scale

To Critic Or Not To Critic

Jason Wei[†1,2], **George Pu**[1], **Jingxuan Fan**[1], **Yuan (Emily) Xue**[1], **Yuan (Christy) Li**[1]

[1]Scale AI, [2]Carnegie Mellon University

Abstract

As large language models tackle longer and more complex tasks, understanding when a critic improves the efficiency of reinforcement learning (RL) becomes increasingly important. Proximal Policy Optimization (PPO) trains a critic to reduce variance and provide fine-grained credit assignment, while Group Relative Policy Optimization (GRPO) drops the critic and avoids the bias of bootstrapping from the critic's value estimates. Although recent critic-based methods have reported strong performance relative to GRPO, it remains unclear when each approach is preferable. We argue that there is no universal answer; instead, the choice depends on several interacting factors, including batch size, critic reliance, and task complexity. We study the bias–variance tradeoff in terms of gradient direction, measuring how closely each method's gradient estimates align with the true policy gradient as batch size grows. Under a shared actor objective, we compare GRPO with PPO in control tasks, short-horizon math, and long-horizon terminal agents. Across more than 1,200 training runs, including 96 with LLMs, we find that PPO generally leads at small batch sizes, while GRPO often overtakes heavily bootstrapped PPO and reaches parity with lightly bootstrapped PPO at larger batches. At longer horizons and in more complex domains, GRPO needs larger batches to catch up to PPO's performance. Accounting for training compute, critics pay off at small budgets, but their returns diminish at larger budgets.

1 Introduction

Reinforcement learning (RL) is an essential stage of large language model (LLM) post-training. Early pipelines for RL from human feedback relied predominantly on Proximal Policy Optimization (PPO) [1, 2], which trains a separate critic network to reduce gradient variance and provide token-level credit assignment through Generalized Advantage Estimation (GAE) [3]. Since DeepSeek-R1 demonstrated strong reasoning performance with critic-free Group Relative Policy Optimization (GRPO) [4, 5], the open-source community has largely abandoned critics, citing reduced memory, implementation simplicity, and unreliable value estimation [5–8].

More recently, critic-based methods have reported strong results relative to GRPO, renewing interest in critics [9–13]. However, these comparisons are typically made at a single batch size and in a narrow set of domains. For example, VAPO and SAO train with batch sizes of 512 and 128, respectively, in math and coding [10, 12]. It therefore remains unclear whether these gains hold at other batch sizes or in other domains.

[†]Work conducted during an internship at Scale AI.

As LLMs tackle longer agentic tasks, individual rollouts grow more expensive, making the choice of advantage estimator increasingly consequential [6, 14, 15]. A critic can reduce gradient variance, but adds training compute and can introduce bias through bootstrapping. We argue that neither GRPO nor PPO is universally preferable. Instead, the choice depends on several interacting factors, and we focus on three that shape this bias–variance tradeoff.

The first is *batch size*, the number of trajectories used per actor update. Larger batches reduce gradient variance but not bias, so the relative performance of PPO and GRPO can change with batch size. The second is *critic reliance*. GRPO avoids the critic entirely, while PPO's GAE parameter λ controls how strongly its advantage calculation bootstraps from the critic. More bootstrapping typically lowers variance at the cost of bias from critic approximation error. The third is *task complexity*, which we study through horizon and domain. Longer horizons increase our upper bound on gradient variance (Section 3.2). Domains differ in properties that may also raise gradient noise, such as the dimensionality of the state and action spaces and the diversity of trajectories relevant to learning [16, 17]. Both can make variance reduction more valuable.

Prior work has studied these factors largely in isolation, including batch size scaling and rollout allocation [18–21], critic reliance in PPO [6, 9, 13], and the growth of gradient variance with horizon [22, 23]. How they jointly affect the relative performance of PPO and GRPO across domains remains poorly understood. We address this gap across more than 1,200 training runs, including 96 with LLMs, by comparing their advantage estimators under a shared actor objective while varying batch size, critic reliance, and task complexity. We study three settings (Table 1): control tasks with Pendulum and HalfCheetah, short-horizon single-turn math, and long-horizon multi-turn terminal agents. Our contributions are as follows:

1. **Batch size, critic reliance, and task complexity affect when GRPO catches up.** PPO generally leads at small batch sizes, while GRPO often overtakes heavily bootstrapped PPO and reaches parity with lightly bootstrapped PPO at larger batch sizes. At longer horizons and in more complex domains, GRPO needs larger batches to catch up to PPO's performance. Table 2 summarizes all comparisons.
2. **Studying the bias–variance tradeoff in terms of gradient direction helps explain the crossover.** This analysis indicates when GRPO's batch gradient estimates point closer to the true policy gradient than PPO's as batch size grows and why lighter bootstrapping can delay this crossover. Gradient measurements in control tasks support the cosine model and the horizon bound, and show higher noise scales at longer horizons and in HalfCheetah than in Pendulum.
3. **Critics pay off most in the low-compute regime.** At small batches, PPO's critic can provide enough accuracy improvement to justify its additional cost. Once GRPO reaches comparable performance at larger batches, it requires less estimated training compute.

2 Preliminaries

2.1 PPO

PPO estimates advantages using a learned value function or critic [1]. Consider a trajectory collected under $\pi_{\theta_{\text{old}}}$, with states s_t and actions a_t for $t = 1, \ldots, H$, where H is the horizon. For LLMs, each action is a generated token, and the state is the context available before generating it. With importance-sampling ratio $\rho_t(\theta) = \pi_\theta(a_t \mid s_t) / \pi_{\theta_{\text{old}}}(a_t \mid s_t)$, PPO maximizes the clipped surrogate objective

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_t\big[\min\{\rho_t \hat{A}_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\}\big], \tag{1}$$

where ϵ sets the clipping range. The critic $V_\phi(s_t)$ estimates the expected discounted future reward from state s_t, and GAE uses its predictions to compute the advantage $\hat{A}_t$.

Generalized advantage estimation. In the finite-horizon setting, let $D = H - t + 1$ denote the number of transitions remaining and set $V_\phi(s_{H+1}) = 0$ at the terminal state. For $1 \le k \le D$, the k-step advantage estimate is

$$\hat{A}_t^{(k)} = -V_\phi(s_t) + \sum_{\ell=0}^{k-1} \gamma^\ell r_{t+\ell} + \gamma^k V_\phi(s_{t+k}),$$

where $\gamma \in [0, 1]$ is the reward discount factor and r_t is the reward after action a_t. GAE mixes these estimates with geometrically decaying weights controlled by $\lambda \in [0, 1]$ [3]:

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = (1-\lambda) \sum_{k=1}^{D-1} \lambda^{k-1} \hat{A}_t^{(k)} + \lambda^{D-1} \hat{A}_t^{(D)}. \tag{2}$$

The estimates with $k < D$ bootstrap from the critic, while $\hat{A}_t^{(D)}$ is the Monte Carlo return minus the baseline. The final estimate receives the remaining weight λ^{D-1}, so the weights sum to one (Appendix A.1). Smaller λ bootstraps more, typically reducing variance at the cost of critic-induced bias. At $\lambda = 1$, only the Monte Carlo estimate remains, with no bootstrap bias. The critic regresses on the λ_V-return target $\hat{V}_t = \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda_V)} + V_\phi(s_t)$ with loss

$$\mathcal{L}_{\mathrm{VF}}(\phi) = \mathbb{E}_t\left[(V_\phi(s_t) - \hat{V}_t)^2\right]. \tag{3}$$

The target $\hat{V}_t$ is held fixed during the critic update. *Decoupled GAE* lets the critic's λ_V differ from the actor's λ, typically with $\lambda_V = 1$ for Monte Carlo critic targets [6, 13].

Length-adaptive GAE. With fixed $\lambda < 1$, the full Monte Carlo estimate receives progressively less weight at early steps as the trajectory grows longer, increasing reliance on intermediate critic predictions. Following VAPO, length-adaptive GAE adjusts λ with the trajectory horizon H [10]:

$$\lambda(H) = 1 - \frac{1}{\alpha H}, \tag{4}$$

for $\alpha H > 1$. Larger α gives more weight to the Monte Carlo estimate and reduces bootstrapping. At step t, this estimate receives weight $\lambda(H)^{H-t}$, which is smallest at the first step. For long trajectories, this minimum approaches $e^{-1/\alpha}$, so α sets an approximately length-independent minimum Monte Carlo weight (Appendix A.1). For long trajectories, the minimum Monte Carlo weights are approximately 4.5×10^{-5}, 0.14, and 0.51 for $\alpha = 0.1$, 0.5, and 1.5, respectively.

Privileged critics. The critic may also receive *privileged information* z hidden from the policy, such as a reference solution, and predict $V_\phi(s_t, z)$ [8, 13, 24]. To keep the privileged baseline from introducing policy-gradient bias, it suffices that z be conditionally independent of the current action given s_t [8, 25, 26].

2.2 GRPO

GRPO estimates advantages by comparing rewards across trajectories sampled from the same prompt or start state [5]. For a group of $G > 1$ trajectories collected under $\pi_{\theta_{\mathrm{old}}}$, it maximizes Equation 1 averaged

over their actions, using advantage $\hat{A}_{i,t}$ [27, 28]. With outcome reward r_i for trajectory i and the group's reward mean $\bar{r}$ and standard deviation σ_r, GRPO centers each reward and divides by σ_r. The division reweights groups by how much their rewards vary, so Dr. GRPO omits it [28]:

$$\hat{A}_{i,t}^{\text{GRPO}} = \frac{r_i - \bar{r}}{\sigma_r}, \quad \hat{A}_{i,t}^{\text{Dr. GRPO}} = r_i - \bar{r}. \tag{5}$$

Groups with identical rewards get zero advantage. In both forms, all actions in a trajectory share one advantage, so GRPO provides no within-trajectory credit assignment, and without a critic its advantages carry no critic approximation bias.

3 A Directional Bias–Variance Account

3.1 Bias and Variance of Gradient Estimators

Let $g = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$ be the policy gradient of the expected outcome reward $r(\tau)$ of a trajectory τ, and let $\hat{g}_B$ average B per-trajectory estimates $\hat{g}$, possibly biased, from an advantage estimator, independent across trajectories or across groups for GRPO (Appendix A.3). With bias $b = \mathbb{E}[\hat{g}] - g$ and variance $V = B\,\mathrm{Var}(\hat{g}_B)$, the mean squared error (MSE) decomposes as

$$\mathbb{E}\left[\|\hat{g}_B - g\|^2\right] = \underbrace{\|b\|^2}_{\text{squared bias}} + \underbrace{V/B}_{\text{variance}}. \tag{6}$$

Gradient direction. MSE measures discrepancies in both gradient direction and magnitude, but popular optimizers such as AdamW and Muon make updates largely invariant to gradient magnitude [29–31]. We therefore focus on gradient directions independently of their magnitude [32]. We measure an estimator's *alignment* by the cosine similarity $c = \cos(\mathbb{E}[\hat{g}], g)$ and its *noise scale* $\nu = V/\|\mathbb{E}[\hat{g}]\|^2$, which extends the simple noise scale of McCandlish et al. [33, Equation 2.9] to biased estimators. These are directional counterparts of the bias and variance terms in Equation 6.

The cosine similarity between $\hat{g}_B$ and g ranks unit-norm SGD steps to first order (Appendix A.3), but it does not rank all update directions. For example, natural policy gradients and adaptive optimizers reshape g [34], and entropy bonuses and asymmetric clipping modify the objective to increase exploration [27, 35].

Under our shared actor objective (Section 4.1), PPO and GRPO both estimate the same g, so cosine similarity to g is a fair basis for comparing them. Under the approximation derived in Appendix A.3, which we call the *cosine model*, a batch of size B has expected cosine similarity

$$\mathbb{E}[\cos(\hat{g}_B, g)] \approx \frac{c}{\sqrt{1 + \nu/B}}. \tag{7}$$

When $B \ll \nu$, Equation 7 gives an expected cosine similarity of approximately $c\sqrt{B/\nu}$, so sampling noise strongly limits the expected cosine similarity. When $B \gg \nu$, expected cosine similarity approaches c, and additional sampling gives diminishing improvements. The cosine model thus expresses a directional bias–variance tradeoff. Larger batches shrink the noise term ν/B but cannot raise the expected cosine similarity above c, so c acts as an alignment ceiling as batch size grows. This can favor PPO's lower noise scale at small batch sizes and GRPO's better-aligned mean gradient at large ones.

Let c_{PPO} and c_{GRPO} denote the respective alignments, and ν_{PPO} and ν_{GRPO} the respective noise scales. Assume $0 < c_{\text{PPO}} < c_{\text{GRPO}}$ and $0 < \nu_{\text{PPO}} < \nu_{\text{GRPO}}$. At batch sizes small relative to both noise scales, the cosine model ranks estimators by c^2/ν. If $c_{\text{PPO}}^2/\nu_{\text{PPO}} > c_{\text{GRPO}}^2/\nu_{\text{GRPO}}$, PPO leads in this regime and the approximation predicts a positive crossover batch size

$$B_{\cos}^* = \frac{c_{\text{PPO}}^2 \nu_{\text{GRPO}} - c_{\text{GRPO}}^2 \nu_{\text{PPO}}}{c_{\text{GRPO}}^2 - c_{\text{PPO}}^2}. \tag{8}$$

3.2 Horizon and Gradient Variance

We next apply a standard Monte Carlo gradient-variance argument to trajectories with bounded outcome rewards [36]. Horizon is one source of task complexity, so the bound motivates our joint study of task complexity and batch size. For a trajectory τ with states and actions $s_0, a_0, s_1, \ldots$ and horizon H, consider the Monte Carlo policy-gradient estimator [25]:

$$\hat{g}(\tau) = r(\tau) \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(a_t \mid s_t), \tag{9}$$

which is unbiased, $\mathbb{E}[\hat{g}] = g$.

Proposition 1 (Bounded outcome rewards)**.** Consider trajectories from a fixed policy and task distribution with $r(\tau) \in [0, 1]$ regardless of horizon. Suppose $\mathbb{E}[\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|^2] \le C$ for every t and $\|g\| \ge m > 0$, where C and m are independent of H. Then

$$V \le CH = O(H), \qquad \nu \le \frac{CH}{\|g\|^2} - 1 \le \frac{CH}{m^2} - 1 = O(H). \tag{10}$$

Proposition 1 gives $O(H)$ bounds because the total reward is bounded, as in our LLM verifiers and rescaled control rewards (Appendix A.2). With accumulated rewards, standard bounds instead grow as $O(H^3)$ [22, 36]. Dimensionality enters through C, which sums over action dimensions [16]. Since the cosine model depends on ν only through ν/B, the batch size needed to approach the alignment ceiling can grow in proportion to CH, that is, with both horizon and dimensionality. Section 5 measures these quantities in control tasks.

3.3 From Analysis to Experiments

A critic can make small batches more effective by reducing the noise scale, but bootstrapping through an imperfect critic can also lower alignment. In the cosine model, larger batches shrink the noise term but leave the alignment ceiling unchanged. We therefore vary batch size and critic reliance together, testing whether additional sampling lets GRPO catch up and whether lighter bootstrapping keeps PPO competitive. The horizon bound motivates repeating this comparison as trajectories grow longer. Our analysis applies most directly to our control tasks, where updates are on-policy and unit-norm stochastic gradient descent (SGD) makes first-order progress depend on the update's cosine similarity with g [37]. In LLM training, off-policy updates, token averaging, and AdamW weaken this link.

4 Experiments

We compare PPO and GRPO in control tasks, short-horizon math, and long-horizon terminal agents, and we vary batch size, critic reliance, and task complexity jointly. For LLMs, we also assess whether critics

	Control	Math	Terminal
Policy model	MLP	Qwen2.5-3B Base	Qwen3.5-4B
Max length	5, 25, 200 actions	4,096 tokens	65,536 tokens
Max turns	–	1	64
Batch size B	4–65,536	4–4,096	128–2,048
PPO α	0.1, 0.5, 1.5	0.1, 0.5, 1.5	0.5, 1.5
Gradient updates	5,000	800	200
Seeds	5	3	1

Table 1. Experimental settings. Batch sizes increase in factors of four. Max length denotes the episode cap in actions for control tasks and in tokens for LLM tasks (response length for math, context length for terminal agents). Max turns is the maximum number of assistant turns.

justify their additional compute. Table 1 outlines the experimental settings, and Table 2 summarizes the outcomes across tested batch sizes.

4.1 Shared Comparison Protocol

Matched batch sizes. Let P be the number of prompts (or start states) and G the number of trajectories per prompt, so that the batch size is $B = PG$. GRPO requires $G > 1$ to form its group baseline, while PPO can use $G = 1$. We compare their advantage estimators under a shared actor loss and matched trajectory batch sizes, starting from the same policy within each setting. Group sizes differ between methods. At the largest batch size we measure, our choices of $G = 16$ for GRPO and $G = 1$ for PPO are at or near the best group size by expected cosine similarity in every setting, so comparisons at the largest batch size do not hinge on this choice (Appendix Figure 12).

Crossover and parity. At each tested batch size B, one method *leads* if its mean score exceeds the other's by more than twice the standard error of the difference. Otherwise, the methods are considered *tied*. For each setting and PPO variant, we summarize the comparison across batch sizes by its outcome at the largest tested batch size: a *crossover* if GRPO leads, *no crossover* if PPO leads, and *parity* if they are tied. A crossover starts at the smallest tested batch size B where GRPO leads at B and every larger batch size. For comparisons ending in parity, parity starts at the smallest tested batch size B where PPO does not lead at B or any larger tested batch size. Standard errors are over training seeds in the control tasks and math, and over evaluations for terminal agents, where we have one training seed. Control comparisons use validation scores, while LLM comparisons use the downstream test benchmark average.

Shared setup. Following SAO and IcePop [12, 38, 39], all methods maximize the masked objective

$$\mathcal{J}_{\text{IcePop}}(\theta) = \mathbb{E}_t\left[\mathcal{M}\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{infer}}(a_t \mid s_t)}; 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\right)\hat{A}_t\right], \tag{11}$$

where π_θ is the policy being trained, π_{infer} is the policy that collected the rollouts, and $\epsilon_{\text{low}} = 0.5$ and $\epsilon_{\text{high}} = 4.0$ set the range of probability ratios that $\mathcal{M}$ keeps. In control tasks, updates are on-policy with no training–inference mismatch, so masking is inactive. PPO uses decoupled GAE with length-adaptive λ to compute $\hat{A}_t$ and $\lambda_V = 1$ for critic updates (Section 2.1). Our critics also see privileged inputs

Setting	H	PPO (α) 0.1	0.5	1.5
Pendulum	5	GRPO (4)	GRPO (4)	GRPO (4)
	25	GRPO (256)	GRPO (1k)	GRPO (1k)
	200	GRPO (65k)	Parity (65k)	Parity (65k)
HalfCheetah	5	GRPO (16)	GRPO (16)	GRPO (16)
	25	GRPO (4k)	Parity (16k)	PPO
	200	Parity (256)	PPO	PPO
Math	—	GRPO (1k)	Parity (1k)	Parity (256)
Terminal	—	—	Parity (2k)	Parity (2k)

GRPO crossover, small B — large B Parity PPO leads

Table 2. PPO–GRPO outcomes across batch sizes. *GRPO (B)* denotes a crossover: GRPO leads at B and every larger tested batch. *Parity (B)* indicates that PPO no longer leads from B onward, with a tie at the largest tested batch. *PPO* indicates a continued PPO lead at the largest tested batch. Darker orange marks earlier GRPO crossovers; gray denotes parity and blue a continued PPO lead. Control results use Dr. GRPO validation scores, and LLM results use downstream test averages. See Section 4.1 for the statistical criterion. A dash indicates an untested variant or inapplicable horizon.

hidden from the actor, and Appendix C.3 assesses their effect in math. We evaluate both Dr. GRPO and GRPO with standard-deviation normalization in control tasks and only GRPO with standard-deviation normalization in the LLM settings (Equation 5).

4.2 Experimental Settings

Control tasks. We use fixed-horizon versions of Gymnasium's Pendulum-v1 and HalfCheetah-v4 [40], with horizons $H \in \{5, 25, 200\}$ and a two-layer multilayer perceptron (MLP) actor. We accumulate the environment's per-step rewards over each episode, rescale their average to $[0, 1]$, and provide the reward only at the final state. GRPO uses $G = 16$ trajectories per start state, except at $B = 4$, where $G = 4$. Both methods take one unit-norm SGD update per batch. Actor and critic learning rates are 10^{-3} and 5×10^{-3}, respectively. We focus on Dr. GRPO because standard-deviation normalization can overweight uninformative groups, especially with continuous rewards, where group reward standard deviations can differ by orders of magnitude. Appendix C.2 analyzes this effect in detail. We report each run's best validation score. Appendix B.1 gives the remaining environment and training details, and Appendix C.2 reports additional results for GRPO with standard-deviation normalization and GAE $\lambda = 1$.

Short-horizon math. We train Qwen2.5-3B Base on 10,000 DeepScaleR problems [41, 42], with a binary reward for final-answer correctness. GRPO uses $G = \min(16, \sqrt{B})$ trajectories per problem. Each step samples $2B$ trajectories for two actor updates of batch size B, and PPO takes one critic update per actor update. Actor and critic learning rates are 10^{-6} and 2.5×10^{-6}, respectively, with AdamW [30]. PPO's critic is pretrained and receives the reference answer and a solution as privileged inputs. We select each run's best checkpoint by avg@16 on 500 held-out DeepScaleR problems and evaluate it on MATH-500, GSM8K, AMC23, Minerva Math, and OlympiadBench [43–47], reporting mean accuracy over multiple samples per problem. Appendix B.2 provides additional training and evaluation details.

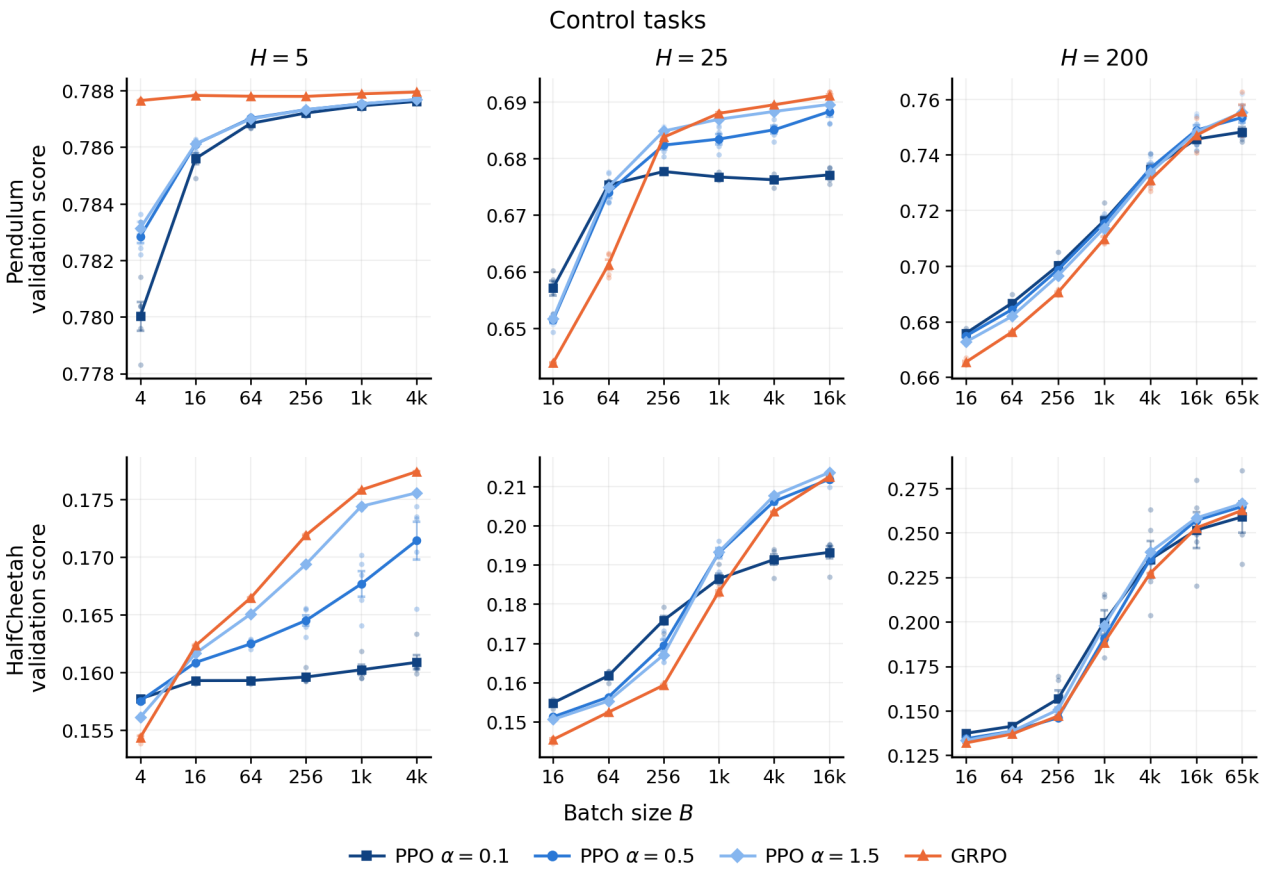


Figure 1. GRPO gains on PPO as batches grow. Best validation score within 5,000 updates in Pendulum (top) and HalfCheetah (bottom). GRPO is the centered Dr. GRPO form. Mean and one standard error over five seeds, with dots for individual seeds.

Long-horizon terminal agents. We train Qwen3.5-4B on TMax-15K terminal tasks, largely following the Tmax setup [48, 49]. The agent runs bash commands in a sandbox, and hidden tests on the final environment state assign the reward. GRPO uses $G = 16$ trajectories per task. Training is asynchronous, with one actor update per step and a maximum staleness of four policy versions. Actor and critic learning rates are 10^{-6} and 5×10^{-6}, respectively, with AdamW. PPO's critic is pretrained for 250 updates of 64 trajectories, and its privileged inputs are the task's grading tests and target state. We report all metrics at the final checkpoint (step 200): validation avg@3 on 500 held-out TMax-15K tasks and test avg@8 on Terminal-Bench 2.1 and TB-Lite [50, 51]. Appendix B.2 gives the remaining LLM training and evaluation details, including the differences from Tmax.

4.3 Batch Size, Critic Reliance, and Task Complexity

PPO leads at small batch sizes, and GRPO catches up as batches grow. Figure 1 shows that PPO leads at the smallest batch sizes in every control setting except Pendulum at $H = 5$, where GRPO leads throughout. As batch size grows, GRPO closes the gap, and at the largest tested batch sizes it leads or ties PPO in 20 of 23 comparisons (Table 2). The LLM settings follow the same pattern (Figure 2). In math, the best PPO variant leads GRPO at the two smallest batch sizes, $B \in \{4, 16\}$, by 1.9–2.7 points, and GRPO leads or ties every PPO variant at the two largest, $B \in \{1{,}024, 4{,}096\}$. In terminal agents, both PPO variants lead GRPO at the two smaller batch sizes, $B \in \{128, 512\}$, by up to 5.9 points, and all three

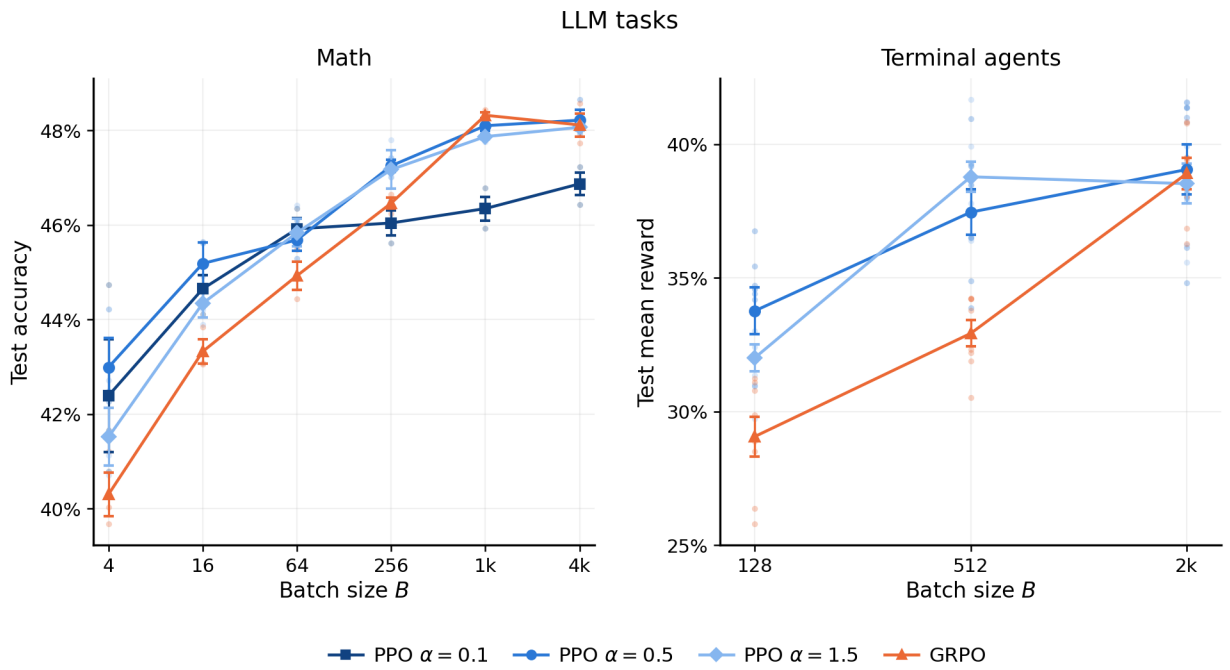


Figure 2. PPO's lead fades as batches grow in LLM tasks. Test benchmark average with mean and one standard error. Left: five math benchmarks at each run's best validation checkpoint, over three training seeds. Right: Terminal-Bench 2.1 and TB-Lite at the final checkpoint, with one training seed and eight evaluation passes. Faint dots show individual seeds or passes, and Appendix Tables 5 and 6 list validation scores.

methods tie at the largest, $B = 2{,}048$, within 0.6 points (Appendix Tables 5 and 6). A critic therefore helps most when samples are scarce.

Lighter bootstrapping keeps PPO competitive at large batch sizes. Whether GRPO crosses over at large batch sizes depends on how heavily PPO bootstraps. Against heavily bootstrapped PPO ($\alpha = 0.1$), GRPO crosses over in six of seven settings, and the exception, HalfCheetah at $H = 200$, ends in parity. In math, it leads from $B = 1{,}024$ onward. In contrast, against lightly bootstrapped PPO, GRPO crosses over only in Pendulum at $H \in \{5, 25\}$ and HalfCheetah at $H = 5$, and every other comparison ends in parity or a continued PPO lead. This pattern is what Section 3.1 predicts. Larger batches shrink the noise term but leave alignment unchanged, so heavy bootstrapping loses its advantage first.

Longer horizons and more complex domains delay the crossover. In Pendulum with $\alpha = 0.1$, the crossover moves from $B = 4$ at $H = 5$ to 256 at $H = 25$ and 65,536 at $H = 200$ (Table 2). With lighter bootstrapping, Pendulum's longest horizon ends in parity, and HalfCheetah's crossovers against lightly bootstrapped PPO disappear beyond $H = 5$. Changing the domain has the same effect. At every matched horizon and α, HalfCheetah's outcome is less favorable to GRPO than Pendulum's. Against the same two lightly bootstrapped PPO variants, terminal agents reach parity at $B = 2{,}048$, whereas math reaches it at 1,024 and 256. The noise scale shows the same ordering, rising with horizon and 4–11$\times$ larger in HalfCheetah than in Pendulum (Section 5). Because the domains also differ in dynamics, rewards, and training, these comparisons suggest a role for task complexity without isolating it.

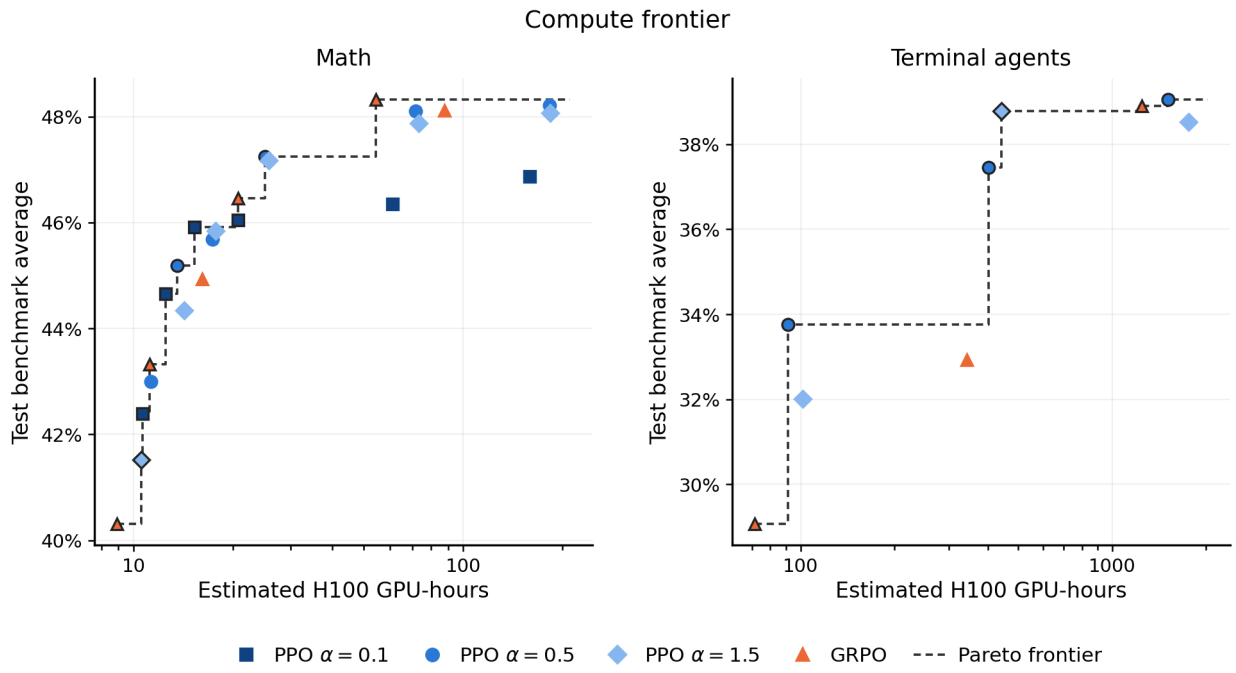


Figure 3. Critics pay off only at small compute budgets. Test benchmark average against estimated H100 GPU-hours for every method and batch size. Left: math over 400 steps, averaged over three training seeds. Right: terminal agents over 200 steps, with one training seed. Costs count only the GPU time that each part of the training system needs, and dashed lines are the empirical Pareto frontiers of the plotted means (Appendix B.2).

4.4 Compute Efficiency in LLMs

Critics pay off only at small compute budgets. Batch size also sets the cost of training. Figure 3 plots test accuracy against estimated H100 GPU-hours, counting only the GPU time that each part of the training system needs. All runs use the step budgets in Table 1, so these are matched-step comparisons rather than converged endpoints. At small budgets, PPO reaches a given accuracy with less compute. In terminal agents, PPO with $\alpha = 0.5$ at $B = 128$ matches GRPO at $B = 512$ with about a quarter of the compute, 91 versus 342 estimated GPU-hours, and in math, PPO with $\alpha = 0.1$ lies on the Pareto frontier at small batch sizes. At large budgets, the ordering reverses. In math, GRPO at $B = 1{,}024$ reaches the highest test mean with less compute than any larger-batch run, and in terminal agents at $B = 2{,}048$, GRPO matches both PPO variants with 18–29% less compute (Appendix B.2). When samples are scarce, the sampling a critic saves outweighs its cost, but once GRPO reaches parity at large batch sizes, the critic's cost buys little.

Together, these results favor choosing batch size and critic reliance jointly for the task and budget at hand. Horizon and domain set where PPO keeps an advantage, and compute decides whether that advantage is worth paying for. Section 5 measures the alignment and noise scale behind these outcomes.

5 Gradient Measurements in Control Tasks

Training curves show where GRPO catches up to PPO but not why. Section 3 attributes the outcome to each estimator's alignment c and noise scale ν, and this section measures both in control tasks, where an unbiased estimate from about two million trajectories gives a reference policy gradient. The measurements confirm three predictions of Section 3: alignment and noise scale determine cosine

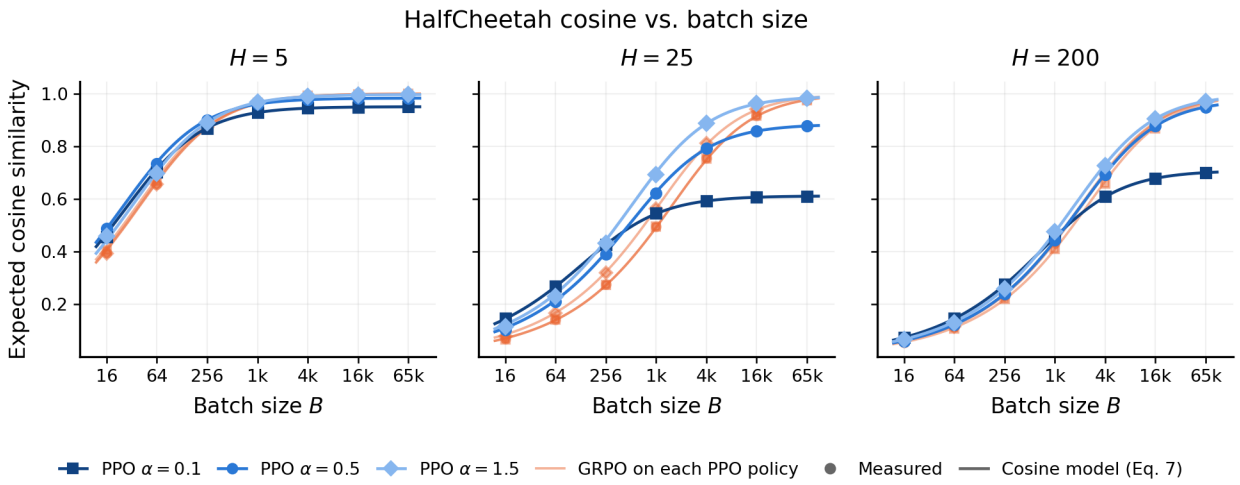


Figure 4. The cosine model tracks measured cosines. Expected cosine to the true policy gradient against batch size in HalfCheetah at update 1,000, before any run's score plateaus. Markers are measurements and lines are Equation 7 with c and ν estimated at $B = 65{,}536$. Each PPO variant is evaluated on its own run trained at $B = 1{,}024$. Faint orange lines evaluate GRPO on each of these three policies, with markers matching the paired PPO variant. Appendix Figure 11 shows all checkpoints in both environments.

similarity at every batch size, critics trade alignment for a lower noise scale, and the noise scale grows with task complexity. We focus on HalfCheetah and bring in Pendulum to compare domains. At fixed checkpoints of each training run, we evaluate PPO's and GRPO's estimators on the same policy, which separates gradient estimation from differences between runs (Appendix D.1).

Alignment and noise scale predict cosine similarity at every batch size. We estimate c and ν once, from batches of 65,536 trajectories, and use Equation 7 to predict the expected cosine at the six smaller batch sizes. Figure 4 shows that the predicted curves pass through the measurements for every estimator and horizon in HalfCheetah. Across all checkpoints and estimators, the median absolute error is 0.003 in HalfCheetah and 0.015 in Pendulum (Appendix Figure 11). Two numbers per estimator are therefore enough to describe how its cosine similarity scales with batch size.

In HalfCheetah, PPO trades alignment for a lower noise scale. Equation 8 predicts a crossover only when PPO pairs lower alignment with a lower noise scale, and Figure 5 shows this tradeoff in HalfCheetah. Across updates 500 to 5,000, every PPO estimator has lower alignment than GRPO's, and heavily bootstrapped PPO loses alignment as training proceeds, while PPO has the lower noise scale in 66 of 90 checkpoint comparisons. Lighter bootstrapping gives up less alignment, so the crossover moves to larger batch sizes. At $H = 25$ in Figure 4, it moves from about 1,500 for $\alpha = 0.1$ to 6,800 for $\alpha = 0.5$ and 65,000 for $\alpha = 1.5$, the same order as in training (Table 2). These fixed-policy measurements explain the direction of the tradeoff.

Noise scale grows with task complexity, in both horizon and domain. Figure 6 shows that GRPO's noise scale rises with horizon in both environments. We report GRPO's noise scale because it involves no critic, and its group baseline removes the reward offset that dominates the Monte Carlo estimator's noise scale (Appendix D.2). From the shortest horizon to the longest, its geometric mean over checkpoints taken before each run plateaus rises from 144 to 4,032 in HalfCheetah and from 13 to 533 in Pendulum.

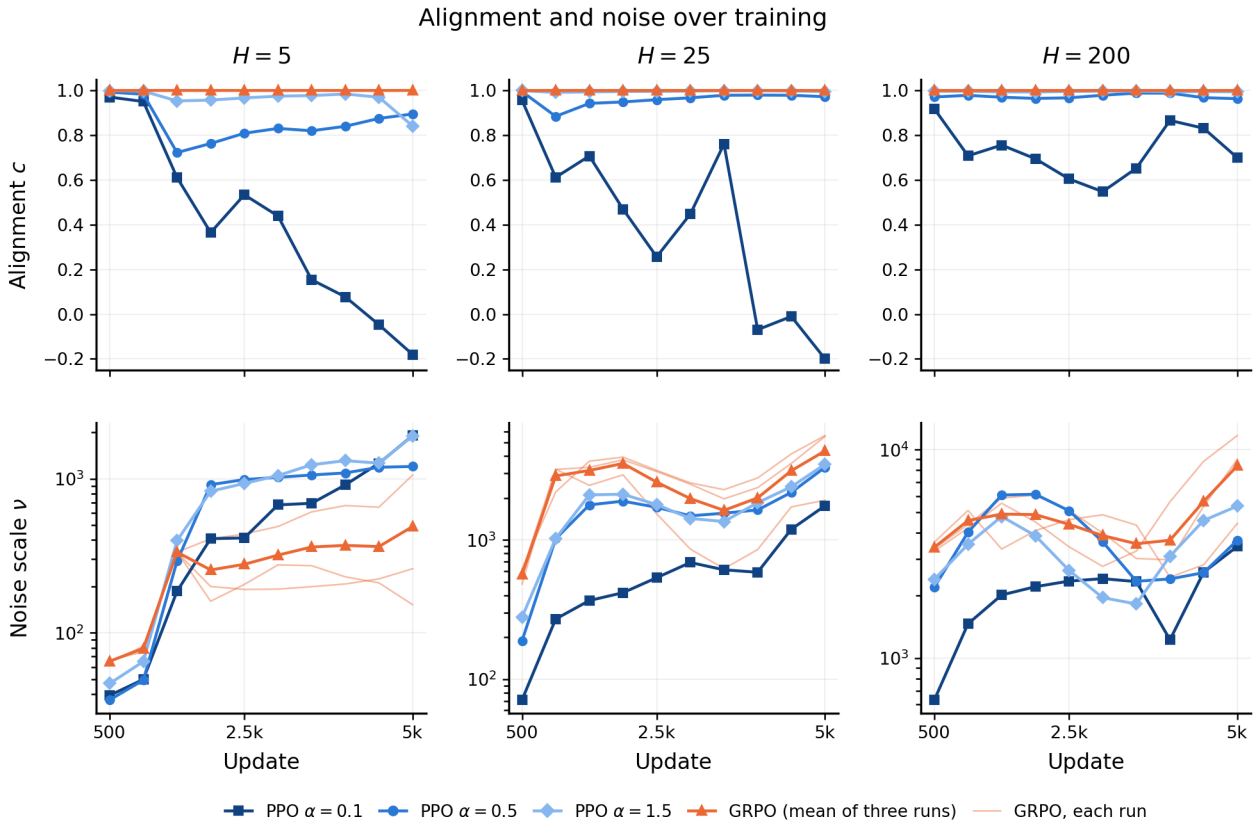


Figure 5. Heavy bootstrapping loses alignment during training. Alignment c (top) and noise scale ν (bottom, log scale) in HalfCheetah. Each PPO variant uses one seed trained at $B = 1{,}024$, measured every 500 updates. Thin GRPO lines evaluate its estimator on each PPO policy, and the thick line is their mean.

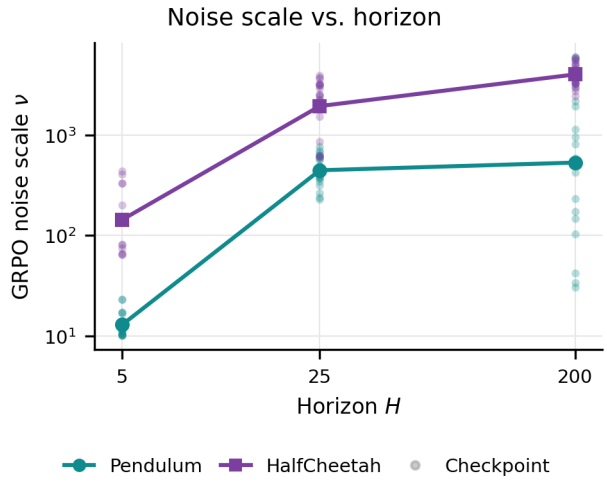


Figure 6. Noise scale grows with task complexity. GRPO's noise scale ν against horizon, evaluated on the policies of the three PPO runs, one seed each. Points are checkpoints from update 500 until each run's score plateau, and lines join geometric means. Pendulum uses training batch sizes 16, 64, and 1,024 at $H = 5$, 25, and 200, and HalfCheetah uses 1,024.

We exclude later checkpoints because the noise scale is inflated near convergence, where the true gradient becomes small. The more complex domain also has the larger noise scale. At every horizon, HalfCheetah's noise scale is 4–11× Pendulum's. HalfCheetah's per-step score bound C is also 2.5–8× larger, consistent with dimensionality entering Proposition 1 through C (Appendix D.2). The environments also differ in dynamics and rewards, so this comparison does not isolate dimensionality. A larger noise scale means GRPO needs a larger batch before its noise term ν / B becomes small, consistent with the later crossovers at longer horizons and in HalfCheetah (Section 4.3).

6 Related Work

Critic-based versus critic-free comparisons. Prior comparisons find PPO stronger than GRPO in longer-horizon control and multi-turn agents [14, 15], with rankings also varying across tasks [20]. Group Policy Gradient finds that increasing parallel rollout counts can favor sampled baselines over PPO in control tasks [21]. EVPO selects learned or sampled baselines using explained variance [52], while Tether mixes them [8]. We examine how the batch-size relationship depends on critic reliance and task complexity, and test it in LLM training through matched comparisons in math and multi-turn terminal agents.

Gradient estimation and batch scaling. Policy-gradient studies examine signal-to-noise ratios and gradient direction [32], and gradient noise scale characterizes diminishing batch-size returns [33]. We use these quantities to characterize alignment crossovers between critic-based and critic-free estimators and examine their relationship to training outcomes. Batch-invariance studies show that learning-rate and policy-update adjustments can preserve sample efficiency over a bounded batch-size range [18, 53]. Prior LLM batch-scaling experiments study PPO and GRPO on different models [18]. IsoCompute studies budget-dependent rollout allocation [19]. These studies show that batch-scaling behavior depends on the training protocol. We compare PPO and GRPO under a shared actor loss to examine how batch size and critic reliance change their relative performance.

7 Discussion and Conclusion

Longer tasks still end in a single reward. LLM tasks have grown from single-turn math answers to agentic trajectories spanning dozens of turns and tens of thousands of tokens, yet under outcome rewards each trajectory still ends in one scalar, often a pass or fail. Our results show the cost of this mismatch. Spreading one reward across more actions raises the noise scale (Proposition 1 and Section 5), and the batch size at which GRPO catches up grows with horizon and task complexity (Section 4.3). A critic is one way to extract more signal from each trajectory, by assigning credit to individual steps rather than spreading one reward across all of them.

Costly rollouts favor learning more from each one. Generation already dominates RL compute. In our terminal-agent runs, it accounts for about 80% of GRPO's estimated GPU-hours, and agentic rollouts add costs beyond the GPU, such as sandboxes, tool and API calls, and LLM judges. When rollouts are expensive and updates comparatively cheap, spending more training compute to learn more from each trajectory becomes attractive. Critics are a natural candidate. Even counting the critic's own training, they already pay off at small compute budgets (Section 4.4), while at large budgets GRPO matches PPO with less compute, so the balance depends on how costly rollouts are relative to updates.

Limitations. We use one starting model within each LLM setting and do not study how actor and critic size affect crossover points. Both LLM settings also change several sources of task complexity at once, so unlike our control experiments they do not isolate the effect of horizon. All runs use fixed step budgets. Our analysis concerns gradient estimates at a fixed policy and critic. It does not guarantee performance over multiple updates, during which PPO and GRPO may follow different training paths, and alignment does not rank methods that intentionally depart from g, such as natural gradients or entropy regularization. Our gradient measurements test the directional bias–variance account in control tasks.

Conclusion. Neither PPO nor GRPO is uniformly better across the settings we study. Our directional bias–variance account suggests why larger batch sizes can favor GRPO's gradient estimates. Training experiments show PPO generally leading at small batch sizes, with GRPO often overtaking heavily bootstrapped PPO or reaching parity with lightly bootstrapped PPO at larger ones, and longer horizons and more complex domains push these changes to larger batch sizes. We recommend that practitioners treat batch size and critic reliance as a joint choice based on their target task and compute budget, comparing both performance and compute cost. As trajectories grow longer and more expensive to collect, this choice becomes increasingly consequential.

References

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[2] Long Ouyang, Jeffrey Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.

[3] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

[4] Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

[5] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[6] Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. What's behind PPO's collapse in Long-CoT? Value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025.

[7] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pages 29557–29590, 2025.

[8] Siddarth Venkatraman, Matthieu Dinot, and Laurence Aitchison. Le Critique: Privileged value functions for LLM reinforcement learning. *arXiv preprint arXiv:2608.16739*, 2026.

[9] Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-Reasoner-Zero: An open source approach to scaling up reinforcement learning on the base model. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[10] Yu Yue, Yufeng Yuan, Qiying Yu, et al. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

[11] Jiashun Liu, Johan S. Obando-Ceron, Han Lu, Yancheng He, Weixun Wang, Wenbo Su, Bo Zheng, Pablo Samuel Castro, Aaron Courville, and Ling Pan. Asymmetric proximal policy optimization: Mini-critics boost LLM reasoning. In *International Conference on Learning Representations*, 2026.

[12] Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026. URL https://arxiv.org/abs/2607.07508.

[13] Penghui Qi, Xiangxin Zhou, and Wee Sun Lee. Best practice critic optimization. *arXiv preprint arXiv:2608.23566*, 2026. URL https://arxiv.org/abs/2608.23566.

[14] Bryan L. M. de Oliveira, Felipe V. Frujeri, Marcos P. C. M. Queiroz, Luana G. B. Martins, Telma W. de L. Soares, and Luckeciano C. Melo. Learning without critics? revisiting GRPO in classical reinforcement learning environments. *arXiv preprint arXiv:2511.03527*, 2025.

[15] Junbo Li, Peng Zhou, Rui Meng, Meet P. Vadera, Lihong Li, and Yang Li. Turn-PPO: Turn-level advantage estimation with PPO for improved multi-turn RL in agentic LLMs. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 6227–6243, 2026. doi: 10.18653/v1/2026.findings-eacl.328.

[16] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

[17] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84 of *PMLR*, pages 1998–2007, 2018.

[18] Ziniu Li, Jinbo Wang, Guanhua Huang, Feiyuan Zhang, Pengbo Li, and Alex Chen. When do larger batches help scale LLM reinforcement learning? *arXiv preprint arXiv:2608.29296*, 2026. URL https://arxiv.org/abs/2608.29296.

[19] Zhoujun Cheng, Yutao Xie, Yuxiao Qu, Amrith Setlur, Shibo Hao, Varad Pimpalkhute, Tongtong Liang, Feng Yao, Zhengzhong Liu, Eric Xing, Virginia Smith, Ruslan Salakhutdinov, Zhiting Hu, Taylor Killian, and Aviral Kumar. IsoCompute playbook: Optimally scaling sampling compute for LLM RL. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 19058–19078, 2026.

[20] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Fei-Fei Li, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

[21] Junhua Chen, Zixi Zhang, Hantao Zhong, and Rika Antonova. Group policy gradient. *arXiv preprint arXiv:2510.03679*, 2025. URL https://arxiv.org/abs/2510.03679.

[22] Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

[23] Yingru Li, Jiawei Xu, Ziniu Li, Jiacai Liu, Wei Liu, Yuxuan Tong, Longtao Zheng, Zhenghai Xue, Yaxiang Zhang, Tianle Cai, Ge Zhang, Qian Liu, and Baoxiang Wang. The optimal token baseline: Variance reduction for long-horizon LLM-RL. *arXiv preprint arXiv:2602.07078*, 2026.

[24] Zhenrong Zhang, Fei Wu, Jun Du, Jianshu Zhang, and Si Wei. ReDiPPO: Reference-guided value calibration and discrepancy-aware token reweighting for mathematical reasoning. *arXiv preprint arXiv:2607.27631*, 2026.

[25] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

[26] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004. URL https://jmlr.org/papers/v5/greensmith04a.html.

[27] Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[28] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling*, 2025.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[31] Jingyuan Liu, Jianlin Su, Xingcheng Yao, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025. URL https://arxiv.org/abs/2502.16982.

[32] Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1811.02553.

[33] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

[34] Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[35] Ronald J. Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[36] Matteo Papini, Matteo Pirotta, and Marcello Restelli. Smoothing policies and safe policy gradients. *Machine Learning*, 111(11):4081–4137, 2022. doi: 10.1007/s10994-022-06232-6.

[37] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2260–2268, 2020.

[38] Xin Zhao, Yongkang Liu, Kuan Xu, Jia Guo, Zihao Wang, Yan Sun, Xinyu Kong, Qianggang Cao, Liang Jiang, Zujie Wen, Zhiqiang Zhang, and Jun Zhou. Small leak can sink a great ship—boost RL training on MoE with IcePop! https://ringtech.notion.site/icepop, 2025. Technical report.

[39] Prime Intellect Team. INTELLECT-3: Technical report. *arXiv preprint arXiv:2512.16144*, 2025.

[40] Mark Towers, Ariel Kwiatkowski, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, et al. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2407.17032.

[41] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115.

[42] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing O1-Preview with a 1.5b model by scaling RL. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

[43] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[44] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

[45] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[46] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022.

[47] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[48] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[49] Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026. URL https://arxiv.org/abs/2606.23321.

[50] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

[51] OpenThoughts-Agent team, Snorkel AI, and Bespoke Labs. OpenThoughts-TBLite: A high-signal benchmark for iterating on terminal agents. https://www.openthoughts.ai/blog/openthoughts-tblite, 2026.

[52] Chengjun Pan, Shichun Liu, Jiahang Lin, Dingwei Zhu, Jiazheng Zhang, Shihan Dou, Songyang Gao, Zhenhua Han, Binghai Wang, Rui Zheng, Xuanjing Huang, Tao Gui, and Yansong Feng. EVPO: Explained variance policy optimization for adaptive critic utilization in LLM post-training. *arXiv preprint arXiv:2604.19485*, 2026.

[53] Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2110.00641.

[54] Fabian Pedregosa. Policy gradients part 1: The REINFORCE estimator. https://fa.bianp.net/blog/2026/policy-gradient/, 2026.

[55] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662.

[56] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[57] Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. SkyRL-v0: Train real-world long-horizon agents via reinforcement learning. https://github.com/NovaSky-AI/SkyRL, 2025.

[58] Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz, Kourosh Hakhamaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Evolving SkyRL into a highly-modular RL framework. https://novasky-ai.notion.site/skyrl-v01, 2025. Blog post.

[59] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

[60] Anonymous. Measuring gradient geometry in language-model RLVR with sketched gradients. https://openreview.net/forum?id=7tjAyMuPu9, 2026. Under review at the Fifteenth International Conference on Learning Representations.

A Theory

A.1 Length-Adaptive GAE

Ignoring the endpoint, the weights of Equation 2 form a geometric distribution over k with mean $1/(1-\lambda)$. Under length-adaptive GAE (Equation 4), this decay scale is αH, and a signal $d = fH$ transitions ahead keeps relative weight

$$\lambda^d = \left(1 - \frac{1}{\alpha H}\right)^{fH} \approx e^{-f/\alpha}. \tag{12}$$

Thus α sets the decay scale relative to the trajectory length. At the first token of a long response, $f \approx 1$, so with terminal-only rewards and $\gamma = 1$ the Monte Carlo return keeps weight about $e^{-1/\alpha}$, the values given in Section 2.1. For comparison, VAPO's $\alpha = 0.05$ gives a limiting first-token weight of approximately 2.1×10^{-9}, while SAO's $\alpha = 1.5$ gives 0.51 [10, 12]. The expression for $\lambda(H)$ lies in $(0, 1)$ only when $\alpha H > 1$. Shorter trajectories require clamping λ, which we set to zero in the control tasks (Appendix B.1).

A.2 Gradient Variance Scaling

Proposition 2 (Monte Carlo gradient variance)**.** Consider trajectories from a fixed policy and task distribution. Suppose $|r(\tau)| \le M_H$ and $\mathbb{E}[\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|^2] \le C$ for every t, where C is independent of H. Then

$$V = \mathrm{Var}(\hat{g}) \le C M_H^2 H. \tag{13}$$

In particular, accumulated rewards with $|r_t| \le r_{\max}$ give $M_H = H r_{\max}$ and $V \le C r_{\max}^2 H^3 = O(H^3)$.

Proposition 1 is the bounded-outcome case $M_H = 1$.

Proof of Propositions 2 and 1. The argument follows the score-function variance calculation of Papini et al. [36, Appendix D, Lemma 28] and the exposition of Pedregosa [54]. Assume a differentiable policy with parameter-independent support and interchangeable differentiation and integration, with rewards, start states, and transitions independent of θ, and write $\mathrm{Var}(Y) = \mathbb{E}\|Y - \mathbb{E}Y\|^2$ and $X_t = \nabla_\theta \log \pi_\theta(a_t \mid s_t)$. The trajectory density satisfies $\nabla_\theta p_\theta(\tau) = p_\theta(\tau) \sum_t X_t$, so differentiating $\mathbb{E}[r(\tau)]$ gives $\mathbb{E}[\hat{g}] = g$. Let $h_t = (s_1, a_1, \ldots, s_t)$ be the history before a_t. Because the policy depends only on s_t and action probabilities integrate to one, $\mathbb{E}[X_t \mid h_t] = \nabla_\theta \int \pi_\theta(a \mid s_t)\, \mathrm{d}a = 0$. For $j < t$, X_j is determined by h_t, so $\mathbb{E}[X_j^\top X_t] = \mathbb{E}[X_j^\top \mathbb{E}[X_t \mid h_t]] = 0$, and the cross terms vanish:

$$\mathbb{E}\left[\left\|\sum_{t=1}^{H} X_t\right\|^2\right] = \sum_{t=1}^{H} \mathbb{E}\|X_t\|^2 \le CH. \tag{14}$$

Bounding the squared return before taking the expectation gives

$$\mathrm{Var}(\hat{g}) \le \mathbb{E}\left[r(\tau)^2 \left\|\sum_{t=1}^{H} X_t\right\|^2\right] \le C M_H^2 H, \tag{15}$$

and independent trajectories divide this bound by B. Taking $M_H = H r_{\max}$ gives $V \le C r_{\max}^2 H^3$, where one factor of H comes from the summed action gradients and two from squaring the accumulated

return. This recovers the known $O(H^3)$ growth [36, 54]. Taking $M_H = 1$ proves the variance bound of Proposition 1. For the noise scale, the same calculation bounds the second moment, $\mathbb{E}\|\hat{g}\|^2 \le CH$, so

$$\nu = \frac{\mathbb{E}\|\hat{g}\|^2 - \|g\|^2}{\|g\|^2} \le \frac{CH}{\|g\|^2} - 1, \tag{16}$$

and $\|g\| \ge m$ gives $\nu \le CH/m^2 - 1 = O(H)$.

Relation to our objectives. The calculation uses summed Monte Carlo gradients. For fixed-length trajectories, averaging over policy tokens divides the estimate by H, which rescales its mean and standard deviation alike and leaves its relative error unchanged when the mean is nonzero. Variable-length token averaging requires accounting for its random denominator. The reward-weighted cross terms $\mathbb{E}[r(\tau)^2 X_j^\top X_t]$ need not vanish, since $r(\tau)$ depends on later actions, but bounding $r(\tau)^2$ first leaves only the unweighted score cross terms, which vanish, so the $O(H)$ bound holds without neglecting any cross terms. The bound also applies to $r(\tau) - \bar{r}(q)$ when rewards and a fixed prompt baseline lie in $[0, 1]$, since such a baseline preserves the expected gradient [25, 26]. It does not cover GRPO's group baseline, which makes gradient contributions within a group dependent, PPO's bootstrapped advantages, or the masking of our shared objective. It motivates varying the batch size with the horizon, but does not by itself predict where PPO and GRPO cross.

A.3 The Cosine Model and the Crossover Batch Size

Setup. Fix the policy, and let $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[r(\tau)]$, so that $g = \nabla_\theta J(\theta)$. The batch gradient $\hat{g}_B$ averages B independent per-trajectory estimates with mean $\mu = \mathbb{E}[\hat{g}]$ and covariance Σ. Write $\hat{g}_B = \mu + \varepsilon$, where $\mathbb{E}[\varepsilon] = 0$ and $\mathrm{Cov}(\varepsilon) = \Sigma/B$. As in the main text, $c = \cos(\mu, g)$ and $\nu = \operatorname{tr}\Sigma/\|\mu\|^2$. We assume g and μ are nonzero and that $\hat{g}$ has finite fourth moments, which the error bound in Equation 19 requires. Both hold in our settings, since rewards, advantages, and response lengths are bounded in the LLM tasks and the Gaussian policy's score has all moments in the control tasks. The cross term has zero mean, so

$$\mathbb{E}\|\hat{g}_B\|^2 = \|\mu\|^2 + \mathbb{E}\|\varepsilon\|^2 = \|\mu\|^2 (1 + \nu/B)\,. \tag{17}$$

Progress per update. Suppose J is β-smooth, so that $|J(\theta+u) - J(\theta) - \langle g, u\rangle| \le \frac{\beta}{2}\|u\|^2$. A step of fixed length, $u = \eta\,\hat{g}_B/\|\hat{g}_B\|$, has $\langle g, u\rangle = \eta\|g\|\cos(\hat{g}_B, g)$ and $\|u\| = \eta$, so

$$\left|\mathbb{E}[\Delta J] - \eta\|g\|\,\mathbb{E}[\cos(\hat{g}_B, g)]\right| \le \tfrac{\beta}{2}\eta^2. \tag{18}$$

At a fixed policy, the factor $\eta\|g\|$ is shared by all estimators, so to first order in η they are ranked by their expected cosine. The ranking is guaranteed when the expected cosines differ by more than $\beta\eta/\|g\|$.

Expected cosine. Let $N = \langle \hat{g}_B, g\rangle/\|g\|$, $D = \|\hat{g}_B\|$, and $\bar{D} = (\mathbb{E}D^2)^{1/2}$, and set $\cos(0, g) = 0$, since GRPO gives $\hat{g}_B = 0$ when every group has identical rewards. The numerator is linear, so $\mathbb{E}N = c\|\mu\|$, and Equation 17 gives $\mathbb{E}N/\bar{D} = c/\sqrt{1+\nu/B}$. Because $N/D = \cos(\hat{g}_B, g)$, the gap between the cosine model and the expected cosine is $\mathbb{E}[\cos(\hat{g}_B, g)(D/\bar{D} - 1)]$. Using $|\cos| \le 1$ and $|x-1| \le |x^2-1|$ for $x \ge 0$,

$$\left|\mathbb{E}[\cos(\hat{g}_B, g)] - \frac{c}{\sqrt{1+\nu/B}}\right| \le \frac{\sqrt{\mathrm{Var}\,\|\hat{g}_B\|^2}}{\mathbb{E}\|\hat{g}_B\|^2}. \tag{19}$$

The cosine model is therefore accurate whenever the squared norm of the batch gradient concentrates, and Appendix D.2 compares it with measured cosines. As $B \to \infty$, $\hat{g}_B \to \mu$ almost surely, and since the cosine is bounded and continuous at $\mu \neq 0$, $\mathbb{E}[\cos(\hat{g}_B, g)] \to c$ by dominated convergence. Like the squared-bias floor, this limit does not depend on ν, so more samples cannot remove it.

Group-mean baseline. With a group-mean baseline that includes the response itself, response i receives advantage $(1 - 1/G)r_i - (1/G)\sum_{j \neq i} r_j$. Given the prompt, the other responses are independent of response i, and its score has zero conditional mean, so the second term contributes nothing in expectation. Hence $\mathbb{E}[\hat{g}] = (1 - 1/G)\, g$, which has $c = 1$ but squared bias $\|g\|^2/G^2$. Squared error therefore counts this estimator as biased even though it points exactly along g [28, 55].

Crossover batch size. Consider two estimators with alignments $c_1 > c_2 > 0$ and noise scales ν_1, ν_2. The main text is the case of GRPO as estimator 1 and PPO as estimator 2. Squaring preserves the order of positive cosines, so under the cosine model, estimator 1 has the larger expected cosine when $c_1^2/(1 + \nu_1/B) > c_2^2/(1 + \nu_2/B)$. Multiplying by $(B + \nu_1)(B + \nu_2)/B > 0$ shows this is equivalent to

$$B > B^*_{\cos} = \frac{c_2^2 \nu_1 - c_1^2 \nu_2}{c_1^2 - c_2^2}. \tag{20}$$

The condition is linear in B, so the two curves cross at most once. Estimator 2 leads for every $B < B^*_{\cos}$ and estimator 1 for every $B > B^*_{\cos}$. If $B^*_{\cos} \leq 0$, estimator 1 leads at every batch size. This proves Equation 8. The same form applies when $c_1 < 1$, as for GRPO with standard-deviation normalization in our LLM settings. Scaling ν_1 and ν_2 by a common factor scales $B^*_{\cos}$ by the same factor, and Proposition 1 allows noise scales to grow linearly with H, so the crossover can move to larger batch sizes at longer horizons.

Parity. When the alignments are close, $c_{\text{PPO}} \approx c_{\text{GRPO}}$, the denominator of Equation 8 is small and $B^*_{\cos}$ is large. At large batch sizes, both expected cosines then approach nearly the same ceiling, so neither estimator has a clear advantage and training can end tied. If the alignments are equal and PPO has the lower noise scale, the two curves approach the same limit without crossing.

Dependent trajectories. GRPO's trajectories share a group baseline and are not independent within a group. The results above hold with the $P = B/G$ independent groups as units. If Σ_G is the covariance of one group's average gradient, then $\text{Cov}(\hat{g}_B) = G\Sigma_G/B$, so defining $\nu_{\text{GRPO}} = G \operatorname{tr}\Sigma_G / \|\mu\|^2$ leaves ν/B unchanged. This ν depends on G. PPO's advantage whitening over the batch also couples trajectories slightly. The cosine model treats them as independent, and Appendix D.2 shows that it still matches the measured cosines of both estimators.

B Experimental Details

B.1 Control Tasks

Table 3 lists the control settings. This section describes the environments and the choices that differ from standard implementations. The control experiments use Python 3.10, PyTorch 2.4.1, Gymnasium 0.29.1, and MuJoCo 2.3.3.

Setting	Value
Environments	Gymnasium Pendulum-v1 and HalfCheetah-v4, fixed horizon $H \in \{5, 25, 200\}$, no early termination
Reward	accumulated per-step reward, rescaled to the score $S(\tau)$ and given only at the terminal state
Batch sizes B	powers of four, 4–4,096 at $H = 5$, 16–16,384 at $H = 25$, and 16–65,536 at $H = 200$
Group size G	GRPO 16 (4 at $B = 4$), PPO 1
Training	5,000 updates, five seeds
Actor	two-layer MLP, width 64, tanh activations, Gaussian mean with one state-independent log standard deviation per action dimension, tanh-squashed and scaled to the action range
Critic (PPO)	separate network with the same architecture and a scalar output
Initialization	orthogonal, gain $\sqrt{2}$ for hidden layers and 0.01 for the action-mean head, log standard deviation 0, critic output weights 0 and bias 0.63 (Pendulum) or 0.1214 (HalfCheetah), identical actors for PPO and GRPO at each seed
Actor update	one unit-norm SGD step per batch, $\eta = 10^{-3}$, no momentum, weight decay, or annealing
Critic update	one full-batch unit-norm step of size 5×10^{-3} after each actor step, on the squared error to $S(\tau)$ at every transition ($\lambda_V = 1$), no value clipping or pretraining
PPO advantages	GAE with $\gamma = 1$ and length-adaptive λ clamped at 0, $\alpha \in \{0.1, 0.5, 1.5\}$, normalized once over the batch's transitions with the population standard deviation plus 10^{-8}
GRPO advantages	return minus the inclusive group mean (Dr. GRPO). GRPO (std) also divides by the group's population standard deviation plus 10^{-8} and zeroes groups whose standard deviation is at most 10^{-8}
Additional variants	PPO with $\lambda = 1$ and GRPO (std) (Appendix C.2)
Evaluation	every 10 updates, four stochastic trajectories from each of 1,024 validation states shared across methods, best score per run

Table 3. Control-task settings.

Pendulum. The policy applies torque to swing up and stabilize a pole. Start states have $\theta_0 \sim \mathcal{U}[-\pi, \pi]$ and $\dot{\theta}_0 \sim \mathcal{U}[-1, 1]$. The actor receives $x_t = [\cos\theta_t,\ \sin\theta_t,\ \dot{\theta}_t/8,\ t/H]$ and outputs one torque $u_t \in [-2, 2]$. With the standard per-step reward $r_t = -(\text{angle}(\theta_t)^2 + 0.1\dot{\theta}_t^2 + 0.001u_t^2)$ and return $R(\tau) = \sum_t r_t$, the score

$$S(\tau) = 1 + \frac{R(\tau)}{H(\pi^2 + 0.1 \cdot 8^2 + 0.001 \cdot 2^2)} \tag{21}$$

lies in $[0, 1]$ by the reward bounds.

HalfCheetah. The policy controls a planar articulated runner, simulated with the Gymnasium MuJoCo model [56]. Start states add $\mathcal{U}[-0.1, 0.1]$ noise to the default positions and $0.1\,\mathcal{N}(0, I)$ noise to the default velocities. The actor observes the eight positions other than absolute horizontal position, all nine velocities, and t/H, with physical features divided by 10 and clipped to $[-1, 1]$, and it outputs six controls in $[-1, 1]$. HalfCheetah therefore has 18 input and 6 action dimensions, against 4 and 1 for Pendulum. The standard per-step reward $r_t = (x_{t+1} - x_t)/0.05 - 0.1\sum_{j=1}^{6} a_{t,j}^2$ rewards forward movement and penalizes control effort. With a per-step reference range of $[-2, 12]$, the score is $S(\tau) = (R(\tau) + 2H)/(14H)$, and we never observe a score outside $[0, 1]$.

Critic inputs and updates. PPO's critic receives the actor's inputs together with the reward accumulated strictly before the current action, under the same affine scaling as the score. This feature summarizes

Setting	Math	Terminal agents
Policy	Qwen2.5-3B (base)	Qwen3.5-4B
Data	10,000 DeepScaleR training and 500 validation questions	14,101 TMax-15K training and 500 validation tasks
Batch sizes B	4, 16, 64, 256, 1,024, 4,096	128, 512, 2,048
Group size G	GRPO $\min(16, \sqrt{B})$, PPO 1	GRPO 16, PPO 1
Steps and seeds	400 steps, three seeds	200 steps, one seed
Updates per step	two actor updates on $2B$ trajectories, the second one step off-policy	one actor update, asynchronous, at most four policy versions stale
Length limits	1,024 prompt and 4,096 response tokens	64 turns, 65,536-token context, 32,768 tokens per turn
Critic learning rate	2.5×10^{-6}, one update per actor update	5×10^{-6}, one update per step
Critic pretraining	200 updates of 16 base-policy trajectories, learning rate 2×10^{-6}	250 updates of 64 initial-policy trajectories, learning rate 2×10^{-6}
Privileged input	reference answer and solution	grading tests and target state
Hardware per run	2–8 H100 GPUs	8–24 H100 GPUs, trainer and inference
Both settings		
Optimizer	AdamW, $\beta = (0.9, 0.999)$, weight decay 0.01, gradient clipping at norm 1.0	
Actor learning rate	10^{-6}, 20-step linear warmup, no decay, no Kullback–Leibler or entropy term	
Sampling	temperature 1.0, top-p 1.0	
GRPO advantages	group mean and standard-deviation normalization	
PPO advantages	GAE with $\gamma = 1$, length-adaptive λ, and $\lambda_V = 1$, whitened over the batch	
Critic	policy backbone with a scalar value head, token-mean squared error, no value clipping, full-attention layers frozen during RL	

Table 4. LLM settings. PPO's critic is initialized from the pretrained critic, which all PPO runs in a setting share.

the relevant trajectory history, allowing a feed-forward critic to predict the final score without recurrence. It excludes the current and future rewards, so it is valid privileged information (Section 2.1), and the actor does not use it. PPO computes GAE with the critic before fitting the critic to the current batch.

Evaluation. Taking each run's best of 500 evaluations raises scores by about 5×10^{-4} over the mean of its last ten, by similar amounts for PPO and GRPO. Because PPO and GRPO share the initial actor and validation states at each seed, we compute the standard errors of Section 4.1 from per-seed differences.

B.2 LLM Tasks

Table 4 lists the LLM settings. We swept the critic learning rate over $\{2.5 \times 10^{-6}, 5 \times 10^{-6}\}$ and the number of critic epochs per batch over $\{1, 2\}$, and use one epoch with the learning rates in the table. The actor learning rate of 10^{-6} follows prior RL fine-tuning of LLMs [27, 42]. GRPO and PPO come from independent runs, so the standard error of their difference is $\mathrm{SE}_{\mathrm{diff}} = \sqrt{\mathrm{SE}_{\mathrm{GRPO}}^2 + \mathrm{SE}_{\mathrm{PPO}}^2}$. The LLM experiments use SkyRL 0.1.0 [57, 58], vLLM 0.20.2 [59], PyTorch 2.11.0, Transformers 5.8.0, and FlashAttention 2.8.3.

Math data and evaluation. We estimate question difficulty by sampling Qwen2.5-3B Instruct five times per DeepScaleR question and keep questions with one to four correct answers, taking 2,500 training and 125 validation questions from each of these four buckets. The prompt asks the model to reason step by step and put the final answer in `\boxed{}`. The reward is 1 for a correct boxed answer and 0 otherwise. During training, we score the validation set with four samples per question every 20 steps. Within 400 steps, we re-score every checkpoint within 2 points of a run's best with 16 samples per question and select the best. At the selected checkpoint, we evaluate MATH-500 (8 samples per question), GSM8K (4), AMC23 (128), Minerva Math (16), and OlympiadBench (8), at temperature 1.0 with a 4,096-token limit, and report mean sampled accuracy. These benchmarks have 500, 1,319, 40, 272, and 581 problems, the last being the problems of OlympiadBench's English text-only math set that have a single final answer.

Terminal environment and evaluation. The agent issues one `bash` tool call per turn in a persistent shell, with a 120-second command timeout and tool outputs truncated to 10,000 characters. An episode ends when the agent submits, or with reward 0 at a format error or when a turn, context, or per-turn token limit is reached. On submission, the task's hidden test script runs in the sandbox, and its score, clipped to $[0, 1]$, is the reward. Model-generated commands run in isolated Modal sandboxes. Relative to Tmax [49], we use our shared objective with standard-deviation normalization instead of mean-centered advantages, vary B instead of fixing it at 256, use $G = 16$ instead of 32, train for 200 instead of 500 steps with a 20-step warmup instead of a constant learning rate, allow 32,768 instead of 16,384 tokens per turn, mask groups with identical rewards out of the loss instead of resampling them, and run the language-model head in bf16 instead of FP32. The 64-turn limit, the 65,536-token context, and the maximum staleness of four match Tmax. Terminal-Bench 2.1 has 89 tasks, and we use 98 of TB-Lite's 100 tasks. Test scores are mean reward with one standard error over eight evaluation passes, each with one attempt per task, and the benchmark average is the mean of the two benchmark scores in each pass. From TB-Lite, we exclude `git-repo-forensics`, whose required repository is missing, and `neutron-submission`, whose missing input data allow an empty report to receive full credit.

Compute. The GPU-hours in Figure 3 estimate the GPU time an efficient implementation of each run needs over its step budget. Both settings count four components. Generation is the GPU time to produce the trained trajectories. Actor training counts forward and backward passes and optimizer steps over the tokens that receive a gradient, so GRPO skips groups whose rewards are identical. Critic training counts the same for PPO's critic, whose forward pass also supplies the values for GAE. Weight synchronization counts copying the updated actor to the inference engines. The masked objective uses the inference engine's log-probabilities, so neither method needs a separate log-probability pass. In math, generation and training alternate on the same GPUs, so we measure generation time directly. In terminal agents, training is asynchronous, and depending on how GPUs are divided between the trainer and the inference engines, either side can sit idle while waiting for the other. We therefore charge every run's generated tokens at 1,583 tokens per second per H100, the highest throughput in an offline replay of 192 of our trajectories under the training limits, which includes the measured 2.6 seconds of sandbox time per turn. Math costs assume the final training configuration, and terminal trainer costs assume balanced micro-batches across data-parallel ranks. Both settings exclude evaluation, checkpointing, idle time, and the one-time critic pretraining that all PPO runs in a setting share, which costs about 0.5 GPU-hours in math and about 50 in terminal agents. In total, we estimate that the LLM experiments used about 21,000 H100 GPU-hours, including critic pretraining, evaluation, tuning, and diagnostics.

B	Method	DeepScaleR	MATH-500	GSM8K	AMC23	Minerva	OlympiadBench	Avg.
4	GRPO	35.0 ± 1.0	57.1 ± 0.8	77.2 ± 0.6	30.4 ± 0.7	13.5 ± 0.7	23.3 ± 0.4	40.3 ± 0.5
	PPO $\alpha = 0.1$	$\mathbf{39.5 \pm 2.1}$	58.8 ± 1.5	80.0 ± 1.7	32.3 ± 1.9	$\mathbf{15.9 \pm 0.7}$	25.0 ± 0.7	42.4 ± 1.2
	PPO $\alpha = 0.5$	39.3 ± 0.6	$\mathbf{58.9 \pm 0.7}$	$\mathbf{81.1 \pm 0.8}$	$\mathbf{33.9 \pm 1.2}$	15.5 ± 0.5	$\mathbf{25.6 \pm 0.2}$	$\mathbf{43.0 \pm 0.6}$
	PPO $\alpha = 1.5$	37.6 ± 0.7	58.1 ± 0.8	78.6 ± 0.3	31.4 ± 1.2	14.5 ± 0.2	25.0 ± 0.8	41.5 ± 0.6
16	GRPO	41.2 ± 0.4	59.8 ± 0.2	81.0 ± 0.4	34.2 ± 0.8	15.7 ± 0.6	26.0 ± 0.1	43.3 ± 0.3
	PPO $\alpha = 0.1$	$\mathbf{45.5 \pm 1.1}$	60.8 ± 0.6	$\mathbf{82.5 \pm 0.3}$	35.5 ± 0.7	$\mathbf{17.0 \pm 0.4}$	$\mathbf{27.5 \pm 0.5}$	44.7 ± 0.3
	PPO $\alpha = 0.5$	44.3 ± 0.9	$\mathbf{62.2 \pm 0.2}$	82.3 ± 1.0	$\mathbf{37.7 \pm 0.2}$	16.4 ± 1.0	27.4 ± 0.2	$\mathbf{45.2 \pm 0.4}$
	PPO $\alpha = 1.5$	43.2 ± 0.4	60.9 ± 0.2	81.5 ± 0.3	35.8 ± 0.6	16.4 ± 0.5	27.2 ± 0.6	44.3 ± 0.3
64	GRPO	45.9 ± 0.2	61.8 ± 0.3	82.5 ± 0.3	35.5 ± 1.1	16.7 ± 0.4	28.1 ± 0.1	44.9 ± 0.3
	PPO $\alpha = 0.1$	47.2 ± 0.6	62.6 ± 0.3	$\mathbf{83.1 \pm 0.2}$	37.3 ± 0.4	$\mathbf{18.1 \pm 0.1}$	28.4 ± 0.6	$\mathbf{45.9 \pm 0.2}$
	PPO $\alpha = 0.5$	$\mathbf{47.9 \pm 0.1}$	63.2 ± 0.2	82.1 ± 0.5	$\mathbf{37.4 \pm 0.8}$	17.0 ± 0.1	28.7 ± 0.4	45.7 ± 0.2
	PPO $\alpha = 1.5$	47.3 ± 0.6	$\mathbf{63.3 \pm 0.1}$	83.0 ± 0.2	36.8 ± 1.3	17.2 ± 0.3	$\mathbf{28.8 \pm 0.2}$	45.8 ± 0.3
256	GRPO	50.2 ± 0.4	63.5 ± 0.0	83.7 ± 0.5	38.0 ± 0.4	17.6 ± 0.2	29.5 ± 0.3	46.5 ± 0.1
	PPO $\alpha = 0.1$	50.7 ± 0.2	63.6 ± 0.2	83.4 ± 0.4	36.9 ± 1.5	17.2 ± 0.2	29.1 ± 0.4	46.0 ± 0.3
	PPO $\alpha = 0.5$	$\mathbf{52.3 \pm 0.5}$	$\mathbf{64.7 \pm 0.6}$	$\mathbf{84.3 \pm 0.4}$	38.6 ± 0.6	17.7 ± 0.3	$\mathbf{31.0 \pm 0.4}$	$\mathbf{47.3 \pm 0.1}$
	PPO $\alpha = 1.5$	51.3 ± 0.1	63.7 ± 0.3	83.5 ± 0.4	$\mathbf{39.9 \pm 1.2}$	$\mathbf{18.0 \pm 0.4}$	30.8 ± 0.3	47.2 ± 0.4
1,024	GRPO	55.1 ± 0.5	$\mathbf{66.2 \pm 0.3}$	84.4 ± 0.1	41.0 ± 0.7	$\mathbf{18.1 \pm 0.2}$	$\mathbf{31.8 \pm 0.1}$	$\mathbf{48.3 \pm 0.1}$
	PPO $\alpha = 0.1$	52.6 ± 0.2	63.8 ± 0.5	83.8 ± 0.4	37.1 ± 0.8	17.1 ± 0.3	29.9 ± 0.3	46.3 ± 0.2
	PPO $\alpha = 0.5$	$\mathbf{55.6 \pm 0.2}$	65.0 ± 0.3	84.5 ± 0.3	$\mathbf{41.4 \pm 0.4}$	17.9 ± 0.3	31.6 ± 0.4	48.1 ± 0.1
	PPO $\alpha = 1.5$	55.0 ± 0.3	65.5 ± 0.2	$\mathbf{85.1 \pm 0.3}$	39.6 ± 0.4	17.5 ± 0.2	31.7 ± 0.3	47.9 ± 0.1
4,096	GRPO	56.8 ± 0.2	65.8 ± 0.4	84.8 ± 0.5	40.2 ± 0.4	$\mathbf{17.6 \pm 0.1}$	32.2 ± 0.4	48.1 ± 0.2
	PPO $\alpha = 0.1$	54.1 ± 0.2	64.1 ± 0.3	84.2 ± 0.3	39.3 ± 0.7	17.4 ± 0.3	29.4 ± 0.4	46.9 ± 0.2
	PPO $\alpha = 0.5$	56.9 ± 0.3	$\mathbf{66.5 \pm 0.1}$	$\mathbf{85.1 \pm 0.2}$	40.5 ± 0.7	17.5 ± 0.3	31.5 ± 0.2	$\mathbf{48.2 \pm 0.2}$
	PPO $\alpha = 1.5$	$\mathbf{57.5 \pm 0.2}$	65.6 ± 0.1	84.5 ± 0.3	$\mathbf{40.9 \pm 0.2}$	16.7 ± 0.1	$\mathbf{32.6 \pm 0.2}$	48.1 ± 0.1
–	Qwen2.5-3B (base)	15.3	29.2	42.0	12.7	5.8	10.2	20.0

Table 5. Full math results. Scores at each run's best checkpoint within 400 steps (mean $\pm$ one standard error over three seeds, accuracy in %). DeepScaleR selects checkpoints. Avg. is the five-benchmark test average in Figure 2. Bold marks the highest mean at each batch size, including ties after rounding. The last row is the base model.

C Additional Results

C.1 Full LLM Results

Tables 5 and 6 list every math and terminal-agent score, including the validation scores.

C.2 Other Control Methods

Figure 7 adds two methods to Figure 1. PPO with $\lambda = 1$, which does not bootstrap, has a mean score within 0.004 of PPO with $\alpha = 1.5$ at every batch size. GRPO (std) normalizes each group's advantages by its standard deviation. Its mean score is within 0.003 of GRPO at every batch size except in Pendulum at $H = 25$.

With binary rewards and $G = 16$, a group with mixed outcomes has a reward standard deviation of at least 0.24, so normalization rescales groups by a bounded factor. Continuous returns have no such floor. Figure 8 shows the spread of group standard deviations along the GRPO run with $B = 1{,}024$, measured with 16 trajectories from each of 512 start states every 500 updates. In HalfCheetah and in Pendulum at $H = 200$, the 95th percentile stays within about $3\times$ the 5th, so normalization rescales groups by similar factors. In Pendulum at $H = 25$, the ratio grows from 22 to 657 over training, and at $H = 5$ it reaches

B	Method	TMax-15K	TB 2.1	TB-Lite	Avg.
128	GRPO	52.7 ± 0.4	17.3 ± 0.8	40.9 ± 1.0	29.1 ± 0.7
	PPO $\alpha = 0.5$	56.1 ± 0.4	$\mathbf{19.0 \pm 1.2}$	$\mathbf{48.6 \pm 1.4}$	$\mathbf{33.8 \pm 0.9}$
	PPO $\alpha = 1.5$	$\mathbf{58.1 \pm 0.9}$	17.3 ± 0.4	46.7 ± 1.0	32.0 ± 0.5
512	GRPO	61.5 ± 0.1	19.2 ± 0.6	46.6 ± 0.9	32.9 ± 0.5
	PPO $\alpha = 0.5$	$\mathbf{66.6 \pm 0.2}$	$\mathbf{23.6 \pm 0.6}$	51.3 ± 1.5	37.5 ± 0.9
	PPO $\alpha = 1.5$	62.3 ± 0.5	22.3 ± 1.1	$\mathbf{55.2 \pm 1.0}$	$\mathbf{38.8 \pm 0.6}$
2,048	GRPO	$\mathbf{70.8 \pm 0.8}$	22.6 ± 1.2	55.2 ± 0.7	38.9 ± 0.6
	PPO $\alpha = 0.5$	69.7 ± 0.8	21.6 ± 0.9	$\mathbf{56.5 \pm 1.4}$	$\mathbf{39.1 \pm 0.9}$
	PPO $\alpha = 1.5$	69.5 ± 0.4	$\mathbf{22.8 \pm 1.1}$	54.3 ± 1.2	38.5 ± 0.7
–	Qwen3.5-4B (base)	35.0	9.6	26.4	18.0
–	TMax-4B	–	16.7	43.8	30.3

Table 6. Full terminal-agent results. Scores with one training seed (mean $\pm$ one standard error over evaluations, in %). All columns use the final checkpoint (step 200). TMax-15K is the validation set, scored with avg@3. Terminal-Bench 2.1 (TB 2.1) and TB-Lite report mean reward over eight attempts per task, and Avg. is their mean. Bold marks the highest mean at each batch size. TMax-4B is the released TMax model [49].

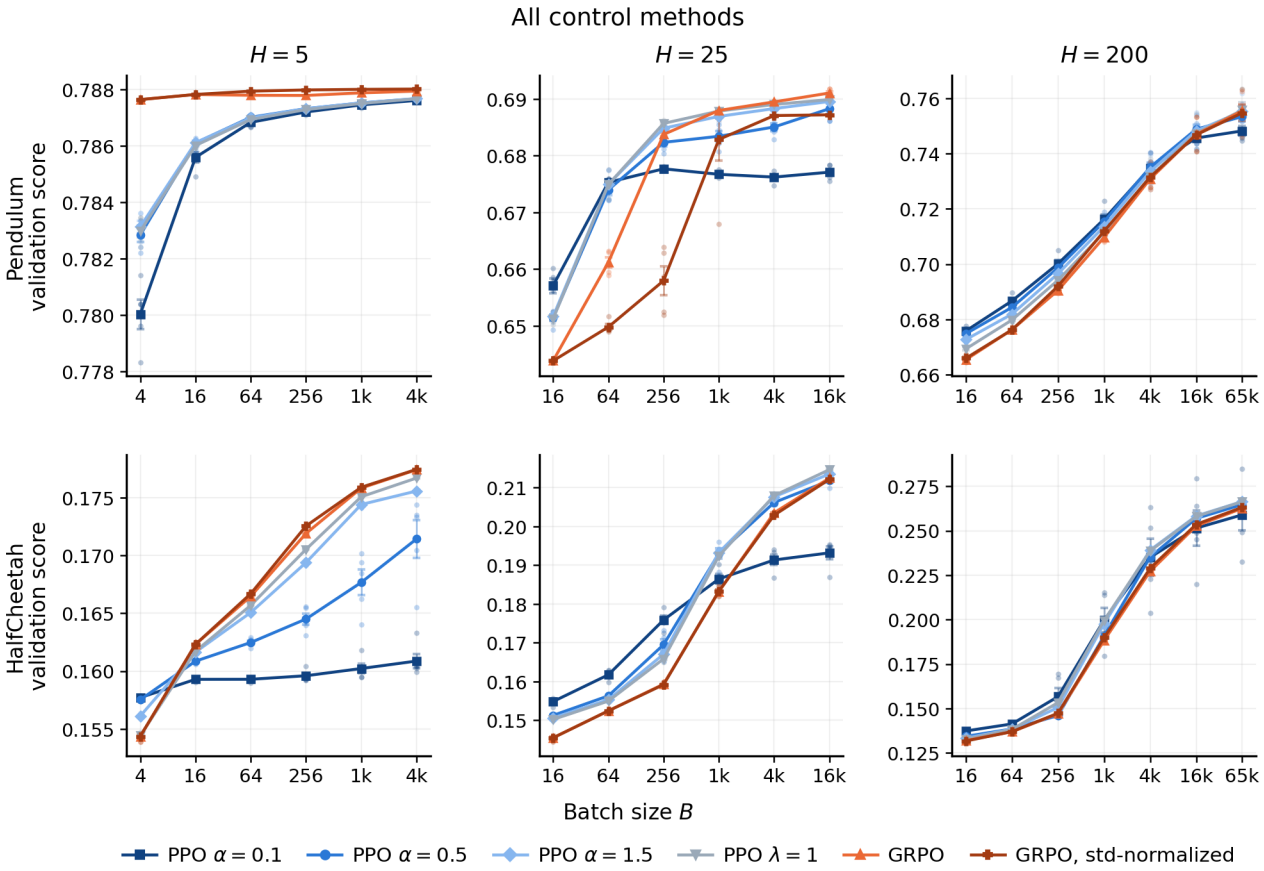


Figure 7. All control methods. Best validation score for every control method, as in Figure 1 with PPO $\lambda = 1$ and GRPO (std) added.

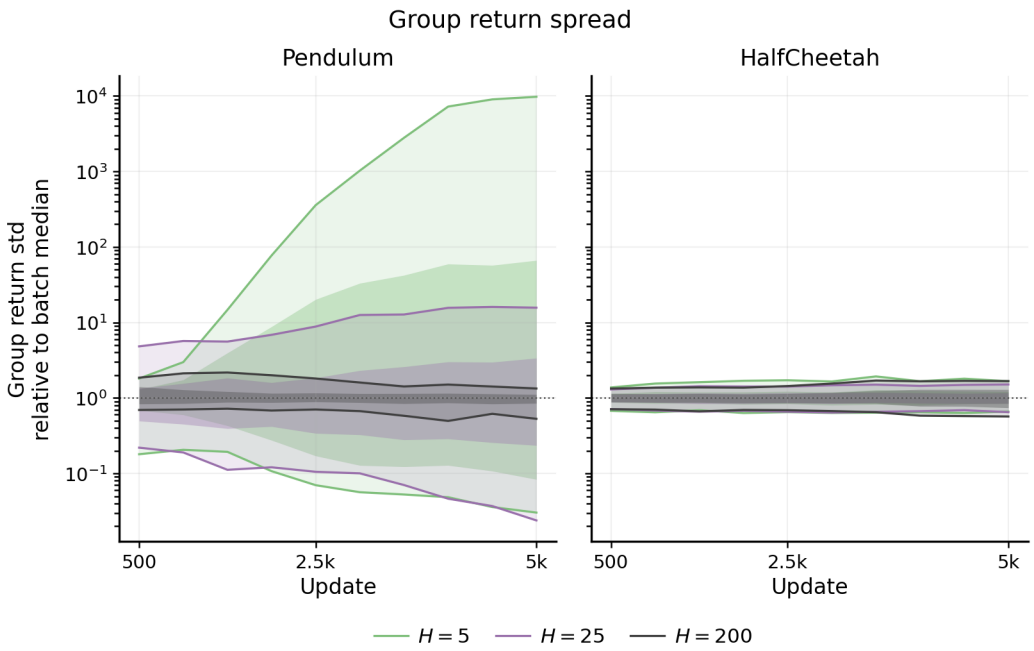


Figure 8. Group return spread in control tasks. Group return standard deviation relative to the batch median over training, for the GRPO run with $B = 1{,}024$. Dark bands span the 25th to 75th percentiles and light bands the 5th to 95th.

3×10^5. Normalization gives every group advantages of unit scale, so the start states whose returns barely vary receive as much weight as those with the most to learn. At $H = 5$, both variants score within 0.001 of their $B = 4{,}096$ score already at $B = 4$, so the reweighting costs nothing. At $H = 25$, the task is still improving, and GRPO (std) falls behind.

C.3 Privileged and Unprivileged Critics

The privileged critic explains more of the return variance than an unprivileged critic that observes only the actor's inputs, and the gap grows along the response (Figure 9). The two critics are nearly identical near the start of a response, while at the last token the privileged critic explains 0.97 of the variance after step 100, against 0.53. Final accuracy is similar. On DeepScaleR validation, the two critics differ by at most 0.9 points at every batch size (Figure 10), and they are within 0.3 points on the five-benchmark average. These comparisons cover math with $\alpha = 1.5$, and the explained-variance curves come from one seed at $B = 1{,}024$.

D Gradient Measurements

This appendix details the control-task gradient measurements of Section 5 and the group-size check of Section 4.1. Each measurement compares an estimator's batch gradient $\hat{g}_B$ with a reference estimate of the policy gradient g at a frozen checkpoint.

D.1 Control Tasks: Method

Checkpoints. In HalfCheetah, for each horizon $H \in \{5, 25, 200\}$ and each PPO variant, we use that variant's training run with $B = 1{,}024$ (seed 1), with its actor and critic, at every 500 updates. GRPO

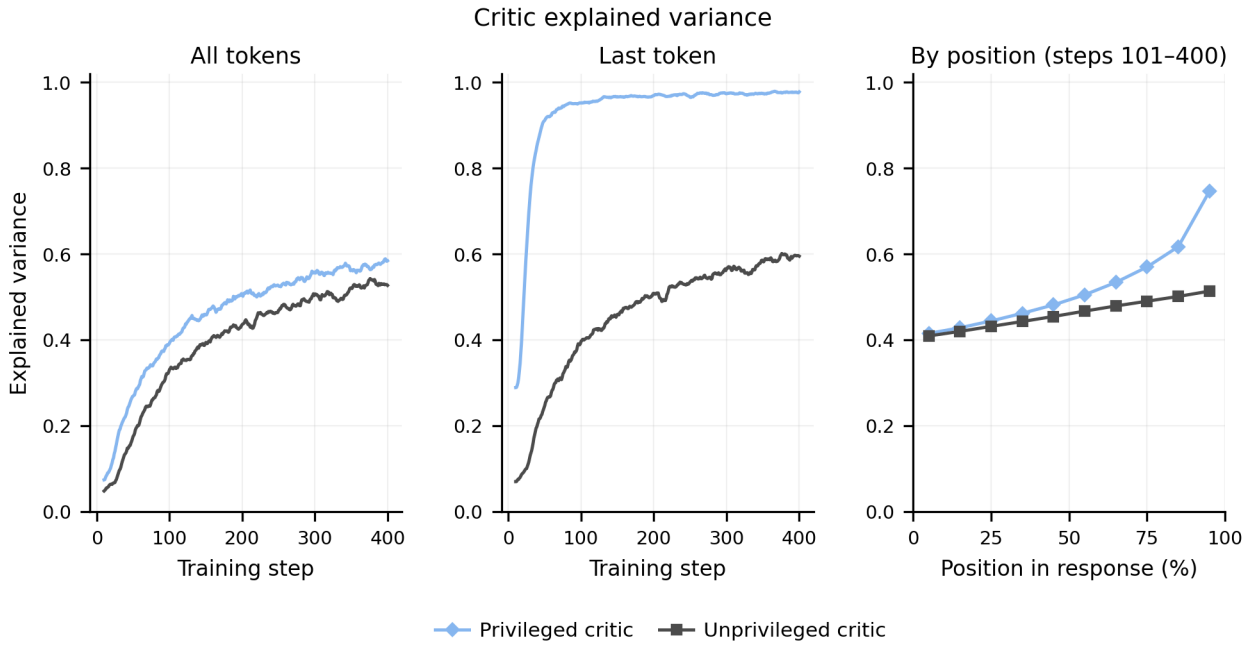


Figure 9. Critic explained variance in math. Critic explained variance for PPO with $\alpha = 1.5$ and $B = 1{,}024$ in math, over all policy tokens (left) and at the last token (middle) as 10-step rolling means, and by position in the response over steps 101–400 (right).

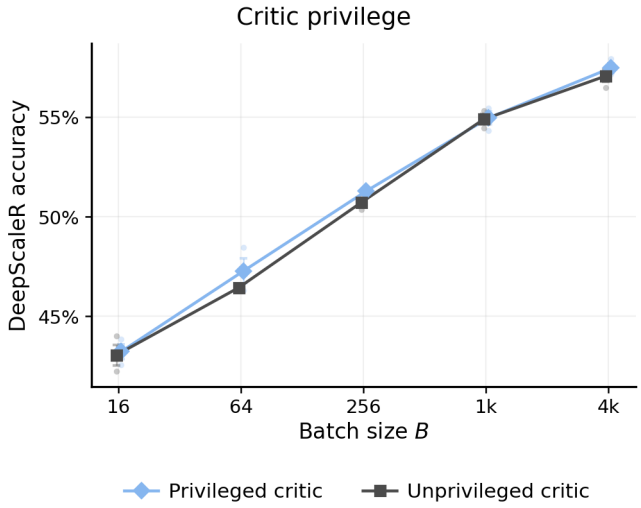


Figure 10. Privileged vs. unprivileged critic. Best DeepScaleR accuracy within 400 steps for PPO with $\alpha = 1.5$ and a privileged or unprivileged critic. Mean and one standard error over three seeds.

Environment	H	Training B	GRPO ν	C	$\|g\|^2$	$V/(CH)$
Pendulum	5	16	13	13	2.0×10^{-4}	0.55
	25	64	447	21	1.1×10^{-3}	0.36
	200	1,024	533	42	6.0×10^{-3}	0.36
HalfCheetah	5	1,024	144	100	2.0×10^{-4}	0.018
	25	1,024	1,949	120	3.0×10^{-4}	0.020
	200	1,024	4,032	105	4.0×10^{-4}	0.018

Table 7. Gradient statistics by environment and horizon. Geometric means over checkpoints from update 500 until each run's score plateau. C is the largest per-step score norm of Proposition 1, and V is the variance of the Monte Carlo estimator of Equation 9.

has no critic, so we evaluate its estimator on the same policies as each PPO variant. In Pendulum, runs at $B = 1{,}024$ reach their score plateau after a median of about 1,200 updates at $H = 5$ and 2,300 at $H = 25$, after which $\|g\|$ shrinks and ν rises sharply. We therefore measure Pendulum on runs that are still improving, trained at $B = 16$, 64, and 1,024 at $H = 5$, 25, and 200. A run's score plateau is the first update at which a centered 201-update moving average of its training score reaches 80% of its total gain.

Reference gradient. We draw a panel of 256 start states from the training start distribution with a fixed seed. For each state, we sample 8,192 trajectories in two halves of 4,096 and compute a leave-one-out policy gradient per state and half, using the score of the pre-tanh Gaussian action. The reference is the mean over states and halves, with reference noise removed using the two halves. We compute exact per-trajectory gradients over all actor parameters (5,772 in HalfCheetah), without sketching. Across all checkpoints, the halves agree with cosine at least 0.989 in Pendulum and 0.986 in HalfCheetah, so the reference has cosine at least 0.996 with the panel's true gradient.

Candidates and batches. We draw 128 independent pools, each with 256 fresh trajectories from every panel state, and take up to 16 disjoint batches from each pool at each batch size. GRPO subtracts the group mean without standard-deviation normalization, and PPO uses GAE with the checkpoint critic and length-adaptive λ (Equation 4, floored at 0), whitened over the BH transitions of the batch, as in training. Both estimators use the same batches of $P = B/16$ groups of $G = 16$ trajectories, for B from 16 to 65,536, so they differ only in their advantages. Up to $B = 4{,}096$, each group has a different state, and above it each of the 256 states supplies $B/4{,}096$ groups, each with its own trajectories. PPO trains with $G = 1$, which would need more distinct states than the panel holds. In HalfCheetah, PPO's expected cosine at $G = 16$ is within 0.01 of its value at $G = 1$ at every batch size of the group-size check (Appendix D.3).

Metrics. We report the mean cosine to the reference, whose noise is negligible here. Both c and ν come from the 128 batches of $B = 65{,}536$, one per pool. The alignment c is the cosine of their mean with the reference, and ν is B times their normalized variance. The standard error of c comes from leaving out each of eight groups of pools in turn.

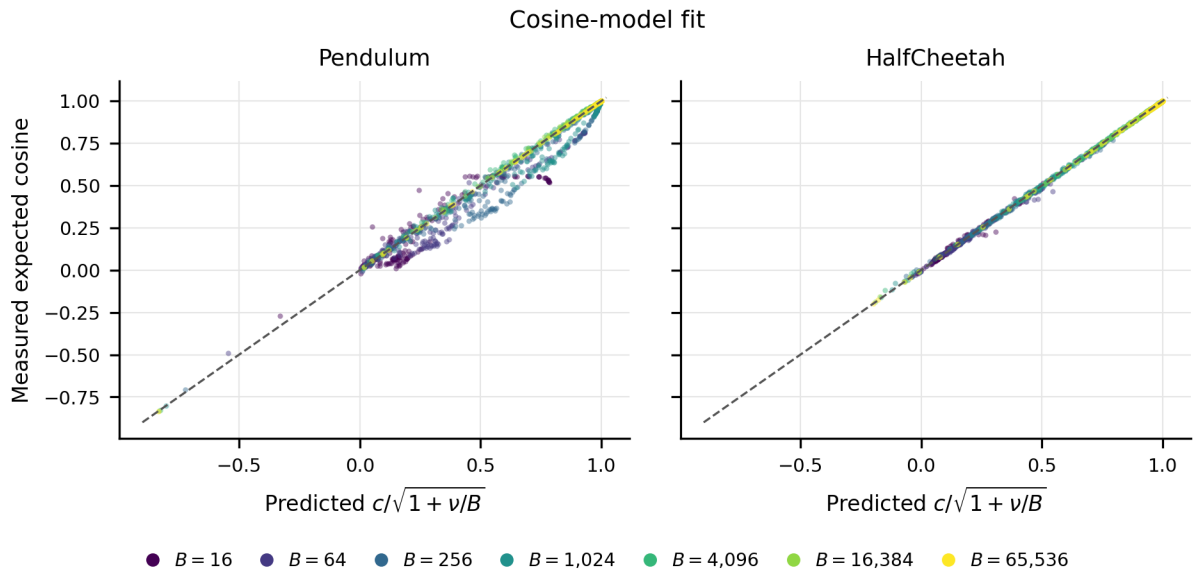


Figure 11. Cosine-model fit across checkpoints. Each point is one checkpoint, estimator, and batch size from updates 0 to 5,000, with 1,302 points per environment. At $B < 65{,}536$, where the prediction is out of sample, the median absolute error is 0.015 in Pendulum and 0.003 in HalfCheetah. The dashed line is equality.

D.2 Cosine Model and Horizon Bound

Cosine model. Figure 11 covers GRPO and each run's own PPO estimator. The median absolute error falls from 0.047 at $B = 16$ to 0.006 at $1{,}024 \leq B \leq 16{,}384$ in Pendulum, and from 0.004 to 0.002 in HalfCheetah, since the squared norm of the batch gradient concentrates less at small batch sizes (Equation 19).

Horizon bound. At each checkpoint, we draw 4,096 fresh trajectories, 16 from each panel state, and compute the Monte Carlo estimator of Equation 9 with raw rewards. We estimate C as the largest per-step $\mathbb{E}\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|^2$ and V as $\mathbb{E}\|\hat{g}\|^2 - \|g\|^2$. Both halves of Equation 10 hold at every checkpoint. The bound's slack, $V/(CH)$ in Table 7, comes mostly from rewards below one.

Why GRPO's noise scale. Without a baseline, every trajectory's gradient is scaled by its full reward, so the Monte Carlo estimator's noise scale mostly reflects the mean reward. At Pendulum's initial policy with $H = 5$, it is about 7×10^4, against 10 for GRPO. GRPO's noise scale also does not change when rewards are rescaled, whereas PPO's reflects its critic's error, which differs between environments.

D.3 Group Size

Figure 12 varies the group size at fixed batch size for GRPO and PPO with $\alpha = 1.5$. In math, we measure steps 0 and 140 of the PPO $\alpha = 1.5$ run with $B = 1{,}024$, following the sketched-gradient methodology of Anonymous [60]. Per-response gradients over all model parameters are compressed with CountSketch (65,536 buckets) and compared with a reference estimated from a larger, disjoint set of responses on a panel of 1,024 DeepScaleR training questions, using independent halves to remove sampling noise from norms and cosines. In control, we use the exact measurement of Appendix D.1 at updates 500 and 3,000 of each $\alpha = 1.5$ run, with Pendulum at $H = 25$ and $H = 200$, trained at $B = 64$ and $B = 1{,}024$, and

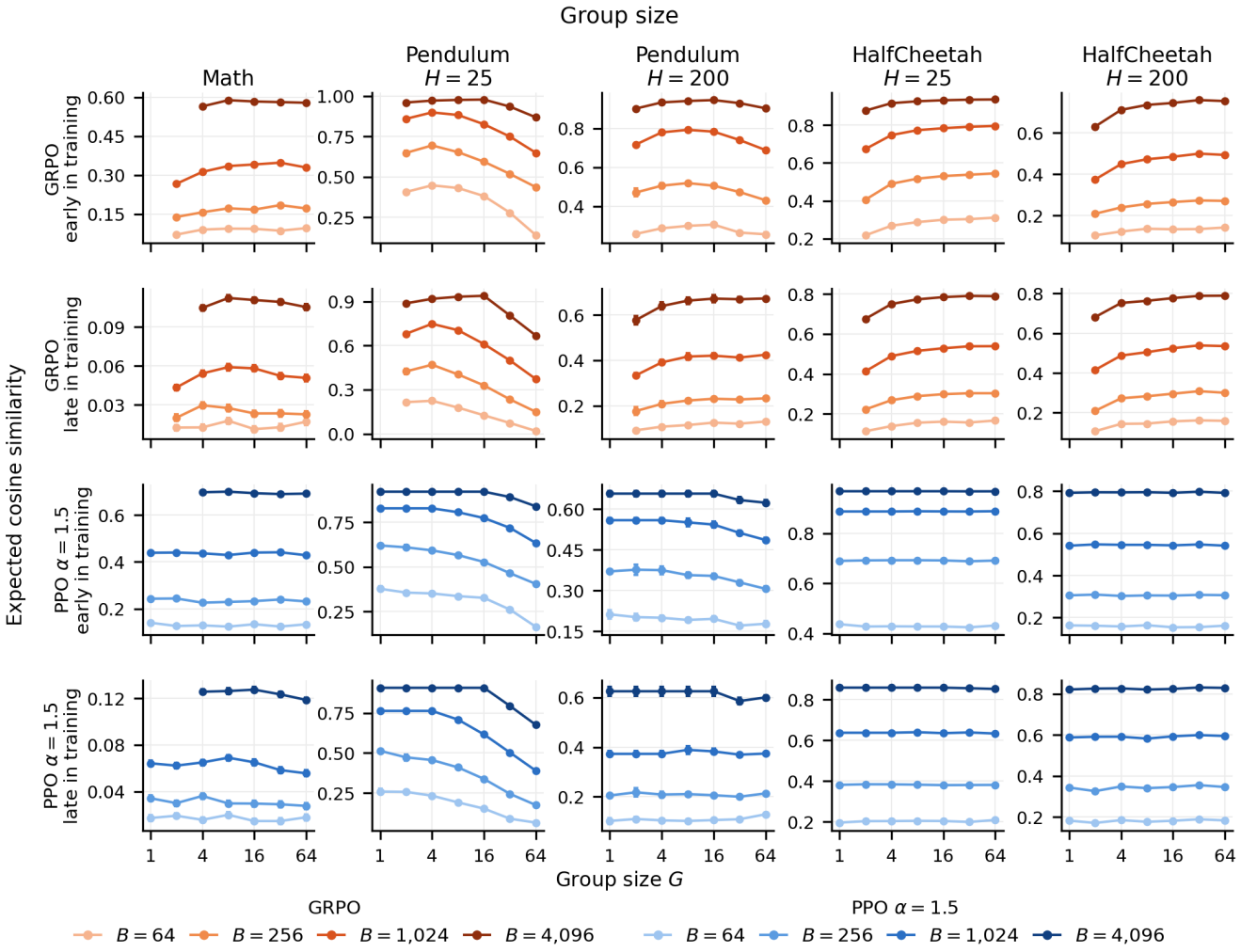


Figure 12. Expected cosine against group size. Cosine to the policy gradient against group size G for GRPO and PPO with $\alpha = 1.5$, with one line per batch size B. Early in training is step 0 in math and update 500 in control, and late in training is step 140 and update 3,000. Math uses GRPO with standard-deviation normalization and control uses Dr. GRPO, as in training. Error bars show one standard error over batches.

HalfCheetah at both horizons, trained at $B = 1{,}024$. Each batch splits into B/G prompts or start states with G trajectories each. Math batches cannot exceed the 1,024 distinct prompts of its panel, so PPO with $G < 4$ is not measured at $B = 4{,}096$.

PPO's expected cosine changes little with group size or is highest at small G in every setting, and at $B = 4{,}096$ its smallest measured group size is within 0.01 of its best. For GRPO, $G = 16$ is within 0.015 of the best group size at $B = 4{,}096$, the largest batch size we measure, in every setting, and within 0.02 at every batch size in math and HalfCheetah. Where $G = 16$ is not GRPO's best, a better group size could only raise GRPO's expected cosine, so the batch sizes at which GRPO crosses over are, if anything, conservative.