Missing Old Logits in Asynchronous Agentic RL: Semantic Mismatch and Repair Methods for Off-Policy Correction

Anonymous Authors[1]

Abstract

Asynchronous reinforcement learning improves rollout throughput for large language model agents by decoupling sample generation from policy optimization, but it also introduces a concrete failure mode for PPO-style off-policy correction. In heterogeneous training systems, the total importance ratio should ideally decompose into two semantically distinct factors: a *training–inference discrepancy term* that aligns inference-side and training-side distributions at the same behavior-policy version, and a *policy-staleness term* that constrains the update from the historical policy to the current policy. We give an operational boundary for this failure: when delayed updates or partial rollouts make the required historical training-side logits unavailable, discrepancy repair and staleness correction become entangled. This missing-old-logit problem breaks the intended semantics of decoupled correction and makes clipping and masking thresholds interact in trace-level diagnostics such as Train-Infer Mask and PPO-CLIP activation. We study both exact and approximate repair routes. For exact repair, we examine snapshot-based version tracking, a dedicated old-logit model, and synchronization via partial rollout interruption, and report their system trade-offs. For approximate repair, we use a revised PPO-EWMA reference policy to preserve much of the benefit of decoupled correction when exact old logits are too expensive to recover. Experiments on dense and MoE backbones show that exact recovery is a useful upper bound, while PPO-EWMA improves training speed and final task performance with substantially lower system overhead.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Large-scale reinforcement learning for large language models (LLMs) increasingly relies on distributed rollout and training pipelines. Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its variants (Yu et al., 2025; Qi et al., 2026; Ahmadian et al., 2024a;b), including GRPO (Shao et al., 2024), remain widely used because they provide a simple and stable mechanism for policy improvement: trajectories are generated by a behavior policy, and the current policy is optimized with an importance ratio and a clipped surrogate objective. In ideal on-policy or near-synchronous settings, this ratio has a well-defined operational meaning. It compares the current policy against the policy that generated the sampled tokens, while clipping trades off update magnitude and optimization stability.

This interpretation becomes fragile in modern Agentic RL systems. To maximize throughput, rollout and training are often physically separated. Rollouts are produced by optimized inference engines such as vLLM (Kwon et al., 2023) or SGLang (Zheng et al., 2024), whereas gradient updates are performed by training engines such as Megatron-LM or FSDP. Even when the inference and training sides nominally use the same model version, numerical kernels, precision scaling, quantization, tensor parallelism, and routing implementations can lead to different token probabilities. We call this effect *training–inference discrepancy* (Yao et al., 2025a). At the same time, asynchronous rollouts, large rollout queues, partial trajectories, and multiple actor updates make the behavior policy stale with respect to the current policy. We call this effect *policy staleness*.

A natural choice for correction is to decompose the total ratio into two terms: a discrepancy-repair ratio that compares the training-side and inference-side distributions at the same old version, and a staleness-correction ratio that compares the current training policy with that old training-side policy (Xiao et al., 2026; Team et al., 2026; Zeng et al., 2026; Wang et al., 2026; Team et al., 2025). Let μ_{old} denote the inference-side rollout policy, and let π_{old} denote the corresponding training-side forward policy. The desired

decomposition is

$$r(\theta) = \frac{\pi_\theta(y|x)}{\mu_{\text{old}}(y|x)} = r_d r_s,\ r_d = \frac{\pi_{\text{old}}(y|x)}{\mu_{\text{old}}(y|x)},\ r_s = \frac{\pi_\theta(y|x)}{\pi_{\text{old}}(y|x)}. \tag{1}$$

Here r_d measures training–inference discrepancy, while r_s measures policy staleness. This decomposition is attractive because the two terms have different meanings and should be controlled differently. Discrepancy repair should filter or down-weight numerically inconsistent tokens. Staleness correction should constrain policy updates with the sign-dependent PPO clipping rule.

However, asynchronous Agentic RL (Dong et al., 2025; Wang et al., 2025b; Zhang et al.) introduces a practical failure boundary: the old training-side policy values $\pi_{\text{old}}(y|x)$ may no longer be available when the trajectory reaches the actor. This boundary is triggered most clearly by delayed updates and partial rollout collection, where one trajectory can span multiple parameter versions and the actor may already have advanced beyond the version that generated earlier tokens. Once these old logits are missing, the decomposition in Eq. (1) is no longer semantically valid. Existing decoupled objectives may then mix discrepancy repair and staleness correction into a proxy ratio, causing the clipping and masking mechanisms to interfere with each other. Current training stacks, including Verl (Sheng et al., 2025b), ROLL (Wang et al., 2025a), and SLIME (Zhu et al., 2025), still leave this old-logit mismatch largely unresolved.

This paper studies the missing-old-logit problem in asynchronous LLM RL as a reproducible correction failure rather than as a generic instability. We first give a unified view of existing objectives as imposing two distinct constraints: a discrepancy constraint and a staleness constraint. This view shows why using one ratio or one threshold for both effects is insufficient. We then analyze interpolation-based proxy policies and show that, under common constructions, they mainly re-parameterize effective clipping boundaries rather than recovering the missing reference policy. Finally, we examine two practical directions: exact acquisition of old logits through system support, and low-cost approximation through an exponentially-weighted moving average PPO (PPO-EWMA) reference policy (Hilton et al., 2022).

Our contributions are summarized as follows.

- **We identify an operational failure boundary in asynchronous Agentic RL.** Missing training-side old logits break the intended separation between training–inference discrepancy repair and policy-staleness correction, creating a semantic failure mode in decoupled correction objectives.
- **We provide a unified analysis and practical repair strategies.** We formulate existing PPO-style objectives under a dual-constraint view, clarify the need to decouple discrepancy repair from staleness correction, and show that interpolation-based proxies mainly re-parameterize clipping boundaries. We further study three exact old-logit acquisition routes and a revised PPO-EWMA reference as a low-cost approximation.
- **We evaluate trace-level behavior and performance–cost trade-offs on dense and MoE LLMs.** Experiments on Agentic benchmarks compare exact recovery, proxy references, and PPO-EWMA using task success, Train-Infer Mask, PPO-CLIP activation, and system-overhead measurements.

2. Related Work

2.1. Off-Policy and Asynchronous Reinforcement Learning for LLMs

PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) are widely used in LLM reinforcement learning because their clipped objectives stabilize policy updates while remaining straightforward to implement at scale (Yu et al., 2025). However, on-policy training can be inefficient for long-horizon Agentic tasks, where rollout generation is expensive and GPU utilization is often limited by synchronization (Guan et al., 2026). This has motivated off-policy and asynchronous RL pipelines that reuse stale trajectories and decouple rollout generation from policy optimization.

Several recent methods (Chen et al., 2025; Su et al., 2025) improve off-policy robustness by modifying the importance-sampling weights. CISPO (Chen et al., 2025) clips or regularizes importance weights for long-sequence training. GPPO (Su et al., 2025) separates gradient propagation from clipping constraints to preserve useful exploratory gradients. M2PO (Zheng et al., 2025c) controls the second moment of importance weights to reduce variance under stale data. VESPO (Shen et al., 2026) and VCPO (Huang et al., 2026) use effective sample size as a stability signal, while MiniRL (Zheng et al., 2025a) and TOPR (Roux et al., 2025) modify trajectory-level importance weighting through tapered or asymmetric weighting. System-oriented work such as AReaL (Fu et al., 2025), HybridFlow (Sheng et al., 2025b), and related asynchronous frameworks study how to overlap rollout and training clusters at large scale.

These works demonstrate that off-policy and asynchronous training can substantially improve throughput. Our work focuses on a complementary issue: in heterogeneous asynchronous LLM RL, the policy version needed for a clean correction may be missing. This creates a failure condition in the correction mechanism itself, making the meaning of the importance ratio ambiguous even before variance control or clipping design is considered.

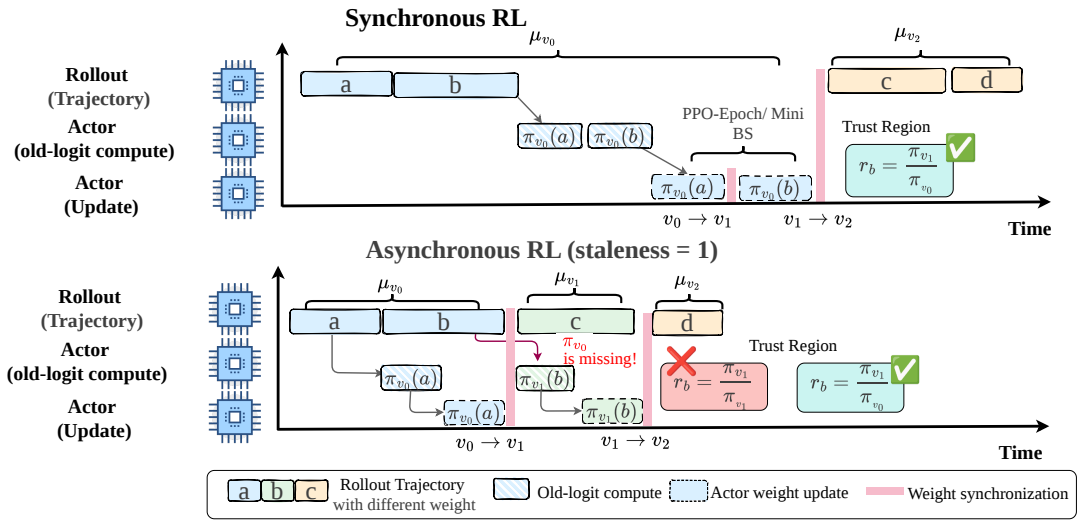


Figure 1. Synchronous versus asynchronous RL. In synchronous RL, the old logits used during training correspond to the same policy version that generated the rollout. In asynchronous RL, delayed updates and partial rollouts can make this version unavailable, which creates an old-logit mismatch and breaks the intended separation between training–inference discrepancy repair and policy-staleness correction.

2.2. Training-Inference Mismatch and Reference Policy Correction

Training–inference mismatch arises when the inference engine that produces rollouts and the training engine that computes gradients implement slightly different numerical computations. The mismatch is especially visible in MoE models, where routing decisions can amplify small numerical differences (Zheng et al., 2025b). Existing approaches mitigate this instability through masking, clipping, or routing replay. Masked Importance Sampling (MIS) (Liu et al., 2025) masks tokens with severe training–inference divergence. IcePop (Zhao et al., 2025) combines bilateral clipping and token masking to reduce the effect of unstable low-probability tokens. Routing-replay methods such as R2 (Zheng et al., 2025a) and R3 (Ma et al., 2025) align expert routing between rollout and training, thereby reducing MoE-specific discrepancy.

A separate line of work builds reference or proximal policies to stabilize stale updates. Decoupled PPO (Zheng et al., 2025a) separates importance correction from proximal constraints, while A-3PO (Li et al., 2025) approximates the proximal policy via log-space interpolation to reduce overhead. PPO-EWMA-style references maintain a smoothed policy anchor. These methods motivate our decoupled view. Our key distinction is that we examine whether the reference policy is semantically correct in asynchronous systems. When exact old training-side logits are absent, a proxy reference can help, but remains an approximation rather than true recovery of Eq. (1).

3. Preliminaries

3.1. PPO-Style Off-Policy Correction

We consider RL fine-tuning of an LLM on prompts $x \sim P$. Given a prompt x, a response $y = (y_1, \ldots, y_T)$ is sampled from an old policy π_{old}. In this standard PPO notation, π_{old} denotes the behavior policy, and we do not yet distinguish the inference-side rollout distribution from the training-side forward distribution. A reward model or environment returns a scalar reward $R(x, y)$, and an advantage estimate A_t is computed for each token or sequence.

For a token-level ratio $r_t(\theta) = \pi_\theta(y_t|x, y_{<t}) / \pi_{\text{old}}(y_t|x, y_{<t})$, the PPO clipped surrogate is

$$\ell_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]. \tag{2}$$

Equivalently, PPO clipping induces an advantage-sign-dependent active region. For $A_t > 0$, ratios above $1+\epsilon$ are clipped; for $A_t < 0$, ratios below $1-\epsilon$ are clipped. Therefore, at the per-token level, we rewrite the gradient contribution of Eq. (2) into a masked importance sampling (MIS) form: $\mathbb{E}_t[\text{MIS} \cdot r_t(\theta)A_t \nabla \log \pi_\theta(y_t|x, y_{<t})]$, where the PPO-side active mask is defined as

$$\begin{aligned} \text{MIS} = &\mathbb{I}\{A_t \geq 0\}\mathbb{I}\{r_t(\theta) \leq 1+\epsilon\} \\ &+ \mathbb{I}\{A_t < 0\}\mathbb{I}\{r_t(\theta) \geq 1-\epsilon\}, \end{aligned} \tag{3}$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function. This advantage-sign-dependent mask is mainly used to enforce the policy-update constraint (Schulman et al., 2017), preventing the policy update from becoming too large.

3.2. Training-Inference Discrepancy, Policy Staleness, and Missing Old Logits

Under modern asynchronous LLM RL systems (Fu et al., 2025; Sheng et al., 2025a;b; Wang et al., 2025a), the same parameter version can induce two distributions: the rollout distribution deployed on the inference engine, such as vLLM or SGLang, and the forward distribution deployed on the training side, such as Megatron or FSDP. Throughout this paper, we use μ to denote the inference-side policy and π to denote the training-side policy. The subscript v denotes the policy version, such as μ_v and π_v. By default, we use θ to denote the current version being optimized on the actor engine, and old to denote the rollout policy version used to generate the token on the inference engine. Therefore, the importance ratio can be naturally decomposed into a staleness ratio r_s and a discrepancy ratio between actor and rollout, r_d, i.e., $r_t(\theta) = r_s \times r_d$, where $r_s = \frac{\pi_\theta(y_t|x,y_{<t})}{\pi_{\mathrm{old}}(y_t|x,y_{<t})}$ and $r_d = \frac{\pi_{\mathrm{old}}(y_t|x,y_{<t})}{\mu_{\mathrm{old}}(y_t|x,y_{<t})}$.

Some works consider controlling these two terms separately, for example by masking the discrepancy ratio to mitigate the impact of numerical discrepancies (Yao et al., 2025b; Ma et al., 2025). More recently, IcePop (Zhao et al., 2025) proposes using a strict masking threshold c for the discrepancy ratio, which can be formulated as an MIS objective:

$$\mathrm{MIS} = M_{[1/c,c]}(r_d)\Big(\mathbb{I}\{A_t \geq 0\}\mathbb{I}\{r_s \leq 1+\epsilon\} + \mathbb{I}\{A_t < 0\}\mathbb{I}\{r_s \geq 1-\epsilon\}\Big), \tag{4}$$

where we define the masking function $M_{[1/c,c]}(r) := \mathbb{I}\{1/c \leq r \leq c\}$.

Although this strategy has also been verified on various foundation models (Xiao et al., 2026; Team et al., 2026; Zeng et al., 2026; Wang et al., 2026; Team et al., 2025), *a central practical difficulty remains: π_{old} may be missing*. Specifically, as shown in Figure 2, the rollout version old is often outdated with respect to both the behavior model and the actor model, and therefore may have already been discarded. This is particularly common in training systems that involve partial rollout collection and asynchronous model updates (Fu et al., 2025; Sheng et al., 2025a;b; Wang et al., 2025a).

In practice, existing works therefore often replace π_{old} with an approximation, for example by using a linearly interpolated policy (Li et al., 2025) or, more generally, a policy version between old and θ as a surrogate. Such a decomposition does not affect the algebraic correctness of the loss function, but the two factors no longer correspond to pure discrepancy repair and pure staleness correction. This semantic entanglement is exactly the old-logit mismatch problem.

4. A Unified Analysis of Decoupled Correction

In this section, we first provide intuition on why discrepancy repair cannot substitute for staleness correction; namely, why the decoupled approach in Eq. (4) cannot be replaced by the standard PPO clip in Eq. (3). We then analyze existing off-policy corrections and show how they can be unified into the form of Eq. (4). Finally, we make explicit how old-logit mismatch creates a falsifiable failure mode: the same nominal correction objective no longer measures the quantities it is intended to control.

4.1. Why Discrepancy Repair Cannot Substitute for Staleness Correction

The intuition for why the dual-side correction in Eq. (4) cannot simply be expressed by the standard PPO correction in Eq. (3) is two-fold. First, PPO primarily prevents overly large update steps by applying an asymmetric filter based on the advantage sign, whereas training-inference discrepancy repair requires a strict, symmetric constraint centered around 1. Second, blending these decomposed terms into a single ratio forces a shared threshold, fundamentally compromising optimization. Because discrepancy repair targets numerical consistency while staleness correction controls update magnitude, they naturally demand different levels of constraint strength. A strict shared constraint stably filters out errors but severely bottlenecks learning, whereas a looser constraint accelerates early training but exposes the policy to noisy, compounded updates that increase the risk of oscillation or collapse.

We further quantify this effect in Section 6.4, where exact-old-logit experiments show how discrepancy masking and PPO-CLIP still interact through the final active-token set.

4.2. A Unified View of Existing Off-Policy Corrections

As shown in Table 1, existing off-policy methods in LLM RL generally decouple the optimization process into a discrepancy ratio r_d and a staleness ratio r_s. In synchronous settings, an accessible and semantically correct old policy π_{old} allows for an exact decomposition of training–inference discrepancy and policy staleness. However, in asynchronous RL, the latency between training and generation engines introduces an unavoidable version mismatch ($\theta \geq \mathrm{async} \geq \mathrm{old}$). This breaks the semantic consistency of the reference policy, corrupting the meanings of both r_d and r_s and causing standard decoupled corrections to fail.

To mitigate this missing reference without heavy infrastructure overhead, a common strategy is to construct a probability-space proximal policy π_{prox} through interpolation between the current and behavior policies, such

Table 1. Summarization of PPO variants under a unified Masked Importance Sampling (MIS) view.

Algorithm	Discrepancy Ratio r_d	Staleness Ratio r_s	Proxy Definition
General Format: $\text{MIS} = M_{[1/c,c]}(r_d)\Big(\mathbb{I}_{A_t\geq 0}\mathbb{I}_{r_s\leq 1+\epsilon} + \mathbb{I}_{A_t<0}\mathbb{I}_{r_s\geq 1-\epsilon}\Big)$			
PPO-clip (standard)	1	$\frac{\pi_\theta}{\pi_{\text{old}}}$	–
PPO-clip & train_infer	1	$\frac{\pi_\theta}{\mu_{\text{old}}}$	–
decoupled PPO & train_infer	$\frac{\pi_{\text{old}}}{\mu_{\text{old}}}$	$\frac{\pi_\theta}{\pi_{\text{old}}}$	–
PPO-EWMA	1	$\frac{\pi_\theta}{\pi_{\text{prox}}}$	$\theta_{\text{prox},t} = \beta\theta_{\text{prox},t-1} + (1-\beta)\theta_t$
PPO-EWMA & train_infer	$\frac{\pi_{\text{prox}}}{\mu_{\text{old}}}$	$\frac{\pi_\theta}{\pi_{\text{prox}}}$	$\theta_{\text{prox},t} = \beta\theta_{\text{prox},t-1} + (1-\beta)\theta_t$
linear_prox	1	$\frac{\pi_\theta}{\pi_{\text{prox}}}$	$\pi_{\text{prox}} = \alpha\mu_{\text{old}} + (1-\alpha)\pi_\theta$
linear_prox & train_infer	$\frac{\pi_{\text{prox}}}{\mu_{\text{old}}}$	$\frac{\pi_\theta}{\pi_{\text{prox}}}$	$\pi_{\text{prox}} = \alpha\mu_{\text{old}} + (1-\alpha)\pi_\theta$
decoupled PPO & Async	$\frac{\pi_{\text{async}}}{\mu_{\text{old}}}$	$\frac{\pi_\theta}{\pi_{\text{async}}}$	$\theta \geq \text{async} \geq \text{old}$

as `linear_prox` or token-wise log-linear interpolation. However, this approach does not genuinely resolve the discrepancy, as stated below:

Proposition 1. *Let $r(\theta) = \pi_\theta/\mu_{\text{old}}$. If π_{prox} is constructed via arithmetic interpolation or token-wise log-linear interpolation, then clipping and masking on the decoupled ratios merely re-parameterizes the effective constraint boundaries of the single total ratio $r(\theta)$. Full derivations are provided in Appendix A.*

Because interpolation only shifts effective boundaries rather than restoring an exact reference, we explore two distinct directions to resolve the old-logit mismatch in the following sections. The first approach relies on systematic infrastructure support to directly acquire the ground-truth old logits. The second acknowledges limited system overhead and constructs a more reliable, approximate reference using a revised exponential moving average that explicitly accounts for asynchronous delays.

5. Recovering and Approximating Old Logits

5.1. Exact Old-Logit Acquisition

We first consider exact acquisition of $\pi_{\text{old}}(y_t|x, y_{<t})$, the training-side token probability under the rollout version. Figure 2 illustrates three possible strategies.

Snapshot-based version tracking. The most direct solution is to retain historical parameter snapshots and reload the version that generated each token or trajectory. This gives the cleanest estimate of π_{old} and therefore restores the semantic decomposition in Eq. (1). Its drawback is system cost. Snapshot retention requires additional CPU or host memory, and exact recovery may require frequent actor-side version switching. With partial rollouts, a single sample can span multiple versions, which further increases switching and I/O overhead.

Dedicated old-logit model. A second option is to maintain a separate model that computes old logits while the main actor continues training. This can reduce contention on the actor path and allow overlap between old-logit computation and gradient updates. It also decouples old-logit computation from update training, so the overlap can reduce the end-to-end time of the actor stage.

Synchronization via partial rollout interruption. A third option computes old logits before a policy version disappears. Before updating parameters from version v to $v+1$, the system interrupts rollout workers and returns partial trajectories. Since rollout is stopped during this interval, we can use Ray scheduling to release the rollout-side placement and temporarily switch the same resources to actor-side old-logit computation. The still-resident version v is then used to compute exact old logits for the returned partial trajectories. After the old-logit pass finishes, the system switches the resources back to rollout execution and resumes generation.

This design avoids storing old weights and can provide exact logits, but it introduces synchronization stalls, resource reconfiguration overhead, and disruption to rollout parallelism.

These three methods represent different points in a system trade-off: snapshots are exact but memory- and I/O-heavy;

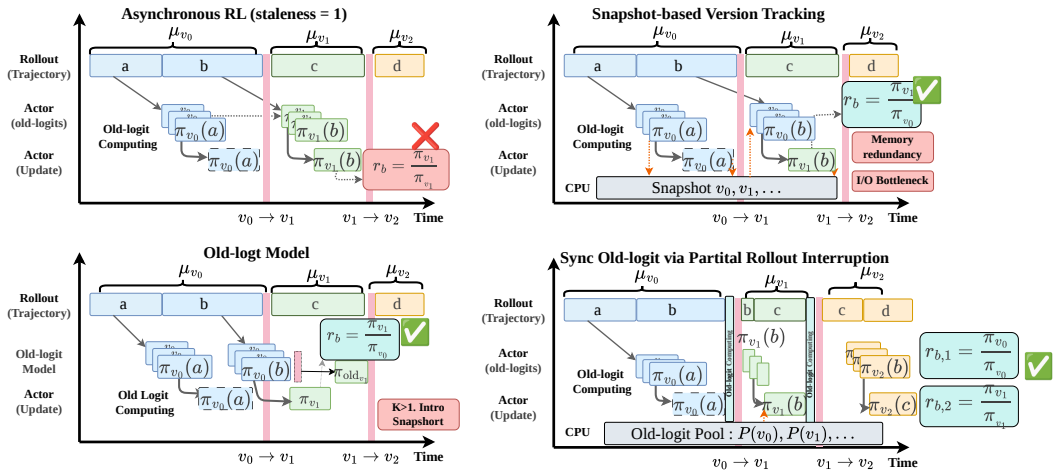


Figure 2. Exact old-logit acquisition in asynchronous RL. The top-left panel shows the original mismatch problem. The remaining panels show snapshot-based version tracking, a dedicated old-logit model, and synchronization via partial rollout interruption.

old-logit models enable overlap but require resource partitioning; partial interruption avoids historical storage but adds synchronization overhead.

5.2. Revised PPO-EWMA as a Low-Cost Reference Policy

Exact old-logit acquisition may be too expensive for large asynchronous Agentic RL. We therefore use a (PPO-EWMA) reference policy as a low-cost approximation (Hilton et al., 2022). The goal is not to claim exact recovery of π_{old}, but to construct a smoother reference that better tracks the center of the asynchronous version window than either the current policy or a static interpolation proxy.

PPO-EWMA maintains π_{prox} as an exponentially averaged reference policy. Given actor parameters $\theta^{(t)}$ after update step t, we use

$$\theta^{(t)}_{\text{prox}} = \frac{\sum_{k=0}^{t} \beta_{\text{prox}}^{t-k} \theta^{(k)}}{\sum_{k=0}^{t} \beta_{\text{prox}}^{t-k}}, \quad r_s = \frac{\pi_\theta}{\pi_{\text{prox}}}, \quad r_d = \frac{\pi_{\text{prox}}}{\mu_{\text{old}}}. \tag{5}$$

That is, we replace the unavailable π_{old} with π_{prox} for both staleness correction and discrepancy repair. This is only an approximate reference, not an exact recovery of old logits. The equivalent recursive update is provided in Appendix B.

Our adjustment is deliberately small. First, instead of using a fixed large decay, we set β_{prox} according to the expected staleness window, $\beta_{\text{prox}} \approx W_{\text{stale}}/(W_{\text{stale}}+2)$. This places the EWMA reference near the middle of the asynchronous version window and prevents it from lagging behind the rollout queue. Appendix B gives the center-of-mass derivation.

Second, we add an automatic reset to avoid accumulating excessively stale versions in the EWMA reference. As θ_{prox} averages over more historical actor states, it may drift away from the policy version used by recent rollouts. This makes the discrepancy ratio $\pi_{\text{prox}}/\mu_{\text{old}}$ deviate further from one and causes the Train-Infer Mask to reject many tokens. We therefore monitor the Train-Infer Mask value ρ_t, defined as the fraction of tokens that remain active after discrepancy masking and PPO clipping. When $\rho_t < \tau$, where τ is set to 0.9 in our experiments unless otherwise stated, we reset

$$\theta^{(t)}_{\text{prox}} \leftarrow \theta^{(t)}. \tag{6}$$

This clears stale history and re-centers the proxy reference around the current actor. Section 6.5 provides a more detailed analysis of this behavior.

6. Experiments

6.1. Experimental Setup

We evaluate Agentic RL tasks using two representative policy backbones: the dense `Qwen3-4B` model and the MoE `Qwen3-30B-A3B` model. Training data are drawn from an Agentic RL corpus spanning multiple environments. Evaluation covers the retail, airline, and telecom domains of τ^2-Bench (Barres et al., 2025), together with the in-store and delivery splits of VitaBench (He et al., 2025). For both benchmarks, we report task-level average success and pass metrics. To expose failures beyond terminal success, we also track trace-level diagnostics, including the Train-Infer Mask and PPO-CLIP activation ratio.

To isolate the failure mechanism studied in this paper, we adopt an asynchronous RL setup with explicit control over the maximum version gap between rollout workers and the actor, which we cap at three. We eliminate additional staleness from minibatch reuse and multiple PPO epochs where possible, ensuring that the observed mismatch is primarily

triggered by rollout–training asynchrony rather than by unrelated sources of off-policy drift. Detailed hyperparameters, resource configurations, and environment descriptions are provided in Appendix F.

We consider several practical methods: **Decoupled PPO**, which employs the decoupled objective but relies on the available asynchronous reference rather than the true π_{old} policy; **Linear_prox**, a lightweight linear interpolation-based proximal policy; and **PPO-EWMA**, our enhanced EWMA reference with staleness-aware decay and optional reset. Additionally, we include **Snapshot**, which recovers exact old logits via true version tracking. This is feasible in our setup because we control the maximum version gap.

6.2. Main Results

Table 2 compares exact old-logit recovery, proxy references, and PPO-EWMA on held-out Agentic benchmarks. Snapshot† serves as an idealized reference because it assumes exact old logits are available. PPO-EWMA is the practical intervention we emphasize: it consistently improves over Decoupled PPO and Linear_prox, and often approaches the Snapshot results.

On the dense 4B model, PPO-EWMA obtains the best pass@4 on retail and the best pass@2 on VitaBench in-store, while tying the best telecom scores. On the 30B MoE model, it is strongest on the airline split and ties the best retail pass@4 and telecom avg@2. These gains are most pronounced on tasks with long interaction horizons, such as VitaBench delivery and in-store evaluation, where stale trajectories accumulate more discrepancy before the update is applied. Compared with Linear_prox, PPO-EWMA improves the dense model by 6.78 points on VitaBench delivery avg@2 and the MoE model by 7.75 points on the same metric. The improvement is therefore not only a consequence of changing the effective clipping threshold; it is consistent with the claim that a maintained parameter-space reference better tracks the asynchronous policy window.

Snapshot† remains the strongest upper bound on several metrics, especially for the MoE backbone. This gap is expected because exact version tracking recovers the true π_{old} rather than approximating it. However, the gap between PPO-EWMA and Snapshot is substantially smaller than the gap between Decoupled PPO and Snapshot on most splits. This supports the intended use of PPO-EWMA: it is not a replacement for exact old-logit recovery when exact recovery is affordable, but it provides a practical reference policy when the system cost of exact recovery is prohibitive.

6.3. System Overhead of Exact Old-Logit Acquisition

Table 3 summarizes the system cost of exact old-logit acquisition. Snapshot is accurate but expensive: it requires version switching, snapshot storage, and extra recovery time, with the cost becoming much larger on the 30B MoE model. This is an important negative result for deployment: exact repair improves correction fidelity, but its infrastructure cost can dominate the actor path. PPO-EWMA is much cheaper because it only maintains a lightweight proxy reference. In the left subtable, its CPU storage and extra time are far lower than Snapshot for both backbones. The right subtable shows that a dedicated old-logit model can overlap part of the computation, but its benefit depends on the resource partition ratio and still requires additional model-side infrastructure.

Overall, Table 3 supports the same conclusion as Table 2: exact recovery is useful as a strong reference, but PPO-EWMA offers a better performance–cost trade-off for practical asynchronous training.

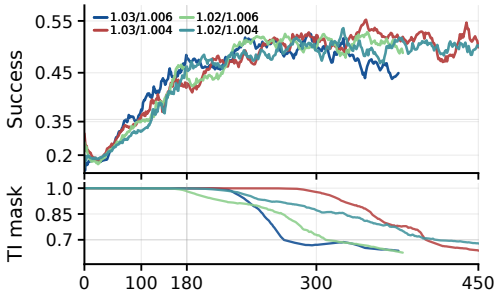


Figure 3. Exact-old-logit threshold trade-off. Each label reports the discrepancy and stale-policy thresholds.

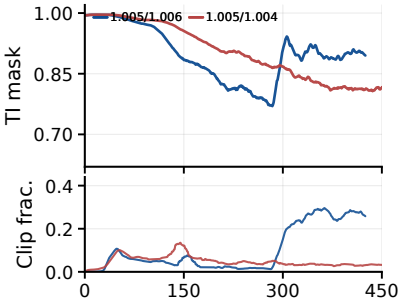


Figure 4. Interaction between discrepancy masking and PPO clipping under exact old logits.

6.4. Threshold Trade-off under Exact Old Logits

Using exact old logits from Snapshot, we study how the discrepancy and stale-policy thresholds affect optimization. All threshold-tradeoff curves in this section are collected with the `Qwen3-4B` backbone. In each run name, the first number denotes the discrepancy threshold and the second denotes the stale-policy threshold; for example, `snap1003_1006` corresponds to 1.003 and 1.006.

Figure 3 summarizes the trade-off. A looser discrepancy threshold keeps more tokens active and improves early learn-

Table 2. Main results on the held-out Agentic benchmark suite for the dense `Qwen3-4B` model and the `Qwen3-30B-A3B` MoE model. VitaBench is separated into in-store and delivery splits. Snapshot† denotes an idealized setting where exact old logits are available. The best performance among practical methods (first three rows) is **bolded**.

Backbone	Method	τ^2-Bench (Retail)		τ^2-Bench (Airline)		τ^2-Bench (Telecom)		VitaBench (In-store)		VitaBench (Delivery)	
		avg@4	pass@4	avg@4	pass@4	avg@2	pass@2	avg@2	pass@2	avg@2	pass@2
`Qwen3-4B`	Decoupled PPO	63.96	88.60	53.5	72	40	50	19.83	37	19.56	33
`Qwen3-4B`	Linear_prox	64.40	86.84	**54**	72	37.5	50	22.37	40	19.10	28
`Qwen3-4B`	PPO-EWMA	**65.72**	**90.35**	**54**	**74**	**42.5**	**52.5**	**25**	**50**	**25.88**	**39**
`Qwen3-4B`	Snapshot†	66.23	89.47	56	76	42.5	52.5	28.89	47	27.33	42
`Qwen3-30B-A3B`	Decoupled PPO	65.43	89.47	57	76	44.75	53	18.28	32	25.88	39
`Qwen3-30B-A3B`	Linear_prox	65.8	87.7	53.5	74	44	51	31.47	47	20.74	33
`Qwen3-30B-A3B`	PPO-EWMA	**67.82**	**92.1**	**60**	**82**	**45**	**54**	**33.41**	**48**	**28.49**	**43**
`Qwen3-30B-A3B`	Snapshot†	69.70	92.1	59	80	45	55	34.62	50	30.74	45

Table 3. System-overhead measurement protocols for exact old-logit acquisition. Snapshot recovery mainly adds version-switch latency and CPU storage; a dedicated old-logit model changes the overlapped time allocation.

(a) Snapshot vs PPO-EWMA.

Metric	4B	30B	4B	30B
Switch latency (s)	7	14.2	7	14.2
CPU storage (GB)	40	76.4	7.9	15.2
Extra time (s)	25	150	8	34

(b) Dedicated old-logit model.

Ratio	Old-logit (s)	Update (s)	Single (s)	Overlap (s)	Change (%)
1:2	243	272	305	284	-6.8
1:3	243	206	237	254	+7.17

ing, but it also admits more biased off-policy tokens and can destabilize later trajectories. A stricter threshold slows the beginning but yields smoother retained-token dynamics and reduces late collapse. Appendix D shows that this behavior persists for nearby threshold pairs: as the discrepancy threshold is tightened from 1.005 to 1.002, looser stale-policy thresholds still tend to accelerate early learning, while stricter settings produce smoother late-stage trajectories.

Figure 4 further shows that discrepancy masking and PPO clipping are coupled. With the same discrepancy threshold, a looser stale-policy threshold lets more problematic tokens survive early, causing the Train-Infer Mask to drop lower; later, stronger PPO clipping caps the remaining updates and helps the mask recover.

6.5. PPO-EWMA Analysis and Ablation

We next study PPO-EWMA through Task success, Train-Infer Mask, and PPO-CLIP ratio. These diagnostics separate terminal performance from whether the proxy reference remains usable during optimization. The main issue is reference-policy lag: the EWMA reference must be smooth enough to stabilize updates, but not so stale that it collapses the Train-Infer Mask. We therefore use a staleness-aware decay and reset the reference when the mask becomes too low.

Appendix E provides the detailed ablations. In brief, large decay values can accumulate stale history and collapse the mask, while very small decay values weaken the stabilizing anchor. The staleness-aware setting with occasional reset preserves most of the early Task success gain while recovering a high Train-Infer Mask, supporting PPO-EWMA as a low-cost approximate repair rather than a black-box tuning trick.

7. Conclusion

We identified and studied the missing-old-logit problem in asynchronous Agentic RL for LLMs, which undermines decoupled off-policy correction. We characterize the failure boundary, analyze the mechanism through a dual-constraint view, and evaluate both exact and approximate repair strategies. Our results show that exact recovery improves correction fidelity but can impose substantial systems overhead, while PPO-EWMA provides a practical alternative when exact recovery is too expensive. A limitation is that PPO-EWMA remains an approximate reference rather than a true reconstruction of the missing old policy, and may become unreliable under highly non-stationary staleness or extreme version gaps.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267,

2024a.

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024b. URL https://arxiv.org/abs/2402.14740.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Chen, A., Li, A., Gong, B., Jiang, B., Fei, B., Yang, B., Shan, B., Yu, C., Wang, C., Zhu, C., et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

Dong, G., Mao, H., Ma, K., Bao, L., Chen, Y., Wang, Z., Chen, Z., Du, J., Wang, H., Zhang, F., et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

Fu, W., Gao, J., Shen, X., Zhu, C., Mei, Z., He, C., Xu, S., Wei, G., Mei, J., JIASHU, W., et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Guan, Z., Sun, H., Guo, Y., Di, S., Bai, X., Long, J., Zhao, T., Luo, M., Zhou, C., Guo, Y., et al. Rl-vla3: Reinforcement learning vla accelerating via full asynchronism. *arXiv preprint arXiv:2602.05765*, 2026.

He, W., Sun, Y., Hao, H., Hao, X., Xia, Z., Gu, Q., Han, C., Zhao, D., Su, H., Zhang, K., et al. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. *arXiv preprint arXiv:2509.26490*, 2025.

Hilton, J., Cobbe, K., and Schulman, J. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

Huang, L. J., Zhang, Z., Hu, Q., Yang, S., and Han, S. Stable asynchrony: Variance-controlled off-policy rl for llms. *arXiv preprint arXiv:2602.17616*, 2026.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Li, X., Wu, S., and Shen, Z. A-3po: Accelerating asynchronous llm training with staleness-aware proximal policy approximation. *arXiv preprint arXiv:2512.06547*, 2025.

Liu, J., Li, Y., Fu, Y., Wang, J., Liu, Q., and Jiang, Z. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, September 2025. URL https://richardli.xyz/rl-collapse.

Ma, W., Zhang, H., Zhao, L., Song, Y., Wang, Y., Sui, Z., and Luo, F. Stabilizing moe reinforcement learning by aligning training and inference routers. *arXiv preprint arXiv:2510.11370*, 2025.

Qi, P., Zhou, X., Liu, Z., Pang, T., Du, C., Lin, M., and Lee, W. S. Rethinking the trust region in llm reinforcement learning. *arXiv preprint arXiv:2602.04879*, 2026.

Roux, N. L., Bellemare, M. G., Lebensold, J., Bergeron, A., Greaves, J., Fréchette, A., Pelletier, C., Thibodeau-Laufer, E., Toth, S., and Work, S. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shen, G., Zhao, C., Cheng, X., Huang, L., and Yu, X. Vespo: Variational sequence-level soft policy optimization for stable off-policy llm training. *arXiv preprint arXiv:2602.10693*, 2026.

Sheng, G., Tong, Y., Wan, B., Zhang, W., Jia, C., Wu, X., Wu, Y., Li, X., Zhang, C., Peng, Y., et al. Laminar: A scalable asynchronous rl post-training framework. *arXiv preprint arXiv:2510.12633*, 2025a.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025b.

Su, Z., Pan, L., Bai, X., Liu, D., Dong, G., Huang, J., Lv, M., Hu, W., Zhang, F., Gai, K., et al. Klear-reasoner: Advancing reasoning capability via gradient-preserving clipping policy optimization. *arXiv preprint arXiv:2508.07629*, 2025.

Team, K., Bai, T., Bai, Y., Bao, Y., Cai, S., Cao, Y., Charles, Y., Che, H., Chen, C., Chen, G., et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Team, L., Shen, A., Li, B., Hu, B., Jing, B., Chen, C., Huang, C., Zhang, C., Yang, C., Lin, C., et al. Every step evolves:

Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025.

Wang, H., Wu, H., Wu, T., Sun, Y., Liu, J., Yu, D., Ma, Y., He, J., He, Z., Hong, D., et al. Ernie 5.0 technical report. *arXiv preprint arXiv:2602.04705*, 2026.

Wang, W., Xiong, S., Chen, G., Gao, W., Guo, S., He, Y., Huang, J., Liu, J., Li, Z., Li, X., et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025a.

Wang, W., Xu, X., An, W., Dai, F., Gao, W., He, Y., Huang, J., Ji, Q., Jin, H., Li, X., et al. Let it flow: Agentic crafting on rock and roll, building the rome model within an open agentic learning ecosystem. *arXiv preprint arXiv:2512.24873*, 2025b.

Xiao, B., Xia, B., Yang, B., Gao, B., Shen, B., Zhang, C., He, C., Lou, C., Luo, F., Wang, G., et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Yao, F., Liu, L., Zhang, D., Dong, C., Shang, J., and Gao, J. Your efficient rl framework secretly brings you off-policy rl training, August 2025a. URL https://fengyao.notion.site/off-policy-rl.

Yao, F., Liu, L., Zhang, D., Dong, C., Shang, J., and Gao, J. Your efficient rl framework secretly brings you off-policy rl training, august 2025. *URL https://fengyao. notion. site/off-policy-rl*, 2025b.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., Chen, B., Yin, D., Ge, C., Huang, C., Xie, C., et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Zhang, G., Geng, H., Yu, X., Yin, Z., Zhang, Z., Tan, Z., Zhou, H., Li, Z.-Z., Xue, X., Li, Y., et al. The landscape of agentic reinforcement learning for llms: A survey. *Transactions on Machine Learning Research*.

Zhao, X., Liu, Y., Xu, K., Guo, J., Wang, Z., Sun, Y., Kong, X., Cao, Q., Jiang, L., Wen, Z., et al. Small leak can sink a great ship–boost rl training on moe with icepop!, 2025.

Zheng, C., Dang, K., Yu, B., Li, M., Jiang, H., Lin, J., Liu, Y., Lin, H., Wu, C., Hu, F., et al. Stabilizing reinforcement learning with llms: Formulation and practices. *arXiv preprint arXiv:2512.01374*, 2025a.

Zheng, C., Liu, S., Li, M., Chen, X.-H., Yu, B., Gao, C., Dang, K., Liu, Y., Men, R., Yang, A., et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025b.

Zheng, H., Zhao, J., and Chen, B. Prosperity before collapse: How far can off-policy rl reach with stale data on llms? *arXiv preprint arXiv:2510.01161*, 2025c.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

Zhu, Z., Xie, C., Lv, X., and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

A. Detailed Derivations for Interpolation-Based Proxies

This appendix gives the derivation behind Proposition 1. We write the total ratio between the current training policy and the behavior-side reference as:

$$r_\theta = \frac{\pi_\theta(y|x)}{\mu_{\text{old}}(y|x)}. \tag{7}$$

For a proximal policy π_{prox}, the decoupled objective applies a discrepancy mask to

$$r_d = \frac{\pi_{\text{prox}}(y|x)}{\mu_{\text{old}}(y|x)} \tag{8}$$

and PPO clipping to

$$r_s = \frac{\pi_\theta(y|x)}{\pi_{\text{prox}}(y|x)}. \tag{9}$$

For simplification, we consider the threshold for masking as ϵ_1 and ϵ_2, instead of c, i.e., the masked importance sampling is:

$$\text{MIS} = M_{[1-\epsilon_1,1+\epsilon_1]}(r_d)\left(\mathbb{I}\{A_t \geq 0\}\mathbb{I}\{r_s \leq 1+\epsilon_2\} + \mathbb{I}\{A_t < 0\}\mathbb{I}\{r_s \geq 1-\epsilon_2\}\right). \tag{10}$$

The key question is whether constructing π_{prox} by interpolation introduces a genuinely new reference, or whether it only changes the effective constraints on r_θ. We use α for the behavior-policy weight, consistent with Table 1.

Log-linear interpolation. Consider the token-wise log-linear interpolation:

$$\log \widetilde{\pi}_{\text{prox}}(y|x) = \alpha \log \mu_{\text{old}}(y|x) + (1-\alpha)\log \pi_\theta(y|x), \quad \alpha \in (0,1). \tag{11}$$

In this case, the discrepancy ratio becomes:

$$r_d = \frac{\widetilde{\pi}_{\text{prox}}}{\mu_{\text{old}}} = \left(\frac{\pi_\theta}{\mu_{\text{old}}}\right)^{1-\alpha} = r_\theta^{1-\alpha}. \tag{12}$$

The mask condition $1-\epsilon_1 \leq r_d \leq 1+\epsilon_1$ restricts r_θ to the interval:

$$(1-\epsilon_1)^{1/(1-\alpha)} \leq r_\theta \leq (1+\epsilon_1)^{1/(1-\alpha)}. \tag{13}$$

The PPO-side ratio is $r_s = \pi_\theta/\widetilde{\pi}_{\text{prox}} = r_\theta^\alpha$. We analyze the constraints based on the sign of A_t:

- **Case** $A_t \geq 0$: The clip requires $r_s \leq 1+\epsilon_2$, implying $r_\theta \leq (1+\epsilon_2)^{1/\alpha}$. Combined with the mask, the active region for r_θ is:

$$(1-\epsilon_1)^{1/(1-\alpha)} \leq r_\theta \leq \min\left((1+\epsilon_1)^{1/(1-\alpha)}, (1+\epsilon_2)^{1/\alpha}\right). \tag{14}$$

- **Case** $A_t < 0$: The clip requires $r_s \geq 1-\epsilon_2$, implying $r_\theta \geq (1-\epsilon_2)^{1/\alpha}$. Combined with the mask, the active region is:

$$\max\left((1-\epsilon_1)^{1/(1-\alpha)}, (1-\epsilon_2)^{1/\alpha}\right) \leq r_\theta \leq (1+\epsilon_1)^{1/(1-\alpha)}. \tag{15}$$

First-order expansion shows $\epsilon_{1,\text{eff}} \approx \frac{\epsilon_1}{1-\alpha}$ and $\epsilon_{2,\text{eff}} \approx \frac{\epsilon_2}{\alpha}$. This shows that log-linear interpolation merely re-parameterizes the active region of the original ratio r_θ.

Arithmetic interpolation. Consider the linear proxy:

$$\pi_{\text{prox}}(y|x) = \alpha\mu_{\text{old}}(y|x) + (1-\alpha)\pi_\theta(y|x), \quad \alpha \in (0,1). \tag{16}$$

The discrepancy ratio is $r_d = \alpha + (1-\alpha)r_\theta$, so the mask condition $1-\epsilon_1 \leq r_d \leq 1+\epsilon_1$ becomes:

$$1 - \frac{\epsilon_1}{1-\alpha} \leq r_\theta \leq 1 + \frac{\epsilon_1}{1-\alpha}. \tag{17}$$

The PPO-side ratio is $r_s = \frac{r_\theta}{\alpha+(1-\alpha)r_\theta}$. Again, we analyze by the sign of A_t:

Original mask	Original clip	n	Interpolation	Mask on r_d	Clip on r_s
$[0.990, 1.010]$	$[0.997, 1.004]$	1	Linear	$[0.9800, 1.0200]$	$[0.9940, 1.0080]$
$[0.990, 1.010]$	$[0.997, 1.004]$	1	Log-linear	$[0.9801, 1.0201]$	$[0.9940, 1.0080]$
$[0.990, 1.010]$	$[0.997, 1.004]$	2	Linear	$[0.9850, 1.0150]$	$[0.9911, 1.0121]$
$[0.990, 1.010]$	$[0.997, 1.004]$	2	Log-linear	$[0.9850, 1.0150]$	$[0.9910, 1.0120]$
$[0.990, 1.010]$	$[0.997, 1.004]$	3	Linear	$[0.9867, 1.0133]$	$[0.9881, 1.0162]$
$[0.990, 1.010]$	$[0.997, 1.004]$	3	Log-linear	$[0.9867, 1.0134]$	$[0.9881, 1.0161]$
$[0.995, 1.005]$	$[0.997, 1.004]$	1	Linear	$[0.9900, 1.0100]$	$[0.9940, 1.0080]$
$[0.995, 1.005]$	$[0.997, 1.004]$	1	Log-linear	$[0.9900, 1.0100]$	$[0.9940, 1.0080]$
$[0.995, 1.005]$	$[0.997, 1.004]$	2	Linear	$[0.9925, 1.0075]$	$[0.9911, 1.0121]$
$[0.995, 1.005]$	$[0.997, 1.004]$	2	Log-linear	$[0.9925, 1.0075]$	$[0.9910, 1.0120]$
$[0.995, 1.005]$	$[0.997, 1.004]$	3	Linear	$[0.9933, 1.0067]$	$[0.9881, 1.0162]$
$[0.995, 1.005]$	$[0.997, 1.004]$	3	Log-linear	$[0.9933, 1.0067]$	$[0.9881, 1.0161]$
$[0.990, 1.010]$	$[0.996, 1.006]$	1	Linear	$[0.9800, 1.0200]$	$[0.9920, 1.0121]$
$[0.990, 1.010]$	$[0.996, 1.006]$	1	Log-linear	$[0.9801, 1.0201]$	$[0.9920, 1.0120]$
$[0.990, 1.010]$	$[0.996, 1.006]$	2	Linear	$[0.9850, 1.0150]$	$[0.9881, 1.0182]$
$[0.990, 1.010]$	$[0.996, 1.006]$	2	Log-linear	$[0.9850, 1.0150]$	$[0.9880, 1.0181]$
$[0.990, 1.010]$	$[0.996, 1.006]$	3	Linear	$[0.9867, 1.0133]$	$[0.9842, 1.0244]$
$[0.990, 1.010]$	$[0.996, 1.006]$	3	Log-linear	$[0.9867, 1.0134]$	$[0.9841, 1.0242]$
$[0.980, 1.020]$	$[0.997, 1.004]$	1	Linear	$[0.9600, 1.0400]$	$[0.9940, 1.0080]$
$[0.980, 1.020]$	$[0.997, 1.004]$	1	Log-linear	$[0.9604, 1.0404]$	$[0.9940, 1.0080]$
$[0.980, 1.020]$	$[0.997, 1.004]$	2	Linear	$[0.9700, 1.0300]$	$[0.9911, 1.0121]$
$[0.980, 1.020]$	$[0.997, 1.004]$	2	Log-linear	$[0.9702, 1.0301]$	$[0.9910, 1.0120]$
$[0.980, 1.020]$	$[0.997, 1.004]$	3	Linear	$[0.9733, 1.0267]$	$[0.9881, 1.0162]$
$[0.980, 1.020]$	$[0.997, 1.004]$	3	Log-linear	$[0.9734, 1.0268]$	$[0.9881, 1.0161]$

Table 4. Effective thresholds on the total ratio $r_\theta = \pi_\theta/\mu_{\text{old}}$ induced by interpolation-based proxies. The first block corresponds to the thresholds used in our experiment. For version gap n, we set the behavior-policy weight $\alpha = 1/(n+1)$.

- **Case** $A_t \geq 0$: Solving $r_s \leq 1 + \epsilon_2$ for r_θ gives:

$$r_\theta \leq \frac{\alpha(1+\epsilon_2)}{1-(1-\alpha)(1+\epsilon_2)}. \tag{18}$$

  The active region is the intersection of the lower mask bound and the tighter of the two upper bounds.

- **Case** $A_t < 0$: Solving $r_s \geq 1 - \epsilon_2$ for r_θ gives:

$$r_\theta \geq \frac{\alpha(1-\epsilon_2)}{1-(1-\alpha)(1-\epsilon_2)}. \tag{19}$$

  The active region is the intersection of the upper mask bound and the tighter of the two lower bounds.

Applying $(1-x)^{-1} \approx 1 + x$ reveals an effective radius $\epsilon_{2,\text{eff}} \approx \frac{\epsilon_2}{\alpha}$. Thus, arithmetic interpolation also functions as a set of scaled constraints on the single total ratio r_θ.

Numerical effective thresholds. Table 4 instantiates the above mappings for version gaps $n \in \{1, 2, 3\}$, where the behavior-policy weight is $\alpha = 1/(n+1)$. We include the thresholds used in our experiments and several nearby settings. The clip intervals are allowed to be asymmetric because our implementation uses separate lower and upper PPO-CLIP thresholds; the symmetric PPO interval is a special case. Across these settings, interpolation changes the effective mask and clip ranges on the same total ratio rather than introducing an independent old-policy reference.

Figure 5 provides an empirical check of the same reparameterization view. After removing the interpolation and applying the corresponding reparameterized constraints directly, the training curve is almost identical to the log-linear interpolation

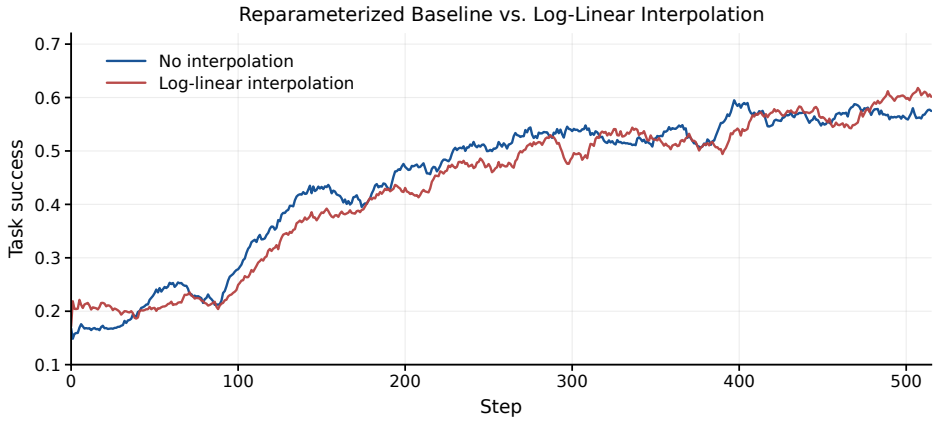


Figure 5. Training curves for the reparameterized no-interpolation baseline and the log-linear interpolation proxy. The two curves are nearly indistinguishable over training, consistent with the derivation that log-linear interpolation mainly changes the ratio parameterization.

run. This supports the interpretation that the interpolation proxy mainly changes the effective ratio boundary rather than introducing an independent correction effect.

These derivations explain why interpolation-based proxies can act as useful stabilizers but should not be interpreted as exact training–inference discrepancy repair. The ratios remain deterministic transformations of $r_\theta = \pi_\theta/\mu_{\text{old}}$; no term reconstructs the missing training-side old policy logits.

B. PPO-EWMA Update Details

This appendix records the standard PPO-EWMA parameterization used in Section 5.2. It is included for completeness; the main text focuses on the two asynchronous adjustments, namely staleness-aware decay selection and auto-reset.

Let $\theta^{(t)}$ be the actor parameters after update step t, and let $\beta_{\text{prox}} \in (0, 1)$ be the EWMA decay. We maintain a cumulative normalization weight $w^{(t)}$ and proximal parameters $\theta^{(t)}_{\text{prox}}$ as

$$w^{(t)} = 1 + \beta_{\text{prox}} w^{(t-1)}, \qquad \theta^{(t)}_{\text{prox}} = \frac{1}{w^{(t)}}\theta^{(t)} + \beta_{\text{prox}} \frac{w^{(t-1)}}{w^{(t)}} \theta^{(t-1)}_{\text{prox}}. \tag{20}$$

Unrolling the recursion gives the closed-form expression used in the main text:

$$\theta^{(t)}_{\text{prox}} = \frac{\sum_{k=0}^{t} \beta_{\text{prox}}^{t-k} \theta^{(k)}}{\sum_{k=0}^{t} \beta_{\text{prox}}^{t-k}}. \tag{21}$$

Thus, $\theta^{(t)}_{\text{prox}}$ is a normalized exponentially weighted average of historical actor states. A larger β_{prox} keeps a longer memory of previous policies, while a smaller value places more weight on the current actor.

The decay factor also determines the effective look-back depth of the reference. In the stationary limit, the center of mass of the EWMA history is

$$\text{COM}_{\text{prox}} = \frac{\sum_{k=0}^{\infty} k \beta_{\text{prox}}^{k}}{\sum_{k=0}^{\infty} \beta_{\text{prox}}^{k}} = \frac{\beta_{\text{prox}}}{1 - \beta_{\text{prox}}}. \tag{22}$$

For an asynchronous version window of width W_{stale}, aligning this center of mass with the midpoint of the window gives

$$\frac{\beta_{\text{prox}}}{1 - \beta_{\text{prox}}} \approx \frac{W_{\text{stale}}}{2}, \qquad \beta_{\text{prox}} \approx \frac{W_{\text{stale}}}{W_{\text{stale}} + 2}. \tag{23}$$

C. Full PPO Variant Comparison

Table 5 provides the expanded comparison behind Table 1. We distinguish two ratios whenever possible: r_1 is the policy-update or staleness ratio used by PPO-style clipping, while r_2 is the training–inference discrepancy or proxy-reference ratio.

Table 5. Expanded comparison of PPO variants and proxy-policy corrections. The columns A_+ and A_- show the active regions for positive and negative advantages, respectively.

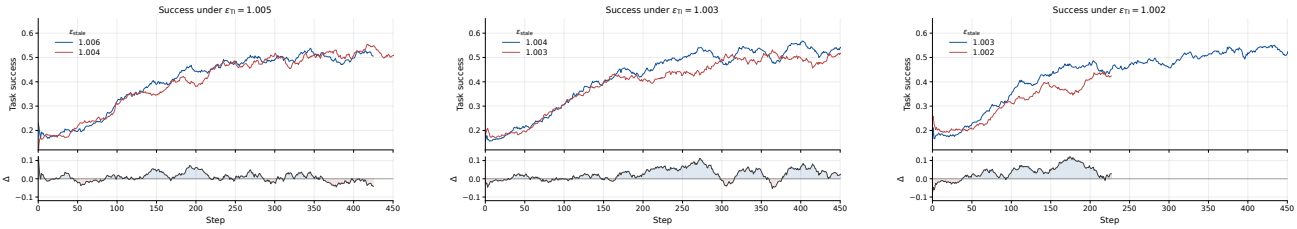
Algorithm	**Ratio Formula**	A_+	A_-	**Role**
PPO-clip (standard)	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{old}}(a_t\|s_t)}$	$r_1 \in (0, 1+\epsilon_1)$	$r_1 \in (1-\epsilon_1, \infty)$	$M(r_1)$
PPO-clip & train_infer	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$	$r_1 \in (0, 1+\epsilon_1)$	$r_1 \in (1-\epsilon_1, \infty)$	$M(r_1)$
decoupled PPO & train_infer	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{old}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{old}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$	$r_1 \in (0, 1+\epsilon_1)$ $r_2 \in (1 \pm \epsilon_2)$	$r_1 \in (1-\epsilon_1, \infty)$ $r_2 \in (1 \pm \epsilon_2)$	$M(r_2) \cdot M(r_1)$
PPO-EWMA	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{prox}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{prox}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$ $\theta_{\text{prox},t} = \beta\theta_{\text{prox},t-1} + (1-\beta)\theta_t$	$r_1 \in (0, 1+\epsilon_1)$	$r_1 \in (1-\epsilon_1, \infty)$	$r_2 \cdot M(r_1)$
PPO-EWMA & train_infer	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{prox}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{prox}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$ $\theta_{\text{prox},t} = \beta\theta_{\text{prox},t-1} + (1-\beta)\theta_t$	$r_1 \in (0, 1+\epsilon_1)$ $r_2 \in (1 \pm \epsilon_2)$	$r_1 \in (1-\epsilon_1, \infty)$ $r_2 \in (1 \pm \epsilon_2)$	$M(r_2) \cdot r_2 \cdot M(r_1)$
linear_prox	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{prox}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{prox}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$ $\pi_{\text{prox}} = \alpha\mu_{\text{old}} + (1-\alpha)\pi_\theta$	$r_1 \in (0, 1+\epsilon_1)$ $r_2 \in (1 \pm \epsilon_2)$	$r_1 \in (1-\epsilon_1, \infty)$ $r_2 \in (1 \pm \epsilon_2)$	$r_2 \cdot M(r_1)$
linear_prox & train_infer	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{prox}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{prox}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$ $\pi_{\text{prox}} = \alpha\mu_{\text{old}} + (1-\alpha)\pi_\theta$	$r_1 \in (0, 1+\epsilon_1)$	$r_1 \in (1-\epsilon_1, \infty)$	$M(r_2) \cdot r_2 \cdot M(r_1)$
decoupled PPO & train_infer & Async	$r_1 = \frac{\pi_\theta(a_t\|s_t)}{\pi_{\text{async}}(a_t\|s_t)}$ $r_2 = \frac{\pi_{\text{async}}(a_t\|s_t)}{\mu_{\text{old}}(a_t\|s_t)}$ $\theta \geq \text{async} \geq \text{old}$	$r_1 \in (0, 1+\epsilon_1)$ $r_2 \in (1 \pm \epsilon_2)$	$r_1 \in (1-\epsilon_1, \infty)$ $r_2 \in (1 \pm \epsilon_2)$	$M(r_2) \cdot M(r_1)$

For positive advantages, PPO clipping keeps samples before the upper clipping boundary; for negative advantages, it keeps samples after the lower clipping boundary. The discrepancy constraint, when used, is symmetric around one and does not depend on the advantage sign.

The standard PPO-clip objective uses a single ratio and therefore cannot separate policy staleness from training–inference discrepancy. The train_infer variant replaces the denominator with the rollout-side behavior distribution, but still applies PPO clipping to a mixed ratio. Decoupled PPO restores the intended two-factor structure when the training-side old policy π_{old} is available. In asynchronous training, however, this reference may be missing; using π_{async} or a constructed π_{prox} then gives a practical proxy rather than an exact decomposition. This is why proxy methods can stabilize optimization while still leaving the semantic old-logit mismatch unresolved.

D. Additional Threshold Examples

We provide additional threshold comparisons to show that the trade-off in Section 6.4 is not specific to one pair of hyperparameters. As in the main text, the first number in each run name is the training–inference discrepancy threshold and the second number is the stale-policy threshold. The upper panel of each success figure reports the task success curve, and the lower panel reports the difference between the two curves. These differences make the early-speed and late-stability



(a) `snap1005_1006` vs. `snap1005_1004`. *(b)* `snap1003_1004` vs. `snap1003_1003`. *(c)* `snap1002_1003` vs. `snap1002_1002`.

Figure 6. Additional exact-old-logit threshold comparisons. Across discrepancy thresholds 1.005, 1.003, and 1.002, looser stale-policy thresholds tend to accelerate early learning, while stricter settings produce smoother late-stage trajectories.

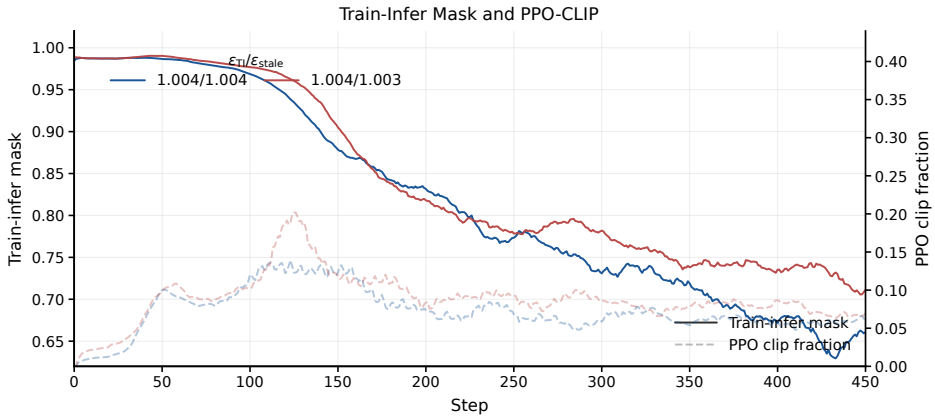


Figure 7. Additional interaction example: `snap1004_1004` vs. `snap1004_1003`. The discrepancy mask and PPO-CLIP activation change together even though the exact old logits are available.

trade-off easier to inspect.

Figures 6a–6c confirm that the early-speed versus late-stability trade-off persists as the discrepancy threshold is tightened from 1.005 to 1.002. At 1.005, loosening the stale-policy threshold produces a faster initial rise but leaves more correction pressure later. At 1.003, the same qualitative pattern remains but is weaker because the discrepancy filter has already removed more mismatched tokens. At 1.002, the retained signal is smallest, so training is slower but the trajectory is more controlled.

Figure 7 provides a second example of the coupling between discrepancy repair and PPO-CLIP. The two runs share discrepancy threshold 1.004 and differ in the stale-policy threshold. Changing the stale-policy threshold changes not only the PPO clip fraction, but also the trajectory of the training–inference mask. This confirms that the two mechanisms interact through the active-token set: a looser stale-policy constraint can leave more questionable tokens for discrepancy repair, while stronger discrepancy filtering changes how often PPO-CLIP becomes active on the remaining tokens. Thus, exact old-logit recovery restores the correct semantic decomposition, but threshold tuning still has to account for the joint behavior of the two constraints.

E. Additional PPO-EWMA Ablations

This appendix provides the detailed PPO-EWMA ablations behind Section 6.5. We use three diagnostic quantities: Task success, Train-Infer Mask, and PPO-CLIP ratio. Task success measures optimization progress, while Train-Infer Mask and PPO-CLIP ratio show whether the EWMA reference remains usable as a proximal policy.

Decay-factor behavior. Figure 8 compares $\beta = 0.9$, $\beta = 0.75$, and $\beta = 0.5$. When $\beta = 0.9$, the EWMA reference has a long memory and remains strongly affected by early actor versions. As training proceeds, π_{prox} becomes increasingly misaligned with recent rollout policies, so the discrepancy ratio $\pi_{\text{prox}}/\mu_{\text{old}}$ drifts away from one. This makes the Train-Infer

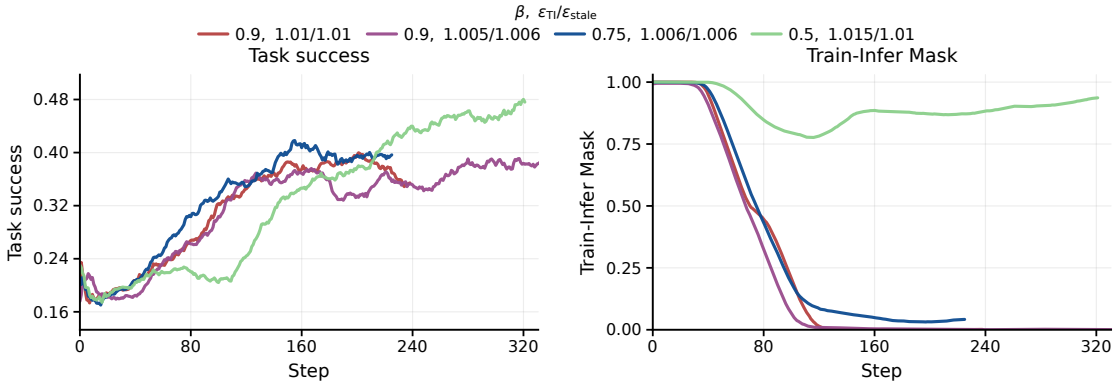


Figure 8. PPO-EWMA decay comparison. A large decay can accumulate stale reference history and collapse the Train-Infer Mask, while the staleness-aware decay gives faster early Task success.

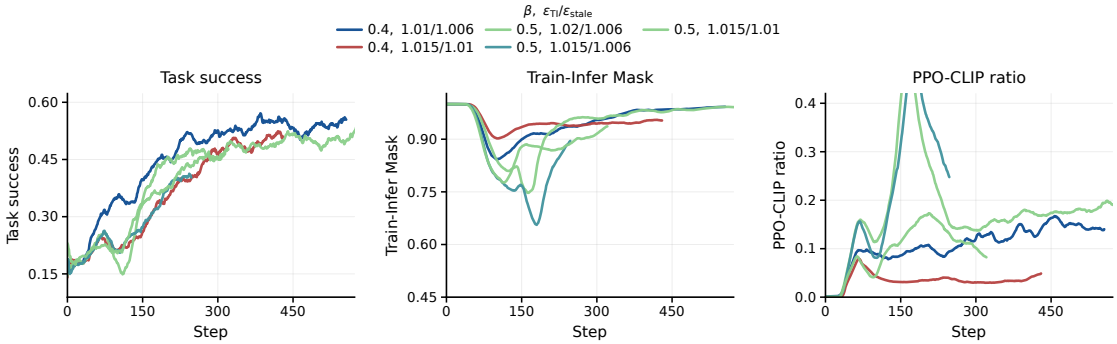


Figure 9. PPO-EWMA threshold interaction for $\beta = 0.4$ and $\beta = 0.5$. Looser Train-Infer Mask or PPO-CLIP ratio thresholds can improve early progress, but they also admit more noisy off-policy updates and may cause a mid-training success drop. The coupled mask and clip dynamics can later recover part of the trajectory by filtering or capping the problematic updates.

Mask progressively more aggressive. Tightening either the discrepancy threshold or the PPO-CLIP ratio threshold does not remove this failure mode: both $\beta = 0.9$ runs eventually reach an almost zero Train-Infer Mask value. This supports our interpretation that the issue is not only threshold choice, but also reference-policy lag.

The theoretically motivated $\beta = 0.75$ setting aligns the EWMA center of mass with the middle of the asynchronous version window. It therefore gives the fastest early Task success increase among the compared decays. However, without any reset, this run also suffers a late Train-Infer Mask collapse: the minimum Train-Infer Mask value falls below 2%, and the final Train-Infer Mask value remains only about 3%. In contrast, $\beta = 0.5$ has a shorter memory and keeps many more tokens active, but its Task success curve is less efficient early. These results show the core trade-off of PPO-EWMA: a longer memory provides a stronger stabilizing anchor, but can accumulate stale-policy bias; a shorter memory adapts faster, but weakens the proximal reference.

Figure 9 shows that threshold choice remains important even when the EWMA decay is small enough to avoid the severe collapse observed for $\beta = 0.9$. For $\beta = 0.4$ and $\beta = 0.5$, looser Train-Infer Mask or PPO-CLIP ratio thresholds retain more update signal early and can produce faster initial gains. However, these looser settings also allow more mismatched tokens to participate in optimization, which can cause a visible mid-training success drop. The subsequent recovery is consistent with the dual-constraint interpretation: discrepancy masking removes tokens whose Train-Infer ratio becomes too unreliable, while PPO-CLIP caps the update magnitude on the remaining tokens.

Automatic reset behavior. Figure 10 shows that automatic reset resolves the late-collapse failure while preserving most of the early efficiency of $\beta = 0.75$. The reset mechanism is conservative: in the two runs with thresholds $\tau = 0.9$ and $\tau = 0.8$, it is triggered only three and two times, respectively. Nevertheless, these few resets are sufficient to clear the stale history accumulated in θ_{prox} and recover a high Train-Infer Mask value. This indicates that the most important effect of reset is early re-centering of the proxy reference; after the reference is corrected, later training usually remains in a healthier region

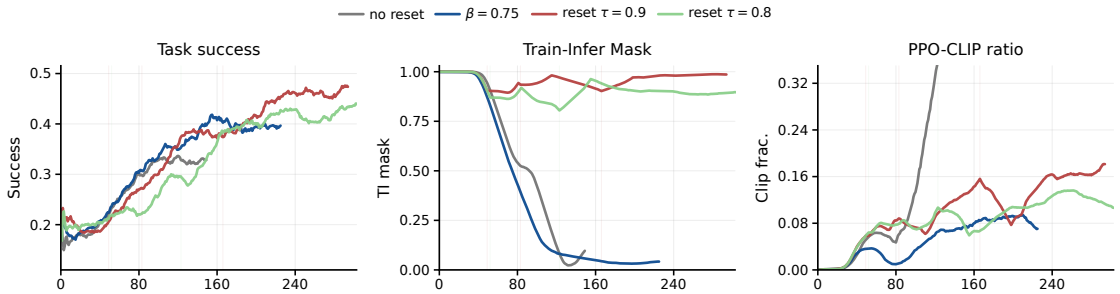


Figure 10. Effect of automatic reset for $\beta = 0.75$. Resetting the EWMA reference when the Train-Infer Mask becomes too low prevents late-stage collapse. Vertical lines mark reset events.

Metric	Qwen3-4B	Qwen3-30B-A3B
Measured steps	77	11
Snapshot size (GB)	8.04	15.29
Max resident snapshots	5	5
Max resident snapshot storage (GB)	40.22	76.43
Snapshot save time / step (s)	3.95	6.07
Snapshot load time (s)	7.11	14.22
Snapshot restore time (s)	3.01	5.66
Historical forward time (s)	45.05	147.18
Total snapshot-old stage time / step (s)	95.20	178.22
Version shards / batch	1.81	1.09
Historical version shards / batch	1.74	1.00
Max staleness gap	3	3

Table 6. Detailed overhead from snapshot-based old-logit computation. The dominant cost is the additional historical-model forward pass used to reconstruct old logits. Snapshot retention also adds substantial resident storage pressure, especially for the 30B-A3B model.

without frequent interventions.

The reset indicator further confirms that reset is not acting as a continuously active stabilizer. Most later steps have zero consecutive low-ratio streaks and no reset trigger. Therefore, the improvement should be interpreted as a small number of re-centering events that remove stale EWMA history after the reference has drifted too far, rather than as frequent intervention throughout training.

F. Experimental Details

Implementation framework. We use ROLL as the base training framework for our experiments. ROLL provides a well-structured agent abstraction.

G. Additional System Measurements

G.1. Snapshot Old-Logit Overhead

Table 6 reports the detailed overhead introduced by enabling snapshot-based old-logit computation. Both runs use `old_log_probs_source=snapshot_old` with asynchronous snapshot recovery enabled. The 4B run contains 77 measured training steps before termination, while the 30B-A3B run contains 11 measured steps. These numbers should therefore be interpreted as system-cost measurements from the observed run windows rather than final training results.

The main additional cost is not snapshot saving itself, but the combination of loading or restoring historical versions and running extra forward passes to compute old logits. In the 4B run, snapshot-old computation adds roughly 95 seconds per measured step on average. In the 30B-A3B run, the same stage adds roughly 178 seconds per measured step. The 30B-A3B setting also keeps up to 76.43GB of resident snapshot state when five snapshots are retained.

Both runs eventually terminate with Ray actor unavailability after the Raylet exits unexpectedly. The logs do not prove that snapshot recovery is the sole cause, but the measured resident snapshot storage and additional forward time show that exact old-logit recovery materially increases memory pressure and actor-side compute load.

H. Limitations

This work has several limitations. First, our experiments are conducted on representative dense and MoE backbones, but we have not yet validated the proposed correction methods on models at the several-hundred-billion-parameter scale. At that scale, memory pressure, communication overhead, expert routing behavior, and rollout–training scheduling can become qualitatively different from the settings studied here. As a result, the empirical trade-offs observed in our current experiments may not fully capture the behavior of extremely large industrial training runs.

Second, our infrastructure analysis focuses on the main sources of overhead for exact old-logit acquisition, including snapshot retention, version switching, historical forward computation, and partial rollout interruption. However, we do not provide a fine-grained end-to-end systems study of all infrastructure components. In particular, we have not exhaustively analyzed scheduler behavior, network communication, placement-group fragmentation, memory swapping, rollout worker idleness, or failure recovery under sustained large-scale asynchronous training.

Third, the throughput measurements in this paper should be interpreted as indicative rather than comprehensive. They are sufficient to show that exact old-logit recovery introduces non-trivial system cost, but we have not performed a large-scale throughput sweep across many cluster sizes, model scales, sequence lengths, rollout lengths, and staleness windows. A more complete systems evaluation would be needed to precisely characterize the throughput frontier and to identify the best engineering design for each deployment regime.