ONLINE SYMBOLIC DISTILLATION: LEARNING FROM TEACHER CONTINUATIONS AT STUDENT STATES

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Online Symbolic Distillation (OSD). At each iteration, the evolving student policy generates a new prefix, the teacher continues it autoregressively, and the student is updated using cross-entropy computed on the teacher-generated tokens. Each design choice targets a corresponding limitation of existing distillation methods: (1) sequential covariate shift in offline supervised fine-tuning (SFT) on fixed teacher trajectories, (2) fragmented supervision under prefix failure in token-level on-policy distillation (OPD), and (3) the need for access to teacher token probabilities in distribution-matching distillation. OSD achieves higher reasoning performance than OPD (with a top-16 KL approximation) at comparable GPU-hour cost. Our asynchronous implementation further reduces OSD's total training time by 23.8%. We evaluate OSD on both hard reasoning tasks and agentic tasks which reflects modern post-training scenarios, and it consistently outperforms existing distillation methods under the same training budget. By regenerating prefixes from the evolving student, OSD continues improving after offline distillation plateaus while better preserving the general capabilities and plasticity of the student. Using only text from GPT-5.4-mini, continuously training with OSD outperforms offline SFT from the same teacher by 13% on ScienceWorld. These results support OSD as an effective and efficient approach to online language-model distillation.

1 INTRODUCTION

Knowledge distillation transfers capability from a stronger teacher to a weaker student, either by matching the teacher's token-level distributions or by training on text the teacher generates (Hinton et al., 2015; Kim & Rush, 2016; West et al., 2022). Offline supervised fine-tuning (SFT) trains the student on fixed teacher trajectories, whereas at inference it conditions on its own outputs. This sequential covariate shift can cause errors to compound over long horizons (Ross & Bagnell, 2010; Bengio et al., 2015). As the student policy changes during training, a fixed dataset also fails to track the states it currently visits. Offline SFT can also degrade the student's prior capabilities (Shenfeld et al., 2026; Chen et al., 2025). Prior work suggests that supervision close to the student's own distribution can improve adaptation and reduce forgetting (Zhang et al., 2026a; Chen et al., 2025).

Recently, on-policy distillation (OPD) has become a promising paradigm for large language model (LLM) post-training (Agarwal et al., 2024; Yang et al., 2025; Lu & Lab, 2025; Xiao et al., 2026). By sampling rollouts from the student policy itself, OPD uses the teacher policy to calculate the reverse-KL loss for each token in the rollout. It thus pairs dense supervision with on-policy states, anchoring learning where the student actually is rather than pulling it toward teacher trajectories (Lu & Lab, 2025). Yet token-level OPD computes teacher targets along each sampled student rollout without revising it. Even when the teacher recommends changing a token, subsequent targets remain conditioned on the student's original continuation. The supervision therefore does not directly demonstrate how to continue from that correction (Jiang et al., 2026). Recent works have also shown that as OPD transfers supervision from teacher at distribution level, it assumes the teacher places meaningful probability mass on the states the student reaches (Zhu et al., 2026; Li et al., 2026c). Such assumption fails once the capability gap between the two policies is too large, and it extends to multi-turn agentic tasks where the irreversible actions made by the student lead the whole trajectory out of the support from the teacher (Wang et al., 2026). Distribution-matching OPD also requires access to teacher token probabilities, limiting its use with teachers that expose only generated text.

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.

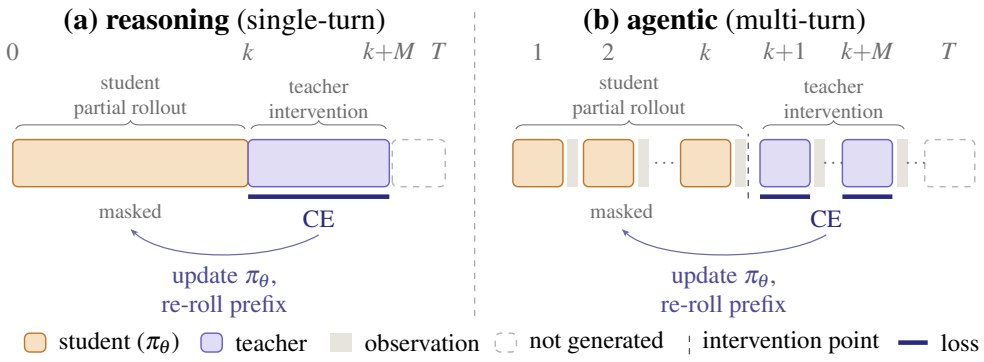


Figure 1: Overview of OSD. **(a) Long chain-of-thought reasoning tasks:** the student generates a reasoning prefix, which the teacher continues. **(b) Long-horizon agentic tasks:** the student interacts with the environment for part of an episode, then the teacher takes over. In both settings, prefixes are refreshed as the student policy changes.

We propose Online Symbolic Distillation (OSD). It trains the evolving student on teacher continuations from student-generated states (Fig. 1). At each iteration, the evolving student policy generates a new prefix, the teacher continues it autoregressively, and the student is updated using cross-entropy computed only on the teacher-generated tokens. For agentic tasks, the prefix consists of the student's actions and the resulting observations; the teacher then takes over interaction with the environment. Repeating this process refreshes the prefixes as the student improves. To reduce the time spent waiting for teacher generation, we implement OSD asynchronously, drawing on asynchronous reinforcement learning (Mnih et al., 2016; Espeholt et al., 2018). Teacher continuation for one batch overlaps with student prefix generation for the next (Fig. 2), reducing total training time by 23.8% relative to synchronous OSD (§4.3).

We evaluate OSD in the two settings at the center of frontier post-training (Team et al., 2026; Xu et al., 2026; Zeng et al., 2026): long chain-of-thought reasoning and long-horizon agentic tasks. For reasoning, we target problems beyond the student's capability, using synthetic tasks from RLVE (Zeng et al., 2025) whose difficulty we can control. For agentic tasks, we use multiple environments from AgentGym (Xi et al., 2025b). Empirically, we find that OSD lifts the performance of the student policy on hard reasoning tasks by 6% to 8% pass@8 points and 7% to 22% avg@4 gains on agentic benchmarks, while only introducing 0.9% average performance drop on general benchmarks (§4, §5.1). The teacher continuation enables OSD to learn where OPD would fail, raising ScienceWorld success to 7.5% from a near-zero student that OPD fails to improve (§4.2). With the nature of CE loss, OSD does not require the teacher logits, enabling black-box distillation (§5.2). Refreshing the student policy online, we show that OSD enables progressive performance improvement as the training proceeds, while offline distillation plateaus and fails to maintain plasticity (§5.2).

Taken together, we present OSD, an online distillation method that trains the rolling student policy on teacher continuations elicited at the states its own prefixes reach. OSD moves the optimization objective in SFT from offline to online, so the supervision is refreshed as the student improves. On top of OPD, OSD demonstrates what to do next from student-visited states, rather than grading past tokens, and needs only teacher text. Empirically, OSD lets training keep improving where offline distillation plateaus, offering better plastcity while causing less forgetting of the capability it already has. Our results suggest that where supervision is placed, and whether it is refreshed as the student changes, is an important axis of distillation design alongside the form the supervision takes.

2 Motivation

Distillation provides supervision through teacher distributions or generated texts (Hinton et al., 2015; Kim & Rush, 2016; West et al., 2022), but its usefulness also depends on the contexts at which that supervision is provided. For difficult reasoning and interaction tasks, we ask: **how should supervision from the teacher connect the student's own attempts to behavior it does not yet generate reliably?**

Teacher continuations at student-generated contexts. Training only on teacher trajectories can leave the student unprepared for situations created by its own decisions, a source of compounding errors in behavioral cloning (Pomerleau, 1991; Ross et al., 2011). OPD addresses this by supervising the student along its own rollouts (Agarwal et al., 2024), but later targets remain conditioned on the student's earlier decisions even where the teacher recommends a different one, so the rollout never demonstrates what would follow that recommendation (Jiang et al., 2026). On-policy supervision also struggles under large capability gaps and when student errors derail multi-turn interaction (Li et al., 2026c; Zhu et al., 2026; Wang et al., 2026). Teacher takeover instead lets the teacher continue from a student-generated prefix, so later decisions, and in interactive environments later observations, follow the teacher's own choices. The student thus learns how to proceed from contexts it actually reaches, without first having to produce the corrective path itself. This mirrors learner roll-in with expert rollout in imitation learning (Ross & Bagnell, 2014) and recent expert-intervention methods for language-model agents (Lauffer et al., 2025; Li et al., 2026a).

Online intervention with a rolling policy. Prefixes collected once reflect the behavior of an earlier student. As training proceeds to update the student policy, the student may encounter different situations and benefit from different continuations. We therefore regenerate prefixes and teacher interventions throughout training, following the learner-state supervision principle of DAgger (Ross et al., 2011). Zhang et al. (2026a) and Zhang et al. (2026b) likewise motivate assessing supervision in relation to the target student and its subsequent learning. We examine whether refreshing intervention contexts improves learning over repeatedly using interventions collected from the initial policy.

Learning through teacher-generated text. Teacher intervention comes in textual forms, learning it only requires cross-entropy loss. The teacher need not expose token probabilities, and its output can be tokenized using the student's tokenizer. Teacher continuation determines the trajectory to learn from; suffix CE makes learning from that trajectory possible through a text-only interface.

Together, these choices motivate OSD as a continuously refreshed procedure for learning from teacher continuations of student attempts. We evaluate its effectiveness on complex reasoning and long-horizon interaction tasks, together with the cost of generating these interventions.

3 Online Symbolic Distillation

We describe OSD in § 3.1 and extend it to multi-turn agentic tasks in § 3.2. We then present an asynchronous implementation that overlaps student sampling with teacher generation to reduce idle time (§ 3.3).

3.1 Main Algorithm

OSD trains a student π_θ from a teacher π_T by collecting student prefixes online, letting the teacher demonstrate how to continue, and learning from teacher text alone. These three design choices address the limitations of offline SFT and token-level OPD, as we explain below using Alg. 1 as a walkthrough.

Algorithm 1 Online Symbolic Distillation (OSD)

Require: Student π_θ, teacher π_T, prompts $\mathcal{D}$, student prefix length k, teacher continuation budget M.
1: **for** each training step **do**
2: Sample prompts $x \sim \mathcal{D}$
3: **Online Roll-in:** $y_{1:k} \sim \pi_\theta(\cdot \mid x)$
4: **Continue:** $\tilde{y} \sim \pi_T(\cdot \mid x, y_{1:k})$, $|\tilde{y}| \leq M$
5: **Update:** Update θ using the masked CE loss in Eq. (1).
6: **end for**
7: **return** π_θ

Collect student prefixes online. To obtain supervision at states reached by the current student, we sample a prompt x from the prompt distribution $\mathcal{D}$ and a k-token prefix $y_{1:k} \sim \pi_\theta(\cdot \mid x)$ (Alg. 1, lines 2–3). We repeat this step after each student update, so the supervision tracks the evolving policy instead of remaining tied to fixed offline trajectories. We use a fixed prefix length k and retain all sampled prefixes without filtering. We report the student and teacher generation lengths for reasoning and agentic tasks in §§ 4.1 and 4.2, respectively.

Demonstrate how to continue. Token-level OPD leaves later targets conditioned on the student's original tokens even when an earlier target recommends a correction. In line 4 of Alg. 1, the teacher

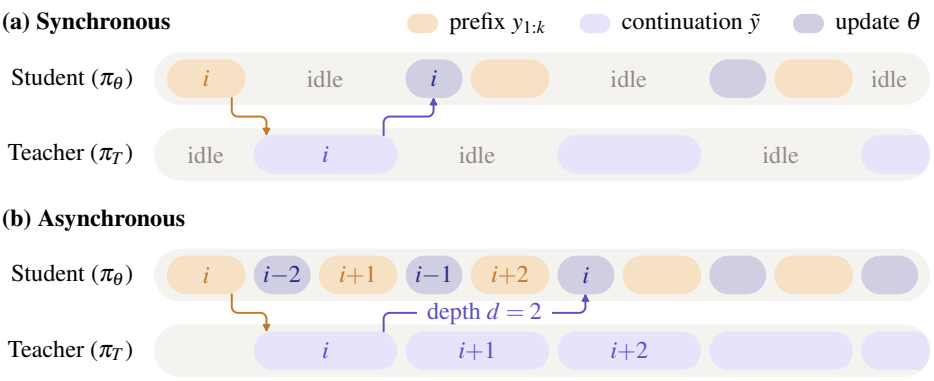


Figure 2: Synchronous and asynchronous implementations of OSD. (a) The student waits for the teacher continuation before updating on the same batch. (b) Prefix sampling for later batches overlaps with teacher generation, while student updates use completed continuations from earlier batches. Arrows connect teacher continuations to the updates that use them.

receives the prompt and student prefix as the input and generates a continuation $\tilde{y} \sim \pi_T(\cdot \mid x, y_{1:k})$ of at most M tokens under its own tokenizer. The teacher conditions each new token on its preceding choices, allowing the continuation to demonstrate how to follow a correction when the student prefix remains recoverable. We use partial continuations to bound the cost of online teacher generation, without requiring a completed, verified solution. As we show in §4.3, OSD achieves higher reasoning performance than OPD at comparable GPU-hour cost with this limited continuation budget.

Learn from teacher text alone. To avoid requiring teacher token probabilities, we train the student with CE on the generated continuation (Alg. 1, line 5). For the student update, we tokenize the teacher continuation with the student's tokenizer, obtaining $\tilde{y}_{1:\ell}$, where ℓ is its length in student tokens. This length may differ from the number of teacher tokens. We feed the full sequence $(x, y_{1:k}, \tilde{y}_{1:\ell})$ to the student and compute

$$\mathcal{L}(\theta; x, y_{1:k}, \tilde{y}_{1:\ell}) = -\sum_{j=1}^{\ell} \log \pi_\theta(\tilde{y}_j \mid x, y_{1:k}, \tilde{y}_{<j}) \,. \tag{1}$$

The prompt and student prefix remain in the conditioning context but are masked out of the loss; the sampled sequences are held fixed during the update. This objective requires only teacher-generated text, so it supports black-box teachers and different teacher and student tokenizers.

3.2 Multi-turn OSD

OSD extends to multi-turn interaction by applying the same prefix–continuation split at the level of turns (Fig. 1b). Here, k and M count interaction turns rather than tokens. The student interacts with the environment for k turns, producing a history of actions and observations. The teacher then takes over for up to M turns, with each action conditioned on the full interaction history and executed in the environment to obtain the next observation. During the student update, we retain the full interaction history as context, mask the student's turns and all environment observations, and apply CE only to the teacher's actions. We find that student prefixes of $k = 5$ or 10 turns, depending on the environment, followed by up to $M = 5$ teacher turns work well (§4.2).

3.3 Asynchronous OSD for Scalable Training

To reduce student GPU idle time, we overlap student prefix sampling with teacher generation (Fig. 2), drawing on asynchronous reinforcement learning (Mnih et al., 2016; Espeholt et al., 2018). While the teacher generates continuations for one batch, the student samples prefixes for the next; completed traces provide training data for the same masked CE objective in Eq. (1). This overlap means that a prefix may come from an earlier student policy than the one being updated. We bound this lag by an asynchronous depth d, the maximum number of student updates between prefix generation and use of

Method	Qwen3-1.7B		Qwen3-4B	
	Pass@8	Avg@8	Pass@8	Avg@8
Original	11.1	3.3	46.1	18.1
Teacher-Gen.	15.0 (+3.9)	5.6 (+2.3)	**53.3** (+7.2)	19.9 (+1.8)
OPD	14.4 (+3.3)	4.6 (+1.3)	45.6 (−0.5)	21.3 (+3.2)
OSD	**19.4** (+8.3)	**7.8** (+4.5)	52.2 (+6.1)	**23.5** (+5.4)

Table 1: Results on RLVE. We report pass@8 and avg@8 on the test set, with two different student models thinking enabled. We use identical teacher model Qwen3-4B-Thinking-2507 for all distillation methods. OSD uses the asynchronous implementation with d=3. All numbers are percentages (%).

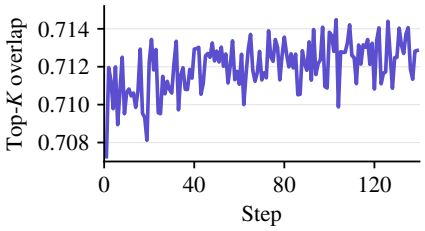


Figure 3: Top-K overlap ratio between student and teacher on validation set during OPD training (Qwen3-1.7B student, Qwen3-4B-Thinking-2507 teacher). It stays nearly flat ($0.707 \rightarrow 0.713$), resonating the finding in Li et al. (2026c).

the resulting trace for training. A larger d allows more overlap but permits greater mismatch between the policy that generated the prefix and the policy being trained. We use $d = 3$ in the reasoning experiments (Tab. 4) and evaluate the efficiency–performance tradeoff in §4.3.

4 Experiments

We evaluate OSD on two main post-training scenarios. We present the experiment in reasoning tasks in §4.1 and multi-turn agentic tasks in §4.2. We further study the training efficiency of OSD in §4.3.

4.1 Reasoning Task

Task. We use RLVE (Zeng et al., 2025) as our primary testbed for single-turn reasoning tasks. RLVE is a synthetic reasoning environment that consists of different reasoning environments, each with verifiable rewards and different difficulty parameters to construct problems with tunable difficulty. It provides a noise-free data collection, training and evaluation pipeline as the instances are not provided during pre-training or post-training of the models themselves to introduce contamination (Shao et al., 2025). Since knowledge distillation targets at introducing new capabilities from the teacher policy to the student policy, we set the difficulty parameters to find the problems that are challenging enough for the student policy to solve. We thus obtain a 18 different reasoning environments subset with 500 training problems for each game, yielding a pool of 9K hard problems. For each task, we pair with 10 test problems with the same difficulty as the training problems.

Models and Evaluation. We consider using models with thinking capabilities that are able to solve the reasoning problems with long-term reasoning generation. To achive this, we use Qwen3-1.7B and Qwen3-4B (Yang et al., 2025) as two student models with their thinking enabled. We use Qwen3-4B-Thinking-2507 (Yang et al., 2025) as the teacher policy since it is the continued scaled model from Qwen3-4B and has stronger reasoning capabilities. We report the pass@8 and avg@8 on the test set.

Training Setup. We report the results of comparison between OSD and other distillation methods, including offline distillation and OPD in Tab. 1. To compare distillation methods under a matched budget, we cap the total number of distilled tokens per rollout at 7168 for both the offline and online baselines. The offline baseline applies cross-entropy on unfiltered teacher trajectories, and the online baseline is OPD. Since OSD does not require the student policy to generate the entire rollout, we set the prefix length to 4096 and the continuation length of 1024 by default. We set the asynchronous depth $d = 3$ for OSD. Detailed experimental settings are provided in the App. A.

Results. We report the results of OSD and comparison across two different student models in Tab. 1. Across both students, OSD achieves the largest avg@8 gains among all distillation methods (+4.42 and +5.35 points). Compare to offline distillation where offline data is collected from the

Method	ALFWorld		ScienceWorld		TextCraft		BabyAI		SearchQA	
	SR (%)	Turns	SR (%)	Turns	SR (%)	Turns	SR (%)	Turns	SR (%)	Turns
Student	19.38	26.97	0.12	28.63	23.00	23.89	38.33	14.52	30.50	11.91
Teacher	52.12	21.24	15.62	23.61	85.50	10.96	83.33	6.22	55.06	9.29
OPD	22.25	26.28	0.00	28.77	29.50	22.28	43.06	14.32	29.56	12.07
TCoD-B2F	37.10	23.30	0.75	29.28	39.50	19.98	66.30	9.03	37.69	11.12
TCoD-F2B	28.00	24.89	0.50	29.55	45.50	18.46	62.50	10.51	37.75	11.19
Guided OPD	27.12	25.26	0.75	29.03	45.50	18.59	**67.78**	**8.30**	37.50	**11.10**
OSD	**40.00**	**23.15**	**7.50**	**23.64**	**55.25**	**16.45**	67.50	9.98	**39.06**	11.13

Table 2: Results on multi-turn agentic benchmarks (ALFWorld, ScienceWorld, TextCraft, BabyAI and SearchQA) with Qwen3-1.7B as the student and Qwen3-32B as the teacher. We report avg@4 success rate (SR, %) and average trajectory score, along with the average turns across the five benchmarks.

teacher policy without filtering, OSD achieves better improvements with less samling from the teacher policy and remain non-filtering. We also include a training dynamic visualiztion of top-K overlap ratio between student and teacher on validation set during OPD training in Fig. 3. Resonating the finding in Li et al. (2026c), the top-K overlap ratio remains nearly flat during OPD training, and little improvement is observed in OPD during training due to different thinking behaviors between the student and the teacher. Overall, supervising the student with teacher continuations from its own states transfers more capability than either training on full teacher trajectories or scoring the student's rollouts, while requiring only a fraction of the teacher's generation.

4.2 Multi-turn Agentic Task

Task. We select 5 multi-turn agentic tasks from AgentGym (Xi et al., 2025a). AgentGym is a multi-turn agentic environment that provides a diverse environments with turn-level feedback for each action the agent takes. Following Xi et al. (2025b), we select 5 environments, including ALFWorld (Shridhar et al., 2020), ScienceWorld (Wang et al., 2022), SearchQA (Dunn et al., 2017), TextCraft (Prasad et al., 2024) and BabyAI (Chevalier-Boisvert et al., 2018). We use the same training and evaluation set for different tasks, except for SearchQA we construct a 6K problems training set with 400 held out problems for evaluation.

Training Setup. We use Qwen3-1.7B as the student model and Qwen3-32B as the teacher model. We compare different online distillation methods on multi-turn agentic tasks, including OPD and subsequent variants that tries to adapt OPD to the multi-turn agentic setting, including two variants of TCoD (Wang et al., 2026) and Guided OPD (Li et al., 2026b). We set the maximum number of turns for ALFWorld, TextCraft and ScienceWorld to 30, 20 for BabyAI and 16 for SearchQA. For each turn we follow the ReAct (Yao et al., 2022) framework to generate the action as AgentGym originally implemented. We set the training epoch for ALFWorld, ScienceWorld and SearchQA to 1, and 3 for BabyAI and TextCraft respectively, since BabyAI and TextCraft have less training data to train the model. For evaluation, we report the average@4 success rate (SR, %), along with the average turns across the five benchmarks. Empirically, we set 10 student turns as prefix for OSD for environments with longer turns like ALFWorld and ScienceWorld, and 5 student turns as prefix for environments with shorter turns. The teacher turns continuation is set to 5 for OSD by default.

Results. We report the results of comparison between OSD and other online distillation methods in Tab. 2. Across different environments, OSD consistently outperforms the other two distillation methods, with each environment showing less turns to achieve higher success rate. Empirically, we observe the biggest performance gain on ScienceWorld, which is the most complex environment among the five according to initial student policy performance. This also explains why OPD does not work well on this environment, as the student action at early turns are likely flawed, making the subsequent turns within the same episode to be incorrect and fail the task. Instead, OSD effectively distill the teacher capabilities to the student policy by directly introducing teacher intervention at given student states, thus correcting the student episode on the right track.

4.3 Training Efficiency

In this section, we study the training efficiency of OSD. We conduct a training comparison between OPD, synchronous OSD and asynchronous OSD on a single turn reasoning task. To be specific, following the setting in Tab. 1, we use the same teacher model Qwen3-4B-Thinking-2507 for all distillation methods, and set student model to be Qwen3-4B. We run training on the 9K training set of RLVE on 8 H200 GPUs, and report the total GPU hours for each method.

We present the results in Fig. 4, and observe that OSD can match the training efficiency of OPD while introducing performance improvement. Since OSD does not require the teacher model to complete the reasoning process or to compute reverse KL on all student generated tokens, it introduces less computation overhead for student rollouts and teacher continuation. We further improves the training efficiency of OSD by using asynchronous training. Asynchronous OSD lets the student policy keep sampling the next batch of prefix rollouts in parallel, and the collected traces are used for the update once the staleness limit is reached. In practice, this significantly reduces the total training time by 28% comparing with OPD. Compared with synchronous OSD, asynchronous OSD only introduces minimal performance degradation while reducing the total training time by 23.8%, which mitigates the effect of additional computation overhead introduced by hosting the teacher model online for sampling.

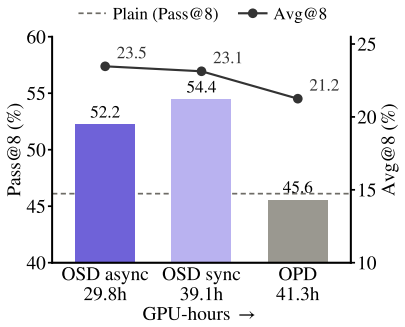


Figure 4: Training comparison between OPD, asynchronous OSD and synchronous OSD.

5 Analysis

5.1 Online Prefixes Learn More while Forgetting Less

OEC (Lauffer et al., 2025) offers an offline variant of OSD, where prefix reasoning content is sampled from the student policy, and then teacher continuation is sampled from the teacher policy. Then they apply a filter to select only the correct reasoning content for finetuning with CE loss and student generation masked. This also resonates SFT baseline, where the teacher model generates full reasoning content and then finetune the student policy on the generated continuations. In this section, we present an analysis study to advocate that **online prefixes enable more effective distillation while forgetting less on general capabilities**, mitigating the exposure bias of offline distillation we mentioned in §1.

We first run teacher 4 times on the same prompts in RLVE, and filter by verifier to get the correct reasoning SFT data for finetuning. We then collect 1 prefix rollout from the student policy for each problem in the training set, and then ask the teacher policy to carry on reasoning as continuation for 4 times, and then filter by verifier, yileding a continuation dataset for finetuning. We finally compare this two baselines with OSD, where instead of only using partial teacher continuation as in Tab. 1, we let the teacher to generate full reasoning content and then filter during online rollouts. The online rollout number for each problem is set to be 4, and we only compute CE loss on the correct reasoning content after filtering.

We report the results in Fig. 5. We observe that OSD achieves a better performance than both baselines, indicating the effectiveness of applying online intervention during training. Instead of collecting the prefix reasoning content directly from the student policy as in OEC, OSD collects the prefix reasoning from the updating student policy during online stage. This effectively boost the performance of distillation by providing more effective supervision during online training. We also measure the forgetting effect of OSD on RLVE. To directly measure the effect of such forgetting between online and offline variants, we test the performance of the finetuned policies on 4 different general benchmarks, including AIME25 (Balunovic et al., 2026) for math tasks, LiveCoding Bench v6 (Jain et al., 2025) for code tasks, IF-Eval (Zhou et al., 2023) for instruction following tasks and GPQA Diamond (Rein et al., 2023) for science tasks. We use avg@16 for math tasks, avg@8 for code tasks, and report the average performance change across the four benchmarks. The results are shown

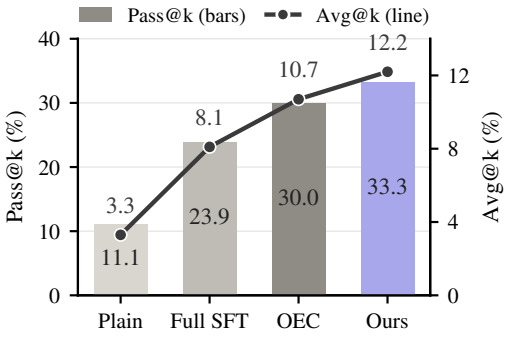


Figure 5: Comparison of OSD with OEC and SFT baselines on RLVE. Obtaining prefix reasoning content from online rolling policy instead of collecting offline prefix or full reasoning content creates effective distillation.

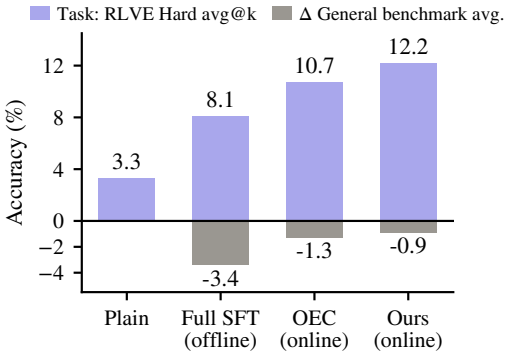
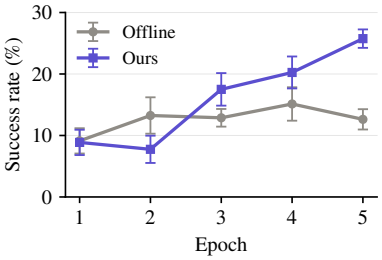


Figure 6: Evaluation of forgetting and task performance gains. Purple bars show pass@8 on the RLVE test set after finetuning. Grey bars show the change in average accuracy on general benchmarks relative to the plain student.

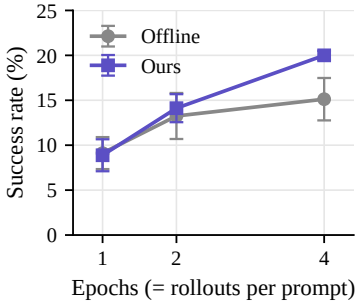
in Fig. 6. We observe that OSD achieves less forgetting on general capabilities while maintaining higher task performance, indicating the advantage of online distillation over offline distillation.

Takeaway. Online prefixes enable more effective distillation than its offline variants, and mitigate the forgetting on general capabilities compared to offline distillation.

5.2 Online Interventions Keep the Student Learning



(a) Over training epochs.



(b) Over rollouts per prompt.

Figure 7: Success rate on ScienceWorld over training 5 epochs and comparison with rollout number, with Qwen3-1.7B as the student and GPT-5.4-mini as the teacher.

In this section we discuss the advantage of optimization under moving policy. With only sampled text from the teacher, suffix CE turns an API model into an online teacher. We use Qwen3-1.7B as the student policy, and GPT 5.4-mini (OpenAI, 2026) as the teacher policy. OPD and logit-based distillation are inapplicable here, leaving offline distillation from the same teacher as the baseline. We use ScienceWorld as the primary benchmark for study this problem. We first collect one episode for each problem in the training set from the teacher policy to get a training set for offline distillation. We then use the same configuration as in §4.2 to train the student policy using OSD. Offline distillation is conducted on static offline data, while OSD collects data as the prefix turns sampling from dynamic student policy. We train the student policy for 5 epochs for both methods, and report the performance as the training progress.

The results are shown in Fig. 7. Training on static offline data leads to a plateau after 2 epochs, as the static data is not updated to follow the moving student policy. Unlike offline distillation where the supervision elicitation context comes solely from static offline prompts, OSD collects supervision from rolling student policy. Such rolling policy naturally introduces diverse and suitable supervision for the student policy (Zhang et al., 2026a). Empirically, although OSD does not perform as well as offline distillation at first two epochs, it keeps improving through the training process, and successfully surpasses the performance of offline distillation by 13% after 5 epochs. OSD is also more flexible in single epoch training. Here we compare the performance of OSD using multi-rollout and compare with corresponding offline epoch checkpoint. We

hypothesize that adding more rollouts per prompt creates same update steps as the offline distillation, yet since directly sampling from the student policy, OSD can collect more diverse and suitable supervision for the student policy. We observe that OSD with multi-rollout performs better than the offline checkpoints, indicating the advantage of distillation during online stage over offline distillation.

We also show OSD preserves plasticity of the student policy than offline distillation methods. For the agentic environments we used in §4.2, we run offline distillation and OSD for 5 epochs with the same configuration in §4.2 sequentially on Qwen3-4B. After this sequential training, we evaluate the performance of the finetuned policy on the test set of each environment, obtaining Fig. 8. Sequentially training on offline data leaves the student less able to learn later environments: the gap is largest on SciWorld, the last environment in the sequence, where OSD improves over offline distillation by 13.0 points. Because later environments are learned by a policy already shifted by earlier stages, fixed offline trajectories increasingly mismatch the states that policy visits, whereas OSD elicits supervision from the current policy's own prefixes. This suggests that maintaining OSD online benefits the student policy to learn more effectively as different tasks proceed, demonstrating the advantage for plasticity preservation for student policy.

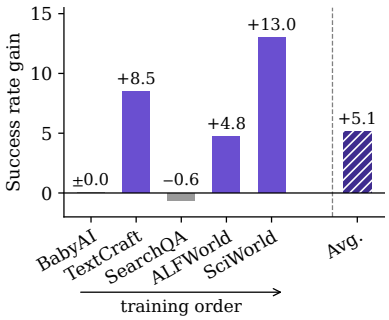


Figure 8: Success rate after 5 epochs of sequential training on each agentic environment.

Takeaway. The online interventions in OSD keep the student policy learning progressively through black-box level access to the teacher policy, while preserving the plasticity of the student policy.

6 Related Work

Knowledge distillation (Hinton et al., 2015) transfers knowledge from a teacher model to a student model by minimizing the KL divergence between the two policy distributions. Subsequently, SeqKD (Kim & Rush, 2016) transfers sequence-level information by optimizing on textual generations from the teacher model. Symbolic KD (West et al., 2022) further extends the idea to selectively distill by designing desired prompts to elicit teacher demonstrations. Recently, many works distill long chains of thought from stronger reasoners (Guo et al., 2025; Ye et al., 2025; Guha et al., 2025) to enhance the reasoning capability of the student model. OPD (Agarwal et al., 2024; Gu et al., 2024; Lu & Lab, 2025) transfers knowledge from the teacher model to the student model by utilizing teacher distribution over student generations, which makes use of student context during distillation to mitigate the distribution mismatch which is common in offline distillation (Ross et al., 2011; Gu et al., 2024). Instead of scoring student generations which fails when the capability gap is large (Zhu et al., 2026; Li et al., 2026c) or at long-horizon tasks (Wang et al., 2026), OSD constructs supervision with student context by online teacher interventions, making it useful when student online rollouts are unreliable and when logit-level supervision is unavailable.

7 Conclusion

We present OSD, a symbolic online distillation method that distills the teacher policy to student via online teacher intervention. OSD first collects online student prefix rollouts, introduces supervision by letting the teacher policy continue, and then calculate loss while masking out the prefix from students. OSD uses refreshed student policy at each gradient step to collect online prefix, and only uses teacher generated text as supervision source. To boost the training efficiency, we further depoly an asynchronous implementation of OSD which lets the student prefix generation and teacher continuation run in parallel. Empirically, OSD achieves 6% to 8% performance improvement on hard reasoning tasks and up to 22% performance improvement on agentic benchmarks, demonstrating effective distillation performance improvement. Further analysis also shows that OSD mitigates the distillation plateauing problem in offline distillation by refreshing student context and introduces less forgetting on general benchmarks. More broadly, our findings suggest that beyond the form of supervision, where and how supervision is placed is also an important axis of distillation design.

AI USE STATEMENT

We used generative AI tools to assist with polishing the writing; the authors verified all content and take full responsibility for it.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38, 2026.

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.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*, 2025.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hamish Ivison, Junjie Oscar Yin, Rulin Shao, Teng Xiao, Nathan Lambert, and Hannaneh Hajishirzi. Tmax: A simple recipe for terminal agents. *arXiv preprint arXiv:2606.23321*, 2026.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Li Jiang, Haoran Xu, Yichuan Ding, and Amy Zhang. Trajectory-refined distillation. *arXiv preprint arXiv:2606.08432*, 2026.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 1317–1327, 2016.

Niklas Lauffer, Xiang Deng, Srivatsa Kundurthy, Brad Kenstler, and Jeff Da. Imitation learning for multi-turn lm agents via on-policy expert corrections. *arXiv preprint arXiv:2512.14895*, 2025.

Changhao Li, Rushi Qiang, Jiawei Huang, Chenxiao Gao, Chao Zhang, Niao He, and Bo Dai. Revisiting dagger in the era of llm-agents. *arXiv preprint arXiv:2605.12913*, 2026a.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, et al. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *arXiv preprint arXiv:2606.15912*, 2026b.

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.

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

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

OpenAI. Introducing GPT-5.4 mini and nano. `https://openai.com/index/introducing-gpt-5-4-mini-and-nano/`, 2026. Accessed: 2026-09-21.

Bespoke Labs OpenThoughts-Agent team, Snorkel AI. OpenThoughts-TBLite: A High-Signal Benchmark for Iterating on Terminal Agents. https://www.openthoughts.ai/blog/openthoughts-tblite, February 2026.

Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991. doi: 10.1162/neco.1991.3.1.88.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.264. URL `https://aclanthology.org/2024.findings-naacl.264/`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/ross10a.html`.

Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey Gordon, and Drew 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, volume 2026, pp. 59839–59864, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

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.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pp. 4602–4625, 2022.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, et al. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, 2025a.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, et al. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning. *arXiv preprint arXiv:2509.08755*, 2025b.

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.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

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.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, et al. Rlve: Scaling up reinforcement learning for language models with adaptive verifiable environments. *arXiv preprint arXiv:2511.07317*, 2025.

Dylan Zhang, Qirun Dai, and Hao Peng. The best instruction-tuning data are those that fit. *Advances in Neural Information Processing Systems*, 38:141172–141208, 2026a.

Dylan Zhang, Yufeng Xu, Haojin Wang, Qingzhi Chen, and Hao Peng. Good sft optimizes for sft, better sft prepares for reinforcement learning. *arXiv preprint arXiv:2602.01058*, 2026b.

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

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026.

A Implementation Details

Hyper-parameter	Value
Training temperature	1.0
Global batch size	64
Mini batch size	64
Student rollouts per prompt	4
LogProb top-K	16
Max response length	7168
Learning rate	1×10^{-6}
Train epochs	1
KL Coefficient	0.0

Table 3: Default configuration of OPD used in our reasoning tasks experiments.

Hyper-parameter	Value
Supervision signal	Token-level CE on teacher continuation
Rollout mode	Online, asynchronous
Maximum staleness	3
Student rollouts per prompt	4
Prefix truncation point	4096
Teacher continuation length	1024
Learning rate	1×10^{-5}
Train batch size	64
Train epochs	1

Table 4: Default configuration of OSD used in our reasoning tasks experiments.

We provide the default configuration of OPD and OSD in Tab. 3 and Tab. 4 on reasoning tasks, respectively. We follow the default configuration of OPD as Li et al. (2026c) did. Tab. 4 lists the full configuration used for OSD in our main experiments. The upper block covers the roadside supervision procedure itself (student rollout, truncation, teacher continuation, and the asynchronous pipeline), and the lower block covers the optimization setup used to train the student on the resulting stitched traces.

B Case Studies

B.1 Multi-turn Agentic Task

We show a ScienceWorld (Wang et al., 2022) trajectory collected during OSD training (Qwen3-1.7B student, GPT-5.4-mini teacher, 10 student turns followed by 5 teacher turns) to demonstrate online intervention works. The student prefix and the teacher continuation are stitched into one trace. The task asks the agent to focus on the longest-lived and then the shortest-lived animal, and the animals are placed `outside`. For ten turns the student cycles through `go to outside`, `open outside`, and `look around`: it treats the location as the object to open and never targets the door, so the score stays at 0. Starting from this prefix, the teacher corrects the action to `open door to the outside` in its first turn and finishes the task four turns later. The student therefore receives supervision on how to recover from the exact failure state it reached on its own, which a teacher-only trajectory starting from the initial observation would not contain.

Case Study on ScienceWorld

Task Find the animal with the longest life span, then the shortest life span. First, focus on the animal with the longest life span. Then, focus on the animal with the shortest life span. The animals are in the `outside` location.
Setup Student **Qwen3-1.7B** (turns 1–10) → Teacher **GPT-5.4-mini** (turns 11–15)
Outcome Score 0.00 at handoff → final score 1.00

Initial observation
This room is called the greenhouse. In it, you see: the agent; a substance called air; a bee hive (door closed); a jug (containing nothing); a shovel; a sink, which is turned off. You also see: a door to the hallway (closed); a door to the outside (closed).

Turn 1 · Student score 0.00
Thought: I need to find the animal with the longest life span and then the shortest life span. The animals are in the 'outside' location. Let me first check the 'outside' location.
Action: `go to outside`
Obs: The door is not open.

Turn 2 · Student score 0.00
Thought: The door to the outside is closed. I need to open it to access the animals.
Action: `open outside`
Obs: The outside is not openable.

Turn 3 · Student score 0.00
Thought: The door to the outside is not openable. I need to check if it can be opened using another method.
Action: `look around`
Obs: (same greenhouse description as the initial observation)

Turns 4–10 · Student (loop) score 0.00
The student repeats turns 1–3 verbatim, with identical thoughts and observations:
Turns 4–6: `go to outside` → `open outside` → `look around`
Turns 7–9: `go to outside` → `open outside` → `look around`
Turn 10: `go to outside` (The door is not open.)

⇄ Handoff: teacher continues from the student prefix (score 0.00)

Turn 11 · Teacher score 0.00
Thought: I need to open the door to the outside, not the location itself.
Action: `open door to the outside`
Obs: The door is now open.

Turn 12 · Teacher score 0.33
Thought: The door to the outside is now open, so I can access the outside area and find the animals.
Action: `go to outside`
Obs: You move to the outside.

Turn 13 · Teacher score 0.33
Thought: I'm in the outside area now. I need to see which animals are present and determine their life spans.
Action: `look around`
Obs: This outside location is called the outside. Here you see: the agent; air; an axe; **a baby dragonfly**; a fire pit; a fountain (containing water); **a giant tortoise egg**; the ground; **a baby wolf**; wood. You also see: doors to the foundry (closed), the greenhouse (open), and the kitchen (closed).

Turn 14 · Teacher score 0.83
Thought: The outside contains a baby dragonfly, a giant tortoise egg, and a baby wolf. The giant tortoise egg is the best candidate for the longest lifespan, so I'll focus on it first.
Action: `focus on giant tortoise egg`
Obs: You focus on the giant tortoise egg.

Turn 15 · Teacher score 1.00 ✔

Thought: Now I need to focus on the animal with the shortest life span. The baby dragonfly is the most likely shortest-lived animal here.
Action: `focus on baby dragonfly`
Obs: You focus on the baby dragonfly.

B.2 Single-turn Reasoning on RLVE

We also show a stitched trace on an RLVE reasoning task. The student prefix (Qwen3-1.7B) is cut at 4096 tokens, and the teacher (Qwen3-4B-Thinking-2507) continues from the same character. [. . .] marks trimmed text and everything else is verbatim. We highlight the wrong step in the student prefix in pink and the correction in the teacher continuation in green. This case illustrates why a short teacher continuation (1024 tokens in our main experiments) is sufficient as supervision: the teacher first completes the half-written path of the student, then corrects the error within its first few sentences.

Case Study on RLVE: Missed Triangle in a Cycle Count

Problem gold answer: 3

You are given an **undirected graph** with 9 vertices, labeled from `0` to `8`. The graph contains the following undirected edges: (2, 3), (2, 5), (0, 8), (1, 3), (0, 4), (0, 1), (0, 7), (3, 8), (1, 8). Please count the number of simple cycles in the graph. A simple cycle is a cycle with at least 3 vertices, with no repeated vertices or edges. [. . .]

Student prefix (Qwen3-1.7B), last part before the cut

[. . .] Wait, what about 0-4-...? No. So maybe there are only two cycles? But that seems low. Maybe I'm missing some. [. . .]
Vertex 1 is connected to 0,3,8. Vertex 3 is connected to 1,2,8. [. . .] Vertex 8 is connected to 0,3,1. [. . .] So maybe there are no other cycles. Then, the answer would be 2 cycles? But that seems low. Let me check again. [. . .]
Another path: 0-1-3-8-0 (cycle), 0-1-8-0 (cycle), 0-1-3-2-5-

⇄ **Cut at student token 4096: teacher continues from the same character**

Teacher continuation (Qwen3-4B-Thinking-2507), first ∼700 characters answer: 3 ✔

2-... no.
Wait, here's an idea: what about the cycle 1-3-8-1? Let's check. 1 is connected to 3, 3 is connected to 8, 8 is connected to 1. So yes! That's a triangle. So 1-3-8-1 is a cycle of length 3. Oh! I didn't think of that. [. . .]
Cycle 1: 0-1-8-0 (triangle); Cycle 2: 1-3-8-1 (triangle); Cycle 3: 0-1-3-8-0 (quadrilateral). [. . .] Is there a triangle 0-3-8? [. . .] 0 and 3 are not directly connected. [. . .]
(Within these first ∼700 characters, the teacher has already led the reasoning onto the correct path: the three cycles it lists are exactly the gold answer.)

C Additional Results

C.1 Intervention Time

We perform ablation studies on OSD during agentic interaction tasks on the length of student turns. In this study, we vary the number of student turns while keeping the number of teacher turns fixed. We use the same setup as in Tab. 2 with Qwen3-1.7B as the student and Qwen3-32B as the teacher on AlfWorld, and report the results in Fig. 9. We set the number of teacher turns to 5 to study the effect of student prefix on the performance of OSD. Adding any student prefix improves the success rate, to 35.6–40.0%, consistent with our main finding that supervision anchored at student-reachable states is more effective than supervision on off-policy teacher states. Beyond this, performance does not increase monotonically with prefix length: 10 student turns performs best (40.0%), while 15 and 20 turns give 35.6% and 39.0%.

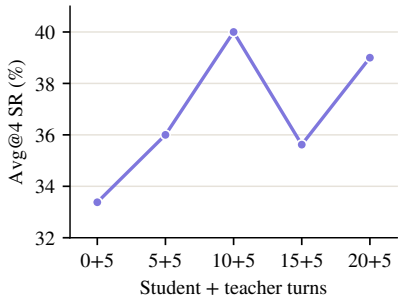


Figure 9: Avg@4 success rate of OSD on ALFWorld with different numbers of student turns before the handoff.

C.2 Generalization to Other Tasks

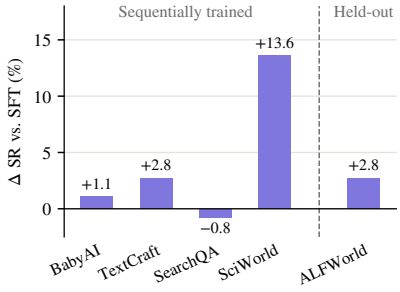


Figure 10: Success-rate gain of OSD over offline distillation. OSD generalizes better.

We perform another sequential training experiment to show OSD can generalize to other tasks. We start with Qwen3-1.7B as the student and GPT-5.4-mini as the teacher, and sequentially train on BabyAI, TextCraft, SearchQA and ScienceWorld. We did not train on ALFWorld during this experiment, and use it as the held out task to evaluate the generalization ability of OSD. We report the success-rate difference between OSD and offline distillation in Fig. 10. We observe that OSD does not only persist the plasticity on sequentially trained environments, but also generalizes to other tasks. After same amount of sequential training, OSD can generalize to other tasks, with average success-rate gain over offline distillation on ALFWorld of $+2.8$.

C.3 Additional Results on Terminal Agents

We further extend the experiments on terminal agent tasks. We select 500 training examples from TMax (Ivison et al., 2026), a terminal agentic dataset that contains over 10,000 terminal agent tasks. We use Qwen3.5-2B as the student and Qwen3.5-9B as the teacher (Team, 2026). For OPD, the student rolls out the first 20 turns and the teacher supervises these turns. For OSD, the student rolls out 10 turns and the teacher continues for another 10 turns, matching the 20-turn budget of OPD. Both methods use 8 rollouts per prompt and a batch size of 16. For evaluation, we randomly sample 50 tasks from TBLite (OpenThoughts-Agent team, 2026), run 4 attempts per task, and report pass@4 in Fig. 11.

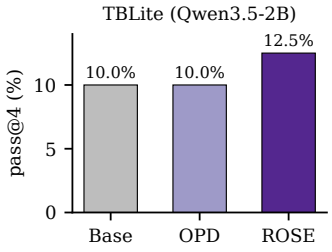


Figure 11: pass@4 on TBLite subsets with Qwen3.5-2B as the student.

As shown in Fig. 11, OPD does not improve over the base student (10.0% for both), while OSD raises pass@4 to 12.5%. Terminal tasks require long horizons, and a 2B student often drifts into states from which it cannot finish the task within the first few turns. OPD only provides token-level corrections on the student's own 20 turns, so when the student is stuck, the teacher distribution conditioned on these failing states provides little signal toward completing the task. In contrast, the teacher continuation in OSD starts from the state the student actually reaches and carries the trajectory forward, which gives the student a demonstration of how to proceed from

its own intermediate states. This is consistent with our findings on the other agentic environments in §4.2. Given the small size of the evaluation subset, we view this result as preliminary evidence that OSD transfers to terminal agents.

D PROMPTS FOR AGENTIC ENVIRONMENTS

We describe the ReAct (Yao et al., 2022) interaction protocol shared by all five agentic environments in §4.2, and list the instruction each environment gives to the model. The same prompts and action parsers are used for student rollouts, teacher continuations, and evaluation.

ALFWorld Instruction

```
Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment
and your target is to perform actions to complete the task goal. At the beginning of your interactions, you
will be given the detailed description of the current environment and your goal to accomplish. For each of
your turn, you will be given a list of actions which you can choose one to perform in this turn. You should
choose from two actions: "THOUGHT" or "ACTION". If you choose "THOUGHT", you should first think about the
current condition and plan for your future actions, and then output your action in this turn. Your output
must strictly follow this format:"Thought:
your thoughts.

Action:
your next action"; If you choose "ACTION", you should directly output the action in this turn. Your output
must strictly follow this format:"Action:
your next action". After your each turn, the environment will give you immediate feedback based on which you
plan your next few steps. if the envrionment output "Nothing happened", that means the previous action is
invalid and you should try more options.
 Reminder:
1. the action must be chosen from the given available actions. Any actions except provided available actions
will be regarded as illegal.
2. Think when necessary, try to act directly more in the process.
```

ScienceWorld Instruction

```
You are an agent for science world. Every round I will give you an observation, you have to respond an action
 based on the observation to finish the given task. Here are the actions you may take: [{"action": "open/
close OBJ", "description": "open/close a container"}, {"action": "de/activate OBJ", "description": "activate/
deactivate a device"}, {"action": "connect OBJ to OBJ", "description": "connect electrical components"}, {"
action": "disconnect OBJ", "description": "disconnect electrical components"}, {"action": "use OBJ [on OBJ]",
 "description": "use a device/item"}, {"action": "look around", "description": "describe the current room"},
{"action": "look at OBJ", "description": "describe an object in detail"}, {"action": "look in OBJ", "
description": "describe a container's contents"}, {"action": "read OBJ", "description": "read a note or book
"}, {"action": "move OBJ to OBJ", "description": "move an object to a container"}, {"action": "pick up OBJ",
"description": "move an object to the inventory"}, {"action": "put down OBJ", "description": "drop an
inventory item"}, {"action": "pour OBJ into OBJ", "description": "pour a liquid into a container"}, {"action
": "dunk OBJ into OBJ", "description": "dunk a container into a liquid"}, {"action": "mix OBJ", "description
": "chemically mix a container"}, {"action": "go to LOC", "description": "move to a new location"}, {"action
": "eat OBJ", "description": "eat a food"}, {"action": "flush OBJ", "description": "flush a toilet"}, {"
action": "focus on OBJ", "description": "signal intent on a task object"}, {"action": "wait", "description":
"take no action for 10 iterations"}, {"action": "wait1", "description": "take no action for 1 iteration"}, {"
action":"examine OBJ","description":"provides a description of the objects present on or in a receptacle."},
{"action": "task", "description": "describe current task"}, {"action": "inventory", "description": "list your
 inventory"}]
Your response should use the following format:
Thought:
your thoughts.

Action:
your next action
```

TextCraft Instruction

```
You are given few useful crafting recipes to craft items in Minecraft. Crafting commands are of the format "
craft [target object] using [input ingredients]".
Every round I will give you an observation, you have to respond an action based on the state and instruction.
 You can "get" an object (ingredients) from the inventory or the environment, look-up the game inventory by "
inventory", or "craft" (target) using any of the crafting commands.
Your output must strictly follow this format:"Thought:
your thoughts.

Action:
your next action"
```

Reminder:
1. Always specify the quantity when using "get" and "craft" commands. - Example of get: get 1 lapis lazuli - Example1 of craft: craft 1 blue dye using 1 lapis lazuli - Example2 of craft: craft 1 golden carrot using 8 gold nugget, 1 carrot
2. When using "get" command, do not specify whether the item comes from the inventory or the environment.
3. You can use ONLY crafting commands provided, do not use your own crafting commands. However, if the crafting command uses a generic ingredient like "planks", you can use special types of the same ingredient e.g. "dark oak planks" in the command instead.

BabyAI Instruction

You are an exploration master that wants to finish every goal you are given. Every round I will give you an observation, and you have to respond an action and your thought based on the observation to finish the given task. You are placed in a room and you need to accomplish the given goal with actions.

You can use the following actions:

- turn right

- turn left

- move forward

- go to <obj> <id>

- pick up <obj> <id>

- go through <door> <id>: <door> must be an open door.

- toggle and go through <door> <id>: <door> can be a closed door or a locked door. If you want to open a locked door, you need to carry a key that is of the same color as the locked door.

- toggle: there is a closed or locked door right in front of you and you can toggle it.
Your response should use the following format:
Thought:
<Your Thought>

Action:
<Your Action>

SearchQA Instruction

You are a question-answering agent with access to a search engine over Wikipedia. Answer the question by interleaving **Thought** and **Action** steps.

Every response must use exactly this format:

Thought: <your reasoning>
Action: <one action>

There are two actions:
search[query]: search Wikipedia. The top 3 passages come back as an Observation.
answer[answer]: give your final answer, as a short phrase (an entity, name, date or number) with no explanation, e.g. answer[Beijing].

Give exactly one action per response and then stop. Do not write the Observation yourself.