f-OPD: Stabilizing Long-Horizon On-Policy Distillation with Freshness-Aware Control

Xianwei Chen, Shimin Zhang, Jibin Wu*

The Hong Kong Polytechnic University

Abstract

Scaling on-policy distillation (OPD) for large language models (LLMs) confronts a fundamental tension: asynchronous execution is necessary for system efficiency, yet it structurally deviates from the ideal on-policy objective. To address this challenge, we theoretically decompose the objective discrepancy into *rollout drift* and *supervision drift*, capturing staleness in student rollout and teacher context, respectively. Building on this, we introduce a sample-level freshness score that quantifies the reliability of a buffered sample with respect to the on-policy objective. Guided by this signal, we further propose ***f*-OPD**, a novel framework that adaptively regulates the influence of stale samples and constrains policy drift accumulated under asynchronous training. Across reasoning, tool-use, and coding-agent tasks of increasing interaction horizon, f-OPD consistently achieves task performance comparable to synchronous optimization while largely retaining the throughput advantages of asynchronous execution. Our results establish the first recipe for achieving a performance–efficiency trade-off in OPD, paving the way for long-horizon agentic post-training at scale.

1 Introduction

The recent leap in large language models (LLMs) has been driven not only by pretraining but increasingly by advances in post-training. Over the past few years, paradigms such as supervised fine-tuning (SFT) and reinforcement learning (RL) have fundamentally reshaped how pretrained models are aligned, specialized, and deployed [1–3]. Most recently, on-policy distillation (OPD), by unifying on-policy learning with dense supervision, has emerged as a powerful technique, driving substantial gains in frontier LLMs [4–7].

A standard OPD pipeline consists of three tightly coupled processes: student policy rollout, teacher supervision for grading generated tokens, and student policy optimization. As shown in Fig. 1(a), these processes should be executed synchronously in principle so that each training step remains fully on-policy. Empirically, however, strict synchronization leads to substantial system under-utilization, especially in long-horizon settings where rollout and supervision become increasingly expensive. As a result, modern OPD systems [8–10] widely adopt asynchronous mode (Fig. 1(b)) to improve throughput by overlapping different stages of the pipeline and executing them in parallel. However, this efficiency gain comes at the cost of *staleness*: buffered trajectories and their associated supervision may no longer match the current student policy when used for optimization. Such a mismatch in freshness can distort the effective on-policy objective, destabilize optimization, and ultimately degrade task performance. This creates a fundamental *performance–efficiency trade-off* in OPD: improving system utilization through asynchrony often deteriorates optimization fidelity, while preserving on-policy consistency requires sacrificing throughput.

*Corresponding author, jibin.wu@polyu.edu.hk

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

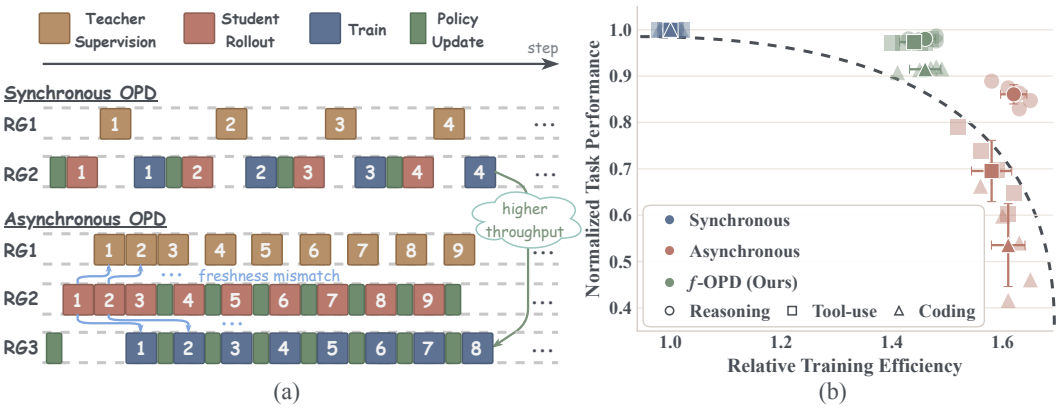


Figure 1: (a) System implementation of OPD under synchronous (top) and asynchronous (bottom) execution. While asynchronous OPD improves training throughput by pipelining computation across multiple resource groups (RGs), the resulting freshness mismatch compromises training fidelity. (b) Performance–efficiency trade-offs across reasoning, tool-use, and coding-agent tasks. Points denote five-seed means with standard deviations. While vanilla OPD executions struggle to simultaneously balance performance and efficiency, f-OPD effectively bridges this gap by freshness-aware control.

Despite the fact that freshness mismatch has been well studied in asynchronous RL [11–14], its implications for OPD remain underexplored due to asynchronous OPD execution extending beyond the conventional decoupling of data collection and optimization. An intuitive way is to periodically refresh the replay buffer[2] maintained during training, preventing stale data from continuously corrupting policy updates. Yet such a hard refresh introduces its own dilemma: refreshing too aggressively discards potentially useful learning signals, while refreshing too conservatively allows stale optimization to accumulate. This tension suggests that staleness should not be treated as a buffer-level binary (pick or drop) trigger for refreshing, but instead be quantified at the level of individual training samples. A natural online proxy for the staleness is policy update lag, which denotes the number of optimization steps elapsed since a sample was generated. However, lag alone is insufficient in the OPD: samples with identical ages can exhibit vastly different mismatches due to heterogeneous drift from both the student rollout policy and teacher supervision.

In this paper, we formulate the freshness mismatch in OPD as an objective discrepancy between asynchronous execution and its ideal on-policy counterpart. Through theoretical analysis, we show that this discrepancy is driven by two structurally distinct sources: *rollout drift*, arising from stale student policy occupancy, and *supervision drift*, arising from outdated teacher conditioning. Based on this, we derive a fine-grained characterization of sample-level staleness and introduce a **freshness score** that quantifies each sample's fidelity to the on-policy objective. We further propose ***f*-OPD**, a freshness-aware framework that compensates for objective discrepancy while actively correcting the policy drift induced by staleness. Our main contributions are summarized as follows:

- We formalize asynchronous OPD as optimization under freshness mismatch, and theoretically characterize the resulting objective discrepancy as arising jointly from both the student and teacher models. Based on this decomposition, we introduce a fine-grained metric for tracking sample-level fidelity to the on-policy distillation objective.
- We propose f-OPD, a freshness-aware OPD framework that bridges the gap between efficient asynchronous execution and ideal on-policy optimization.
- We conduct comprehensive experiments on a range of tasks, including reasoning, tool-use, and coding-agent tasks with progressively longer interaction horizons, and demonstrate that f-OPD consistently achieves a favorable balance between task quality and system throughput.

[2] A buffer of previously collected rollouts and their associated teacher supervision used for subsequent policy updates.

2 Related Work

LLM Post-training and On-Policy Distillation. Post-training has become the primary mechanism for eliciting task competence and interactive reasoning in LLMs. Existing paradigms exhibit complementary trade-offs. SFT [15] optimizes forward-KL objectives over fixed datasets, offering highly stable token-level supervision but limited exploration beyond the training distribution. RL [2, 3], in contrast, learns from self-generated trajectories with reward-driven objectives, enabling stronger exploration [16–18] at the cost of unstable optimization and potential policy collapse. Positioned between these two paradigms, OPD [19–22] combines on-policy learning with distribution-level teacher supervision, inheriting RL's exploratory capability while preserving the dense, stable learning signal of distillation. Recent on-policy self-distillation [23–26] and online teacher-supervised variants [27–33] further demonstrate the promise of this paradigm for LLM post-training. Despite this broader progress, a central challenge remains unresolved: maintaining the statistical fidelity of on-policy optimization while achieving the system efficiency required for large-scale OPD training.

Asynchronous Systems and Freshness Mismatch. Asynchronous learning systems are widely adopted to improve utilization by decoupling data collection and optimization. Asynchronous reinforcement learning systems [11, 12] have become a standard paradigm for scaling training by decoupling experience collection from policy optimization, enabling higher throughput, better resource utilization, and large-scale distributed execution. However, this decoupling also leads to stale trajectories when acting and learning evolve under inconsistent policy states. Similar staleness effects appear in distributed optimization and parallel training [34, 35]. In policy-gradient methods, this mismatch can already be viewed through the classical on-policy/off-policy tension underlying REINFORCE [36], and it becomes especially visible in modern LLM-RL, where recent analyses link training collapse directly to training-inference mismatch [13]. Prior work further shows that such a mismatch can compound across sequential rollouts, motivating drift-control principles such as dataset aggregation and trust-region optimization [37, 38, 2]. More recent long-horizon RL work sharpens this connection by studying explicit trust-region masking [39] and by showing that decoupled PPO can separate the proximal policy used to control update size from the behavior policy used for off-policy correction, thereby making better use of stale data [14]. Our setting extends this line of work to teacher-supervised OPD, where freshness mismatch arises jointly from stale student rollouts and outdated teacher supervision.

3 Theoretical Analysis

Although the freshness mismatch arises mainly from systems-level asynchrony, it can be analyzed as an objective-level discrepancy. In this section, we formalize the gap between the ideal on-policy distillation objective and the objective optimized under an asynchronous pipeline, and show how this discrepancy arises from both the student and the teacher.

3.1 Objective Discrepancy under Different OPD Training Modes

We begin by distinguishing training objectives under synchronous and asynchronous training pipelines. Let π_θ^t denote the student policy at update step t, and let $\pi_{\text{teacher}}(\cdot \mid x)$ denote the teacher distribution conditioned on prefix x. The ideal synchronous OPD objective is defined as

$$\mathcal{J}_{\text{sync}}(\pi_\theta^t) = \mathbb{E}_{x\sim d^t}\left[\ell\big(\pi_\theta^t(\cdot \mid x), \pi_{\text{teacher}}(\cdot \mid x)\big)\right], \tag{1}$$

where d^t denotes the mixture distribution over trajectory prefixes induced by rolling out π_θ^t across token positions, and d_h^t denotes the corresponding depth-h prefix distribution when depth-wise arguments are needed below. Under asynchronous execution, training proceeds on an active buffer $\mathcal{B}^t$ composed of samples generated and graded at current or earlier policy update steps. For each sample $i \in \mathcal{B}^t$, let $r(i) \le t$ denote the step at which the sample was produced, let x_i denote the stored prefix, and let $c_i^{r(i)}$ denote the teacher-conditioning context actually used when the sample was graded. The corresponding objective implemented by the asynchronous pipeline can be formulated as

$$\mathcal{J}_{\text{async}}(\pi_\theta^t) = \mathbb{E}_{i\sim\mathcal{B}^t}\left[\ell\big(\pi_\theta^t(\cdot \mid x_i), \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\big)\right]. \tag{2}$$

Since student rollout and teacher supervision may be conditioned on different information in the asynchronous pipeline, which is context-dependent, we distinguish them throughout this paper by

notation. Specifically, x denotes the student-side prediction prefix, while c denotes the teacher-side conditioning context used to generate supervision. In the simplest prefix-only setting, $c = x$, whereas in general $c \neq x$. Accordingly, x_i denotes the stored sample prefix, $c_i^{r(i)}$ the teacher context used at labeling time, and c_i^t the teacher context that would be used if the same sample were relabeled at update step t. We then define the asynchronous objective discrepancy at t as

$$\Delta^t = \left|\mathcal{J}_{\text{async}}(\pi_\theta^t) - \mathcal{J}_{\text{sync}}(\pi_\theta^t)\right|. \tag{3}$$

3.2 Two-Fold Decomposition of the Asynchronous Objective Discrepancy

Unlike reward-estimation bias in off-policy RL, Δ^t arises from optimizing over prefixes generated by older student policies and potentially, on teacher supervision tied to outdated conditioning contexts. We next theoretically decompose these two sources of discrepancy under the following assumption.

Assumption 3.1 (Truncation and regularity)**.** We evaluate the distillation loss on a finite support where both student and teacher probabilities are bounded away from zero by a positive constant. On this domain, $\ell(\cdot, \cdot)$ is nonnegative and Lipschitz with respect to total-variation distance.

To isolate the student- and teacher-sides of freshness mismatch, we first introduce an intermediate distillation objective that pairs stale student rollouts with up-to-date teacher conditioning contexts:

$$\widetilde{\mathcal{J}}_{\text{async}}(\pi_\theta^t) = \mathbb{E}_{x \sim d^{r(i)}}\left[\ell\big(\pi_\theta^t(\cdot \mid x), \pi_{\text{teacher}}(\cdot \mid x)\big)\right], \tag{4}$$

where $d^{r(i)}$ denote the prefix distribution of sample i induced by the stale policy. We then define the buffer-level stale distributions by $d^{\text{stale},t} := \mathbb{E}_{i \sim \mathcal{B}^t}[d^{r(i)}]$ and yield the following proposition for discrepancy decomposition.

Proposition 3.2 (Two-fold Decomposition for Asynchronous Objective Discrepancy)**.** *Under Assumption 3.1, there exist nonnegative constants C_{roll} and C_{sup} such that:*

$$\Delta^t \leq \underbrace{C_{\text{roll}}\, \text{TV}\big(d^t,\, d^{\text{stale},t}\big)}_{\textit{rollout drift}} + \underbrace{C_{\text{sup}}\, \mathbb{E}_{i \sim \mathcal{B}^t}\left[\text{TV}\Big(\pi_{\text{teacher}}(\cdot \mid c_i^t),\, \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\Big)\right]}_{\textit{supervision drift}}, \tag{5}$$

where $\text{TV}(\cdot, \cdot)$ denotes the total-variation distance. The detailed proof of decomposition is provided in the Appendix A.2.

By the convexity of $\text{TV}(\cdot, \cdot)$:

$$\text{TV}\big(d^t,\, d^{\text{stale},t}\big) \leq \mathbb{E}_{i \sim \mathcal{B}^t}\left[\text{TV}\Big(d^t,\, d^{r(i)}\Big)\right], \tag{6}$$

which recasts the rollout drift term in Eq. 5 into a buffer-level expectation to align with the supervision drift term. Thus, the objective discrepancy in asynchronous OPD is decomposed into two structurally distinct terms: ***rollout drift***, arising when the active buffer reflects prefixes drawn from the older student policy rather than the current policy; and ***supervision drift***, arising when teacher supervision is attached to outdated conditioning contexts that no longer align with the current contexts.

3.3 Diagnostics for Tracking Staleness

While the above decomposition identifies the sources of staleness and formally characterizes their contributions, the resulting drift terms are not directly observable during optimization. We therefore introduce sample-level diagnostics that empirically track them through the distributions of the student and teacher models. Specifically, for rollout drift, we use an on-support divergence evaluated on buffered prefixes:

$$\delta_i^{\text{roll}} = \text{TV}\Big(\pi_\theta^t(\cdot \mid x_i),\, \pi_\theta^{r(i)}(\cdot \mid x_i)\Big), \qquad D_i^{\text{roll}} = \text{KL}\Big(\pi_\theta^t(\cdot \mid x_i) \,\|\, \pi_\theta^{r(i)}(\cdot \mid x_i)\Big). \tag{7}$$

By Pinsker's inequality, $\delta_i^{\text{roll}} \leq \sqrt{\frac{1}{2}\, D_i^{\text{roll}}}$, so the empirical KL serves as a conservative upper bound on the corresponding total-variation distance. In sequential decision-making settings, however, even local policy drift can compound over time, amplifying stale occupancy. We provide analysis for this

effect in Proposition A.4. As for supervision drift, we can also derive an upper bound by Pinsker's inequality as (see Appendix A.5 for details):

$$\delta_i^{\text{sup}} = \text{TV}\Big(\pi_{\text{teacher}}(\cdot \mid c_i^t),\, \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\Big) \le \sqrt{\tfrac{1}{2}\, D_i^{\text{sup}}}, \tag{8}$$

where $D_i^{\text{sup}} = \text{KL}\Big(\pi_{\text{teacher}}(\cdot \mid c_i^t) \,\|\, \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\Big)$ denotes the divergence between the teacher distribution under the current and outdated contexts. In summary, D_i^{roll} and D_i^{sup} serve as monotone surrogates rather than exact estimators of the discrepancy terms, but provide observable signals that can be incorporated into freshness-aware control. Together with policy update lag, we illustrate in Figure 2 (top) three complementary diagnostics for characterizing sample-level staleness. These diagnostics form the foundation for incorporating freshness-aware control into OPD optimization.

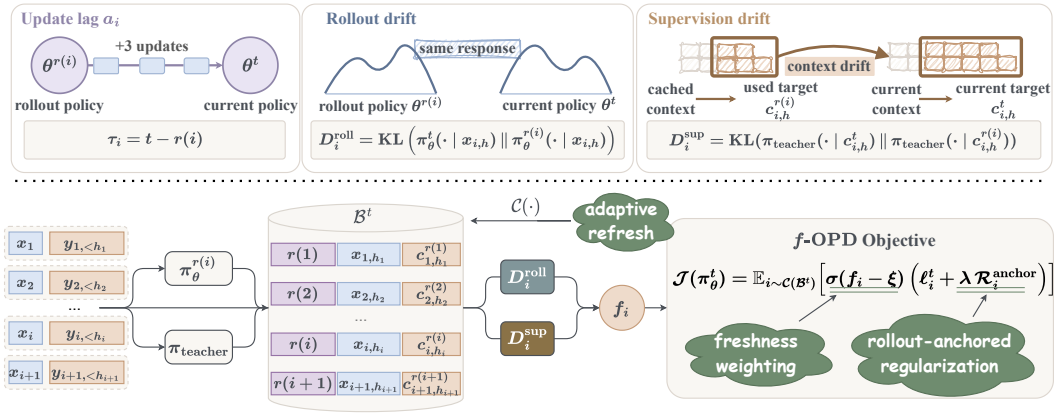


Figure 2: Systematic overview of f-OPD. *Top*: three sample-level diagnostics used to characterize staleness. *Bottom*: the overall f-OPD pipeline, where sample freshness is estimated from these diagnostics and integrated into OPD optimization through three complementary mechanisms.

4 Freshness-Aware Control for Long-Horizon OPD

Our analysis in Sec. 3 has decomposed asynchronous OPD staleness into rollout and supervision drift and introduced the corresponding diagnostics D_i^{roll} and D_i^{sup}. In this section, we define a sample-level freshness score based on these diagnostics, and further propose f-OPD with three complementary mechanisms for freshness-aware control.

4.1 Fine-Grained Sample Freshness Scoring

To evaluate how stale each buffered sample is, each buffer must retain sufficient provenance to replay the trajectory under both the current and the stale policies and to reconstruct the teacher-conditioning contexts used now and at grading time. For each sample i, we maintain its actual rollout step $r(i)$, the stored prefix x_i, and the token-level information $\{(y_{i,h},\, c_{i,h}^{r(i)})\}_{h=1}^{H_i}$, where H_i denotes the token length of sample i, $y_{i,h}$ is the token generated by student at position h, and $c_{i,h}^{r(i)}$ is the teacher-conditioning context attached to that token when the sample was graded. Following the previous introduction of the distinction between x and c, we now refine the sample-level notation from Sec. 3 by lifting x_i to token-level replay prefixes and c_i^t to token-level current teacher contexts, since replay-based comparison is performed only at aligned token positions. We likewise refine the stored prefix into token-level replay prefixes:

$$x_{i,h} := (x_i, y_{i,<h}), \tag{9}$$

so that the diagnostics and losses below are evaluated on aligned token positions and later aggregated back to the sample level. Let $\mathcal{U}_i^{\text{roll}}, \mathcal{U}_i^{\text{sup}} \subseteq \{1, \ldots, H_i\}$ denote token positions for which replay yields valid and comparable conditioning. We assume that the aligned sets are nonempty whenever a

sample is used to compute diagnostics or optimization losses. For each position, we use the replayed token prefix $x_{i,h}$ to compute token-level diagnostics:

$$\begin{cases} D_i^{\text{roll}} = \dfrac{1}{|\mathcal{U}_i^{\text{roll}}|} \displaystyle\sum_{h \in \mathcal{U}_i^{\text{roll}}} \mathrm{KL}\Big(\pi_\theta^t(\cdot \mid x_{i,h}) \,\|\, \pi_\theta^{r(i)}(\cdot \mid x_{i,h})\Big), \\ D_i^{\text{sup}} = \dfrac{1}{|\mathcal{U}_i^{\text{sup}}|} \displaystyle\sum_{h \in \mathcal{U}_i^{\text{sup}}} \mathrm{KL}\Big(\pi_{\text{teacher}}(\cdot \mid c_{i,h}^t) \,\|\, \pi_{\text{teacher}}(\cdot \mid c_{i,h}^{r(i)})\Big). \end{cases} \tag{10}$$

Moreover, each sample carries an update lag $\tau_i = t - r(i) \geq 0$, providing a coarse-grained online proxy (see Appendix A.3) for global freshness: older samples are generally less reliable and should eventually be discarded. The replay-based diagnostics D_i^{roll} and D_i^{sup} therefore complement this coarse age signal with finer-grained sample-specific drift information. At the same time, the inverse-age factor $(\tau_i + 1)^{-1}$ remains a cheap operational proxy for the budget over which rollout drift may have accumulated. We therefore describe the sample-level freshness of i at t with

$$f_i = \frac{1}{\tau_i + 1} \exp(-\widetilde{\Delta}_i^t), \;\; \text{where} \;\; \widetilde{\Delta}_i^t = \alpha \sqrt{D_i^{\text{roll}}} + \beta \sqrt{D_i^{\text{sup}}} \approx \Delta_i^t. \tag{11}$$

Coefficients $\alpha, \beta \geq 0$ map the two diagnostics onto a common operational scale. Importantly, the approximation $\widetilde{\Delta}_i^t$ to the discrepancy in Proposition 3.2 should be interpreted as a monotone surrogate that ranks samples by their predicted contribution to stale-objective bias, rather than as an exact estimator of that bias. The $\exp(\cdot)$ in freshness calculation thus smoothly and monotonically downweights high-staleness samples, reducing variance while preserving relative ordering.

4.2 f-OPD

Next, we provide a step-by-step derivation of f-OPD with freshness-aware control, starting from the vanilla sample-level distillation loss:

$$\ell_i^t = \frac{1}{|\mathcal{U}_i^{\text{sup}}|} \sum_{h \in \mathcal{U}_i^{\text{sup}}} \ell\big(\pi_\theta^t(\cdot \mid x_{i,h}), \pi_{\text{teacher}}(\cdot \mid c_{i,h}^t)\big). \tag{12}$$

Building on the freshness score defined in Eq. 11, we first introduce ***freshness weighting*** by scaling each sample's contribution to the distillation loss according to its fidelity to the on-policy objective:

$$\mathcal{L}_{\text{distill}}^t = \mathbb{E}_{i \sim \mathcal{B}^t}\Big[\sigma(f_i - \xi)\, \ell_i^t\Big], \tag{13}$$

where $\sigma(\cdot)$ is the ReLU function, so that a sample with the freshness score below the threshold ξ will be excluded from optimization. Weighting is a first layer of defense: it reduces further deviation caused by over-optimizing on stale samples. However, it does not explicitly constrain how far the current policy can deviate from the rollout policy associated with surviving samples, nor can it repair a mismatch already present in an active buffer that has become globally stale. Moreover, replay-based diagnostics are meaningful only at token positions where the stale and current trajectories remain alignable. When the active buffer has low mean freshness, or when the aligned-position coverage ratios $|\mathcal{U}_i^{\text{roll}}|/H_i$ and $|\mathcal{U}_i^{\text{sup}}|/H_i$ become too small, the buffer no longer provides sufficiently reliable supervision or drift estimates.

To address these limitations, we introduce two additional mechanisms. On the one hand, to enforce local consistency between the current and rollout policy, we incorporate ***rollout-anchored regularization*** into the objective. Specifically, the regularization term $\mathcal{R}_i^{\text{anchor}} = D_i^{\text{roll}}$ uses the rollout-drift diagnostic from Eq. 10 as an observable surrogate for the stale-occupancy mismatch analyzed in Sec. 3.3. Notably, $\mathcal{R}_i^{\text{anchor}}$ is also modulated by freshness weighting, since corrective regularization becomes less effective as sample fidelity deteriorates. On the other hand, we introduce ***adaptive refresh*** to monitor the aggregate freshness score and alignment statistics of the active buffer:

$$\bar{f}^t = \frac{1}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} f_i, \quad \bar{\mathcal{M}}^{t,\text{roll}} = \frac{1}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} \frac{|\mathcal{U}_i^{\text{roll}}|}{H_i}, \quad \bar{\mathcal{M}}^{t,\text{sup}} = \frac{1}{|\mathcal{B}^t|} \sum_{i \in \mathcal{B}^t} \frac{|\mathcal{U}_i^{\text{sup}}|}{H_i}. \tag{14}$$

The buffer $\mathcal{B}^t$ is refreshed whenever it no longer satisfies the required freshness or alignment conditions, either due to degraded sample-mean freshness or insufficient aligned-support coverage:

$$\bar{f}^t \leq \kappa_f, \qquad \text{or} \qquad \bar{\mathcal{M}}^{t,\text{roll}} < \kappa_{\text{roll}}, \qquad \text{or} \qquad \bar{\mathcal{M}}^{t,\text{sup}} < \kappa_{\text{sup}}, \tag{15}$$

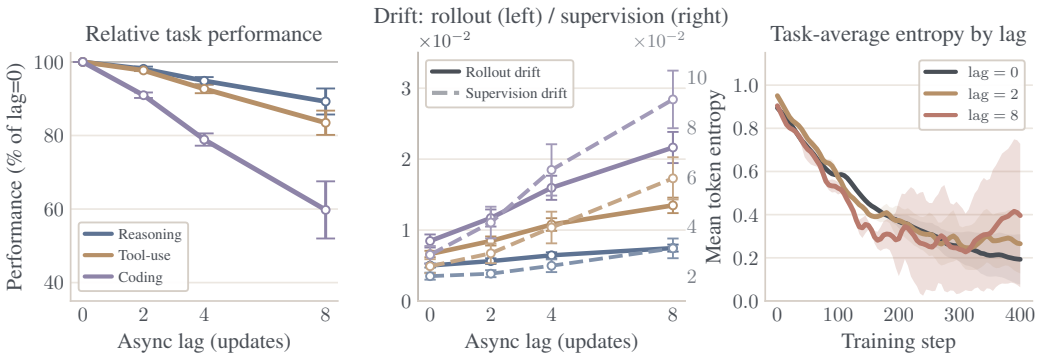


Figure 3: Failure modes of vanilla OPD under increasing policy update lag. (a) Relative task performance degradation with lags. (b) Drift diagnostics: rollout drift on the left axis, and supervision drift on the right axis, both increasing with lag. (c) Token-level entropy dynamics, averaged uniformly over three tasks, for lag 0, 2, and 8 across training. Markers and curves show five-seed means; error bars and shaded bands denote a standard deviation.

where $\kappa_f, \kappa_{\text{roll}}, \kappa_{\text{sup}}$ denote the corresponding thresholds.

Combining the above three mechanisms, we propose f-OPD with a freshness-aware objective:

$$\mathcal{J}(\pi_\theta^t) = \mathbb{E}_{i\sim\mathcal{C}(\mathcal{B}^t)}\Big[\sigma(f_i - \xi)\left(\ell_i^t + \lambda\,\mathcal{R}_i^{\text{anchor}}\right)\Big], \tag{16}$$

where λ adjusts the strength of rollout-anchored regularization, and $\mathcal{C}(\mathcal{B}^t)$ denotes the refresh mechanism of renewing the active buffer to restore a more reliable supervision stream and more informative drift diagnostics for f-OPD. We provide pseudocode in Appendix Algorithm 1, which summarizes the complete procedure in an implementation-oriented form.

5 Experiments

Our experiments highlight the challenge of balancing performance and efficiency in long-horizon OPD training, and demonstrate how the proposed f-OPD effectively navigates this trade-off with freshness-aware control.

5.1 Settings

We summarize the key experimental settings from the following three perspectives.

Protocols. We follow the student-teacher distillation setup introduced by Thinking Machines Lab [40]. For model selection across tasks, we use Qwen family models as student models, trained under frozen supervision from larger Qwen teacher models. Training is conducted under both synchronous and asynchronous pipelines.

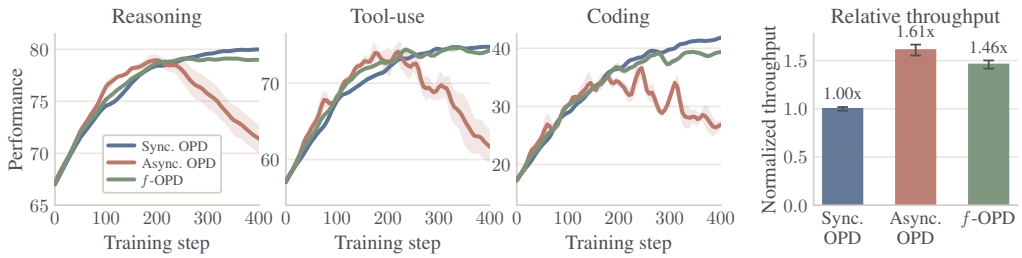


Figure 4: (a–c) Training dynamics across tasks for synchronous OPD, asynchronous OPD, and f-OPD. (d) Relative training throughput across tasks.

Task taxonomies. We consider three task families with increasing interaction horizon: (i) short-horizon reasoning, where supervision is effectively single-shot; (ii) medium-horizon tool-use, where intermediate actions influence subsequent observations and supervision; and (iii) long-horizon coding agent, where errors compound across iterative edits and executions. DAPO-Math-17K [41] ($\sim$17K samples) is adopted as the underlying prompt distribution for both short-horizon reasoning and medium-horizon tool-use. In the tool-use setting, those problems are wrapped in a deterministic ReTool-style code-interpreter harness [42]. For long-horizon coding-agent tasks, we use mini-SWE-agent [43] as the coding-agent scaffold on real repository-level software-engineering tasks.

Evaluations. We evaluate short-horizon reasoning on MATH500 [44], tool-use on a held-out olympiad-style set of 1,000 problems executed in the same harness, and coding agents on SWE-bench Verified [45]. For short-horizon reasoning, we report *Avg@1 accuracy*. For tool-use, we report *Avg@4 accuracy* under the ReTool-style harness, together with *invalid tool rate* (the fraction of episodes containing at least one invalid code-interpreter invocation) and *peak-final drop* (the drop in resolve rate from the peak checkpoint to the final checkpoint). For coding agents, we report the SWE-bench Verified *resolve rate* under a single-run, single-patch evaluation protocol, along with repeated-loop, premature-stop, and *post-patch regression* rates (the performance of the patch drops from peak to final training). Beyond model performance, we measure end-to-end training throughput as the number of completed training samples per wall-clock hour to quantify system efficiency. Throughput is normalized to vanilla synchronous OPD, which serves as the baseline (i.e., Sync OPD $= 1.00\times$). All reported metrics are averaged over five random seeds. More details about the experimental setting and implementation are provided in Appendix B.1.

5.2 Failure Analysis of Vanilla OPD

Before evaluating the effectiveness of f-OPD, we first conduct a systematic failure analysis of vanilla OPD. Specifically, we investigate two questions: (i) *how does increasing policy lag affect task performance under OPD training?* and (ii) *which factors are responsible for this degradation under increasing policy lag?* To answer these questions, we evaluate four OPD variants across three tasks with manually controlled policy update lags of 0, 2, 4, and 8, spanning conditions from fully on-policy optimization to increasing asynchrony.

As shown in Figure 3(a), all three tasks exhibit a clear and monotonic performance degradation as policy lag increases, confirming that policy staleness is inherently harmful to asynchronous OPD fidelity. Moreover, this degradation becomes substantially more severe as the interaction horizon grows. Particularly for the coding-agent task, performance drops by roughly 40% under a lag of 8. Figure 3(b) further reports the KL-based diagnostics of rollout and supervision drift. Both drifts grow systematically with policy lag and interaction horizon, and supervision drift scales more steeply with lag than rollout drift, highlighting the importance of controlling teacher-context staleness in long-horizon OPD. Interestingly, Figure 3(c) shows that, unlike synchronous training, asynchronous OPD exhibits progressively less stable task-averaged entropy dynamics as lag increases. The highest-lag setting ($\mathrm{lag} = 8$, red) develops the widest late-training uncertainty band, indicating that some runs retain high-entropy exploration whereas others move toward partial collapse. This suggests that policy drift under asynchronous execution not only amplifies objective mismatch but also destabilizes generation dynamics, making entropy a practical early-warning signal for training instability.

5.3 Main Results of f-OPD

Our main results are organized into two parts. We first benchmark f-OPD against synchronous and asynchronous OPD baselines across the three tasks, measuring task quality (evaluation accuracy along training) and relative system throughput (completed training samples per wall-clock hour, normalized to synchronous OPD). Fig. 4 shows that f-OPD inherits the task quality of synchronous OPD and much of the throughput advantage of asynchronous OPD simultaneously, recovering the on-policy fidelity that vanilla asynchronous mode sacrifices without paying the full throughput cost of synchronous execution. Additionally, as shown in Appendix Figure 5, f-OPD maintains entropy dynamics comparable to synchronous training, demonstrating the stability brought by freshness-aware control.

To assess f-OPD on the most demanding setting, we compare it against a broad spectrum of post-training methods on the long-horizon coding-agent task: SFT, RL, and two asynchronous OPD

Table 1: Comparison of f-OPD against representative post-training methods on the coding-agent task [45]. Relative throughput is normalized to synchronous OPD.

Method	Scale	Throughput ↑	Resolve ↑	Post-patch reg. ↓	Collapse ratio ↓
Supervised Fine-tuning					
Vanilla SFT	8B	2.18×	39.1	—	0/5
SWE-Gym* [46]	32B	—	20.6	—	—
Reinforcement Learning					
Vanilla GRPO	8B	0.92×	40.2	—	—
SA-SWE* [47]	32B	—	39.4	—	—
DeepSWE* [48]	32B	—	36.4	—	—
On-Policy Distillation					
Qwen3-Coder† [49]	30B	—	51.6	—	—
Sync. OPD	8B	1.00×	41.8	1.6	0/5
Async. OPD	8B	1.61×	26.8	7.4	3/5
Async. OPD + KL	8B	1.60×	28.2	7.6	3/5
Async. OPD + clipped IW	8B	1.60×	29.9	7.2	2/5
Async. OPD + hard refresh	8B	1.15×	35.1	6.6	1/5
Async. OPD + lag-only weighting	8B	1.54×	34.9	7.6	2/5
***f*-OPD (ours)**	8B	**1.46×**	**39.4**	2.6	0/5

* Results reported from publicly available sources. — Results not available. † Teacher model used for OPD.

baselines with simplified freshness-aware control—one applying hard refresh to buffered samples, the other scoring sample freshness only by policy update lag (see Appendix B.1 for details). The methods we reproduced share the same base model (Qwen3-8B). Results are reported in Table 1. SFT achieves the highest throughput but is bottlenecked by its reliance on high-quality offline trajectories, which are costly to scale in long-horizon agentic settings. RL is on-policy by design but suffers from low throughput due to long-tailed rollout latency. Synchronous OPD achieves the highest task quality among on-policy methods (41.8 resolve rate) but inherits the throughput penalty of synchronous execution. Vanilla asynchronous OPD delivers a $1.61\times$ throughput gain over synchronous OPD, but task quality collapses to a 26.8 resolve with a 60% collapse rate across runs. To examine the limitations of directly transferring trust-region methods from RL to OPD, we incorporate KL- and importance-weighting (IW)-based approaches into asynchronous OPD. Both yield only modest reductions in the collapse rate because the drift inherent in teacher supervision is overlooked. Additionally, simplified freshness-aware variants (hard refresh & lag-only weighting) also partially mitigate the degradation and remain well below the synchronous ceiling. In contrast, f-OPD attains a $1.46\times$ throughput gain while preserving 94% synchronous resolve and incurring only a 2.6-point post-patch regression, with no observed collapse. These results position f-OPD as a method that closes the performance–efficiency gap left open by both general-purpose post-training paradigms and vanilla asynchronous OPD variants.

5.4 Generalization and Compatibility

In addition to the effectiveness on tasks with different horizons, we evaluate the generalizability of f-OPD across model families, distillation settings, and objective variants.

First, we consider two non-Qwen student-teacher pairs: Llama-3.1-8B-Instruct/Llama-3.1-70B-Instruct, representing capacity-gap distillation, and DeepSeekMath-7B-Base/DeepSeekMath-7B-RL, representing same-scale capability transfer. As shown in Appendix Table 3, f-OPD consistently recovers the performance degradation induced by asynchronous execution, achieving results comparable to synchronous OPD across three mathematical reasoning benchmarks. These results demonstrate its generalizability beyond the Qwen family and across distinct distillation settings. On the other hand, freshness-aware control is orthogonal to the underlying distillation objective. We therefore integrate f-OPD with two alternative OPD methods, EOPD [32] and SOD [50], to test the compatibility. As shown in Appendix Table 4, f-OPD recovers most of the accuracy lost to asynchronous execution while incurring less than 10% throughput overhead over the corresponding asynchronous baselines.

Method	Tool-use		Coding agent		
	Acc.$_{\text{avg@4}}$	**Peak-final drop**	**Resolve**	**Post-patch regression**	**Collapse ratio**
Synchronous OPD	74.8	2.7	41.8	1.6	0/5
Incremental composition					
Asynchronous OPD	61.6	10.4	26.8	7.4	3/5
+ Freshness weighting	70.8	4.3	36.5	4.4	1/5
+ Rollout-anchored regularization	73.2	3.4	38.1	3.2	1/5
+ Adaptive refresh (f-OPD)	74.4	3.1	39.4	2.6	0/5
Leave-one-out					
***f*-OPD**	74.4	3.1	39.4	2.6	0/5
w/o Freshness weighting	69.8	3.6	37.4	3.5	3/5
w/o Rollout-anchored regularization	73.1	3.2	37.7	3.4	1/5
w/o Adaptive refresh	73.2	3.4	38.1	3.2	1/5

Table 2: Ablation of the three freshness-aware control mechanisms in f-OPD on tool-use and coding-agent tasks. The upper block incrementally adds each mechanism, while the lower block removes each component from the full f-OPD.

For example, on AIME24, it improves EOPD from 21.2 to 32.1, approaching the synchronous performance of 35.8.

5.5 Ablation Study

The effectiveness of f-OPD stems from the synergy of three principled mechanisms: *freshness weighting*, *rollout-anchored regularization*, and *adaptive refresh*. We conduct two complementary ablation studies to assess their individual contributions.

First, we incrementally introduce each component on top of vanilla asynchronous OPD and evaluate their cumulative effects on the tool-use and coding-agent tasks (Table 2). Specifically, freshness weighting recovers most of the lost task quality (+9.2 avg@4 accuracy, +9.7 coding resolve). It reduces the collapse ratio to 1/5, confirming that down-weighting stale samples directly counters both rollout and supervision drift. Rollout-anchored regularization further closes the gap by stabilizing per-update drift (+2.4 avg@4 accuracy, +1.6 coding resolve), while adaptive refresh delivers the final gains and eliminates collapse. Second, we conduct a leave-one-out ablation by removing each mechanism individually from the full f-OPD. The results further highlight the critical role of *freshness weighting*: removing it leads to the largest performance degradation, reducing tool-use accuracy from 74.4 to 69.8 and coding-agent Resolve from 39.4 to 37.4, while the collapse ratio increases from 0/5 to 3/5, matching that of vanilla asynchronous OPD. In contrast, removing either *rollout-anchored regularization* or *adaptive refresh* causes only a modest increase in the collapse ratio (from 0/5 to 1/5), while still resulting in consistent degradation across the performance metrics. These suggest that freshness weighting is particularly important for maintaining training stability, while the other two mechanisms provide complementary gains in overall performance.

Alternatively, we conduct a hyperparameter sensitivity analysis (details in Appendix C.3) by grid-searching over the drift coefficients (α, β) and refresh threshold (κ). While the best-performing configuration benefits from hyperparameter tuning, f-OPD maintains stable performance across a relatively broad range of values, demonstrating its robustness to hyperparameter choices.

6 Conclusion

In this paper, we systematically analyze the performance–efficiency trade-off of vanilla OPD under both synchronous and asynchronous execution. We identify the principal source of freshness mismatch in OPD from both the student rollout and the teacher context, and propose f-OPD to guide asynchronous optimization through freshness-aware control. Its effectiveness is established across tasks of varying interaction horizons, with the most pronounced gains on long-horizon coding agents. Looking ahead, we aim to scale f-OPD to larger foundation models and a broader spectrum of general-purpose agentic tasks.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (Grant Nos. PolyU25216423 and PolyU15217424), and The Hong Kong Polytechnic University (Project Nos. P0056503 and P0053699).

References

[1] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

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

[4] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[5] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. `https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf`, 2026. technical report.

[6] Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

[7] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[8] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

[9] Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. `https://github.com/THUDM/slime`, 2025. GitHub repository, accessed 2026-05-06.

[10] Nvidia. Nemo rl: A scalable and efficient post-training library. `https://github.com/NVIDIA-NeMo/RL`, 2025. GitHub repository.

[11] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[12] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[13] Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. When speed kills stability: Demystifying RL collapse from the training-inference mismatch. `https://richardli.xyz/rl-collapse`, September 2025. Research blog post, accessed 2026-05-06.

[14] Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022. Introduces decoupled PPO by separating the proximal policy for update control from the behavior policy for off-policy correction.

[15] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28, 2015.

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

[17] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

[18] Shimin Zhang, Xianwei Chen, Yufan Shen, Ziyuan Ye, and Jibin Wu. Relax: Reasoning with latent exploration for large reasoning models. *arXiv preprint arXiv:2512.07558*, 2025.

[19] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

[20] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models. In *International Conference on Learning Representations*, 2024.

[21] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

[22] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

[23] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[24] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

[25] Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. Opsdl: On-policy self-distillation for long-context language models. *arXiv preprint arXiv:2604.17535*, 2026.

[26] Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*, 2026.

[27] Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation. *arXiv preprint arXiv:2603.11137*, 2026.

[28] Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

[29] Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation. *arXiv preprint arXiv:2602.12125*, 2026.

[30] Yuqian Fu, Haohuan Huang, Kaiwen Jiang, Yuanheng Zhu, and Dongbin Zhao. Revisiting on-policy distillation: Empirical failure modes and simple fixes. *arXiv preprint arXiv:2603.25562*, 2026.

[31] Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, 2025.

[32] Woogyeol Jin, Taywon Min, Yongjin Yang, Swanand Ravindra Kadhe, Yi Zhou, Dennis Wei, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models. *arXiv preprint arXiv:2603.07079*, 2026. Also available on OpenReview as SPOT 2026.

[33] Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler II, Qian Qian, Gregory D. Lyng, Sanjit Singh Batra, and Robert E. Tillman. Fast and effective on-policy distillation from reasoning prefixes. *arXiv preprint arXiv:2602.15260*, 2026.

[34] Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. Staleness-aware Async-SGD for distributed deep learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[35] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. PipeDream: Fast and efficient pipeline parallel DNN training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019.

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

[37] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[38] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[39] Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, Qian Liu, and Baoxiang Wang. Trust region masking for long-horizon LLM reinforcement learning. *arXiv preprint arXiv:2512.23075*, 2025.

[40] Kevin Lu and Thinking Machines Lab. On-policy distillation. `https://thinkingmachines.ai/blog/on-policy-distillation/`, 2025. Thinking Machines Lab blog post, published 2025-10-27, accessed 2026-05-06.

[41] BytedTsinghua-SIA. DAPO-Math-17k. `https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k`, 2025. Hugging Face dataset repository, accessed 2026-05-06.

[42] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. ReTool: Reinforcement learning for strategic tool use in LLMs. *arXiv preprint arXiv:2504.11536*, 2025.

[43] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL `https://arxiv.org/abs/2405.15793`. Recommended citation for mini-SWE-agent from the project repository.

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

[45] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

[46] Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. arXiv:2412.21139.

[47] Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025.

[48] Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Colin Cai, Tarun Venkat, Manan Roongta, Li Erran Li, Raluca Ada Popa, Koushik Sen, Ion Stoica, Ameen Patel, Qingyang Wu, Alpay Ariyak, Shang Zhu, Ben Athiwaratkun, and Ce Zhang. Deepswe: Training a fully open-sourced, state-of-the-art coding agent by scaling rl. `https://pretty-radio-b75.notion.site/DeepSWE-Training-a-Fully-Open-sourced-State-of-the-Art-Coding-Agent-by-Scaling-RL-22281902c1468193aabbe9a8c59bbe33`, 2025. Technical blog post, accessed 2026-05-06.

[49] Ruisheng Cao, Mouxiang Chen, Jiawei Chen, Zeyu Cui, Yunlong Feng, Binyuan Hui, Yuheng Jing, Kaixin Li, Mingze Li, Junyang Lin, et al. Qwen3-coder-next technical report. *arXiv preprint arXiv:2603.00729*, 2026.

[50] Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. Sod: Step-wise on-policy distillation for small language model agents. *arXiv preprint arXiv:2605.07725*, 2026.

[51] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

[53] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024.

[54] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2024.

[55] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[56] John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=63iVrXc8cC`. Datasets and Benchmarks Track Spotlight.

[57] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[58] Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pages 94743–94765, 2025.

A Additional Results for Theoretical Analysis

In this section, we provide the formal assumptions and proof sketches that support the theoretical analysis in Sec. 3 on objective discrepancy between synchronous and asynchronous OPD.

A.1 Alternative Assumptions

We reuse the notation from Sections 3 and 4. The current student policy at update t is π_θ^t, the rollout step attached to sample i is $r(i)$, and the active stale buffer is $\mathcal{B}^t$. The ideal policy update objective is

$$\mathcal{J}_{\mathrm{sync}}(\pi_\theta^t) = \mathbb{E}_{x\sim d^t}\left[\ell\big(\pi_\theta^t(\cdot \mid x), \pi_{\mathrm{teacher}}(\cdot \mid x)\big)\right],$$

where d^t denotes the mixture prefix distribution from Sec. 3. The stale-buffer objective is

$$\mathcal{J}_{\mathrm{async}}(\pi_\theta^t) = \mathbb{E}_{i\sim\mathcal{B}^t}\left[\ell\big(\pi_\theta^t(\cdot \mid x_i), \pi_{\mathrm{teacher}}(\cdot \mid c_i^{r(i)})\big)\right].$$

The discrepancy of interest is

$$\Delta^t = \left|\mathcal{J}_{\mathrm{async}}(\pi_\theta^t) - \mathcal{J}_{\mathrm{sync}}(\pi_\theta^t)\right|.$$

For the subsequent proof sketches, we first supplement the following assumptions.

Assumption A.1 (Total-variation stability of the loss)**.** On the truncated support from Assumption 3.1, there exist constants $L_\pi, L_q \geq 0$ such that for any predictive distributions p, p', q, q',

$$|\ell(p,q) - \ell(p',q)| \leq L_\pi\,\mathrm{TV}(p,p') \qquad \text{and} \qquad |\ell(p,q) - \ell(p,q')| \leq L_q\,\mathrm{TV}(q,q').$$

Assumption A.2 (Bounded one-step rollout drift on stale support)**.** For every update step s and every stale training unit x represented in the active buffer,

$$\mathrm{TV}\big(\pi_\theta^{s+1}(\cdot \mid x), \pi_\theta^s(\cdot \mid x)\big) \leq \varepsilon_s(x).$$

Assumptions 3.1, A.1, and A.2 are stronger than the unconstrained LLM setting and are introduced only to make a discrepancy analysis finite and interpretable. The resulting bounds should therefore be read as conservative control justifications rather than a complete optimization theory for asynchronous OPD.

A.2 Sketch of the Objective Discrepancy Decomposition

In this section, we sketch the argument behind Proposition 3.2.

Proof. Recall the intermediate objective defined in Eq. 4

$$\widetilde{\mathcal{J}}_{\mathrm{async}}(\pi_\theta^t) = \mathbb{E}_{x\sim d^{\mathrm{stale},t}}\left[\ell\big(\pi_\theta^t(\cdot \mid x), \pi_{\mathrm{teacher}}(\cdot \mid x)\big)\right],$$

where $d^{\mathrm{stale},t}$ denotes the mixture stale-prefix distribution induced by the active buffer. Then we derive

$$\Delta^t \leq \left|\mathcal{J}_{\mathrm{async}}(\pi_\theta^t) - \widetilde{\mathcal{J}}_{\mathrm{async}}(\pi_\theta^t)\right| + \left|\widetilde{\mathcal{J}}_{\mathrm{async}}(\pi_\theta^t) - \mathcal{J}_{\mathrm{sync}}(\pi_\theta^t)\right|.$$

The first term is a pure supervision-mismatch quantity. By Assumption A.1,

$$\left|\mathcal{J}_{\mathrm{async}}(\pi_\theta^t) - \widetilde{\mathcal{J}}_{\mathrm{async}}(\pi_\theta^t)\right| \leq L_q\,\mathbb{E}_{i\sim\mathcal{B}^t}\left[\mathrm{TV}\Big(\pi_{\mathrm{teacher}}(\cdot \mid c_i^t), \pi_{\mathrm{teacher}}(\cdot \mid c_i^{r(i)})\Big)\right].$$

The second term depends only on replacing the stale distributional occupancy with the current occupancy, while keeping the current policy and labels fixed. Because the loss is bounded on the truncated support, standard change-of-measure arguments give

$$\left|\widetilde{\mathcal{J}}_{\mathrm{async}}(\pi_\theta^t) - \mathcal{J}_{\mathrm{sync}}(\pi_\theta^t)\right| \leq L_{\mathrm{occ}}\,\mathrm{TV}(d^t, d^{\mathrm{stale},t}),$$

for some constant L_{occ} depending only on the bounded support and loss normalization. Writing $d^{\mathrm{stale},t} = \mathbb{E}_{i\sim\mathcal{B}^t}[d^{r(i)}]$, convexity of total variation further gives

$$\mathrm{TV}(d^t, d^{\mathrm{stale},t}) \leq \mathbb{E}_{i\sim\mathcal{B}^t}\left[\mathrm{TV}(d^t, d^{r(i)})\right].$$

This is why the student-side discrepancy may also be interpreted as the buffer-average staleness induced by the sample-specific rollout steps. Combining the two inequalities yields Proposition 3.2. □

The consequence is important: the theoretical discrepancy is a two-term statement—rollout drift and supervision drift.

A.3 Relationship between Policy Update Lag and Rollout Drift

The following lemma shows that, under a mild step-wise smoothness assumption, the per-sample rollout drift can be upper-bounded by the number of update steps separating the rollout from the current student. This integer is essentially free to track during training.

Lemma A.3 (Lag bounds rollout drift)**.** *Under Assumption A.2, for every buffered sample $i \in \mathcal{B}^t$,*

$$\delta_i^{\text{rollout}} \;\le\; \sum_{s=r(i)}^{t-1} \varepsilon_s(x_i),$$

where the right-hand side accumulates over exactly $\tau_i := t - r(i)$ steps.

Proof. By the triangle inequality for total variation,

$$\delta_i^{\text{rollout}} = \mathrm{TV}\Big(\pi_\theta^t(\cdot \mid x_i),\, \pi_\theta^{r(i)}(\cdot \mid x_i)\Big) \le \sum_{s=r(i)}^{t-1} \mathrm{TV}\big(\pi_\theta^{s+1}(\cdot \mid x_i),\, \pi_\theta^{s}(\cdot \mid x_i)\big)\,.$$

Applying Assumption A.2 to each summand yields the claim. □

Update lag is not a divergence per se, but a budget: it bounds rather than measures policy drift, with the bound becoming looser as per-step errors accumulate. Hence, τ_i is incorporated in our computation of the sample-level freshness score in Eq. 11.

A.4 Rollout Drift Compounds with Horizon

We further analyze how rollout drift scales with horizon length, which is particularly important for practical OPD deployment in long-horizon agentic scenarios.

Proposition A.4 (Horizon compounding of rollout staleness)**.** *Consider a finite-horizon setting with horizon H. Suppose that perturbing the policy on a prefix by total variation at most δ perturbs the next-step prefix distribution by at most $L\delta$. Then there exists a constant $K_H = O(HL)$ such that*

$$\mathrm{TV}\big(d^t,\, d^{\text{stale},t}\big) \le K_H\, \mathbb{E}_{i\sim\mathcal{B}^t}\big[\delta_i^{\text{roll}}\big] \le K_H\, \mathbb{E}_{i\sim\mathcal{B}^t}\left[\sqrt{\tfrac{1}{2}\, D_i^{\text{roll}}}\right].$$

The main-text horizon proposition can be interpreted as a sequential perturbation bound on the mixture prefix distributions induced by a finite-horizon rollout. Let d^π denote the resulting mixture distribution over prefixes under policy π. Suppose that replacing one policy by another changes the next-step prefix distribution by at most L times the total-variation distance between the two policies on the current prefix. Then, by recursively unrolling the perturbation through the trajectory,

$$\mathrm{TV}\big(d^t, d^{\text{stale},t}\big) \le K_H\, \mathbb{E}_{i\sim\mathcal{B}^t}\big[\delta_i^{\text{roll}}\big]$$

for some constant $K_H = O(HL)$ depending on horizon and the one-step sensitivity constant. This is the same compounding mechanism familiar from sequential prediction and imitation learning: small early action differences alter later prefixes, which in turn alter the observations, tool outputs, and teacher-conditioning contexts seen downstream.

In asynchronous OPD, this means that rollout drift and supervision drift need not be independent at long horizons; stale early actions can create the context drift that later enlarges D_i^{sup}.

A.5 Additional Analysis for Supervision Drift

The teacher-side staleness is defined in the sample-level shorthand from Sec. 3 by

$$D_i^{\text{sup}} = \mathrm{KL}\Big(\pi_{\text{teacher}}(\cdot \mid c_i^t) \,\|\, \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\Big)\,.$$

In Sec. 4, this shorthand is instantiated by averaging token-level teacher comparisons over aligned positions. Here $\pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})$ is the teacher distribution actually cached and consumed during training. In contrast, $\pi_{\text{teacher}}(\cdot \mid c_i^t)$ is the teacher distribution obtained by relabeling the same training unit under the current context. The key point is that this term measures supervision drift, not teacher-parameter drift. It is enough for the effective teacher to change the context.

Algorithm 1: f-OPD training step. The algorithm first measures per-sample freshness from replay-based rollout and supervision diagnostics, then performs buffer-level refresh if aggregate freshness or aligned support degrades, and finally optimizes the freshness-weighted, rollout-anchored objective on the surviving samples.

Input: active buffer $\mathcal{B}^t$; current student π_θ^t; stale rollout checkpoints $\{\pi_\theta^{r(i)}\}$; teacher model π_{teacher}; thresholds $\xi, \kappa_f, \kappa_{\text{roll}}, \kappa_{\text{sup}}$; weights α, β, λ

foreach $i \in \mathcal{B}^t$ **do**
- replay sample i on aligned token positions $\mathcal{U}_i^{\text{roll}}, \mathcal{U}_i^{\text{sup}}$;
- **if** $|\mathcal{U}_i^{\text{roll}}| = 0$ ***or*** $|\mathcal{U}_i^{\text{sup}}| = 0$ **then**
  - set $f_i = 0$, $m_i^{\text{roll}} = 0$, and $m_i^{\text{sup}} = 0$;
  - mark sample i invalid for optimization and continue;
- compute rollout drift D_i^{roll} and supervision drift D_i^{sup} by Eq. 10;
- compute lag $\tau_i = t - r(i)$ and freshness score $f_i = (\tau_i + 1)^{-1} \exp(-\alpha\sqrt{D_i^{\text{roll}}} - \beta\sqrt{D_i^{\text{sup}}})$;
- compute alignment ratios $m_i^{\text{roll}} = |\mathcal{U}_i^{\text{roll}}|/H_i$ and $m_i^{\text{sup}} = |\mathcal{U}_i^{\text{sup}}|/H_i$;

aggregate $\bar{f}^t = |\mathcal{B}^t|^{-1} \sum_{i \in \mathcal{B}^t} f_i$, $\bar{\mathcal{M}}^{t,\text{roll}} = |\mathcal{B}^t|^{-1} \sum_i m_i^{\text{roll}}$, and $\bar{\mathcal{M}}^{t,\text{sup}} = |\mathcal{B}^t|^{-1} \sum_i m_i^{\text{sup}}$;

if $\bar{f}^t \le \kappa_f$ ***or*** $\bar{\mathcal{M}}^{t,\text{roll}} < \kappa_{\text{roll}}$ ***or*** $\bar{\mathcal{M}}^{t,\text{sup}} < \kappa_{\text{sup}}$ **then**
- refresh $\mathcal{B}^t$ with newly rolled out and relabeled samples;
- recompute $\{D_i^{\text{roll}}, D_i^{\text{sup}}, f_i\}_{i \in \mathcal{B}^t}$ on the refreshed buffer;

foreach *minibatch* $\mathcal{M} \subset \mathcal{B}^t$ **do**
- form freshness weights $w_i = \sigma(f_i - \xi)$ for $i \in \mathcal{M}$;
- form aggregated current-context loss ℓ_i^t by averaging $\ell\big(\pi_\theta^t(\cdot \mid x_{i,h}), \pi_{\text{teacher}}(\cdot \mid c_{i,h}^t)\big)$ over $h \in \mathcal{U}_i^{\text{sup}}$;
- form anchored loss $\mathcal{L}_\mathcal{M} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} w_i \left[\ell_i^t + \lambda D_i^{\text{roll}}\right]$;
- update student parameters $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\mathcal{M}$;

Output: updated student π_θ^{t+1} and refreshed metadata for the next asynchronous step

Proposition A.5 (Context mismatch perturbs the objective)**.** *Under Assumption A.1, the loss perturbation induced by stale teacher labels satisfies*

$$\begin{aligned}&\left|\ell\big(\pi_\theta^t(\cdot \mid x_i), \pi_{\text{teacher}}(\cdot \mid c_i^t)\big) - \ell\big(\pi_\theta^t(\cdot \mid x_i), \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\big)\right| \\ &\le L_q \,\mathrm{TV}\Big(\pi_{\text{teacher}}(\cdot \mid c_i^t), \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)})\Big),\end{aligned}$$

and therefore the buffer-averaged supervision drift is bounded by a constant times the mean supervision-side total variation. By Pinsker, this is, in turn, controlled by the square root of the mean supervision KL.

This proposition is the formal justification for including D_i^{sup} in the sample-level freshness score. The quantity need not be large in every setting. Still, whenever teacher outputs depend on the mutable trajectory context, a stale label can shift the implemented objective away from the supervision attached to the current trajectory.

Remark A.6 (When the supervision gap is negligible). If the teacher is fixed, the effective context is unchanged, and labeling is deterministic, then

$$\pi_{\text{teacher}}(\cdot \mid c_i^t) = \pi_{\text{teacher}}(\cdot \mid c_i^{r(i)}) \qquad \Longrightarrow \qquad D_i^{\text{sup}} = 0.$$

More generally, if the teacher distribution is stable under small context perturbations, then D_i^{sup} remains small. In those situations, the freshness framework naturally reduces to control for update lag and rollout drift.

B Additional Experimental Details

B.1 Settings

Benchmark inventory and splits The paper is organized around three task families: short-horizon mathematical reasoning, medium-horizon tool-use reasoning, and long-horizon coding-agent issue resolution. The short-horizon benchmark uses MATH500 [44] together with a held-out 1,200-problem

olympiad-style pool constructed from recent public contest problems; 200 problems are reserved as a pilot split for threshold selection, and 1,000 are used for the reported evaluation tables. The tool-use benchmark uses that same 1,000-problem report split, but each problem is wrapped in a deterministic code-interpreter environment following the ReTool formulation of interleaved reasoning and tool execution [42]. For coding, training trajectories are drawn from the mini-SWE agent, while the final coding evaluation uses 300 SWE-bench Verified issues [45]; 50 issues form a pilot split, and 250 form the reported evaluation set. We reference Codex-style evaluation [51], early program-synthesis benchmarks [52], AgentBench [53], WebArena [54], and VisualWebArena [55] only as neighboring evaluation settings rather than as the paper's primary evidence.

Details for the tool-use and coding-agent environment The tool-use environment follows ReTool [42] and exposes a single deterministic code-interpreter tool that can be interleaved with natural-language reasoning. Each tool invocation is a structured Python execution request; the harness returns the execution status along with stdout and stderr, and that observation is appended to the running dialogue state before the next student action. An invalid tool call is any malformed tool block, an unparsable execution payload, an unsupported invocation format, or a tool response that cannot be re-integrated into the active context. Each episode allows at most six tool calls and a maximum of ten total assistant turns.

The coding-agent setup adopts mini-swe-agent as the scaffold [56], and the final benchmarked runs are evaluated on SWE-bench Verified instances [45]. For each evaluation instance, the agent receives a repository snapshot before the human fix, the issue description, and the associated test-facing sandbox, then interacts with the repository through an edit-execute-observe loop within an isolated, containerized environment. We use the same internal edit-execute-observe scaffold training and evaluation, rather than switching to a separate leaderboard-specific scaffold; only the repository instances and the final grading harness differ across the two stages. The final patch is graded using the Docker-based SWE-bench evaluation procedure: it is applied to the repository state for that instance, then checked against the corresponding test suite.

Computation resources The reported compute budget is about 16k H100-80GB GPU-hours in total: 3k for reasoning, 5k for tool-use, and 8k for coding. Reasoning and tool-use runs use 16 H100-80GB GPUs, while coding runs use 24 H100-80GB GPUs.

Baselines Unless otherwise stated, the method comparisons below use the hardest asynchronous setting, $\mathrm{lag} = 8$, because that is where freshness mismatch is most visible. We use the following names throughout the section. **Sync. OPD** is the fully synchronized on-policy baseline. **Async. OPD** is the bare overlapped pipeline with no freshness-aware control. **Async. OPD + fixed refresh** keeps the bare asynchronous objective but deterministically refreshes the active buffer every 10 optimizer steps. **Async. OPD + lag-only freshness weighting** replaces the full freshness score with

$$f_i^{\text{lag-only}} = (\tau_i + 1)^{-1}, \qquad w_i = \sigma(f_i^{\text{lag-only}} - \xi),$$

so it tests whether update lag alone is an adequate surrogate for freshness. To evaluate the effect of directly extending trust-region methods from RL to OPD, we implement **Async. OPD + KL** (KL-based trust-region rollout anchoring) and **Async. OPD + clipped IW** (clipped importance weighting) to correct for student policy drift.

C Additional Results

C.1 Entropy dynamics across lags

Fig. 5 supplements Fig. 1 with task-wise entropy dynamics across different lags. We can see that reasoning remains comparatively stable even as lag increases, while tool-use and coding agent progressively develop wider late-training uncertainty bands. The upper side of those bands expands especially quickly at larger lag, indicating that some runs retain high-entropy exploration, whereas others partially collapse.

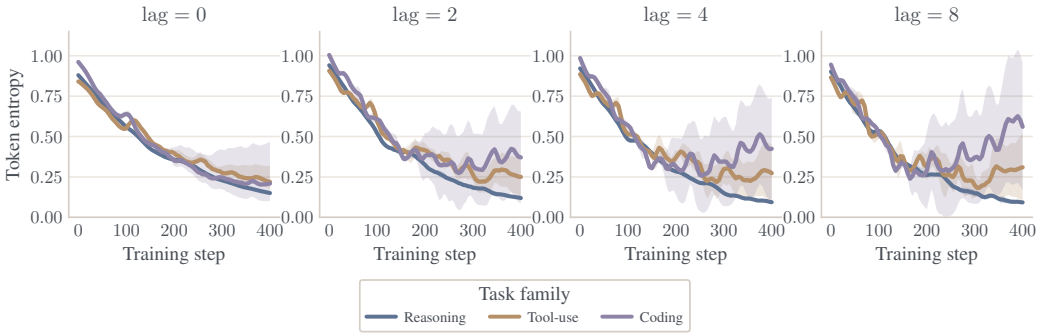


Figure 5: Supplementary entropy dynamics for three evaluated tasks across 4 different lags. Lines show five-seed means, and shaded regions show standard deviation.

Table 3: Avg@1 accuracy (%) on non-Qwen student–teacher pairs.

Student	Teacher	Method	GSM8K	MATH500	AMC23
Llama-3.1-8B-Instruct	Llama-3.1-70B-Instruct	Sync. OPD	79.2	23.0	15.0
		Async. OPD	63.8	16.1	7.5
		f-OPD	78.1	22.7	15.6
DeepSeekMath-7B-Base	DeepSeekMath-7B-RL	Sync. OPD	78.5	39.6	20.0
		Async. OPD	61.2	27.4	11.8
		f-OPD	79.0	40.2	19.4

C.2 Generalization Beyond Qwen Models and Vanilla OPD

Non-Qwen model families To assess generalization beyond Qwen, we evaluate two non-Qwen student–teacher pairs under complementary settings: (i) capacity-gap distillation with Llama-3.1-8B-Instruct/Llama-3.1-70B-Instruct [57], and (ii) same-scale capability transfer with DeepSeekMath-7B-Base/DeepSeekMath-7B-RL [3]. We evaluate both pairs on GSM8K [58], MATH500 [44], and AMC23 using Avg@1 accuracy. As shown in Table 3, asynchronous OPD substantially degrades performance in both settings, while f-OPD consistently recovers most of the loss and achieves performance comparable to synchronous OPD across all three benchmarks.

Compatibility with OPD variants To assess compatibility with different OPD objectives, we extend two representative variants to asynchronous execution, with and without f-OPD: Entropy-aware OPD (EOPD) [32] for mathematical reasoning and Step-wise OPD (SOD) [50] for tool-integrated reasoning. We follow the original settings and evaluation metrics of each method. As shown in Table 4, f-OPD recovers most of the performance lost under asynchronous execution for both variants, while introducing only 5% and 8% throughput overhead for EOPD and SOD, respectively. For example, on AIME24, it improves EOPD from 21.2 to 32.1, approaching the reported synchronous result of 35.8. These results demonstrate that freshness-aware control is complementary to different OPD objectives.

C.3 Hyperparameter Sensitivity

We study the sensitivity of f-OPD to its key hyperparameters on mathematical reasoning (AIME25, avg@4) with two grid searches: (i) the drift coefficients α and β in the freshness score (Eq. 11), which balance rollout and supervision drift; and (ii) the refresh thresholds κ_f and κ_d, where κ_d denotes the shared threshold of κ_{roll} and κ_{sup} used throughout the paper. The freshness threshold ξ is fixed across all three tasks. When sweeping one group, the other is kept at its default value ($\alpha = 1.0$, $\beta = 0.5$, $\kappa_f = 0.10$, $\kappa_d = 0.5$).

Table 5 shows that rollout and supervision drift provide complementary signals: removing either channel ($\alpha = 0$ or $\beta = 0$) consistently degrades performance, and disabling both drift terms gives the lowest accuracy (30.8). The best configuration ($\alpha = 1.0$, $\beta = 0.5$) places relatively more weight on

Table 4: Compatibility of f-OPD with OPD variants. Synchronous results are from the original papers; throughput is normalized to the asynchronous baseline.

EOPD (Qwen3 8B $\rightarrow$ 8B-Base)			SOD (Qwen3 4B $\rightarrow$ 0.6B)		
Method	AIME24 (no-tool)	Throughput	Method	AIME24 (tool)	Throughput
Sync. OPD[†]	35.8	–	Sync. OPD[†]	20.84	–
Async. OPD	21.2	1.00×	Async. OPD	17.6	1.00×
f-OPD	32.1	0.95×	f-OPD	19.3	0.92×

[†] Synchronous results are reported from the original papers and are not used for throughput comparison.

rollout drift, which is consistent with rollout drift playing a larger role in short-horizon mathematical reasoning. For the refresh thresholds, accuracy varies only between 33.9 and 35.9 over the entire grid, well above vanilla asynchronous OPD (25.8). This indicates that f-OPD is robust to moderate hyperparameter variations and is not tuned to a fragile operating point.

Table 5: Hyperparameter sensitivity of f-OPD on AIME25 (avg@4 accuracy, %). Left: drift coefficients (α, β). Right: refresh thresholds (κ_f, κ_d).

	$\beta = 0$	$\beta = 0.5$	$\beta = 1.0$
$\alpha = 0$	30.8	31.7	32.6
$\alpha = 0.5$	33.2	34.8	34.2
$\alpha = 1.0$	34.1	**35.8**	35.4

	$\kappa_f = 0.05$	$\kappa_f = 0.10$	$\kappa_f = 0.15$
$\kappa_d = 0.4$	33.9	34.4	34.8
$\kappa_d = 0.5$	34.7	35.8	**35.9**
$\kappa_d = 0.6$	35.0	35.7	35.6

D Discussion and Limitations

Our claims are deliberately scoped: asynchronous OPD becomes unstable when the student outruns its data and supervision streams, and f-OPD reduces this mismatch through principled, freshness-aware control. Several limitations qualify this scope.

As for the limitation, first, our empirical evaluation is limited to relatively small-scale models, with the largest student and teacher models being 8B and 30B parameters, respectively. While the experiments cover diverse tasks, model families, and distillation settings, it remains to be validated whether the proposed f-OPD remains effective and efficient at substantially larger scales. Second, our theoretical analysis provides a discrepancy bound under bounded-regularity assumptions rather than a convergence theory. The bound characterizes how rollout and supervision staleness can contribute to the gap between asynchronous and synchronous objectives, but does not establish convergence of asynchronous training or eliminate asynchronous bias. Extending the analysis toward stronger guarantees and validating f-OPD at larger model scales are important directions for future work.

E Broader Impact

f-OPD offers a practical recipe for navigating the performance–efficiency trade-off in on-policy distillation, supporting more reliable post-training of large language models under realistic asynchronous pipelines. Beyond technical contributions, our method can lower the resource barrier for academic and resource-constrained research groups, helping democratize a direction currently concentrated in a few well-resourced institutions.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Yes, please refer to the conclusion.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We have a complete theoretical analysis for asynchronous OPD objective discrepancy. Please refer to section 3 and appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Please refer to section and appendix.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Our code and instructions are included in the appendix. The data we use for the experiments are all from open-access datasets.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Please refer to our experiment section and appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Most of our results are reported by five-seed means with standard deviations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Please refer to our appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Please refer to our appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Not related to this research.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We properly credit data, code base, and student/teacher models that we used in this paper.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We document well about the asset.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: Generative AI tools were used solely for writing, language editing, and formatting, and did not contribute to the core methodology, experimental design, scientific analysis, or original research contributions.

Guidelines:

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