DO WE NEED ONLINE ENVIRONMENT INTERACTION IN AGENTIC OPD? AN EMPIRICAL ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) has become a common post-training approach where the student rolls out complete trajectories on its own, and the teacher provides token-level supervision. In multi-turn agentic tasks, on-policy rollouts of the student rely on iterative environment interaction to receive tool call feedback, incurring heavy time cost for long-horizon tasks. Yet, it remains poorly understood whether continuous environment interaction is necessary for agentic OPD. In this work, we analyze the limitations of agentic OPD with full rollouts, compare it with its environment-free alternative, and characterize the trade-offs between the two. We observe that in long-horizon task, teacher-student agreement increases with trajectory depth, yielding little supervision signal despite a substantial gap in task performance. In contrast, conditioning the student on replayed teacher prefixes retain stronger signal at later turns. Empirically, OPD with replayed prefixes matches or exceeds full-rollout performance on SWE-bench Verified and BFCL v4 with lower wall-clock cost, without online environment interaction. Our analysis reveals that OPD with replayed prefix achieve larger gains at pass@1 under a fixed token budget, whereas full rollouts brings pass@k closer to the teacher. Replayed prefixes reduce training instability associated with overlong responses under limited SFT cold start and yield larger gains across different cold-start levels, although more complete SFT initialization still leads to higher final performance. Together, our results establish that OPD with replayed prefix is ideal when training efficiency is the priority, while full rollouts are better suited for pushing the model's performance upper bound.

1 INTRODUCTION

On-policy distillation (OPD) has become a standard stage in language model post-training (Qwen Team, 2025; Lu & Thinking Machines Lab, 2025). It combines the benefits of on-policy exploration of reinforcement learning (RL) with the dense, token-level supervision of supervised fine-tuning (SFT) by training the student on its own generated outputs using feedback from a teacher model (Gu et al., 2024; Agarwal et al., 2024). For long-horizon agentic tasks such as software engineering (Jimenez et al., 2024), however, an on-policy rollout requires repeated interaction with an environment. The training steps may involve executing shell commands in sandboxes or calling external tools, and waiting for execution feedback. With full rollouts, each update requires fresh environment interactions, and synchronous batches can be delayed by a few long trajectories (Lu et al., 2025; Fu et al., 2025). Yet the cost of reaching later turns does not guarantee that those turns provide useful supervision, as student trajectories drift further from states the teacher can reliably evaluate.

The cost of online interaction has motivated several approaches to efficient long-horizon agentic OPD. TCOD and TurnOPD control rollout depth and the allocation of supervision across turns (Wang et al., 2026; Zhou et al., 2026). ReOPD removes online environment interactions during OPD training by replaying teacher histories and supervising student-generated single turns (Liao et al., 2026). MiMo-V2.6 introduces a similar recipe of combining full student rollouts with single-turn generation from teacher trajectories (Xiaomi MiMo Team, 2026). These works suggest that replaying previously collected histories can be a potential alternative to full on-policy rollouts. Yet, the mechanism and consequences of this substitution remain poorly understood. In long-horizon agentic tasks, prefix replay changes not only the computational cost of training but also the distribution of histories on which the student receives supervision. Prior works still leave open how

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

teacher supervision actually evolves along long-horizon trajectories, the trade-off when online environment interaction is removed, and under which conditions full rollouts remain preferable. We therefore study the following question: ***What is gained or lost when online environment interaction is replaced by prefix replay in agentic OPD, and under what conditions should each recipe be used?***

To answer this question, we first measure how teacher supervision changes with trajectory depth in long-horizon agentic tasks. We then train on several student and teacher pairs on coding and tool use tasks, and compare agentic OPD with full rollouts to replayed prefixes against wall-clock time. We further analyze the capability boundary of the trained students, and study how much each method recovers from students cold-started to different degrees. Our findings are summarized as follows:

- **On long-horizon agentic task, the teacher provides weak supervision on student's own rollouts.** As trajectories deepen, the teacher assigns increasingly high probability to most student-generated tokens, while the fraction of tokens receiving low teacher probability declines. Despite the student-teacher gap in task performance, continued environment interaction produces little supervision signal (Section 3.1).
- **OPD with replayed prefix retain stronger supervision signal and faster training.** Conditioning on offline teacher prefixes preserves more supervision signal at later turns. At an equal supervised-token budget, OPD with replayed prefixes match full rollouts in a fraction of the wall-clock time, without online environment interaction (Section 3.2).
- **OPD with replayed prefix yields larger gain in pass@1, full rollouts closes more of the gap at pass@k.** Replayed prefixes focus supervision on teacher-visited histories and raise pass@1. Full rollouts supervise every path the student takes and bring pass@k closer to teacher, which suggests its potential role as a cold start before RL to lift the upper bound for the student (Section 4.1).
- **Replayed prefixes reduce training instability under limited SFT cold start, but do not eliminate its importance.** More complete cold start leads to higher final performance, and these differences persist after OPD. When SFT cold start is light, full-rollout OPD exhibits reward-hacking-like behavior where KL loss decreases with overlong responses, yet the task score remains unchanged. OPD with replayed prefixes could instead train more stably and yield larger performance gains at every cold-start level we evaluate (Section 4.2).

We hope these findings give researchers of agentic OPD a choice of recipe rather than a single default: replayed prefixes when the goal is to reach the teacher's score quickly at a fixed budget or to start from a lightly cold-started student, full rollouts when the goal is to raise the student's ceiling.

2 PRELIMINARIES

2.1 AGENTIC ON-POLICY DISTILLATION

An agent solves a task $x \sim \mathcal{D}$ over T turns. At turn t it sees the history $H_t = (o_1, a_1, \ldots, a_{t-1}, o_t)$, emits a turn $a_t = (a_t^1, \ldots, a_t^{L_t})$ of reasoning followed by an action, and receives $o_{t+1} \sim \mathcal{E}(\cdot \mid x, H_t, a_t)$ from the environment $\mathcal{E}$. A student π_θ and a teacher π_T define next-token distributions over a shared vocabulary. On-policy distillation (Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025) samples trajectories from the student and aligns it to the teacher on every assistant token it generates. Observation tokens are not scored:

$$\mathcal{L}_{\text{agentic}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\ \tau\sim\pi_\theta\circ\mathcal{E}}\Big[\sum_{t=1}^{T}\sum_{j=1}^{L_t} D_{\text{KL}}\big(\pi_\theta(\cdot \mid x, H_t, a_t^{<j}) \,\|\, \pi_T(\cdot \mid x, H_t, a_t^{<j})\big)\Big]. \quad (1)$$

In practice the KL is optimized by policy gradient with the per-token advantage

$$A_t^j = \log \pi_T(a_t^j \mid x, H_t, a_t^{<j}) - \log \pi_\theta(a_t^j \mid x, H_t, a_t^{<j}), \quad (2)$$

evaluated at the sampled token. Every update rolls the student out to termination, the teacher scores every turn, and the batch waits for its longest trajectory.

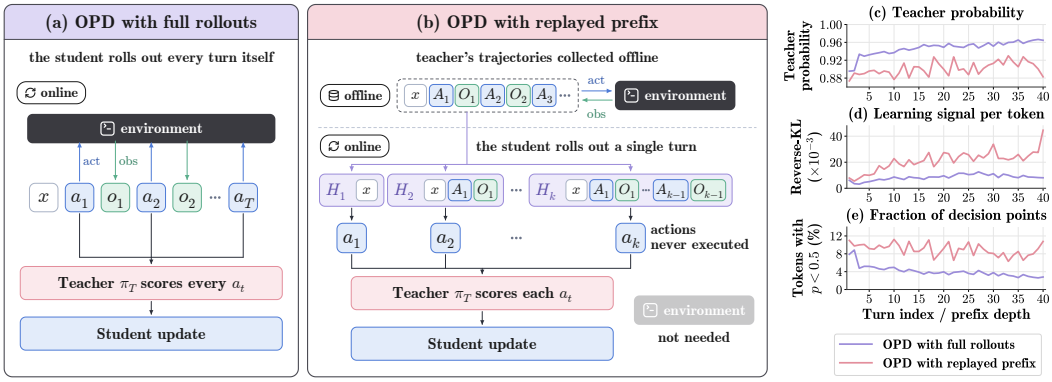

Figure 1: **OPD with full rollouts versus OPD with replayed prefixes. (a)** OPD with full rollouts generates complete student trajectories through online environment interaction, with the teacher scoring every assistant turn. **(b)** OPD with replayed prefixes generates one student turn from a stored teacher history, without executing the sampled actions. **(c–e)** On coding tasks, OPD with full rollouts exhibits increasing teacher–student agreement and diminishing per-token supervision as trajectories deepen. OPD with replayed prefixes retains larger per-token divergence and more tokens with teacher probability below 0.5 at later turns.

2.2 Every Turn Is Already Trained on a Prefix

Let $d_\theta^t(H_t \mid x)$ be the distribution of turn-t histories the student induces in $\mathcal{E}$, and let

$$\ell(\theta; x, H_t) = \mathbb{E}_{a\sim\pi_\theta(\cdot|x,H_t)}\Big[\sum_{j=1}^{|a|} D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x, H_t, a^{<j}) \,\|\, \pi_T(\cdot \mid x, H_t, a^{<j})\big)\Big] \tag{3}$$

be the loss of one turn generated at a given history. Then Eq. 1 is a sum of such terms,

$$\mathcal{L}_{\mathrm{agentic}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\Big[\sum_{t=1}^{T} \mathbb{E}_{H_t\sim d_\theta^t(\cdot|x)}\big[\ell(\theta; x, H_t)\big]\Big]. \tag{4}$$

For a fixed history, the per-turn loss and its update do not depend on where the history came from. With $A^j = \log \pi_T(a^j \mid \cdot) - \log \pi_\theta(a^j \mid \cdot)$, the token-local update is

$$g_{\mathrm{local}}(\theta; x, H_t) = -\mathbb{E}_{a\sim\pi_\theta(\cdot|x,H_t)}\Big[\sum_j \mathrm{sg}(A^j)\, \nabla_\theta \log \pi_\theta(a^j \mid x, H_t, a^{<j})\Big], \tag{5}$$

where sg is stop-gradient. It holds H_t fixed and needs no environment call. Eq. 4 is one case of

$$\mathcal{L}_\mu(\theta) = \mathbb{E}_{(x,t,H_t)\sim\mu}\big[\ell(\theta; x, H_t)\big], \tag{6}$$

where μ picks a task, a turn, and a history. Full rollouts draw histories from d_θ^t. A fixed pool of stored histories can leave the student's own error states uncovered, the exposure-bias concern behind DAgger (Ross et al., 2011). Thus, sharing a local update does not eliminate the state-coverage trade-off.

2.3 On-Policy Distillation with Replayed Prefixes

Replayed-prefix OPD draws histories from a fixed pool instead of from d_θ^t. It was introduced for agents by Liao et al. (2026) and is used at scale in the post-training of MiMo-V2.6 (Xiaomi MiMo Team, 2026), where single-turn rollouts conditioned on prefixes from teacher trajectories train the decision points without regenerating the turns before them. A prefix writer π_w interacts with $\mathcal{E}$ once, before distillation, producing a pool $\mathcal{P} = \{\tau_n\}_{n=1}^N$ of complete trajectories. Here π_w is the RL teacher and the pool is never refreshed. Each trajectory is cut at the boundaries between assistant

turns and the observations that follow them, giving prefixes $H_1, \ldots, H_{T_n}$, and ω selects one turn index per trajectory, uniformly unless stated otherwise (Appendix C). A step places each prefix into the context verbatim with its tokens masked, samples one turn from the student, and scores it with the teacher:

$$\mathcal{L}_{\text{replay}}(\theta) = \mathbb{E}_{\tau\sim\mathcal{P}}\, \mathbb{E}_{t\sim\omega(\cdot|T_\tau)}\big[\ell(\theta; x, H_t)\big]. \tag{7}$$

This is Eq. 6 with $\rho_t = d_w^t$, the writer's occupancy on $\mathcal{P}$, and the advantage, estimator, and optimizer are those of full rollouts. In the single-turn variant the sampled turn is not executed and training never touches $\mathcal{E}$. In the continuation variant it is executed and the student continues for its remaining turns, all scored. A single-turn example costs one turn of generation regardless of T, so supervised tokens per example fall from $\sum_t L_t$ to L_t and rollout time has no long tail. What is given up is on-policyness in the history.

3 The Efficiency of Agentic OPD on Replayed Prefixes

With the settings of agentic OPD on full rollouts and replayed prefixes established, we now compare their supervision signals and practical performance. We first examine how teacher supervision varies with trajectory depth in long-horizon task (Section 3.1). We then evaluate task performance against training time across coding and tool use tasks with different student and teacher pairs (Section 3.2).

3.1 Long-Horizon Agentic OPD Signal Dynamics

Setup. We use SWE-Master-4B-SFT as the student and SWE-Master-4B-RL as the teacher (Song et al., 2026). We sample four trajectories from each model per task on 60 SWE-bench Verified tasks, allowing up to 40 turns for this analysis. To evaluate OPD with full rollouts, we condition the teacher on the student's rollouts and score every student-generated token. For OPD with replayed prefixes, we split each teacher trajectory at every assistant-turn boundary, let the student generate one turn from each prefix, and score its tokens with the teacher. We record both models' token log probabilities and entropies, estimate reverse KL with the sampled-token k_3 estimator (Schulman, 2020). We define a **decision point** as a sampled student token assigned probability below 0.5 by the teacher. All measurements are grouped by turn index or prefix depth.

OPD with full rollouts provides weak supervision despite a substantial performance gap. The student scores 23.2 on SWE-bench Verified, compared with 32.2 for the teacher (Table 1). Yet the teacher assigns high probability to most tokens along the student's trajectories. As shown in Figure 1(c), its average probability of the sampled student token exceeds 0.95 at most later turns and reaches approximately 0.965 by turn 40. Figure 1(d,e) shows that per-token KL remains small and the fraction of decision points falls from roughly 8% to 3%. Thus, the performance gap is accompanied by little disagreement on most visited tokens. Continued environment interaction produces additional tokens, but few receive substantial corrective supervision.

OPD with replayed prefixes retains more training signal at later turns. As shown in Figure 1(d,e), the fraction of decision points on replayed teacher prefixes remains around 10%, and per-token KL is about two times larger than on the student's own histories at later turns. The entropy measured in Figure A2(c) show a similar difference. Both the teacher and student retain their entropy at later turns when trained on replayed prefixes, while their entropies decrease on full student rollouts. Teacher prefixes therefore continue to expose differences between the models at depths where the student's own histories contain few decision points. The concentration of supervision signal in a small fraction of tokens is related to the finding that high-entropy tokens account for much of effective RLVR training (Wang et al., 2025a). We identify that the percentage of tokens with low teacher confidence is higher when student is trained on replayed teacher prefix. We next examine whether this difference translates into better training efficiency.

Insights

- On long-horizon agentic task, the teacher's signal on student's own rollouts is weak despite their capability gap. Replayed teacher prefixes retain more training signal toward later turns.

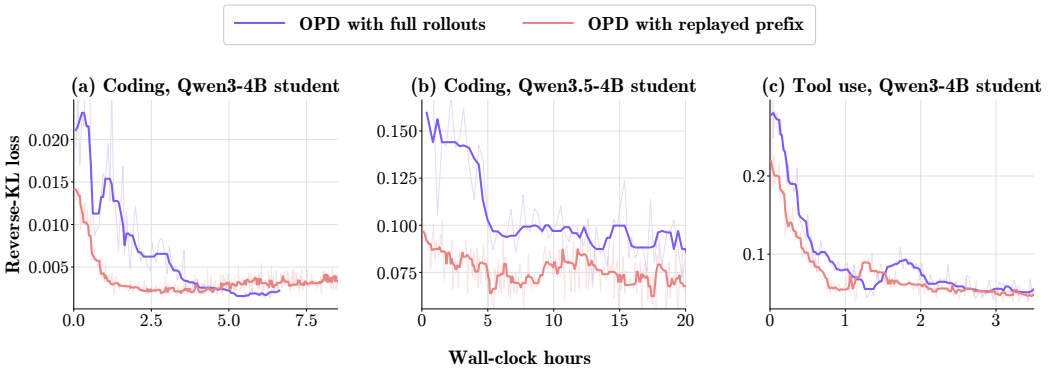

Figure 2: Reverse-KL training loss against wall-clock for OPD on full rollouts and on replayed prefixes at the operating points of Table 1. (a) Coding with the Qwen3-4B student, (b) coding with the Qwen3.5-4B student, and (c) tool use with the Qwen3-4B student. Each curve is cut off at the point where its loss has converged. On coding, replayed prefixes reach the converged loss level of full rollouts in about a third of the wall-clock in (a) and stay below it throughout in (b). On tool use, the two converge to the same level, with replayed prefixes arriving in about half the time.

3.2 Training Efficiency of OPD with Full Rollouts and Replayed Prefixes

Setup. We evaluate three student-teacher pairs across coding and tool use. For our same-size configurations, the teacher is an RL-trained counterpart initialized from the student's SFT checkpoint. For coding, we use SWE-MASTER-4B-SFT and SWE-MASTER-4B-RL. For tool use, we pair a Qwen3-4B student fine-tuned on the FunReason-MT SFT data with the released FunReason-MT RL model (Xu et al., 2025). To study distillation across model scales, we also evaluate the distillation from Qwen3.5-9B to Qwen3.5-4B on coding. The training data for coding is drawn from SWE-rebench (Badertdinov et al., 2025), and we evaluate on SWE-bench Verified. For tool use, the data comes from FunReason-MT RL data, and we evaluate on BFCL v4 multi-turn (Patil et al., 2025). Both settings use sampled-token OPD with the advantage in Eq. 2. OPD with replayed prefixes generates one student turn from a stored teacher history without executing the sampled action.

OPD with replayed prefixes reduces training loss faster. Figure 2 shows reverse-KL training loss against wall-clock time. For the Qwen3-4B coding student and the tool-use student, OPD with replayed prefixes reaches a comparable loss level sooner. For the Qwen3.5-4B coding student, its loss remains below that of OPD with full rollouts throughout the run. Each replay sample requires only one generated turn and no environment execution, reducing the time spent collecting training samples. Because the losses are measured on different histories, we also compare task scores to assess the benefit of faster loss reduction.

With a shared SFT initialization, replayed prefixes reach comparable or higher scores faster. On SWE-bench Verified, OPD with full rollouts reaches 29.7 in 5.78 hours with 0.95M supervised tokens. OPD with replayed prefixes reaches 29.6 in 1.96 hours with 0.92M supervised tokens (Table 1). It achieves nearly the same score in about one third of the time at a comparable token budget. The student generates one turn per sample instead of an average of 16.2 turns. On BFCL v4 multi-turn, OPD with replayed prefixes reaches 49.63 in 4.45 hours, compared with 47.38 in 5.65 hours for OPD with full rollouts. Replayed prefixes therefore reduce training cost in both task families while retaining or improving the final score.

Replayed prefixes support distillation across model sizes without additional SFT cold start. The Qwen3.5-4B student starts at 33.4 on SWE-bench Verified, below the Qwen3.5-9B teacher's 50.8. OPD with full rollouts reaches 35.2 after 15.38 hours. OPD with replayed prefixes reaches 40.8 in 8.30 hours, recovering about four times more of the initial gap compared to OPD with full rollouts (Table 1). Supplying teacher histories lets the student receive supervision at those states without first generating the preceding interactions itself. In this setting, replayed prefixes allow substantial improvement where OPD with full rollouts makes limited progress.

Table 1: Task performance of OPD with full rollouts and OPD with replayed prefixes on coding and tool use. Time cost is reported as training wall-clock with the ratio to full rollouts in purple.

Method	Rollout Turns	Time Cost (h)	Score ↑
Coding SWE-Master-4B-SFT (23.2) → SWE-Master-4B-RL (32.2)			
Full rollouts	16.2	5.78	29.7↑6.5
Replayed prefixes	1	1.96 (0.34×)	29.6↑6.4
Coding Qwen3.5-4B (33.4) → Qwen3.5-9B (50.8)			
Full rollouts	16.5	15.38	35.2↑1.8
Replayed prefixes	1	8.30 (0.54×)	40.8↑7.4
Tool use FunReason-4B-SFT (45.00) → FunReason-4B-RL (55.62)			
Full rollouts	6.2	5.65	47.38↑2.4
Replayed prefixes	1	4.45 (0.79×)	49.63↑4.6

Insights

- On the student and teacher pairs we train, OPD with replayed prefixes matches or exceeds the performance of OPD with full rollouts at lower wall-clock cost, without environment interaction during distillation.

4 Analysis of OPD with Full Rollouts and Replayed Prefixes

The efficiency gains of OPD with replayed prefixes compared to full rollouts raise further questions about what the student learns and when to use each setting. We first examine their effects on the student's capacity boundary (Section 4.1). We then study how their training behavior and performance depend on the amount of SFT used to initialize the student (Section 4.2).

4.1 Agentic OPD's Effects on Model Capacity Boundary

Setup. We compare the tool use student model trained using OPD with full rollouts and OPD with replayed prefixes at the same supervised-token budget. We randomly select 200 tasks from BFCL v4 multi-turn, with 50 tasks from each of its four categories. We sample 64 complete trajectories per task from the tool use SFT student, the RL teacher, and both distilled models in Table 1. Following Yue et al. (2025), we examine pass@k together with the distribution of per-task accuracy.

OPD with replayed prefixes yields higher pass@1 at the same token budget. OPD with replayed prefixes raises pass@1 from 44.0 to 50.8, compared with 45.4 for OPD with full rollouts. It also places more tasks in the high-accuracy bins of Figure 3 (right), indicating more consistent success across repeated attempts. Its improvement at pass@64 is smaller, from 74.5 to 77.0. Together, these results suggest that much of its advantage comes from increasing the probability of successful solutions, with a more modest increase in the number of tasks solved within 64 attempts.

This pattern is consistent with the stronger supervision signal observed in Section 3.1. Replayed prefixes repeatedly place the student at teacher-visited histories where the two models still disagree. The teacher has itself undergone RLVR, which prior work finds can increase the probability of already available successful solutions (Yue et al., 2025). Conditioning on its trajectories may help transfer and sharpen these more reliable behaviors. However, the history pool remains fixed even as the student samples new turns, limiting direct supervision on histories outside that pool. This offers a possible explanation for the smaller gain at large k.

OPD with full rollouts approaches the teacher at large k. The gap between the two distilled models narrows as the sampling budget grows in Figure 3 (left). Their curves meet at $k = 16$. At $k = 64$, OPD with full rollouts reaches 78.5, compared with 77.0 for OPD with replayed prefixes and 80.0 for the teacher. Although its pass@1 remains far below the teacher's 57.5, it solves nearly as many tasks as the teacher when given 64 attempts. OPD with full rollouts provides supervision

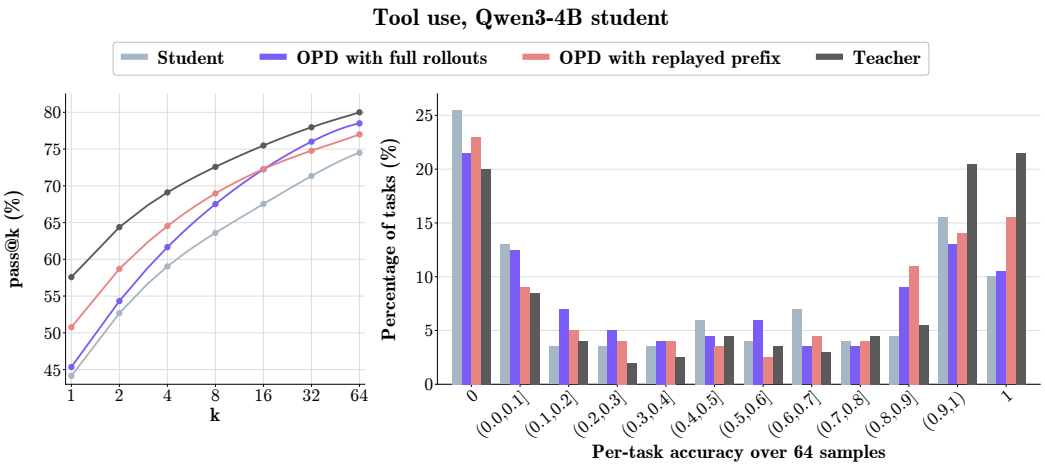

Figure 3: **Left:** pass@k on tool use for the SFT start, the two distilled students, and the RL teacher. **Right:** distribution of per-task accuracy over 64 samples. OPD with replayed prefixes improves pass@1 the most and moves the most tasks into the high-accuracy bins; OPD with full rollouts draws level at large k.

along fresh student-generated interaction histories, including paths absent from the replay pool. This may improve success across a wider range of trajectories, even when weaker per-token supervision produces smaller gains in pass@1.

The remaining gap between pass@1 and pass@64 suggests a possible role for subsequent RLVR. Li et al. (2026a) find that OPD followed by RLVR improves reasoning performance by increasing solution coverage before concentrating probability on successful solutions. Related work also shows that expert guidance can improve pass@k during RLVR (Li et al., 2026b; Nath et al., 2025). Our results suggest that full-rollout OPD might serve as a useful initialization for RLVR, as it brings pass@k closer to the teacher's level under the same supervised-token budget.

Insights

- At the same supervised-token budget, OPD with replayed prefixes yields higher pass@1, while OPD with full rollouts approaches the teacher at large k. The latter suggests a potential role as initialization for subsequent RLVR.

4.2 How Does SFT Cold Start Affect Agentic OPD?

Setup. In our same-size OPD configurations, the student and the teacher share the same SFT checkpoint, and the teacher was obtained through RL. We now examine how the amount of SFT coldstart of the student affects the outcome. Starting from the same Qwen3-4B instruct model, we construct three checkpoints using 2,000, 5,000, and all 55,729 tool use SFT trajectories. We then apply OPD with full rollouts and OPD with replayed prefixes to each checkpoint, keeping the OPD training schedule fixed across cold-start levels.

More complete cold start leads to higher final performance. The three initializations reach different performance levels on BFCL after distillation. For OPD with full rollouts, final scores rise from 37.12 to 41.62 and 47.38 as the SFT data increase. For OPD with replayed prefixes, they rise from 43.38 to 45.62 and 49.63. Although the 5,000-trajectory checkpoint gains the most from distillation, the fully cold-started student achieves the highest final score under both settings. OPD therefore does not erase the differences left by SFT initialization.

Insufficient cold start exposes a failure mode in long rollouts. For the 2,000- and 5,000-trajectory students, OPD with full rollouts drives KL downward and teacher–student overlap upward, yet task performance remains well below that of the fully cold-started student. The 2,000-trajectory student

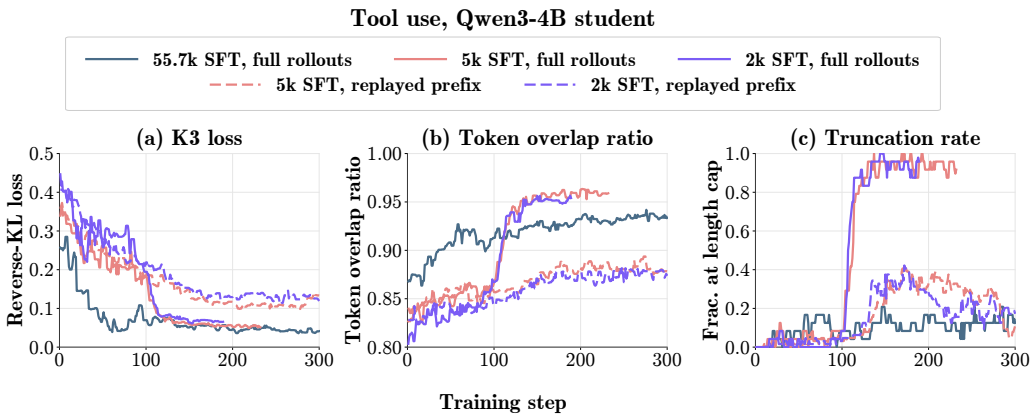

Figure 4: Training dynamics of OPD from three degrees of cold start on tool use: reverse-KL loss, sampled-token overlap between student and teacher, and the fraction of turns reaching the length cap, for full rollouts (solid) and replayed prefixes (dashed).

shows no score improvement at all. Meanwhile, the fraction of turns reaching the generation length cap rises toward one in Figure 4. Training increasingly produces overlong continuations on which teacher agreement is high but corrective supervision is weak. This resembles reward hacking of the OPD KL loss objective, where student-teacher agreement improves with overlong response during training, as illustrated by an excerpt in Table A6, yet the final task score remains unchanged.

Replayed prefixes mitigate this failure but do not replace sufficient cold start. OPD with replayed prefixes improves all three initializations and shows substantially lower truncation rates for the lightly cold-started students. Supplying the interaction history reduces the opportunity for earlier student mistakes and overlong continuations to determine subsequent training contexts. This is consistent with its more stable training and the stronger per-token signal observed in Section 3.1. Nevertheless, the lightly cold-started students still finish below the fully cold-started student. Together with the lower training cost in Section 3.2, these results support replayed prefixes when SFT data and training time are limited.

Table 2: Distillation from three degrees of cold start on tool use, BFCL v4 multi-turn overall. Start is the score of the SFT checkpoint, the next two columns are its score after distillation on full rollouts and on replayed prefixes, with the gain in green. The teacher scores 55.62.

SFT traj.	Start	Full rollouts	Replayed prefixes
2,000	37.12	37.12↑0.0	43.38↑6.26
5,000	37.62	41.62 ↑4.0	45.62 ↑8.0
55,729	45.00	47.38↑2.4	49.63↑4.63

Insights

- With insufficient SFT cold start, OPD with full rollouts can improve teacher agreement while producing overlong continuations and little task improvement. OPD with replayed prefixes mitigates this failure, although more complete cold start still yields higher final scores.

5 Discussion

Relation to previous prefix replay works. ReOPD introduces prefix replay for agentic OPD and analyzes the trade-off between visiting student-generated histories and obtaining reliable teacher targets (Liao et al., 2026). Its experiments, however, are conducted on tool-integrated mathematical reasoning and search-based question answering, where trajectories are comparatively short and the student needs only a few tool calls per task. It reports final task accuracy but does not compare how the capabilities of students trained with replayed prefixes differ from those trained with full rollouts, nor does it characterize the settings in which each recipe is preferable. MiMo-V2.6 also uses histories from teacher trajectories and to train individual student turns alongside full student

rollouts (Xiaomi MiMo Team, 2026). Prefix replay is introduced as one component of a larger post-training recipe, and no ablation isolates its contribution from that of full student rollouts. We isolate prefix replay and compare it against full rollouts on long-horizon coding and multi-turn tool use settings. Our analysis measures supervision across trajectory depths, distinguishes gains in student capacity boundary, and examines dependence on SFT cold start.

OPD as cold start and future work. OPD is often used as a final stage to close the gap to a stronger teacher, but Li et al. (2026a) find that on single-turn reasoning, OPD mainly expands the set of solutions the student can reach, while RLVR sharpens probability within that set, so running OPD first and RLVR afterwards outperforms either stage alone or training both jointly. In that view, OPD serves as a better cold start for RL than SFT. Our results in Section 4.1 point in a similar direction for agentic tasks. OPD with full rollouts gains less at pass@1 than OPD with replayed prefixes, but it approaches the teacher at large k, suggesting that it widens what the student can solve under the same training budget. This makes full rollouts a natural candidate for the first stage of an OPD-then-RLVR pipeline, where the later RL stage can convert that coverage into higher pass@1. Whether this holds in multi-turn settings, and in long-horizon agentic tasks in particular, remains an open question. We leave it to future work, which could explore different training pipelines for agentic tasks.

6 Related Work

On-policy distillation. MiniLLM and generalized knowledge distillation train students using teacher feedback on student-generated sequences (Gu et al., 2024; Agarwal et al., 2024). DistiLLM-2 further studies distillation using teacher- and student-generated outputs (Ko et al., 2025). OPD has since been adopted in LLM post-training (Qwen Team, 2025; Lu & Thinking Machines Lab, 2025). Recent studies examine the conditions under which a teacher improves a student, the alignment of their token distributions, and the role of training data (Li et al., 2026c; Fu et al., 2026). Our work examines supervision in multi-turn settings, where the histories used for distillation depend on environment interaction.

OPD for multi-turn agents. TCOD progressively increases the trajectory depth exposed to the student, while TurnOPD adjusts rollout depth and supervision across turns (Wang et al., 2026; Zhou et al., 2026). SAGE-OPD studies selective teacher intervention, and Ye et al. (2026) examine the cost of short teacher continuations. For single-turn reasoning, Zhang et al. (2026) reduce generation cost by terminating student outputs early. These methods change where generation or supervision is allocated. Our comparisons examine how using full rollouts or replayed prefixes changes the supervision and the resulting student behavior.

Analysis of RLVR and distillation. Yue et al. (2025) use pass@k and per-task accuracy distributions to distinguish improved sampling efficiency from changes in problem coverage. Wang et al. (2025a) find that a small fraction of high-entropy tokens accounts for much of effective RL training. Li et al. (2026a) study the complementary effects of OPD and RLVR, while Zhai et al. (2026) examine agent performance across sampling and turn budgets. These studies motivate our analysis of supervision across trajectory depths and task success across sampling budgets. We additionally vary SFT initialization to examine when teacher agreement fails to translate into better task performance.

7 Conclusion

We studied what online environment interaction contributes to agentic OPD. In our long coding trajectories, agreement between teacher and student increases with depth while corrective supervision becomes sparse. OPD with replayed prefixes retains a stronger per-token signal, matches or exceeds scores from OPD with full rollouts at lower wall-clock cost, and mitigates failures under limited SFT cold start. The two settings also differ across sampling budgets. OPD with replayed prefixes yields higher pass@1 on tool use, while OPD with full rollouts approaches the teacher at large k. These results support replayed prefixes when training efficiency is the priority and motivate testing full rollouts as initialization for subsequent RLVR.

AI USE STATEMENT

Generative AI tools were used in this work for literature search and summarization, for drafting and editing portions of the text, and for assistance in writing analysis and plotting code. All AI-assisted text was reviewed and revised by the authors; all AI-assisted code was verified by the authors on held-out examples; and the research questions, experimental design, and interpretation of results are the authors' own. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies training procedures for language-model agents on public benchmarks and involves no human subjects or personal data. Procedures that lower the cost of transferring capabilities between models could be applied to models whose capabilities raise safety concerns; the procedures studied here transfer the behaviour of a teacher that is already available and do not create capabilities the teacher lacks.

REPRODUCIBILITY STATEMENT

Section 3.2 and Appendix A name the models, benchmarks, and list the full training configuration. Appendix B gives the objective, the estimators, and the training loop. Appendix C gives the rule used to segment trajectories into turns and the construction of the prefix pools. Relevant code will be submitted as supplementary material.

REFERENCES

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. arXiv:2306.13649.

Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Andrei Andriushchenko, Maria Trofimova, Daria Litvintseva, and Boris Yangel. SWE-rebench: An automated pipeline for task collection and decontaminated evaluation of software engineering agents. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. arXiv:2505.20411.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.24298.

Zixuan Fu, Bingxiang He, Yuxin Zuo, Haohuan Huang, Jinqian Zhang, Ruhang Xiao, Cheng Qian, Qinyu Luo, Huan-ang Gao, Yudong Wang, Zhiyuan Liu, Ning Ding, and Chaojun Xiao. Rethinking on-policy distillation of large language models II: One training example. *arXiv preprint arXiv:2609.04172*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. arXiv:2306.08543.

Zeyu Huang, Tianhao Cheng, Zihan Qiu, Zili Wang, Yinghui Xu, Edoardo M. Ponti, and Ivan Titov. Blending supervised and reinforcement fine-tuning with prefix sampling. In *International Conference on Machine Learning*, 2026. Prefix-RFT; arXiv:2507.01679.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Conference on Language Modeling*, 2025.

Zishang Jiang, Jinyi Han, Tingyun Li, Xinyi Wang, Sihang Jiang, Jiaqing Liang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. Selective expert guidance for effective and diverse

exploration in reinforcement learning of LLMs. In *International Conference on Learning Representations*, 2026. MENTOR; arXiv:2510.04140.

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 *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 31044–31062, 2025. arXiv:2503.07067.

Boyan Li, Bingsen Chen, Chenghao Yang, Ping Nie, Chen Zhao, and Xi Ye. Sequential beats joint: On the interplay between on-policy distillation and RLVR. *arXiv preprint arXiv:2609.04108*, 2026a.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. QuestA: Expanding reasoning capacity in LLMs via question augmentation. In *International Conference on Learning Representations*, 2026b. arXiv:2507.13266.

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

Baohao Liao, Hanze Dong, Christof Monz, Xinxing Xu, Li Dong, and Furu Wei. Multi-turn on-policy distillation with prefix replay. *arXiv preprint arXiv:2607.04763*, 2026.

Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. UFT: Unifying supervised and reinforcement fine-tuning. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.16984.

Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yanan Wu, Weixun Wang, Jiashun Liu, Yang Li, Haizhou Zhao, Ju Huang, Siran Yang, Xiaoyang Li, Yijia Luo, Zihe Liu, Ling Pan, Junchi Yan, Wei Wang, Wenbo Su, Jiamang Wang, Lin Qu, and Bo Zheng. Part II: ROLL Flash – accelerating RLVR and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

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

Vaskar Nath, Elaine Lau, Anisha Gunjal, Manasi Sharma, Nikhil Baharte, and Sean Hendryx. Adaptive guidance accelerates reinforcement learning of reasoning models. *arXiv preprint arXiv:2506.13923*, 2025. Guide.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Stéphane 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 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

John Schulman. Approximating KL divergence. http://joschu.net/blog/kl-approx.html, 2020.

Huatong Song, Lisheng Huang, Shuang Sun, Jinhao Jiang, Ran Le, Daixuan Cheng, Guoxin Chen, Yiwen Hu, Zongchao Chen, Yiming Jia, Wayne Xin Zhao, Yang Song, Tao Zhang, and Ji-Rong Wen. SWE-Master: Unleashing the potential of software engineering agents via post-training. *arXiv preprint arXiv:2602.03411*, 2026.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. TCOD: Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. *arXiv preprint arXiv:2604.24005*, 2026.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, 2025a. arXiv:2506.01939.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025b. arXiv:2407.16741.

Xiaomi MiMo Team. MiMo-V2.6 technical report. `https://huggingface.co/XiaomiMiMo/MiMo-V2.6-Pro-RL/blob/main/MiMo_V2_6_technical_report.pdf`, 2026.

Pei-Xi Xie, Che-Yu Lin, and Cheng-Lin Yang. Mitigating distribution sharpening in math RLVR via distribution-aligned hint synthesis and backward hint annealing. *arXiv preprint arXiv:2604.07747*, 2026.

Zengzhuang Xu, Bingguang Hao, Zechuan Wang, Yuntao Wen, Xinyi Xu, Yang Liu, Long Chen, Dong Wang, Maolin Wang, Tong Zhao, Yicheng Chen, Cunyin Peng, Jinjie Gu, Leilei Gan, Xiangyu Zhao, Chenyi Zhuang, and Shi Gu. FunReason-MT technical report: Advanced data synthesis solution for real-world multi-turn tool-use. *arXiv preprint arXiv:2510.24645*, 2025.

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

Junze Ye, Jiayi Cheng, Miao Lu, Michal Mankowski, Jose Blanchet, and Mohsen Bayati. A few teacher steps go a long way: Cost-efficient on-policy data augmentation for agent post-training. In *ICML 2026 Workshop on Reinforcement Learning from World Feedback (RLxF)*, 2026. arXiv:2607.04574.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *Advances in Neural Information Processing Systems*, 2025. arXiv:2504.13837.

Zhiyuan Zhai, Wenjing Yan, Xiaodan Shao, and Xin Wang. Does RL expand the capability boundary of LLM agents? a PASS@(k,T) analysis. *arXiv preprint arXiv:2604.14877*, 2026.

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. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 25553–25569, 2026. arXiv:2602.15260.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. BREAD: Branched rollouts from expert anchors bridge SFT & RL for reasoning. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2506.17211.

Yuhang Zhou, Kai Zheng, Haoling Li, Dengyun Peng, Can Xu, and Jingjing Chen. TurnOPD: Making on-policy distillation turn-aware for efficient long-horizon agent training. *arXiv preprint arXiv:2607.05804*, 2026.

A Experimental Details

Three student and teacher pairs are trained. Table A1 lists them. In the two same-size pairs the teacher is the RL-trained sibling of the SFT student, so the two share a base model, a tokenizer, and a task corpus, and what is distilled is what RL added. The cross-size pair starts from the released Qwen3.5-4B instruct model without any SFT cold start. Prefix writers other than the teacher, used in Appendix F, are the student itself, Qwen3.5-9B, and GPT-5.4. Throughout the appendix, V60 denotes a fixed 60-task subset of SWE-bench Verified evaluated with four samples per task, which is the setting of every coding ablation.

Table A1: Student and teacher pairs. Scores are on the benchmark of the corresponding task, SWE-bench Verified for coding and BFCL v4 multi-turn for tool use.

Task	Base model	Student	Teacher	Start	Teacher
Coding	Qwen3-4B-Instruct	SWE-Master-4B-SFT	SWE-Master-4B-RL	23.2	32.2
Coding	Qwen3.5	Qwen3.5-4B	Qwen3.5-9B	33.4	50.8
Tool use	Qwen3-4B	FunReason-4B-SFT	FunReason-4B-RL	45.00	55.62

Training prompts come from public corpora and the replay pools are collected once. On coding, distillation prompts are repository-level tasks from SWE-rebench (Badertdinov et al., 2025), each shipped with its own container image. The replay pool consists of trajectories written by the teacher, or by another writer, on the same tasks, collected once before distillation and never refreshed. Appendix C gives the rule that cuts a trajectory into prefixes and shows where the cuts fall.

Coding rollouts run in R2E-Gym with a CodeAct-style scaffold. Rollouts on coding, during training and at evaluation alike, run in the R2E-Gym environment (Jain et al., 2025). R2E-Gym provides the agent loop, the container runtime, and the test-based grading, and it accepts any task in the SWE-bench format, so the same loop serves the SWE-rebench training tasks and the SWE-bench Verified evaluation tasks. The scaffold is the OpenHands CodeAct configuration that R2E-Gym ships (Wang et al., 2025b). The agent sees a system prompt adapted from CodeAct and three tools, a string-replacement file editor for viewing and editing files, a bash tool for running commands in the repository, and a submit action that ends the episode. Tool calls are written by the model as text tags and parsed by the loop rather than passed through a function-calling interface, and each environment response is appended to the conversation as a user message.

The loop never compresses the history, so an episode ends by submitting or by running out of budget. An episode ends when the agent submits, when it reaches the turn budget, or when the conversation reaches the context limit. During training the turn budget is 20, each turn may generate at most 4,096 tokens, and the conversation is capped at 26,624 tokens so that one more turn always fits in the 32,768-token window of the model. No task reward is computed during distillation. At evaluation the same scaffold runs with a budget of 150 steps, a context window of 131,072 tokens, a conversation cap of 112,640 tokens that reserves 16,384 tokens for the final turn, and sampling at temperature 0.7. The evaluation budget is therefore far above the training budget, and the long-horizon behaviour of Appendix G is measured in this regime. The R2E-Gym runtime is not part of the released training code and is installed separately.

Evaluation uses the full benchmarks for the main results and fixed subsets for the ablations. On coding we report the resolve rate on SWE-bench Verified with all 500 tasks, sampled twice per task at temperature 0.7. Ablations use a fixed 60-task subset with four samples per task, written V60, and report the mean resolve rate over the four samples, so that differences of about a point lie within the run-to-run variation. LiveCodeBench with 131 problems serves as a single-turn check that a distilled student has not lost general coding ability. On tool use we report BFCL v4 multi-turn overall accuracy (Patil et al., 2025) with greedy decoding on all 800 tasks. Ablations and the pass@k study use a fixed 200-task subset with 50 tasks from each of the four categories, and for pass@k we draw 64 samples per task at temperature 0.7 and use the unbiased estimator of Yue et al. (2025).

The two regimes share one training configuration and differ only in how a batch is formed. Table A2 lists the optimizer, the loss, and the length limits. A full-rollout batch on coding holds four trajectories for the same-size pair and sixteen for the cross-size pair, each of up to 20 turns, and every assistant token of those trajectories receives a loss. A replayed-prefix batch holds 64 prefixes and supervises one generated turn per prefix. On tool use a full-rollout batch holds 32 prompts with eight rollouts each. Training runs on a single node with eight accelerators, and the teacher is loaded once as a frozen reference model that scores every sampled token of the batch in one forward pass.

Operating points are reported as wall-clock. The wall-clock of Table 1 is the sum of per-step training time, which includes rollout generation, teacher scoring, and the optimizer update, and excludes checkpoint writing and any interruption of the job.

Table A2: Training configuration shared by full rollouts and replayed prefixes. Where the two tasks differ, the coding value is given first.

Setting	Value
Optimizer	AdamW
Learning rate	1×10^{-6}, constant
Gradient clipping	1.0
Distillation loss	reverse KL (k_3)
Loss coefficient	1.0
Entropy bonus	0
Task reward	none
Sampling temperature	1.0
Max prompt length	16,384 tokens
Max response length	16,384 tokens
Per-turn cap	4,096 tokens
Context window	32,768 tokens
Turns per rollout	20 / to termination
Rollouts per step	4 (16 cross-size) / 256
Prefixes per step	64 / 64
Turns per prefix	1
Evaluation decoding	0.7 / greedy
Evaluation step budget	150 / to termination
Evaluation context	131,072 tokens

B Objective, Estimator, and Training Loop

The per-token divergence is the k_3 estimator of the reverse KL. For a student π_θ and a teacher π_T we measure the divergence at a sampled token as

$$k_3 = (r - 1) - \log r, \qquad r = \frac{\pi_T(a^j \mid \cdot)}{\pi_\theta(a^j \mid \cdot)}, \tag{8}$$

following Schulman (2020). It is a non-negative, low-variance estimate of the reverse KL at that position. Averaged over the assistant tokens of a batch, it is the training loss reported in every loss curve of this paper, and summed over a set of tokens we call it the distillation signal on that set. The token entropy at a position is $H = -\sum_v \pi(v \mid \cdot) \log \pi(v \mid \cdot)$, a property of the distribution rather than of the sampled token. A decision point is a sampled student token to which the teacher assigns probability below 0.5.

Both regimes use the sampled-token estimator, and one ablation uses a full-vocabulary one. Both regimes optimize Eq. 6 by policy gradient with the sampled-token advantage of Eq. 2, so that one student sample and one teacher scoring pass suffice for a gradient. Full rollouts set ρ_t to the current student's occupancy and score every visited turn. Replayed prefixes set ρ_t to the writer's occupancy on the pool and score one generated turn per prefix. The token-type study of Appendix D uses a full-vocabulary estimator instead, which evaluates the KL of Eq. 3 exactly at each sampled position over the whole vocabulary. That estimator gives higher absolute scores on V60, 25.8 against

22.9 for full rollouts after 100 steps, so scores from the two estimators are never compared with each other.

The replayed-prefix loop touches no environment. Algorithm 1 gives the loop for the single-turn variant. The turn index is drawn once per trajectory and frozen, which yields a fixed set of prefixes that is reused across epochs. Nothing the student generates is executed, and no environment call is made during training. The continuation variant, used in Appendix F, instead executes the generated turn in the container and lets the student continue for its remaining turns, all of which are scored.

Algorithm 1 Supervision on a replayed prefix, single-turn variant

Require: cold-started student π_{θ_0}, teacher π_T, prefix pool $\mathcal{P}$ written by π_w, position distribution ω

For each trajectory in $\mathcal{P}$ draw $t \sim \omega(\cdot \mid T)$ once and store the pair (x, H_t)
$\theta \leftarrow \theta_0$
for each training step **do**
  Sample a batch of stored prefixes $\{(x_i, H_{t_i}^{(i)})\}$ without any environment call
  Sample one turn per prefix, $a^{(i)} \sim \pi_\theta(\cdot \mid x_i, H_{t_i}^{(i)})$
  Score every token of $a^{(i)}$ with π_T and update θ with the gradient of Eq. 5
end for

C Prefix Construction

A prefix ends with an observation, and the position distribution decides where. A stored trajectory is segmented into assistant turns at the boundaries between an assistant message and the environment response that follows it. A prefix H_t is the trajectory up to, but not including, assistant turn t, so it always ends with an observation. Random-turn, the default in the main text, draws one t uniformly from $\{1, \ldots, T\}$ per trajectory. Last-turn keeps only H_T, all-turn keeps every H_t and therefore weights long trajectories by their length, and early-decay draws t with probability proportional to 0.6^t, the schedule of Liao et al. (2026). In every case the draw is made once and frozen, and the deep cut used in Appendix G restricts the draw to the later half of the trajectory.

The coding prefixes concentrate in the first ten turns. Figure A1 shows where they end. The teacher's trajectories on SWE-rebench are long, so a uniform draw over turns produces prefixes that concentrate in the first ten turns and thin out beyond turn 15, and the pools of the two coding pairs have nearly the same shape although one cuts each trajectory once and the other several times. This is the depth at which the single-turn regime places the student on coding, and Appendix D shows that it is also the depth at which the decision points sit that a full rollout stops visiting.



Figure A1: Turn at which the coding prefixes are cut, for the pool of the same-size pair and the pool of the cross-size pair, first 20 turns.

D WHERE THE SIGNAL LIES ALONG A TRAJECTORY

This section supplements Section 3.1. All measurements are on the same-size coding pair before distillation, with four rollouts per task on the 60-task SWE-bench subset and 223,214 assistant tokens in total, grouped by turn index up to turn 40.

Both models become confident together on the student's own path. Figure A2(c) shows that on the student's own rollouts the entropies of the student and of the teacher fall together, from about 0.24 nats at the first turn to below 0.1 nats by turn 40, and that the two curves lie almost on top of each other at every depth. The teacher conditioned on the student's history is nearly as certain as the student itself, so there is little for it to correct. On a replayed teacher prefix both entropies remain between 0.15 and 0.25 nats at every depth, and the per-token divergence in panel (a) is two to four times larger from turn 10 onward.

The teacher-side statistics tell the same story. Figure A3 scores the same rollouts from the teacher's side. The teacher's probability of the sampled student token rises from 0.90 at the first turns to about 0.965 by turn 40, the share of tokens with teacher probability above 0.99 rises from 60 to 87 percent, and the share of decision points falls from 8 to 3 percent. As the trajectory deepens the teacher has less and less to say about the tokens it is asked to score.

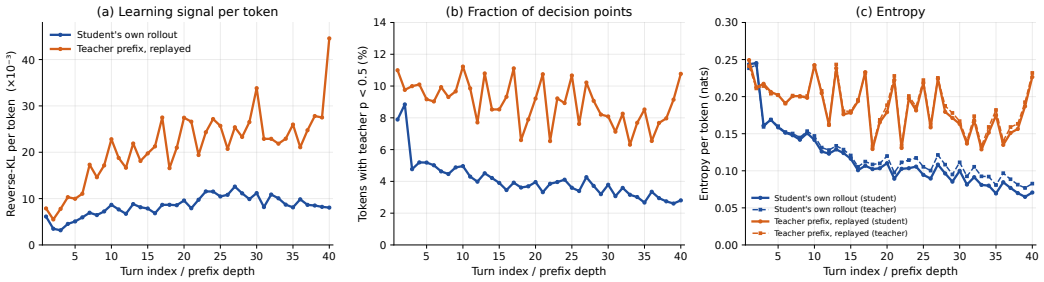

Figure A2: The SFT student scored by the RL teacher on its own rollouts (blue) and on one turn generated from a replayed teacher prefix (orange), as a function of turn index or prefix depth on coding. (a) Per-token reverse KL. (b) Fraction of decision points, tokens with teacher probability below 0.5. (c) Entropy per token of the student and of the teacher at the same states, the teacher drawn with the dashed line. The deepest positions have few samples.

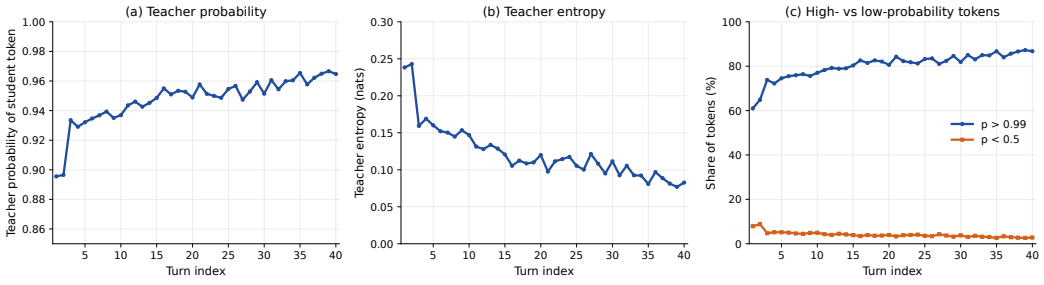

Figure A3: Teacher-side statistics along the SFT student's own rollouts on coding, by turn index. (a) Mean teacher probability of the sampled student token. (b) Teacher entropy at the same positions. (c) Share of tokens to which the teacher assigns probability above 0.99 and below 0.5.

The student leaves the teacher's path within a few turns. The two curves of Figure A2 are measured on different histories, and one may ask how different they are. Figure A4 lets the SFT student and the teacher roll out the same 60 tasks independently and compares their actions turn by

turn. At the first turn both pick the same action on more than nine tasks in ten. By the second turn they still call the same tool about half of the time but rarely on the same file, and by turn 8 the two trajectories share almost no identical action. The tool name alone stays in agreement about half of the time at every depth, which is the agreement of two agents that both alternate between reading and editing.

The drift is a property of long rollouts rather than of the gap between the models. Two seeds of the teacher itself diverge at nearly the same rate as the student and the teacher do. After a handful of turns the student is on states of its own making, and it is on those states, not the teacher's, that a full rollout collects its supervision. The replayed prefix removes this drift by construction, since the state the student is placed on is the teacher's at every depth.

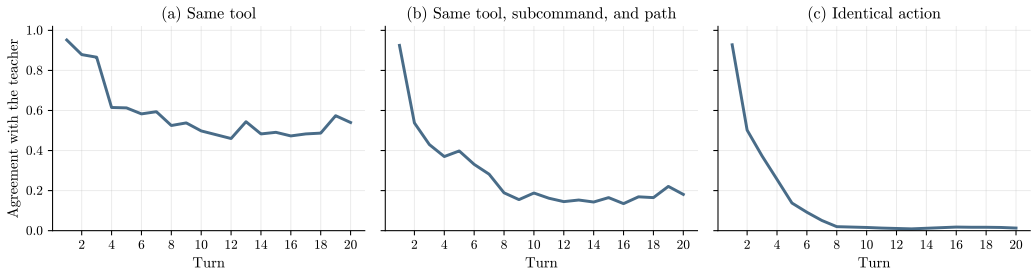

Figure A4: Per-turn agreement between the actions of the SFT student and the teacher when both roll out the same coding tasks independently, V60 with four seed pairs per task. (a) Same tool. (b) Same tool, subcommand, and file path. (c) Identical action including all arguments.

Reasoning tokens carry four times the signal of action tokens, at every depth. Figure A5 distributes the same signal over turn buckets and token types. The first five turns hold about four percent of the total and turns 11 to 20 hold three quarters of it, so the signal is back-loaded as well as sparse. Within every bucket the reasoning tokens dominate. Per token, the reverse KL on reasoning segments is about four times that on action segments, the ratio holds from the first training step and at every depth, and both curves rise with depth. The action tokens of an agent are largely determined once the reasoning has been written. The tool name, the arguments, and the patch follow from the plan, and the SFT student writes them as the teacher would given the same plan. The reasoning is where the branch is chosen.

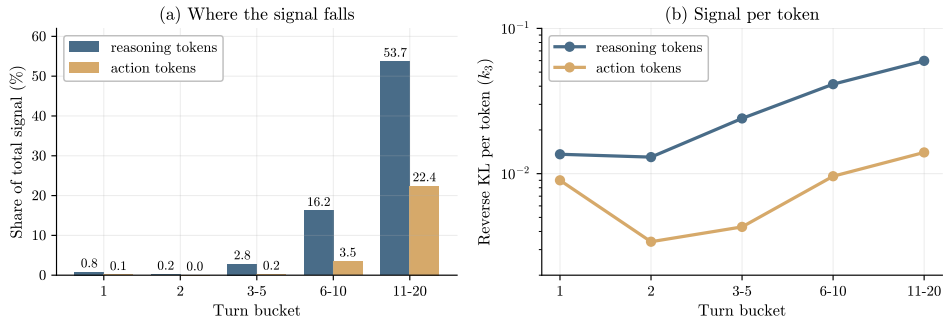

Figure A5: Distillation signal by turn bucket and token type on the 60 coding trajectories. (a) Share of the total k_3 mass held by each bucket, split into reasoning and action tokens. (b) Mean k_3 per token in each bucket.

Reasoning tokens decide, but action tokens must still be scored. If the divergence lies in the reasoning, one might restrict the loss to it. Table A3 shows what happens on the same-size coding

pair after 100 steps. Restricting the loss to reasoning tokens follows the same early loss curve as full distillation, and the loss on the action tokens keeps falling even though they receive no gradient, but the run does not reach the same endpoint, on V60 and on LiveCodeBench alike. Replacing the reverse KL by the forward KL on the action tokens only, by contrast, leaves the outcome unchanged on both coding and tool use.

The action tokens carry little signal, but the reasoning reaches the environment only through them. The form of the divergence on the action tokens is immaterial, yet removing their loss altogether costs two to three points, because the reasoning is translated into behaviour only through the action that follows it. The scores in Table A3 come from the full-vocabulary estimator of Appendix B and are not comparable with the sampled-estimator scores elsewhere in the paper.

Table A3: Restricting or changing the loss by token type, 100 steps. V60 is the 60-task SWE-bench subset with four samples per task. The tool-use column uses an evaluation of BFCL whose start and teacher scores differ slightly from Table 1.

Configuration	Reasoning loss	Action loss	V60	LiveCodeBench	BFCL
SFT start	–	–	17.1	22.9	44.88
Full rollouts	reverse	reverse	25.8	32.1	50.75
Reasoning only	reverse	none	23.3	29.4	–
Mixed	reverse	forward	26.7	–	51.63
Teacher	–	–	–	32.06	54.75

E What RL Changed in the SFT Student

What distillation has to transfer is what RL changed. The teacher on coding is the RL-trained sibling of the SFT student, so we ask token by token what RL changed, on the same 60 trajectories. Every token is assigned to a percentile band of its entropy under the SFT model, and within each band we compare the models before and after a training stage. Two stages are compared on the same footing, the supervised fine-tuning that turned the instruct model into the SFT student, and the RL that turned the SFT student into the teacher. Figure A6 shows the result.

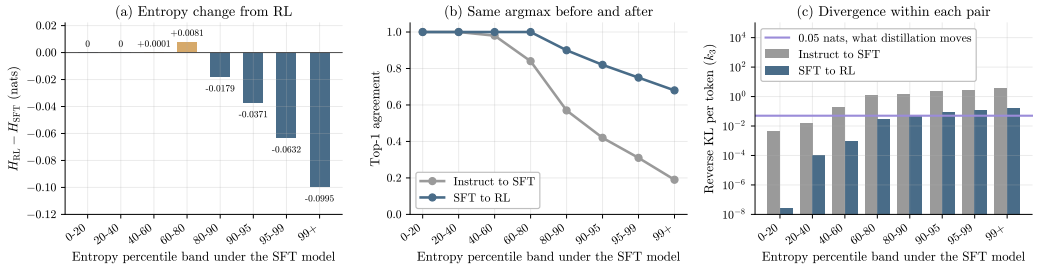

Figure A6: Where training moves the model, by entropy percentile band under the SFT model, measured on 60 coding trajectories with 223,214 tokens. (a) Mean change in token entropy from the SFT student to the RL teacher. (b) Share of tokens on which the two models of a pair select the same argmax, for the instruct-to-SFT pair and the SFT-to-RL pair. (c) Mean per-token reverse KL within each pair on a logarithmic axis, with the level of about 0.05 nats that a course of distillation moves marked for reference.

RL leaves the confident majority of tokens untouched. On the lowest 60 percent of tokens by entropy the teacher and the SFT student are the same model. Their entropies differ by less than 10^{-4} nats, they select the same argmax on every token, and the per-token divergence between them is below 10^{-3}. RL changes the model only in the high-entropy tail, and not uniformly. The band between the 60th and 80th percentile is the only one whose entropy rises, by about 0.008 nats,

whereas every band above the 80th percentile is sharpened, and the sharpening grows with the entropy of the band, reaching a tenth of a nat on the top one percent.

RL changed which branch the model takes when it is uncertain, not which tokens it knows. Two thirds of the total entropy change falls in the top 20 percent of tokens, and it falls twice as hard on reasoning tokens as on action tokens. Even on the most uncertain one percent, the teacher still picks the same argmax as the student on 68 percent of tokens. The instruct-to-SFT pair looks entirely different. Supervised fine-tuning reshaped the whole high-entropy tail, agreement between the instruct model and the SFT student falls to 19 percent on the top band, and the per-token divergence between them is one to two orders of magnitude larger than between the SFT student and the teacher.

The RL gap is small enough for distillation to close. The divergence within the SFT-to-RL pair sits between 0.03 and 0.15 nats on the bands that matter, which is the scale that a short course of distillation moves during training. This is why the SFT cold start of Section 4.2 matters. Supervised fine-tuning brings the student onto the teacher's support and does most of the reshaping, and distillation has only to transfer the residual preferences at the forks. It also explains why a replayed teacher prefix is such a good place to look for signal. The forks are where the teacher's history keeps arriving, and Appendix D shows that the student's own history increasingly avoids them.

F Whose History the Student Stands On

The question is whether the student must have written the history it stands on. Section 2.2 showed that every turn of agentic OPD is trained on a prefix and that the environment enters the objective only through where that prefix comes from. This section asks whether the writer of the history matters, who should write it if not the student, and how long it should be. All experiments are on the same-size coding pair with the sampled estimator, and all scores are on V60, the 60-task SWE-bench subset with four samples per task. The SFT start scores 17.08 on this subset and the teacher 26.8.

A teacher history is as good as the student's own. Every configuration in Table A4 lets the student generate 20 online turns with the environment in the loop and applies the loss to those turns only. What differs is what stands in front of them, which is nothing, the student's own first k turns with their loss masked, or a teacher prefix of k turns replayed from the pool and executed in the container before the student takes over. The masked configuration is the on-policy history of full rollouts with its gradient removed, and the replayed configuration is the off-policy history of Section 3.2 with the student's continuation scored in full. The two are compared at the same k, which places the scored turns at the same depth.

With fifteen replayed turns the two histories give the same result. After 100 steps the student's own masked turns reach 23.34 and the teacher's replayed turns 23.75, and both are within a point of the 22.92 obtained without any prefix. Once the scored turns are the student's own, the student gains nothing from having produced the history itself. A teacher history that places it at the same depth is as good, and by Section 3.1 it carries more decision points per token.

A short replayed prefix hurts, a long one does not. The one configuration that loses is the short teacher prefix. Five replayed teacher turns followed by 20 online turns reach 19.17 after 100 steps, whereas five masked turns of the student's own reach 22.78, and ten replayed turns land in between. A short off-policy prefix displaces the student from its own distribution without yet bringing it to the decisions the teacher learned, so the student spends its online turns recovering from a start it did not choose. A long prefix hands the student a history that the teacher has already carried to the point where the remaining turns are decisions, which is where the single-turn regime of Section 3.2 cuts.

Online continuation washes out the off-policyness of the state. With 20 online turns behind it, the writer of the first 15 turns has no measurable effect on the score. The history only matters when the student generates a single turn on it, because then the state it is placed on is the only state it learns from. This is the setting of the next comparison.

Table A4: Whose history the student stands on, coding, V60. Every configuration allows up to 20 online student turns and applies the loss to those turns only. The prefix is absent, the student's own first k turns with their loss masked, or a teacher prefix of k turns replayed from the pool. All scores are reported after 100 training steps. The SFT start scores 17.1 and the teacher scores 26.8.

Prefix writer	Turns k	Loss on turns	V60 ↑
None	0	1–20	22.9
Student	5	6–25	22.8
Teacher	5	6–25	19.2
Teacher	10	11–30	19.4
Student	15	16–35	23.3
Teacher	15	16–35	23.8

Table A5: Prefix writer on coding, V60, after 100 training steps. Single-turn replay is compared with full rollouts of up to 20 turns and with up to 20 online continuation turns after the teacher's full trajectory. Token shares are relative to full rollouts. The teacher scores 26.8.

Prefix writer / method	Student turns	Sup. tokens	Share	V60 ↑
SFT start	–	–	0.0%	17.1
Full rollouts	20	1.28M	100.0%	22.9
SWE-Master-4B-RL	1	0.077M	6.0%	20.8
Qwen3.5-9B	1	0.085M	6.6%	20.4
GPT-5.4	1	0.157M	12.2%	**22.5**
SWE-Master-4B-RL	20	1.95M	152.0%	21.1

Who wrote the history matters little. Table A5 keeps the student to a single generated turn on a replayed prefix and varies who wrote the prefix. Prefixes from the teacher, from the larger Qwen3.5-9B, and from GPT-5.4 give scores within two points of each other, and the ordering does not follow model size. This agrees with the interpretation of Liao et al. (2026) that a prefix is useful to the extent the teacher can complete it. The GPT-5.4 pool also induces about twice as many generated tokens per turn, so part of its advantage is budget.

One replayed turn recovers most of the gain, and a full replayed trajectory does not add to it. One turn on the teacher's own prefixes recovers two thirds of the full-rollout gain with six percent of its supervised tokens. Replaying the teacher's full 20 turns and scoring the student's continuation, the last row of Table A5, uses more supervised tokens than full rollouts and scores lower than a single replayed turn. The useful part of a prefix is the decision that follows it, not its length.

These comparisons measure the two terms of the ReOPD bound. Liao et al. (2026) bound the gap between Eq. 7 and Eq. 1 by an occupancy term, the distance between d_w^t and d_θ^t, and a reliability term, the quality of the teacher's target on the replayed history. Section 3.1 measures the reliability term directly through the decision points the teacher finds on each kind of history, and Table A4 tests the occupancy term by swapping the student's own masked history for a teacher history. In our same-size setting the replayed prefixes are the teacher's own rollouts, so reliability does not fall with depth and uniform sampling is the schedule that reaches the signal. We expect early sampling to be preferable when the writer is unreliable at depth.

G Long-Horizon Behaviour of the Distilled Students

The distilled students are allowed to run far past their training horizon. The training rollouts on coding stop at 20 turns, whereas evaluation allows 150 steps. Figure A7 follows free-running trajectories on V60 under the evaluation budget and reports, per step bucket, how often the agent repeats an action it has taken within the last five steps, how often it re-opens a file it has already viewed, and how often a tool call fails. Tool errors and re-viewing rise with depth at the same rate for every model, including the teacher.

What separates the models late in a trajectory is repetition. Beyond step 100 the SFT student repeats an action it has just taken on more than four steps in five, whereas the teacher does so on one step in three. The distilled students lie between the two, and the student trained on full rollouts and the one trained on deep replayed prefixes come closest to the teacher. The long-horizon failure of the SFT student is a loop, and it is the loop, rather than tool errors or aimless browsing, that distillation removes.

A student trained on early cuts repeats itself more once it is past the depths it was trained on. The student trained on single replayed turns cut in the first ten turns has seen fewer late states than one trained on deep cuts or on its own 20-turn rollouts, and its repetition rate beyond step 100 lies between the SFT student and the other distilled students. Where the prefixes are cut therefore matters not only for the training signal but for what the student does when it runs long.

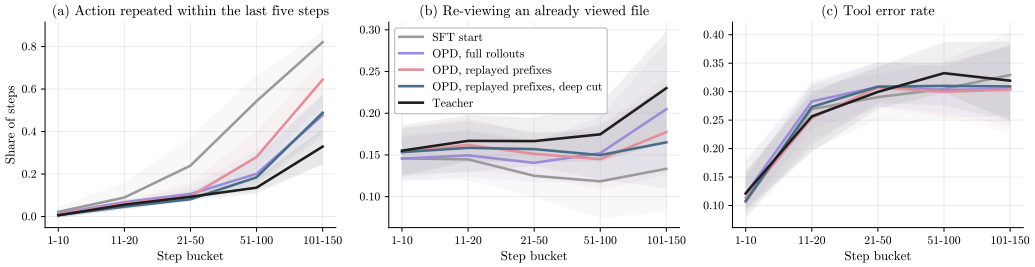

Figure A7: Behaviour along free-running coding trajectories of up to 150 steps on V60, per step bucket, with bootstrap confidence bands over tasks. (a) Share of steps whose action was already taken within the previous five steps. (b) Share of steps that re-open a file the agent has already viewed. (c) Share of tool calls that return an error. Deep cut denotes replayed prefixes drawn from the later half of the teacher's trajectories.

H Additional Cold-Start Results

An overlong continuation is fluent, well scored by the teacher, and circular. Section 4.2 reports that full-rollout distillation of a lightly cold-started student drives the length of its turns to the generation cap while the teacher keeps agreeing with it. Table A6 shows the beginning of an evaluation episode of the 2,000-trajectory student after 300 steps of full-rollout distillation on a tool-use task that asks for the distance between two cities and whether the current fuel suffices. The first call runs to 1,990 tokens of reasoning before it emits two lookups of a zipcode, the second spends another 1,102 tokens to issue the one distance query that the first had already planned, and the whole episode spends 10,945 generated tokens over 14 assistant calls without completing the task.

The loss cannot see the collapse because every sentence of it is one the teacher would write. The reasoning in Table A6 is locally fluent and the teacher assigns it high probability, which is what makes the collapse invisible to the distillation loss. It is also circular. The first call re-derives the same five-step plan four times before acting, and the second call re-derives it again from the tool results, each time interrupting itself with the same doubt about the fuel level. A replayed prefix never lets the student's own earlier turns become the context of its next one, which is why the same student trained on replayed prefixes does not reach the cap in Figure 4.

I Additional Discussion

One replayed turn will not suffice when the teacher's advantage is a multi-turn strategy. Replayed prefixes assume that a reliable prefix exists, that the decisions the teacher learned are local to a turn, and that the student can reach the corresponding states at test time. Tasks that require recovering from the student's own earlier errors, or in which the teacher's advantage is a multi-turn

Table A6: Beginning of one evaluation episode of the 2,000-trajectory student after 300 steps of full-rollout distillation, tool use. Bracketed ellipses mark omitted text. The first assistant call generates 1,990 tokens and the second 1,102, and the episode goes on for twelve more calls without completing the task. The third call opens with the same two sentences as the second.

Role	Content
User	Could you map out the precise mileage from San Francisco to Rivermist for me, along with an assessment of whether my current fuel should suffice the journey?
Assistant, first call	Okay, let's tackle the user's request. First, I need to check the available functions. The function `estimate_distance` requires zipcodes for both cities. But the user provided city names. So I need to get the zipcodes first. After getting the distance, the next step is to check if the fuel is sufficient. The function `estimate_drive_feasibility_by_mileage` takes a distance in miles. But `estimate_distance` returns km. So I need to convert km to miles. Wait, the user's query says "mileage," so maybe the function expects miles. Hmm. [...] To convert km to miles, I can use the `gallon_to_liter` function? No, that's for gallons to liters. Wait, maybe I need a different conversion. [...] So steps: 1. Get zipcode for San Francisco. 2. Get zipcode for Rivermist. 3. Use `estimate_distance`. 4. Convert km to miles. 5. Use `estimate_drive_feasibility_by_mileage`. Wait, but the user might not have provided the current fuel level. [...] So the steps should be: 1. Get current fuel level. 2. Get zipcodes for both cities. 3. Estimate distance in km. 4. Convert km to miles. 5. Use `estimate_drive_feasibility_by_mileage` with the distance in miles and current fuel level? Wait, no. The function takes the distance needed to drive. [...] Therefore, the steps are: 1. `get_zipcode_based_on_city` for San Francisco. 2. `get_zipcode_based_on_city` for Rivermist. 3. `estimate_distance`. 4. Convert km to miles. 5. Call `estimate_drive_feasibility_by_mileage`. Wait, but the user might not have provided the current fuel level. [...] But since the user's query is a single turn, I can only make the initial function calls here. [...] Wait, but the user's query is asking for both the distance and the assessment. [...] Wait, but the user's query is a single turn. The assistant must output all possible function calls in this turn. However, the next functions depend on the previous results. [...] Therefore, the current answer should be the two `get_zipcode` calls.
Tool call	`get_zipcode_based_on_city(city="San Francisco"), get_zipcode_based_on_city(city="Rivermist")`
Tool result	`{"zipcode": "94016"}, {"zipcode": "83214"}`
Assistant, second call	Okay, let's see what the user is asking for. [...] The tool responses gave 94016 for San Francisco and 83214 for Rivermist. Now, the next step should be to use those zipcodes to estimate the distance between them. [...] Wait, the user mentioned "mileage" but the function's parameter is in miles. So first, convert the km distance to miles by dividing by 1.60934. [...] But the user's original query says "current fuel," so I need to know the current fuel level. [...] Hmm, the user might need to first check the current fuel level. But the user's question didn't mention checking fuel, so maybe the assistant should first get the fuel level. Wait, the user says "my current fuel," so the assistant needs to know the current fuel level. [...] So the steps needed are: 1. Get distance between the two cities. 2. Convert that distance from km to miles. 3. Check the current fuel level. 4. Use `estimate_drive_feasibility_by_mileage` with the converted distance and current fuel level? Wait, no. [...] Wait, the previous step already got the zipcodes. So the next step is to call `estimate_distance` with those zipcodes. Therefore, the answer should be the `estimate_distance` function call with the zipcodes.
Tool call	`estimate_distance(cityA="94016", cityB="83214")`
Tool result	`{"distance": 980.0}`

strategy rather than a set of local choices, lie outside this regime, and there full rollouts or the continuation variant remain necessary. Appendix G shows a milder version of the same limit, where a student trained on early cuts repeats itself more than one trained on deep cuts once it runs past the depths it was trained on.

Short trajectories are the other boundary. When there are only a few turns, the position of the cut cannot matter, and replayed prefixes offer only their cost advantage. The two observations of Section 3.1 need depth to appear, and without depth the two regimes supervise nearly the same positions.

The mechanism is not specific to agents. Any long generation in which a small number of decisions determine the outcome, and in which a per-token divergence understates the outcome gap, should show the same two observations and the same benefit from replaying a prefix and scoring one student continuation. Long-form reasoning is the obvious case. Zhang et al. (2026) report exactly this for single-turn reasoning, where distilling on the student's own reasoning prefix and terminating early cuts training compute by an order of magnitude. The agentic case differs in that the environment makes each turn of the student's own search expensive, which is why replaying the teacher's turns rather than the student's is the right unit there.

J Additional Related Work

Prefix-guided RLVR. Off-policy guidance enters RLVR as full traces (Yan et al., 2025), as expert prefixes with on-policy continuations (Huang et al., 2026; Zhang et al., 2025), as partial solutions in the prompt (Li et al., 2026b), as hints annealed over training (Liu et al., 2025; Xie et al., 2026), as natural-language hints on failure (Nath et al., 2025), and as guidance at high-entropy decision points only (Jiang et al., 2026). These works establish that an expert prefix makes hard problems reachable and preserves pass@k; Section 5 relates them to the distillation setting, where the reward is dense and the prefix supplies reliability rather than reachability.