Your Language Model is Its Own Critic: Reinforcement Learning with Value Estimation from Actor's Internal States

Yunho Choi* **Jongwon Lim***
Woojin Ahn **Minjae Oh** **Jeonghoon Shim** **Yohan Jo†**
Seoul National University
{dbsgh7177, elijah0430, awj1204, kosair, jhshim98, yohan.jo}@snu.ac.kr

Abstract

Reinforcement learning with verifiable rewards (RLVR) for Large Reasoning Models rests on variance reduction, which requires both a reliable baseline and high prompt diversity within each training batch. This is especially difficult in multi-domain training for general reasoning models, where prompts from different tasks induce highly diverse gradient signals. Existing approaches fall short in different ways: GRPO estimates its baseline as the group mean over rollouts from the same prompt, so an accurate baseline leaves fewer distinct prompts in the batch, while PPO avoids this trade-off by training a policy scale critic, roughly doubling the cost of training. We introduce **POISE** (Policy Optimization with Internal State Value Estimation), a reinforcement learning algorithm that turns the model's internal states into a value model. A lightweight probe reads the signals already computed during the forward pass to predict the baseline, and is trained online alongside the policy. To preserve gradient unbiasedness, we introduce a cross-rollout construction that predicts each rollout's value from an independent rollout's internal states. On Qwen3-4B and OLMo3-7B-Instruct-DPO across a six-domain verifiable-reward corpus, POISE outperforms other RLVR baselines while achieving more stable training. Moreover, the probe matches a separate LLM-scale value model, generalizes to various tasks, and remains accurate as the policy scales. By leveraging the model's internal representations, POISE enables stable policy optimization. [1]

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has become a key driver of progress in large language model (LLM) reasoning [Guo et al., 2025, Lambert et al., 2025], extending from basic mathematical tasks to recent multi-domain post-training pipelines [Olmo et al., 2026, Yang et al., 2025, Lambert et al., 2025, Cheng et al., 2025]. However, RL training is notoriously unstable, and this instability becomes more pronounced in multi-task settings [Teh et al., 2017]. Thus, enabling stable RL across multiple domains has become increasingly important.

One common way modern RL algorithms address this instability is through advantage estimation: rather than using the sampled reward directly, they subtract a baseline to obtain a relative measure of performance and thus reduce the gradient variance. Obtaining a reliable baseline, however, is nontrivial. GRPO and its variants, among the most widely used RLVR algorithms, estimate the

[1] Code: https://github.com/holi-lab/poise

*Equal contribution.

†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

baseline by averaging rewards over multiple rollouts from the same prompt [Shao et al., 2024, Yu et al., 2025]. Accurate baseline estimation requires a large group size, but under a fixed rollout budget, this leaves fewer distinct prompts in each training batch. Thus, improving baseline accuracy comes at the cost of prompt diversity, another key determinant of training stability. We demonstrate this trade-off both theoretically, by decomposing policy gradient covariance into within-prompt and between-prompt components (§ 2.2), and empirically (Figure 1). Taken together, these results reveal a fundamental flaw in GRPO: the same mechanism that improves baseline estimation reduces prompt diversity, potentially increasing training instability.

Alternative approaches that avoid group sampling exist, but each brings limitations of its own. PPO [Schulman et al., 2017] learns the baseline with a critic, but the critic is a LLM-scale model that processes every generated sequence, roughly doubling memory consumption and optimization cost. A second line instead relies on knowing how informative each prompt is from previous rollout outcomes [Zheng et al., 2025, Xu and Ding, 2025], and estimating it for every prompt and every model to be trained is prohibitively expensive when the training corpus is large. This raises the central question of our work: *can we obtain a reliable baseline for any domain, using only minimal rollouts and no separate value model?*

Our answer starts from a simple observation: the signal such a baseline needs may already be inside the policy model. A growing body of work shows that the hidden states of LLMs and LRMs encode outcome-relevant information, including how difficult the model perceives a problem to be [Zhang et al., 2025], whether it falls within the model's capabilities [Zhu et al., 2025], and whether an answer is correct [Cencerrado et al., 2026]. However, their use as a core component of the training algorithm itself remains underexplored. In this paper, we propose **POISE** (Policy Optimization with Internal State Value Estimation), a reinforcement learning algorithm that turns the policy model's internal states into a value model. A lightweight probe reads the policy's hidden states from before and after generation along with the uncertainty of the tokens it produced, and predicts the expected reward on that prompt, which serves as the baseline. Additionally, because a baseline read from a response would bias the gradient for that same response [Williams, 1992, Tucker et al., 2018], we adopt a *cross-rollout* construction, sampling two independent rollouts per prompt and using each rollout's baseline predicted from the other.

In a six-domain verifiable-reward training corpus [Cheng et al., 2025], POISE outperforms DAPO, a state-of-the-art RLVR algorithm, while achieving lower gradient variance (Figure 4) and maintaining stable value estimation throughout training (Figure 5). Thorough ablations on both value estimation and downstream policy optimization trace these gains to each of our design choices. A critic must scale with the policy while the probe does not, so our approach becomes more valuable as models get larger, precisely where RLVR is most expensive (Table 10). By bringing internal states into policy optimization itself, POISE attains a reliable baseline with high prompt diversity, enabling stable training.

2 Background

2.1 Preliminaries

Policy gradient and baseline estimation. We formulate RLVR for LLM reasoning as a contextual bandit problem over prompt–response pairs [Wang et al., 2026c]. Given a prompt $x \sim \mathcal{D}$ and a response $y \sim \pi_\theta(\cdot \mid x)$ sampled from the policy model, the objective is to maximize the expected verifiable reward $R(x, y)$,

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[R(x, y)\right]. \tag{1}$$

By the policy gradient theorem,

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[R(x, y)\, \nabla_\theta \log \pi_\theta(y \mid x)\right], \tag{2}$$

which yields the REINFORCE estimator [Williams, 1992]. In practice, this estimator is typically combined with a baseline $b(x)$ to reduce variance [Sutton et al., 2000], giving the advantage $A(x, y) = R(x, y) - b(x)$ and the gradient estimator

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[(R(x, y) - b(x))\, \nabla_\theta \log \pi_\theta(y \mid x)\right]. \tag{3}$$

The standard near-optimal choice for variance reduction is the value function [Weaver and Tao, 2001, Greensmith et al., 2004]

$$V^{\pi_\theta}(x) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[R(x, y)\right], \tag{4}$$

which is unknown and must be estimated in practice. PPO approximates $V^{\pi_\theta}(x)$ with a learned critic v_ϕ that is trained jointly with the policy. GRPO instead samples a group of G responses $\{y^{(1)}, \ldots, y^{(G)}\}$ for the same prompt and uses their mean reward as an empirical prompt-level baseline, $b_{\mathrm{GRPO}}(x) = \frac{1}{G}\sum_{j=1}^{G} R(x, y^{(j)})$, obtaining it directly from on-policy rollouts.

Unbiasedness condition for baselines. Subtracting a baseline preserves the unbiasedness of the policy gradient only when the baseline term has zero expectation:

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[b(x)\,\nabla_\theta \log \pi_\theta(y \mid x)\right] = 0. \tag{5}$$

This condition holds when the baseline is conditionally independent of the sampled response y given the prompt x, in which case

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[b(x)\,\nabla_\theta \log \pi_\theta(y \mid x)\right] = b(x)\,\nabla_\theta \sum_{y} \pi_\theta(y \mid x) = 0. \tag{6}$$

Equivalently, a baseline may depend only on the prompt or on any quantity that is independent of the sampled response given the prompt. Violating this condition biases the gradient and can drive the policy to converge suboptimally. We therefore adopt a *cross-rollout* construction (§3.2), where the baseline for a response is computed from another independent response, preserving Eq. 5.

2.2 Motivation: gradient variance and number of prompts in the batch

In this section, we show theoretically and empirically that, under a fixed rollout budget, using more rollouts per prompt reduces prompt diversity and increases policy-gradient variance. This trade-off naturally motivates a reliable baseline estimator that requires minimal per-prompt sampling.

Theoretical analysis. Let B be the total number of completions in a training batch, with n distinct prompts and m completions each, so $B = n \cdot m$. For prompt $x^{(i)} \sim D$ and completion $y^{(ij)} \sim \pi_\theta(\cdot \mid x^{(i)})$, define the per-sample gradient:

$$Z(x, y) = \nabla_\theta \log \pi_\theta(y \mid x)\big(R(x, y) - b(x)\big) \tag{7}$$

where $b(x)$ is a fixed baseline independent of the batch rollouts. The batch gradient estimator is:

$$\hat{g} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m} Z(x^{(i)}, y^{(ij)}). \tag{8}$$

Proposition 1 (Gradient variance decomposition)**.** *Let Σ_w and Σ_b denote the within-prompt and between-prompt covariance matrices of Z. Both Σ_w and Σ_b are fixed properties of (D, π_θ, R), independent of the allocation (n, m). Then:*

$$\mathrm{Cov}(\hat{g}) = \frac{1}{B}\Sigma_w + \frac{m}{B}\Sigma_b. \tag{9}$$

Proof. See § B.1. □

Corollary 1 (Optimal allocation)**.** *For a fixed budget B and baseline $b(x)$, the variance of $\hat{g}$ is monotonically non-decreasing in m (in the Loewner order) and is minimized at $m = 1$, $n = B$.*

Proof. See § B.2. □

Empirical validation. We next test whether the allocation effect predicted by Proposition 1 arises in practice. At each policy checkpoint, we fix the rollout budget B and compare $m = 1$ (one rollout from each of B prompts) with $m = 8$ (eight rollouts from each of $B/8$ prompts). Thus, the total number of rollouts remains fixed while prompt diversity changes. Both conditions use fixed prompt-specific baselines computed from eight separate precomputed rollouts, controlling for baseline quality. We run this comparison using OLMo-3-7B-Think-DPO on math-only and multi-domain subsets of Dolci-Think-RL-7B dataset. As shown in Figure 1, $m = 8$ consistently yields higher gradient variance, with a larger gap in the multi-domain setting. Further details are provided in Appendix C.

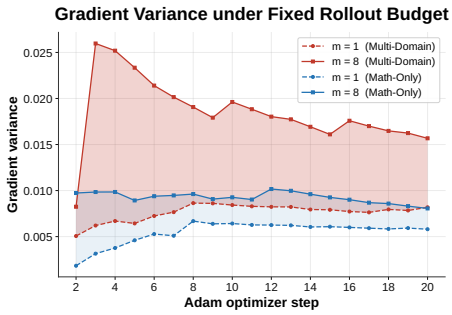


Figure 1: Estimated policy-gradient variance under a fixed total rollout budget. The $m = 1$ condition allocates one rollout to each of B distinct prompts, whereas the $m = 8$ condition allocates eight rollouts to each of $B/8$ prompts.

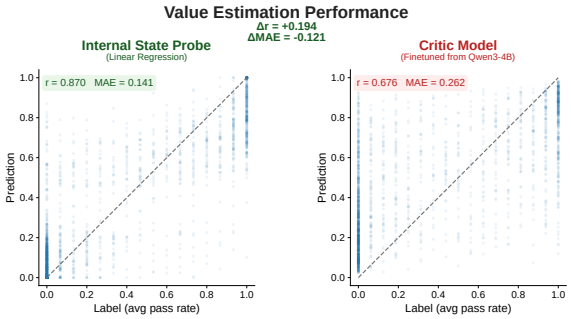


Figure 2: Value prediction between our probe and a separately trained critic. Predictions are compared against empirical Avg@8 scores. Our probe achieves higher Pearson correlation (r) and lower MAE, indicating that the policy's own internal representations provide an effective low-cost signal for value estimation.

In other words, given the same total budget, using as many diverse prompts as possible is critical to stable learning. Yet GRPO requires repeated sampling from the same prompt to estimate a faithful baseline $b(x)$. This motivates our method of estimating a reliable baseline with minimal sampling without training a separate value network as in PPO.

We provide an extended discussion of related work on value estimation in RL, outcome-relevant internal representations and multi-domain RL training in § A.

3 Policy Optimization with Internal State Value Estimation (POISE)

We now introduce POISE, which leverages the policy model's internal state signals for value estimation in RLVR. We first show that a lightweight probe can predict the value function, i.e., the expected verifier reward, directly from the policy model's internal states (§ 3.1). We then integrate this probe into policy optimization to compute per-rollout advantages, yielding the full POISE algorithm without requiring a separate LLM-scale value model (§. 3.2).

3.1 Value function estimation from policy model internal states

We introduce a probe designed to estimate baseline values directly from the policy model's internal representations. Since the viability of this method hinges on the presence of such information, we additionally present preliminary empirical results demonstrating that these internal states inherently encode the necessary signals for accurate value estimation.

Probe prediction objective. The probe is trained to predict the prompt-level value under the current policy, defined as the expected verifier reward:

$$V^{\pi_\theta}(x) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[R(x, y)].$$

Since the ground-truth quantity is unknown, we instead sample K rollouts for each prompt x, $y^{(1)}, \ldots, y^{(K)} \sim \pi_\theta(\cdot|x)$, and collect their verifier rewards $r^{(i)} = R(x, y^{(i)}) \in \{0, 1\}$. For the supervised example associated with rollout $y^{(i)}$, we use the leave-one-out Monte Carlo target as its gold value:

$$\widehat{V}_{-i}(x) = \frac{1}{K-1} \sum_{j \neq i} r^{(j)}.$$

By excluding $r^{(i)}$, $\widehat{V}_{-i}(x)$ remains conditionally independent of the input rollout $y^{(i)}$ given x, while still estimating $V^{\pi_\theta}(x)$ in expectation. This prevents the target from leaking the reward of the same rollout whose features are used by the probe.

Probe input features. As shown in Figure 3 (left), each supervised example for our probe is indexed by a prompt and one rollout, $(x, y^{(i)})$. For each pair, we construct the probe input from three complementary signals produced during the forward pass of the current policy π_θ. (All hidden-state features are extracted from a fixed layer ℓ, which we omit below for readability.) Let $H_{\theta,t}^{(i)}$ denote the residual-stream hidden state at token position t for $(x, y^{(i)})$, and let P and $R^{(i)}$ denote the final n prompt-token and reasoning-token positions, respectively.

First, we use the prompt-state feature $h_{\theta,p}^{(i)} = \mathrm{Avg}_{t\in P} H_{\theta,t}^{(i)}$, motivated by evidence that prompt hidden states encode pre-generation estimates of difficulty and capability boundaries [Zhu et al., 2025]. Second, we use the reasoning-state feature $h_{\theta,r}^{(i)} = \mathrm{Avg}_{t\in R^{(i)}} H_{\theta,t}^{(i)}$, since trajectory-level hidden states can expose value-relevant information not available from the prompt states alone [Zhang et al., 2025]. Third, we use token-level entropy statistics $u_\theta^{(i)}$ as lightweight uncertainty features [Wang et al., 2026b]: the mean predictive entropy over the vocabulary, averaged separately over all generated tokens, reasoning tokens, and answer tokens. The final probe input is

$$\phi_\theta^{(i)} = [h_{\theta,p}^{(i)};\, h_{\theta,r}^{(i)};\, u_\theta^{(i)}].$$

We ablate these input components in § 5 and the hidden-state extraction hyperparameters in § H.3. It is important to clarify that, while the input features include the generated reasoning, the estimator learns to predict the prompt-based value, rather than verifying its own reasoning, because the prediction target during training is the expected reward derived from other responses.

Probe implementation. We train lightweight regressors g_f to minimize the following loss.

$$\mathcal{L}_{\mathrm{value}}(f) = \mathbb{E}_{x,i}\left[\left(g_f(\phi_\theta^{(i)}) - \widehat{V}_{-i}(x)\right)^2\right].$$

Although our framework can theoretically support any regression architecture, we implement the probe using linear regression because its computational efficiency allows for fast, lightweight updates at each training step. This lightweight design also enables domain-specific baseline estimation by training separate probes for different domain groups, without additional policy-model computation. We provide an ablation of probe designs in § H.4, and provide detailed implementations and hyperparameters in § D.3.

Preliminary experiment. Before using the probe for policy optimization, we first test whether the policy model's internal states contain enough information to reliably estimate the prompt-level value across domains and model scales. We use reward-labeled rollouts from DAPO-Math [Yu et al., 2025], DeepScaleR [Luo et al., 2025], AceCoder [Zeng et al., 2025b], ToolDial [Shim et al., 2025], and IF-RLVR [Pyatkin et al., 2025], spanning mathematical reasoning, coding, tool calling, and instruction following. We evaluate four policy backbones—Qwen3-4B, DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and GPT-OSS-120B—and, for each model–dataset pair, compare our probe with a separately trained policy-model-scale critic [Wang et al., 2026c]. Both estimators are trained on the same rollout data and evaluated against empirical Avg@8 values on held-out prompts.

We find that our probe is competitive with and often more accurate than the critic across models and domains, despite requiring only a lightweight readout over signals already produced by the actor. Figure 2 shows a representative result on Qwen3-4B with DAPO-Math, with full results in Appendix G.1. Unlike the sequence-level critic used in this comparison, the probe also observes trajectory-level states and token-level uncertainty, giving it access to richer signals about the current policy. We additionally compare POISE with PPO using a token-level critic conditioned on generated prefixes and GAE in end-to-end policy optimization (Appendix G.2). However, these response-dependent features cannot be used naively as a baseline without biasing the policy gradient. We next introduce a cross-rollout construction that retains their predictive benefit while preserving unbiasedness.

3.2 Policy optimization with cross-rollout baselines

We now integrate the internal state probe into RL training as a value estimator, forming the full POISE algorithm (Figure 3 right).

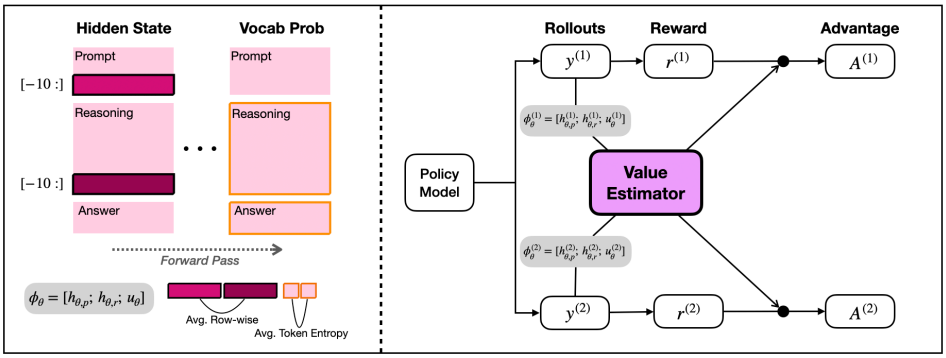


Figure 3: Overview of POISE. **Left**: Probe features $\phi(x, y, \pi)$ combine hidden states with token entropy. **Right**: The value estimator predicts each rollout's baseline from the other rollout's features.

Two rollouts per prompt. For each prompt $x \sim \mathcal{D}$ in the training batch we sample two independent rollouts from the current policy,

$$y^{(1)},\ y^{(2)} \overset{\text{i.i.d.}}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid x), \tag{10}$$

and evaluate their verifiable rewards $R(x, y^{(1)})$ and $R(x, y^{(2)})$.

Cross-rollout baseline and advantage. The baseline for each rollout is predicted from the internal signals of the *other* rollout:

$$b^{(1)}(x) \;=\; g_f\big(\phi^{(2)}\big), \qquad b^{(2)}(x) \;=\; g_f\big(\phi^{(1)}\big), \tag{11}$$

This yields the cross-rollout advantages

$$A^{(i)}(x) \;=\; R(x, y^{(i)}) - b^{(i)}(x), \qquad i \in \{1, 2\}. \tag{12}$$

By construction, the baseline used to update $y^{(i)}$ depends only on the independently sampled rollout $y^{(j)}$, $j \neq i$, satisfying the conditional-independence condition in Eq. 5.

PPO-style policy update. We optimize the policy with a PPO-style clipped surrogate objective. Let $r_t^{(i)}(\theta) = \pi_\theta(y_t^{(i)} \mid x, y_{<t}^{(i)}) / \pi_{\theta_{\text{old}}}(y_t^{(i)} \mid x, y_{<t}^{(i)})$ be the importance ratio at token t of rollout i. The objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim D,\, y^{(1)}, y^{(2)} \sim \pi_{\theta_{\text{old}}}(\cdot|x)} \left[\frac{1}{2} \sum_{i=1}^{2} \frac{1}{|y^{(i)}|} \sum_{t=1}^{|y^{(i)}|} \min\Big\{ r_t^{(i)}(\theta) A^{(i)}(x), \right.$$
$$\left. \text{clip}\Big(r_t^{(i)}(\theta), 1 - \epsilon_{\text{low}}, 1 + \epsilon_{\text{high}} \Big)\, A^{(i)}(x) \Big\} \right]. \tag{13}$$

which we maximize with respect to θ.

Online estimator training with a trajectory buffer. The value estimator g_f is trained jointly with the policy on a sliding buffer of recent trajectories. At each step, for each prompt x with two independent rollouts $(y^{(1)}, y^{(2)})$, we construct value-estimator examples $\{(x, \phi(x, y^{(i)}), \widehat{V}_{-i}(x))\}_{i=1,2}$, where $\widehat{V}_{-i}(x) = R(x, y^{(j)})$, $j \neq i$. To prevent leakage through the buffer, the current baselines are computed with g_f before it is updated on the current batch. We then update f by minimizing a regression loss over the new examples and a buffer from recent steps. The buffer stabilizes the training signal under policy drift, while the joint update keeps g_f aligned with the evolving policy. Because g_f is a lightweight probe over signals already computed during the forward pass, this update incurs negligible cost. The full procedure is summarized in Algorithm 1.

4 Experiments

4.1 Experimental setup

Training. We instantiate our method on Qwen3-4B [Yang et al., 2025] and OLMo-3-7B-Instruct-DPO [Olmo et al., 2026], and train on GURU [Cheng et al., 2025], a curated corpus of 92K verifiable prompts spanning six reasoning domains: math, code, science, logic, simulation, and tabular reasoning. Each domain is paired with its own programmatic verifier assigning a binary correctness reward. Rollouts are sampled with temperature 1.0 and top-p 1.0, and all methods are given an equal compute budget. For more details, refer to (§ D.3).

We compare against two critic-free baselines, which obtain their baseline from the group of rollouts. Throughout, m denotes the number of rollouts sampled per prompt. RLOO [Ahmadian et al., 2024] with $m = 2$ uses the same rollout allocation as POISE but replaces our predicted value with the paired rollout's reward, isolating the effect of value estimation. DAPO [Yu et al., 2025], a state-of-the-art GRPO-based RLVR algorithm, serves as our main competitive baseline; we run it with $m = 8$ and adopt the implementation of Zheng et al. [2025], which performs DAPO's dynamic sampling more efficiently. Training details and full hyperparameters are provided in § D.4.

Evaluation. We follow the evaluation suite of GURU [Cheng et al., 2025], covering all six training domains as well as held-out task types. For math we use MATH500 [Hendrycks et al., 2021] and AIME 2025 [Mathematical Association of America, 2024–2026]; for code, HumanEval [Chen et al., 2021a], MBPP [Austin et al., 2021], and LiveCodeBench [Jain et al., 2025]; for logic, Zebra Puzzle [Lin et al., 2025a] and ARC-AGI-1 [Chollet, 2019]; for simulation, CodeI/O [Li et al., 2025] and CRUXEval [Gu et al., 2024]; for science, GPQA Diamond [Rein et al., 2024] and SuperGPQA [Du et al., 2025]; and for tabular reasoning, HiTab [Cheng et al., 2022], MultiHiertt [Zhao et al., 2022], and FinQA [Chen et al., 2021b]. We additionally report LiveBench [White et al., 2025] and IFeval [Zhou et al., 2023] to assess for task types not represented in training.

For each method we select a checkpoint using the validation split released with the training corpus. The main multi-domain evaluations decode with temperature 0.6 and top-p 0.95. The full evaluation protocol is provided in § D.5.

4.2 Main results

Table 1 reports benchmark-level performance. POISE achieves the best overall average on both backbones, improving over DAPO by 1.35 points on Qwen3-4B (58.10 vs. 56.75) and 1.96 points on OLMo-3-7B-Instruct-DPO (49.92 vs. 47.96). The comparison with RLOO, which uses the same allocation of $m = 2$ but replaces our predicted baseline with the paired rollout's reward, isolates the contribution of value estimation: POISE leads it by 1.68 and 1.60 points, a margin comparable to its lead over DAPO. Notably, RLOO with $m = 2$ is itself competitive with DAPO at $m = 8$, even surpassing it on OLMo-3-7B-Instruct-DPO. This is consistent with the allocation effect in §2.2: under a fixed rollout budget, prompt diversity can offset a substantial loss in baseline quality, especially in the multi-domain setting. POISE keeps that diversity while restoring a reliable baseline. The improvements are spread across the suite rather than driven by a single benchmark: POISE scores highest among the trained methods on 11 of 17 benchmarks for both backbones, and attains the best domain average on four of the six training domains for Qwen3-4B and on all six for OLMo-3-7B-Instruct-DPO.

Additional experiments on instruction following. Our training corpus contains no training data for instruction following, so none of the methods is trained on this domain in our main experiments. For completeness, we additionally train each method on the IF-RLVR training split released with IFBench [Pyatkin et al., 2025] and evaluate on the IFBench test set and IFEval [Zhou et al., 2023]. POISE achieves the best performance on both benchmarks on Qwen3-4B and DeepSeek-R1-Distill-Qwen-1.5B, indicating that its advantage carries over to instruction following when in-domain data are available. Full results are provided in Appendix E.

Training stability. We track a proxy for policy-gradient variance during training, estimated from the Adam optimizer state following Takashiro et al. [2026]. This is not the exact gradient variance, but it is practical at LLM scale and computed identically for all methods. Since DAPO divides

Table 1: Benchmark-level performance on Qwen3-4B and OLMo-3-7B-Instruct-DPO. The best and second-best scores within each model are bolded and underlined, respectively.

		Qwen3-4B				**OLMo-3-7B-Instruct-DPO**			
Domain	Benchmark	Base	RLOO $m=2$	DAPO $m=8$	POISE $m=2$	Base	RLOO $m=2$	DAPO $m=8$	POISE $m=2$
Code	HumanEval	87.20	93.90	**95.73**	94.51	15.24	87.80	89.63	**91.46**
	LiveCodeBench	22.58	30.82	24.73	**35.13**	6.81	**32.62**	25.81	30.82
	MBPP	80.20	83.40	85.00	**85.60**	53.00	67.20	68.80	**69.60**
Logic	ARC-AGI-1	2.25	**3.50**	**3.50**	**3.50**	1.00	**2.25**	1.75	**2.25**
	Zebra Puzzle	12.02	42.39	**43.77**	41.05	18.06	24.37	28.53	**31.90**
Math	AIME(avg@32)	18.30	38.44	**44.90**	43.44	18.33	33.54	33.85	**35.83**
	MATH500	82.40	90.80	**92.80**	92.00	78.20	88.80	91.40	**92.20**
Simulation	CodeIO	11.00	13.50	15.50	**17.50**	10.50	19.00	16.00	**20.00**
	CRUXEval-I	87.00	86.12	88.00	**88.75**	**80.12**	75.75	70.38	75.00
	CRUXEval-O	87.13	87.12	80.88	**87.62**	49.12	79.75	79.62	**80.88**
Science	GPQA Diamond	42.93	54.04	54.55	**56.06**	40.40	44.44	42.42	**47.98**
	SuperGPQA	36.50	41.10	43.40	**44.30**	26.30	30.80	30.30	**31.40**
Tabular Reasoning	FinQA	**46.05**	44.78	44.50	45.78	28.61	34.97	34.79	**38.96**
	HiTab	65.00	72.10	71.50	**73.40**	36.10	56.50	57.60	**58.00**
	MultiHiertt	49.40	50.60	51.49	**52.08**	20.83	**32.44**	28.87	31.55
LiveBench	LiveBench	39.59	**45.78**	45.20	45.35	30.18	34.95	**36.59**	33.54
Instruction Following	IFEval	**83.18**	80.78	79.30	81.70	78.56	76.34	**78.93**	77.26
Average		50.16	56.42	56.75	**58.10**	34.79	48.32	47.96	**49.92**

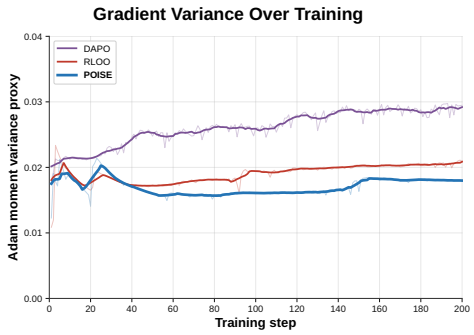


Figure 4: Adam-moment variance proxy during training (Qwen3-4B). The proxy is estimated from the optimizer's bias-corrected moments as $\mathrm{Var}(g) \approx \hat{v}_t - \hat{m}_t^2$ and summed across parameters.

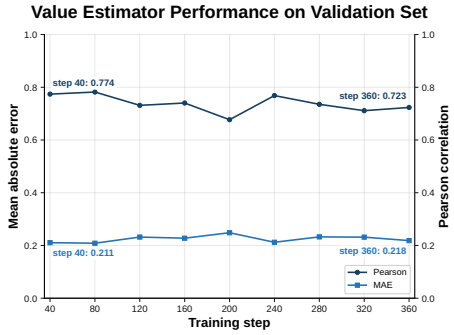


Figure 5: Accuracy of the value estimator over training steps on Qwen3-4B. Estimator predictions are compared against an oracle computed from 8 rollouts per prompt; we report MAE (left axis) and Pearson correlation (right axis).

its advantages by the group standard deviation by design, its values come out larger purely from this scale difference. For a fair comparison we multiply them back by the squared group standard deviation, estimated from the logged batch reward statistics.

Figure 4 shows the result. POISE maintains lower gradient variance than RLOO throughout training, indicating that predicting the baseline from internal states reduces variance beyond what the paired rollout's reward achieves at the same allocation. DAPO remains higher than both even after rescaling, and drifts upward as training proceeds. This suggests that insufficient prompt diversity

Table 2: Ablation of estimator input features on Qwen3-4B. We report MAE and Pearson correlation (r) against empirical Avg@8 values.

Input feature	MAE ↓	r ↑
Prompt hidden states only	0.234	0.569
Reasoning hidden states only	0.132	0.821
Mean entropy only	0.152	0.780
Response length only	0.251	0.494
Full estimator	**0.126**	**0.838**

Table 3: End-to-end ablations and comparison with a policy-model-scale critic on Qwen3-4B. We report Avg@32 over seven mathematical-reasoning benchmarks.

Method	Rollouts	Avg@32
base model	–	25.6
POISE (no trajectory-level feature)	1	47.5
POISE (no cross-rollout)	1	40.7
PPO (Token-level critic)	1	49.3
POISE (Ours)	2	**50.1**

may make training progressively less stable as the policy moves further from its initialization in a multi-domain setting.

Value estimator performance over training. We also track how the estimator itself holds up as the policy changes. Every 40 steps we sample eight rollouts per prompt on the validation set to obtain an oracle baseline, and compare the estimator's predictions against it. Figure 5 shows that both MAE and Pearson correlation remain stable throughout training, with no degradation as the policy improves. Since the value function is nonstationary during RL, this stability indicates that the online update keeps the estimator aligned with the evolving policy rather than with its initialization.

5 Analysis

We analyze the estimator through controlled ablations and compare POISE with critic-based PPO in end-to-end RL. The ablations use Qwen3-4B in a controlled mathematical-reasoning setting based on DAPO-Math-17K training set [Yu et al., 2025]. The RL comparison also includes the main multi-domain experiments on both policy backbones.

Ablations. We begin by evaluating the estimator itself, holding the policy and data fixed while varying its input features, hidden-state extraction settings, and probe architecture. As shown in Table 2, all of our three internal signals (prompt hidden states, reasoning hidden states, and mean token entropy) contain value-relevant information. However, combining all three features gives the best overall performance, suggesting that the features capture complementary aspects of prompt difficulty and policy uncertainty. Response length, by contrast, is substantially less predictive, indicating that the estimator does not simply rely on a superficial length signal. We further vary the extraction layer and pooling window, and compare the linear probe with several MLP architectures. The estimator performs well across a broad range of layers and is relatively insensitive to the pooling window. The linear probe also matches or surpasses the MLP probes in Pearson correlation, despite being substantially cheaper. Detailed results are reported in Appendices H.3 and H.4.

We next test whether these design choices actually improve online policy optimization (Table 3). We first compare full POISE with a prompt-only estimator using a single rollout. Although this comparison does not isolate the effect of trajectory-level features, the prompt-only variant benefits from twice as many distinct prompts under the same rollout budget. Full POISE nevertheless performs better, supporting the practical benefit of the two-rollout design despite its lower prompt diversity. We then remove the cross-rollout construction while retaining the full feature set. The resulting same-rollout variant performs substantially worse, confirming that cross-rollout estimation is necessary when using trajectory-dependent features.

End-to-end RL comparison to a policy-model-scale critic. We compare POISE with PPO equipped with a separate policy-model-scale critic. The critic predicts token-level values, is trained jointly with the policy, and forms advantages using generalized advantage estimation. In the controlled mathematical-reasoning setting, Table 3 shows that POISE achieves a higher downstream score than critic-based PPO.

We extend this comparison to the main multi-domain setting on Qwen3-4B and OLMo-3-7B-Instruct-DPO. POISE achieves a higher overall benchmark average on both models (Table 11). This

result shows that the actor's own representations can provide an effective value signal without training an additional policy-scale model. Because critic cost grows with the actor while POISE adds only a lightweight readout, this advantage becomes particularly relevant at larger scales. The full PPO setup and domain-level results are reported in Appendix G.2. We further show that value remains decodable across diverse domains and models, including GPT-OSS-120B, in Appendix G.1. Additional mathematical-reasoning results, including three-seed statistics and compute accounting, are reported in Appendix I.

6 Conclusion

We introduce **POISE** (Policy Optimization with Internal State Value Estimation), which predicts rollout value from the policy's internal states instead of relying on a group-mean baseline or a separate critic. To preserve unbiasedness, we couple this baseline with a cross-rollout construction. In multi-domain RL training over a six-domain verifiable-reward corpus, our method consistently outperforms existing RLVR baselines while keeping optimization stable. We further show that the value estimator closely tracks the evolving policy throughout training, generalizes across diverse task types, and scales robustly to substantially larger models.

7 Limitations and future work

Our experiments are conducted under a fixed compute budget, and while the trends we report are consistent across backbones and benchmarks, characterizing the behavior of internal state value estimation under substantially longer training horizons remains an interesting direction that we leave to future work with greater compute resources.

Several extensions naturally follow from our framework. First, our current estimator predicts value at the sequence level; extending it to token-level credit assignment would yield finer-grained advantages that more precisely reward the tokens responsible for a successful trajectory and penalize those that derail it, an effect known to be especially impactful for long reasoning trajectories [Lin et al., 2025b, Wang et al., 2026b]. Second, internal state value estimates may also be useful beyond policy gradient optimization: for preference learning algorithms such as Direct Preference Optimization [Rafailov et al., 2024], they could inform the construction of response pairs by identifying rollouts with meaningfully different predicted values, potentially yielding more informative preference comparisons. Third, we trained our estimator on verifiable rewards, but many non-verifiable tasks are instead scored by rubrics or generative judges [Gunjal et al., 2025, Su et al., 2026]. Whether a baseline for such graded and noisier rewards can likewise be predicted from internal states remains open. Finally, long-horizon and agentic tasks make group sampling even harder to afford, since a single rollout spans many turns [Hou et al., 2026]. Reading value from the internal states carried across those turns, and tracking how that value shifts from one turn to the next, is a promising direction for obtaining a baseline in this setting.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant RS-2024-00333484 and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant under the AI Computing Support Project for R&D (RS-2026-25505492, High-Performance Research AI Computing Infrastructure Support at the 2 PFLOPS Scale), both funded by the Korean government (MSIT). This research was also supported by the Korea Institute of Science and Technology Information (KISTI) in 2026 (No. (KISTI)K26L3M1C1) as part of the development of KONI (KISTI Open Neural Intelligence), a large language model specialized in science and technology. The National Supercomputing Center provided supercomputing resources and technical support (KSC-2025-CRE-0514).

We thank Haesung Pyun for helpful feedback and advice on improving the writing of this paper.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68. URL https://aclanthology.org/2023.findings-emnlp.68/.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Min Cai, Yu Liang, Longzheng Wang, Yan Wang, Yueyang Zhang, Long Xia, Zhiyuan Sun, Xi Ye, and Daiting Shi. Advancing general-purpose reasoning models with modular gradient surgery. *arXiv preprint arXiv:2602.02301*, 2026. URL https://arxiv.org/abs/2602.02301.

Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. No answer needed: Predicting LLM answer accuracy from question-only linear probes, 2026. URL https://openreview.net/forum?id=OhN25uxVab.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021a.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711. Association for Computational Linguistics, 2021b.

Zizhe Chen, Jiqian Dong, Yizhou Tian, Garry Yang, Yongqiang Chen, Zhitang Chen, and James Cheng. Hista and numca: Estimate state value effectively for large language model reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.78.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, et al. Revisiting reinforcement learning for LLM reasoning from a cross-domain perspective. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

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.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. CRUXEval: A benchmark for code reasoning, understanding and execution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 16568–16621. PMLR, 2024. URL https://proceedings.mlr.press/v235/gu24c.html.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025. URL https://arxiv.org/abs/2507.17746.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

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. *arXiv preprint arXiv:2103.03874*, 2021.

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.

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. *Advances in Neural Information Processing Systems*, 38:162239–162262, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

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 ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025.

Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. CodeI/O: Condensing reasoning patterns via code input-output prediction. In *Forty-second International Conference on Machine Learning*, 2025.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/81b8390039b7302c909cb769f8b6cd93-Paper-Conference.pdf.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. Modomodo: Multi-domain data mixtures for multimodal llm reinforcement learning. *arXiv preprint arXiv:2505.24871*, 2025. URL https://arxiv.org/abs/2505.24871.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *Forty-second International Conference on Machine Learning*, 2025a.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability, 2025b. URL https://arxiv.org/abs/2411.19943.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, 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.

Wenhan Ma, Jianyu Wei, Liang Zhao, Hailin Zhang, Bangjun Xiao, Lei Li, Qibin Yang, Bofei Gao, Yudong Wang, Rang Li, Jinhao Dong, Zhifang Sui, and Fuli Luo. Mopd: Multi-teacher on-policy distillation for capability integration in llm post-training. *arXiv preprint arXiv:2606.30406*, 2026. URL https://arxiv.org/abs/2606.30406.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). https://maa.org/maa-invitational-competitions/, 2024–2026.

NVIDIA. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025. URL https://arxiv.org/abs/2512.20856.

Minjae Oh, Sangjun Song, Gyubin Choi, Yunho Choi, and Yohan Jo. KL for a KL: On-policy distillation with control variate baseline, 2026. URL https://arxiv.org/abs/2605.07865.

Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng

Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *Advances in Neural Information Processing Systems 38: Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=yfYgwjj5F8.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Shyam Sundhar Ramesh, Xiaotong Ji, Matthieu Zimmer, Sangwoong Yoon, Zhiyong Wang, Haitham Bou Ammar, Aurelien Lucchi, and Ilija Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks. *arXiv preprint arXiv:2602.05547*, 2026. URL https://arxiv.org/abs/2602.05547.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

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

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.

Jeonghoon Shim, Gyuhyeon Seo, Cheongsu Lim, and Yohan Jo. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. *arXiv preprint arXiv:2503.00564*, 2025.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding reinforcement learning with verifiable rewards across diverse domains. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. URL https://arxiv.org/abs/2503.23829.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=HvoG8SxggZ.

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 (NIPS) 12*, pages 1057–1063. MIT Press, 2000. URL https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.

Shota Takashiro, Soichiro Nishimori, Paavo Parmas, Yongmin Kim, Kohsei Matsutani, Gouki Minegishi, Yusuke Iwasawa, Takeshi Kojima, and Yutaka Matsuo. On advantage estimates for Max@K policy gradients. *arXiv preprint arXiv:2606.06080*, 2026.

Yee Whye Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E. Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning, 2018. URL https://arxiv.org/abs/1802.10031.

Haoqing Wang, Xiang Long, Ziheng Li, Yilong Xu, Tingguang Li, and Yehui Tang. To mix or to merge: Toward multi-domain reinforcement learning for large language models. *arXiv preprint arXiv:2602.12566*, 2026a. URL https://arxiv.org/abs/2602.12566.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=yfcpdY4gMP.

Tianyi Wang, Yixia Li, Long Li, Yibiao Chen, Shaohan Huang, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. Sppo: Sequence-level ppo for long-horizon reasoning tasks, 2026c. URL https://arxiv.org/abs/2604.08865.

Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 538–545, 2001.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Runzhe Wu, Ankur Samanta, Ayush Jain, Scott Fujimoto, Jeongyeol Kwon, Ben Kretzu, Youliang Yu, Kaveh Hassani, Boris Vidolov, and Yonathan Efroni. Imbalanced gradients in rl post-training of multi-task llms. *arXiv preprint arXiv:2510.19178*, 2025. URL https://arxiv.org/abs/2510.19178.

Guowei Xu, Mert Yuksekgonul, and James Zou. Sparse reward subsystem in large language models, 2026. URL https://arxiv.org/abs/2602.00986.

Zhongwen Xu and Zihan Ding. Single-stream policy optimization, 2025. URL https://arxiv.org/abs/2509.13232.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yongjin Yang, Jiarui Liu, Yinghui He, Lechen Zhang, Bernhard Schölkopf, and Zhijing Jin. Transferability for general reasoning: An automated curriculum for multi-domain rlvr. *arXiv preprint arXiv:2606.25178*, 2026a. URL https://arxiv.org/abs/2606.25178.

Zhuolin Yang, Zihan Liu, Yang Chen, Wenliang Dai, Boxin Wang, Sheng-Chieh Lin, Chankyu Lee, Yangyi Chen, Dongfu Jiang, Jiafan He, Renjie Pi, Grace Lam, Nayeon Lee, Alexander Bukharin, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade 2: Post-training llms with cascade rl and multi-domain on-policy distillation. *arXiv preprint arXiv:2603.19220*, 2026b. URL https://arxiv.org/abs/2603.19220.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Zongji Yu, Wenshui Luo, Yiliu Sun, Hao Fang, Runmin Cong, Chaochao Lu, and Chen Gong. Harmony in diversity: Multi-domain contrastive policy optimization for large reasoning models. *arXiv preprint arXiv:2605.25443*, 2026. URL https://arxiv.org/abs/2605.25443.

Guanning Zeng, Zhaoyi Zhou, Daman Arora, and Andrea Zanette. Shrinking the variance: Shrinkage baselines for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2511.03710*, 2025a.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. ACECODER: Acing coder RL via automated test-case synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12023–12040, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.587. URL https://aclanthology.org/2025.acl-long.587/.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=O6I0Av7683.

Rui Zhang, Xinle Wu, and Yao Lu. Care-rl: Capability-aware reinforcement learning for mitigating cross-domain conflicts. *arXiv preprint arXiv:2606.00609*, 2026a. URL https://arxiv.org/abs/2606.00609.

Yi-Kai Zhang, Yueqing Sun, Hongyan Hao, Qi Gu, Xunliang Cai, De-Chuan Zhan, and Han-Jia Ye. $v_{0.5}$: Generalist value model as a prior for sparse rl rollouts, 2026b. URL https://arxiv.org/abs/2603.10848.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland, 2022. Association for Computational Linguistics.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts, 2025. URL https://arxiv.org/abs/2506.02177.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Yubo Zhu, Dongrui Liu, Zecheng Lin, Wei Tong, Sheng Zhong, and Jing Shao. The LLM already knows: Estimating LLM-perceived question difficulty via hidden representations. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1160–1176, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.61. URL https://aclanthology.org/2025.emnlp-main.61/.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Appendix Contents

A Related work **18**

B Theoretical proofs **20**

B.1 Proof of proposition 1 20

B.2 Proof of corollary 1 20

C Details of the gradient-variance experiment **21**

D Implementation details **23**

D.1 Pseudocode for POISE 23

D.2 Baseline algorithm 23

D.3 Value estimator 24

D.4 Training details 24

D.5 Evaluation protocol 25

D.6 Prompt templates 26

E Instruction-following experiments **28**

F Training dynamics **29**

G Comparisons with policy-model-scale critics **31**

G.1 Critic for held-out value prediction 31

G.2 End-to-end comparison with a token-level PPO critic 32

H Ablations **34**

H.1 Controlled setting for end-to-end ablations 34

H.2 Ablation of estimator input features 34

H.3 Ablations of hidden-state extraction 34

H.4 Ablations of probe architecture 35

H.5 End-to-end RL ablations 36

I Additional experimental evidence **37**

I.1 Additional controlled comparisons 37

I.2 Three-seed mathematical-reasoning results 37

I.3 Dynamic sampling and rollout accounting 37

I.4 Direct gradient-variance measurement 38

A Related work

Value Estimation in RL for LLM Reasoning. RL algorithms for LLMs reasoning differ mainly in how they estimate the baseline that reduces policy-gradient variance [Schulman et al., 2017, Ouyang et al., 2022, Shao et al., 2024]. Recent work extends this along three axes. The first introduces explicit value models either a generalist value prior for sparse rollouts [Zhang et al., 2026b] or a sequence-level value model that treats reasoning as a contextual bandit [Wang et al., 2026c] but incurs the substantial training, calibration, and deployment cost of an additional LLM-scale model. Chen et al. [2026] instead estimate the value of a state non-parametrically, averaging the returns of the states closest to it in hidden-state space, which requires a large group of rollouts per prompt to populate the neighborhood. Xu and Ding [2025] forgo groups, tracking a persistent per-prompt value estimate across training steps. The tracker is tabular and thus carries no information for a prompt it has not yet seen, so each prompt is initialized from substantial offline rollouts; without them the estimate only becomes informative after a prompt has been revisited over epochs, which is infeasible on corpora large enough that most prompts are seen only once. The second axis reduces rollout cost by skipping uninformative prompts before generation, predicting their informativeness from past reward dynamics [Zheng et al., 2025], which likewise depends on prompts being revisited across training. The third refines how the empirical-mean baseline is formed: Zeng et al. [2025a] apply the James–Stein principle to interpolate between the per-prompt mean and the batch mean, obtaining a lower-MSE baseline at no additional cost.

The first two axes obtain a better baseline by paying for it, either with an auxiliary model or with additional rollouts. Our method instead predicts the baseline at negligible cost from the policy's hidden states and generation signals already computed during the forward pass, requiring neither an auxiliary value model nor more than a pair of rollouts per prompt, while preserving the variance-reduction benefit of value estimation. The third axis asks how to best combine the reward means already available in a batch, whereas ours asks how to obtain a more accurate baseline in the first place; the two approaches are therefore composable and using our predicted value as the shrinkage target is a natural extension we leave to future work.

The idea of obtaining a baseline from existing model outputs also appears in on-policy distillation (OPD) [Lu and Lab, 2025]. Oh et al. [2026] formulate OPD as policy-gradient RL and derive its per-token value baseline directly as the negative reverse KL between the student and teacher distributions. This baseline is available from the existing forward pass without an additional critic or extra rollouts, illustrating how the structure of the distillation reward enables direct value computation.

Outcome-relevant Information in Hidden States. A growing body of work shows that the hidden states of large language models encode information relevant to assessing their outputs and task outcomes. Early studies on language model interpretability support this by showing that simple linear probes over internal representations can recover latent properties such as factuality [Burns et al., 2023], truthfulness [Azaria and Mitchell, 2023, Li et al., 2023], confidence [Zou et al., 2023], and answer correctness. Recent studies extend this idea to reasoning models, where activations have been used to predict final correctness [Cencerrado et al., 2026], identify capability boundaries between solvable and unsolvable prompts [Zhu et al., 2025], estimate perceived difficulty and reasoning effort [Zhang et al., 2025], and support self-verification or early stopping during generation [Sui et al., 2025]. Moving beyond black-box probes over hidden states, Xu et al. [2026] localize this information to a sparse subset of neurons, identifying value neurons whose activations predict state value and dopamine neurons that encode step-level TD errors, yet these too are deployed only at inference time.

Overall, prior work primarily uses hidden-state information as diagnostic signals or test-time control mechanisms. In contrast, we incorporate such signals directly into RL training by learning an online value estimator from the policy model's own hidden states, yielding a cheap baseline without requiring an auxiliary LLM-scale critic or large rollout groups.

Multi-Domain RL. Modern post-training pipelines combine verifiable tasks from many domains, and differ mainly in how those domains are combined. One line keeps domains apart during optimization: either training them in separate sequential stages, or training per-domain experts and consolidating them afterwards through weight merging or on-policy distillation [Yang et al., 2026b, Ma et al., 2026, Wang et al., 2026a]. The other mixes all domains into a single RL run. Recent evidence favors the latter. Wang et al. [2026a] find that mixed multi-task RLVR matches separate-

then-merge pipelines at substantially lower compute cost, and NVIDIA [2025] report that training all tasks simultaneously is more stable and less prone to reward hacking than the staged pipelines they previously used, which often degraded capabilities acquired in earlier stages. Following this practice, we train in the mixed setting, where each batch contains prompts drawn from all domains [Cheng et al., 2025].

Given heterogeneous training distributions, a growing body of work asks how to train on them well. These efforts largely fall into three groups: controlling what enters the batch, through domain-mixture optimization, transfer-aware curricula, or samplers that align each task's realized contribution with its intended weight [Liang et al., 2025, Yang et al., 2026a, Ramesh et al., 2026]; reconciling conflicting updates once they arise, through gradient surgery, task reweighting, or capability-subspace projection [Cai et al., 2026, Ramesh et al., 2026, Zhang et al., 2026a]; and shaping cross-domain representations with auxiliary objectives [Yu et al., 2026]. Underlying these is the observation that heterogeneous tasks contribute gradients of widely differing magnitudes, so naive mixing implicitly biases optimization toward large-gradient domains [Wu et al., 2025].

All of these operate either on what enters the batch or on the update it produces. Our work asks a different question: how the RL algorithm itself should be modified to train well on such batches. The starting point is that the number of distinct prompts per batch is what governs the stability of the gradient estimate, so an algorithm for this regime should be designed to preserve prompt diversity under a fixed rollout budget. POISE achieves this by estimating the baseline from the policy's own internal states, freeing the remaining budget for distinct prompts. Because it touches neither of the two levers above, it remains complementary to all of these approaches.

B Theoretical proofs

B.1 Proof of proposition 1

Proof. Define:

$$\mu(x) = \mathbb{E}[Z(x, y) \mid x], \quad \Sigma_w = \mathbb{E}_x[\mathrm{Cov}(Z(x, y) \mid x)], \quad \Sigma_b = \mathrm{Cov}_x(\mu(x)).$$

Let $G_i = \frac{1}{m} \sum_{j=1}^{m} Z(x_i, G_{ij})$. Since completions within a prompt are conditionally independent given x_i,

$$\mathrm{Cov}(G_i \mid x_i) = \frac{1}{m} \mathrm{Cov}(Z(x_i, y) \mid x_i).$$

Applying the law of total covariance to G_i:

$$\mathrm{Cov}(G_i) = \mathbb{E}_x[\mathrm{Cov}(G_i \mid x_i)] + \mathrm{Cov}_x(\mathbb{E}[G_i \mid x_i]) = \frac{1}{m}\Sigma_w + \Sigma_b.$$

Since prompts are sampled independently, $\mathrm{Cov}(\hat{g}) = \frac{1}{n} \mathrm{Cov}(G_i)$. Substituting $n = B/m$:

$$\mathrm{Cov}(\hat{g}) = \frac{m}{B}\left(\frac{1}{m}\Sigma_w + \Sigma_b\right) = \frac{1}{B}\Sigma_w + \frac{m}{B}\Sigma_b. \qquad \square$$

B.2 Proof of corollary 1

Proof. From Proposition 1, for any allocation (n, m) with $nm = B$,

$$\mathrm{Cov}(\hat{g}) = \frac{1}{B}\Sigma_w + \frac{m}{B}\Sigma_b.$$

Consider any two allocations $m_1 < m_2$ with the same budget B. Their difference is

$$\mathrm{Cov}(\hat{g}_{m_2}) - \mathrm{Cov}(\hat{g}_{m_1}) = \frac{m_2 - m_1}{B} \Sigma_b.$$

Since $\Sigma_b = \mathrm{Cov}_x(\mu(x))$ is a covariance matrix, it is positive semidefinite, so for any vector v,

$$v^\top \big(\mathrm{Cov}(\hat{g}_{m_2}) - \mathrm{Cov}(\hat{g}_{m_1})\big) v = \frac{m_2 - m_1}{B} \, v^\top \Sigma_b \, v \geq 0.$$

Therefore $\mathrm{Cov}(\hat{g}_{m_2}) \succeq \mathrm{Cov}(\hat{g}_{m_1})$ for any $m_2 > m_1$, meaning the variance of $\hat{g}$ in every gradient direction is non-decreasing in m. The minimum is thus attained at $m = 1$, giving $n = B$. $\square$

C Details of the gradient-variance experiment

Table 4: Key training hyperparameters used in the experiment.

Hyperparameter	Value
policy model	Olmo-3-7B-Think-DPO
training data	`allenai/Dolci-Think-RL-7B`
training steps	20
train batch size	64
mini batch size	64
micro batch size per GPU	1
max prompt length	2048
max response length	32680
learning rate	1×10^{-6}
LR schedule	constant (no warmup)
optimizer	AdamW
weight decay	0.01
gradient clipping	1.0
clip ratio (low / high)	0.2 / 0.28
use KL loss	False
KL in reward	False
sampling temperature	1.0
sampling top-p	1.0
max batched tokens	34728
GPUs	4×1 node

Experimental objective. The purpose of this experiment is to isolate the effect of rollout allocation on policy-gradient variance. In particular, we test whether allocating a fixed rollout budget across fewer distinct prompts increases gradient variance, as predicted by Proposition 1. This experiment does not compare different baseline estimators or training algorithms; instead, it holds the baseline fixed and varies only the number of rollouts sampled per prompt.

Policy and data. We conduct the experiment using Olmo-3-7B-Think-DPO and prompts from Dolci-Think-RL-7B. We consider two prompt distributions. In the math-only setting, prompts are drawn exclusively from the mathematical reasoning subset. In the multi-domain setting, prompts are sampled uniformly at random from the full dataset without any domain filtering, so the domain composition follows the natural mixture of Dolci-Think-RL-7B. In both settings the data seed is held fixed across rollout-allocation conditions, so that the two conditions see the same prompts in the same order and differ only in how the rollout budget is distributed.

Oracle baseline and advantage estimation. We use the pass rate that Dolci-Think-RL-7B already provides for each prompt and it is held fixed across all rollout-allocation conditions.

Under a fixed budget of trajectories per optimizer step, we compare two allocations: $m = 1$ draws a single rollout from each of B distinct prompts, while $m = 8$ draws eight rollouts from each of $B/8$ prompts. For every sampled trajectory (x, y), the advantage is

$$A(x, y) = R(x, y) - b_{\mathrm{orc}}(x), \tag{14}$$

with no group centering and no standard-deviation normalization. Because $b_{\mathrm{orc}}(x)$ does not depend on the sampled rollouts, the advantage remains well defined at $m = 1$, and both allocations are trained against an identical baseline. Any observed difference in gradient variance therefore reflects the rollout allocation itself.

Gradient variance. Computing the gradient variance exactly is impractical at LLM scale, so we adopt the Adam-state proxy of Takashiro et al. [2026]: after bias correction, the element-wise variance is estimated as $\widehat{\mathrm{Var}}(g) \approx \hat{v}_t - \hat{m}_t^2$ and summed over all parameters. This is not the exact instantaneous gradient variance, but it is computed identically in both conditions and is therefore

suitable for comparison. Because the two conditions share the same policy checkpoint, total rollout budget, and fixed oracle baseline, this measurement isolates the variance induced by the number of distinct prompts in the batch. As shown in Figure 1, the $m = 8$ allocation yields a higher variance proxy than $m = 1$ at every optimizer step in both settings, and the separation is larger under the multi-domain distribution. Prompt diversity therefore has a direct effect on gradient variance, and that effect is strongest when the training distribution is heterogeneous. Preserving diversity under a fixed budget requires obtaining a reliable baseline from as few rollouts as possible, which is the problem we address.

D Implementation details

D.1 Pseudocode for POISE

Algorithm 1 POISE

Require: Prompt distribution $\mathcal{D}$; initial policy π_{θ_0}; initial value estimator g_{f_0}; prompt batch size M; value-buffer capacity N; PPO clip $\epsilon_{\text{low}}, \epsilon_{\text{high}}$.

Initialize $\theta \leftarrow \theta_0$, $f \leftarrow f_0$, value buffer $\mathcal{B}_V \leftarrow \emptyset$.
for step $= 1, 2, \ldots, T$ **do**
  $\theta_{\text{old}} \leftarrow \theta$
  Sample a mini-batch of prompts $\{x_b\}_{b=1}^{M} \sim \mathcal{D}$.
  $\mathcal{R} \leftarrow \emptyset$, $\mathcal{S}_V \leftarrow \emptyset$
  for each prompt x_b **do**
    Sample $y_b^{(1)}, y_b^{(2)} \overset{\text{i.i.d.}}{\sim} \pi_{\theta_{\text{old}}}(\cdot \mid x_b)$.
    Extract internal state features $\phi_b^{(i)} \leftarrow \phi_{\theta_{\text{old}}}(x_b, y_b^{(i)})$ for $i \in \{1, 2\}$ via forward hooks.
    Compute rewards $r_b^{(i)} \leftarrow R(x_b, y_b^{(i)})$ for $i \in \{1, 2\}$.
    Compute cross-rollout baselines:
    $b_b^{(1)} \leftarrow g_f\big(\phi_b^{(2)}\big), \quad b_b^{(2)} \leftarrow g_f\big(\phi_b^{(1)}\big)$.
    Compute advantages $A_b^{(i)} \leftarrow r_b^{(i)} - b_b^{(i)}$ for $i \in \{1, 2\}$.
    Add policy-update examples:
    $\mathcal{R} \leftarrow \mathcal{R} \cup \big\{\big(x_b, y_b^{(i)}, A_b^{(i)}\big)\big\}_{i=1,2}$.
    Add value-estimator examples:
    $\mathcal{S}_V \leftarrow \mathcal{S}_V \cup \Big\{\big(\phi_b^{(1)}, r_b^{(2)}\big), \big(\phi_b^{(2)}, r_b^{(1)}\big)\Big\}$.
  end for
  Update θ by maximizing the PPO objective in Eq. 13 on $\mathcal{R}$.
  Update f by minimizing

$$\sum_{(\phi, \widetilde{r}) \in \mathcal{B}_V \cup \mathcal{S}_V} \big(g_f(\phi) - \widetilde{r}\big)^2.$$

  Append $\mathcal{S}_V$ to $\mathcal{B}_V$; retain only the N most recent examples.
end for
return π_θ, g_f.

D.2 Baseline algorithm

We adopt DAPO [Yu et al., 2025] as our baseline RL algorithm. For each prompt $q \sim \mathcal{D}$, a group of G responses $\{o^{(i)}\}_{i=1}^{G}$ is sampled from the old policy $\pi_{\theta_{\text{old}}}$, and the following objective is optimized:

$$\mathcal{J}_{\text{DAPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\, \{o^{(i)}\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G} |o^{(i)}|} \sum_{i=1}^{G} \sum_{t=1}^{|o^{(i)}|} \min\Big(r_t^{(i)}(\theta)\, \hat{A}_t^{(i)},\ \text{clip}\big(r_t^{(i)}(\theta),\, 1-\varepsilon_{\text{low}},\, 1+\varepsilon_{\text{high}}\big)\hat{A}_t^{(i)} \Big) \right], \tag{15}$$

$$\text{s.t.} \quad 0 \ < \ \Big| \{\, o^{(i)} \mid \text{is_equivalent}(a, o^{(i)}) \,\} \Big| \ < \ G,$$

where the importance ratio and the group-relative advantage are

$$r_t^{(i)}(\theta) = \frac{\pi_\theta(o_t^{(i)} \mid q, o_{<t}^{(i)})}{\pi_{\theta_{\text{old}}}(o_t^{(i)} \mid q, o_{<t}^{(i)})}, \qquad \hat{A}_t^{(i)} = \frac{R^{(i)} - \text{mean}(\{R^{(j)}\}_{j=1}^{G})}{\text{std}(\{R^{(j)}\}_{j=1}^{G})}.$$

DAPO augments GRPO with four key techniques: (i) **Clip-Higher** decouples the lower and upper clipping bounds ($\varepsilon_{\text{high}} > \varepsilon_{\text{low}}$), giving low-probability tokens more room to be promoted and mitigating entropy collapse; (ii) **Dynamic Sampling** filters out prompts whose responses are all correct or all incorrect, ensuring that every batch yields effective gradients; (iii) **Token-Level Policy Gradient Loss** normalizes by the total token count $\sum_i |o^{(i)}|$ rather than per-sequence, so longer responses

contribute proportionally to the loss; (iv) **Overlong Reward Shaping** applies a length-aware penalty to truncated samples to reduce reward noise.

Reward. Unlike the original DAPO, we drop the length penalty and use a purely binary reward based solely on correctness for simplicity:

$$R^{(i)} = \begin{cases} 1, & \text{if } o^{(i)} \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

D.3 Value estimator

Hidden states used for estimator training are collected from the model's teacher-forced log-probability forward pass, which is already required by our policy optimization algorithm and therefore incurs no additional policy-model forward pass. Specifically, our implementation registers a forward hook on a single transformer layer during this computation and separately pools the prompt and reasoning hidden states.

For the single-domain mathematical-reasoning (§ H.1) and instruction-following (§ E) experiments, we use a single estimator. In the main multi-domain experiments, however, the domain label of each training prompt is known, allowing us to use domain-specific estimators without evaluating multiple probes for each trajectory. Each trajectory is routed to exactly one lightweight estimator, so this design introduces no additional policy-model computation relative to using a single estimator. Specifically, we maintain three estimators: one for mathematics, one for code, and one shared by logic, science, simulation, and tabular reasoning. During the first 16 steps, no estimator is applied and training proceeds with the paired rollout's reward as the baseline, which fills the trajectory buffer at no additional rollout cost. Each estimator is then fitted to the trajectories of its domain group and updated online thereafter.

Table 5 lists the model-specific estimator hyperparameters.

Table 5: Hyperparameters of the hidden-state value estimators.

Component	**Qwen3-4B**	**OLMo-3-7B-Instruct-DPO**
Hidden layer	19	19
Prompt hidden-state pooling	Last 10 prompt-token mean	Last 4 prompt-token mean
Reasoning hidden-state pooling	Last 10 reasoning-token mean	Last 4 reasoning-token mean
Scalar features	3 entropy statistics	3 entropy statistics
Final estimator input dimension	5123	8195
Regressor	StandardScaler + Ridge	StandardScaler + Ridge
Ridge penalty	$\alpha = 0.01$	$\alpha = 100$
Estimator routing	Math / code / others	Math / code / others
Random seed	42	42
Prediction range	Clipped to $[0, 1]$	Clipped to $[0, 1]$

D.4 Training details

We instantiate our method on two reasoning model backbones, Qwen3-4B and OLMo-3-7B-Instruct-DPO, the checkpoint released immediately before the RLVR stage of the OLMo3 post-training pipeline. All training is implemented in the `verl` library. As training data we use GURU [Cheng et al., 2025], a curated corpus of 92K verifiable examples spanning six reasoning domains: mathematics, code, science, logic, simulation, and tabular reasoning. Each domain is paired with its own verifier, so that a single training run receives heterogeneous but automatically checkable rewards, and each batch contains prompts drawn from all six domains. All experiments are run on H100 GPUs. We allocate a fixed compute budget of two days on four GPUs for Qwen3-4B and six days on four GPUs for OLMo-3-7B-Instruct-DPO.

Our main baseline is DAPO [Yu et al., 2025], for which we adopt the implementation of Zheng et al. [2025] that improves the efficiency of DAPO's dynamic sampling, and follow the configu-

rations of the original paper for all other settings. RLOO uses exactly the same configuration as POISE, except that each rollout's baseline is the other rollout's observed reward. Across all methods, each optimization step consumes the same number of rollouts. Detailed hyperparameter values are reported in Tables 6, 7.

Table 6: Key training hyperparameters used in POISE, RLOO

Hyperparameter	**Value**
algorithm	POISE, RLOO
training epochs	2
train batch size	256
samples per prompt	2
max prompt length	4096
max response length	8192
learning rate	1×10^{-6}
clip ratio (low / high)	0.2 / 0.28
use KL loss	False
sampling temperature	1.0
sampling top-p	1.0

Table 7: Key training hyperparameters used in DAPO with efficient dynamic sampling [Zheng et al., 2025].

Hyperparameter	**Value**
algorithm	DAPO
training epochs	2
train batch size	64
samples per prompt	8
p_{easy}	0.5
p_{hard}	0.5
target zero variance	0.25
default br size	96
GRESO min p	0.05
GRESO max p	0.95
β	1.25
max prompt length	4096
max response length	8192
learning rate	1×10^{-6}
clip ratio (low / high)	0.2 / 0.28
use KL loss	False
sampling temperature	1.0
sampling top-p	1.0

D.5 Evaluation protocol

This section details our main multi-domain evaluation protocol and per-domain scoring rules.

Evaluation protocol. We follow the officially recommended decoding setting with temperature 0.6 and top-p 0.95, and set the maximum response length to 8192 tokens the same as the training setting. All inferences are performed with the vLLM library [Kwon et al., 2023] on a single node equipped with four NVIDIA H100 GPUs. We evaluate on 17 benchmarks spanning six domains.

Per domain scoring rules. All benchmarks are scored by deterministic, execution- or rule-based verifiers; we use no LLM judge. Each response is reduced to a single answer and mapped to a binary score, except where noted. When a response contains a `</think>` delimiter, only the text after the final delimiter is scored, so that reasoning traces are never mistaken for the answer.

- **Math** (MATH-500, AIME 2025). The prompt appends *"Please output the final answer within* \boxed{} *"* to the problem statement. We extract the last \boxed{} expression and compare it to the reference answer by symbolic equivalence rather than string match, so that algebraically equal forms are accepted.
- **Code generation** (HumanEval, MBPP, LiveCodeBench). The prompt asks for a complete, self-contained Python solution in a markdown code block; LiveCodeBench uses the stdin/stdout formulation. We extract the final code block and execute it in a sandbox against the reference test cases, scoring 1 only if every test passes.
- **Simulation** (CRUXEval-I, CRUXEval-O, CodeIO). CRUXEval asks the model to complete an assertion, predicting an input that yields a given output (I) or the output of a given input (O), inside a ‘‘‘ python block. We concatenate the extracted assertion with the reference function and execute it, scoring 1 if the assertion holds. CodeIO asks for a feasible input as a JSON object, which we compare against the reference input.
- **Logic** (Zebra Puzzle, ARC-AGI-1). Both prompts require the answer inside <answer> </answer> tags: a grid dictionary for Zebra Puzzle and a 2D array for ARC-AGI-1. ARC-AGI-1 is scored as exact grid match. Zebra Puzzle is the one benchmark scored continuously: we report the fraction of correctly filled cells, since a fully binary score is uninformative on grids of this size.
- **Table reasoning** (HiTab, FinQA, MultiHiertt). The prompt supplies the table and, for FinQA, the surrounding text, and asks for the final answer within \boxed{}. We extract the boxed answer and compare it to the reference with numeric tolerance and unit normalization; multi-part answers are compared element-wise after sorting.
- **STEM** (GPQA-Diamond, SuperGPQA). Both are multiple choice with the instruction to place only the option letter within \boxed{}. We extract that letter and score exact match; a missing or malformed box scores 0.
- **Out-of-domain** (IFEval, LiveBench-Reasoning, LiveBench-Language, LiveBench-Data-Analysis). IFEval carries no added instruction beyond its own constraints; we run the official verifiable-instruction checkers and score 1 only if *all* constraints of a prompt are satisfied. LiveBench prompts specify a task-appropriate answer format, and we apply the official per-task scorer for each split.

D.6 Prompt templates

Each domain uses a fixed single-turn template that encodes the answer format its verifier expects; the templates are identical at training and inference time and across models, and each rendered prompt is then wrapped in the evaluated model's own chat template. The placeholder {problem} is replaced with the problem statement.

Mathematical reasoning (MATH-500, AIME 2025)

{problem} Please output the final answer within \boxed{}.

Code generation (HumanEval, MBPP)

Write a complete, self-contained Python solution to the following problem. Your solution must include all necessary imports and the full function definition including the signature exactly as specified. Do not modify the function signature or docstring.

{problem}

(MBPP additionally lists the assertion test cases the solution must pass, followed by:) Please do not include the test cases in your solution.

Code generation (LiveCodeBench)

You are an expert Python programmer. You will be given a question (problem specification) and will generate a correct Python program that matches the specification and passes all tests.

Below is the question:

{problem}

Read the inputs from stdin solve the problem and write the answer to stdout (do not directly test on the sample inputs). Enclose your code within delimiters.

Simulation (CRUXEval-O)

You will be given a Python function and an assertion containing an input to the function. Complete the assertion with a literal (no unsimplified expressions, no function calls) containing the output when executing the provided code on the given input, even if the function is incorrect or incomplete. Do NOT output any extra information. Execute the program step by step before arriving at an answer, and provide the full assertion with the correct output in ```python and ``` tags, following the examples.

{problem}

CRUXEval-I inverts the direction, asking for any input satisfying `f(??) == output`; CodeIO asks the model to predict a feasible input as a JSON object without writing code.

Logic (ARC-AGI-1)

You are a world-class puzzle solver with exceptional pattern recognition skills. Your task is to analyze puzzles, spot patterns, and provide direct solutions. Given input-output grid pairs as reference examples, carefully observe the patterns to predict the output grid for new test input.

{problem}

What is the output grid? Please put your answer within <answer> and </answer> tags, your final answer should be only the output grid (2d array).

Zebra Puzzle uses the same <answer> convention, but asks for the completed grid as a dictionary with `header` and `rows` keys.

Table reasoning (HiTab, FinQA, MultiHiertt)

You are given one or more tables. Use the information in the table to answer the following question.

{table}

The question is:
{problem}
Please output the final answer within \boxed{}. *(HiTab adds: If there are multiple answers, please output them separated by |.)*

STEM (GPQA-Diamond, SuperGPQA)

{problem}
{options}

Please reason step by step, and put your final answer option within \boxed{}. Only put the letter in the box, e.g. \boxed{A}. There is only one correct answer.

IFEval and LiveBench are used verbatim as released: IFEval prompts already carry their own verifiable constraints, and each LiveBench task specifies its own answer format (for example, *"put your answer in **bold** as a single integer"*). We add no instruction of our own to either.

E Instruction-following experiments

Table 8: Prompt-level loose accuracy (%) after instruction-following training. The best and second-best results for each backbone are bolded and underlined, respectively.

Backbone	Method	IFEval	IFBench	Average
Qwen3-4B	Base	83.0	34.3	58.7
	RLOO ($m = 2$)	81.7	35.0	58.4
	DAPO ($m = 8$)	<u>83.9</u>	<u>37.7</u>	<u>60.8</u>
	POISE ($m = 2$)	**84.8**	**39.7**	**62.3**
DeepSeek-R1-Distill-Qwen-1.5B	Base	36.0	10.3	23.2
	RLOO ($m = 2$)	52.9	15.3	34.1
	DAPO ($m = 8$)	<u>54.0</u>	<u>16.7</u>	<u>35.3</u>
	POISE ($m = 2$)	**56.6**	**17.3**	**36.9**

Setup. We evaluate POISE in instruction-following RL using the IF-RLVR training split released with IFBench [Pyatkin et al., 2025]. We use the same optimization setup as in our main experiments and compare POISE and RLOO with two rollouts per prompt ($m = 2$) against DAPO with eight rollouts per prompt ($m = 8$). We evaluate on the IFBench test set, which contains constraint types not observed during training, and on IFEval [Zhou et al., 2023]. Following the IFBench evaluation protocol, we use greedy decoding with temperature 0 and report prompt-level loose accuracy for both benchmarks.

Results. Table 8 presents the results. POISE achieves the highest accuracy on both evaluation sets and across both backbones. Compared with DAPO, POISE improves IFBench accuracy by 2.0 percentage points on Qwen3-4B and 0.6 points on DeepSeek-R1-Distill-Qwen-1.5B. It also improves IFEval accuracy by 0.9 and 2.6 points, respectively. The consistent gains across in-domain instruction-following training and unseen evaluation constraints provide further evidence that POISE is applicable beyond mathematical reasoning.

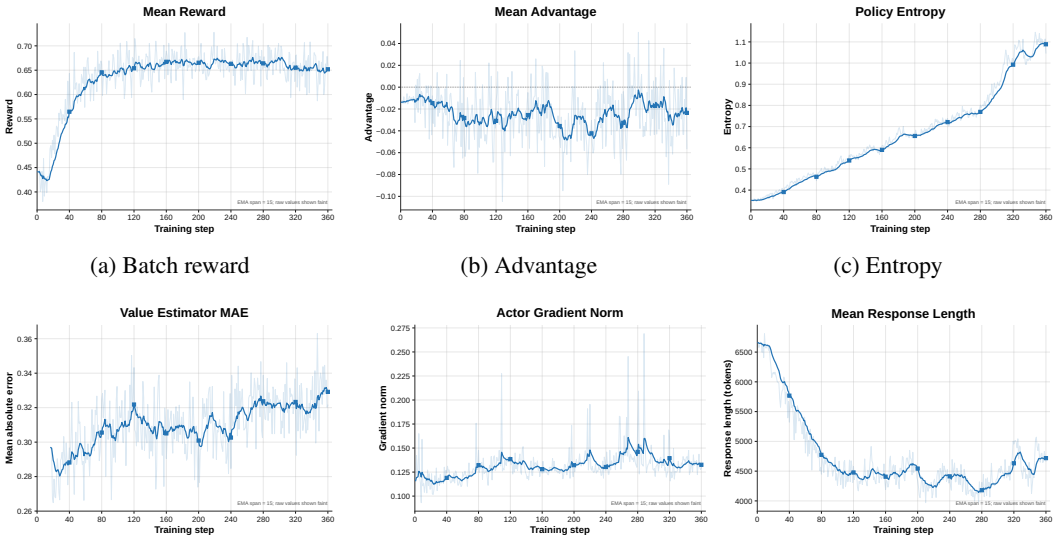
F Training dynamics

We provide additional training dynamics for POISE on Qwen3-4B and Olmo-3-7B-Instruct-DPO. In the main text, we report aggregate evidence that the internal state value estimator remains stable during training. Here, we expand the analysis by tracking six quantities over policy updates: the batch reward, advantage, token-level entropy, the value-estimation error against the online target, gradient norm and response length.

For each training step t, let R_t denote the verifier reward of the sampled rollouts, $b_t = g_f(\phi_t)$ the predicted baseline, and $A_t = R_t - b_t$ the resulting advantage. We report the batch reward $\bar{R}_t = \mathbb{E}_{\text{batch}}[R_t]$, the online target error $\text{MAE}_t = \mathbb{E}_{\text{batch}}[|b_t - \widehat{V}_t(x)|]$. Here, $\widehat{V}_t(x)$ is the empirical online target estimated from rollouts sampled from the current checkpoint policy.

Qwen3-4B. Figure 6 shows that the mean reward rises rapidly from approximately 0.45 to 0.65 during the first 100 steps and remains near that level thereafter. The mean advantage stays within a relatively narrow negative range without sustained drift. Despite the substantial change in policy reward, the online-target MAE remains bounded between approximately 0.28 and 0.33, with only a modest upward trend, indicating that the estimator continues to track the evolving policy.

Policy entropy increases throughout training, with a sharper rise near the end, while response length drops rapidly from approximately 6,500 to 4,500 tokens before stabilizing. Thus, the reward improvement is not driven by increasing response length or collapsing policy entropy, and both the estimator and policy optimization remain stable after the initial adaptation phase.



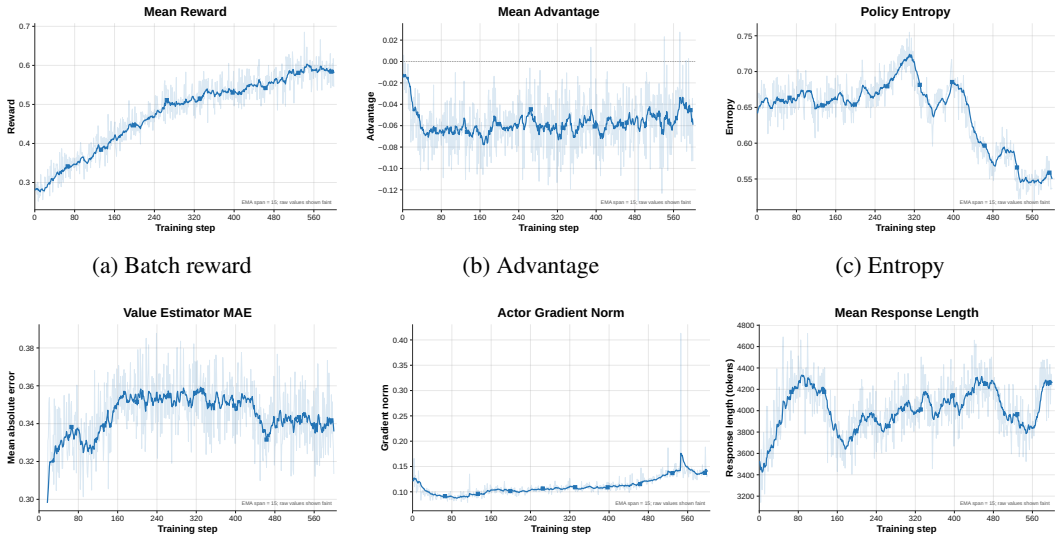
(a) Batch reward (b) Advantage (c) Entropy

(d) Online target MAE (e) Gradient norm (f) Response length

Figure 6: Training dynamics of POISE on Qwen3-4B.

OLMo-3-7B-Instruct-DPO. Figure 7 shows a more gradual improvement: the mean reward increases steadily from approximately 0.28 to 0.60 over training. The mean advantage settles into a narrow negative range after the initial steps, while the online-target MAE rises during early adaptation and then remains around 0.34–0.36 for most of training. The actor gradient norm likewise stabilizes after an initial decline, aside from an isolated spike late in training.

Unlike Qwen3-4B, policy entropy evolves non-monotonically, increasing modestly during the first half of training before declining later. Response length also fluctuates after its initial increase but exhibits no sustained growth. Despite these model-specific changes in exploration and generation behavior, reward continues to improve without a corresponding divergence in estimation error or gradient magnitude.



(a) Batch reward (b) Advantage (c) Entropy

(d) Online target MAE (e) Gradient norm (f) Response length

Figure 7: Training dynamics of POISE on Olmo-3-7B-Instruct-DPO.

G Comparisons with policy-model-scale critics

We compare POISE with policy-model-scale critics in two settings. The first examines held-out value prediction and supports the preliminary analysis in Section 3.1. The second directly compares POISE with a token-level critic in end-to-end RL.

G.1 Critic for held-out value prediction

Table 9: Hyperparameters for the held-out prompt-level scalar critic.

Hyperparameter	Value
Trainable parameters	Full critic backbone and scalar head
Objective	Bernoulli cross-entropy (`BCEWithLogitsLoss`)
Optimizer	AdamW
Backbone/head learning rate	10^{-5} / 10^{-4}
Learning-rate schedule	None
Weight decay	0.01 (bias and normalization parameters excluded)
Batch size	8 / 16
Gradient clipping	Global norm 1.0
Precision	FP32 (BF16 autocast)

For the held-out value-prediction comparison, we implement a prompt-level scalar critic following the sequence-level critic design of SPPO [Wang et al., 2026c]. For each policy backbone, the critic is initialized from the same pretrained checkpoint as the policy and augmented with a bias-free scalar value head. Given a prompt x, it processes the prompt only, extracts the hidden state at the final non-padding prompt token, and predicts

$$z_\phi(x) = w^\top h_\phi(x), \qquad V_\phi(x) = \sigma(z_\phi(x)) \,. \tag{16}$$

We optimize both the scalar head and the full critic backbone, except for GPT-OSS 120B where we use a LoRA adapter instead due to memory constraints. Thus, although the critic produces a single prompt-level scalar, it trains a policy-model-scale representation rather than fitting only a small output head.

For each training prompt, we sample K_{train} responses from the target policy and obtain binary rewards $r^{(j)} = R(x, y^{(j)})$. The implementation minimizes Bernoulli cross-entropy over the individual prompt–rollout pairs,

$$\mathcal{L}_{\text{critic}}(\phi) = -\frac{1}{K_{\text{train}}} \sum_{j=1}^{K_{\text{train}}} \left[r^{(j)} \log V_\phi(x) + \left(1 - r^{(j)}\right) \log\left(1 - V_\phi(x)\right) \right]. \tag{17}$$

Because all rollouts for a prompt share the same prediction, this objective is exactly equivalent to soft-label cross-entropy with

$$\widehat{V}(x) = \frac{1}{K_{\text{train}}} \sum_{j=1}^{K_{\text{train}}} R\Big(x, y^{(j)}\Big) \,. \tag{18}$$

We train this critic and the internal-state estimator on the same reward-labeled examples and evaluate both against empirical Avg@8 values on held-out prompts using MAE and Pearson correlation. The critic is used only as an analysis baseline and does not participate in POISE training.

Comparisons on multiple domains and models We evaluate whether our internal-state value estimator generalizes to different verifiable-reward domains and policy backbones. We consider five datasets spanning math reasoning, coding, tool use, and instruction following: DAPO-Math [Yu et al., 2025], DeepScaleR [Luo et al., 2025], AceCoder [Zeng et al., 2025b], ToolDial [Shim et al., 2025], and IF-RLVR [Pyatkin et al., 2025]. We also evaluate four policy backbones: Qwen3-4B, DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B, and GPT-OSS 120B.

Table 10: Full value-prediction results across policy backbones and verifiable-reward domains. Where available, we compare our internal-state estimator against a separately trained critic and report MAE and Pearson correlation r.

Policy model	Domain	Dataset	Critic		Internal-state estimator	
			MAE ↓	r ↑	MAE ↓	r ↑
Qwen3-4B	Math reasoning	DAPO-Math	0.262	0.676	**0.141**	**0.870**
		DeepScaleR	0.393	0.384	**0.231**	**0.609**
	Coding	AceCoder	0.499	0.056	**0.234**	**0.612**
	Tool calling	ToolDial	0.303	0.440	**0.188**	**0.840**
	Instruction following	IF-RLVR	0.350	0.150	**0.195**	**0.642**
DeepSeek-R1-Distill-Qwen-1.5B	Math reasoning	DAPO-Math	0.127	0.723	**0.123**	**0.834**
		DeepScaleR	0.251	0.586	**0.151**	**0.829**
	Coding	AceCoder	**0.135**	**0.706**	0.196	0.580
	Tool calling	ToolDial	0.201	0.337	**0.122**	**0.672**
	Instruction following	IF-RLVR	**0.101**	0.545	0.171	**0.557**
DeepSeek-R1-Distill-Qwen-7B	Math reasoning	DAPO-Math	0.300	0.441	**0.191**	**0.784**
		DeepScaleR	0.270	0.457	**0.164**	**0.814**
	Coding	AceCoder	0.229	0.691	**0.173**	**0.804**
	Tool calling	ToolDial	0.194	0.663	**0.153**	**0.842**
	Instruction following	IF-RLVR	**0.155**	**0.716**	0.191	0.596
GPT-OSS-120B	Math reasoning	DAPO-Math	0.241	0.495	**0.206**	**0.661**
		DeepScaleR	0.219	0.531	**0.166**	**0.806**
	Coding	AceCoder	**0.222**	0.602	0.233	**0.623**
	Tool calling	ToolDial	0.191	0.752	**0.179**	**0.836**
	Instruction following	IF-RLVR	0.281	0.341	**0.190**	**0.671**

For each model–dataset pair, we train the internal-state estimator and a separately trained critic on the same reward-labeled rollout data. To make the comparison consistent across settings, we use 4,096 training examples for each estimator, matching the number of examples stored in the value-estimator buffer used by the POISE algorithm. We then evaluate both estimators against empirical Avg@8 values on held-out prompts. For evaluation, we use 1,024 held-out prompts for all datasets except ToolDial, for which we use 13,492 held-out examples.

Table 10 shows that our internal-state estimator outperforms the critic in most settings, reducing MAE in 16 of 20 model–dataset pairs and improving Pearson correlation in 18. On Qwen3-4B, the estimator improves both metrics across all five datasets, with particularly large gains outside the original math setting: Pearson correlation increases from 0.056 to 0.612 on AceCoder and from 0.150 to 0.642 on IF-RLVR. The trend also holds for larger models. On both DeepSeek-R1-Distill-Qwen-7B and GPT-OSS-120B, the estimator improves both metrics on four of the five datasets. Results are more mixed but still favorable overall on DeepSeek-R1-Distill-Qwen-1.5B: the estimator improves both metrics on DAPO-Math, DeepScaleR, and ToolDial, while the critic performs better on AceCoder and achieves lower MAE on IF-RLVR.

Overall, these results show that value-relevant information can be recovered from policy internal states across different domains and model scales, motivating our use of these signals for baseline estimation within the POISE training algorithm (§ 3.2).

G.2 End-to-end comparison with a token-level PPO critic

The preceding held-out comparison uses a prompt-level critic to evaluate value prediction. We now compare complete RL systems in the main multi-domain setting. We run PPO with a separate, policy-model-scale token-level critic on Qwen3-4B and OLMo-3-7B-Instruct-DPO. The policy initialization, six-domain training mixture, actor-side optimization settings, per-update rollout budget,

Table 11: Token-level PPO critic configuration **(a)** and domain-level performance **(b)**. Domain scores are averaged over the corresponding benchmarks in Table 1; Overall averages all 17 benchmarks. PPO uses $m = 1$ and POISE uses $m = 2$. OLMo-3-7B abbreviates OLMo-3-7B-Instruct-DPO.

(a) Critic configuration

Hyperparameter	Value
Initialization	Actor checkpoint
Trainable parameters	Full critic
Critic learning rate	10^{-5}
Critic value clipping	0.5
Critic objective	Clipped value regression
GAE (γ, λ)	$(1.0, 1.0)$

(b) Domain-level performance

	Qwen3-4B		**OLMo-3-7B**	
Domain	PPO	POISE	PPO	POISE
Code	**73.05**	71.75	62.46	**63.96**
Logic	13.50	**22.28**	**18.68**	17.08
Math	**69.28**	67.72	62.22	**64.02**
Simulation	**65.33**	64.62	**59.63**	58.63
Science	49.28	**50.18**	34.60	**39.69**
Tabular reasoning	55.71	**57.09**	**43.28**	42.84
LiveBench	45.18	**45.35**	**33.62**	33.54
Instruction following	81.52	**81.70**	**77.82**	77.26
Overall	57.24	**58.10**	49.33	**49.92**

validation-based checkpoint selection, and evaluation protocol follow the main experiments. PPO samples one rollout per prompt ($m = 1$), while POISE samples two ($m = 2$). At the same per-update trajectory budget, PPO can therefore cover twice as many distinct prompts.

The PPO critic is initialized from the actor checkpoint, predicts a value for each generated prefix, and is trained as a full-parameter model using clipped value regression. Following the setup in Open-Reasoner-Zero [Hu et al., 2026] and Deepseek-R1 [Guo et al., 2025], PPO computes advantages using generalized advantage estimation with $\gamma = \lambda = 1$.

Table 11 (right) shows that POISE achieves higher overall performance than token-level PPO on both policy backbones. PPO performs better in some individual domains, but POISE leads on the overall benchmark average. This holds even though PPO's $m = 1$ allocation allows it to cover more distinct prompts per update under the same rollout budget. A possible explanation is that the full-parameter critic must learn prefix-level values from sparse final rewards while tracking a changing policy. Inaccurate or lagging value predictions can increase noise in the resulting advantages. POISE instead fits a small readout to the actor's value-relevant representations and uses an independent rollout to construct each baseline. The separate critic also requires a model-scale set of parameters, together with gradients and optimizer states during training. Keeping these on GPU increases GPU memory requirements; offloading shifts much of the requirement to host RAM and introduces data transfers. POISE does not train an additional model-scale critic.

H Ablations

H.1 Controlled setting for end-to-end ablations

The end-to-end ablations in Section 5 are conducted separately from our main multi-domain experiments. We use Qwen3-4B as the policy model and train it on the English subset of DAPO-Math-17K. This single-domain setting allows us to isolate individual estimator design choices without variation from the domain mixture used in the main experiments.

All variants use the same actor initialization, training data, optimization settings, and total trajectory budget per update (see Table 6). When the number of rollouts per prompt changes, the number of distinct prompts is adjusted accordingly so that the total rollout budget remains fixed. We evaluate each trained policy using Avg@32 on AMC23, AMC24, AIME24, AIME25, AIME26, HMMT25, and BRUMO25, and report the average across the seven benchmarks. For these mathematical-reasoning runs, we use an 8,192-token response limit and a strict verifier requiring exactly one `</think>` delimiter and an answer in `\boxed{}`.

H.2 Ablation of estimator input features

We first isolate the information contributed by each estimator input. We use the held-out value-prediction setting from Section 3.1: the policy and rollout data are held fixed, and predictions are evaluated against empirical Avg@8 values on held-out prompts. Each single-feature variant receives only the indicated feature type.

All three actor-internal signals are predictive of value. Reasoning hidden states provide the strongest individual feature, followed by token entropy, while prompt hidden states remain informative but substantially weaker. Response length performs worse than any of the internal signals, indicating that the estimator is not simply exploiting a correlation between response length and correctness. Combining all three internal signals gives the best MAE and Pearson correlation, supporting their complementary use in the full estimator.

H.3 Ablations of hidden-state extraction

We next vary the layer and pooling window used to extract prompt and reasoning hidden states from Qwen3-4B. Unless otherwise specified, the estimator uses StandardScaler followed by ridge regression, and all non-ablated components are held fixed.

Layer index. We fix the pooling window to the final 10 tokens and sweep the transformer layer used for both prompt and reasoning features.

Table 12: Layer ablation for Qwen3-4B using last-10-token mean pooling.

Layer	MAE ↓	Pearson r ↑
1	0.128	0.807
3	0.126	0.821
5	0.129	0.816
7	0.129	0.820
9	0.131	0.814
11	0.131	0.819
13	0.130	0.823
15	0.128	0.828
17	0.124	0.831
19	0.123	**0.834**
21	0.128	0.830
23	0.130	0.824
25	0.132	0.817
27	0.125	0.827
29	0.125	0.830
31	0.126	0.824
33	**0.120**	0.831
35	0.123	0.827

Performance is strong across a broad range of layers. Layer 19 gives the highest Pearson correlation, while layer 33 gives the lowest MAE. We use layer 19 in the main experiments because it provides the strongest correlation while remaining close to the best MAE.

Pooling window. We fix the extraction layer to 19 and vary the number of final tokens included in mean pooling.

Table 13: Pooling-window ablation for Qwen3-4B at layer 19.

Pooling window	MAE ↓	Pearson r ↑
Last 5	0.127	0.822
Last 10	**0.124**	**0.834**
Last 15	**0.124**	0.833

The last-10 and last-15 variants perform nearly identically, while last-5 pooling is slightly weaker. The estimator is therefore not sensitive to the exact pooling window within this range. We use the final 10 tokens as the default.

H.4 Ablations of probe architecture

We compare the linear ridge estimator with MLP probes of different depths and widths. All variants receive the same prompt hidden states, reasoning hidden states, and entropy features extracted from Qwen3-4B at layer 19 with last-10-token mean pooling. MLPs use ReLU activations, dropout, AdamW, and early stopping on a validation split.

Table 15: End-to-end ablation of estimator inputs on Qwen3-4B. All estimator variants use two rollouts per prompt and cross-rollout baseline estimation. We report Avg@32 (%).

Method	AMC23	AMC24	AIME24	AIME25	AIME26	HMMT25	BRUMO25	Avg.
Base	42.2	31.9	26.3	19.6	24.4	12.9	21.7	25.6
Entropy only	83.5	56.8	44.3	38.4	45.1	22.3	33.9	46.3
Reasoning states only	84.2	**60.0**	47.9	**42.0**	**46.9**	**24.4**	35.5	48.7
Full POISE	**88.7**	59.2	**51.0**	41.7	46.4	**24.4**	**39.3**	**50.1**

Table 14: Probe-architecture ablation for Qwen3-4B. MLP names denote depth $\times$ hidden width.

Probe architecture	MAE $\downarrow$	Pearson r $\uparrow$
Linear ridge	0.124	**0.834**
MLP 1×512	0.154	0.778
MLP 3×512	0.123	0.780
MLP 5×512	0.128	0.765
MLP 7×512	0.124	0.763
MLP 9×512	0.133	0.814
MLP 3×1024	**0.117**	0.801
MLP 5×1024	0.128	0.754

The best MLP slightly improves MAE, but none improves Pearson correlation over the linear ridge estimator. Because the linear probe is also substantially cheaper to refit online and introduces fewer optimization hyperparameters, we use it throughout POISE.

H.5 End-to-end RL ablations

Ablations on algorithm design. We next test the central design choices in end-to-end RL. The aggregate results are reported in Table 3.

First, we test a version of the estimator without any trajectory-level features. This variant uses only prompt hidden states and requires a single rollout per prompt. Because the estimator does not use trajectory-dependent features, its baseline is action-independent without cross-rollout estimation. Comparing this variant with full POISE tests whether the information added by reasoning states and entropy is sufficient to justify sampling a second rollout.

Second, we test a variant of the estimator without cross-rollout design. This variant retains the full feature set but predicts each rollout's baseline from its own internal states. It therefore removes the cross-rollout construction and makes the baseline dependent on the response being optimized. We include this variant only as a diagnostic: it intentionally violates the action-independence condition discussed in Section 2.

Full POISE outperforms both the prompt-only and same-rollout variants. The first comparison shows that trajectory-level features provide sufficient additional information to justify the second rollout. The second shows that these features must be introduced through an independent rollout rather than directly from the trajectory being optimized.

Ablations on estimator components. We further test the trajectory-level estimator inputs in end-to-end RL. The entropy-only, reasoning-state-only, and full variants all use two rollouts per prompt and the same cross-rollout construction. Their rollout allocation and policy-optimization settings are therefore identical, isolating the effect of the input feature set.

As visible in Table 15, reasoning hidden states provide the strongest individual feature set, consistent with the held-out value-prediction results. The full estimator nevertheless achieves the highest aggregate score, showing that prompt states and entropy provide complementary information that improves policy optimization overall.

I Additional experimental evidence

We report supplementary results from the DAPO-Math-17K mathematical-reasoning experiments.

I.1 Additional controlled comparisons

Table 16 compares POISE with the matched RLOO baseline and varies the number of rollouts per prompt. With two rollouts, POISE outperforms RLOO on both backbones. Increasing the rollout count from two to four improves Qwen3-4B by 1.0 percentage point but does not improve DeepSeek-R1-Distill-Qwen-1.5B.

Table 16: Additional controlled mathematical-reasoning results. Scores are Avg@32 (%) averaged over AMC23, AMC24, AIME24, AIME25, AIME26, HMMT25, and BRUMO25. DeepSeek denotes DeepSeek-R1-Distill-Qwen-1.5B. These are individual-run results, not the three-seed means reported below.

Method	Rollouts	Qwen3-4B	DeepSeek
RLOO	2	48.3	30.5
POISE	2	50.1	32.4
POISE	4	51.1	32.3

I.2 Three-seed mathematical-reasoning results

We repeat DAPO and POISE training on Qwen3-4B with three independent training seeds under identical training and evaluation settings. Table 17 reports the mean and standard deviation across these runs. In this setting, POISE obtains a slightly lower average than DAPO (50.0% versus 50.7%).

Table 17: Qwen3-4B mathematical-reasoning results across three independent training seeds. Values are mean $\pm$ one standard deviation of Avg@32 (%).

Benchmark	DAPO	POISE
AMC23	88.3 ± 0.6	88.7 ± 1.1
AMC24	60.4 ± 0.3	59.0 ± 1.1
AIME24	50.1 ± 2.2	48.6 ± 1.5
AIME25	44.5 ± 2.6	43.7 ± 1.3
AIME26	46.3 ± 0.5	45.3 ± 1.6
HMMT25	26.0 ± 1.6	27.7 ± 0.6
BRUMO25	39.6 ± 1.1	36.8 ± 1.7
Average	50.7 ± 0.1	50.0 ± 0.1

I.3 Dynamic sampling and rollout accounting

Dynamic sampling is complementary to POISE, so we apply it to both methods in this section. DAPO discards groups whose rollouts all receive identical rewards and regenerates until the update batch is filled. For POISE, we resample prompts whose clipped prediction yields a near-zero advantage.

Table 18 reports rollout accounting over the first 100 training steps. Each method forms a 512-trajectory policy-update batch per step, but the generation required to fill that batch differs. POISE generates approximately 57% fewer tokens on Qwen3-4B and 43% fewer on DeepSeek-R1-Distill-Qwen-1.5B, because a continuous baseline leaves far fewer trajectories with zero advantage and therefore triggers far less regeneration.

Table 19 translates this accounting into end-to-end cost, reporting the peak Avg@32 averaged over AIME24, AIME25, and AIME26 together with the GPU-hours required to reach it. POISE reaches a comparable peak with substantially less compute on both backbones.

Table 18: Rollout accounting over the first 100 steps of the mathematical-reasoning experiments, with dynamic sampling applied to both methods. Every method uses 512 trajectories per policy-update batch; generated groups and tokens include those produced by resampling. DeepSeek denotes DeepSeek-R1-Distill-Qwen-1.5B.

Backbone	Method	Rollouts/prompt	Generated tokens	Generated groups	Zero-advantage trajectories
DeepSeek	DAPO	8	631,758,484	13,520	46.18%
DeepSeek	POISE	2	360,471,895	30,800	9.02%
Qwen3-4B	DAPO	8	753,076,909	18,176	61.41%
Qwen3-4B	POISE	2	320,214,762	31,690	16.31%

Table 19: Peak mathematical-reasoning accuracy and GPU-hours to the reported peak, with dynamic sampling applied to both methods. Accuracy is Avg@32 (%) averaged over AIME24, AIME25, and AIME26. DeepSeek denotes DeepSeek-R1-Distill-Qwen-1.5B.

Backbone	Method	Peak accuracy	GPU-hours
Qwen3-4B	POISE	46.88	22.20
Qwen3-4B	DAPO	47.40	40.24
DeepSeek	POISE	26.18	18.11
DeepSeek	DAPO	25.94	24.10

I.4 Direct gradient-variance measurement

We additionally measure policy-gradient variance directly from micro-batch gradients in the controlled mathematical-reasoning setting. Following Zeng et al. [2025a], we estimate the variance of the batch-gradient estimator as

$$\widehat{\mathrm{Var}}(g) = \frac{1}{M(M-1)} \left(\sum_{i=1}^{M} \|g_i\|^2 - \frac{1}{M} \left\| \sum_{i=1}^{M} g_i \right\|^2 \right), \tag{19}$$

where g_i is the gradient computed from micro-batch i and M is the number of micro-batches. The second term removes the contribution of the mean gradient, so the estimate reflects variation across micro-batch gradients rather than differences in gradient magnitude.

We run this comparison fully on-policy with DeepSeek-R1-Distill-Qwen-1.5B and a batch size of 512. Because the measured variance scales with the magnitude of the advantages, we disable standard-deviation normalization for DAPO so that RLOO, DAPO, and POISE are compared on a common advantage scale.

Table 20: Direct estimates of batch-gradient variance ($\times 10^{-4}$) during training.

Method	1	5	10	15	20	25	30	35	40
RLOO	8.3	8.7	7.5	6.1	7.9	8.7	9.0	9.1	7.2
DAPO	6.9	7.1	6.8	6.6	7.7	7.1	6.2	6.9	5.8
POISE	9.6	**4.6**	**4.0**	**4.8**	**3.8**	**4.3**	**5.2**	**5.2**	**4.9**

POISE has lower gradient variance than both baselines at every measured checkpoint from step 5 onward. Averaged over steps 5–40, the measured variance is 4.60×10^{-4} for POISE, compared with 8.03×10^{-4} for RLOO and 6.78×10^{-4} for DAPO. Thus, POISE reduces gradient variance by approximately 43% relative to RLOO and 32% relative to DAPO, corresponding to $1.7\times$ and $1.5\times$ lower variance, respectively.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims concern actor-internal value estimation with cross-rollout baselines, improved multi-domain RL performance and training stability, and value prediction across evaluated domains and model scales. These claims are supported by Section 3, Table 1, Figures 4 and 5, and Appendices G and I.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the fixed compute budget, uncharacterized behavior under longer training horizons, and the current sequence-level formulation. It also identifies extensions to non-verifiable rewards and long-horizon agentic tasks as future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 2.2 presents the fixed-budget, fixed-baseline variance decomposition and allocation result, with proofs in Appendices B.1 and B.2. Equation 5 states the zero-expectation baseline condition, and Section 3.2 describes the cross-rollout construction.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4, Algorithm 1, and Appendix D describe the models, training data, rollout allocation, baseline construction, estimator updates, and training and evaluation protocols. Additional experimental details are provided in Appendices C, E, G, H, and I.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The POISE implementation and instructions for environment setup, data preparation, training, and evaluation are publicly available at `https://github.com/holi-lab/POISE`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix D describe the main training and evaluation settings, including estimator hyperparameters, scoring rules, and prompt templates. The controlled mathematical-reasoning protocol and estimator design choices are detailed in Appendices H.1, H.3, and H.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Given the substantial compute cost of RLVR training, we report Avg@32 across 32 independent samples per problem for AIME in Table 1 and the controlled mathematical-reasoning experiments, providing stable Monte Carlo estimates of expected pass rates. We also report means and standard deviations across three independent training seeds in Table 17. However, we do not report seed-level error bars across full multi-domain RL training runs.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix D.4 reports the hardware and compute budgets for the main training experiments, including four H100 GPUs for two days on Qwen3-4B and six days on OLMo-3-7B-Instruct-DPO. Appendix I.3 further reports GPU-hours and rollout costs for the additional mathematical-reasoning experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics. The work uses existing public benchmarks and open-weight models and does not involve human participants or the collection of personal data.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: POISE is a foundational method for RL optimization rather than a study of a specific deployment. We do not identify direct societal impacts specific to the proposed baseline estimator beyond the general risks associated with training and deploying language models.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work introduces an RL algorithm and associated code rather than a new pretrained foundation model or a newly collected dataset. It uses existing model checkpoints and public benchmarks and does not introduce a separate high-risk data or model release.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The models, datasets, and baseline implementations are identified in Section 4 and Appendices D and G. However, we do not provide a complete asset-by-asset list of license names and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released POISE implementation is accompanied by documentation covering environment setup, installation, data preparation, configuration, training, and evaluation. The repository also includes an Apache-2.0 license and upstream attribution notices.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects; all experiments are run on publicly available models and benchmarks.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so no IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Sections 3 and 4 describe the policy LLMs and how their internal states are used for baseline estimation. Appendix G describes the LLM-based critic baselines.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.