CRITICAL BATCH SIZE FOR LLM POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy optimization has become the dominant post-training technique for large language models, but it is computationally expensive. In pretraining and supervised learning more broadly, the critical batch size (CBS), which is the batch size beyond which further increases yield diminishing returns in training steps, is well studied and guides how compute is parallelized at scale. No analogous understanding exists for policy optimization, where the question is harder: the objective changes with the policy, sequences generated from the same prompt are not independent, and the batch size itself decomposes into the number of prompts and the number of rollouts per prompt. We develop a theoretical and empirical framework to study the CBS of policy optimization, focusing on GRPO in the on-policy regime. Theoretically, we present a noise model that decomposes the gradient noise of GRPO into a prompt-level and a rollout-level term, and predicts that adding prompts reduces both while adding rollouts reduces only one. Empirically, for GRPO on mathematical reasoning, we observe near-linear reductions in optimizer steps up to 16K sequences per step when scaling prompts. Together, we give concrete guidance on scaling prompts, rollouts, and learning rate, while shedding further light on GRPO batch scaling dynamics.

1 INTRODUCTION

As more compute is devoted to policy optimization for large language model (LLM) post-training, a central question is how to best utilize this growing compute during training. One way to effectively use compute is to increase parallelism via a larger batch size. In the ideal linear-scaling regime, doubling the batch size doubles the amount of data processed per optimization step and halves the number of steps required to attain a given performance level. However, beyond a certain batch size, each additional sample contributes less independent information about the update direction, and hence the number of steps no longer decrease linearly with batch size. This point is referred to as the *critical batch size* (CBS) (McCandlish et al., 2018; Shallue et al., 2019; Jain et al., 2018): the largest batch size that provides near-linear reductions in optimization steps (and thus, the serial wall-clock training time) without sacrificing sample or compute efficiency.

In this work, we study the CBS for LLM policy optimization, with a particular focus on GRPO (Shao et al., 2024). While CBS has been studied extensively for supervised learning and pretraining (Hilton et al., 2023; Zhang et al., 2025; Merrill et al., 2025), analyzing the CBS for policy optimization poses distinct challenges. Foremost, the batch size is composed of multiple axes; namely, the number of prompts and the number of generations per prompt. Thus, different allocations of the same generation budget can yield different CBS behavior. To that end, we ask: *how far can post-training be parallelized along each batch size dimension before returns diminish?*

We begin by developing a noise-scale model of the CBS for GRPO-style algorithms. We build on the framework of McCandlish et al. (2018) to decompose gradient noise into variation across prompts (B) and across rollouts per prompt (K). We derive expressions for the local CBS along both of these batch-size axes. Under this model, we show that increasing the B under fixed K reduces both sources of noise, whereas increasing K under fixed B reduces only within-prompt noise. Further, under a fixed total generation budget $N = B \times K$, scaling B affords a larger step size than scaling K. Hence, the number of prompts is a more effective scaling dimension for batch size.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

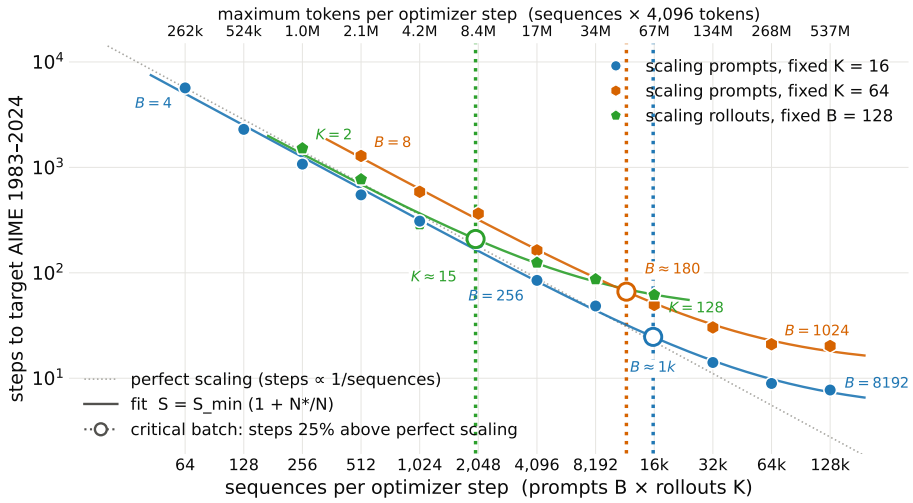


Figure 1: **Critical batch size for GRPO.** We scale the batch along two axes: scaling prompts B at a fixed number of rollouts per prompt ($K = 16$ and $K = 64$), and scaling rollouts K at a fixed number of prompts ($B = 128$). The dotted line is perfect scaling, where doubling the batch halves the number of steps; solid lines are fits of the form $S = S_{\min}(1 + \mathcal{N}_{\text{crit}}/N)$, and circles mark the fitted critical batch size $\mathcal{N}_{\text{crit}}$, beyond which returns diminish. Along the prompt axis, steps fall almost linearly up to ∼16K sequences (∼64M tokens) per step at $K = 16$. Along the rollout axis, returns diminish an order of magnitude earlier, at ∼2K sequences. At a matched number of sequences, batches with larger B and fewer K reaches the target in fewer steps.

Next, we empirically study CBS for GRPO in a controlled setting. We train Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) on the DAPO (Yu et al., 2025) training set and perform evaluation on the AIME 1983–2024 dataset (Veeraboina, 2023). We measure CBS across a variety of batch size settings: scaling total number of sequences in a batch N up to 128K, number of prompts B up to 8192, and number of rollouts per prompt K up to 128. We carefully tune the learning rate for all our runs and also study the effect of KL regularization. In all, we perform more than 469 GRPO runs for this study, spending about 27K GPU hours. Our results present conclusive practical implications. Most notably, we find that GRPO is highly parallizable. That is, we could scale our batch size to ∼65M tokens per step while getting linear returns in training time. To our knowledge, this is considerably larger than batch sizes commonly used to post-train LLMs (Yu et al., 2025; Mistral AI, 2025; Wang et al., 2025). Our results also shed light on effective batch scaling axes, learning rate scaling, KL regularization, and advantage estimation.

Overall, we present the following key contributions:

(i) **A noise-scale model of the CBS for GRPO (§3).** Building on McCandlish et al. (2018), we separate gradient noise across prompts and within prompts, yielding expressions for the local CBS along both batch-size dimensions (Theorem 1). Adding prompts reduces both sources of noise while adding rollouts reduces only one, making it a more effective batch scaling dimension.

(ii) **Measuring the CBS along prompts and rollouts (§4.2).** Scaling prompts yields near-linear reductions in optimizer steps up to ∼16K sequences (∼65M tokens) per step, demonstrating substantial scope for parallelism without sacrificing sample efficiency. Scaling rollouts per prompt reaches diminishing returns at roughly one-tenth the sequence batch size, and the CBS in sequences is nearly unchanged between 16 and 64 rollouts per prompt.

(iii) **Learning rate scaling along each axis (§4.3).** The optimal learning rate grows with the square root of the number of prompts, as in supervised learning for Adam, but remains nearly constant with rollouts. We validate that the rule from $K = 16$ transfers to $K = 64$ without further tuning.

(iv) **Further characterization of GRPO dynamics (§4.4, §5).** Stronger KL regularization widens the range of stable learning rates but leaves the CBS unchanged (§4.4). Estimating advantages from additional rollouts improves early training at small rollouts but does not shift the CBS (§5.1). Finally, as has been shown for pretraining, the CBS increases as training progresses (§5.2).

2 BACKGROUND

We begin our study of critical batch size for GRPO by providing background on GRPO (§2.1) and the McCandlish noise-scale model of critical batch size (§2.2)

2.1 GROUP RELATIVE POLICY OPTIMIZATION

GRPO (Shao et al., 2024) alternates between sampling and policy updates. Each iteration draws B prompts $\{q_i\}_{i=1}^{B}$ from the training set, and samples K responses $\{o_{i,k}\}_{k=1}^{K}$ per prompt from the current policy. Typically in GRPO, there is a correct final answer, such as solving a math problem correctly. Each response is scored by a verifier reward $r_{i,k} \in \{0, 1\}$, where 0 signifies incorrect and 1 signifies correct. One of the key innovation of GRPO is to use a *group-relative* advantage,

$$A_{i,k} = \frac{r_{i,k} - \mu(q_i)}{\sigma(q_i)}, \tag{1}$$

with $\mu(q_i)$ is the average reward of responses to prompt q_i, and $\sigma(q_i)$ is the standard deviation of rewards. The training objective for policy π_t given rollouts from policy π_{old} is as follows:

$$J_m(\pi_t) = \frac{1}{b} \sum_{(i,k)\in\mathcal{B}_m} \frac{1}{|o_{i,k}|} \sum_{\tau=1}^{|o_{i,k}|} \min\left\{ \rho_{i,k,\tau} A_{i,k}, \text{clip}(\rho_{i,k,\tau}, 1-\varepsilon, 1+\varepsilon)\, A_{i,k} \right\}. \tag{2}$$

where $\mathcal{B}_m$ is the set of all prompt and response pairs, $|o_{i,k}|$ denotes the length of response $o_{i,k}$, and $\rho_{i,k,\tau} = \frac{\pi_t(o_{i,k,\tau}|x_i, o_{i,k,<\tau})}{\pi_{\text{old}}(o_{i,k,\tau}|x_i, o_{i,k,<\tau})}$. In the on-policy setting on which we focus, each set of BK prompt and response sequences is used for policy update in one pass, and $\pi_t = \pi_{\text{old}}$, hence $\rho_{i,k,\tau} = 1$.

2.2 CRITICAL BATCH SIZE

The critical batch size quantifies the extent to which batch size can be increased before losing sample efficiency, an important consideration for data-parallelism (McCandlish et al., 2018; Shallue et al., 2019; Jain et al., 2018; Zhang et al., 2025). We build directly on the noise-scale model of McCandlish et al. (2018). For a loss $L(\theta) = \mathbb{E}_x[L_x(\theta)]$ with gradient $G = \nabla L$ and Hessian $H = \nabla^2 L$, the mini-batch estimator $\text{G}_{\text{est}} = \frac{1}{B}\sum_i \nabla L_{x_i}$ is unbiased with covariance $\text{Cov}(\text{G}_{\text{est}}) = \Sigma/B$. Expanding $L(\theta - \eta\ \text{G}_{\text{est}})$ to second order, the optimal step size and the resulting per-step loss decrement both scale as $1/(1 + \mathcal{B}_{\text{noise}}/B)$, where

$$\mathcal{B}_{\text{noise}} = \frac{\text{tr}(H\Sigma)}{G^\top H G}, \qquad \text{and} \qquad S(B) = S_\infty\left(1 + \frac{\mathcal{B}_{\text{crit}}}{B}\right), \qquad \mathcal{B}_{\text{crit}} \equiv \mathcal{B}_{\text{noise}}, \tag{3}$$

with S_∞ the infinite-batch step count. For $B \ll \mathcal{B}_{\text{crit}}$, doubling B roughly halves the steps to a target; for $B \gg \mathcal{B}_{\text{crit}}$, extra samples are largely wasted. The steps-to-target form on the right is the one-parameter fit we use throughout experiments. We adopt the McCandlish formulation because it expresses $\mathcal{B}_{\text{crit}}$ through gradient-noise quantities that we can decompose along the prompt and rollout axes of GRPO.

The RLVR objective shifts during training, since the advantage in (1) depends on the current policy. We therefore follow standard practice for RL with language models and fit $\mathcal{B}_{\text{crit}}$ against a stable downstream metric (`mean@1` on the evaluation set) used as a proxy for loss, recording the outer steps $S(B)$ needed to first reach a target accuracy.

3 A NOISE-SCALE MODEL OF CBS FOR GRPO

We theoretically analyze the critical batch size of a single on-policy update around the current policy $\pi_{\theta_{\text{old}}}$. Let $q \sim \mathcal{Q}$ denote a prompt and $o \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ a rollout. We begin by assuming that advantages are exact: each rollout has an advantage $A(q, o)$ that depends only on the prompt and rollout, and relax this assumption later in the section. Conditioned on q, rollouts are sampled independently and their gradient contributions are independent and identically distributed. For a rollout $o = (t_1, \ldots, t_{L(o)})$, define its length-normalized score

$$s(q, o) = \frac{1}{L(o)} \sum_{l=1}^{L(o)} \nabla_\theta \log \pi_\theta(t_l \mid q, t_{<l}) \Big|_{\theta=\theta_{\text{old}}},$$

and its single-rollout gradient contribution

$$g(q,o) = A(q,o)\, s(q,o).$$

The population gradient is

$$G = \mathop{\mathbb{E}}_{q,o}[g(q,o)].$$

We decompose the covariance of a single-rollout gradient into variation across prompts and variation across rollouts of the same prompt. Define

$$\Sigma_q = \mathrm{Cov}_q\left(\mathop{\mathbb{E}}_{o|q}[g(q,o)]\right), \qquad \Sigma_{o|q} = \mathop{\mathbb{E}}_{q}\left[\mathrm{Cov}_{o|q}(g(q,o))\right].$$

Thus, Σ_q measures heterogeneity in the expected gradient across prompts, while $\Sigma_{o|q}$ measures stochasticity among rollouts of a fixed prompt. As is standard for the analysis of CBS, we approximate the population surrogate objective to second order around θ_{old}. Let

$$C = -\nabla^2_\theta J(\theta)\Big|_{\theta=\theta_{\mathrm{old}}}$$

denote the local positive curvature matrix, and assume $G^\top CG > 0$. The resulting critical batch size should therefore be interpreted as a *local quantity governing a single update under this quadratic approximation,* rather than as a global property of the optimization problem.

For a batch consisting of B independently sampled prompts and K independent rollouts per prompt, define the curvature-weighted prompt-level and rollout-level noise scales as:

$$\sigma^2_{\mathrm{prompt}} = \frac{\mathrm{tr}(C\Sigma_q)}{G^\top CG}, \qquad \sigma^2_{\mathrm{rollout}} = \frac{\mathrm{tr}(C\Sigma_{o|q})}{G^\top CG}.$$

We define $\epsilon_{\mathrm{opt}}(B,K)$ as the optimal step size to attain the maximum expected improvement of the local quadratic approximation to the surrogate objective for B prompts and K. $\epsilon_{\max}$, is the optimal step size when using the exact population gradient, providing a reference for how gradient noise limits the step size.

Theorem 1 (CBS) *Under the assumptions above, the step size that maximizes the expected improvement under the local quadratic approximation to the population surrogate objective is*

$$\epsilon_{\mathrm{opt}}(B,K) = \frac{\epsilon_{\max}}{1 + \dfrac{\sigma^2_{\mathrm{prompt}}}{B} + \dfrac{\sigma^2_{\mathrm{rollout}}}{BK}},$$

where $\epsilon_{\max} = \|G\|^2/(G^\top CG)$ is the optimal step size for the noiseless population gradient. For fixed K, increasing the number of prompts drives $\epsilon_{\mathrm{opt}}(B,K)$ toward $\epsilon_{\max}$. For fixed B, increasing the number of rollouts per prompt instead drives it toward $\epsilon'_{\max}(B) := \lim_{K\to\infty} \epsilon_{\mathrm{opt}}(B,K)$. We define the critical prompt and rollout sizes respectively as:

$$\mathcal{B}_{crit}(K) = \sigma^2_{\mathrm{prompt}} + \frac{\sigma^2_{\mathrm{rollout}}}{K}, \qquad \mathcal{K}_{\mathrm{crit}}(B) = \frac{\sigma^2_{\mathrm{rollout}}}{B + \sigma^2_{\mathrm{prompt}}}.$$

We present the proof in Appendix A.1. The maximal expected improvement predicted by the local quadratic model is $\frac{1}{2}\epsilon_{\mathrm{opt}}(B,K)\|G\|^2$. The two axes have different limiting improvements: increasing the number of prompts removes both sources of noise, whereas increasing rollouts at fixed B leaves prompt-level noise.

Theorem 1 concerns a single on-policy update, rather than the number of steps required to reach a target reward or accuracy. Connecting these quantities requires additional assumptions: the quadratic model must remain informative along training, surrogate improvement must track progress toward the target, and different batch configurations must follow comparable training trajectories. Under these assumptions and with optimally chosen step sizes, one could estimate steps to a target by accumulating the predicted local progress over training, following McCandlish et al. (2018).

Corollary 1 (Scaling comparison under a fixed budget) *Fix a prompt count B and a rollout count per prompt K. Under the assumptions of Theorem 1, doubling B yields an optimal step size at least as large as doubling K:*

$$\epsilon_{\mathrm{opt}}(2B,K) \geq \epsilon_{\mathrm{opt}}(B,2K). \tag{4}$$

Proof. By substituting from Theorem 1, we obtain:

$$\epsilon_{\text{opt}}(2B, K) = \frac{\epsilon_{\max}}{1 + \frac{\sigma^2_{\text{prompt}}}{2B} + \frac{\sigma^2_{\text{rollout}}}{2BK}}, \qquad \epsilon_{\text{opt}}(B, 2K) = \frac{\epsilon_{\max}}{1 + \frac{\sigma^2_{\text{prompt}}}{B} + \frac{\sigma^2_{\text{rollout}}}{2BK}}. \tag{5}$$

The denominator is no larger in the expression for $\epsilon_{\text{opt}}(2B, K)$ than it is in the expression for $\epsilon_{\text{opt}}(B, 2K)$, and thus $\epsilon_{\text{opt}}(2B, K) \geq \epsilon_{\text{opt}}(B, 2K)$. This concludes the proof.

Both allocations in Corollary 1 generate $2BK$ sequences. Since the optimal expected improvement under the local quadratic surrogate model is proportional to ϵ_{opt}, doubling the number of prompts yields at least as much predicted improvement as doubling the number of rollouts per prompt, when each allocation uses its optimal step size. The improvement is strictly greater when $\sigma^2_{\text{prompt}} > 0$, because increasing the number of prompts reduces both prompt-level and rollout-level noise.

Corollary 2 (Advantage estimation.) *Under the assumptions of Theorem 1, let $A(q, o) = r(q, o) - \mathbb{E}_{o|q}[r(q, o)]$ and assume all rollouts have the same fixed length. Suppose we use leave-one-out advantage estimates (Ahmadian et al., 2024): $\hat{A}_k = r(q, o_k) - \frac{1}{K-1}\sum_{j \neq k} r(q, o_j)$. Then the critical prompt size under the local quadratic model is*

$$B^{\text{RLOO}}_{\text{crit}}(K) = \sigma^2_{\text{prompt}} + \frac{\sigma^2_{\text{rollout}}}{K} + \mathcal{O}\left(\frac{1}{K^2}\right).$$

Under these assumptions, baseline-estimation noise contributes a correction that decays quadratically with rollout count, leaving the leading dependence of the CBS unchanged as K grows.

Takeaways: The gradient noise of GRPO decomposes into a prompt-level and a rollout-level scale: σ^2_{prompt} and $\sigma^2_{\text{rollout}}/K$. Adding prompts under fixed number of rollouts averages down both noise terms, but adding rollouts averages down only the rollout-level term. Hence, allocating additional samples to prompts rather than rollouts admits the larger step.

4 Measuring the CBS for GRPO

In this section, we empirically examine the CBS for LLM policy optimization. We examine (i) the effects and trade-offs of scaling B and K, under fixed B, K, or $N = B \times K$ (§4.2), (ii) step size scaling rules with increasing batch size (§4.3), and (iii) the role of KL regularization (§4.4).

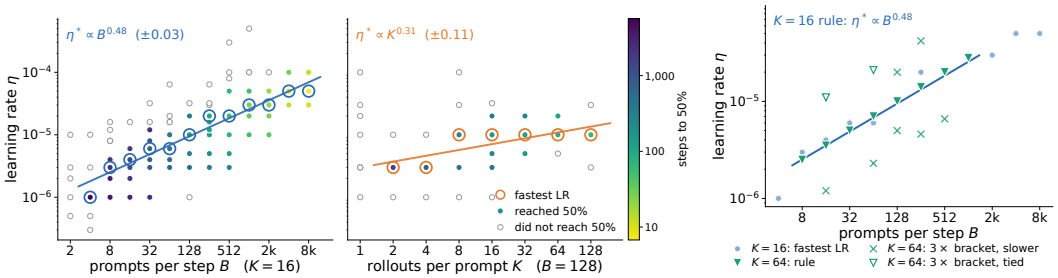
4.1 Experimental Setup

We train Qwen2.5-Math-Instruct-1.5B (Yang et al., 2024) on the DAPO training dataset (Yu et al., 2025) and evaluate the `mean@1` performance on the AIME-1983–2024 dataset (Veeraboina, 2023) as our target metric. For all our experiments, we use on-policy GRPO (Shao et al., 2024) as our policy optimization algorithm, and a maximum response length of 4096 tokens. Throughout, we measure CBS by the steps-to-target protocol of §2.2: for each run we record the number of steps S needed to first reach `mean@1` $\geq T$, where T is a target accuracy. We fit the hyperbola $S(x) = S_{\min}(1 + \frac{\mathcal{B}_{\text{crit}}}{B})$ along the swept axis to estimate the critical value $\mathcal{B}_{\text{crit}}$[1]. Training details, hyperparameter grids, and the measurement protocol are provided in Appendix B; the validation trajectories behind every point are shown in Appendix C.

4.2 Batch size scaling for GRPO

To study batch size scaling for GRPO and to delineate the effects of scaling the prompts B versus rollouts per prompt K, we empirically evaluate the CBS in two main settings: (1) scaling $B \in [2, 8192]$ under two distinct fixed rollouts per prompt settings: $K \in \{16, 64\}$, and (2) scaling $K \in [1, 128]$ under fixed $B = 128$.

[1]We use $T = 0.5$ as the target accuracy; it is an accuracy that most batch sizes are able to attain, while being sufficiently far from the base accuracy of approximately 0.25.



(a) Scaling η with B ($K = 16$) and K ($B = 128$).

(b) $\eta^* \propto \sqrt{B}$ transfers to $K = 64$

Figure 2: **Scaling step size (η^*) with batch size.** (a) Each point is a run, colored by steps to reach 50% AIME accuracy; rings (○) mark the η^* at each batch size. We find that η^* follows the square-root rule of supervised learning along the prompt axis ($\eta^* \propto \sqrt{B}$) and remains nearly constant along the rollout axis (beyond a certain small K). (b) We use the rule from $K = 16$ to set η for every $K = 64$ run (▾). Each of these runs reach the target at every batch size. Validation runs at neighboring η values remain slower (×) or tied (▽).

Scaling prompts is more effective than scaling rollouts. Steps-to-target plateau sooner when increasing rollouts at fixed B than when increasing prompts at fixed K, with little benefit beyond $K = 16$ (Figure 1). At the same sequence budget, allocating more samples to distinct prompts is more effective: for $N \approx 16$k, $(B, K) = (1024, 16)$ reaches the target in 25 steps, compared with 62 for $(128, 128)$. This finding is qualitatively consistent with Theorem 1 and Corollary 1.

The sequence-level CBS does not depend on K. To test whether the observed CBS is specific to $K = 16$, we repeat the prompt sweep at $K = 64$ (Figure 1). The two curves have the same shape: near-perfect scaling up to about 16k sequences per step, with the knee at the same place. However, at matched sequence counts, $K = 64$ requires approximately twice as many steps to reach the target as $K = 16$. Thus, the fitted CBS is similar across these two rollout counts, even though allocating the sequence budget to more distinct prompts is substantially more efficient.

GRPO affords a large CBS in prompts. We observe that policy optimization through GRPO can afford a surprisingly large CBS, $\mathcal{B}_{\text{crit}}(K)$, for fixed K. As observed in Figure 1, we see perfect scaling of steps-to-target with increasing batch sizes up to $\mathcal{N}_{\text{crit}} = 16$k sequences, which corresponds to $\sim 65M$ tokens per step at our 4096 token response budget. We note that this is consistent with McCandlish et al. (2018), which expects RL settings with sparse rewards to have high CBS.

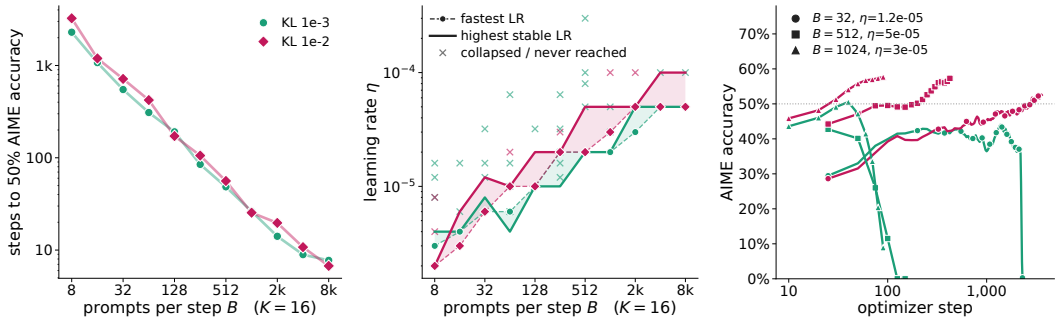
4.3 Learning rate scaling along each axis

For every (B, K) setting in §4.2, we sweep the learning rate and record the one that reaches the target in the fewest steps. We report the sweeps and the resulting scaling in Figure 2.

Prompts and rollouts follow different learning rate regimes. When scaling prompts, the optimal learning rate scales proportionally to the square root of the batch size, all the way up to $\mathcal{N}_{\text{crit}}$ (Figure 2a, left). This resembles the square-root scaling rule developed for Adam (Kingma & Ba, 2015; Malladi et al., 2022). When scaling rollouts at fixed B, the best learning rate remains approximately constant beyond $K = 4$ (Figure 2a, right). This contrast is qualitatively consistent with the noise decomposition in Theorem 1: adding prompts reduces both noise contributions, whereas adding rollouts leaves prompt-level noise.

The square-root rule transfers across K. We use the rule fitted at $K = 16$ to set the learning rate for the $K = 64$ runs. As reported in Figure 2b, we find that the prescribed learning rate reaches the target at every batch size. We further validate the rule $\eta^* \propto \sqrt{B}$ by evaluating learning rates in the neighborhood of the predicted optimal learning rate and find that they are suboptimal.

At fixed N, larger B admits a larger η. A consequence of this η scaling rule is that at a matched number of sequences N, the allocation with more prompts also supports the larger η, as Corollary 1 predicts. At $N = 16$k, the optimal η for $(B, K) = (1024, 16)$ is $3\times$ that for $(128, 128)$.



(a) Steps to target at the fastest η. (b) Fastest vs. highest stable η. (c) Effect of KL on stability.

Figure 3: **KL regularization does not change the CBS, but a stronger penalty makes larger learning rates stable.** All runs use $K = 16$. (a) Steps to 50% AIME accuracy at the fastest η coincide for KL coefficients $1e^{-3}$ and $1e^{-2}$ at every batch size. (b) The fastest η (---) is nearly the same under both coefficients, but the largest stable one (—) is about $2\times$ higher under $1e^{-2}$ for $B \geq 64$; $\times$ are runs that collapsed or never reached the target. (c) At the same B and η, the weakly regularized run collapses while the strongly regularized one reaches and holds the target.

4.4 KL REGULARIZATION CHANGES STABILITY, NOT CBS

GRPO is oftentimes trained with KL regularization towards the reference policy. Since regularization changes the objective, it could in principle change the gradient noise, and hence the CBS, or it could merely change which learning rates are usable. We further perform the prompt sweep of §4.2 ($K = 16$) with KL coefficients $1e^{-3}$ and $1e^{-2}$. We report the results in Figure 3.

Stronger KL regularization widens the range of stable learning rates. The two coefficients differ in training stability and the range of learning rates. With a KL regularization coefficient of $1e^{-3}$, the fastest learning rate is typically also the largest one that reaches the target without collapsing; with KL coefficient $1e^{-2}$, the largest stable learning rate is about $2\times$ higher for $B \geq 64$ (Figure 3b). Figure 3c shows matched pairs of runs at the same B and η: the KL $1e^{-3}$ run collapses, while the KL $1e^{-2}$ regularized run reaches and holds the target. In practice, a stronger KL penalty could therefore be a way to make larger step sizes more stable, without any loss in steps-to-target.

KL regularization does not appear to affect CBS. At the fastest learning rate, the steps-to-target are similar for the two coefficients at every batch size (Figure 3a). Further, the fastest learning rate also remains consistent in the two settings, within the resolution of our sweep (dashed lines in Figure 3b). The batch size scaling behavior appears in this setting to be governed by the noise of the policy gradient and is insensitive to the strength of KL regularization in this range.

Takeaways:
- The CBS of on-policy GRPO is large along the prompt axis, with near-perfect speedups up to $\sim$ 16k sequences per step, well beyond batch sizes in common use.
- The optimal learning rate grows proportionally to $\sqrt{B}$ in prompts and is flat in rollouts.
- A stronger KL penalty makes larger learning rates stable without moving the CBS.

5 FURTHER CHARACTERIZATION OF CBS DYNAMICS

5.1 EFFECT OF ADVANTAGE NOISE ON CBS

In GRPO, the number of rollouts per prompt does not only affect the training batch size, but also the advantage estimation: the advantage is computed as a function of the rewards of rollouts for each prompt. Too few rollouts may lead to an unreliable advantage estimate; more rollouts will increase the quality of the estimate. Thus, we additionally consider whether the batch size of rollouts used for *advantage estimation* is an important axis independent of the number of rollouts used for training.

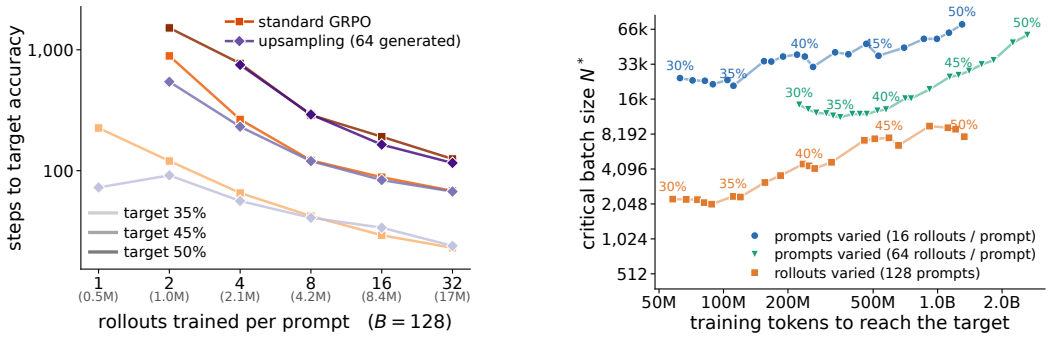


Figure 4: **(a) Better advantage estimates speed up early training.** Steps to target against rollouts trained per prompt at $B = 128$, across three accuracies. GRPO trained with upsampled advantages train faster for small K but the curves coincide thereafter: advantage noise does not set the CBS in rollouts. **(b) CBS grows over training.** Fitted $\mathcal{N}_{\text{crit}}$ at various stages of GRPO training for the prompt sweeps at $K \in \{16, 64\}$ and the rollout sweep at $B = 128$. Along every axis, $\mathcal{N}_{\text{crit}}$ grows severalfold over training, and both prompt sweeps stay well above the rollout sweep throughout.

Standard GRPO generates K rollouts, computes group-relative advantages with those K rollouts, and uses the same K rollouts for the gradient update. Here, we consider an *upsampling* setting in which we use K_{train} rollouts for the optimizer step but $K_{\text{adv}} > K_{\text{train}}$ rollouts for advantage estimation. In our experiments, we fix $K_{\text{adv}} = 64$ rollouts for advantage estimation and sweep K_{train}, and compare this to the standard GRPO setting with $K_{\text{adv}} = K_{\text{train}}$. Both sequences use the same batch size and differ only in the *quality* of the advantage estimate. This allows us to study whether a higher quality advantage estimate, decoupled from the number of rollouts used for training, improves performance. We report results in Figure 4a; all details are given in Appendix B.4.

Better advantage estimates help early in training. At $K_{\text{train}} = 1$, where a single rollout provides no within-group baseline, upsampling reaches the low targets in up to $3\times$ fewer steps than standard GRPO. At $K_{\text{train}} \in \{2, 4\}$ the gain is at most $1.6\times$. The gains further diminish quickly with larger K: for $K_{\text{train}} \geq 8$, the two settings take the same number of steps at every target we consider.

Advantage noise minimally affects CBS. While upsampling reduces the noise in the advantage estimate, we observe that it leaves the steps-to-target curve, and hence $\mathcal{K}_{\text{crit}}$, nearly unchanged. This suggests that the diminishing returns from scaling rollouts in §4.2 stem from the gradient samples, not from noisy advantages. This is consistent with Corollary 2, where we hypothesize that the estimation noise enters $\mathcal{B}_{\text{crit}}$ only at order $1/K^2$, and hence should be negligible for larger groups.

5.2 CBS GROWS OVER TRAINING

As the policy improves over training, the noise scales, σ^2_{prompt} and $\sigma^2_{\text{rollout}}$, in Theorem 1 change. Hence, the CBS need not be constant over the course of training. Here, we trace the CBS through training by refitting $\mathcal{N}_{\text{crit}}$ along both axes of §4.2. We report the results in Figure 4b.

The CBS grows as training progresses. Along both axes, $\mathcal{N}_{\text{crit}}$ increases roughly threefold from the earliest to the latest stage we measure, over a $20\times$ increase in training tokens. Later stages of training thus tolerate larger batches than earlier stages. This mirrors the growth of the CBS with training tokens in pretraining (Zhang et al., 2025; Merrill et al., 2025).

The prompt–rollout gap persists throughout training. At every stage, the CBS along the prompt axis stays about an order of magnitude above that along the rollout axis. The asymmetry of §4.2 is therefore not specific to a training stage: prompt-level noise remains a large share of the gradient noise throughout training, and prompts remain the effective dimension for parallelism.

Takeaways:

- Better advantage estimates speed up early training for small K but do not alter the CBS. Thus, the diminishing returns in rollouts come from gradient noise, not from noisy advantages.
- CBS grows severalfold over training and the prompt–rollout gap persists throughout.

6 Related Work

RL for LLMs. While policy gradient methods (Sutton et al., 1999; Kakade, 2001; Schulman et al., 2015) have been used in many settings, the works of Ouyang et al. (2022); Dai et al. (2024) have garnered significant attention for their application to LLM post-training. Various methods such as DPO (Rafailov et al., 2023), PPO (Schulman et al., 2017), and GRPO (Shao et al., 2024) have been shown to be effective in improving accuracy on reasoning benchmarks. Recent work further extends this for compute efficient post-training (Yu et al., 2025; Chen et al., 2025; Mohri et al., 2026).

Critical batch size. Several works have characterized CBS and related limits of mini-batch parallelism (McCandlish et al., 2018; Shallue et al., 2019; Jain et al., 2018). Shallue et al. (2019) empirically characterize diminishing returns from large mini-batches across neural-network workloads, while Jain et al. (2018) theoretically analyze analogous limits in least-squares regression. McCandlish et al. (2018) develop the noise-scale theory on which we base our analysis and include empirical findings for RL tasks (Atari and Dota), but do not consider RL for LLMs. Zhang et al. (2025) and Merrill et al. (2025) study the CBS of pretraining empirically and find that it grows with the amount of training data, a trend we also observe over the course of policy optimization (§5.2).

Batch size and compute allocation in LLM RL. ScaleRL (Khatri et al., 2026) provides empirical scaling laws for validation performance under increased rollout and training compute. Gao et al. (2026) study batch size, including the tradeoff between prompts and rollouts, jointly with prompt difficulty, and propose a prompt curriculum to improve RL training efficiency. Li et al. (2026) explore when larger batches improve sample and wall-clock efficiency in RL settings, and Cheng et al. (2026) study compute-optimal allocation across prompts, rollouts, and sequential optimization steps. Our work instead focuses on the critical batch size itself: we decompose it into prompt- and rollout-level gradient noise and measure where step efficiency saturates along each axis.

7 Discussion

We formulate CBS for GRPO-style algorithms by decomposing gradient noise into prompt-level and rollout-level terms, and show that increasing the number of prompts can decrease both noise terms while increasing rollouts cannot decrease prompt-level noise. Through empirical analysis with Qwen2.5-Math-Instruct-1.5B (Yang et al., 2024) trained on DAPO, we characterize how scaling prompts and rollouts respectively affects the CBS: scaling prompts affords a CBS an order of magnitude higher than scaling rollouts. We additionally measure how optimal learning rate scales and observe that the CBS is consistent across both KL and quality of advantage estimates.

Practical implications. While our experiments are limited to one setting, our results suggest several practical takeaways. First, we find that the CBS is large along the prompt-axis, affording a high degree of data parallelism without losing sample efficiency. To our knowledge, the CBS is considerably larger than batch sizes used to train current LLMs for similar settings (Yu et al., 2025; Mistral AI, 2025; Wang et al., 2025). Our findings thus indicate that RLVR training through GRPO-style algorithms might afford more parallelism than is leveraged by current practice. Additionally, the results on learning rate scaling and CBS consistence with KL regularization offer practical guidance for how to set training hyperparameters.

Limitations and open questions. Our theoretical analysis assumes a curvature-limited SGD step (due to ease of analysis), while our experiments use standard adaptive optimizers like AdamW. Our analysis characterizes the optimal step size for a single update under a local quadratic surrogate model. Translating this result into the number of training steps required to reach a target reward remains challenging. Establishing this connection requires additional assumptions or further analysis. Hence, reconciling the noise-scale analysis with adaptive optimizers is an open direction for future work. Further, due to compute constraints, most of empirical analysis takes place over a single model and task. Understanding how the critical batch size scales with model and dataset size, like has been done for pre-training, is an exciting next step. Since CBS grows over training, studying batch-size warmup and optimization dynamics for GRPO could reveal new training regimes.

8 AI USE STATEMENT

In this work, we used generative AI tools for assisting writing and finding related work. We have also used generative AI tools to help with code auto-completion, but not with writing main logic. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean or reformat datasets, support qualitative and thematic data analysis, or interpret results. The remaining required disclosure tasks are not applicable to this work.

We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

9 REPRODUCIBILITY STATEMENT

We have made efforts toward experiment reproducibility by carefully reporting our training hardware, codebase, and hyperparameters in Appendix B.

REFERENCES

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 *Annual Meeting of the Association for Computational Linguistics*, pp. 12248–12267, 2024. URL https://aclanthology.org/2024.acl-long.662/.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. MiniMax-M1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025. URL https://arxiv.org/abs/2506.13585.

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. *arXiv preprint arXiv:2603.12151*, 2026. URL https://arxiv.org/abs/2603.12151.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TyFrPOKYXw.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zqOCacBD3P.

Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023. URL https://arxiv.org/abs/2301.13442.

Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18(223):1–42, 2018. URL https://jmlr.org/papers/v18/16-595.html.

Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, pp. 1531–1538, 2001. URL https://dl.acm.org/doi/10.5555/2980539.2980738.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S. Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for LLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FMjeC9Msws.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

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.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and scaling rules for adaptive gradient algorithms. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/32ac710102f0620d0f28d5d05a44fe08-Abstract-Conference.html.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018. URL https://arxiv.org/abs/1812.06162.

Will Merrill, Shane Arora, Dirk Groeneveld, and Hannaneh Hajishirzi. Critical batch size revisited: A simple empirical approach to large-batch language model training. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=XUKUx7Xu89.

Mistral AI. Magistral. *arXiv preprint arXiv:2506.10910*, 2025. URL https://arxiv.org/abs/2506.10910.

Clara Mohri, Amir Globerson, Haim Kaplan, Tomer Koren, and Yishay Mansour. Cost-aware learning. *arXiv preprint arXiv:2604.28020*, 2026. URL https://arxiv.org/abs/2604.28020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019. URL https://jmlr.org/papers/v20/18-789.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025. doi: 10.1145/3689031.3696075. URL https://dl.acm.org/doi/10.1145/3689031.3696075.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 1999. URL https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Hemish Veeraboina. AIME problem set 1983–2024. Kaggle dataset, 2023. URL https://www.kaggle.com/datasets/hemishveeraboina/aime-problem-set-1983-2024.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-Cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025. URL https://arxiv.org/abs/2512.13607.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=olimv9xl9K.

APPENDIX CONTENTS

A Proofs **15**

A.1 Proof of Theorem 1 15

A.2 Proof of Corollary 2 17

B Experiment Details **19**

B.1 Model, data, and evaluation 19

B.2 Training configuration 19

B.3 Hyperparameter sweeps 20

B.4 Upsampling implementation 21

B.5 Measuring the critical batch size 21

C Training trajectories **21**

A PROOFS

A.1 PROOF OF THEOREM 1

Theorem 1 (CBS) *Under the assumptions above, the step size that maximizes the expected improvement under the local quadratic approximation to the population surrogate objective is*

$$\epsilon_{\mathrm{opt}}(B,K) = \frac{\epsilon_{\max}}{1 + \dfrac{\sigma^2_{\mathrm{prompt}}}{B} + \dfrac{\sigma^2_{\mathrm{rollout}}}{BK}},$$

where $\epsilon_{\max} = \|G\|^2/(G^\top C G)$ is the optimal step size for the noiseless population gradient. For fixed K, increasing the number of prompts drives $\epsilon_{\mathrm{opt}}(B,K)$ toward $\epsilon_{\max}$. For fixed B, increasing the number of rollouts per prompt instead drives it toward $\epsilon'_{\max}(B) := \lim_{K\to\infty} \epsilon_{\mathrm{opt}}(B,K)$. We define the critical prompt and rollout sizes respectively as:

$$\mathcal{B}_{crit}(K) = \sigma^2_{\mathrm{prompt}} + \frac{\sigma^2_{\mathrm{rollout}}}{K}, \qquad \mathcal{K}_{\mathrm{crit}}(B) = \frac{\sigma^2_{\mathrm{rollout}}}{B + \sigma^2_{\mathrm{prompt}}}.$$

Proof. Let

$$\hat{g}(q) = \frac{1}{K}\sum_{k=1}^{K} g(q, o_k)$$

denote the gradient contribution of a single prompt, where $o_1, \ldots, o_K \sim_{\text{i.i.d.}} \pi_{\theta_{\mathrm{old}}}(\cdot \mid q)$ and

$$g(q,o) = A(q,o)\, s(q,o)$$

is the single-rollout gradient contribution defined above. The stochastic gradient over B prompts is therefore

$$\hat{G} = \frac{1}{B}\sum_{i=1}^{B} \hat{g}(q_i).$$

By definition,

$$G = \mathop{\mathbb{E}}_{q,o}[g(q,o)] = \mathbb{E}[\hat{G}].$$

Gradient covariance. Define the conditional mean gradient

$$\mu(q) = \mathop{\mathbb{E}}_{o|q}[g(q,o)].$$

Because the advantages are exact, $g(q,o_k)$ depends only on q and the corresponding rollout o_k. Hence, conditional on q, the K rollout gradient contributions are i.i.d. Thus,

$$\mathbb{E}[\hat{g}(q) \mid q] = \mu(q),$$

and

$$\mathrm{Cov}(\hat{g}(q) \mid q) = \frac{1}{K}\,\mathrm{Cov}_{o|q}(g(q,o)).$$

Applying the law of total covariance gives

$$\begin{aligned}\mathrm{Cov}(\hat{g}(q)) &= \mathrm{Cov}_q(\mathbb{E}[\hat{g}(q) \mid q]) + \mathop{\mathbb{E}}_q[\mathrm{Cov}(\hat{g}(q) \mid q)] \\ &= \Sigma_{\mathrm{prompt}} + \frac{1}{K}\Sigma_{\mathrm{rollout}},\end{aligned}$$

where

$$\Sigma_{\mathrm{prompt}} = \mathrm{Cov}_q\left(\mathop{\mathbb{E}}_{o|q}[g(q,o)]\right), \qquad \Sigma_{\mathrm{rollout}} = \mathop{\mathbb{E}}_q\left[\mathrm{Cov}_{o|q}(g(q,o))\right].$$

Since the B prompts are sampled independently,

$$\mathrm{Cov}(\hat{G}) = \frac{1}{B}\,\mathrm{Cov}(\hat{g}(q)) = \frac{\Sigma_{\mathrm{prompt}}}{B} + \frac{\Sigma_{\mathrm{rollout}}}{BK}.$$

Local quadratic approximation. Let $\bar{J}(\theta)$ denote the population surrogate objective, and define

$$H = -\nabla_\theta^2 \bar{J}(\theta)\big|_{\theta=\theta_{\text{old}}},$$

with $H \succeq 0$ locally For the stochastic ascent update

$$\theta^+ = \theta_{\text{old}} + \epsilon \hat{G},$$

the second-order approximation around θ_{old} gives

$$\bar{J}(\theta^+) \approx \bar{J}(\theta_{\text{old}}) + \epsilon G^\top \hat{G} - \frac{\epsilon^2}{2} \hat{G}^\top H \hat{G}.$$

Taking expectations and using $\mathbb{E}[\hat{G}] = G$,

$$\mathbb{E}[G^\top \hat{G}] = \|G\|^2.$$

Moreover,

$$\begin{aligned}\mathbb{E}[\hat{G}^\top H \hat{G}] &= \text{tr}\Big(H\ \mathbb{E}[\hat{G}\hat{G}^\top]\Big) \\ &= G^\top H G + \text{tr}\Big(H\ \text{Cov}(\hat{G})\Big).\end{aligned}$$

Substituting the covariance decomposition above therefore yields

$$\mathbb{E}[\bar{J}(\theta^+)] \approx \bar{J}(\theta_{\text{old}}) + \epsilon \|G\|^2 - \frac{\epsilon^2}{2}\left(G^\top H G + \frac{\text{tr}(H\Sigma_{\text{prompt}})}{B} + \frac{\text{tr}(H\Sigma_{\text{rollout}})}{BK}\right).$$

Optimal step size and critical batch size. Maximizing the quadratic approximation with respect to ϵ gives

$$\epsilon_{\text{opt}}(B,K) = \frac{\|G\|^2}{G^\top H G + \frac{\text{tr}(H\Sigma_{\text{prompt}})}{B} + \frac{\text{tr}(H\Sigma_{\text{rollout}})}{BK}}.$$

Let

$$\epsilon_{\max} = \frac{\|G\|^2}{G^\top H G},$$

and define

$$\sigma^2_{\text{prompt}} = \frac{\text{tr}(H\Sigma_{\text{prompt}})}{G^\top H G}, \qquad \sigma^2_{\text{rollout}} = \frac{\text{tr}(H\Sigma_{\text{rollout}})}{G^\top H G}.$$

Then

$$\epsilon_{\text{opt}}(B,K) = \frac{\epsilon_{\max}}{1 + \frac{1}{B}\left(\sigma^2_{\text{prompt}} + \frac{\sigma^2_{\text{rollout}}}{K}\right)}.$$

Defining the local critical batch size in prompts through

$$\epsilon_{\text{opt}}(B,K) = \frac{\epsilon_{\max}}{1 + \mathcal{B}_{\text{crit}}(K)/B},$$

we obtain:

$$\mathcal{B}_{\text{crit}}(K) = \sigma^2_{\text{prompt}} + \frac{\sigma^2_{\text{rollout}}}{K}.$$

Critical number of rollouts. For fixed B, separate the K-independent part of the denominator and factor it out:

$$1 + \frac{\sigma^2_{\text{prompt}}}{B} + \frac{\sigma^2_{\text{rollout}}}{BK} = \Big(1 + \frac{\sigma^2_{\text{prompt}}}{B}\Big)\Big(1 + \frac{K_{\text{crit}}(B)}{K}\Big),$$

with

$$K_{\text{crit}}(B) = \frac{\sigma^2_{\text{rollout}}/B}{1 + \sigma^2_{\text{prompt}}/B} = \frac{\sigma^2_{\text{rollout}}}{B + \sigma^2_{\text{prompt}}},$$

which gives the second form of ϵ_{opt} in Theorem 1. Along the rollout axis, $K_{\text{crit}}(B)$ therefore plays the role of $\mathcal{B}_{\text{crit}}$ in (3), with the noiseless step size replaced by $\epsilon_{\max}/(1 + \sigma^2_{\text{prompt}}/B)$.

A.2 Proof of Corollary 2

Corollary 2 (Advantage estimation.) *Under the assumptions of Theorem 1, let* $A(q,o) = r(q,o) - \mathbb{E}_{o|q}[r(q,o)]$ *and assume all rollouts have the same fixed length. Suppose we use leave-one-out advantage estimates (Ahmadian et al., 2024):* $\hat{A}_k = r(q,o_k) - \frac{1}{K-1}\sum_{j\neq k} r(q,o_j)$. *Then the critical prompt size under the local quadratic model is*

$$B_{\text{crit}}^{\text{RLOO}}(K) = \sigma^2_{\text{prompt}} + \frac{\sigma^2_{\text{rollout}}}{K} + \mathcal{O}\left(\frac{1}{K^2}\right).$$

Proof. Fix a prompt q and sample $K \geq 2$ conditionally independent rollouts. Let $A_k = r_k - \mathbb{E}[r_k \mid q]$ be the exact advantage and write the RLOO advantage as

$$\hat{A}_k = A_k + \xi_k, \qquad \xi_k = -\frac{1}{K-1}\sum_{j\neq k} A_j.$$

Assume that rollout length is fixed conditional on q, so that $\mathbb{E}[s_k \mid q] = 0$ under the usual score identity. Assume that all moments and expectations below are finite. Define

$$\mu(q) = \mathbb{E}[A_k s_k \mid q], \qquad \sigma_r^2(q) = \text{Var}(r_k \mid q), \qquad F(q) = \mathbb{E}[s_k s_k^\top \mid q].$$

Because ξ_k depends only on the other rollouts, it is conditionally independent of (A_k, s_k), has conditional mean zero, and has conditional variance $\sigma_r^2(q)/(K-1)$. Decompose the per-prompt estimator as

$$\hat{g}_{\text{RLOO}}(q) = \underbrace{\frac{1}{K}\sum_{k=1}^{K} A_k s_k}_{\hat{g}_\star(q)} + \underbrace{\frac{1}{K}\sum_{k=1}^{K} \xi_k s_k}_{e(q)}.$$

Thus,

$$\mathbb{E}[\hat{g}_{\text{RLOO}}(q) \mid q] = \mu(q), \qquad \text{Cov}(\hat{g}_\star(q) \mid q) = \frac{\text{Cov}(A_k s_k \mid q)}{K}.$$

For the baseline-noise terms, conditional independence gives

$$\begin{aligned}\text{Cov}(\xi_k s_k \mid q) &= \mathbb{E}[\xi_k^2 s_k s_k^\top \mid q] \\ &= \mathbb{E}[\xi_k^2 \mid q]\,\mathbb{E}[s_k s_k^\top \mid q] \\ &= \frac{\sigma_r^2(q) F(q)}{K-1}.\end{aligned}$$

For $k \neq \ell$, let $U = \sum_{j\notin\{k,\ell\}} A_j$. Then U is conditionally independent of rollouts k, ℓ, $\mathbb{E}[U \mid q] = 0$, and

$$\xi_k = -\frac{A_\ell + U}{K-1}, \qquad \xi_\ell = -\frac{A_k + U}{K-1}.$$

Consequently,

$$\begin{aligned}\text{Cov}(\xi_k s_k, \xi_\ell s_\ell \mid q) &= \frac{\mathbb{E}[(A_k A_\ell + A_\ell U + A_k U + U^2) s_k s_\ell^\top \mid q]}{(K-1)^2} \\ &= \frac{\mathbb{E}[A_k A_\ell s_k s_\ell^\top \mid q]}{(K-1)^2} \\ &= \frac{\mu(q)\mu(q)^\top}{(K-1)^2}.\end{aligned}$$

The terms linear in U vanish because $\mathbb{E}[U \mid q] = 0$; the U^2 term vanishes because $\mathbb{E}[s_k \mid q] = \mathbb{E}[s_\ell \mid q] = 0$. Summing the K diagonal and $K(K-1)$ off-diagonal terms yields

$$\text{Cov}(e(q) \mid q) = \frac{\sigma_r^2(q) F(q) + \mu(q)\mu(q)^\top}{K(K-1)}.$$

The cross-covariance between $\hat{g}_\star(q)$ and $e(q)$ also vanishes. For $k = \ell$, this follows from the independence and zero mean of ξ_k. For $k \neq \ell$,

$$\mathrm{Cov}(A_k s_k, \xi_\ell s_\ell \mid q) = -\frac{\mathbb{E}[A_k^2 s_k \mid q]\,\mathbb{E}[s_\ell \mid q]^\top}{K-1} = 0.$$

Therefore,

$$\mathrm{Cov}(\hat{g}_{\mathrm{RLOO}}(q) \mid q) = \frac{\mathrm{Cov}(A_k s_k \mid q)}{K} + \frac{\sigma_r^2(q)F(q) + \mu(q)\mu(q)^\top}{K(K-1)}.$$

Recall the between-prompt and within-prompt covariances

$$\Sigma_q = \mathrm{Cov}_q(\mu(q)), \qquad \Sigma_{o|q} = \mathop{\mathbb{E}}_q[\mathrm{Cov}(A_k s_k \mid q)],$$

and define

$$\Sigma_{\mathrm{RLOO}} = \mathop{\mathbb{E}}_q[\sigma_r^2(q)F(q) + \mu(q)\mu(q)^\top].$$

Since the conditional mean remains $\mu(q)$, the law of total covariance gives

$$\begin{aligned}\mathrm{Cov}(\hat{g}_{\mathrm{RLOO}}(q)) &= \mathrm{Cov}_q(\mathbb{E}[\hat{g}_{\mathrm{RLOO}}(q) \mid q]) + \mathop{\mathbb{E}}_q[\mathrm{Cov}(\hat{g}_{\mathrm{RLOO}}(q) \mid q)] \\ &= \Sigma_q + \frac{\Sigma_{o|q}}{K} + \frac{\Sigma_{\mathrm{RLOO}}}{K(K-1)}.\end{aligned}$$

Averaging over B independent prompts therefore gives

$$\mathrm{Cov}(\hat{G}_{\mathrm{RLOO}}) = \frac{\Sigma_q}{B} + \frac{\Sigma_{o|q}}{BK} + \frac{\Sigma_{\mathrm{RLOO}}}{BK(K-1)}.$$

Applying the same quadratic optimization as in Theorem 1, we obtain

$$\epsilon_{\mathrm{opt}}^{\mathrm{RLOO}}(B,K) = \frac{\epsilon_{\max}}{1 + \frac{1}{B}\left(\sigma_{\mathrm{prompt}}^2 + \frac{\sigma_{\mathrm{rollout}}^2}{K} + \frac{\sigma_{\mathrm{RLOO}}^2}{K(K-1)}\right)},$$

where

$$\sigma_{\mathrm{prompt}}^2 = \frac{\mathrm{tr}(C\Sigma_q)}{G^\top C G}, \qquad \sigma_{\mathrm{rollout}}^2 = \frac{\mathrm{tr}(C\Sigma_{o|q})}{G^\top C G}, \qquad \sigma_{\mathrm{RLOO}}^2 = \frac{\mathrm{tr}(C\Sigma_{\mathrm{RLOO}})}{G^\top C G}.$$

Identifying the coefficient of $1/B$ gives

$$\mathcal{B}_{\mathrm{crit}}^{\mathrm{RLOO}}(K) = \sigma_{\mathrm{prompt}}^2 + \frac{\sigma_{\mathrm{rollout}}^2}{K} + \frac{\sigma_{\mathrm{RLOO}}^2}{K(K-1)}.$$

B EXPERIMENT DETAILS

This appendix records the model, data, training configuration, hyperparameter sweeps, and measurement protocol behind every figure in the paper. All runs use the verl library (Sheng et al., 2025) and were launched from a single script per experiment family; the values below are taken from the launch scripts and the logged run configurations. In all, the study comprises 469 GRPO runs and about 27K GPU hours.

B.1 MODEL, DATA, AND EVALUATION

Policy	Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024)
Training prompts	DAPO-Math (Yu et al., 2025), 17,917 problems, integer answers
Evaluation set	AIME 1983–2024 (Veeraboina, 2023), 933 problems
System prompt	"Please reason step by step, and put your final answer within `\boxed{}`."
Reward	Rule-based verifier: 1 if the boxed answer matches the reference, else 0
Metric	`mean@1`: one sampled response per problem at temperature 1.0
Evaluation cadence	Every 25 optimizer steps (every 10 or every step for $B \geq 1024$)
Maximum prompt / response length	1,024 / 3,072 tokens (overlong prompts filtered, left truncation)
Sequence length used for token counts	4,096 tokens (the prompt plus response budget)

Table 1: Model, data, and evaluation.

Training prompts are sampled without replacement within an epoch and reshuffled between epochs; runs are capped at 15 epochs. Each validation pass samples one response per AIME problem, so `mean@1` is the fraction of the 933 problems answered correctly by a single sample. The measured mean response length is close to 1,000 tokens; we nonetheless quote tokens per step as $4{,}096 \times BK$, the context budget, so that the conversion is the same constant on every figure.

B.2 TRAINING CONFIGURATION

Algorithm	GRPO (Shao et al., 2024), fully on-policy (one optimizer step per batch of rollouts)
Advantage	Group-relative, mean-centered and standard-deviation-normalized within each prompt
Policy loss	PPO-clip objective with clip ratio 0.2 (inactive on-policy, since $\rho = 1$), token-mean aggregation
KL regularization	Loss term against the initial policy, low-variance estimator, coefficient 10^{-3} (10^{-2} in the KL study)
Entropy bonus	None
Optimizer	AdamW (Kingma & Ba, 2015), $\beta = (0.9, 0.999)$, weight decay 0.01, gradient clipping at norm 1
Learning rate	Constant, no warmup; swept per batch size (Appendix B.3)
Mini-batch	The full batch of BK trained sequences (no gradient accumulation across mini-batches)
Rollout sampling	vLLM, temperature 1.0, top-p 1.0, no top-k
Precision and parallelism	bf16, FSDP over 4 GPUs, gradient checkpointing, dynamic token-length batching (32,768 tokens per GPU)
Hardware	One node with 4 NVIDIA H100 or H200 GPUs per run
Stopping	15 epochs, or the wall-clock limit of the job (25–72 hours); later runs stop once validation accuracy reaches 52%
Seeds	One seed per configuration unless noted; duplicate runs of the same configuration are treated as separate runs

Table 2: Training configuration shared by all experiments.

Loss normalization. The verl actor normalizes the policy loss of each mini-batch by the configured mini-batch size. Because our scripts pass the mini-batch size in rollouts while verl multiplies it by the number of rollouts internally, runs launched before September 11, 2026 scaled the gradient by a constant factor of $1/K$. With Adam this constant is absorbed by the second-moment normalization, and we verified that the fixed-K sweeps (the $K = 16$ prompt sweep and the $B = 128$ rollout sweep for $K \leq 16$) are unaffected; those runs are retained. The scaling is not constant in the upsampling experiment, where the number of generated and trained rollouts differ, nor across $K \geq 32$ at $B = 128$; all such runs were rerun on the corrected code, and only corrected runs enter the corresponding figures.

B.3 Hyperparameter sweeps

Every point in Figures 1–4 is the fastest run to the target among all learning rates (and both KL coefficients, unless stated otherwise) tried at that batch size. Tables 3–6 list the grids. Learning rates are in units of 10^{-6}; "longest run" is the largest number of optimizer steps logged in that cell.

B	LRs, KL 10^{-3} ($\times 10^{-6}$)	LRs, KL 10^{-2} ($\times 10^{-6}$)	runs	longest run
2	0.5, 1, 2, 3, 10	–	5	8,375
4	0.3, 0.5, 1, 3, 10	–	5	8,350
8	1, 2, 3, 4, 8, 12, 16	1, 2, 4, 8	12	7,750
16	1, 3, 4, 6, 16	2, 3, 4, 6	10	4,950
32	1, 3, 4, 6, 8, 12, 16, 32	4, 6, 12	12	4,600
64	3, 4, 6, 10, 16, 64	6, 10, 20	11	2,300
128	1, 3, 10, 32	5, 10, 20	9	2,075
256	3, 10, 12, 16, 20, 32, 64	5, 10, 20, 30	11	825
512	3, 10, 20, 50, 80, 100, 300	5, 10, 20, 50	13	425
1024	10, 20, 30, 50	20, 30, 50, 100	8	170
2048	10, 20, 30, 50	10, 20, 30, 50, 100	9	65
4096	30, 50, 100	30, 50, 100	6	49
8192	30, 50, 100	30, 50, 100	6	15

Table 3: Prompt sweep at $K = 16$ rollouts per prompt (Figures 1, 2, 3, 4b). Three cells ($B \in \{8, 32, 64\}$) additionally have runs with the KL coefficient set to zero; they are used only in Figure 3.

K	LRs, KL 10^{-3} ($\times 10^{-6}$)	LRs, KL 10^{-2} ($\times 10^{-6}$)	runs	longest run
1	1, 3, 10, 20, 30, 100	–	8	2,075
2	1, 3, 10	–	3	2,075
4	1, 3, 10	–	3	2,075
8	1, 3, 10, 30, 100	–	6	2,075
16	1, 3, 10, 32	5, 10, 20	9	2,075
32	5, 10, 20	–	3	300
64	5, 10, 20	–	3	250
128	3, 10, 30	–	3	125

Table 4: Rollout sweep at $B = 128$ prompts (Figures 1, 2, 4a, 4b).

K trained	LRs, KL 10^{-3} ($\times 10^{-6}$)	LRs, KL 10^{-2} ($\times 10^{-6}$)	runs	longest run
1	3, 10	–	4	725
2	3, 10	–	5	1,250
4	3, 10, 30	–	5	950
8	3, 10, 30	–	4	650
16	3, 10, 30	30	5	1,000
32	3, 10, 30	–	3	375

Table 5: Upsampling sweep at $B = 128$ prompts: 64 rollouts generated per prompt, K trained (Figure 4a). In Figure 4a both arms are restricted to the learning rates and KL coefficients run in both.

B	LRs, KL 10^{-3} ($\times 10^{-6}$)	LRs, KL 10^{-2} ($\times 10^{-6}$)	runs	longest run
4	1	–	1	25
8	2.5	–	1	1,625
16	1.2, 3.5, 11	–	3	700
32	5	–	1	400
64	2.3, 7, 21	–	3	200
128	5, 10, 20	–	3	250
256	4.6, 14, 42	–	3	75
512	6.6, 20	–	2	50
1024	28	–	1	25

Table 6: Prompt sweep at $K = 64$ rollouts per prompt. The central learning rate at each B follows the square-root rule fitted on the $K = 16$ sweep; where three values are listed, the outer two bracket it by a factor of three. The $B = 4$ run failed after 34 steps and is not used.

B.4 Upsampling implementation

In the upsampling experiment the rollout worker generates $N = 64$ responses per prompt and the group-relative advantage of every response is computed from all 64, exactly as in standard GRPO with $K = 64$. Before the policy update, K of the 64 responses of each prompt are selected uniformly at random without replacement, using a generator seeded by the step index, and only the $128K$ selected sequences, carrying their 64-sample advantages, are passed to the actor. The mini-batch size is set to the number of trained sequences so that the loss normalization is identical to a standard run with K rollouts. The two arms therefore differ only in the rollouts used to estimate the advantage.

B.5 Measuring the critical batch size

For each run we take the logged validation trajectory and record the first optimizer step at which `mean@1` reaches the target, linearly interpolated between the two bracketing evaluations. Runs that were resumed from a checkpoint are stitched into one trajectory. For each batch size the reported steps-to-target is the minimum over all runs in that cell, that is, over the learning-rate and KL grid, which follows the steps-to-target protocol of Shallue et al. (2019) and McCandlish et al. (2018). Critical batch sizes are obtained by fitting $S = S_{\min}(1 + N^*/N)$ to these points by least squares in $\log S$. In Figure 1, the critical batch marked on each curve is the point at which the fitted curve exceeds the perfect-scaling asymptote $S_{\min}N^*/N$ by 25%, that is, $N_{\text{crit}} = N^*/4$. For Figure 4b the fit is repeated for targets from 30% to 50% in steps of one point; the training tokens attributed to a target are $S_{\min}N^*$ sequences, the data needed in the small-batch limit, times 4,096 tokens. In the KL study a run is labelled collapsed if the mean of its last 10% of evaluations falls below 60% of its peak accuracy.

C Training trajectories

Figures 5–8 show the validation trajectories behind the steps-to-target measurements in Section 4.2 and Section 5.1. For each batch size we plot the run that reaches 50% AIME accuracy in the fewest optimizer steps over all learning rates and KL coefficients swept; its learning rate is given in the legend. When no run reaches the target, the run with the highest peak accuracy is shown as a dashed line and marked with †. Runs are shown for their full logged length, so a few of the most aggressive learning rates can be seen to become unstable after passing the target; this does not affect the steps-to-target measurement.

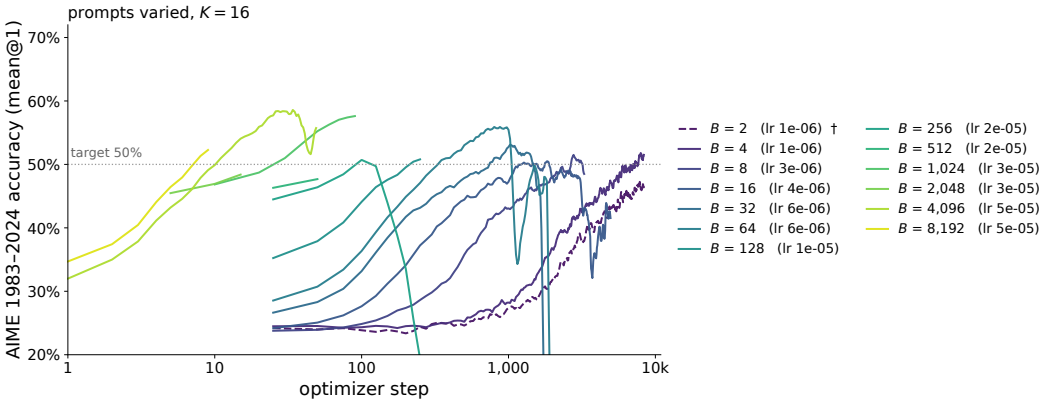


Figure 5: Scaling prompts at $K = 16$ rollouts per prompt: best run at each number of prompts B.

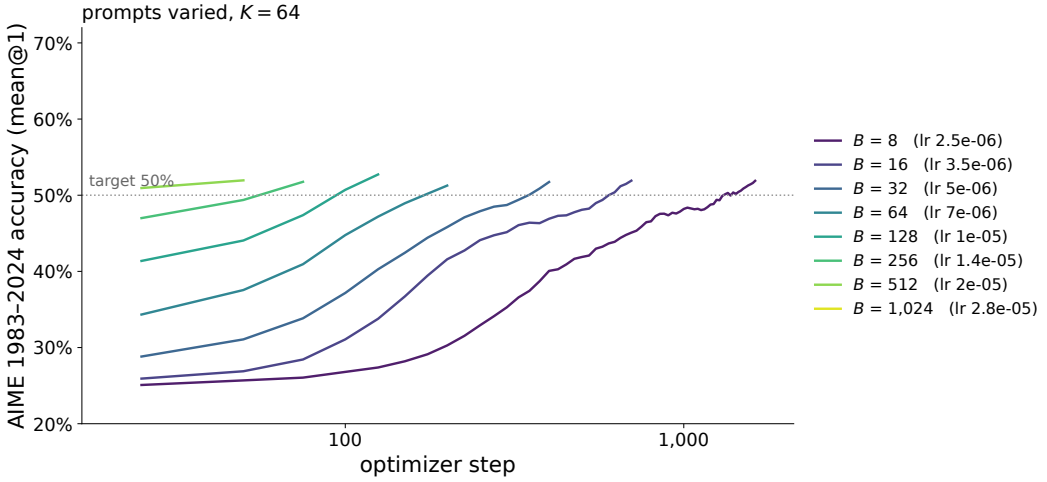


Figure 6: Scaling prompts at $K = 64$ rollouts per prompt: best run at each number of prompts B. Learning rates follow the square-root rule fitted at $K = 16$ (Figure 2b); runs stop once validation accuracy reaches 52%.

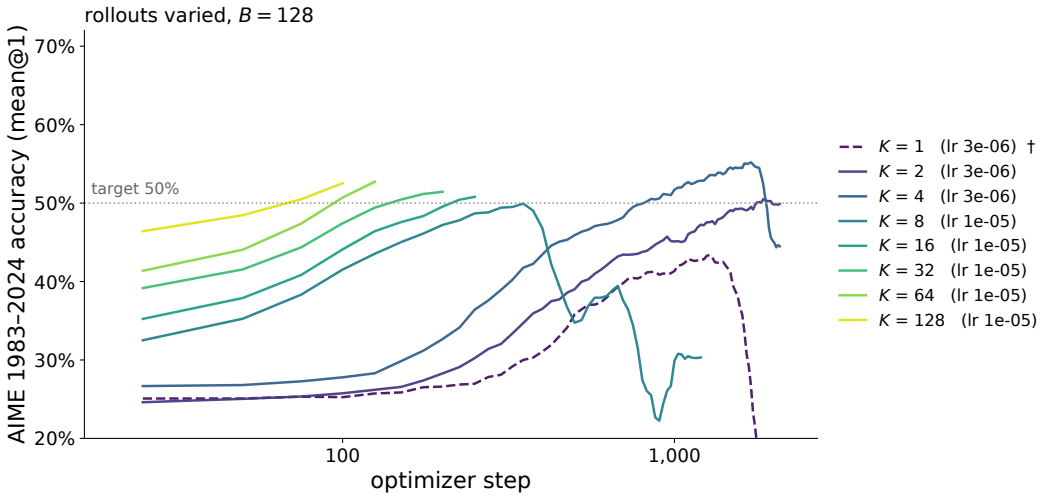


Figure 7: Scaling rollouts at $B = 128$ prompts: best run at each number of rollouts per prompt K.

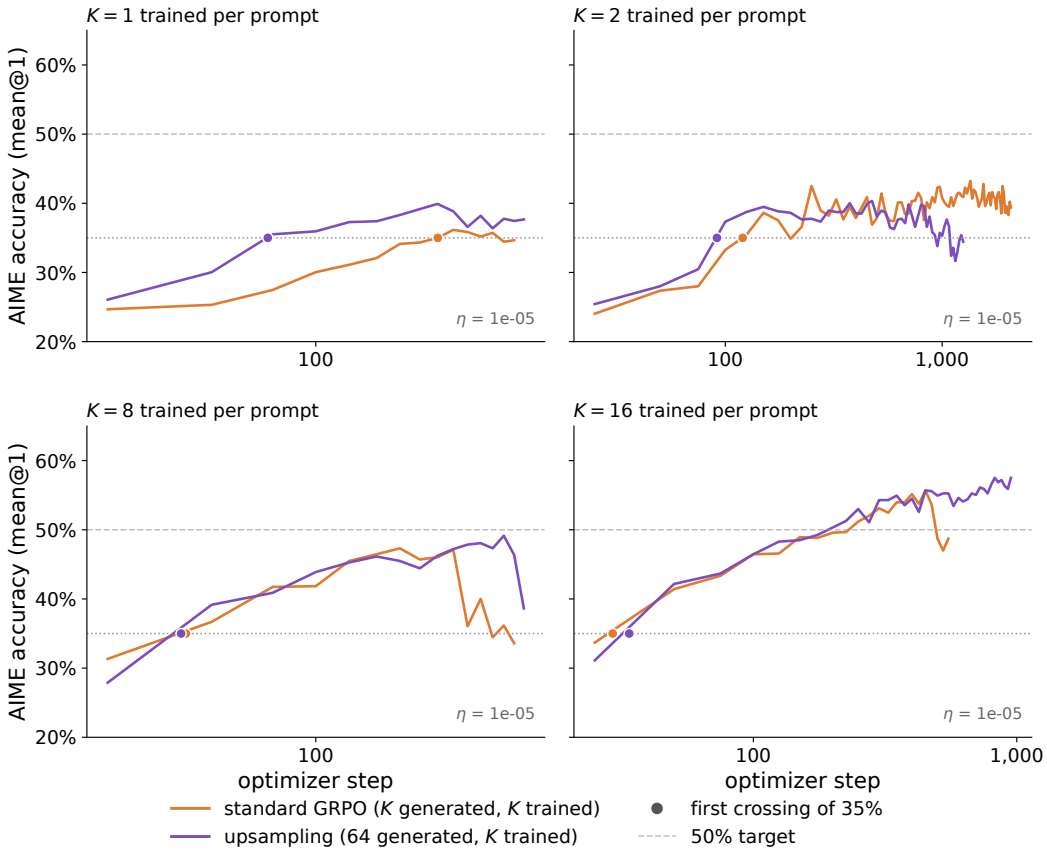


Figure 8: Upsampling versus standard GRPO at $B = 128$, one run per method at each number of trained rollouts K. Upsampling (purple) generates 64 rollouts per prompt for the advantage estimate and trains on K; standard GRPO (orange) generates and trains on the same K. Both methods are restricted to the learning rates run for both at that K, and the run shown is the one that rises fastest, i.e. first reaches 35% (dots), so that a run is shown even where neither method reaches 50% (dashed line). All runs shown use $\eta = 10^{-5}$; each trajectory is cut off once it collapses. With one or two rollouts the better advantage estimate gives a visibly faster rise; at $K \geq 8$ the two trajectories are indistinguishable.