PVPO: Pre-Estimated Value-Based Policy Optimization for Agentic Reasoning

Wenfeng Feng[1], **Penghong Zhao**[1], **Guochao Jiang, Chuzhan Hao, Guohua Liu, Yuewei Zhang**[*]
Alibaba Cloud Computing
{wenfeng.fwf, zhaopenghong.zph, anyue.jgc, haochuzhan.hcz, liyou.zyw}@alibaba-inc.com

Abstract

Grouping-based methods have emerged as a significant frontier in Reinforcement Learning (RL), yet agentic reasoning poses a fundamental challenge for grouping-based methods: frequent environmental interactions and multi-step tool invocation generate highly variable trajectories, rendering intra-group advantage estimation unstable. In response, practitioners resort to excessive rollouts to stabilize training, which in turn incurs prohibitive computational costs. This negative feedback loop between advantage estimation instability and sampling inefficiency severely limits learning performance. We present PVPO, a stable and efficient RL framework that breaks this cycle through a pre-estimated value baseline and pre-sampled data filtering. Specifically, before training begins, PVPO performs a single round of rollouts to compute two signals: (1) Static V, a Monte Carlo estimate of the expected return that serves as a fixed baseline to stabilize advantage estimation; and (2) sample-level accuracy, as a difficulty metric to filter out trivial samples and inject ground-truth trajectories into hard ones, thereby enhancing training efficiency. As shown in Figure 1, experiments demonstrate that PVPO outperforms other grouping-based methods in both multi-step retrieval tasks and advanced mathematical reasoning benchmarks. Notably, our 7B model trained with PVPO matches or exceeds the performance of large language models (LLMs) on most multi-step retrieval benchmarks. Moreover, PVPO achieves a 2.5x speedup in training time compared to prior methods while maintaining comparable final performance.

1 Introduction

RL has demonstrated remarkable success in fine-tuning LLMs for complex decision-making tasks.

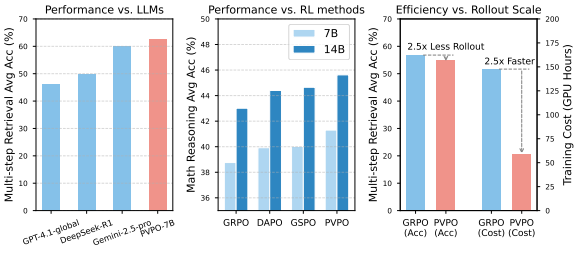


Figure 1: Comparison of performance and training efficiency between diverse baselines (including LLMs and RL methods) and our PVPO. **Left**: PVPO-7B matches or exceeds the performance of LLMs in multi-step retrieval tasks. **Middle**: PVPO consistently achieves superior accuracy across different model scales compared to existing RL methods. **Right**: By reducing rollout scales, PVPO significantly accelerates training while preserving comparable accuracy.

Within this paradigm, grouping-based methods have emerged as a particularly valuable research direction (Shao et al., 2024; Feng et al., 2025a). However, extending these methods to agentic reasoning presents unique challenges. Unlike standard generation, agentic reasoning involves frequent environment interactions and multi-step tool invocations. For the same input, rollouts often diverge substantially in reasoning paths, actions, and tool responses. Consequently, advantage estimates derived from intra-group comparisons (intra-group advantage estimates) cannot reliably reflect action quality or provide a meaningful value baseline, thereby exacerbating advantage estimates instability.

To mitigate this, some studies group by sample, running multiple trajectories within each group to compute relative advantage (Zuo et al., 2025; Lyu et al., 2025). Others group by action or timestep, enabling finer partitioning (Feng et al., 2025a; Li et al., 2025b). These methods can improve the stability of advantage estimation by leveraging structured groupings. However, such grouping-based

[1]Equal contribution.
[†]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 3729–3748
July 2-7, 2026

methods usually require several-fold more rollouts, which greatly reduces training efficiency. Methods like DAPO (Yu et al., 2025) reduce rollouts by sampling high-value data; however, this often primarily shifts computational overhead from rollouts to sampling rather than achieving genuine efficiency gains. Consequently, existing methods struggle to balance stability and throughput, particularly for specialized small-scale models required in agentic reasoning (Belcak et al., 2025; Sharma and Mehta, 2025). This raises a critical question: how can we achieve stable and efficient training in agentic reasoning tasks, particularly when scaling down to small models under constrained computational resources?

In this paper, we introduce Pre-estimated Value-based Policy Optimization (PVPO), a generalized RL method based on Proximal Policy Optimization (PPO) (Schulman et al., 2017). PVPO achieves stable and efficient training through two key components: Static V Estimate and Group Sampling, both of which are computed in a single round of rollouts before training begins. We pre-estimate a fixed value baseline called Static V. Static V is computed via Monte Carlo estimate of the expected return, independently of online policy updates, to enhance advantage estimation stability. Group Sampling pre-filters the training data based on sample-level accuracy. Trivial samples with perfect accuracy are removed, while ground-truth trajectories are optionally injected into hard samples to provide explicit demonstrations of successful reasoning. This reduces the volume of training data while ensuring informative gradient signals, thereby improving training efficiency. In summary, our core contributions are as follows.

- We propose PVPO, a novel RL framework that addresses the advantage estimation instability inherent in grouping-based methods through Static V Estimate.
- We employ a single round of rollouts for Group Sampling, thereby enabling efficient policy optimization even for small-scale models under constrained computational resources.
- PVPO achieves state-of-the-art (SOTA) performance on multi-step retrieval and demonstrates robust generalization across advanced mathematical benchmarks, significantly enhancing models' complex retrieval and reasoning capabilities.

2 Related Work

2.1 Agentic Reasoning

Agentic reasoning has been shown to significantly improve the performance of agents. In contrast to conventional reasoning, which responds to inputs directly with generative responses and performs general-purpose tasks, agentic reasoning enhances a model's problem understanding by enabling it to initiate, plan, execute, and adapt its actions based on environmental feedback, thereby improving its performance on domain-specific tasks (Jin et al., 2025; Jiang et al., 2025; Li et al., 2025c). To further boost interactive capabilities, recent works (Ouyang et al., 2025; Zhang et al., 2025) leverage experience distillation and memory augmentation. However, as task complexity increases, reasoning trajectories grow longer, making it increasingly difficult to effectively decompose problems and accurately solve individual subproblems within extended reasoning chains. To address this challenge, planning approaches have been proposed to structure the reasoning process. Notable examples include Reflexion (Shinn et al., 2023), AirRAG (Feng et al., 2025b), and RLoT (Hao et al., 2025b), which integrate reflection into multi-step planning.

2.2 Retrieval and Tool Invocation in Agentic Reasoning

Beyond optimizing reasoning structures, learning to invoke external tools has been shown to significantly enhance an LLM's ability to solve complex problems step-by-step. By leveraging objective tools to resolve subproblems accurately, agents can improve both reasoning efficiency and correctness. For instance, DeepResearcher (Zheng et al., 2025b) trains agents to navigate the open web for deep research tasks, while Search-R1 (Jin et al., 2025) and ReSearch (Chen et al., 2025) enable models to interleave reasoning with Wikipedia-based retrieval. Similarly, R1-Searcher and its variants (Song et al., 2025a,b) employ a two-stage RL framework to incentivize autonomous search invocation. More recent work in web retrieval (Li et al., 2025a; Wu et al., 2025) further demonstrates that agents can master long-horizon web browsing by RL. Despite these advances, training remains unstable due to the difficulty of reliably evaluating the relative quality of diverse decision paths. As the number of tool

interactions increases, trajectories diverge significantly. Moreover, effective actions are sparse and unevenly distributed across timesteps, making it challenging to assign credit accurately. Even with fine-grained rewards, the need for a large number of rollouts to obtain stable advantage estimates reduces sample efficiency, particularly for smaller models with limited reasoning capacity. This instability stems from the critical dependence of downstream outcomes on tool responses and the non-stationary nature of tool-augmented environments.

2.3 Off-policy Method

To address these challenges, incorporating off-policy components into reinforcement learning has proven effective in stabilizing training (Chen et al., 2024). In particular, augmenting grouping policy optimization with offline policy has yielded significant improvements. LUFFY (Yan et al., 2025) and SRL (Deng et al., 2025) uses expert or teacher trajectories as off policy guidance and incur multi-stage online overhead; and ROLL Flash (Wang et al., 2025; Lu et al., 2025a) enables asynchronous training that naturally supports off-policy updates; the replay mechanism stabilizes training by reusing the agent's own high-quality past rollouts (Li et al., 2025b; Zhan et al., 2025; Lu et al., 2025b). In this context, we perform a single round of rollouts to identify the hardest samples, injecting ground truth trajectories only where needed. This one-time design avoids both the recurring per-epoch rollout cost and the dense expert supervision.

In contrast to prior work focusing on reasoning architectures or tool-invocation methods, our method PVPO decouples the two components of advantage estimation: the trajectory return and the value baseline. Specifically, returns are computed from intra-group rollout rewards under the current policy, while value baseline is estimated offline from historical checkpoints. This design preserves the responsiveness of grouping policy while stabilizing advantage estimates through a sparsely updated value baseline.

3 Preliminary

In this section, we review the fundamental concepts of policy optimization in RL, with a particular focus on the role of the advantage function and its various estimation methods.

3.1 Proximal Policy Optimization

In actor-critic frameworks, a critic network predicts state-value function (V), which combines with action-value function (Q) to compute the advantage and then guides policy updates. Classic methods, such as PPO, train a critic network $V_\phi(s)$ to provide a low-variance estimate of the state-value function $V^\pi(s)$ of state s. The state-value function is used to compute the advantage at each time step t, typically via Generalized Advantage Estimation (GAE) (Schulman et al., 2016):

$$\begin{aligned} \hat{A}_t^{\text{GAE}} &= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \\ \delta_t &= r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \end{aligned} \tag{1}$$

where λ is a hyper-parameter, δ_t is the temporal difference error at time step t, r_t is the immediate reward received at time step t, γ is the discount factor. PPO then optimizes a clipped surrogate objective to update the actor network in a stable manner:

$$\begin{aligned} &\mathcal{J}^{\text{PPO}}(\theta) = \mathbb{E}_{q\sim P(D), o\sim\pi_{\theta_{\text{old}}}(O|q)} \\ &\left[\min\left(r_t(\theta)\hat{A}_t^{\text{GAE}}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\text{GAE}}\right)\right], \end{aligned} \tag{2}$$

where q are questions sampled from the dataset D, o are outputs sampled from the old policy π_{old}, importance sampling ratio $r_t(\theta) = \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{<t})}$, ϵ is the clipping range of $r_t(\theta)$.

3.2 Group Relative Policy Optimization

Since the critic network is typically as large as the actor network, it adds substantial memory and computational burden. Critic-free methods, such as GRPO, eliminate this costly component by estimating the advantage directly from rewards. For each question, GRPO generates a group of outputs $\{o_i\}$ from the old policy $\pi_{\theta_{\text{old}}}$. The advantage for each output o_i is then calculated based on normalized reward $\mathbf{r}$ relative to the group:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}. \tag{3}$$

This grouping-based advantage estimate is then used to optimize a PPO-like objective function:

$$\begin{aligned} &\mathcal{J}^{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim P(D),\{o_i\}\sim\pi_{\theta_{\text{old}}}(O|q)} \left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\right. \\ &\left\{\min\left(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right)\right. \\ &\left.\left. - \beta D_{KL}[\pi_\theta||\pi_{\text{ref}}]\right\}\right], \end{aligned} \tag{4}$$

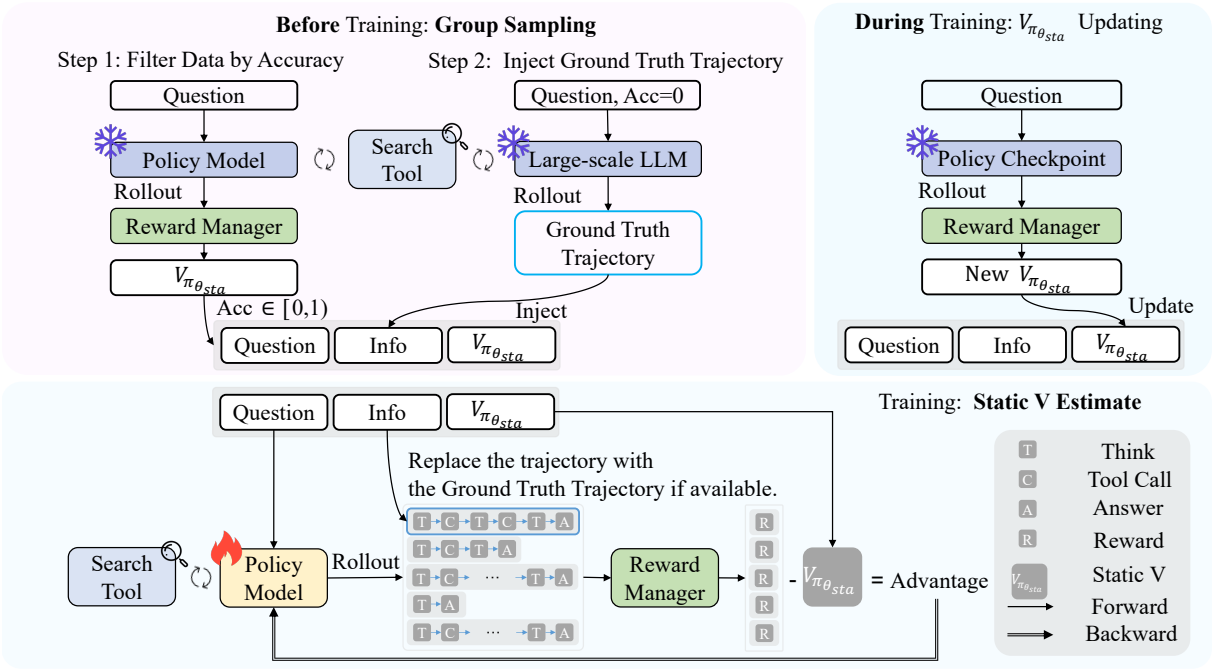


Figure 2: The framework of PVPO. **Before training**, the policy model is used to perform a single round of rollouts. We compute the sample-level accuracy to filter out samples with accuracy = 1. For samples with accuracy $\in [0, 1)$, we also compute the Monte Carlo estimate of the expected return based on trajectory rewards and store it in the dataset as $V_{\pi_{\theta_{\text{sta}}}}$. Optionally, an LLM is used to perform rollouts to generate ground truth trajectories for accuracy = 0 samples, which are injected into the dataset. **During training**, $V_{\pi_{\theta_{\text{sta}}}}$ is updated at a fixed step using rollouts from the checkpoint of policy model, aligning the baseline with the current policy's empirical return distribution. **In the training process**, PVPO exclusively employs the policy model for sampling and parameter updates. For samples with all incorrect rollouts, a ground truth trajectory (if available) replaces one existing rollout. During advantage estimation, $V_{\pi_{\theta_{\text{sta}}}}$ serves as a static value baseline. Reward manager do not restrict the generation of reward.

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$, D_{KL} is the KL divergence between the trained policy π_θ and the reference policy π_{ref}, N denotes the rollout size during training.

4 Methodology

Agentic reasoning poses two challenges for grouping-based RL training: trajectory divergence destabilizes intra-group advantage estimation, and sparse rewards necessitate a large number of rollouts to obtain sufficient training signal. PVPO addresses these through two algorithmic components: (1) Static V Estimate, decoupling the value baseline from online policy updates to stabilize advantage estimation under divergent trajectory, and (2) Group Sampling, maximizing the training utility of each rollout under sparse reward conditions for efficient policy optimization.

The overall framework of PVPO is illustrated in Figure 2. Specifically, PVPO optimizes the policy by maximizing the following objective function:

$$\mathcal{J}^{\text{PVPO}}(\theta) = \mathbb{E}_{q\sim P(D),\{o_i\}\sim\pi_{\theta_{\text{old}}}(O|q)}\left[\frac{1}{\sum_{i=1}^{N}|o_i|}\sum_{i=1}^{N}\sum_{t=1}^{|o_i|}\min\left(r_{i,t}(\theta)\hat{A}_{i,t}^{\text{PVPO}}, \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}^{\text{PVPO}}\right)\right], \tag{5}$$

$$r_{i,t}(\theta) = \begin{cases} \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, & \text{if } o_i \notin \text{GT}. \\ \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{gt}}}(o_{i,t}|q,o_{i,<t})}, & \text{if } o_i \in \text{GT}. \end{cases} \tag{6}$$

where $\pi_{\theta_{\text{gt}}}$ is the off-policy LLM that generates ground truth trajectories (GT).

4.1 Static V Estimate

In policy gradient methods with trajectory-level rewards, where a scalar reward $R(\tau)$ is observed only upon completion of a full trajectory $\tau = (s_0, a_0, ..., s_{T-1}, a_{T-1})$, and all intermediate environmental rewards satisfy $r_t = 0$ for $t < T-1$, and $r_{T-1} = R(\tau)$, the return from any time step t

is simply the discounted final reward:

$$\hat{R}_t = \sum_{l=t}^{T-1} \gamma^{l-t} r_l = \gamma^{T-t-1} R(\tau). \tag{7}$$

This is a Monte Carlo return, as it relies solely on the observed outcome of a complete rollout without intermediate value predictions, in which $\lambda = 1$. Consequently, the natural and unbiased advantage estimator is the Monte Carlo advantage:

$$\hat{A}_t = \gamma^{T-1-t} R(\tau) - V(s_t). \tag{8}$$

The full derivation is provided in Appendix A.6.1. In PVPO, we further enhance stability by replacing the dynamic state-value function $V(s_t)$ with a static state-value function $V_{\pi_{\theta_{\text{sta}}}}(s_t)$. $\hat{V}_{\pi_{\theta_{\text{sta}}}}$ is a Monte Carlo estimate of the expected return computed via a single round of rollouts, abbreviated as Static V. And $\pi_{\theta_{\text{sta}}}$ is instantiated from a checkpoint of the policy π_θ at a fixed training step including the initial untrained policy model at step 0. Combining the Equation 8, we arrive at the PVPO advantage estimator:

$$\hat{A}_t^{\text{PVPO}} = \gamma^{T-1-t} R(\tau) - \hat{V}_{\pi_{\theta_{\text{sta}}}}(s_t). \tag{9}$$

See Appendix A.6.2 for the detailed derivation. Equation (9) assumes access to fine-grained state-value estimates $\hat{V}_{\pi_{\theta_{\text{sta}}}}(s_t)$. However, in agentic reasoning tasks with sparse rewards, the environment provides only a scalar outcome $R(\tau)$ upon trajectory completion, with no intermediate per-step feedback. This makes computing state-conditional values infeasible, which is typical in multi-step retrieval and tool invocation. We therefore adopt a trajectory-level approximation that replaces $\hat{V}_{\pi_{\theta_{\text{sta}}}}(s_t)$ with the trajectory-level expected return $\mathbb{E}_{\tau\sim\pi_{\theta_{\text{sta}}}}[R(\tau)]$, under the static policy.

$$\begin{aligned} \hat{A}^{\text{PVPO}}(\tau_i) &= R(\tau_i) - \mathbb{E}_{\tau\sim\pi_{\theta_{\text{sta}}}}[R(\tau)] \\ &\approx r_i - \frac{1}{M}\sum_{j=1}^{M} r_{\pi_{\theta_{\text{sta}}},j}, \end{aligned} \tag{10}$$

where $r_i = R(\tau_i)$ is the trajectory reward of the i-th on-policy rollout, $r_{\pi_{\theta_{\text{sta}}},j}$ denotes the trajectory reward of the j-th rollout sampled from $\pi_{\theta_{\text{sta}}}$ and M denotes the rollout size before training used to compute Static V.

In summary, our advantage function follows the original definition without further normalization, where r_i is obtained from the immediate reward of on-policy π_θ. It reflects the current performance of the policy and is highly adaptive. $\pi_{\theta_{\text{sta}}}$ is sparsely updated during training to serve as a conservative performance lower bound, as shown in During Training part of Figure 2. This formulation provides a stable advantage estimate from the final reward.

4.2 Group Sampling

In the same round of rollouts used to compute Static V, the frozen policy model is also utilized to compute the sample-level accuracy of each trajectory. For every input sample, we aggregate these trajectory accuracies into a mean accuracy score, which serves as a measure of task difficulty under the initial policy. Specifically, samples are categorized into three groups:

- Group 1: Samples with a mean accuracy of 1 are excluded from the training set, as they are considered too trivial to facilitate effective learning.
- Group 2: Samples with a mean accuracy strictly between 0 and 1 are retained, given their nonzero advantage.
- Group 3: For samples with zero mean accuracy, successful trajectories are optionally generated by querying a large-scale LLM when such a model is available.

Group Sampling enhances the training efficiency through two complementary mechanisms. First, filtering out solved samples (Group 1) reduces the training volume, directly lowering computational overhead and shortening training time. Second, for the most challenging cases (Group 3), the inference-generated successful trajectories are stored in the sample's metadata and injected during training by replacing one of the policy-generated rollouts in the group. By providing an explicit successful reasoning path, this step bootstraps the optimization process and mitigates training stagnation for otherwise unrecoverable failures. As a curriculum enhancement targeting the most challenging training samples, it is an optional rather than mandatory component of PVPO, and Section 6.4 provides evidence that the core algorithmic gains from Static V Estimate are independent of this injection. Together, Group Sampling improve both computational efficiency by data reduction and sample efficiency by guided exploration on difficult cases.

Table 1: Datasets used for training and evaluation across two tasks. †(AI-MO, 2024; Bytedance and Tsinghua-SIA, 2025). ‡(HuggingFaceH4, 2023; Lightman et al., 2024).

Task	Dataset	Split
Multi-step Retrieval	Musique (Trivedi et al., 2022)	Train / Dev / Test
	2WikiMultiHopQA (Ho et al., 2020)	Dev / Test
	HotpotQA (Yang et al., 2018)	Dev / Test
	Bamboogle (Press et al., 2023)	Dev / Test
Mathematical Reasoning	DAPO-Math-17k (Yu et al., 2025)	Train
	DAPO-AIME-2024†	Test
	AIME-2025 (Lin, 2025)	Test
	MATH500‡	Test
	AMC23 (AI-MO, 2024)	Test
	Olympiad (He et al., 2024)	Test

Table 2: Hyperparameter settings for multi-step retrieval and mathematical reasoning tasks.

Hyperparameter	Retrieval	Math
Training		
Temperature	1.0	
Top-p	1.0	
Learning rate	1e-6	
Max response length	8192	
Static V update step	500	
M	5	16
N	5	16
Batch size	8	32
Training steps	1,439	1,000
Inference		
Temperature	0.6	
Top-p	0.95	

5 Expriments Setting

Metrics. For multi-step retrieval tasks, we employ answer accuracy (Acc, %) and LLM-as-a-Judge (LasJ, %) (Song et al., 2025b) as evaluation metrics. For mathematical reasoning tasks, we measure answer accuracy (Acc, %), reporting the mean accuracy across 32 independent rollouts for each sample (i.e., acc@32).

Datasets. For multi-step retrieval tasks, we train on the Musique training split (20k examples) and evaluate on the full development and test sets of four benchmarks. For mathematical reasoning tasks, we train on DAPO-Math-17k-Processed (17k examples) and evaluate on five benchmarks. Dataset details are summarized in Table 1.

Baselines and Training Details. Table 2 presents the key hyperparameters for training and inference across the two tasks. For the multi-step retrieval tasks, We adopt the ReSearch, ReCALL frameworks (Chen et al., 2025) and DynaSearcher (Hao et al., 2025a) frameworks. For mathematical reasoning tasks, we primarily adopt GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), and GSPO (Zheng et al., 2025a) as baselines. We use the verl (Sheng et al., 2025) framework. We use *Qwen2.5-7B-Instruct* and *Qwen2.5-14B-Instruct* as base models and *Qwen2.5-72B-Instruct* as the large LLM to generate GT. All experiments are conducted on a server equipped with an Intel(R) Xeon(R) Platinum 8369B CPU and 8×NVIDIA A100-SXM4-80GB GPUs. More details can be found in Appendix A.1.

6 Experiments

We first validate its effectiveness in the domain of Agentic Reasoning. We then analyze its efficiency and stability. We further demonstrate its generalization capabilities through cross-task, multi-scale models and compatibility with other RL methods. Through rigorous ablation studies, we examine the contributions of each module and configuration detail. Finally, case studies demonstrate its performance on small models under constrained computational resources.

6.1 Main Results

We evaluate PVPO against both zero-shot leading LLMs (*DeepSeek-R1-0528*, *GPT-4.1-0414*, *o4-mini-0416*, and *Gemini-2.5-pro-0325*) and trained RL-based search methods(Search-R1 (Jin et al.,

Table 3: Performance comparisons between PVPO and the baselines on multi-step retrieval datasets. The best and second best results are **bold** and underlined, respectively.

Method	Musique		2Wiki		HotpotQA		Bamboogle		Average	
	Acc	LasJ	Acc	LasJ	Acc	LasJ	Acc	LasJ	Acc	LasJ
Prompt Based										
Qwen2.5-7B-Instruct	5.1	13.5	27.9	29.3	22.4	31.0	12.8	17.1	17.1	22.7
DeepSeek-R1-0528	32.0	40.7	57.5	59.4	43.0	58.3	66.4	76.6	49.7	58.8
o4-mini-0416	38.0	44.1	61.5	67.4	49.5	67.4	74.4	84.2	55.9	65.8
GPT-4.1-global-0414	31.0	40.9	58.0	58.5	44.5	57.7	51.2	61.6	46.2	54.7
Gemini-2.5-pro-0325	42.5	50.8	70.0	71.2	53.0	71.1	**75.2**	**84.5**	60.2	69.4
Training Based										
Qwen2.5-7B-Instruct										
Search-R1-v0.3	24.7	34.6	58.7	61.1	53.6	66.9	48.0	54.5	46.3	54.4
R1-Searcher	24.7	34.2	67.8	68.2	59.7	71.5	46.4	52.0	50.5	56.5
GRPO-ReSearch	33.4	46.7	60.8	67.0	54.5	63.7	45.6	54.4	48.6	58.0
GRPO-DynaSearcher	38.9	52.0	74.3	76.8	62.7	68.3	51.2	58.7	56.8	64.0
GRPO-ReCALL	33.3	45.4	64.9	67.8	57.1	67.4	46.4	54.8	50.4	58.9
PVPO-ReSearch	36.5	51.4	70.1	72.4	65.5	72.3	45.6	54.3	54.4	62.6
PVPO-DynaSearcher	**46.9**	**59.4**	**77.7**	**80.6**	69.0	78.4	50.4	59.7	61.0	69.6
PVPO-ReCALL	45.1	54.6	77.2	79.8	**72.4**	**80.7**	55.2	65.8	**62.5**	**70.2**

2025), R1-Searcher (Song et al., 2025b), ReSearch, ReCALL frameworks (Chen et al., 2025) and DynaSearcher), with results in Table 3 underscoring its effectiveness. Specifically, applying PVPO substantially improves the base frameworks, boosting ReSearch's Avg Acc/LasJ scores by 5.8/4.6 points, DynaSearcher's by 4.2/5.6 points and ReCALL's Avg Acc/LasJ scores by 12.1/11.3. PVPO-ReCALL significantly surpasses all RL baselines and even outperforms leading proprietary LLMs. On the Bamboogle dataset, LLMs significantly outperform 7B-trained models largely due to the outdated 2018 Wikipedia corpus used in our experiments (see Appendix A.1 and Figure 6). Overall, these results demonstrate that PVPO consistently achieves state-of-the-art performance across agentic reasoning methods.

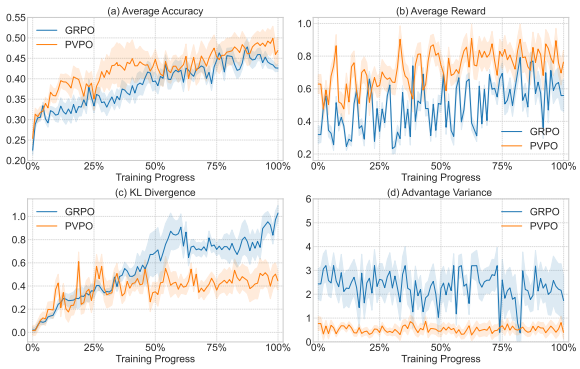


Figure 3: Training efficiency and stability analysis of PVPO on multi-step retrieval datasets. Solid lines represent the mean of 5 independent runs, while shaded areas denote the standard deviation.

6.2 Efficiency and Stability Evaluation

Figure 3(a, b) demonstrates that PVPO accelerates the learning process and achieves superior final performance. These results empirically validate Group Sampling design, which enhances computational efficiency through data reduction (yielding a 1.7–2.5× speedup; see Appendix A.2) and improves sample efficiency via successful reasoning path on difficult cases. Furthermore, stability metrics in Figure 3(c, d) reveal that PVPO maintains controlled KL divergence and significantly lower advantage variance. This confirms that the Static V Estimate provides stable advantage estimation, facilitating policy optimization in complex agentic reasoning tasks. A systematic analysis of the resulting reasoning behavior is provided in Appendix A.3.

6.3 Generalization Evaluation

To demonstrate the generalization capabilities of PVPO, we evaluate it on mathematical reasoning tasks of varying difficulty levels. As shown in Table 4, PVPO consistently outperforms all baselines

Table 4: Performance comparison of PVPO and baseline methods on mathematical reasoning datasets using different model scales. "w/" means trained with. Best results are shown in **bold** and second-best results are underlined.

Method	MATH500	AMC23	Olympiad	AIME-2024	AIME-2025	Avg Acc
Qwen2.5-7B-Instruct	75.68	42.92	38.94	12.10	6.67	35.26
w/ GRPO	78.60	49.10	42.14	13.86	10.10	38.76
w/ DAPO	78.58	51.38	43.36	14.96	11.30	39.92
w/ GSPO	78.66	50.12	43.60	**15.02**	12.70	40.02
w/ PVPO	**80.30**	**52.02**	**44.62**	14.86	**14.70**	**41.30**
Qwen2.5-14B-Instruct	79.68	51.52	44.00	14.82	12.29	40.46
w/ GRPO	82.12	53.50	47.42	16.14	15.86	43.01
w/ DAPO	82.50	56.44	49.34	18.04	15.66	44.40
w/ GSPO	83.56	56.02	49.28	18.18	16.20	44.65
w/ PVPO	**83.64**	**56.78**	**50.72**	**19.24**	**17.74**	**45.62**

Table 5: Ablation comparison of PVPO and GRPO methods on multi-step retrieval datasets. "w/" means trained with and "w/o" means not trained with. "GTIR" means Ground Truth trajectory Injection and Replacement, "VU" means $V_{\pi_{\theta_{\text{sta}}}}$ Updating. Starting from ReCALL on Qwen2.5-7B-Instruct, we first establish the Static V Estimate as a baseline, then incrementally augment the model with GTIR and VU. Best results are shown in **bold** and second-best results are underlined.

Method	Musique	2Wiki	HotpotQA	Bamboogle	Avg Acc
GRPO-ReCALL					
Default	33.3	64.9	57.1	46.4	50.4
w/ GTIR	33.4	66.0	59.1	46.4	51.2
PVPO-ReCALL					
w/o both	33.0	74.5	65.9	44.8	54.6
w/ GTIR	38.0	71.8	64.1	**55.5**	57.4
w/ VU	43.4	77.4	70.8	54.4	61.5
w/ both	**45.1**	**77.2**	**72.4**	55.2	**62.5**

at both the 7B and 14B scales. Furthermore, PVPO exhibits high compatibility with other RL methods, yielding additional performance gains. Since these integrations are not the primary focus of this work, we provide details in Appendix A.4.

6.4 Ablation Study

We conduct an ablation study to isolate the contribution of each component in PVPO, as shown in Table 5. To ensure a fair comparison, both GRPO-ReCALL (Default) and PVPO-ReCALL (w/o both) are trained on the same filtered dataset, comprising 11,512 samples (8,379 Group 2 + 3,133 Group 3). The only difference is the advantage estimation mechanism: GRPO uses the intra-batch group mean (Eq. 3), while PVPO uses the pre-estimated Static V (Eq. 10).

Starting from the PVPO-ReCALL only with Static V Estimate (54.6 Avg Acc), the integration of ground truth trajectory injection and replacement / static V updating first raises the scores to 57.4 / 61.5, respectively. Subsequently adding both components further boosts the performance to 62.5, which represents our full PVPO model and outperforms all baselines. This incremental improvement validates the effectiveness of each proposed component. For fairness, we also apply ground truth trajectory injection and replacement to GRPO to mitigate ground truth trajectory influence across different methods. The results show that the ground truth trajectory enhances GRPO's performance by 2.8, comparable to its effect on PVPO. Furthermore, more detailed ablation experiments can be found in Appendix A.5, including model scales, updating steps and variations in ground truth trajectory counts.

Table 6: Time-consuming comparison of PVPO and GRPO on training set of multi-step retrieval datasets. "DF" means Data Filtering in Group Sampling. "GTIR" means Ground Truth trajectory Injection and Replacement, "VU" means $V_{\pi_{\theta_{\text{sta}}}}$ Updating. Performance metrics and time statistics are the averages of five experiments. Best results are shown in **bold** and second-best results are underlined.

	Avg Acc	Wall-Clock/h	GPU×Hour
GRPO (N=5)			
Train	56.8	36.91	147.65
PVPO (N=5)			
DF	-	2.06	16.44
GTIR	-	3.09	12.36
VU	-	2.40	19.23
Train	62.5	22.87	91.46
Total	**62.5**	30.42	139.49
PVPO (N=2)			
DF	-	0.77	6.18
GTIR	-	2.00	8.00
VU	-	0.98	7.83
Train	55.0	9.33	37.32
Total	55.0	**13.08**	**59.33**

6.5 Case Study: Low Sampling Budget

The aforementioned experiments have demonstrated PVPO's outstanding performance on small models. In this section, we quantify the total computational time to prove PVPO's contribution under constrained computational resources. For comparison, we report GRPO's performance with a full budget. As shown in Table 6, even when performing model rollouts both before and during training, PVPO consumes 8.16 GPU×hour less computational resources than GRPO under 5 rollouts (used in the main experiments). To further examine PVPO's performance under constrained computational resources, we reduce the number of rollouts from 5 to 2. PVPO achieves 97% of GRPO's performance (55.0% vs 56.8%) while using less than 40% (59.33/147.65) of total time consumption.

7 Conclusions

In this paper, we propose PVPO, a critic-free RL method designed to achieve stable and efficient training in agentic reasoning tasks. By introducing Static V Estimate and Group Sampling, PVPO addresses training instability caused by intra-group advantage estimates and inefficiencies caused by oversampling, particularly on small models under constrained computational resources. Extensive experiments across nine diverse benchmarks in multi-step retrieval and mathematical reasoning demonstrate that PVPO achieves SOTA performance and strong generalization. PVPO introduces substantial improvements in tool invocation and reasoning, supports scalable training, and ensures consistent performance, thereby demonstrating strong potential for widespread real-world application.

Limitations

In this work, our Static V Estimate introduces some rollout overhead for value baseline updates compared to purely online approaches, yet this presents opportunities for further research into adaptive re-estimation during long-horizon training. Also, our approach focuses on optimizing small models by leveraging successful trajectories from larger counterparts. Investigating how to autonomously discover successful reasoning paths for the largest SOTA models remains an intriguing avenue for future research.

References

AI-MO. 2024. Aimo-validation-aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AI-MO. 2024. AIMO Validation AMC Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *arXiv preprint arXiv:2506.02153*.

Bytedance and Tsinghua-SIA. 2025. AIME-2024. https://huggingface.co/datasets/BytedTsinghua-SIA/AIME-2024. Hugging Face Dataset.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yihe Deng, I Hsu, Jun Yan, Zifeng Wang, Rujun Han, Gufeng Zhang, Yanfei Chen, Wei Wang, Tomas

Pfister, Chen-Yu Lee, and 1 others. 2025. Supervised reinforcement learning: From expert trajectories to step-wise reasoning. *arXiv preprint arXiv:2510.25992*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025a. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Guochao Jiang, Jingyi Song, and Hao Wang. 2025b. Airrag: Autonomous strategic planning and reasoning steer retrieval augmented generation. *Preprint*.

Chuzhan Hao, Wenfeng Feng, Yuewei Zhang, and Hao Wang. 2025a. Dynasearcher: Dynamic knowledge graph augmented search agent via multi-reward reinforcement learning. *arXiv preprint arXiv:2507.17365*.

Qianyue Hao, Sibo Li, Jian Yuan, and Yong Li. 2025b. Rl of thoughts: Navigating llm reasoning with inference-time reinforcement learning. *arXiv preprint arXiv:2505.14140*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

HuggingFaceH4. 2023. Math-500. https://huggingface.co/datasets/HuggingFaceH4/MATH-500.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, and 1 others. 2025a. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. 2025b. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025c. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yenting Lin. 2025. Aime 2025 dataset. https://huggingface.co/datasets/yentinglin/aime_2025.

Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yanan Wu, Weixun Wang, Jiashun Liu, Yang Li, Haizhou Zhao, and 1 others. 2025a. Part ii: Roll flash–accelerating rlvr and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*.

Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Haotian Xu, Jiaqi Guo, Chutian Wang, Haonan Chen, Xiaoxi Jiang, and Guanjun Jiang. 2025b. Search self-play: Pushing the frontier of agent capability without supervision. *arXiv preprint arXiv:2510.18821*.

Shangke Lyu, Linjuan Wu, Yuchen Yan, Xingyu Wu, Hao Li, Yongliang Shen, Peisheng Jiang, Weiming Lu, Jun Xiao, and Yueting Zhuang. 2025. Hierarchical budget policy optimization for adaptive reasoning. *arXiv preprint arXiv:2507.15844*.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Raghav Sharma and Manan Mehta. 2025. Small language models for agentic systems: A survey of architectures, capabilities, and deployment trade offs. *arXiv preprint arXiv:2510.03847*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025a. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025b. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *arXiv preprint arXiv:2505.17005*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, and 1 others. 2025. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Gang Fu, Yong Jiang, and 1 others. 2025. Webdancer: Towards autonomous information seeking agency. *arXiv preprint arXiv:2505.22648*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. 2025. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025a. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025b. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, and 1 others. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

A Appendix

A.1 Implementation Details

Hyperparameters and Framework Details. For multi-step retrieval tasks, rather than training on the full Musique dataset (19,938 samples), PVPO utilizes Group Sampling to reduce the training volume. The final training set comprises 11,512 instances, consisting of 8,379 samples from Group 2 (0<acc<1) and 3,133 from Group 3 (acc=0, with GT). Given a batch size of 8, this configuration corresponds to 1,439 training steps, significantly reducing the total computational overhead compared to full-dataset training. For DynaSearcher, we remove the "kg_filter" during inference.

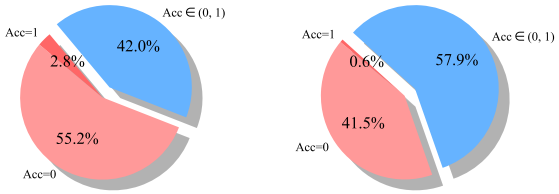
For mathematical reasoning tasks, since DAPO-Math-17k-Processed datasets contain high levels of redundancy—typically due to extensive data augmentation—we set a fixed training budget of 1,000 steps. This iteration count was found to be sufficient for achieving stable convergence while maintaining computational efficiency. For DAPO, we set the clipping parameter $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. For GRPO, we set the "loss_agg_mode" to "seq-mean-token-mean", which is aligned with the original paper. For GSPO, the clipping parameter ϵ is set to 0.0003.

Retriever and Corpus. For the multi-step retrieval task, we employ *multilingual-e5-base* as the retriever model and use the December 2018 Wikipedia dump as the primary retrieval corpus, which contains over 21 million passages. To improve retrieval efficiency, we construct the final corpus by combining supporting document passages from three multi-hop datasets (i.e., Musique, 2Wiki, and HotpotQA) with one million randomly sampled documents from the Wikipedia dump. Notably, Bamboogle only provides questions and answers without ground truth passages, so it cannot be incorporated into the retrieval corpus. This may contribute to the lower scores on Bamboogle for most methods, as shown in Table 3. Passage retrieval is implemented using FAISS*, and for each query, the top 5 passages are retrieved during both training and testing. For the KG (Knowledge Graph) data used in PVPO-DynaSearcher, we follow the approach and dataset provided by Wang et al. (2021), which is aligned with Hao et al. (2025a).

*https://pypi.org/project/faiss-gpu/

A.2 Group Sampling Analysis

We calculate the data filtering ratio on two training sets, as shown in Figure 4. Group Sampling removes samples with Acc = 1 or 0 before training, filtering out 40%-60% of the total dataset. This leads to a 1.7–2.5× increase in training efficiency.



(a) Musique (b) DAPO-Math-17k-Processed

Figure 4: Group Sampling study on datasets from different fields. The Acc is the mean of the answer accuracies from M trajectories rolled out by the reference model. M=5 in Figure (a) and M=16 in Figure (b).

A.3 Reasoning Behavior Analysis

To verify that PVPO does not merely improve accuracy at the cost of undesirable behavioral changes, we provide a systematic analysis of reasoning behavior. Table 7 reports trajectory and tool-use statistics averaged across four multi-step retrieval datasets using the DynaSearcher framework and Qwen2.5-7B-Instruct model. Two key behavioral shifts emerge from PVPO training.

Think efficiency: PVPO substantially reduces redundant deliberation, with Think tokens per call dropping 66.6% (from 96.6 to 32.3). This indicates that PVPO consolidates fragmented reasoning into more compact, decisive inference rather than over-analyzing before acting.

Tool call quality: PVPO issues more tool calls per query (from 2.2 to 3.0) with richer, more context-aware queries (Tool tokens per call: from 35.6 to 39.8), reflecting deeper and more systematic information seeking.

Together, these behavioral changes indicate that PVPO shifts the model from hesitant over-analysis toward precise, action-oriented reasoning, a pattern well aligned with effective agentic behavior.

A.4 Additional Generalization Analysis

We evaluate PVPO's generalization along two dimensions: compatibility with existing RL algorithms, and transferability across model families.

We examine whether PVPO is compatible with existing RL algorithms by integrating it with DAPO and GSPO on multi-step retrieval tasks. Specifi-

Table 7: Reasoning behavior statistics of GRPO and PVPO on multi-step retrieval tasks averaged across four benchmarks. "Total" denotes trajectory length excluding system prompt. "w/o Tool Results" excludes retrieved document tokens. "Calls" denotes number of tool invocations. "Think" refers to tokens within <think> tags.

Metric	GRPO	PVPO
Trajectory Length (tokens)		
Total	1652.9	2169.7
w/o Tool Results	446.7	301.3
Tool Interaction		
Call count	2.2	3.0
Tool tokens / sample	76.4	115.9
Tool tokens / call	35.6	39.8
Reasoning Tokens		
Think tokens / sample	294.9	71.2
Think tokens / call	96.6	32.3

cally, we combine PVPO with the sequence-level importance ratio module proposed in GSPO and remove the KL loss constraint as introduced in DAPO. The results, shown in Table 8, demonstrate that PVPO not only provides strong baseline improvements over GRPO, but also achieves further performance gains when integrated with these advanced RL methods. In particular, the combination with DAPO (w/o KL) yields the best accuracy and LasJ scores, while integration with GSPO's sequence-level importance ratio also presents consistent improvements. In particular, the combination with DAPO (w/o KL) yields the best accuracy and LasJ scores, but also incurs significantly more tool calls (8.14 per query), resulting in greater inference costs. By contrast, GSPO's sequence-level importance ratio offers improvements with relatively lower tool call overhead (2.19 per query). Therefore, the trade-off between performance and inference cost should be considered when choosing an integration strategy for different practical scenarios. These findings confirm that PVPO is highly compatible and complementary when used alongside other state-of-the-art RL algorithms.

We validate PVPO's transferability across model families by conducting experiments on Qwen3-8B under the ReSearch framework. As shown in Table 9, PVPO consistently improves over GRPO across all four retrieval benchmarks, achieving a 5.5-point gain in average accuracy. This improvement is consistent in magnitude with that observed on Qwen2.5-7B (+5.8 points, Table 3), confirming that PVPO's Static V mechanism generalizes effectively across model families.

Table 8: Experimental results of PVPO's orthogonal integration with SOTA RL algorithms (DAPO, GSPO) and scalability evaluation on multi-step retrieval tasks. "w/ Seq-Ratio" refers to the sequence-level importance ratio from GSPO, and "w/o KL" means removing the KL loss constraint as in DAPO.

Method	Average		
	Acc	LasJ	ToolCalls
GRPO-ReSearch	48.6	58.0	2.46
PVPO-ReSearch	54.4	62.6	2.96
w/ Seq-Ratio (GSPO)	55.1	62.4	2.19
w/o KL (DAPO)	**58.8**	**67.1**	8.14

Table 9: Generalization of PVPO to Qwen3-8B on multi-step retrieval tasks.

	Musique	2Wiki	HotpotQA	Bam	Avg
GRPO	37.5	63.7	59.5	46.4	51.8
PVPO	**41.3**	**76.9**	**63.6**	**47.2**	**57.3**

A.5 Additional Ablation Analysis

To evaluate the effectiveness of PVPO across different model capacities, we conducted ablation experiments on a 14B parameter model, maintaining the same experimental setup as our 7B runs. As illustrated in Table 10, the results demonstrate that PVPO maintains its superior performance and efficiency at a larger scale. The complete PVPO framework consistently outperforms all baseline variants, confirming that our design principles, particularly the Group Sampling and Static V Estimate, remain robust as the model capacity increases.

We investigated whether the timing (i.e., the specific training step) of the Static V Estimate update significantly influences final performance. As shown in Table 11, performance remains stable across a broad interval of 400–900 steps (Avg Acc: 60.7–62.5), all of which substantially exceed the no-update baseline of 57.4 (Table 3, PVPO w/ GTIR). This plateau suggests that the mechanism is robust to moderate variation in update timing and does not require intensive hyperparameter search. And our analysis reveals three distinct phases:

- **Early Stage:** Updating V too prematurely

Table 10: Model-scale ablation comparison of PVPO and GRPO methods on multi-step retrieval datasets. "GTIR" means Ground Truth trajectory Injection and Replacement, "VU" means $V_{\pi_{\theta_{\text{sta}}}}$ Updating. Starting from ReCALL on **Qwen2.5-14B-Instruct**, we first establish the Static V Estimate as a baseline, then incrementally augment the model with GTIR and VU.

Method	Musique	2Wiki	HotpotQA	Bamboogle	Avg Acc
GRPO-ReCALL					
Default	37.4	73.5	59.1	53.6	55.9
w/ GTIR	39.6	77.0	62.4	55.2	58.6
PVPO-ReCALL					
w/o both	36.7	76.4	60.6	50.2	56.0
w/ GTIR	36.2	79.1	**67.3**	52.0	58.7
w/ VU	42.9	82.8	65.3	56.1	61.8
w/ both	**44.7**	**83.7**	66.6	**56.8**	**63.0**

Table 11: Ablation experiments comparing different update step for PVPO. The experiment is conducted using Qwen2.5-7B-Instruct model with ReCALL on multi-step retrieval datasets.

step for VU	Musique	2Wiki	HotpotQA	Bamboogle	Avg Acc
200	40.0	73.7	67.4	48.8	57.5
300	39.0	77.4	64.6	53.6	58.7
400	44.1	75.9	68.6	54.0	60.7
500	**45.1**	77.2	72.4	**55.2**	**62.5**
600	44.5	**80.6**	**72.6**	51.2	62.2
700	43.0	76.6	70.5	54.4	61.1
800	**45.1**	76.9	67.9	54.4	61.1
900	44.5	76.7	69.8	52.8	60.9

yields limited gains. At this stage, the model has not yet grasped the fundamental patterns of the reasoning task; a premature baseline can be disproportionately high relative to the model's current competence, leading to suboptimal advantage estimation.

- **Mid-Stage (Optimal):** Updating V after the model has mastered basic task regularities provides the most stable and significant performance boost. The baseline during this period accurately reflects the task difficulty, facilitating effective and stable policy optimization. Based on this observation, we adopt step 500 as a universal hyperparameter across all 7B and 14B models and all nine benchmarks, without per-task tuning.

- **Late Stage:** Delaying the update until the late training stage leads to performance degradation. Due to the high baseline V and the model's converged state, exploratory attempts struggle to yield better solutions, making it difficult for the model to refine its policy further.

For practitioners applying PVPO to new tasks or model families, we recommend updating V at approximately one-third to one-half of the total training steps as a reliable heuristic.

To explore the performance upper bound of PVPO and the influence of GT count, we extended the GT generation to samples of Group 2 (0<Acc<1), effectively transitioning PVPO toward a more off-policy optimization methods. Following the configuration of LUFFY, we also ablated the clipping mechanism in this setting. The experimental results are shown in Table 12.

- **GT Count vs. Efficiency:** The results indicate a positive correlation between the number of GT trajectories and final performance. However, this gain comes at the cost of a significantly higher computational burden, reducing the overall "economic efficiency" of the training process.

Table 12: Number of ground truth trajectories ablation comparison of PVPO on multi-step retrieval datasets. "GT" means Ground Truth trajectory . "CLIP" means the truncation operation in importance sampling. Starting from ReCALL on Qwen2.5-7B-Instruct, we incrementally add the number of GT.

number of GT	CLIP	Musique	2Wiki	HotpotQA	Bamboogle	Avg Acc
0	w/	43.4	77.4	<u>70.8</u>	<u>54.4</u>	61.5
3k	w/	42.1	**83.7**	66.0	**56.0**	<u>62.0</u>
3k	w/o	39.2	68.1	63.5	48.8	54.9
11k	w/	**45.6**	<u>80.4</u>	**73.5**	**56.0**	**63.9**
11k	w/o	<u>44.5</u>	75.1	70.2	52.8	60.7

- **Importance of Clipping:** Removing the clipping operation significantly destabilizes the optimization. Without this constraint, aggressive exploration leads to large policy shifts that degrade performance, highlighting that the clipping mechanism is crucial for maintaining stability when leveraging high-density reference data.

A.6 Detailed Formula Derivation

A.6.1 Explanation of GAE Simplification

The derivation in Figure 5 Eq. 11 illustrates how the GAE formulation behaves under trajectory-level rewards.

Reward Substitution: In the third line, we apply the condition that all intermediate rewards r_k are zero, except for the final step $T-1$. This isolates the terminal reward $R(\tau)$ from the summation.

Telescoping Rearrangement: The fourth line represents a regrouping of the value function terms $V(s)$. By expanding the summation, we observe that each intermediate state value $V(s_{t+l})$ is scaled by $(1-\lambda)$.

Convergence to Monte Carlo: The final line shows that the estimator naturally simplifies to the Monte Carlo advantage. In the context of trajectory-level rewards, relying on the full outcome $R(\tau)$ to ensure an unbiased signal logically corresponds to the boundary case of GAE where $\lambda = 1$. As the coefficient $(1-\lambda)$ vanishes, the dependency on potentially biased intermediate value predictions is removed. This yields an advantage signal that is purely grounded in the final realized reward and the current state's baseline $V(s_t)$.

A.6.2 Explanation of PVPO Advantage

Building upon the Monte Carlo advantage, Static V Estimate of PVPO (Figure 5 Eq. 12) introduces stability by anchoring the baseline to a static policy checkpoint.

Static Value Substitution: In the first line, we replace the standard dynamic critic $V(s_{i,t})$ with a static state-value function $\hat{V}_{\pi_{\theta_{\text{sta}}}}$. This serves as a fixed reference point to calculate the "relative" improvement of the current policy.

Expectation via Static Rollouts: The second and third lines define the static value as the expected Monte Carlo return if the trajectory were completed by a fixed reference policy $\pi_{\theta_{\text{sta}}}$ starting from the current state $s_{i,t}$.

Practical Implication: This formulation ensures that an action receives a positive advantage only if it leads to a final outcome $R(\tau_i)$ that exceeds the average performance of the frozen baseline policy on the same task.

A.7 Prompts

We implement **PVPO-ReSearch**, **PVPO-DynaSearcher** and **PVPO-ReCALL** based on the ReSearch framework[†]. The system prompts for ReSearch and DynaSearcher are set following their respective original papers, detailed prompt templates are shown in Figure 7 , 8 and 9. For prompt-based advanced LLMs, we first retrieve 5 passages from the corpus for each question, and then organize these passages using the template shown in Figure 6 as the prompt for answer generation. For mathematical reasoning tasks, we use `verl` version `0.3.1.dev0`.

A.8 Code

Since the ReSearch codebase is also developed on top of the verl framework, we provide the core implementation of our PVPO method based on verl in code Listing 1 and Listing 2.

[†] https://github.com/Agent-RL/ReCall/tree/re-search

A. Derivation of Monte Carlo Advantage from GAE

$$
\begin{aligned}
\hat{A}_t^{\text{GAE}} &= \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \quad \text{where} \quad \delta_k = r_k + \gamma V(s_{k+1}) - V(s_k). \\
&= \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \big[r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l}) \big] \\
&= (\gamma\lambda)^{T-t-1} R(\tau) + \sum_{l=0}^{T-t-2} (\gamma\lambda)^l \big[\gamma V(s_{t+l+1}) - V(s_{t+l}) \big], \quad \text{where} \quad r_k = \begin{cases} 0, & k < T-1, \\ R(\tau), & k = T-1. \end{cases} \\
&= -V(s_t) + \sum_{l=1}^{T-t-1} \gamma^l \lambda^{l-1} (1-\lambda) V(s_{t+l}) + (\gamma\lambda)^{T-t-1} R(\tau) \\
&\overset{\lambda=1}{=} \gamma^{T-1-t} R(\tau) - V(s_t).
\end{aligned} \tag{11}
$$

B. Derivation of PVPO Advantage with Static V

$$
\begin{aligned}
\hat{A}_{i,t}^{\text{PVPO}} &= \gamma^{T-1-t} R(\tau_i) - \hat{V}_{\pi_{\theta_{\text{sta}}}}(s_{i,t}) \\
&= \gamma^{T-1-t} R(\tau_i) - \mathbb{E}_{\tau \sim \pi_{\theta_{\text{sta}}}} \left[\sum_{l=t}^{T-1} \gamma^{l-t} r_l \,\middle|\, S_t = s_{i,t} \right] \\
&= \gamma^{T-1-t} R(\tau_i) - \mathbb{E}_{\tau \sim \pi_{\theta_{\text{sta}}}} \left[\gamma^{T-1-t} R(\tau) \,\middle|\, S_t = s_{i,t} \right]
\end{aligned} \tag{12}
$$

Figure 5: Step-by-step mathematical derivation of advantage estimators. (A) Reduction of GAE to Monte Carlo advantage under trajectory-level rewards. (B) Formulation of the PVPO advantage using a static policy checkpoint as the value baseline.

You are an expert in question answering. Given a question within <question> </question> and some contexts within <context> </context>, you first think about the reasoning process within <think> </think> and put the answer within <answer> </answer>. For example, <question> This is a question <question> <context> Here are contexts <context> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{ answer here } </answer>. If the answer could not be deduced from the contexts or it's wrong, give the right answer based on your own knowledge. In the last part of the answer, the final exact answer is enclosed within \boxed{}.

Figure 6: Prompt for zero-shot LLM RAG.

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search for fact information about specific topics if needed. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags respectively, and the search query and result are enclosed within <search> </search> and <result> </result> tags respectively. For example, <think> This is the reasoning process. </think> <search> search query here </search> <result> search result here </result> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{ answer here } </answer>. In the last part of the answer, the final exact answer is enclosed within \boxed{}.

Figure 7: System prompt for ReSearch.

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search for fact information about specific topics if needed. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags respectively, and the search input and result are enclosed within <search> </search> and <result> </result> tags respectively. Search input is json format like {"query": "xxx", "entity": ["yyy"], "relation": ["zzz"]} and applied to the search tools, where query is used to search wikipedia articles, entity(s) and relation(s) are used to search wikidata, a knowledge base of entities and relations.
For example, <think> This is the reasoning process. </think> <search> {"query": "Who is the director of Avatar", "entity": ["Avatar"], "relation": ["director"]} </search> <result> search result here </result> <think> This is the reasoning process. </think> <answer> The final answer is \boxed{ answer here }</answer>. In the last part of the answer, the final exact answer is enclosed within \boxed{}.

Figure 8: System prompt for DynaSearcher.

In this environment you have access to a set of tools you can use to assist with the user query. You may perform multiple rounds of function calls. In each round, you can call one or more functions. Here are available functions in JSONSchema format: json {func_schemas}
In your response, you need to first think about the reasoning process in the mind and then conduct function calling to get the information or perform the actions if needed. The reasoning process and function calling are enclosed within <think> </think> and <tool_call> </tool_call> tags. The results of the function calls will be given back to you after execution, and you can continue to call functions until you get the final answer for the user's question. Finally, if you have got the answer, enclose it within \boxed{{}} with latex format and do not continue to call functions, i.e., <think> Based on the response from the function call, I get the weather information. </think> The weather in Beijing on 2025-04-01 is [\boxed{{20C}}].
For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags: <tool_call> {{"name": <function-name>, "arguments": <args-json-object>}} </tool_call>.

Figure 9: System prompt for ReCALL.

You will be provided with three pieces of content: the questioner's question, the user's response, and the reference answer list. Your task is to score the accuracy of the user's response based on the criteria outlined below. Please ensure that you carefully read and understand these instructions.
Evaluation Criteria: 1. The pred answer doesn't need to be exactly the same as any of the ground truth answers, but should be semantically same for the question. 2. Each item in the ground truth answer list can be viewed as a ground truth answer for the question, and the pred answer should be semantically same to at least one of them. 3. The user's response may be longer and more detailed; as long as it is logically correct, contains the correct answer, it should be scored appropriately.
Evaluation Steps: 1. Carefully read the questioner's question and understand its key points. 2. Carefully read the reference answer and understand the key points relevant to the question. 3. Based on the evaluation criteria, assign a score in the range of 0 to 5, where 0 indicates that the user's response does not include any of the key points from the reference answer and completely fails to answer the questioner's question; 5 indicates that the user's response includes all the key points from the reference answer and fully and correctly answers the questioner's question.
Questioner's question: {question}
Reference answer: {answer}
User's response: {response}
Evaluation result (output only the score between 0 and 5):

Figure 10: Prompt for LLM-as-Judge score.

```python
# verl/trainer/ppo/ray_trainer.py
class RayPPOTrainer(object):
    def fit(self):
        ...
        for epoch in range(self.config.trainer.total_epochs):
            for batch_dict in self.train_dataloader:
            ...
            if self.config.algorithm.adv_estimator == AdvantageEstimator.PVPO:
                fully_incorrect_uids = find_fully_incorrect_examples(acc_tensor, index=batch.non_tensor_batch["uid"])

                for prompt_uid in fully_incorrect_uids:
                    batch, reward_tensor = replace_rollout_with_gt(
                        prompt_uid_to_replace=prompt_uid,
                        data=batch,
                        reward_tensor=reward_tensor,
                        uids=batch.non_tensor_batch["uid"],
                        pad_token_id=self.tokenizer.pad_token_id
                        )
            ...
        ...

# verl/trainer/ppo/ray_trainer.py
def replace_rollout_with_gt(prompt_uid_to_replace, data, reward_tensor, uids, pad_token_id):
    response_length = data.batch["responses"].shape[1]
    candidate_rows = np.where(uids == prompt_uid_to_replace)[0]

    if len(candidate_rows) == 0: return data, reward_tensor
    row_idx = candidate_rows[0] # Pick the first rollout of this prompt to replace

    try:
        gt_tokens = data.non_tensor_batch['gt_tokens'][row_idx]
        gt_log_probs = data.non_tensor_batch['gt_log_probs'][row_idx]
        actual_gt_len = len(gt_tokens)
        if actual_gt_len > response_length: return data, reward_tensor
    except: return data, reward_tensor

    device = data.batch["responses"].device

    padded_tokens = torch.full((response_length,), pad_token_id, device=device, dtype=torch.long)
    padded_tokens[:actual_gt_len] = torch.tensor(gt_tokens, device=device)
    data.batch["responses"][row_idx] = padded_tokens

    padded_log_probs = torch.full((response_length,), 0.0, device=device, dtype=torch.float)
    padded_log_probs[:actual_gt_len] = torch.tensor(gt_log_probs, device=device)
    data.batch["old_log_probs"][row_idx] = padded_log_probs


    prompt_len = data.batch["input_ids"].shape[1] - response_length
    data.batch["input_ids"][row_idx, prompt_len:] = padded_tokens
    new_mask = (padded_tokens != pad_token_id).to(data.batch["attention_mask"].dtype)
    data.batch["attention_mask"][row_idx, prompt_len:] = new_mask


    reward_tensor[row_idx, :] = 0.0
    if actual_gt_len > 0:
        reward_tensor[row_idx, actual_gt_len - 1] = 1.0

    return data, reward_tensor
```

Listing 1: PyTorch-style pseudocode for ground truth trajectory replacement of Group Sampling in PVPO

```python
# verl/trainer/ppo/ray_trainer.py
def compute_advantage(...):
    if adv_estimator == AdvantageEstimator.PVPO:
        # compute pvpo advantages
        advantages, returns = core_algos.compute_pvpo_advantage(
            token_level_rewards=data.batch["token_level_rewards"],
            token_level_values=data.non_tensor_batch["static_value"],
            response_mask=data.batch["response_mask"],
        )
        data.batch["advantages"] = advantages
        data.batch["returns"] = returns
    ...
# verl/trainer/ppo/core_algos.py
def compute_pvpo_outcome_advantage(
    token_level_rewards: torch.Tensor,
    token_level_values: torch.Tensor,
    response_mask: torch.Tensor,
):
    scores = token_level_rewards.sum(dim=-1)
    values = torch.tensor(token_level_values.astype(np.float32),
                          device=scores.device,
                          dtype=scores.dtype
                          )

    with torch.no_grad():
        for i in range(scores.shape[0]):
            scores[i] = (scores[i] - values[i])
        scores = scores.unsqueeze(-1) * response_mask
    return scores
```

Listing 2: PyTorch-style pseudocode for advantage calculation of Static V Estimate in PVPO