TURNOPD: MAKING ON-POLICY DISTILLATION TURN-AWARE FOR EFFICIENT LONG-HORIZON AGENT TRAINING

Anonymous authors
Paper under double-blind review

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 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 depth, 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.

1 INTRODUCTION

Language models are increasingly deployed as agents for planning, tool use, and environment interaction (Gao et al., 2025; Hu et al., 2026; Luo et al., 2025b;a; Zhou et al., 2026). On-policy distillation (OPD) offers a promising training framework for such agentic models (Lu & Lab, 2025; Li et al., 2026), 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 (Zhong et al., 2026; Wang et al., 2026). 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 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 (Shridhar et al., 2021), 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

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.

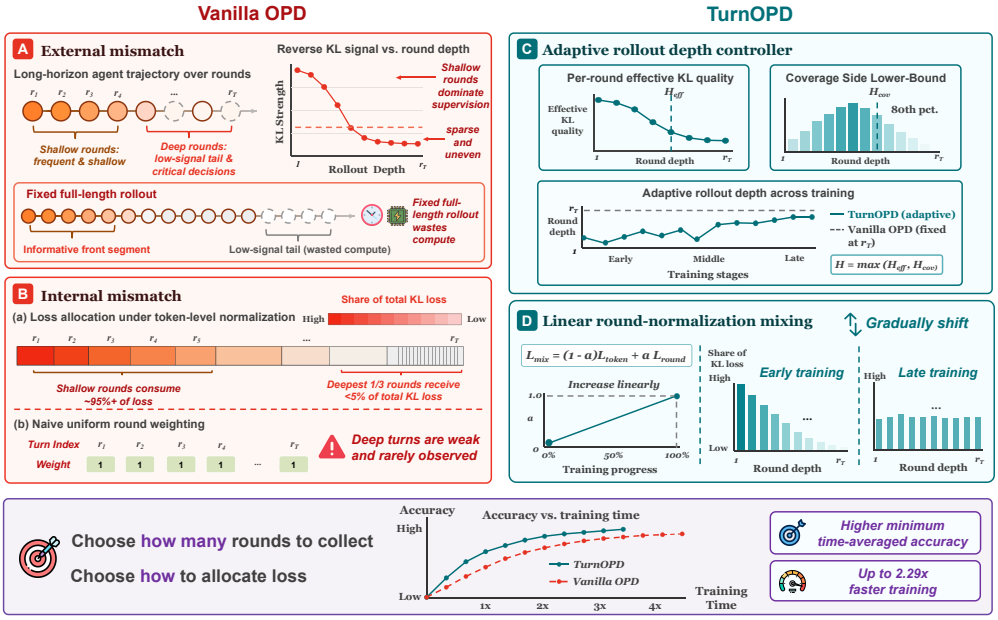

Figure 1: A turn-aware perspective on agent OPD. Standard OPD fixes the rollout depth and averages KL across the full trajectory, often wasting resources on low-value tail turns and over-focusing loss on shallow tokens. TurnOPD budgets both rollout depth and KL aggregation based on turn-level statistics,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

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) (Yao et al., 2022), and Multi-Hop Search (including PopQA (Mallen et al., 2023), NQ (Kwiatkowski et al., 2019), 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018)). 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× faster training.

2 Related Work

On-policy distillation. MiniLLM and GKD introduced OPD recipes based on student-generated samples and alternative divergence objectives (Gu et al., 2024; Agarwal et al., 2024). Later work connects OPD to KL-constrained policy optimization and studies stabilized targets, teacher entropy, or relaxed imitation (Lu & Lab, 2025; Yang et al., 2026; Jang et al., 2026; Jin et al., 2026; Ko et al., 2026; Zhao et al., 2026). Dense supervision can nevertheless become unreliable on long trajectories (Li et al., 2026; Zhang et al., 2026). TCOD curricula trajectory length (Wang et al., 2026), while SOD reweights step-level distillation (Zhong et al., 2026). We instead analyze and control rollout and loss allocation across agent turns.

Long-horizon agent tasks. ALFWorld and WebShop study embodied planning and grounded web navigation (Shridhar et al., 2021; Yao et al., 2022); later benchmarks extend to real websites, GUI control, repository editing, and persistent research (Deng et al., 2023; Zhou et al., 2023; Xie et al., 2024; Rawles et al., 2025; Jimenez et al., 2024; Merrill et al., 2026; Wei et al., 2025). Across these settings, the common structure is a stateful interaction trace with external observations, changing action validity, and a changing survivor population rather than a flat response.

3 Preliminaries: On-Policy Distillation

Multi-turn agent interaction. At turn t, the student observes o_t, conditions on the history

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

and generates $r_t \sim \pi_\theta(\cdot \mid h_t)$, which may contain reasoning and a tool call. The environment maps its executable part to o_{t+1}, forming an interaction trace until task completion or a maximum horizon. Early actions therefore change the future contexts on which the teacher is queried.

Multi-turn OPD. We use on-policy distillation (OPD) (Lu & Lab, 2025): 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 (Wang et al., 2026; Zhong et al., 2026). 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.

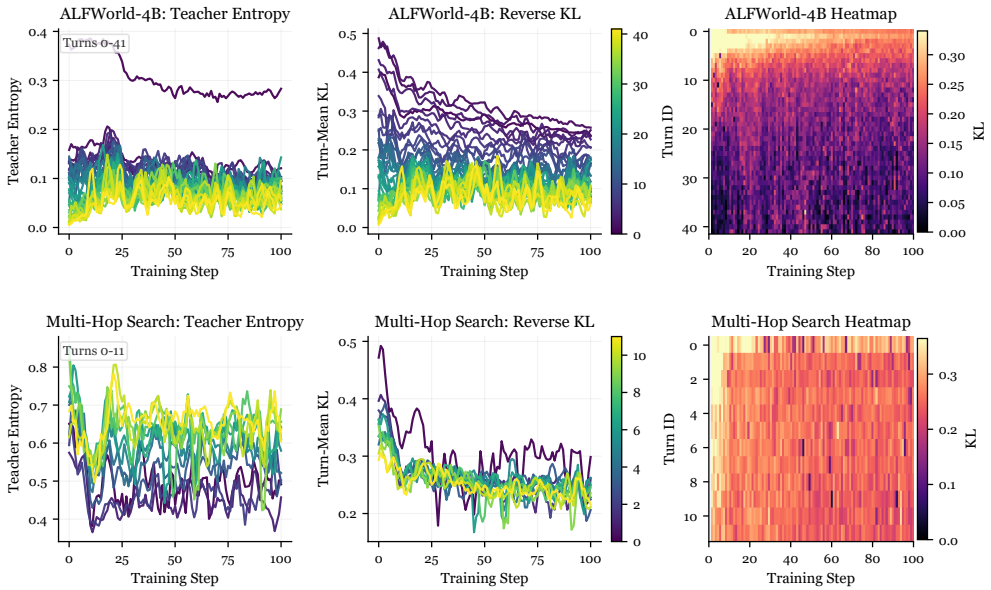

Figure 2: Turn-resolved teacher uncertainty and reverse-KL under vanilla OPD. The ALFWorld row uses the Qwen3-4B student baseline, and the Multi-Hop Search row uses the Qwen3.5-2B student baseline. For each task, the left panel shows smoothed per-turn teacher entropy, the middle panel shows smoothed 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.

Setup. We use ALFWorld (Shridhar et al., 2021) and Multi-Hop Search as example tasks. Both diagnostics use vanilla OPD over 100 optimizer steps: ALFWorld with a Qwen3-4B (Yang et al., 2025) student distilled from a Qwen3-8B-GRPO teacher (ALFWorld-4B), and Multi-Hop Search with a Qwen3.5-2B (Yang et al., 2025) 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. More importantly, declining per-turn KL is not by itself evidence of genuine alignment: the growing on-policy context may increasingly constrain both models to the same local continuation. This ambiguity motivates the contamination-compression mechanism in the next subsection.

4.2 A Contamination-Compression Mechanism

Raw reverse-KL decreases along the reliable turn prefix, but 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.

For a fixed context c, suppose part of the next-token distribution is nearly forced by the context itself. Let $p_F(\cdot \mid c)$ denote this shared forced component, and let the remaining mass contain free

components on which the student and teacher may genuinely disagree:

$$\pi_S = \lambda p_F + (1-\lambda) p_S^{\text{free}}, \qquad \pi_T = \lambda p_F + (1-\lambda) p_T^{\text{free}}. \tag{2}$$

Proposition 1 (Contamination compression)**.** *Under the decomposition in Equation 2,*

$$D_{\mathrm{KL}}(\pi_S \,\|\, \pi_T) \le (1-\lambda)\, D_{\mathrm{KL}}\big(p_S^{\text{free}} \,\|\, p_T^{\text{free}}\big)\,. \tag{3}$$

Thus, a large context-forced mass can compress the measured KL even when the free-component disagreement remains nonzero. The outcome-separated diagnostic, the full model implications, and the proof are provided in Appendix B and Appendix C.

4.3 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, let turn t contain n_t supervised tokens with token-level reverse-KL losses $\ell_{t,i}$, and let $N = \sum_{t=1}^{T} n_t$ be the total number of supervised tokens. A standard trajectory-level reduction is

$$L_{\text{traj}} = \frac{1}{N} \sum_{t=1}^{T} \sum_{i=1}^{n_t} \ell_{t,i}. \tag{4}$$

This reduction gives every supervised token the same base weight, so the token share of turn t is $b_t^{\text{traj}} = n_t/N$. The realized share of the batch KL loss mass is instead

$$s_t^{\text{traj}} = \frac{\sum_{i=1}^{n_t} \ell_{t,i}}{\sum_{u=1}^{T} \sum_{j=1}^{n_u} \ell_{u,j}}. \tag{5}$$

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 3.

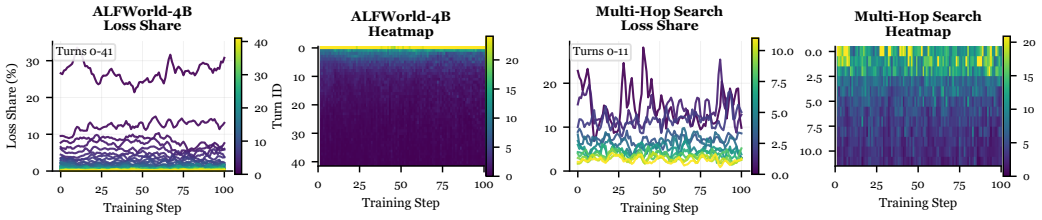

Figure 3: Turn-level KL loss-share under the vanilla trajectory-level KL objective. Line panels show one smoothed training-step curve per reliably observed turn; heatmaps show the raw step-by-turn loss-share values. The metric is the fraction of the current batch's raw KL loss mass contributed by turn t.

Figure 3 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

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 3. 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%

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 3.

(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.

4.4 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

Building on the diagnosis, we identify two key challenges in applying OPD to agents. We propose **TurnOPD**, a turn-level budgeting method for efficient OPD in long-horizon tasks, comprising two budget controllers: (1) adaptive rollout-depth control and (2) progressive KL loss allocation from trajectory-level to turn-wise refinement.

5.1 External Mismatch: Adaptive Rollout-Depth Budgeting

A key issue is the mismatch between rollout depth and effective supervision: the ideal trajectory length for data collection should reflect both task and student progress in order to maximize supervision efficiency. Let $H^\star$ be the (unobservable) optimal rollout horizon yielding maximal marginal validation improvement per supervision cost. Its existence is guaranteed by an efficiency–coverage tradeoff: too short under-explores, too long wastes computation (see Appendix D).

Since $H^\star$ is not directly measurable during training, TurnOPD estimates it online using two complementary signals: an efficiency-side proxy and a coverage-based lower bound. The rollout-depth budget is defined as

$$H_{\text{ctrl}} = \max\big(H_{\text{eff}},\, H_{\text{cov}}\big), \tag{6}$$

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

Efficiency-side Mass. H_{eff} is derived by treating the reverse-KL signal over turns as a survivor-weighted distribution of distillation mass. Although raw KL may not directly measure behavioral error, it indicates supervision value and cost. For turn t, let K_t be mean per-token reverse-KL and n_t/n_0 the on-policy survivor probability:

$$m_t = [K_t]_+ \cdot \frac{n_t}{n_0}, \qquad q_t = \frac{m_t}{\sum_j m_j + \epsilon}, \tag{7}$$

where $[K_t]_+$ is the positive part of K_t[1]. Here, m_t estimates the value of supervision at turn t, and q_t is the normalization. The centroid and its rounding are

$$\bar{H}_{\text{eff}} = \sum_t t\, q_t, \qquad H_{\text{eff}} = \text{round}(\bar{H}_{\text{eff}}). \tag{8}$$

This statistic adapts to changes in the distribution of supervision signal with training.

Coverage-side Lower Bound. To avoid excessive truncation, H_{cov} sets a minimum rollout depth based on successful completions:

$$H_{\text{cov}} = \hat{Q}_p(L_{\text{succ}}) = \min\{H : F_{\text{succ}}(H) \geq p\}, \qquad F_{\text{succ}}(H) = 1 - \frac{n^{\text{succ}}_{H+1}}{n^{\text{succ}}_1}, \quad p = 0.80, \tag{9}$$

so that at least 80% of successful trajectories are covered by periodic probe rollouts.

Dynamic Update. The controller updates H_{eff} on every batch and H_{cov} only on periodic full-length probes; routine steps use only the capped depth $\hat{H}_k$. Since capped batches cannot reveal turns beyond the cap, H_{eff} computed on them cannot raise the depth, so growth relies on the periodic full-depth probes.

After every step, TurnOPD computes $H_{\text{ctrl},k} = \max(H_{\text{eff},k}, H_{\text{cov},k})$ and updates rollout depth with exponential smoothing:

$$\bar{H}_k = (1 - \alpha_{\text{ema}})\bar{H}_{k-1} + \alpha_{\text{ema}} H_{\text{ctrl},k}, \qquad \hat{H}_{k+1} = \text{clip}\big(\text{round}(\bar{H}_k) + 1,\, H_{\min},\, H_{\max}\big). \tag{10}$$

The $+1$ compensates for zero-based indexing.

5.2 Internal Mismatch: Progressive Turn-Normalized Loss Budgeting

A second challenge is the internal budget mismatch: trajectory-level loss normalization allocates most KL loss to shallow turns, leaving deeper turns undersupervised—particularly in later training. While early emphasis on shallow errors is reasonable, this later impedes the correction of deep decision errors.

To address this, TurnOPD introduces a linear blend of trajectory-level and turn-level normalization, gradually shifting loss allocation during training. Let n_t be the number of tokens at turn t (with T turns). The standard trajectory weight 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 blend is

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

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

Initially ($\alpha = 0$), loss follows token distribution for stability. As α increases, supervision is distributed more evenly across turns, addressing imbalance and improving deep turn learning.

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

Table 2: Main accuracy–efficiency results. Avg@4 is reported under the Least-Time and Same-Step protocols, with the smaller $\pm$ term denoting standard deviation over four evaluation checkpoints. Wall time covers 100 optimizer steps; speedup is relative to vanilla OPD. **Bold** marks the best student-training method in each group.

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	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	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	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	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**×

5.3 Unification and Algorithm

The two controllers regulate complementary resources: rollout depth sets the interaction budget, and loss normalization allocates supervision across collected tokens. Full algorithm details and hyperparameters are in Appendix E.

6 Experiments

We organize the evaluation around three questions: (1) does TurnOPD improve the accuracy–time frontier across tasks and model scales; (2) do its two budget controllers play distinct and complementary roles; and (3) does progressive turn-level normalization improve loss allocation over either fixed extreme?

6.1 Accuracy–Efficiency Frontier

Experiment settings. 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 reverse-KL training stack. We compare vanilla OPD, TCOD-F2B (Wang et al., 2026), and TurnOPD under two regimes: *Least-Time* compares accuracy at the smallest wall-clock cutoff at which any method reaches 100 steps, whereas *Same-Step* evaluates every method after 100 steps. We report Avg@4 and its standard deviation over the last four evaluations.

Results. Table 2 summarizes the main results across three long-horizon tasks and four student–teacher settings. TurnOPD attains the highest Least-Time Avg@4 in every setting. On ALFWorld-1.7B, it reaches 85.60, versus 73.52 for vanilla OPD and 80.06 for TCOD-F2B, while raising Same-Step accuracy to 86.29. The only exception is Multi-Hop Search under Same-Step evaluation, where vanilla OPD retains a small advantage (47.82 versus 47.24); Least-Time still favors TurnOPD. For training efficiency, TurnOPD reduces ALFWorld-1.7B training from 4.42 h to 1.93 h ($2.29\times$) while improving accuracy. It likewise reduces Multi-Hop Search from 4.45 h to 2.94 h ($1.51\times$) and WebShop from 1.57 h to 1.24 h ($1.26\times$). Overall, TurnOPD significantly improves accuracy and training efficiency on tested tasks, advancing the accuracy-time frontier beyond vanilla OPD. Additional experimental results and analyses are presented in Appendix G.

6.2 Complementary Roles of the Two Budget Controllers

The aggregate results establish the accuracy–efficiency gain, but do not show which intervention produces it. We therefore isolate the two controllers on ALFWorld-1.7B under the same protocol. *Adaptive Depth* changes only the rollout horizon while retaining trajectory-level KL. *Linear blend norm* retains full-horizon collection while changing only loss allocation. TurnOPD combines both interventions.

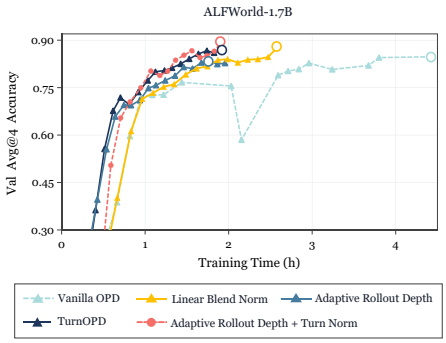

Figure 4: Validation curves for the ALFWorld-1.7B component ablations.

Table 3: Controller decomposition on ALFWorld-1.7B over 100 optimizer steps. Acc is Same-Step Avg@4 over the last four evaluation checkpoints; wall time sums per-step training time.

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**

Table 4: Impact of KL normalization on ALFWorld-1.7B. Same-Step Avg@4 is the mean over the last four evaluation checkpoints. Deep budget and the blend coefficient α are reported for early/mid/late training.

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

Results. Figure 4 shows the validation trajectories, while Table 3 summarizes Same-Step accuracy and cumulative wall time. Adaptive depth alone cuts wall time from 4.42 h to 1.96 h, but slightly lowers accuracy from 83.0 to 82.8: shorter collection is an efficiency lever, not a complete optimization solution. Linear blending instead raises accuracy to 85.1, confirming that loss reallocation improves training, but still costs 2.59 h. TurnOPD combines both effects, achieving the best accuracy while matching the low wall time of adaptive depth. The validation curves further show that the combined method improves the accuracy–time trajectory rather than only its final checkpoint. This division of labor matches the external and internal budget mismatches identified in Section 4.4.

6.3 Progressive Versus Fixed KL Normalization

Setup. Section 4.3 shows that trajectory-level KL concentrates loss on shallow turns, whereas immediately equalizing all turns may overweight low-support deep estimates. We therefore compare trajectory-level KL, hard turn-level KL, and the linear blend while measuring both Same-Step accuracy and the realized loss budget assigned to the deepest third of valid turns.

Results. Table 4 directly verifies the loss-allocation mechanism. Trajectory-level KL leaves only $3.2/0.7/1.2\%$ of the early/mid/late budget for the deepest third. Hard turn-level KL jumps immediately to roughly one third, improving Avg@4 to 85.0 but discarding the reliability-aware warm-up. The linear blend instead increases α from 0.17 to 0.83 and moves the deep budget gradually from 12.8% to 27.7%, attaining the best Avg@4 (85.1). The measured budget trajectory confirms the intended progressive transition rather than a static reweighting.

7 Conclusion

Vanilla agent 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 both with adaptive rollout depth and progressive turn-normalized supervision, advancing the accuracy–time frontier across ALFWorld, WebShop, and Multi-Hop Search. More broadly, long-horizon supervision should be treated as turn-conditioned decisions rather than flat token positions.

AI USE STATEMENT

We used generative AI tools to polish manuscript language and identify and review related literature. Before the pre-submission review, AI assistance was limited to these tasks; the method, theoretical development, experimental design, implementation, data processing, and interpretation of results were conducted by the authors. The authors take responsibility for the final manuscript, including its claims, references, and results.

REPRODUCIBILITY STATEMENT

The environment interfaces and task-specialized teachers are described in Appendix A. Appendix E provides the training algorithm and controller configuration, and Appendix F describes the data construction. The proof of the contamination-compression bound appears in Appendix C; further experimental results and ablations appear in Appendices G and H.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes, 2024. URL https://arxiv.org/abs/2306.13649.

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*, pp. 41–75. Springer, 2018.

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.

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*, pp. arXiv–2507, 2025.

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

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*, pp. 6609–6625, 2020.

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.

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.

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.

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.

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.

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.

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.

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

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.

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*, 2025a.

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*, 2025b.

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*, pp. 9802–9822, 2023.

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.

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.

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.

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.

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.

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.

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, 2025.

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.

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.

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`.

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*, pp. 2369–2380, 2018.

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.

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.

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.

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.

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.

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 (Shridhar et al., 2021), Multi-Hop Search (consisting of several sub-benchmarks including PopQA (Mallen et al., 2023), NQ (Kwiatkowski et al., 2019), 2WikiMultiHopQA (Ho et al., 2020), HotpotQA (Yang et al., 2018)), and WebShop (Yao et al., 2022).

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 (Côté et al., 2018) 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 (Jin et al., 2025). 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` (Mallen et al., 2023), `Natural Questions` (NQ) (Kwiatkowski et al., 2019), `2WikiMultiHopQA` (Ho et al., 2020), and `HotpotQA` (Yang et al., 2018). 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` (Xi et al., 2025). An episode proceeds search $\rightarrow$ inspect results $\rightarrow$ open a product $\rightarrow$ select attributes $\rightarrow$ `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 (Shao et al., 2024) on the three environments. Figure 5 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.

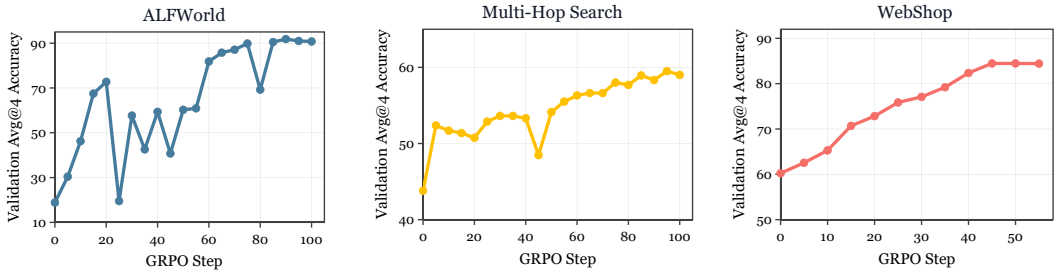

Figure 5: GRPO validation curves for the task-specialized teacher models.

B Extended Diagnostic Analysis

B.1 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

$$G_t = K_t^{\text{fail}} - K_t^{\text{succ}}, \tag{11}$$

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.

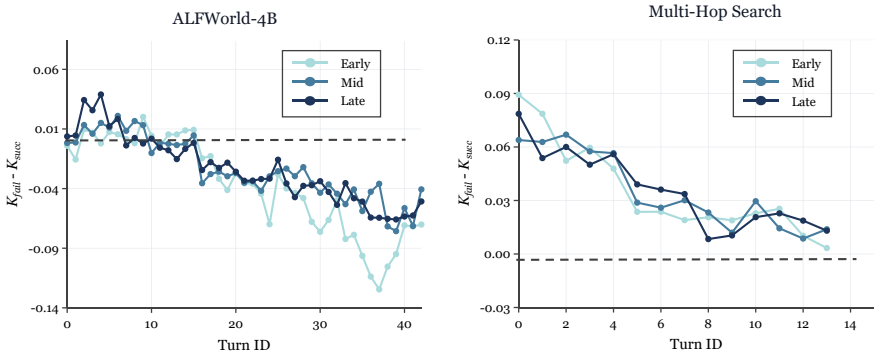

Figure 6: Outcome separation of reverse-KL by turn under vanilla OPD. Each curve plots the phase mean of $G_t = K_t^{\text{fail}} - K_t^{\text{succ}}$ over early ($0 \le$ step < 30), mid ($30 \le$ step < 60), or late ($60 \le$ step ≤ 100) training. Positive values indicate larger KL on failed trajectories.

Figure 6 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.

B.2 Further Implications of Contamination Compression

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

$$K_t = \mathbb{E}_{c\sim\mathcal{C}_t}\left[D_{\mathrm{KL}}(\pi_S(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c))\right], \tag{12}$$

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

Under the forced–free mixture in Equation 2, define the latent free-component disagreement as

$$\Delta_{\mathrm{free}}(c) = D_{\mathrm{KL}}\big(p_S^{\mathrm{free}}(\cdot \mid c) \,\|\, p_T^{\mathrm{free}}(\cdot \mid c)\big). \tag{13}$$

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

$$\rho(c) = \frac{D_{\mathrm{KL}}(\pi_S(\cdot \mid c) \,\|\, \pi_T(\cdot \mid c))}{\Delta_{\mathrm{free}}(c)} \le 1 - \lambda(c). \tag{14}$$

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: $\lambda_t^{\mathrm{fail}} > \lambda_t^{\mathrm{succ}}$. 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

$$G_t \approx (1 - \bar{\lambda}_t^{\mathrm{fail}}) K_t^{\mathrm{fail,free}} - (1 - \bar{\lambda}_t^{\mathrm{succ}}) K_t^{\mathrm{succ,free}}. \tag{15}$$

Equation 15 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 6.

C 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 $\alpha \in [0, 1]$,

$$D_{\mathrm{KL}}(\alpha P_1 + (1-\alpha)P_2 \,\|\, \alpha Q_1 + (1-\alpha)Q_2) \le \alpha D_{\mathrm{KL}}(P_1 \,\|\, Q_1) + (1-\alpha) D_{\mathrm{KL}}(P_2 \,\|\, Q_2). \tag{16}$$

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

$$\begin{aligned}\big(a_1 + a_2\big) \log \frac{a_1 + a_2}{b_1 + b_2} \\ &\le a_1 \log \frac{a_1}{b_1} + a_2 \log \frac{a_2}{b_2} \\ &= \alpha P_1(x) \log \frac{P_1(x)}{Q_1(x)} + (1-\alpha) P_2(x) \log \frac{P_2(x)}{Q_2(x)},\end{aligned} \tag{17}$$

where the factors α and $1-\alpha$ cancel inside the logarithms. Summing over all tokens x gives Equation 16.

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

$$\begin{aligned}D_{\mathrm{KL}}\big(\lambda p_F + (1-\lambda) p_S^{\text{free}} \,\|\, \lambda p_F + (1-\lambda) p_T^{\text{free}}\big) \\ &\le \lambda D_{\mathrm{KL}}(p_F \,\|\, p_F) + (1-\lambda) D_{\mathrm{KL}}(p_S^{\text{free}} \,\|\, p_T^{\text{free}}).\end{aligned} \tag{18}$$

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

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

which proves Equation 3.

D 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

$$\rho(H) = \frac{\sum_{t \le H} v_t w_t}{\sum_{t \le H} c_t w_t + \epsilon}, \tag{20}$$

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 $H^\star_{\text{eff}}$ 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:

$$\rho(H+1) \ge \rho(H) \quad \Longleftrightarrow \quad \frac{v_{H+1}}{c_{H+1}} \ge \rho(H), \tag{21}$$

obtained by cross-multiplying $(A+a)/(B+b) \ge A/B$ 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

$$H^\star_{\text{eff}} \le \tau_c - 1. \tag{22}$$

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

$$F_{\text{succ}}(H) = \Pr(L_{\text{succ}} \le H), \qquad H_{\text{cov}} = \widehat{Q}_p(L_{\text{succ}}) = \min\{H : F_{\text{succ}}(H) \ge p\}, \tag{23}$$

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

$$\tau_{\text{done}} = \min\{H : F_{\text{succ}}(H) = 1\}. \tag{24}$$

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

$$H_{\text{cov}} \le \tau_{\text{done}}. \tag{25}$$

Algorithm 1: TurnOPD training algorithm.

Input: π_θ, π_T, task distribution $\mathcal{D}$, environment turn limit $T_{\max}$, cap bounds $H_{\min}, H_{\max}$, training horizon K, 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 $\bar{H} \leftarrow \varnothing$ (unset) and $H_{\text{cov}} \leftarrow 0$;
for $k = 1$ **to** K **do**
  $\text{probe}_k \leftarrow (k \le K_{\text{warm}}) \vee (k \bmod r_{\text{probe}} = 0)$;
  if probe_k **then**
    $H_{\text{roll}} \leftarrow T_{\max}$; `// full-depth probe, no cap`
  else
    $H_{\text{roll}} \leftarrow \hat{H}_k \leftarrow \text{clip}\big(\text{round}(\bar{H}) + 1,\, H_{\min},\, H_{\max}\big)$;
  Collect on-policy trajectories B_k from π_θ on $\mathcal{D}$ up to H_{roll} turns;
  Query π_T on the supervised tokens in B_k and compute top-K reverse-KL token losses ℓ_i;
  Compute per-turn raw-KL means K_t and survivor counts n_t on B_k;
  $m_t \leftarrow [K_t]_+ \, n_t / n_0, \quad q_t \leftarrow m_t / (\sum_j m_j + \epsilon)$;
  $H_{\text{eff}} \leftarrow \text{round}(\sum_t t \, q_t)$;
  if $\text{probe}_k \wedge |B_k^{\text{succ}}| \ge n_{\min}^{\text{cov}}$ **then**
    Estimate $F_{\text{succ}}(H)$ from successful probe trajectories and set
    $H_{\text{cov}} \leftarrow \min\{H : F_{\text{succ}}(H) \ge p\}$;
  $H_{\text{ctrl}} \leftarrow \max(H_{\text{eff}}, H_{\text{cov}})$;
  if $\bar{H} = \varnothing$ **then**
    $\bar{H} \leftarrow H_{\text{ctrl}}$; `// lazy init at the first probe`
  else
    $\bar{H} \leftarrow (1 - \alpha_{\text{ema}})\, \bar{H} + \alpha_{\text{ema}}\, H_{\text{ctrl}}$;
  $\alpha_k \leftarrow \text{clip}\big((k/K - s)/(e - s),\, 0,\, 1\big)$;
  Construct trajectory-normalized weights w^{traj} and turn-normalized weights w^{turn} on the observed tokens in B_k;
  $w^{\text{blend}} \leftarrow (1 - \alpha_k)\, w^{\text{traj}} + \alpha_k \, w^{\text{turn}}$;
  $L_k \leftarrow \sum_{i \in B_k} w_i^{\text{blend}} \, \ell_i$; update θ by descending $\nabla_\theta L_k$;

Sandwich. The latent target should respect both sides:

$$H^\star = \max(H_{\text{eff}}^\star,\ H_{\text{cov}})\,. \tag{26}$$

Therefore

$$H_{\text{eff}}^\star \le H^\star \le \max(\tau_c - 1, \tau_{\text{done}})\,. \tag{27}$$

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.

E Complete TurnOPD Hyperparameters

Table 5 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.

Table 5: 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

F 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 6 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 $\approx$ 30k question–answer instances; evaluation uses a separate held-out test set (≈ 400 questions) drawn from PopQA, Natural 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.

Table 6: 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

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.

G Additional Experimental Results

G.1 Task-Level Performance Breakdowns

The aggregate results in Table 2 compare methods at common wall-clock and optimization-step cutoffs. Here, we complement those endpoints with the full accuracy–time trajectories and category-level results. These breakdowns indicate whether an aggregate gain persists over training and whether it is broadly shared across task categories or concentrated in a small subset.

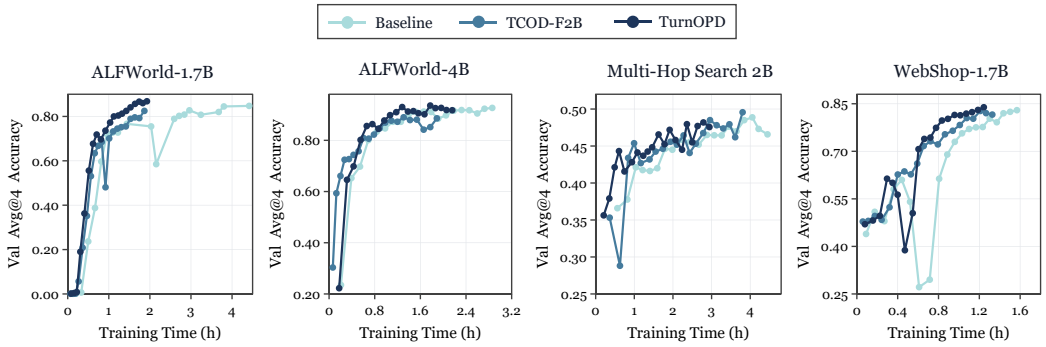

Figure 7: Main-result iso-time efficiency. The x-axis is cumulative training time, computed as the running sum of per training step time. TurnOPD improves the accuracy–time frontier on each task type.

Figure 7 demonstrates that TurnOPD consistently achieves higher or comparable accuracy with reduced training time across all benchmarks. On ALFWorld-1.7B, it shows an early and sustained advantage, while on ALFWorld-4B, all methods approach similar high accuracy, but TurnOPD reaches this level faster. For Multi-Hop Search and WebShop, TurnOPD attains strong performance early and reaches the accuracy plateau in the shortest time range. Overall, TurnOPD offers clear efficiency and accuracy gains throughout the training process.

The ALFWorld breakdown in Table 7 further shows that the Qwen3.5-1.7B student improvement is not driven by a single task type. TurnOPD exceeds TCOD-F2B in all six categories by a relatively uniform 4.92–5.91 points. Compared with vanilla OPD, its largest gain occurs on *Cool* (80.34 versus 65.45), which is also the weakest category for the vanilla student. At Qwen3.5-4B, where vanilla OPD is already close to the teacher reference, the gains are smaller and category dependent: TurnOPD is the best student-training method in five of six categories, with *Place* as the only exception (90.96 versus 91.49 for vanilla OPD). Its overall score can still exceed the teacher reference because improvements on *2Obj*, *Cool*, and *Look* offset the remaining category-level gaps.

Table 8 reveals a more heterogeneous pattern on Multi-Hop Search. TurnOPD improves most clearly on PopQA and 2Wiki, exceeding vanilla OPD by 1.38 and 5.19 points, respectively, while remaining close but slightly lower on NQ and HotpotQA. In contrast, it exceeds TCOD-F2B on all four sources.

Table 7: 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

Table 8: 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

G.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 8 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 non-trivial efficiency, while H_{cov} prevents the controller from selecting overly short rollouts that would limit coverage of successful completions.

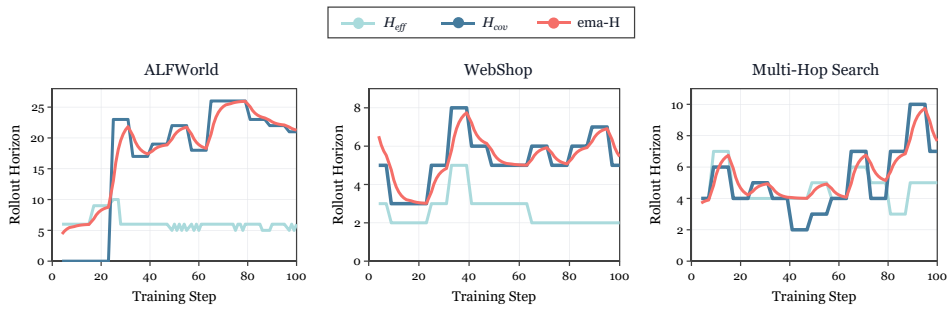

Figure 8: Rollout-depth diagnostic replay. For each TurnOPD run, we plot the survivor-weighted raw-KL centroid H_{eff}, the success-coverage lower bound H_{cov}, and the counterfactual EMA horizon $\hat{H}_{\text{EMA}}$ obtained by replaying $\max(H_{\text{eff}}, H_{\text{cov}})$. Values use the 0-based turn-index convention; the applied rollout cap is approximately $\text{round}(\hat{H}) + 1$.

H Additional Ablation Analysis

H.1 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 9 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:

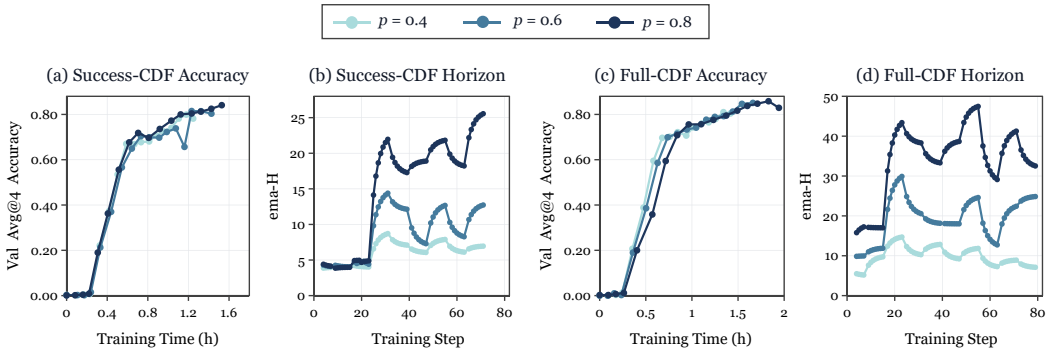

Figure 9: Coverage-floor sensitivity on ALFWorld-1.7B. The four panels show validation accuracy versus cumulative training time and logged EMA horizon for the success-conditioned CDF and the full-population CDF.

(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 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.