated state-value $\hat{b}_N(s)$ to be the average discounted return from the bin of s i.e

$$\hat{b}_N(s) = \text{mean}(\{R_t^{(i)} | (i,t) : f(s_t^i) = f(s) \text{ and } t \text{ is first visit in } \tau_i \text{ to a state in bin } f(s)\}) \tag{6}$$

We illustrate this advantage calculation in Algorithm 1 and Algorithm 2. In our paper, we consider several different possibilities of f, based on time and spatial partitioning of the states. We will see in the experimental section that for the purposes of policy gradient estimation, there is a need to strike a balance in the bin granularity, with both too fine or too coarse a bin size being counter-productive to agent learning. Here, we highlight 4 bin functions to give examples:

- $f(s,t) = 0$, where only 1 bin is present
- $f(s,t) = t$, where an average baseline is computed for each timestep
- $f(s,t) = \epsilon \cdot \text{Round}(s/\epsilon)$ the discretization of space into ϵ-sized packets, for continuous state spaces in $\mathbb{R}^d$.
- $f(s,t) = s$ for discrete state spaces, where each state is its own bin

Relation of GPG to GRPO, PPO and Sampling Methods: Our framework generalizes GRPO with Outcome Supervision. To see this, note that Outcome supervision corresponds to using the trivial binning function $f(s) = 0$, $\forall s \in \mathcal{S}$, where the same mean reward (equal to return when only a terminal reward is given) is subtracted as the baseline for all states, as well as PPO advantage normalization [15] afterwards. Moreover, using $f(s) = 0$ with advantage normalization also corresponds to the REINFORCE++ Algorithm [13]. Similarly, process supervision can be interpreted as subtracting a group-estimated baseline from the R_t^ns, then normalizing. The subtracted baseline depends only on the timestep used, which bears similarity to the choice $f(s,t) = t$. In general, our algorithm is identical to PPO and GRPO, with the sole difference being in our advantage-estimation mechanism.

We further note that the idea of using group-estimated quantities for estimation variance reduction has its roots in Monte-Carlo literature, such as with control variates [16]. The trade-off between a learned and group-estimated control variate also features in GFlownet and Variational Inference literature, such as through comparing the Vargrad and Trajectory Balance Losses [17, 18].

Remark: The form we choose to take for the binning function is not the most general. In fact, as Prop 1 suggests, the binning function can be modified suitably to take the whole *history* up to time t as input. However, for simplicity, we only study binning functions of the current state.

3.2 Theoretical Guarantees of GPG

Many Policy Gradient Algorithms, ranging from the simple REINFORCE [2] method to PPO itself, has the property that if only one gradient step is taken over each batch of data, then as the batch size tends to infinity the gradient update converges in probability to the true policy gradient $\nabla_\theta \eta(\theta)$. Here, we prove the corresponding statement for GPG. We omit technical conditions and only present a proof sketch here, deferring the full proof and details to the Appendix A.

Prop 2. *Consider a MDP environment without discounting and with fixed duration T steps*[1]*. For a group of N iid trajectories $\tau_{1:N}$ sampled from $\pi_{\theta_{old}}$, the GPG Policy-Gradient estimator is*

$$\nabla_\theta \mathcal{L}_N = \nabla_\theta \left[\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \min(\hat{A}_t^n \frac{\pi_\theta(a_t^n|s_t^n)}{\pi_{\theta_{old}}(a_t^n|s_t^n)}, \hat{A}_t^n clip(\frac{\pi_\theta(a_t^n|s_t^n)}{\pi_{\theta_{old}}(a_t^n|s_t^n)}, 1-\epsilon, 1+\epsilon)) \right] \tag{7}$$

where as before $A_t^n = R_t^{(i)} - \widehat{b}_N(s_t^n)$ for some countably-valued binning function f. Then we have

(i) *In the case of one update per group i.e $\pi_\theta = \pi_{\theta_{old}}$, we have*

$$\nabla_\theta \mathcal{L}_N = \nabla_\theta \left[\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \hat{A}_t^n \frac{\pi_\theta(a_t^n|s_t^n)}{\pi_{\theta_{old}}(a_t^n|s_t^n)} \right] = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \hat{A}_t^n \nabla_\theta \log \pi_\theta(a_t^n|s_t^n) \tag{8}$$

(ii) *Moreover in the one update case, assuming some regularity assumptions (see Appendix A), the GPG gradient estimator is a consistent estimator (in group size N) of the policy gradient i.e*

$$\nabla_\theta \mathcal{L}_N \longrightarrow^{\mathbb{P}} \nabla_\theta \eta(\tau) \tag{9}$$

Proof Sketch: **(i)** is the same as a well-known result for PPO and GRPO (c.f [10] or [8]). For **(ii)**, a direct application of the strong law of large numbers (SLLN) is not possible as the $\hat{A}_t^n$ is not independent for different trajectories. Nevertheless, the proof makes heavy use of SLLN, and follows the intuition that for all s, $\hat{b}_N(s) \rightarrow^{\text{almost sure}} b(s)$ as $N \to \infty$, for the bin-value function $b(s)$. This is a valid baseline function as per Prop 1. Some care must be taken when there is an infinite number of bins, but we can show that for sufficiently large group sizes, we can visit *most* of the states (in a likelihood weighted sense) sufficiently often to get good estimates of $b(s)$ for each state.

Corollary: We can also show that the GRPO Policy-Gradient estimator is a consistent estimator of the normalized policy gradient $\nabla_\theta \eta(\tau)/\text{std}(R_1)$. To see that, note that the GRPO gradient estimator is $\boldsymbol{g}_N = \frac{\nabla \mathcal{L}_N}{\hat{\sigma}_N}$ where $\hat{\sigma}_N = \sqrt{\frac{1}{N}\sum_{i=1}^N (R_1^i)^2 - \left(\frac{1}{N}\sum_{i=1}^N (R_1^i)\right)^2}$ is the usual standard deviation estimator of the total returns. By standard results we have that $\hat{\sigma}_N \rightarrow^{\mathbb{P}} \text{std}(R_1)$, and so by Slutsky's Lemma we have the desired result $\boldsymbol{g}_N \rightarrow^{\mathbb{P}} \nabla_\theta \eta(\tau)/\text{std}(R_1)$.

[1]This constraint, merely for ease of notation and analysis, is easily relaxable

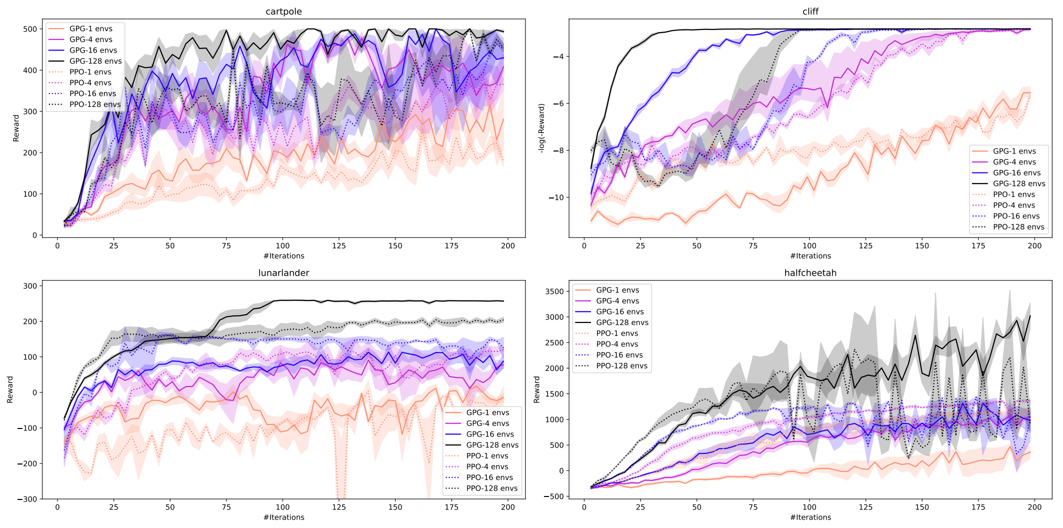

Figure 2: Average episodic reward of GPG and PPO for different numbers of parallel environments. For clarity, we plot on a logarithmic scale for the CliffWalker environment. Given a large number of parallel environments, GPG dominates on all tasks.

4 Experiments

We conduct a comprehensive comparison between PPO and GPG across a suite of reinforcement learning environments from the OpenAI Gymnasium library [19]. Additionally, we present ablation studies to further analyze performance differences.

4.1 Experimental Setup

We build GPG by modifying the advantage-estimation component of the reference CleanRL PPO implementation [20], and otherwise reuse standard PPO hyperparameters when applicable. In GPG each rollout from a vectorized environment forms a *group*, so the nominal group size equals the number of parallel environments (automatic resets can increase the effective size). Because GPG can be sensitive to group size, we sweep the number of parallel environments and report evaluation reward curves for PPO and GPG (5 evaluation seeds, 4 training seeds per configuration), training each run for 200 iterations. All GPG experiments use the time-binning $f(s,t) = t$. We evaluate on four Gymnasium tasks (CartPole, CliffWalker, LunarLander, HalfCheetah) using default environment parameters; further implementation and hyperparameter details are in Appendix B.

4.2 Results

Compared to PPO, GPG exhibits strong performance on all tasks. We comment on the following.

Performance of different methods: As indicated in Figure 2, both PPO and GPG generally benefit from larger numbers of parallel environments. However, due to its use of group-based baselines, the improvement from this parallelism is more pronounced for GPG, similar to GRPO and as hinted by the proof of Prop 2. As a result, for all considered tasks, GPG with large group size performs the best, as displayed in Table 1 or Figure 2.

Sample Efficiency: For a fixed number of parallel environments, GPG matches PPO in sample efficiency. In some tasks, such as LunarLander or Half-Cheetah (Figure 2), PPO learns slightly faster early on, but GPG reliably catches up, and with large group sizes, often surpasses PPO. As shown in Figure 3, increasing the number of parallel environments reduces sample efficiency: for a fixed budget of environment steps, training with fewer environments over more iterations generally yields

Table 1: Rewards for GPG and PPO for various numbers of parallel environments (equivalent to group size for GPG) after 200 Iterations. GPG with group size 128 exhibits dominant performance on all benchmarks.

# Rollouts	Algorithm	CartPole	CliffWalker	Half-Cheetah	Lunarlander
1	GPG	$255.73_{\pm 44.20}$	$-261.30_{\pm 70.56}$	$333.70_{\pm 94.87}$	$-18.78_{\pm 6.76}$
	PPO	$205.82_{\pm 10.05}$	$-442.93_{\pm 73.40}$	$679.33_{\pm 278.55}$	$-19.58_{\pm 12.73}$
4	GPG	$388.65_{\pm 23.65}$	$-17.45_{\pm 0.11}$	$1031.65_{\pm 24.77}$	$74.29_{\pm 10.33}$
	PPO	$316.85_{\pm 31.80}$	$-17.62_{\pm 0.15}$	$1346.30_{\pm 15.33}$	$117.77_{\pm 5.39}$
16	GPG	$428.05_{\pm 33.31}$	$\mathbf{-17.00_{\pm 0.00}}$	$1000.05_{\pm 130.02}$	$75.81_{\pm 12.26}$
	PPO	$423.17_{\pm 22.28}$	$\mathbf{-17.00_{\pm 0.00}}$	$736.45_{\pm 149.77}$	$157.59_{\pm 10.03}$
32	GPG	$481.10_{\pm 10.85}$	$-17.15_{\pm 0.09}$	$1940.86_{\pm 142.08}$	$169.67_{\pm 20.79}$
	PPO	$442.80_{\pm 15.04}$	$\mathbf{-17.00_{\pm 0.00}}$	$1142.48_{\pm 570.77}$	$157.77_{\pm 10.00}$
128	GPG	$\mathbf{495.45_{\pm 2.13}}$	$\mathbf{-17.00_{\pm 0.00}}$	$\mathbf{2773.61_{\pm 222.93}}$	$\mathbf{257.39_{\pm 0.80}}$
	PPO	$474.20_{\pm 7.61}$	$\mathbf{-17.00_{\pm 0.00}}$	$1516.90_{\pm 305.42}$	$200.97_{\pm 4.42}$

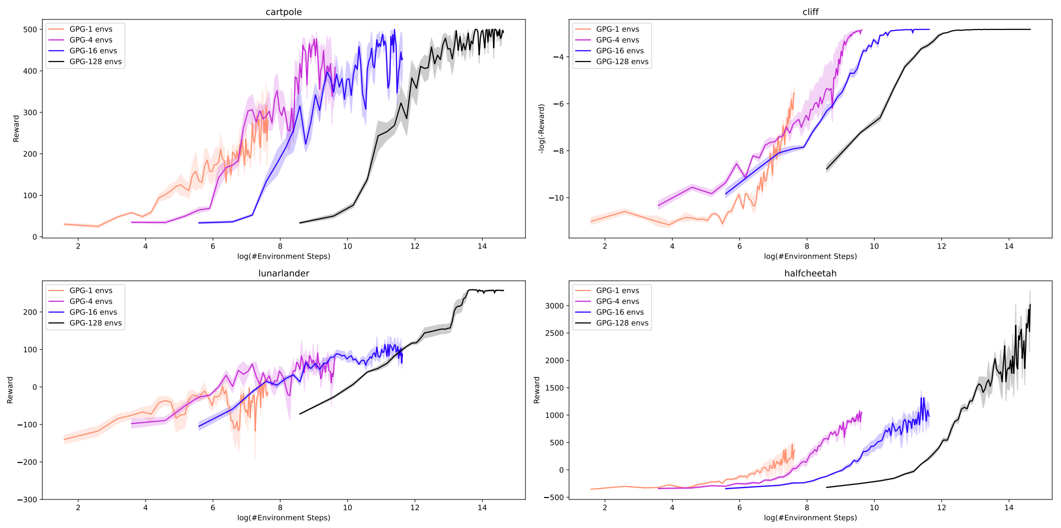

Figure 3: Average episodic rewards for GPG with varying numbers of parallel environments are shown, plotted against the number of evaluated environment steps on a logarithmic scale for clarity. While increasing the number of parallel environments generally reduces sample efficiency, requiring more total environment interactions (but fewer iterations) to reach a given performance threshold, it leads to higher iteration-based performance.

better results. This is intuitive: more iterations give the agent more opportunities to refine its policy, and sequential updates are more expressive than an equivalent amount of parallel computation.

However, small environment counts tend to cap final performance below what large group sizes can achieve, as demonstrated in Table 1. In other words, more parallel environments improve iteration-based performance at the cost of sample efficiency. In settings where parallel simulation is cheap and abundant, iteration-based performance becomes the more relevant metric, and our experiments show that GPG excels under such conditions.

4.3 Ablation Studies

We compare four types of binning functions on the LunarLander task:

1. **Time-based binning**: $f(s, t) = t$, where the baseline value is averaged over samples in the same timestep.

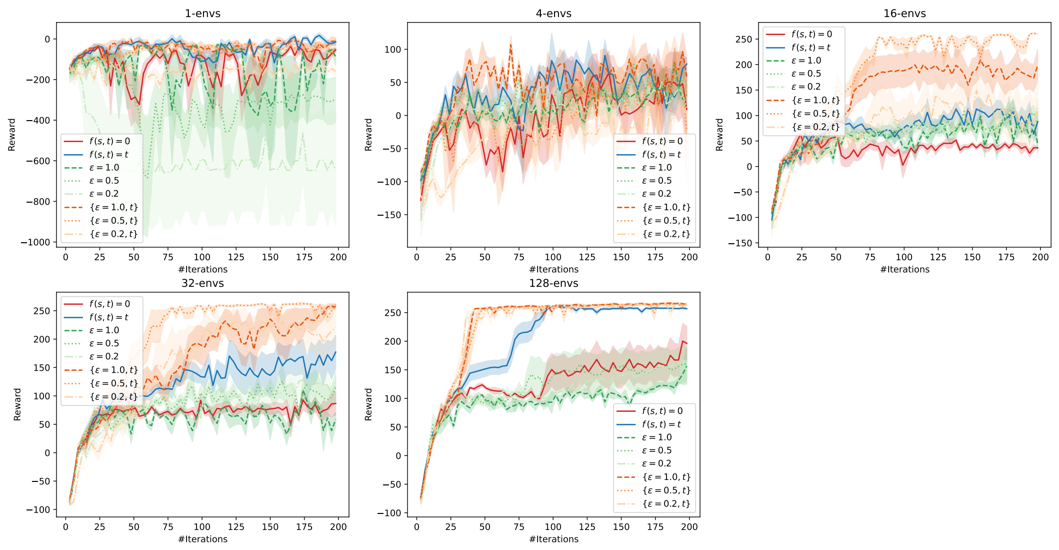

Figure 4: Average episodic reward of GPG on LunarLander with different binning functions.

2. **Universal binning**: $f(\mathbf{s},t)=0$, where all samples share a single bin, regardless of state or time.
3. **Spatial-based binning**: $f(\mathbf{s},t)=\epsilon\cdot\mathbf{Round}(\mathbf{s}/\epsilon)$, which discretizes the continuous state space into bins of size ϵ along each dimension. To keep this manageable, we test $\epsilon=1.0$, 0.5, and 0.2 uniformly across all state dimensions.
4. **Spatial-time-based binning**: $f(\mathbf{s},t)=\{\epsilon\cdot\mathbf{Round}(\mathbf{s}/\epsilon),t\}$, combining spatial and time-based binning. States are grouped only if they fall into the same spatial bin and occur at the same timestep.

These choices trade bias and variance: coarser bins reduce bias but keep high variance (like REINFORCE), while very fine bins lower variance at the cost of bias and small-sample issues. Figure 4 summarizes results. Universal binning consistently underperforms time-based binning. Spatial binning rarely beats time-based binning except at large environment counts, indicating heterogeneous returns within spatial bins. Spatial-time-based matches time-based performance at small parallelism but improves as the number of environments grows: $\epsilon=1.0$ works best at 1–4 envs, $\epsilon=0.5$ at intermediate counts (16–32), and $\epsilon=0.2$ only matches coarser settings once parallelism is very large (128 envs). This progression underscores how increased environment counts mitigate low-sample issues in finer-grained bins. When more environments are available, the higher trajectory count neutralizes the bias from small sample sizes in each bin. As a result, finer-grained binning becomes advantageous, offering better variance reduction and ultimately leading to faster, more stable convergence.

5 Conclusion

We propose Group Policy Gradient (GPG), a simple generalization of PPO and GRPO that replaces the learned critic with a group-based advantage estimator. We prove consistency of the GPG policy-gradient estimator and provide empirically validated guidelines for key design choices (notably group size and binning). Empirically, we demonstrate that GPG matches or exceeds PPO on a suite of standard benchmarks, and analyze its practical tradeoffs and limitations. By replacing the value network with a group-estimated baseline, GPG reduces memory and computational overhead, making it especially attractive when training a critic is costly or unstable, as well as utilizing parallel simulations more efficiently. We hope this study motivates further analysis of variance reduction mechanisms, such as through group baselines, in policy gradient algorithms.

References

[1] R. Sutton and A. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998. doi:10.1109/TNN.1998.712192.

[2] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. doi:10.1038/nature16961.

[4] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

[5] M. Uehara, Y. Zhao, T. Biancalani, and S. Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024. URL https://arxiv.org/abs/2407.13734.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

[8] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

[9] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Ding, H. Xin, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Wang, J. Chen, J. Yuan, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, S. Ye, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Zhao, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, and Z. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[10] OpenAI. Proximal policy optimization (ppo). https://spinningup.openai.com/en/latest/algorithms/ppo.html, 2018. Accessed: 2025-03-02.

[11] A. Juliani and J. T. Ash. A study of plasticity loss in on-policy deep reinforcement learning, 2024. URL https://arxiv.org/abs/2405.19153.

[12] S. Sane. Hybrid group relative policy optimization: A multi-sample approach to enhancing policy optimization, 2025. URL https://arxiv.org/abs/2502.01652.

[13] J. Hu. Reinforce++: A simple and efficient approach for aligning large language models, 2025. URL https://arxiv.org/abs/2501.03262.

[14] L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for llm agent training, 2025. URL https://arxiv.org/abs/2505.10978.

[15] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/. https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

[16] J. Goodman. Variance reduction techniques for monte carlo. Lecture Notes, NYU, 2005. URL https://math.nyu.edu/~goodman/teaching/MonteCarlo2005/notes/VarianceReduction.pdf.

[17] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio. Trajectory balance: Improved credit assignment in gflownets, 2023. URL https://arxiv.org/abs/2201.13259.

[18] L. Richter, A. Boustati, N. Nüsken, F. J. R. Ruiz, and Ömer Deniz Akyildiz. Vargrad: A low-variance gradient estimator for variational inference, 2020. URL https://arxiv.org/abs/2010.10436.

[19] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.

[20] S. Huang, R. F. J. Dossa, C. Ye, and J. Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms, 2021. URL https://arxiv.org/abs/2111.08819.

[21] H. F. Community. Ppo - lunarlander-v2. Hugging Face, 2024. https://huggingface.co/sb3/ppo-LunarLander-v2.

A Formal Statement and Proof of Prop 2

Technical Conditions: Assume the regularity condition that $\sup_{\theta,s,a} \|\nabla_\theta \log \pi_\theta(a|s)\| = C < \infty$ and that all rewards per action are bounded in magnitude by $r_{\max} < \infty$. Assume further that the policy have nonzero probabilities of visiting states in every bin.

Proof:

(i): Same as discussed in [8] A.1.6. When $\pi_\theta = \pi_{\theta_{\text{old}}}$ the clip and min operations are irrelevant, leading to the first equality. The final equality uses the fact that $\nabla_\theta \log f(\theta) = \frac{\nabla_\theta f(\theta)}{f(\theta)}$.

$$\begin{aligned}\nabla_\theta \mathcal{L}_N &= \nabla_\theta \left[\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} \hat{A}_t^n \frac{\pi_\theta(a_t^n|s_t^n)}{\pi_{\theta_{\text{old}}}(a_t^n|s_t^n)}\right] \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} \hat{A}_t^n \frac{\nabla_\theta \pi_\theta(a_t^n|s_t^n)|_{\theta=\theta_{\text{old}}}}{\pi_{\theta_{\text{old}}}(a_t^n|s_t^n)} \\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T} \hat{A}_t^n \nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\end{aligned} \tag{10}$$

(ii): Starting from the simplified form of the gradient in **(i)**, we have

$$\nabla \mathcal{L}_N = \frac{1}{N}\left[\sum_{n=1}^{N}\sum_{t=1}^{T}(R_t^n + b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n) + \sum_{n=1}^{N}\sum_{t=1}^{T}(\hat{b}_N(s_t^n) - b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\right]$$

where recall $b(s) = \mathbb{E}_{s' \sim \pi_\theta | f(s') = f(s)}[V(s)] = \mathbb{E}_{\tau, s_0' \sim \pi_\theta | f(s_0') = f(s)}[R(s_0')]$ is the average (weighted by the policy distribution over states) value function (or equivalently expected forward-return) of a state in the same bin as s. Observe that for $n = 1 \ldots N$,

$$\sum_{t=1}^{T}(R_t^n + b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)$$

is now independently and identically distributed, and moreover corresponds to the REINFORCE gradient estimator with baseline b. As $\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)$ is bounded and r_t^n (and by consequence R_t^n and $b(s_t^n)$) is bounded, the Law of Large Numbers is applicable and thus by Prop 1 we have

$$\begin{aligned}\frac{1}{N}\left[\sum_{n=1}^{N}\sum_{t=1}^{T}(R_t^n + b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\right] &\to^{\mathbb{P}} \mathbb{E}[\sum_{t=1}^{T}(R_t + b(s_t))\nabla_\theta \log \pi_\theta(a_t|s_t)] \\ &= \nabla_\theta \eta(\theta)\end{aligned} \tag{11}$$

Recall standard results that if $X_n \to^{\mathbb{P}} c$ and $Y_n \to^{\mathbb{P}} d$ then $X_n + Y_n \to^{\mathbb{P}} c + d$ and $X_n Y_n \to^{\mathbb{P}} cd$. To establish **(ii)**, it thus suffice to show that

$$\frac{1}{N}\left[\sum_{n=1}^{N}\sum_{t=1}^{T}(\hat{b}_N(s_t^n) - b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\right] \to^{\mathbb{P}} \mathbf{0} \tag{12}$$

We will show this by checking $N \to \infty$,

$$K_N = \mathbb{E}_{\tau_{1:N} \sim \pi_\theta}\left[\left\|\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}(\hat{b}_N(s_t^n) - b(s_t^n))\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\right\|\right] \to 0 \tag{13}$$

as convergence in L^1 implies convergence in $\mathbb{P}$. The key intuition guiding this part of the proof is that for sufficiently large group sizes, we can visit *most* of the bins (in a visit frequency weighted sense) sufficiently often to get good baseline estimates. By triangle inequality and boundedness, have

$$\begin{aligned}K_N &\leq \mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}|\hat{b}_N(s_t^n) - b(s_t^n)|\|\nabla_\theta \log \pi_\theta(a_t^n|s_t^n)\|\right] \\ &\leq \mathbb{E}\left[\frac{C}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}|\hat{b}_N(s_t^n) - b(s_t^n)|\right]\end{aligned} \tag{14}$$

Now observe that $\sum_{t=1}^{T}|\hat{b}_N(s_t^n) - b(s_t^n)|$ is identically distributed for $n = 1 \dots N$ (by symmetry) so by linearity of expectation we obtain

$$\mathbb{E}\left[\frac{C}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}|\hat{b}_N(s_t^n) - b(s_t^n)|\right] = C\mathbb{E}\left[\sum_{t=1}^{T}|\hat{b}_N(s_t^N) - b(s_t^N)|\right] = C\sum_{t=1}^{T}\mathbb{E}\left[|\hat{b}_N(s_t^N) - b(s_t^N)|\right]$$

Next, let $\beta : \mathcal{B} \to \mathbb{R}$ be the mapping from bins to expected-bin value (i.e $\beta(f(s)) = b(s)$) and $\hat{\beta}_N$ to be the empirical bin value function from $\tau_{1:N}$ (i.e such that $\hat{\beta}_N(f(s)) = \hat{b}_N(s)$). Then we have

$$\begin{aligned}C\mathbb{E}\left[\sum_{t=1}^{T}|\hat{b}_N(s_t^N) - b(s_t^N)|\right] &= C\mathbb{E}\left[\sum_{t=1}^{T}|\hat{\beta}_N(f(s_t^N)) - \beta(f(s_t^N))|\left(\sum_{x\in\mathcal{B}}\mathbb{1}_{f(s_t^N)=x}\right)\right] \\ &= C\mathbb{E}\left[\sum_{x\in\mathcal{B}}\sum_{t=1}^{T}|\hat{\beta}_N(x) - \beta(x)|\mathbb{1}_{f(s_t^N)=x}\right] \\ &= C\sum_{x\in\mathcal{B}}\sum_{t=1}^{T}\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\mathbb{1}_{f(s_t^N)=x}\right] \\ &= C\sum_{x\in\mathcal{B}}\sum_{t=1}^{T}\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\,|\text{bin } x \text{ nonempty}\right]\mathbb{E}[\mathbb{1}_{f(s_t^N)=x}]\end{aligned} \tag{15}$$

where the last equality follows from the fact that $\hat{\beta}_N(x)$ depends only on rewards which happen after visiting x, and where we switch order of summation and expectation by convergence theorems. Letting $\rho(x) = \mathbb{E}[\sum_{t=1}^{T}\mathbb{1}_{f(s_t^N)=x}]$, we have

$$\begin{aligned}C\mathbb{E}\left[\sum_{t=1}^{T}|\hat{b}_N(s_t^N) - b(s_t^N)|\right] &= C\sum_{x\in\mathcal{B}}\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\,|\text{bin } x \text{ nonempty}\right]\sum_{t=1}^{T}\mathbb{E}[\mathbb{1}_{f(s_t^N)=x}] \\ &= C\sum_{x\in\mathcal{B}}\rho(x)\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\,|\text{bin } x \text{ nonempty}\right]\end{aligned} \tag{16}$$

It thus remains to show $\sum_{x\in\mathcal{B}}\rho(x)\mathbb{E}_{\tau_{1:N}}\left[|\hat{\beta}_N(x) - \beta(x)|\,|\text{bin } x \text{ nonempty}\right] \to 0$ as $N \to \infty$. Let $\epsilon > 0$ be given in the definition of convergence. We use several facts and observations:

- As rewards are bounded in magnitude and there are T total steps, $|\hat{\beta}_N(x)|, |\beta(x)| \leq r_{\max}T$ always. Consequently, $|\hat{\beta}_N(x) - \beta(x)| \leq 2r_{\max}T$
- For N trajectories, the number of samples in bin x, N_x (i.e the number of trajectories that visit a bin x state), has $N_x \to \infty$ for all $x \in \mathcal{B}$ almost surely, due to the non-zero visit probability assumption. As such, the returns falling in bin x are all independently distributed (as they're from different trajectories by first-visit assumption), and identically distributed according to $R(s), s \sim \pi_\theta | f(s) = x$. Thus, as rewards are bounded, SLLN applies to $\hat{\beta}_N(x)$ and so $\hat{\beta}_N(x) \to \beta(x)$ almost surely for all $x \in \mathcal{B}$.
- Moreover, since as mentioned $\hat{\beta}_N$ and β are bounded by $r_{\max}T$ always, the bounded convergence theorem shows that $\mathbb{E}[|\hat{\beta}_N(x) - \beta(x)|] \to 0$ as $N \to \infty$ for all x

- $\sum_{x\in\mathcal{B}} \rho(x) = \mathbb{E}[\sum_{t=1}^{T} \mathbb{1}_{s_t^n \text{ is any state}}] = T$ is finite
- As $\mathcal{B}$ is countable, it is possible to pick $\mathcal{T} \subset \mathcal{B}$ finite that $\sum_{s\in\mathcal{B}\setminus\mathcal{T}} \rho(s) < \delta$ for any $\delta > 0$ (for instance by enumerating the bins and truncating the sequence per definition of convergence)

Thus, pick $\mathcal{T}$ such that $\sum_{s\notin\mathcal{T}} \rho(s) < \frac{\epsilon}{6r_{\max}T}$. As $\mathcal{T}$ is finite, $\sup_{x\in\mathcal{T}} \mathbb{E}[|\hat{\beta}_x(s) - \beta(s)|] \to 0$ (a finite number of sequences converge uniformly). Thus, there exist $M_{\mathcal{T}}$ such that for all $N > M_{\mathcal{T}}$, have $\sup_{x\in\mathcal{T}} \mathbb{E}[|\hat{\beta}_N(x) - \beta(x)|] \le \frac{\epsilon}{3T}$. Then we have for $N > M_{\mathcal{T}}$:

$$\begin{aligned}
\sum_{x\in\mathcal{B}} \rho(x)\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\right] &= \sum_{x\in\mathcal{T}} \rho(x)\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\right] + \sum_{x\in\mathcal{B}\setminus\mathcal{T}} \rho(x)\mathbb{E}\left[|\hat{\beta}_N(x) - \beta(x)|\right] \\
&\le \sup_{x\in\mathcal{T}} \mathbb{E}[|\hat{\beta}_N(x) - \beta(x)|] \sum_{x\in\mathcal{T}} \rho(x) + \sum_{x\in\mathcal{B}\setminus\mathcal{T}} \rho(x)\mathbb{E}\left[2r_{\max}T\right] \\
&\le \frac{\epsilon}{3T} \sum_{x\in\mathcal{T}} \rho(x) + 2r_{\max}T \sum_{x\in\mathcal{B}\setminus\mathcal{T}} \rho(x) \\
&\le \frac{\epsilon}{3T} T + 2r_{\max}T \frac{\epsilon}{6r_{\max}T} \\
&= \frac{2\epsilon}{3} < \epsilon
\end{aligned}$$

This establishes $K_N \to 0$ as $N \to \infty$. It thus follows that

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \hat{A}_t^n \nabla_\theta \log \pi_\theta(a_t^n | s_t^n) \to^{\mathbb{P}} \nabla_\theta \eta(\theta)$$

as required □

B Environments

Cart Pole A pole is attached to a cart that moves along a track. The goal is to push the cart left or right to keep the pole upright. The environment features a continuous observation space representing the cart's (angular) position and velocity, and a discrete action space with left and right movements. Reference hyper-parameters are given in the CleanRL implementation.

Cliff Walking The agent must traverse a 4×12 grid world from the bottom-left to the bottom-right corner, avoiding a cliff along the bottom row. The observation space is discrete, representing the agent's grid position, while the action space consists of four directional moves: up, down, left, and right. Reference hyper-parameters are given in the CleanRL implementation.

Lunar Lander A lander starts from the top of the environment and must safely land on a designated pad. The agent receives continuous observations of its position, velocity, angle, angular velocity, and leg-ground contact status. The action space is discrete, controlling the main engine and two orientation engines (left and right). Rewards and penalties are provided for successful landings, crashes, proximity to the landing pad, velocity reduction, and engine usage. Reference hyper-parameters are obtained from [21].

MuJoCo Half Cheetah MuJoCo (Multi-Joint dynamics with Contact) is a physics-based robotics simulation. The Half Cheetah environment features a cheetah-like robot with two legs and six joints. The agent applies torques to the joints to propel the robot forward as quickly as possible. Both the observation and action spaces are continuous. Reference hyper-parameters are given in the CleanRL implementation.