Institute of Trustworthy Embodied AI
Fudan University

Tencent Hy

TurnOPD: Making On-Policy Distillation Turn-Aware for Efficient Long-Horizon Agent Training

Yuhang Zhou[1,2], Kai Zheng[2,*,†], Haoling Li[2], Dengyun Peng[1], Can Xu[2,†], Jingjing Chen[1,†]

[1]Fudan University, [2]Tencent Hunyuan

Abstract

On-policy distillation (OPD) trains a student policy by matching a stronger teacher on the student's own trajectories, offering a promising framework for language agent training. However, its application to long-horizon agentic tasks remains insufficiently explored. We identify two key inefficiencies in vanilla agent OPD: (1) full-horizon rollouts often waste wall-clock resources on tail turns that provide weak and noisy KL supervision, and (2) trajectory-level KL objectives concentrate most of the loss on shallow tokens, leaving deeper decision turns under-trained once initial behaviors are aligned. To address these challenges, we propose **TurnOPD**, a turn-level budgeting strategy for efficient on-policy distillation of long-horizon agents. TurnOPD consists of two budget controllers: adaptive rollout-depth budgeting, which uses probe-based turn statistics to determine rollout length, and progressive turn-normalized loss budgeting, which gradually shifts KL weighting from token-level to turn-balanced supervision. Experiments on ALFWorld, WebShop, and Multi-Hop Search with task-specialized teacher models show that TurnOPD achieves superior validation accuracy under equal wall-clock training budgets and advances the accuracy–time frontier beyond vanilla OPD.

Correspondence: ralph.yh.zhou@gmail.com, {kevinezheng,leocaxu}@tencent.com, chenjingjing@fudan.edu.cn

1 Introduction

Language models are increasingly deployed as agents for planning, tool use, and environment interaction [4, 7, 16, 17, 38]. On-policy distillation (OPD) offers a promising training framework for such agentic models [14, 15], where a student samples rollouts and a stronger teacher supervises via a reverse-KL objective at visited states. This keeps supervision on-policy and provides dense feedback while avoiding sparse reward signals.

Yet, applying OPD to long-horizon agent tasks is challenging because these are more than simple sequence-generation problems [24, 36]. Agent rollouts involve multiple turns, tool calls, environmental shifts, and state changes. Early decisions impact all subsequent states, and later turns often contain key but infrequent decisions, so token-level feedback alone may not provide proper supervision for all crucial points.

We systematically analyze OPD in long-horizon agent tasks, asking: *How are supervision signals and optimization budgets allocated along the interaction trajectory, and with what consequences?* Our diagnosis identifies two main problems: **(1) Per-turn KL distribution:** The reverse-KL signal is heavily skewed toward early turns, changes over training, and becomes less informative for deeper steps. Student and teacher outputs often converge as

*Project lead.
†Corresponding authors.

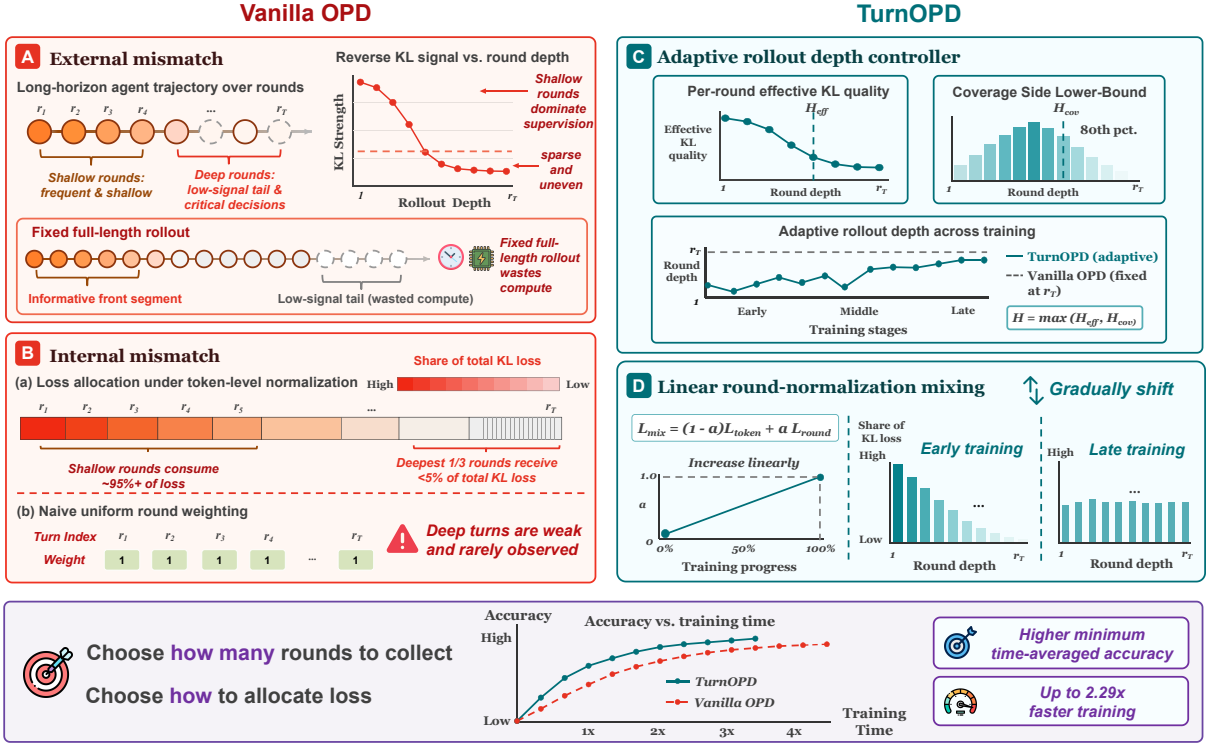

Figure 1 A turn-aware perspective on agent OPD. Standard OPD fixes the rollout depth and averages KL across the full trajectory, often wasting resources on low-value tail turns and over-focusing loss on shallow tokens. TurnOPD budgets both rollout depth and KL aggregation based on turn-level statistics, providing a more balanced supervision signal.

more context is self-generated, reducing observed KL even when meaningful disagreements persist. **(2) Loss budget:** The optimizer's loss is not equitably distributed—shallow turns are overrepresented and carry higher KL, so they dominate the loss, while deep turns contribute little. For example, only 3.6–4.5% of the KL loss is assigned to the deepest third of turns on ALFWorld [23], and 11–13% on Multi-Hop Search. While strict turn-level normalization can mitigate this, it risks over-weighting deep, low-data turns. Overall, standard KL supervision and normalization along full trajectories are insufficient.

These issues stem from two mismatches: (1) **External mismatch:** Fixed rollout depth ignores that both correction signal and survivor count vary markedly with turn. Expending compute up to the maximum horizon wastes effort on steps with little reliable signal. (2) **Internal mismatch:** Trajectory-level normalization gives uniform token weights, concentrating the KL signal on easy, shallow turns and starving deep, informative ones once the model learns basic interaction patterns.

To address these, we propose **TurnOPD**, a turn-level budgeting strategy for efficient on-policy distillation of long-horizon agents. TurnOPD consists of two budget controllers. The **adaptive rollout-depth** controller dynamically selects rollout length via periodic probes, guided by survivor-weighted KL and coverage thresholds. The **progressive turn-normalized loss** controller gradually transitions KL loss aggregation from trajectory-level to turn-balanced, increasing the weight of deep turns.

We evaluate TurnOPD on three representative multi-turn agent benchmarks: embodied planning (ALFWorld), web navigation (WebShop) [32], and Multi-Hop Search (including PopQA [18], NQ [13], 2WikiMultiHopQA [6], HotpotQA [31]). TurnOPD improves minimal-time Avg@4 across all tasks and models. For example, on ALFWorld-1.7B, it increases Same-Step Avg@4 from 83.0 to 86.3 and cuts 100-step wall time from 4.42h to 1.93h. On WebShop, both accuracy and wall time (from 1.57h to 1.24h) are improved. Our ablations further support the decomposition. Adaptive depth is the main efficiency lever: on ALFWorld-1.7B, it

nearly halves wall time but does not by itself solve the loss-allocation problem, while linear KL blending improves Same-Step Avg@4 by reallocating supervision toward deeper turns. Combining both gives the best accuracy–time tradeoff, and coverage-floor studies show that the controller can be tuned smoothly between conservative and aggressive rollout horizons.

In summary, our contributions are as follows:

- We provide a turn-level analysis of OPD for long-horizon agents, revealing the general distribution of supervision and optimization signals across decision turns.
- We formalize one contamination-compression mechanism for the mismatch phenomenon of OPD in long-horizon agent tasks, and demonstrate the existence of a potentially optimal rollout horizon.
- We propose **TurnOPD**, which combines adaptive rollout-depth budgeting with progressive turn-normalized loss budgeting.
- We demonstrate that TurnOPD achieves the best Least-Time accuracy on tested benchmarks, advancing the accuracy–time frontier. Notably, TurnOPD achieves comparable or even better performance with up to 2.29$\times$ faster training.

2 Related Work

On-policy distillation. Early work such as MiniLLM and GKD introduced on-policy distillation (OPD) recipes for LLMs: MiniLLM applies reverse-KL training on student-generated samples [5], while GKD mixes on- and off-policy data and explores various divergence objectives [1]. Later work viewed OPD as KL-constrained policy optimization, with the teacher–student log-ratio as a token-level reward [15, 30], inspiring methods that stabilize target distributions, consider teacher entropy, or relax imitation constraints [8, 11, 12, 35]. Diagnostic studies point out that dense supervision on long trajectories can be unreliable [14, 33]. Still, most work regards each example as a single response, whereas long-horizon agents require finer granularity. TCOD introduces a curriculum over trajectory length [24], and SOD reweights distillation at the step level for tool-assisted reasoning [36], both highlighting that OPD for agents demands more than a flat sequence objective. Our work complements these by analyzing rollout and gradient allocation across turns, and adapting rollout depth and loss accordingly.

Long-horizon agent tasks. Agent benchmarks have shifted from controlled interaction environments to realistic computer-use tasks. ALFWorld links text planning with embodied environments [23], while WebShop studies grounded e-commerce navigation [32]. Later web and GUI benchmarks broaden the setting to real websites and desktop or mobile interfaces, including Mind2Web, WebArena, OSWorld, and AndroidWorld [3, 21, 27, 37]. Recent evaluations further target practical work: SWE-bench measures repository editing from GitHub issues [9], Terminal-Bench evaluates command-line workflows [19], BrowseComp measures persistent web research [25], and TheAgentCompany, GDPval, and ClawBench move toward professional or everyday online tasks [20, 28, 34]. Across these settings, the common structure is not a flat response but a stateful interaction trace with external observations, changing action validity, and changing sets of active trajectories across turns.

3 Preliminaries: On-Policy Distillation

Multi-turn agent interaction. We consider a student agent that interacts with an external environment over multiple turns. At turn t, the agent observes o_t, conditions on the full history

$$h_t = (x, o_1, r_1, o_2, r_2, \ldots, o_t),$$

and generates a model response $r_t \sim \pi_\theta(\cdot \mid h_t)$. The response may contain reasoning text and tool arguments. The environment then maps the executable part of r_t to the next observation o_{t+1}. A rollout is therefore an

interaction trace

$$\tau = (x, o_1, r_1, o_2, r_2, \ldots, o_T, r_T),$$

which terminates when the task is completed, the environment returns a terminal state, or a maximum horizon is reached. This turn structure matters because an early student action changes later observations and thus changes the future contexts on which the teacher will be queried.

Multi-turn OPD. We use on-policy distillation (OPD) [15]: the student samples its own rollout $\tau \sim \pi_\theta(\cdot \mid x)$ for a prompt $x \sim p_{\text{data}}$, and a frozen teacher π_T provides token-level supervision on the student-visited prefixes. For token position i in the concatenated model responses, let s_i denote the full prefix before that token, including previous observations, previous student responses, and the current partial response. This differs from offline distillation because the teacher is queried on histories induced by the current student rather than only on teacher-generated demonstrations.

For a response-token mask m_i, we use reverse KL as the OPD objective:

$$\mathcal{L}_{\text{OPD}}(\theta) = \mathbb{E}_{x,\tau}\left[\frac{1}{\sum_i m_i}\sum_{i=1}^{L} m_i \, D_{\text{KL}}(\pi_\theta(\cdot \mid s_i) \,\|\, \pi_T(\cdot \mid s_i))\right]. \tag{1}$$

4 Diagnosis: Signal Structure in Agent OPD

Long-horizon agent OPD still optimizes a teacher-matching loss over a generated sequence, but the sequence is also an interaction trace. This section gives a turn-resolved diagnosis of **how OPD supervision behaves in long-horizon agent tasks**, focusing on where the teacher signal appears, how reliably it separates outcomes, and how much loss budget it receives.

4.1 Raw Turn-Level KL Exposes Non-Uniform Supervision

Prior OPD and agent-distillation work has emphasized long-trajectory exposure and the need for temporal or step-wise treatment [24, 36]. Our diagnosis asks a complementary question: *How is the reverse-KL supervision itself distributed across turns?* We begin with the least processed measurement: the mean reverse-KL signal at each model turn during vanilla OPD.

Setup. We use ALFWorld [23] and Multi-Hop Search as example tasks. Both diagnostics use vanilla OPD over 100 optimizer steps: ALFWorld with a Qwen3-4B [29] student distilled from a Qwen3-8B-GRPO teacher (ALFWorld-4B), and Multi-Hop Search with a Qwen3.5-2B [29] student from a Qwen3.5-9B-GRPO teacher (Multi-Hop Search-2B). Turn-level means are computed only over trajectories that reach each turn. We plot the longest prefix where each turn is supported by at least 8 surviving trajectories, covers at least 10% of turn-0 cases, and meets these criteria in at least 60% of training steps. This flexible rule allows deeper analysis than a strict cutoff while avoiding noisy tails. Dataset, environment, and teacher model details are in Appendix A.

Result. Figure 2 plots the mean reverse-KL signal at each model turn. Two patterns are visible. First, teacher uncertainty is itself turn-dependent and task-dependent. In ALFWorld, deeper turns often have lower teacher entropy. Multi-Hop Search shows a different entropy profile: later turns can have higher teacher entropy, yet the reverse-KL curves are still front-loaded. Second, the reverse-KL signal is non-uniform and non-stationary over training. On ALFWorld, early turns start with much larger KL and then drop quickly during the first 20–30 optimizer steps, while a long tail of later turns remains lower and noisier. Multi-Hop Search shows a shorter but still structured profile: the first few turns have the largest KL, and the rest settle into a narrower band after the initial alignment period. The immediate conclusion is that vanilla agent OPD cannot be diagnosed by a single average KL. However, the per-turn KL trend itself is ambiguous: late-turn decay is consistent both with genuine alignment and with the teacher signal losing discriminative power. We next test the latter directly.

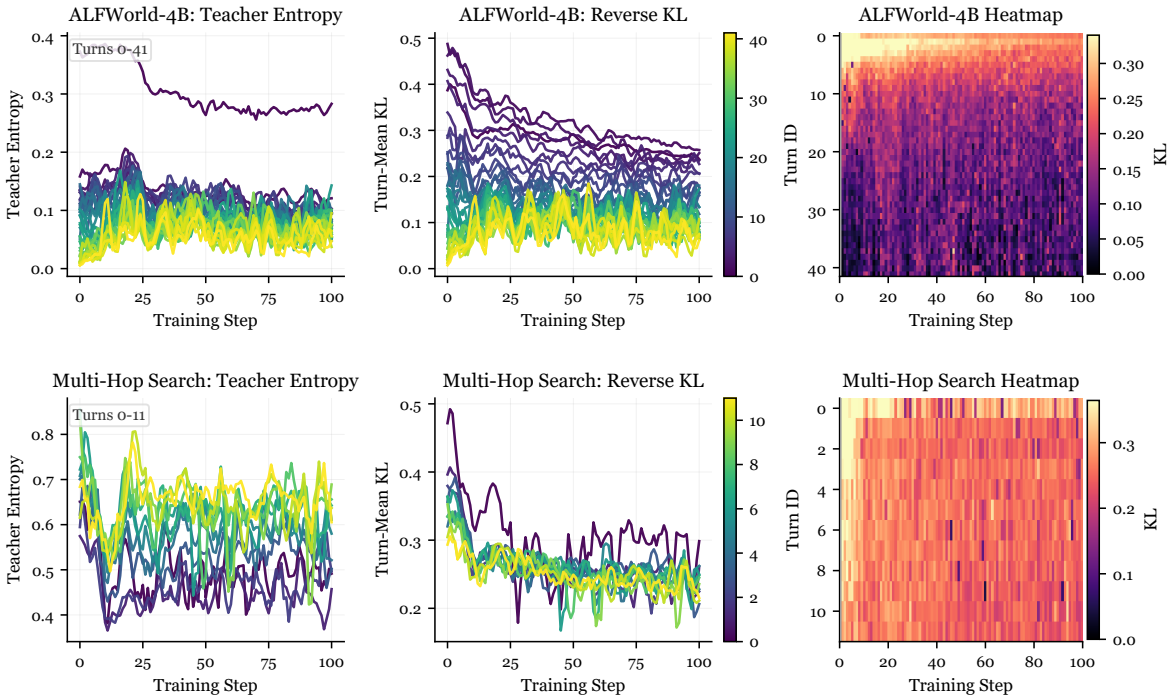

Figure 2 Turn-resolved teacher uncertainty and reverse-KL under vanilla OPD. The ALFWorld row uses the Qwen3-4B student baseline, and the Multi-Hop Search row uses the Qwen3.5-2B student baseline. For each task, the left panel shows smoothed per-turn teacher entropy, the middle panel shows smoothed per-turn reverse-KL, and the right panel shows the raw turn-by-step reverse-KL heatmap. All three panels use the same survivor-supported reliable turn prefix.

4.2 Outcome Separation Degrades and Inverts at Depth

In a multi-turn setting, raw KL is not necessarily a reliable proxy for the remaining student–teacher policy gap. We therefore ask a more outcome-oriented question: at each turn, can KL distinguish rollouts that eventually succeed from those that eventually fail, and how does this distinction evolve over turn indices and training phases? For this purpose, define

$$G_t = K_t^{\text{fail}} - K_t^{\text{succ}}, \tag{2}$$

where K_t^{fail} and K_t^{succ} are the per-turn reverse-KL means computed separately over failed and successful rollouts. If raw KL were a clean local mismatch signal, failed rollouts should generally show larger teacher corrections than successful rollouts at the same turn, leading to positive G_t.

Figure 3 evaluates whether local KL distinguishes successful from failed trajectories. The sign of G_t varies by task, but both show that deep-turn KL lacks outcome-predictive power. For ALFWorld, G_t is mostly negative and decreases with depth: successful trajectories often have higher per-turn KL than failed ones, especially at later turns. This is counterintuitive if KL reflects local correction demand, as failed rollouts should need more correction. We attribute this to branch-dependent compression: failed rollouts quickly fall into loops or template-like patterns where both models find the next-token prediction trivial, reducing KL artificially. In contrast, successful rollouts visit more diverse states where the teacher's corrections are less compressed. In Multi-Hop Search, the gap has the expected positive sign: failed trajectories have larger KL. However, this separation is mostly present in early turns. In later turns, the gap shrinks, demonstrating that KL's discriminative power is concentrated at the front and weakens with depth.

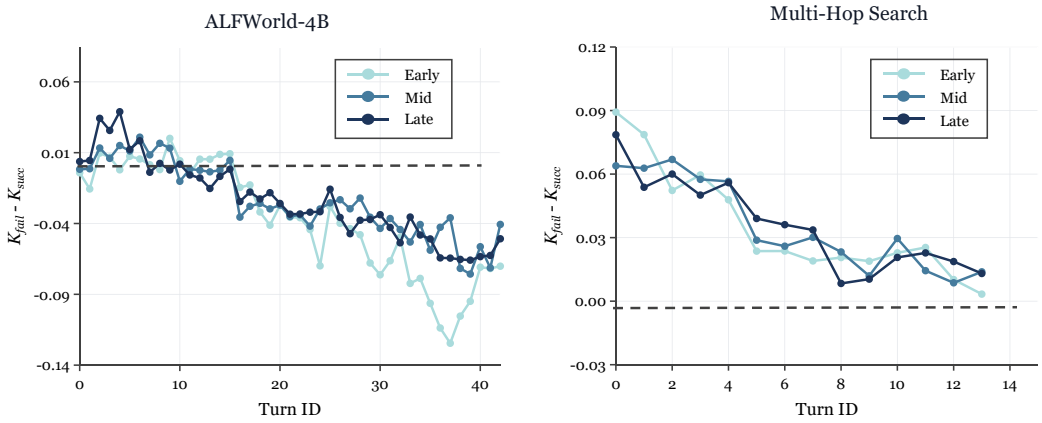

Figure 3 Outcome separation of reverse-KL by turn under vanilla OPD. Each curve plots the phase mean of $G_t = K_t^{\text{fail}} - K_t^{\text{succ}}$ over early ($0 \leq$ step < 30), mid ($30 \leq$ step < 60), or late ($60 \leq$ step ≤ 100) training. Positive values indicate larger KL on failed trajectories.

4.3 A Contamination-Compression Mechanism

The two measurements above leave a specific ambiguity. Raw reverse-KL decreases along the reliable turn prefix, and its outcome-separation power also weakens with depth and can even invert. Therefore a small late-turn KL can have two different explanations: the student may have learned the teacher, or the student-generated context may make both models follow the same local continuation even when their policy-level preferences still differ. We formalize this ambiguity with a contamination-compression model.

Setup. Consider a supervised token position in model turn t, and let c denote the full on-policy context before that token, including previous student outputs and environment/tool observations. Let $\pi_S(\cdot \mid c)$ and $\pi_T(\cdot \mid c)$ be the next-token distributions of the student and teacher under the same context. The observable raw KL at turn t is

$$K_t = \mathbb{E}_{c \sim \mathcal{C}_t} \left[D_{\mathrm{KL}}(\pi_S(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c)) \right], \tag{3}$$

where $\mathcal{C}_t$ is the empirical distribution of supervised token contexts collected at turn t.

For a fixed context c, suppose part of the next-token distribution is nearly forced by the context itself: repeated strings, formatting constraints, copied entities, closing delimiters, or low-level continuations that both pretrained language models assign high probability to. Let $p_F(\cdot \mid c)$ denote this shared forced component. The remaining mass contains the free component, where the student and teacher may genuinely disagree. We write

$$\pi_S(\cdot \mid c) = \lambda(c) p_F(\cdot \mid c) + (1 - \lambda(c)) p_S^{\text{free}}(\cdot \mid c), \qquad \pi_T(\cdot \mid c) = \lambda(c) p_F(\cdot \mid c) + (1 - \lambda(c)) p_T^{\text{free}}(\cdot \mid c), \tag{4}$$

where $\lambda(c) \in [0, 1]$ is the context-forced mass, p_S^{free} is the student's free-component distribution, and p_T^{free} is the teacher's free-component distribution. Define the latent free-component disagreement as

$$\Delta_{\text{free}}(c) = D_{\mathrm{KL}}\big(p_S^{\text{free}}(\cdot \mid c) \,\|\, p_T^{\text{free}}(\cdot \mid c)\big). \tag{5}$$

Equation 4 assumes only that the student and teacher share the context-induced forced component; their free components may differ. In long-horizon settings, increasing student-generated context makes a larger portion of the next-token distribution surface-determined, compressing real decision differences into the remaining free component.

Proposition 1 (Contamination compression)**.** *Under the decomposition in Equation 4,*

$$D_{\mathrm{KL}}(\pi_S(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c)) \le (1 - \lambda(c))\, D_{\mathrm{KL}}\big(p_S^{\mathrm{free}}(\cdot \mid c) \,\|\, p_T^{\mathrm{free}}(\cdot \mid c)\big)\,. \tag{6}$$

A full proof is provided in Appendix B.

Proposition 1 shows why raw KL is not identifiable as pure policy disagreement. If $\Delta_{\mathrm{free}}(c) > 0$, the observable fraction of free disagreement captured by raw KL satisfies

$$\rho(c) = \frac{D_{\mathrm{KL}}(\pi_S(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c))}{\Delta_{\mathrm{free}}(c)} \le 1 - \lambda(c). \tag{7}$$

Thus a high forced mass $\lambda(c)$ can compress the measured KL even when the free-component disagreement remains nonzero.

Why depth can increase compression. In a student rollout, the context at turn t contains all earlier student outputs and tool observations. Autoregressive generation is positively self-conditioning: previous high-probability continuations make some future continuations more predictable, and repeated or format-locked context can increase the forced mass. It provides a structural explanation for the envelope seen in Figure 2: as the rollout becomes longer, the measured KL can decay because $(1 - \lambda(c_t))$ contracts, not necessarily because the free-component disagreement has vanished.

Asymmetric compression and gap inversion. The success/failure gap reveals a stronger version of this mechanism: the context-forced mass need not be symmetric across outcome branches. Failed trajectories can exhibit stereotyped patterns, such as repeated invalid actions, loops over the same subgoal, or template-like continuations. These patterns are exactly the low-level linguistic regularity and local self-correlation captured by the forced component p_F. Consequently, when the teacher is conditioned on a failed context, its next-token distribution can be more strongly locked into the local continuation, making the average forced mass systematically larger on failed rollouts than on successful ones: $\bar{\lambda}_t^{\mathrm{fail}} > \bar{\lambda}_t^{\mathrm{succ}}$. Successful trajectories, in contrast, continue to traverse task states and observations; their contexts remain more information-rich, so the teacher must still make genuine semantic decisions and the forced mass is lower.

For $y \in \{\mathrm{succ}, \mathrm{fail}\}$, let $\bar{\lambda}_t^y$ be the average forced mass and $K_t^{y,\mathrm{free}}$ be the average free-component reverse-KL among turn-t contexts in outcome group y. Reading the compression bound as a first-order approximation gives $K_t^y \approx (1 - \bar{\lambda}_t^y) K_t^{y,\mathrm{free}}$, and therefore

$$G_t \approx (1 - \bar{\lambda}_t^{\mathrm{fail}}) K_t^{\mathrm{fail,free}} - (1 - \bar{\lambda}_t^{\mathrm{succ}}) K_t^{\mathrm{succ,free}}. \tag{8}$$

Equation 8 explains why the gap can become negative. If $\bar{\lambda}_t^{\mathrm{fail}} > \bar{\lambda}_t^{\mathrm{succ}}$, the failed term is suppressed more strongly. Therefore G_t can fall below zero even if the uncompressed disagreement on failed rollouts is at least as large as that on successful rollouts. This also explains why the curve can keep decreasing after it has already become negative. As t grows, failed trajectories accumulate more degenerate context, so $\bar{\lambda}_t^{\mathrm{fail}}$ can keep increasing; successful trajectories remain more task-driven, so $\bar{\lambda}_t^{\mathrm{succ}}$ grows more slowly. The differential $\bar{\lambda}_t^{\mathrm{fail}} - \bar{\lambda}_t^{\mathrm{succ}}$ therefore widens with depth, producing the monotone downward trend observed in Figure 3.

4.4 Loss Aggregation Creates Shallow Budget Concentration

The diagnostics above show **where** local correction signal appears along the turn axis. They do not show **where the optimizer spends its budget**. This distinction matters because vanilla OPD updates the student through raw reverse-KL. We therefore audit the realized KL loss mass assigned to each turn.

Let a trajectory contain T supervised model turns, and let turn t contain n_t supervised tokens with token-level

reverse-KL losses $\ell_{t,i}$. A standard trajectory-level reduction is

$$L_{\text{traj}} = \frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{n_t} \ell_{t,i}, \qquad N = \sum_{t=1}^{T} n_t. \tag{9}$$

This reduction gives every supervised token the same base weight. Its token share at turn t is

$$b_t^{\text{traj}} = \frac{n_t}{N}. \tag{10}$$

The realized share of the batch KL loss mass is instead

$$s_t^{\text{traj}} = \frac{\sum_{i=1}^{n_t} \ell_{t,i}}{\sum_{u=1}^{T} \sum_{j=1}^{n_u} \ell_{u,j}}. \tag{11}$$

With this metric, we can now answer: **under the trajectory-level KL objective, what proportion of the current batch's KL loss mass is contributed by turn t?** We plot the turn-level KL loss-share for the ALFWorld and Multi-Hop Search tasks in Figure 4.

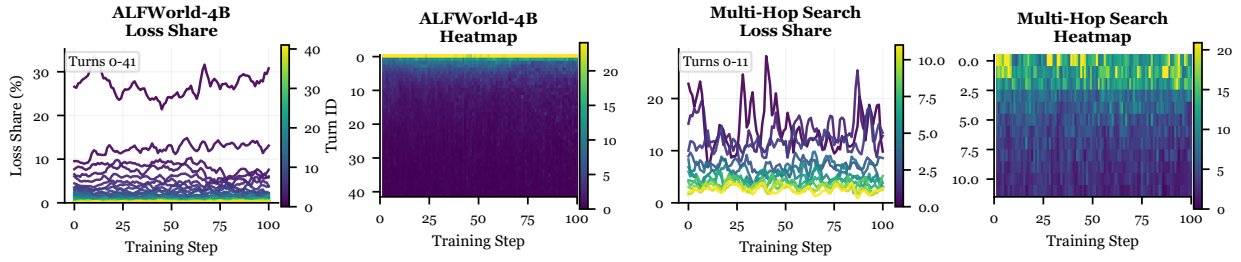

Figure 4 Turn-level KL loss-share under the vanilla trajectory-level KL objective. Line panels show one smoothed training-step curve per reliably observed turn; heatmaps show the raw step-by-turn loss-share values. The metric is the fraction of the current batch's raw KL loss mass contributed by turn t.

Figure 4 shows that the KL loss is mainly concentrated in the early turns. In ALFWorld-4B, turn 0 alone uses about a quarter of the KL budget. The first three turns take nearly half. The reliable deep third receives only 3.6–4.5% of the KL loss. In Multi-Hop Search, the effect is weaker, but the first three turns still take 38–40% of the budget. The reliable deep third gets only 11–13%. This makes sense: not every rollout reaches the deep turns, so there are fewer tokens from deep turns in a batch. The KL loss in shallow turns is also larger. As a result, the KL loss is mostly assigned to earlier turns. The observations lead to two main conclusions:

(1) Trajectory-level normalization creates a budget mismatch. This is not a claim that every deep turn is more valuable. The point is simpler: the trajectory-level reducer spends loss by token mass and realized KL mass. It can therefore keep most updates near the shallow prefix even when later turns still contain correction signal. Table 1 quantifies this with the same runs and the same reliable prefixes as Figure 4.

(2) A hard turn-level replacement may also be too crude. The other extreme gives each observed turn the same total weight. This removes the shallow token-count bias, but it may also ignore reliability. In ALFWorld-4B, the reliable deep third has only 31–42% of the shallow prefix's raw KL. In Multi-Hop Search, the deep third has comparable raw KL, but only about 16–17% survivor support. Equalizing all turns from the beginning would therefore amplify weaker or lower-support estimates too early.

Table 1 Loss-allocation diagnostics from vanilla OPD baselines. Deep/shallow ratios compare the deepest third of the reliable turn prefix with the shallowest third, using the same prefixes as Figure 4. Raw KL is the actual optimization signal. "Deep support" is survivor coverage in the deepest third, and "Deep loss budget" is the fraction of total raw-KL loss mass assigned to that deepest third by the trajectory-level objective.

Task	Phase	Deep/Shallow raw KL	Deep support	Deep loss budget
ALFWorld	early	31%	23.0%	3.6%
ALFWorld	late	42%	18.2%	4.5%
Multi-Hop Search	early	90%	17.3%	12.9%
Multi-Hop Search	late	92%	15.5%	11.1%

4.5 Summary: Two Budget Mismatches

The diagnosis isolates two budget mismatches in vanilla agent OPD:

First, an external mismatch. Rollout depth is fixed, but the survivor-weighted local correction signal and the survivor population both vary sharply with turn depth. Collecting to the maximum horizon can spend rollout compute on turns whose observable correction mass is low or whose estimates are dominated by a small survivor subset. This motivates an adaptive rollout-depth controller that decides **how far to collect**.

Second, an internal mismatch. After a trajectory is collected, trajectory-level normalization starts from uniform token weighting and realizes a shallow KL-mass concentration. However, in agent tasks, turns with few tokens are not necessarily less important. After mastering initial interactions, remaining errors are often in deeper decision turns, yet the token-weighted objective assigns little loss budget to these steps. This motivates a progressive turn-level weighting controller that decides **where the collected loss mass should go**.

These two failures have the same underlying structure: the useful unit of supervision in a long-horizon agent is not a flat token position, but a turn-conditioned decision embedded in an evolving interaction trace.

5 Method: TurnOPD

Based on the analysis in the diagnosis section, we identify two main issues in applying OPD to agents. First, the per-turn KL signal decreases sharply with turn depth, especially in the early training stages. The KL loss is dominated by shallow turns. This suggests that it is not necessary to wait until the end of the full rollout to compute the KL loss. Shortening the rollout horizon can improve overall training efficiency. Second, after a trajectory is collected, trajectory-level normalization assigns loss mass according to token count. This approach can neglect deeper decision turns once the shallow interaction patterns have converged.

In this section, we propose **TurnOPD**, a turn-level budgeting strategy for efficient OPD of long-horizon agents. TurnOPD consists of two budget controllers. The first controller adaptively budgets rollout depth. The second controller progressively shifts KL loss allocation from trajectory-level matching toward turn-balanced decision refinement.

5.1 External Mismatch: Adaptive Rollout-Depth Budgeting

A key limitation identified in the diagnosis is the mismatch between rollout depth and the effective utility of supervision: the optimal trajectory length for data collection should account for both task and student progress to maximize the value of supervision per unit cost. Let $H^\star$ denote this optimal (but unobservable) rollout horizon, the depth yielding the greatest marginal validation improvement for the supervision cost. The existence of $H^\star$ is guaranteed by an efficiency–coverage tradeoff: too shallow a rollout under-explores, while overly deep rollouts waste computation on low-signal regions. A full proof is provided in Appendix C.

However, $H^\star$ cannot be measured directly during training. Instead, TurnOPD estimates it online by synthesizing two complementary signals: an efficiency-centric proxy and a coverage-based lower bound.

Specifically, we define the rollout-depth budget controller as

$$H_{\text{ctrl}} = \max\big(H_{\text{eff}},\, H_{\text{cov}}\big), \tag{12}$$

where H_{eff} captures the centroid of effective supervision and H_{cov} enforces minimum task completion coverage.

Efficiency-side Mass. To construct H_{eff}, we treat the reverse-KL signal over turns as a survivor-weighted distribution of distillation mass. Raw KL may be unreliable as a behavioral-error measure, but still useful as an observable supervision-value and cost proxy. For each turn t, let K_t be the mean per-token reverse-KL on probed student rollouts and n_t/n_0 the on-policy survivor probability. We define

$$m_t = [K_t]_+ \cdot \frac{n_t}{n_0}, \qquad q_t = \frac{m_t}{\sum_j m_j + \epsilon}, \tag{13}$$

where $[K_t]_+$ denotes the positive part of K_t[1]. m_t estimates the expected value of distillation signal contributed by turn t, and q_t normalizes over all turns. The centroid and its discretized projection are then

$$\bar{H}_{\text{eff}} = \sum_t t\, q_t, \qquad H_{\text{eff}} = \text{round}(\bar{H}_{\text{eff}}). \tag{14}$$

This first-moment statistic responds adaptively: when meaningful supervision mass is concentrated in shallow turns, H_{eff} remains shallow; when a correction tail persists at deeper turns, H_{eff} deepens accordingly. Relative weighting ensures robustness to variation in absolute KL magnitude.

Coverage-side Lower Bound. To prevent overly aggressive truncation, the coverage component sets a minimum rollout depth based on successful completions:

$$H_{\text{cov}} = \hat{Q}_p(L_{\text{succ}}) = \min\{H : F_{\text{succ}}(H) \geq p\}, \qquad F_{\text{succ}}(H) = 1 - \frac{n^{\text{succ}}_{H+1}}{n^{\text{succ}}_1}, \quad p = 0.80, \tag{15}$$

where H_{cov} is the p-quantile of the success-conditioned completion depth. This ensures the rollout is deep enough to include at least 80% of successful trajectories, as estimated from periodic probe rollouts.

Dynamic Update. The controller is causal: it updates the uncensored turn statistics for H_{eff} and H_{cov} using periodic full-length probe rollouts, while routine training steps use only the current cap $\hat{H}_k$. Truncated rollouts still contribute to OPD updates, but are excluded from estimating rollout depth, since later turns are unobserved and would bias the statistics if naively included. Periodic full-depth probes are essential to prevent bias and cascading drift that would arise from directly relying on capped trajectories.

After each probe snapshot, TurnOPD sets $H_{\text{ctrl},k} = \max(H_{\text{eff},k}, H_{\text{cov},k})$ and applies exponential moving average smoothing:

$$\bar{H}_k = (1 - \alpha_{\text{ema}})\bar{H}_{k-1} + \alpha_{\text{ema}} H_{\text{ctrl},k}, \qquad \hat{H}_{k+1} = \text{clip}\big(\text{round}(\bar{H}_k) + 1,\, H_{\min},\, H_{\max}\big). \tag{16}$$

The $+1$ converts from a 0-based turn index to the actual rollout depth.

5.2 Internal Mismatch: Progressive Turn-Normalized Loss Budgeting

A second challenge is the internal budget mismatch: with standard trajectory-level normalization, most of the KL loss is assigned to shallow turns, so deeper turns often receive little or no supervision—especially at later training stages, when optimizing long-horizon decisions becomes crucial. This is reasonable early on (when shallow errors dominate), but later hampers correction of harder, deeper decisions.

TurnOPD remedies this by introducing a linear blend between trajectory-level and turn-level normalization,

[1] In practice, K_t can be slightly negative due to noise, so $[K_t]_+$ ensures m_t is stable and non-negative.

shifting the loss allocation over the course of training. Specifically, let n_t be the number of tokens at turn t (with T total turns). The standard trajectory-based weight for turn t is $q_t^{\text{traj}} = n_t / \sum_j n_j$, while the uniform turn-wise weight is $q_t^{\text{turn}} = 1/T$. The final weight is a linear interpolation:

$$q_t^{\text{blend}} = (1-\alpha)\, q_t^{\text{traj}} + \alpha\, q_t^{\text{turn}},$$

where the blend coefficient α is tied to normalized training progress, e.g., $\alpha = \text{clip}((\text{progress} - s)/(e-s),\, 0,\, 1)$ with progress $= k/K$ (step k of K, and typically $s = 0, e = 1$).

Initially ($\alpha = 0$), loss follows token mass, ensuring stability in early stages. As α grows, supervision smoothly shifts to cover all turns more equally, explicitly mitigating turn-depth imbalance and enabling better learning of deep decisions.

5.3 Unification and Algorithm

The two interventions regulate complementary resources: rollout depth determines how much interaction is collected, while loss normalization dictates how supervision is allocated over the collected tokens. For the complete training algorithm and all hyperparameters, see Appendix D.

6 Experiments

We evaluate TurnOPD on three representative agent tasks: ALFWorld (embodied text planning), Multi-Hop Search, and WebShop (web navigation), using three different student-teacher pairs that span multiple model sizes and model families. We test whether TurnOPD achieves a better accuracy–time frontier across different environments and models.

Students and teachers. Students are distilled from stronger, task-specialized GRPO-trained teachers: ALFWorld uses Qwen3-1.7B/4B students with a Qwen3-8B-GRPO teacher; WebShop uses Qwen3-1.7B and Qwen3-8B-GRPO; Multi-Hop Search uses Qwen3.5-2B and Qwen3.5-9B-GRPO. All methods use the same OPD reverse-KL training stack.

Baselines and Metrics. We evaluate three primary methods: vanilla OPD, TCOD-F2B [24], and TurnOPD. To ensure a fair and comprehensive comparison, we conduct evaluations under two regimes: *Least-Time*—where we compare all methods based on accuracies achieved at the minimal wall-clock time required for any method to reach 100 training steps; and *Same-Step*—where each method is evaluated after exactly 100 training steps. Accuracy for each method is reported as the average accuracy over the last four evaluation points (each point is calculated using Avg@4) before the corresponding cutoff. We also report the standard deviation of the accuracy over the last four evaluation points. In addition, we include the performance of the GRPO-trained teacher models as reference points. Note that teacher accuracy is intended for context and not as a baseline for efficiency comparisons.

6.1 Overall Performance

Table 2 and Figure 5 summarize the main results across three long-horizon tasks and multiple student-teacher pairs. Both vanilla OPD and TCOD-F2B serve as strong baselines, but TurnOPD consistently improves the overall accuracy–time tradeoff.

Accuracy. Under the Least-Time setting, TurnOPD achieves the highest overall avg@4 across all task and model combinations among the student-training methods. For the Same-Step setting, TurnOPD matches or surpasses the best baseline in most cases. For instance, on ALFWorld with a Qwen3-1.7B student, TurnOPD reaches 86.29 avg@4, compared to 83.00 for vanilla OPD and 80.06 for TCOD-F2B. On Multi-Hop Search, vanilla OPD keeps a small Same-Step advantage, but TurnOPD gives the best Least-Time accuracy, showing that it reaches stronger performance under the shared wall-clock budget. Tables 3 and 4 further provide the task-specific breakdowns. On ALFWorld, TurnOPD is the strongest student-training method on nearly all categories for both student sizes. On Multi-Hop Search, TurnOPD improves PopQA and 2Wiki and gives the

Table 2 Main comparison across tasks. Overall accuracy is reported as Avg@4 before the corresponding cutoff; the smaller ± term reports the standard deviation across the same four evaluation checkpoints. Wall time is the cumulative per-step training time through 100 optimizer steps, and speedup is relative to vanilla OPD within each task–student group. **Bold** numbers mark the best student-training method.

Task	Student	Teacher	Method	Overall Avg@4 Least-Time	Overall Avg@4 Same-Step	Same Step Wall (h)	Same Step Speedup
ALFWorld	Qwen3-1.7B	Qwen3-8B-GRPO	Zero-Shot	0.00	0.00	–	–
			Teacher	90.75	90.75	–	–
			Vanilla OPD	73.52±1.84	83.00±1.69	4.42	1.00×
			TCOD-F2B	80.06±1.43	80.06±1.43	**1.87**	**2.37**×
			TurnOPD	**85.60**±0.95	**86.29**±0.48	1.93	2.29×
ALFWorld	Qwen3-4B	Qwen3-8B-GRPO	Zero-Shot	8.17	8.17	–	–
			Teacher	90.75	90.75	–	–
			Vanilla OPD	90.79±0.61	91.81±0.83	2.86	1.00×
			TCOD-F2B	86.50±1.88	86.50±1.88	**1.89**	**1.51**×
			TurnOPD	**91.73**±1.41	**92.21**±0.42	2.16	1.33×
Multi-Hop Search	Qwen3.5-2B	Qwen3.5-9B-GRPO	Zero-Shot	36.08	36.08	–	–
			Teacher	59.00	59.00	–	–
			Vanilla OPD	45.77±0.48	**47.82**±0.92	4.45	1.00×
			TCOD-F2B	45.64±1.06	47.77±1.21	3.80	1.17×
			TurnOPD	**47.24**±1.03	47.24±1.03	**2.94**	**1.51**×
WebShop	Qwen3-1.7B	Qwen3-8B-GRPO	Zero-Shot	25.80	25.80	–	–
			Teacher	84.46	84.46	–	–
			Vanilla OPD	76.98±0.79	81.65±1.47	1.57	1.00×
			TCOD-F2B	80.45±1.57	81.66±0.89	1.33	1.18×
			TurnOPD	**82.80**±0.75	**82.80**±0.75	**1.24**	**1.26**×

Table 3 ALFWorld category-level accuracy under the Least-Time protocol. Category scores report avg@4 before the same wall-clock cutoff used in Table 2.

Student	Method	2Obj	Heat	Cool	Clean	Place	Look	Overall Avg@4 Least-Time
Qwen3-1.7B	Zero-Shot	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Teacher	92.50	93.40	84.09	92.34	91.49	91.28	90.75
	Vanilla OPD	76.41±2.77	73.82±2.15	65.45±3.51	73.39±2.57	75.93±2.27	78.34±2.68	73.52±1.84
	TCOD-F2B	82.66±2.70	80.19±1.20	74.43±3.63	79.03±0.99	82.18±2.14	83.87±1.76	80.06±1.43
	TurnOPD	**87.97**±0.27	**85.97**±2.09	**80.34**±0.38	**84.58**±1.55	**87.10**±2.67	**89.53**±2.06	**85.60**±0.95
Qwen3-4B	Zero-Shot	4.38	5.19	3.64	5.24	11.70	21.51	8.17
	Teacher	92.50	93.40	84.09	92.34	91.49	91.28	90.75
	Vanilla OPD	92.97±0.52	89.50±0.39	90.34±2.21	90.02±1.08	**91.49**±1.95	91.28±0.92	90.79±0.61
	TCOD-F2B	88.91±1.20	85.26±2.86	85.00±2.51	84.17±2.29	87.23±3.24	90.26±1.86	86.50±1.88
	TurnOPD	**94.53**±2.44	**91.04**±1.45	**90.68**±0.75	**91.94**±2.26	90.96±1.36	**91.86**±2.36	**91.73**±1.41

best overall Least-Time score. Notably, with the Qwen3-4B student, TurnOPD even exceeds the ALFWorld teacher reference in overall avg@4.

Training Efficiency and Wall-Clock Speedup. Beyond accuracy, TurnOPD also reduces training time. On ALFWorld-1.7B, TurnOPD reduces the 100-step wall-clock time from 4.42 hours for vanilla OPD to 1.93 hours while improving accuracy. Similar improvements appear on Multi-Hop Search (2.94h vs. 4.45h) and WebShop (1.24h vs. 1.57h). Thus TurnOPD is not only more accurate under the common wall-clock budget, but also substantially accelerates vanilla OPD.

In summary, these comprehensive results establish TurnOPD as a strong performer on both accuracy and efficiency axes. Its turn-level budgeting approach yields robust accuracy improvements with less computation, making it attractive for both research and real-world applications where computational resources are a critical constraint.

6.2 Diagnostic–Controller Alignment

The main results confirm that TurnOPD effectively improves both the accuracy and compute efficiency frontiers. To further validate the effectiveness of the adaptive controller, we examine whether the controller's behavior aligns with our diagnostic metrics that motivated its design.

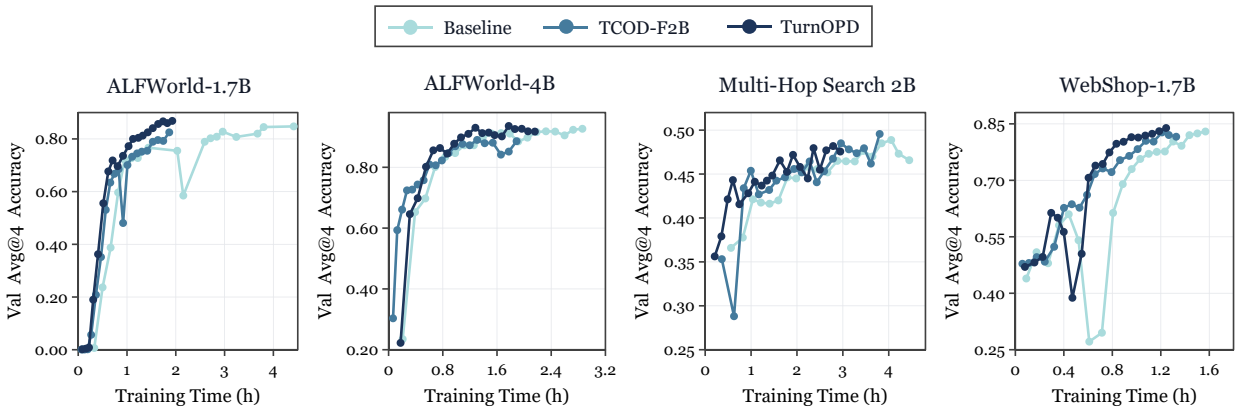

Figure 5 Main-result iso-time efficiency. The x-axis is cumulative training time, computed as the running sum of per training step time. TurnOPD improves the accuracy–time frontier on each task type.

Table 4 Multi-Hop Search category-level accuracy under the Least-Time protocol. Category scores report avg@4 over the four held-out QA sources.

Method	PopQA	NQ	2Wiki	HotpotQA	Overall Avg@4 Least-Time
Zero-Shot	38.25	29.00	39.25	37.81	36.08
Teacher	59.50	52.50	72.00	52.00	59.00
Vanilla OPD	48.00±0.31	**40.88**±1.07	50.81±1.15	**43.39**±0.53	45.77±0.48
TCOD-F2B	47.81±1.01	39.38±0.38	53.25±3.05	42.12±1.26	45.64±1.06
TurnOPD	**49.38**±1.64	40.69±1.12	**56.00**±2.49	42.89±0.98	**47.24**±1.03

Figure 6 illustrates the dynamics of TurnOPD's rollout-depth adaptation. Specifically, we plot the survivor-weighted raw-KL centroid H_{eff} (representing the efficiency arm) and the success-conditioned completion quantile H_{cov} (representing the coverage arm) as defined in Section 5.1. The controller adaptively targets the maximum of these two quantities at each step. During initial warm-up phases before sufficient successful rollouts appear, H_{cov} is set to zero to ensure that the observed KL centroid remains visible. Across tasks, we observe distinct controller behaviors: (1) in ALFWorld, the coverage constraint (H_{cov}) quickly dominates once successful trajectories emerge, encouraging longer rollouts; (2) for WebShop, the controller maintains a moderate coverage lower bound throughout; and (3) in Multi-Hop Search, the balance shifts dynamically between the efficiency and coverage arms. These plots confirm the intended division of labor: H_{eff} ensures nontrivial efficiency, while H_{cov} prevents the controller from selecting overly short rollouts that would limit coverage of successful completions.

7 Ablation Studies and Analysis

The previous section demonstrates that the learned controller tracks the diagnostic signals underlying the design of TurnOPD. We now turn to a more fine-grained question: How do individual budget controllers in TurnOPD contribute to its overall performance?

To answer this, we conduct ablation studies on ALFWorld-1.7B (Qwen3-1.7B distilled from Qwen3-8B-GRPO on ALFWorld task) using the same 100-step protocol as in the main experiment. The analysis proceeds in three parts. First, we disentangle the two core interventions to evaluate whether each is effective on its own. Second, we ablate the KL normalization scheme, comparing trajectory-level, strict turn-level, and linear turn-level reductions. Third, we analyze the effect of altering the coverage floor, which determines the conservativeness of the adaptive-depth controller.

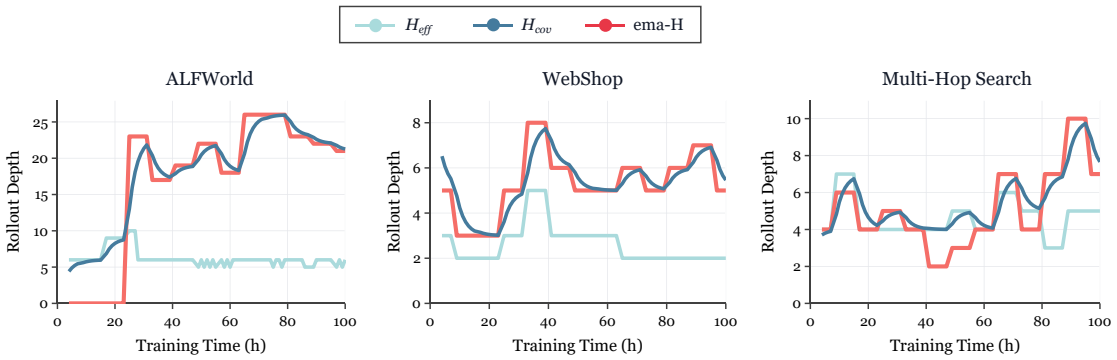

Figure 6 Rollout-depth diagnostic replay. For each TurnOPD run, we plot the survivor-weighted raw-KL centroid H_{eff}, the success-coverage lower bound H_{cov}, and the counterfactual EMA horizon $\hat{H}_{\text{EMA}}$ obtained by replaying $\max(H_{\text{eff}}, H_{\text{cov}})$. Values use the 0-based turn-index convention; the applied rollout cap is approximately $\text{round}(\hat{H}) + 1$.

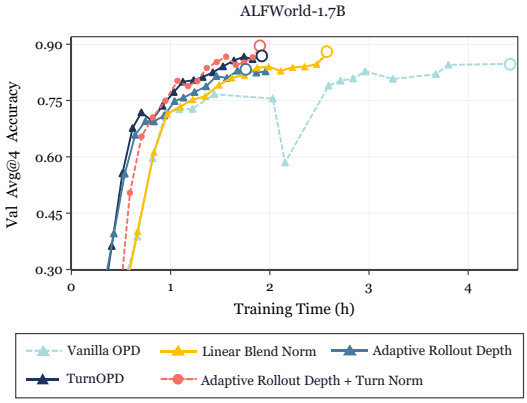

Figure 7 ALFWorld-1.7B component ablations.

Table 5 Intervention decomposition on ALFWorld-1.7B over the total 100 optimizer steps. Acc reports Same-Step Avg@4, averaging the last four evaluation checkpoints. Wall time sums per-step training time over all training steps.

Config	Depth	Blend	Avg@4	Wall (h)
Vanilla OPD			83.0	4.42
+ Adaptive Depth	✓		82.8	1.96
+ Linear blend norm		✓	85.1	2.59
TurnOPD	✓	✓	**86.3**	**1.93**

7.1 Intervention Component Decomposition

This subsection analyzes the individual contribution of each TurnOPD budget controller to overall performance.

Setup. We conduct experiments on the ALFWorld task. The *Adaptive Depth* variant applies the adaptive rollout-depth budget controller while keeping the original trajectory-level KL reduction unchanged, to test if simply shortening rollout depth is sufficient. In contrast, the *Linear blend norm* variant maintains the full-horizon OPD rollout depth but replaces the KL reduction with a linear turn-level normalizer, assessing whether rebalancing loss allocation alone can be effective. The full TurnOPD approach integrates both budget controllers. We report the overall Avg@4 accuracy under full training, averaging the last four evaluation checkpoints.

Results. Figure 7 presents the full validation curves across training, while Table 5 summarizes the Same-Step Avg@4 accuracy and cumulative wall time after 100 optimizer steps. The results support the intended division of labor. Adaptive depth alone cuts the 100-step wall time from 4.42 h to 1.96 h, but its accuracy drops slightly from 83.0 to 82.8. This indicates that reducing rollout depth is not a complete training solution by itself: it saves compute, but it does not fix the loss allocation problem identified in Section 4.4. In contrast, the linear KL blend improves accuracy to 85.1, showing that the loss allocator directly improves optimization, but it still costs 2.59 h. Full TurnOPD combines the two effects: it achieves the best average accuracy (86.3) while matching the low wall time of the adaptive-depth run (1.93 h). Notably, using adaptive rollout depth together with turn-level KL normalization alone can sometimes yield final performance close to, or slightly

higher than, TurnOPD. However, TurnOPD outperforms these alternatives in the middle stages of training, demonstrating faster progress and better intermediate results. Thus, the ablations tell a coherent story: the rollout-depth budget controller provides the efficiency lever, the progressive loss-budget controller provides the optimization lever, and their combination gives the best accuracy–time tradeoff.

7.2 Effect of KL Normalization

Motivation and Experiment Setup. Section 4.4 points out that KL loss normalization has a budget allocation problem: the standard (trajectory-level) approach mostly focuses on shallow turns, while a hard turn-level strategy can overcompensate and put too much weight on rarely supported deep turns. To better understand this, we compare three KL normalization schemes: **trajectory-level KL**, **hard turn-level KL**, and our **linear blend method**. For fairness, we report each method's Same-Step Avg@4 and the actual loss budget spent on the deepest third of valid turns (i.e., turns reached by enough trajectories).

Results. As shown in Table 6, trajectory-level KL is stable but strongly favors shallow turns: the deepest third gets only 3.2/0.7/1.2% of the budget in early/mid/late training. Hard turn-level KL sharply reverses this, instantly giving around a third of the budget to deep turns, but makes an abrupt shift and may overweight unreliable estimates early on. Our linear blend offers a smoother transition: α rises from 0.17 to 0.83 across the early/mid/late phases, while the deep-turn budget increases from 12.8% to 27.7%. This method balances stability and targeted allocation, yielding the best Same-Step Avg@4, though the margin over hard turn-level KL is small.

Table 6 Impact of KL normalization strategies on ALFWorld-1.7B. "Same-Step Avg@4" is the mean over the last four evaluation checkpoints. "Deep budget" gives the KL budget assigned to the deepest third of valid turns in early/mid/late training. "α" is the blend coefficient, with $\alpha = 0$ for trajectory-level KL and $\alpha = 1$ for hard turn-level KL.

KL normalizer	Same-Step Avg@4	Deep budget E/M/L	α E/M/L
Trajectory-level KL	83.0	3.2/0.7/1.2%	0/0/0
Turn-level KL (hard switch)	85.0	29.9/32.0/31.9%	1/1/1
Linear blend (ours)	**85.1**	12.8/21.6/27.7%	0.17/0.50/0.83

7.3 Adaptive-Depth Coverage Sensitivity

The Adaptive Rollout Depth module exposes two main hyperparameters: (1) the coverage quantile p, and (2) the choice of whether the completion cumulative distribution function (CDF) is computed from only successful trajectories or from the full trajectory population. In practice, many open-ended agentic tasks do not provide a reliable distinction between successful and unsuccessful trajectories. Moreover, the OPD algorithm itself does not fundamentally depend on a reward signal denoting task success. To assess robustness in more general settings, we conducted additional experiments evaluating the adaptive-depth controller without access to a success-conditioned CDF.

For this hyperparameter study, we swept $p \in \{0.4, 0.6, 0.8\}$ across both success-conditioned and full-population CDFs. Figure 8 summarizes four diagnostic metrics: validation accuracy and EMA-smoothed rollout horizon (ema-H) for both the success-conditioned and full-population CDF settings. Here, ema-H refers to the exponentially smoothed average rollout depth, which directly determines the effective student rollout horizon. We summarize the main observations as follows:

(1) The quantile parameter p directly controls the target rollout horizon. Increasing p yields slightly higher final success rates, but also results in deeper rollout horizons and correspondingly increased training time. For example, with the full-population CDF, setting $p = 0.6$ achieves an accuracy of 85.1 in 1.66 hours, while $p = 0.8$ improves the accuracy to 85.8 but increases the required time to 1.83 hours.

(2) The CDF source significantly affects efficiency. The full-population CDF is more aggressive: it consistently

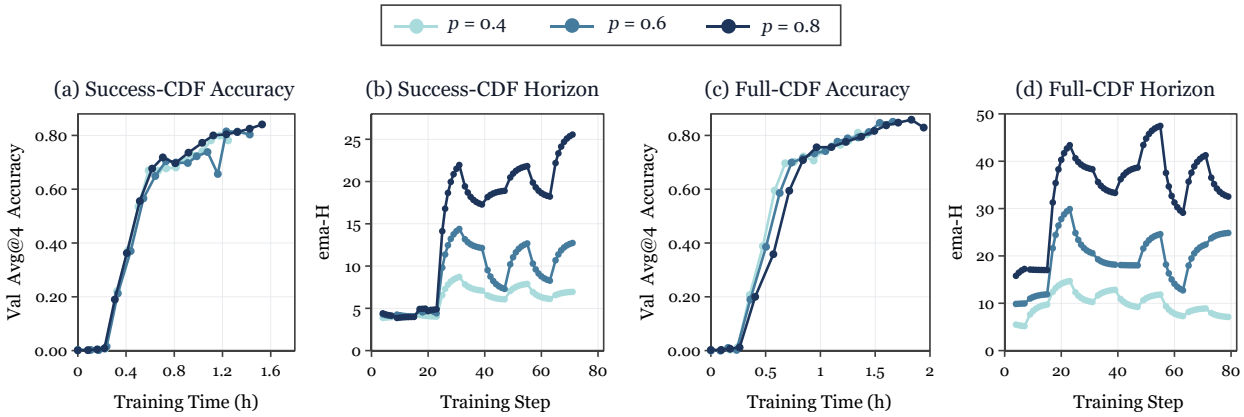

Figure 8 Coverage-floor sensitivity on ALFWorld-1.7B. The four panels show validation accuracy versus cumulative training time and logged EMA horizon for the success-conditioned CDF and the full-population CDF.

increases ema-H and wall time compared to the success-conditioned CDF. This matches intuition, as the full-population CDF incorporates many failed trajectories, including those that stall at the maximum rollout depth due to repeated or unproductive tool usage, thereby overestimating the required depth. Nonetheless, as shown in the figure, the full-population CDF remains effective, but requires appropriately lower values of p to match the efficiency and performance of the success-conditioned CDF.

8 Conclusion

We studied on-policy distillation for long-horizon language agent tasks. Our diagnosis shows that vanilla OPD suffers from two allocation mismatches: full-horizon rollouts can spend compute on low-yield tail turns, while trajectory-level KL reduction can concentrate loss on shallow tokens. TurnOPD addresses these mismatches with turn-level budgeting: it adapts rollout depth and progressively shifts KL normalization toward turn-balanced supervision. Across ALFWorld, WebShop, and Multi-Hop Search, TurnOPD advances the accuracy–time frontier over vanilla OPD. Overall, our results suggest a broader lesson for agent OPD. The unit of supervision in long-horizon agents should not be treated as a flat token position, but as a turn-conditioned decision inside an evolving interaction trace. TurnOPD opens a path toward turn-aware and budget-adaptive training of long-horizon language agents.

References

[1] 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, 2024. URL https://arxiv.org/abs/2306.13649.

[2] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In Workshop on Computer Games, pages 41–75. Springer, 2018.

[3] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In Advances in Neural Information Processing Systems (NeurIPS), 2023.

[4] Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. arXiv e-prints, pages arXiv–2507, 2025.

[5] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In International Conference on Learning Representations (ICLR), 2024.

[6] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In Proceedings of the 28th International Conference on Computational Linguistics, pages 6609–6625, 2020.

[7] Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Ziyu Ye, Bowei Xia, Tao Sun, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. Advances in Neural Information Processing Systems, 38:50859–50906, 2026.

[8] Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation, 2026. URL https://arxiv.org/abs/2601.07155.

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

[10] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. arXiv preprint arXiv:2503.09516, 2025.

[11] Woogyeol Jin, Taywon Min, Yongjin Yang, Dennis Wei, Yi Zhou, Swanand Ravindra Kadhe, Nathalie Baracaldo, and Kimin Lee. Entropy-aware on-policy distillation of language models, 2026. URL https://arxiv.org/abs/2603.07079.

[12] Jongwoo Ko, Sara Abdali, Young Jin Kim, Tianyi Chen, and Pashmina Cameron. Scaling reasoning efficiently via relaxed on-policy distillation, 2026. URL https://arxiv.org/abs/2603.11137.

[13] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics, 7:452–466, 2019.

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

[15] 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.

[16] Haipeng Luo, Huawen Feng, Qingfeng Sun, Can Xu, Kai Zheng, Yufei Wang, Tao Yang, Han Hu, and Yansong Tang. Agentmath: Empowering mathematical reasoning for large language models via tool-augmented agent. arXiv preprint arXiv:2512.20745, 2025.

[17] Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. arXiv preprint arXiv:2503.21460, 2025.

[18] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, pages 9802–9822, 2023.

[19] Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. arXiv preprint arXiv:2601.11868, 2026.

[20] Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie S. Kim, Patrick Chao, Samuel Miserendino, Gildas Chabot, David Li, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks. arXiv preprint arXiv:2510.04374, 2025.

[21] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In International Conference on Learning Representations (ICLR), 2025.

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

[23] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In International Conference on Learning Representations (ICLR), 2021.

[24] 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.

[25] Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. arXiv preprint arXiv:2504.12516, 2025.

[26] 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), pages 27914–27961, 2025.

[27] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2024.

[28] Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z. Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Melroy Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks. arXiv preprint arXiv:2412.14161, 2024.

[29] 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.

[30] Wenkai Yang, Weijie Liu, Ruobing Xie, Kai Yang, Saiyong Yang, and Yankai Lin. Learning beyond teacher: Generalized on-policy distillation with reward extrapolation, 2026. URL https://arxiv.org/abs/2602.12125.

[31] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2369–2380, 2018.

[32] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

[33] Yaocheng Zhang, Jiajun Chai, Songjun Tu, Yuqian Fu, Xiaohan Wang, Wei Lin, Guojun Yin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Are full rollouts necessary for on-policy distillation? arXiv preprint arXiv:2605.31490, 2026.

[34] Yuxuan Zhang, Yubo Wang, Yipeng Zhu, Penghui Du, Junwen Miao, Xuan Lu, Wendong Xu, Yunzhuo Hao, Songcheng Cai, Xiaochen Wang, Huaisong Zhang, Xian Wu, Yi Lu, Minyi Lei, Kai Zou, Huifeng Yin, Ping Nie, Liang Chen, Dongfu Jiang, Wenhu Chen, and Kelsey R. Allen. ClawBench: Can AI agents complete everyday online tasks? arXiv preprint arXiv:2604.08523, 2026.

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

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

[37] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. arXiv preprint arXiv:2307.13854, 2023.

[38] Yuhang Zhou, Kai Zheng, Qiguang Chen, Mengkang Hu, Qingfeng Sun, Can Xu, and Jingjing Chen. Offseeker: Online reinforcement learning is not all you need for deep research agents. arXiv preprint arXiv:2601.18467, 2026.

A Environments, Tool Configurations, And Task Examples

In this work, we use the following three environments and task types: ALFWorld [23], Multi-Hop Search (consisting of several sub-benchmarks including PopQA [18], NQ [13], 2WikiMultiHopQA [6], HotpotQA [31]), and WebShop [32].

For the student and teacher models, we explicitly activate the "thinking" mode, so each model turn consists of a segment of chain-of-thought and a tool call. Client-side validation rejects malformed or inadmissible actions before they reach the environment so a bad action does not consume a real step. All experiments are conducted on 32 NVIDIA H20 GPUs, each with 80 GB of memory.

A.1 Environment Details

A.1.1 ALFWorld (Embodied Household Planning)

Tool / action space. A single tool `alfworld_step(action: str)`; the action is a natural-language household command from the templates: `go to {loc}`, `take {obj} from {loc}`, `put {obj} in/on {loc}`, `open/close {loc}`, `use {loc}`, `clean/heat/cool {obj} with {loc}`, `examine {x}`, `inventory`, `look`. The tool returns the new TextWorld observation plus a refreshed list of admissible commands; actions not in the admissible set are rejected client-side.

Environment. A FastAPI server wraps TextWorld [2] with a spawned worker pool. The six task types are `pick_and_place`, `pick_clean_then_place`, `pick_heat_then_place`, `pick_cool_then_place`, `look_at_obj_in_light`, and `pick_two_obj_and_place`; evaluation uses the `valid_unseen` (out-of-distribution) split. The reward is **binary** (1 on goal completion, else 0).

Task example.

```
[system] You are an AI agent operating in a household environment. You can
interact with objects and navigate between locations using the alfworld_step tool.
[user]   Your task is to: put a clean mug in coffeemachine.
[assistant -> alfworld_step] {"action": "go to countertop 1"}
[tool]   On the countertop 1 you see a mug 1, a knife 1, ...
[assistant -> alfworld_step] {"action": "take mug 1 from countertop 1"}
[tool]   You pick up the mug 1 from the countertop 1.
[assistant -> alfworld_step] {"action": "clean mug 1 with sinkbasin 1"} ...
```

A.1.2 Multi-Hop Search

Tool / action space. We instantiate Multi-Hop Search with a Search-R1-style interface [10]. A single retrieval tool `search_r1/search(query: str)` POSTs the query to a dense-retrieval server and returns the top-k passages formatted as `Doc i(Title: ...) ...`. The model reasons in `<think>...</think>`, issues retrieval via the search tool (or the equivalent `<search>...</search>` text channel), receives results in `<information>...</information>`, and emits the final answer in `<answer>...</answer>`.

Environment. The retrieval endpoint serves a fixed Wikipedia corpus; rollouts are capped at `max_turns`=50, allowing many search–read hops. The reward is **exact match (EM)** of the extracted `<answer>` against the gold answer.

Sub-benchmarks. The held-out Multi-Hop Search test set is a mixture of four QA sources: `PopQA` [18], `Natural Questions` (NQ) [13], `2WikiMultiHopQA` [6], and `HotpotQA` [31]. PopQA and NQ provide open-domain retrieval questions, while 2WikiMultiHopQA and HotpotQA require chained retrievals where a later query depends on an earlier passage, making the evaluation mixture cover both direct retrieval and long-horizon dependency cases.

Task example.

```
[user] Answer the question. Reason in <think>...</think>; if you lack knowledge,
search via <search> query </search> and read the returned <information>...
</information>; give the final answer in <answer>...</answer>.
Question: Who is the director of the film that won Best Picture in the year
Titanic was released?
[assistant] <think>I need the Best-Picture winner of 1997.</think>
            <search> 1997 Academy Award Best Picture winner </search>
[tool]      <information>Doc 1(Title: Titanic (1997 film)) ... won Best Picture ...
            </information>
[assistant] <think>Director of Titanic is James Cameron.</think>
            <answer> James Cameron </answer>
```

A.1.3 WebShop (Web Navigation / Online Shopping)

Tool / action space. The agent operates a single tool `webshop_step(action: str)` with two formats: `search[<keywords>]` for query input (when a search bar is available), and `click[<button>]` to interact with one of the currently visible buttons (e.g., product ID, attribute options like "x-large", or navigation buttons such as `Buy Now`, `Next >`, etc.). The tool outputs the updated page view and available actions. Invalid clicks are blocked on the client side.

Environment. A FastAPI server wraps AgentGym's `agentenv_webshop` [26]. An episode proceeds search → inspect results → open a product → select attributes → `click[Buy Now]`. The reward is a **dense terminal score in** $[0, 1]$ reflecting partial credit for matching the requested product type, attributes, options, and price; only the terminating `Buy Now` carries reward, and a machine-parseable `[WEBSHOP_FINAL_SCORE=...]` marker is appended for the reward path.

Task example.

```
[user] Instruction: I am looking for a milk chocolate of 1 pound size in a single
pack for valentine day, and price lower than 40 dollars.
Available actions: search[<your keywords>]
[assistant -> webshop_step] {"action": "search[valentine milk chocolate 1 pound
single pack]"}
[tool]  [Results] B07... Valentine Milk Chocolate 1 lb ... $19.99  (clickables:
        "B07...", "Next >", "Back to Search")
[assistant -> webshop_step] {"action": "click[B07...]"}
[tool]  ... (clickables: "1 pound", "Buy Now", "Description", ...)
[assistant -> webshop_step] {"action": "click[Buy Now]"}
[tool]  Episode finished with score=1.000.  [WEBSHOP_FINAL_SCORE=1.000000]
```

A.2 Task-Specialized Teacher Training

We train task-specialized teachers using GRPO [22] on the three environments. Figure 9 reports the validation dynamics of the task-specialized teachers used in the main experiments. All curves report avg@4 accuracy. ALFWorld and WebShop use Qwen3-8B-GRPO teachers, while Multi-Hop Search uses a Qwen3.5-9B-GRPO teacher.

B Proof Of The Contamination-Compression Bound

We provide the full proof of Proposition 1. Fix a context c and write $\lambda = \lambda(c)$ for readability. The only ingredient is the joint convexity of KL divergence. For any distributions P_1, Q_1, P_2, Q_2 over the same vocabulary and any $\alpha \in [0, 1]$,

$$D_{\mathrm{KL}}(\alpha P_1 + (1-\alpha)P_2 \,\|\, \alpha Q_1 + (1-\alpha)Q_2) \le \alpha D_{\mathrm{KL}}(P_1 \,\|\, Q_1) + (1-\alpha) D_{\mathrm{KL}}(P_2 \,\|\, Q_2). \tag{17}$$

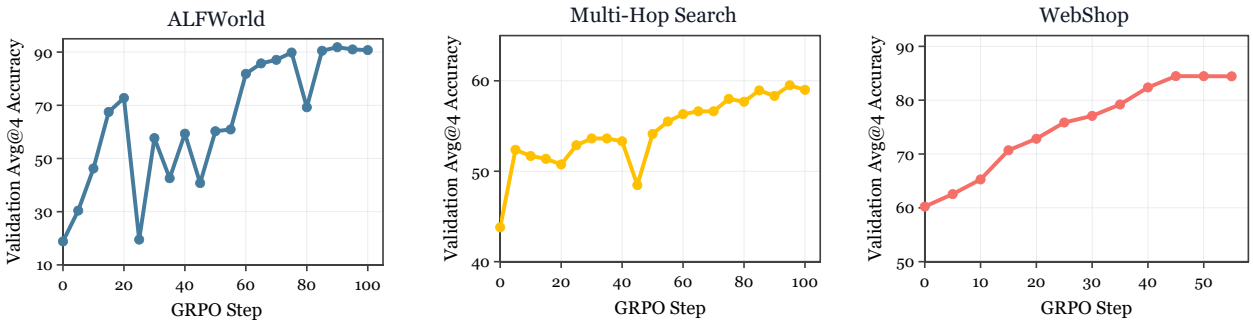

Figure 9 GRPO validation curves for the task-specialized teacher models.

Joint convexity from log-sum. For each token x, set $a_1 = \alpha P_1(x)$, $a_2 = (1-\alpha)P_2(x)$, $b_1 = \alpha Q_1(x)$, and $b_2 = (1-\alpha)Q_2(x)$. The log-sum inequality gives

$$\begin{aligned}
&(a_1 + a_2) \log \frac{a_1 + a_2}{b_1 + b_2} \\
&\quad \le a_1 \log \frac{a_1}{b_1} + a_2 \log \frac{a_2}{b_2} \\
&\quad = \alpha P_1(x) \log \frac{P_1(x)}{Q_1(x)} + (1-\alpha) P_2(x) \log \frac{P_2(x)}{Q_2(x)},
\end{aligned} \tag{18}$$

where the factors α and $1-\alpha$ cancel inside the logarithms. Summing over all tokens x gives Equation 17.

Applying the inequality. Set $\alpha = \lambda$, $P_1 = Q_1 = p_F$, $P_2 = p_S^{\text{free}}$, and $Q_2 = p_T^{\text{free}}$. Then

$$\begin{aligned}
&D_{\mathrm{KL}}\big(\lambda p_F + (1-\lambda) p_S^{\text{free}} \,\|\, \lambda p_F + (1-\lambda) p_T^{\text{free}}\big) \\
&\quad \le \lambda D_{\mathrm{KL}}(p_F \,\|\, p_F) + (1-\lambda) D_{\mathrm{KL}}(p_S^{\text{free}} \,\|\, p_T^{\text{free}}).
\end{aligned} \tag{19}$$

Because $D_{\mathrm{KL}}(p_F \,\|\, p_F) = \sum_x p_F(x) \log 1 = 0$,

$$D_{\mathrm{KL}}\big(\lambda p_F + (1-\lambda) p_S^{\text{free}} \,\|\, \lambda p_F + (1-\lambda) p_T^{\text{free}}\big) \le (1-\lambda) D_{\mathrm{KL}}(p_S^{\text{free}} \,\|\, p_T^{\text{free}}), \tag{20}$$

which proves Equation 6.

C Existence of a Latent Rollout-Depth Optimum

This appendix justifies the notation $H^\star$ used in Section 5. The useful rollout horizon is not defined by a single force. It is squeezed between two opposing requirements. If the horizon is too short, the rollout fails to cover the decision turns on which successful trajectories finish. If the horizon is too long, the remaining turns add collection cost after the measurable teacher-correction signal has already decayed. This gives a two-sided efficiency–coverage sandwich.

Efficiency side. Fix a student checkpoint, a task distribution, and a finite maximum rollout cap $H_{\max}$. For an idealized depth H, let

$$\rho(H) = \frac{\sum_{t \le H} v_t w_t}{\sum_{t \le H} c_t w_t + \epsilon}, \tag{21}$$

where $v_t \ge 0$ is the local teacher-correction value at turn t, $w_t = n_t / n_0$ is the survivor probability, and $c_t > 0$ is the per-turn collection cost. Let $H_{\text{eff}}^\star$ be the smallest maximizer of $\rho(H)$ on $1 \le H \le H_{\max}$. Equivalently,

adding turn $H+1$ improves the ratio only when its marginal rate is at least the current average:

$$\rho(H+1) \geq \rho(H) \quad \Longleftrightarrow \quad \frac{v_{H+1}}{c_{H+1}} \geq \rho(H), \tag{22}$$

obtained by cross-multiplying $(A+a)/(B+b) \geq A/B$ for positive B, b. Suppose there is a signal-exhaustion frontier τ_c after which the marginal teacher-correction value is zero, or more generally the marginal rate stays below the running average. Then every turn beyond τ_c can only decrease the efficiency ratio, so

$$H^\star_{\text{eff}} \leq \tau_c - 1. \tag{23}$$

Coverage side. Let L_{succ} be the completion length of a successful rollout, and let

$$F_{\text{succ}}(H) = \Pr(L_{\text{succ}} \leq H), \qquad H_{\text{cov}} = \widehat{Q}_p(L_{\text{succ}}) = \min\{H : F_{\text{succ}}(H) \geq p\}, \tag{24}$$

for a fixed $p \in (0, 1]$. This is a coverage floor: stopping before H_{cov} misses the completion decisions of more than a $(1-p)$ fraction of successful trajectories. Define the full-coverage completion depth as

$$\tau_{\text{done}} = \min\{H : F_{\text{succ}}(H) = 1\}. \tag{25}$$

Equivalently, for a finite empirical support this is the largest observed successful completion length, $\max \operatorname{supp}(L_{\text{succ}})$. This definition is deliberately the first point that has already covered all successful completions, not the last point before full coverage. Therefore

$$H_{\text{cov}} \leq \tau_{\text{done}}. \tag{26}$$

Sandwich. The latent target should respect both sides:

$$H^\star = \max(H^\star_{\text{eff}},\ H_{\text{cov}})\,. \tag{27}$$

Therefore

$$H^\star_{\text{eff}} \leq H^\star \leq \max(\tau_c - 1, \tau_{\text{done}})\,. \tag{28}$$

The lower side prevents an overly shallow efficiency optimum from truncating turns that successful trajectories need; the upper side prevents the horizon from extending past both the signal-exhaustion frontier and the deepest successful completion.

D Complete TurnOPD Hyperparameters

Table 7 lists the full configuration of the main TurnOPD method, mapping each implementation key to the symbol used in the formal model (Sections 5.1–5.2). The `ema_alpha` parameter is the EMA weight on the depth proxy H_{ctrl}, distinct from the loss-blend coefficient α.

Algorithm 1: TurnOPD training algorithm.

Input: π_θ, π_T, task distribution $\mathcal{D}$, rollout bounds $H_{\min}, H_{\max}$, training horizon K, and controller hyperparameters $K_{\text{warm}}, r_{\text{probe}}, \alpha_{\text{ema}}, p, n_{\min}^{\text{cov}}, (s, e)$.

Output: updated student policy π_θ.

Initialize $\bar{H}_0 \leftarrow H_{\max}$ and $H_{\text{cov}} \leftarrow 0$

for $k = 1$ **to** K **do**

- $\hat{H}_k \leftarrow \text{clip}(\text{round}(\bar{H}_{k-1}) + 1, H_{\min}, H_{\max})$
- $\text{probe}_k \leftarrow (k \le K_{\text{warm}}) \ \vee \ (k \bmod r_{\text{probe}} = 0)$
- **if** probe_k **then**
  - $H_{\text{roll}} \leftarrow H_{\max}$
- **else**
  - $H_{\text{roll}} \leftarrow \hat{H}_k$
- Collect on-policy trajectories $\mathcal{B}_k$ from π_θ on $\mathcal{D}$ up to H_{roll} turns
- Query π_T on the supervised tokens in $\mathcal{B}_k$ and compute top-K reverse-KL token losses ℓ_i
- **if** probe_k **then**
  - Compute per-turn raw-KL means K_t and survivor counts n_t from the uncensored probe batch
  - $m_t \leftarrow [K_t]_+ \, n_t / n_0, \quad q_t \leftarrow m_t / (\sum_j m_j + \epsilon)$
  - $H_{\text{eff}} \leftarrow \text{round}(\sum_t t \, q_t)$
  - **if** $|\mathcal{B}_k^{\text{succ}}| \ge n_{\min}^{\text{cov}}$ **then**
    - Estimate $F_{\text{succ}}(H)$ from successful probe trajectories and set $H_{\text{cov}} \leftarrow \min\{H : F_{\text{succ}}(H) \ge p\}$
  - $H_{\text{ctrl}} \leftarrow \max(H_{\text{eff}}, H_{\text{cov}})$
  - $\bar{H}_k \leftarrow (1 - \alpha_{\text{ema}}) \bar{H}_{k-1} + \alpha_{\text{ema}} H_{\text{ctrl}}$
- **if** $\neg \, \text{probe}_k$ **then**
  - $\bar{H}_k \leftarrow \bar{H}_{k-1}$
- $\rho_k \leftarrow k/K, \quad \alpha_k \leftarrow \text{clip}((\rho_k - s)/(e - s), 0, 1)$
- Construct trajectory-normalized weights w_{traj} and turn-normalized weights w_{turn} on the observed tokens in $\mathcal{B}_k$
- $w_{\text{blend}} \leftarrow (1 - \alpha_k) w_{\text{traj}} + \alpha_k w_{\text{turn}}$
- $\mathcal{L}_k \leftarrow \sum_{i \in \mathcal{B}_k} w_{\text{blend},i} \ell_i$
- Update θ by descending $\nabla_\theta \mathcal{L}_k$

Table 7 Full TurnOPD configuration (ALFWorld-1.7B reference run). "Symbol" is the variable in the formal model; "Key" is the implementation config name.

Group	Symbol	Key = value	Role
External signal	K_t	`kl_per_turn/*`	raw per-turn reverse-KL signal used in $m_t = [K_t]_+ n_t / n_0$
	n_t	`num_traj/*`	all-trajectory survivor count used in the efficiency-side mass
	K_{top}	`distill_topk`=50	teacher top-K_{top} support size for the reverse-KL loss
External depth	p	`coverage_quantile`=0.80	success-length quantile for $H_{\text{cov}} = \hat{Q}_p(L_{\text{succ}})$
	—	`use_success`=True	read H_{cov} from success subset (False → all-trajectory CDF)
	—	`min_cov_traj`=8	min successful trajectories to refresh H_{cov}
	$H_{\min}/H_{\max}$	`min/max`=2/50	clamp on applied depth $\hat{H}$
External online	α_{ema}	`ema_alpha`=0.30	EMA weight on H_{ctrl} (not the blend α)
	—	`probe_interval`=8	steps between full-depth probes
	—	`warmup_steps`=3	probe-only warmup before truncation
Internal blend	s, e	`blend start/end`=0/1	linear progress window for $\alpha = \text{clip}((\text{prog} - s)/(e - s), 0, 1)$
	—	`min_floor`=8	turn-level $n_{\min} = \max(\text{floor}, \lceil \text{frac} \cdot n_{\text{traj}} \rceil)$
	—	`min_frac`=0.15	fraction term of $n_{\min}$
	—	`turn_norm_blend`=True	enable linear trajectory→turn blend
OPD / optimization	—	`kl_coef`=0, `kl_loss_coef`=0	no reward/penalty KL; loss is pure top-K RKL
	—	`loss_mode`=topk_reverse_kl	differentiable objective
	—	`seq-mean-token-mean`	base (trajectory-level) aggregation
	—	`clip_ratio`=0.9/9	importance-ratio clip (low/high)
	—	lr=10^{-6}, batch=64, steps=100	AdamW
Rollout / eval	—	`max_turns`=80 (ALFWorld) / 50 (Multi-Hop Search)	rollout turn cap before adaptive-depth truncation
	—	temperature=1.0, n=1	training rollout sampling
	—	`val_n`=4, val temp.=0.85	validation: mean@4

E OPD Data Scale And Train/Test Construction

Each task feeds the same on-policy distillation pipeline a parquet of prompts; the agent then rolls out against the live environment, so a "sample" is a task **instance** (a goal / question) rather than a fixed teacher trajectory—trajectories are generated online during training. Table 8 summarizes the reference scale and split construction; the three tasks differ fundamentally in how their train/test partitions are obtained, which we detail below.

ALFWorld (procedurally generated, OOD evaluation). Task instances are synthesized by sampling one of the six task types uniformly (Appendix A.1.1) and instantiating it with an object–receptacle pair drawn from fixed vocabularies under a seeded RNG, so train and validation prompts are reproducible and non-overlapping by seed. Crucially the two splits are routed to **different** environment partitions: training instances use the `train` games, whereas evaluation instances use the `eval_out_of_distribution` (valid-unseen) games, so reported accuracy measures generalization to unseen household layouts rather than memorization. The reference configuration generates ≈ 5000 training and ≈ 300 evaluation instances with a per-trajectory step cap of 80; the underlying ALFWorld task pool has roughly 3.5k unique games, which upper-bounds the distinct training layouts.

Multi-Hop Search (external QA mixture, separate test set). Training uses an externally prepared corpus of short-chain multi-hop / open-domain questions merged from Search-R1-format sources, at ≈ 30k question–answer instances; evaluation uses a separate held-out test set (≈ 400 questions) drawn from PopQA, Natural

Table 8 OPD dataset scale and split construction. "Train/Eval" counts are task instances (online rollouts, not stored trajectories); the per-trajectory step cap is the data-side `max_iterations` before any adaptive-depth truncation.

Task	Train	Eval	Eval–train relation
ALFWorld	≈ 5000	≈ 300	held-out OOD split (`eval_out_of_distribution`)
Multi-Hop Search	≈ 30000	≈ 400	separate four-source QA test mixture
WebShop	≈ 3000	≈ 200	disjoint goal-id ranges

Questions (NQ), 2WikiMultiHopQA, and HotpotQA (Appendix A.1.2), with the originating source preserved in the `data_source` field so that exact-match reward and per-source breakdowns route correctly. Prompts are prepared with thinking enabled (a `<think>` reasoning channel before each search), and the step cap is 50 to accommodate several search–read hops. Because the questions are real (not synthesized), train and test are disjoint by construction at the dataset level.

WebShop (goal-id partition, disjoint ranges). A WebShop instance is selected purely by a goal id (forwarded to the environment as the session/goal index); the instruction text itself is part of the environment's initial observation and is not duplicated into the prompt. Train and evaluation use **strictly disjoint** goal-id ranges—evaluation takes ids $[0, N_{\text{eval}})$ and training takes $[N_{\text{eval}}, N_{\text{eval}} + N_{\text{train}})$—so no goal appears in both splits. The reference configuration uses ≈ 3000 training and ≈ 200 evaluation goals with a step cap of 30 (WebShop episodes are short, typically 4–12 actions). Goal ids must stay within the goal count of the launched product corpus.