ACT FIRST, REASON LATER: ACCELERATING ON-POLICY DISTILLATION FOR MULTI-TURN AGENTS VIA REFERENCE-CONDITIONED INVERSE DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

On-policy distillation (OPD) trains multi-turn language agents with dense teacher supervision on student-generated responses. However, standard think-then-act rollouts require lengthy reasoning before each short action, delaying environment transitions and experience collection. Generating actions directly reduces this delay but can degrade rollout quality. To address this, we propose **ActFirst-OPD**, an act-first, reason-later training framework that decouples environment interaction from full-response generation. The student infers and executes actions through reference-conditioned inverse dynamics using its current interaction context and a reference next observation, and switches to autonomous next-action prediction when the resulting transition deviates from the reference trajectory. From the collected interaction contexts, the student asynchronously generates full think-then-act responses for token-level teacher supervision. Experiments across 0.6B-, 1.7B-, and 4B-parameter Qwen3 students show that ActFirst-OPD achieves average wall-clock training speedups of $2.3\times$ on ALFWorld, $1.8\times$ on WebShop, and $4.9\times$ on ScienceWorld over Vanilla OPD. It matches or exceeds all compared OPD baselines in mean task success rate across eight of nine benchmark–model settings. These results demonstrate that reasoning need not block acting during multi-turn agent distillation.

1 INTRODUCTION

Large language models (LLMs) are evolving from static text generators into agents that reason and act in interactive environments (Yao et al., 2023). In multi-turn tasks, agents execute actions based on current observations, driving environment transitions that produce the next observation for subsequent reasoning and action (Shridhar et al., 2021; Yao et al., 2022; Wang et al., 2022). On-policy distillation (OPD) (Gu et al., 2024; Agarwal et al., 2024) trains such agents with dense token-level teacher supervision on student-generated responses (Wang et al., 2026; Zhou et al., 2026b). Unlike supervised fine-tuning (SFT) on offline trajectories, OPD continually collects turn-level responses under the evolving student policy, covering response prefixes the student may encounter at inference and mitigating exposure bias (Gu et al., 2024; Agarwal et al., 2024).

However, online experience collection remains costly in multi-turn OPD (Zhou et al., 2026b; Liao et al., 2026). In standard think-then-act rollouts, the student completes lengthy reasoning before submitting a short action at each turn, delaying the environment transition (Figure 1(a)). Our profiling of Qwen3-1.7B on ALFWorld shows that reasoning accounts for 81–95% of per-turn rollout time (Figure 1(b)). These delays accumulate across turns, ultimately increasing training time (Figure 1(c)). We characterize this as a reasoning-blocked transition bottleneck in experience collection. Generating actions directly while deferring full-response generation can reduce this delay but may degrade rollout quality. Controlled comparisons show that direct-action rollouts exhibit more unproductive repetition (Figure 1(d, e)) and yield fewer successful trajectories per 100 environment transitions (Figure 1(f)). These observations raise the question: *Can the student act to advance the environment without waiting for full reasoning while maintaining rollout quality?*

To address this challenge, we propose **ActFirst-OPD**, an act-first, reason-later training framework that decouples environment interaction from full-response generation. Our key idea is to use high-

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.

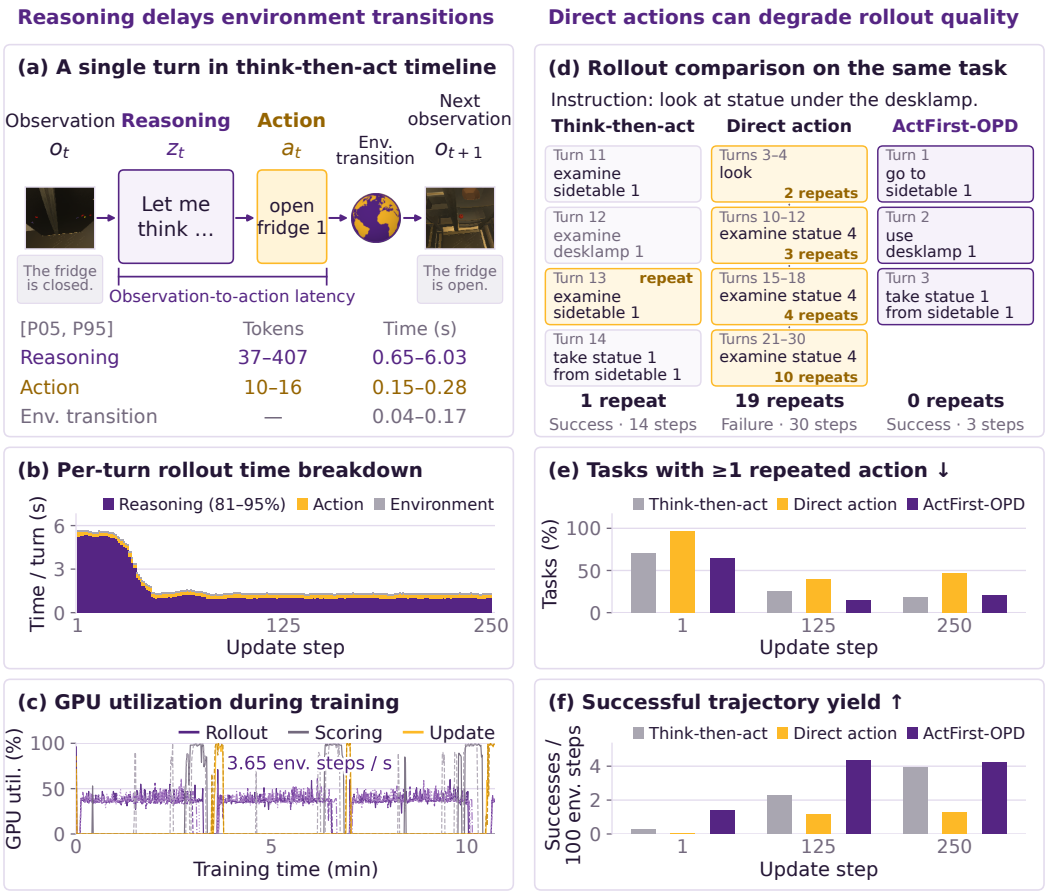


Figure 1: **Rollout latency and quality on ALFWorld with Qwen3-1.7B.** Ranges in (a) denote the 5th–95th percentiles. (c) shows GPU utilization during the first 10 minutes of training. In (d–f), all rollout strategies use the same frozen Vanilla OPD checkpoint at each update step, with the same 3,553 training tasks, and a 30-turn limit. (d) shows selected turns from the same task at update 1.

quality reference trajectories to provide local transition targets for fast action generation, maintaining rollout quality. Drawing on inverse dynamics (Pavse et al., 2020), we have the student infer and execute an action from its current interaction context and a reference next observation. When interaction deviates from the reference trajectory, the student switches to autonomous next-action prediction for the rest of the rollout. While fast actions advance the environment, the same student asynchronously generates full think-then-act responses from the collected interaction contexts without using reference next observations for token-level teacher supervision. This preserves full-response distillation while removing reasoning from the critical path of environment transitions.

The main contributions of this work are summarized below.

- We identify and characterize reasoning-blocked environment transitions in multi-turn OPD and the rollout-quality risks of direct action through profiling and controlled comparisons.
- We introduce ActFirst-OPD, combining reference-conditioned inverse dynamics and next-action prediction with asynchronous full-response generation for distillation. We also derive the idealized rollout speedup and its upper bound, and measure empirical gains.
- Across 0.6B-, 1.7B-, and 4B-parameter Qwen3 students, ActFirst-OPD achieves average wall-clock training speedups of $2.3\times$ on ALFWorld, $1.8\times$ on WebShop, and $4.9\times$ on ScienceWorld over Vanilla OPD under matched hardware and update steps. It matches or exceeds all compared OPD baselines in mean task success rate across eight of nine benchmark–model settings.

2 Related Work

On-Policy Distillation. Early OPD methods train students with dense teacher supervision on student-generated sequences (Gu et al., 2024; Agarwal et al., 2024). Applications span reasoning, strong-to-weak distillation, and integration of domain-specific capabilities (Lu & Thinking Machines Lab, 2025; Qwen Team, 2025; DeepSeek-AI, 2026). Recent studies examine OPD training dynamics and mechanisms (Song & Zheng, 2026; Li et al., 2026c). For multi-turn agents, TCOD (Wang et al., 2026) uses temporal curricula to mitigate trajectory-level KL instability, while TurnOPD (Zhou et al., 2026b) combines adaptive rollout depth with turn-level loss weighting. Our work addresses delays from pre-action reasoning and the rollout-quality risks of direct action generation. Additional comparisons with recent OPD methods are provided in Appendix D.

Multi-Turn LLM Agents. LLM agents interleave reasoning and actions, as exemplified by ReAct (Yao et al., 2023), for tasks such as embodied planning (Shridhar et al., 2021) and web navigation (Yao et al., 2022). Recent coding agents, including Claude Code (Anthropic, 2025b) and Codex (OpenAI, 2025), and agent harnesses (Anthropic, 2025a; Ning et al., 2026) support long-horizon, tool-using workflows. Despite these advances, training such agents remains challenging due to credit assignment under sparse or delayed rewards (Feng et al., 2025) and compounding distribution shift across turns (Xue et al., 2026).

Inverse Dynamics. Inverse dynamics models infer actions from state transitions and have been used for self-supervised reinforcement learning (Pathak et al., 2017) and imitation without expert action annotations (Torabi et al., 2018). Recent embodied-agent methods use predicted visual futures or their representations to guide action generation (Hu et al., 2025; Li et al., 2026b). Most closely related to our formulation, RIDM (Pavse et al., 2020) infers actions from the learner's current observation and the expert's next observation to drive environment interaction without expert actions. We adapt this conditioning scheme to multi-turn OPD, using reference next observations to guide fast student actions and maintain rollout quality when collecting interaction contexts.

3 Preliminaries

Multi-Turn Agent Interaction. We consider a language agent interacting with an environment to complete a task specified by an instruction q. At turn t, the agent receives an observation o_t and maintains the history of preceding interactions,

$$h_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}), \qquad h_1 = \varnothing, \tag{1}$$

where a_i is the action submitted to the environment at turn i; past reasoning is excluded from h_t. We write the agent input schematically as interaction context $\boldsymbol{c}_t = q \oplus h_t \oplus o_t$, where $\oplus$ denotes concatenation and benchmark-specific system instructions and formatting are implicit. In standard think-then-act interaction (Yao et al., 2023), the student policy $\pi_{\boldsymbol{\theta}}$ generates a full response

$$y_t = (z_t, a_t) \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{c}_t), \tag{2}$$

where z_t denotes the reasoning segment and a_t denotes the executable action segment; only a_t is submitted to the environment. The environment executes a_t and returns the next observation o_{t+1}. Interaction ends when the environment terminates or the prescribed turn limit is reached, yielding the trajectory $\tau = (o_1, a_1, o_2, \ldots, o_T, a_T, o_{T+1})$ over T interaction turns. Prompt construction, history management, and thinking budgets are detailed in Appendices B.1, B.2, and B.3, respectively.

On-Policy Distillation for Multi-Turn Agents. Multi-turn OPD trains a student policy $\pi_{\boldsymbol{\theta}}$ with a frozen teacher $\pi_{\boldsymbol{\phi}}$ on responses collected from online student rollouts. Let $\boldsymbol{x}_t = (x_{t,1}, \ldots, x_{t,L_t})$ denote the tokenization of the full response $y_t = (z_t, a_t)$, where L_t is the number of response tokens at turn t. For each response position $j \in \{1, \ldots, L_t\}$, define the token context

$$\boldsymbol{c}_{t,j} = \boldsymbol{c}_t \oplus \boldsymbol{x}_{t,<j}, \tag{3}$$

where $\boldsymbol{x}_{t,<j} = (x_{t,1}, \ldots, x_{t,j-1})$ is the response prefix preceding token $x_{t,j}$. Following prior studies (Wang et al., 2026), multi-turn OPD minimizes the expected student-to-teacher reverse Kullback–Leibler (KL) divergence over supervised token contexts,

$$\mathcal{L}_{\mathrm{OPD}}(\boldsymbol{\theta}) = \mathbb{E}\left[D_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{c}_{t,j}) \,\|\, \pi_{\boldsymbol{\phi}}(\cdot \mid \boldsymbol{c}_{t,j})\right)\right]. \tag{4}$$

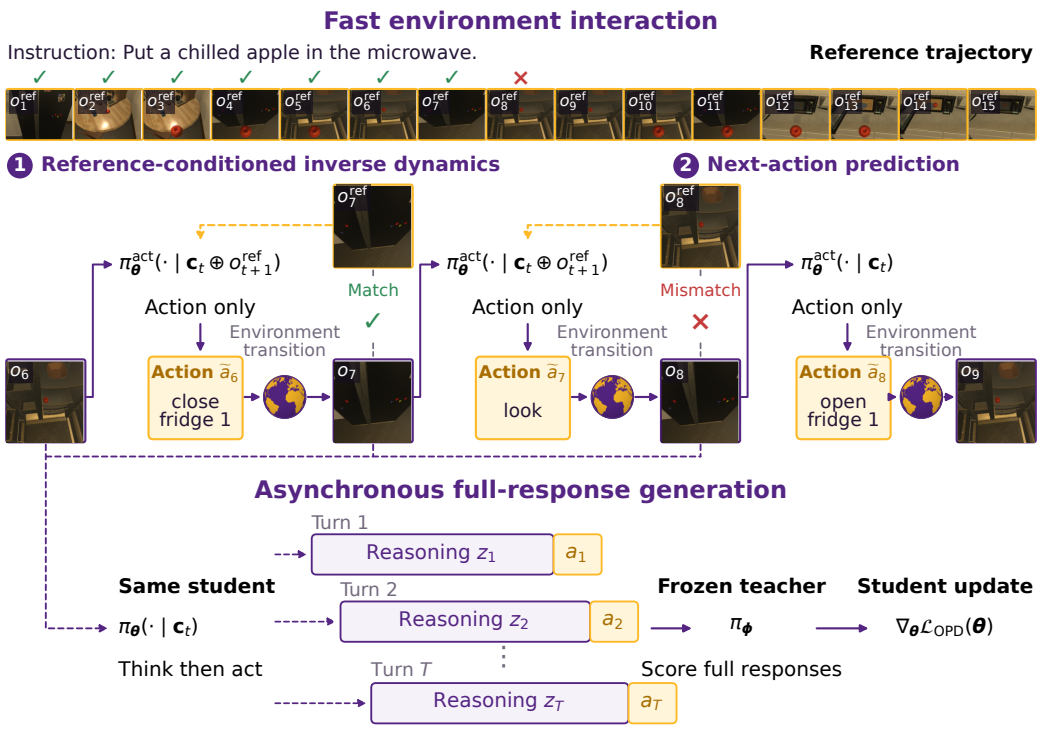


Figure 2: **Overview of ActFirst-OPD.** Top: Reference-conditioned inverse dynamics generates fast actions $\tilde{a}_t$ for environment interaction, with autonomous next-action prediction once the rollout diverges from the reference trajectory. Bottom: The same student asynchronously generates full responses $y_t = (z_t, a_t)$ from collected interaction contexts for teacher supervision.

The expectation is over collected turn-level full responses y_t at turns t of interaction trajectories τ, and supervised token positions j within each response. For each token context, the reverse KL is

$$D_{\mathrm{KL}}\left(\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{c}_{t,j}) \,\|\, \pi_{\boldsymbol{\phi}}(\cdot \mid \boldsymbol{c}_{t,j})\right) = \sum_{v \in \mathcal{V}} \pi_{\boldsymbol{\theta}}(v \mid \boldsymbol{c}_{t,j}) \log \frac{\pi_{\boldsymbol{\theta}}(v \mid \boldsymbol{c}_{t,j})}{\pi_{\boldsymbol{\phi}}(v \mid \boldsymbol{c}_{t,j})}, \tag{5}$$

where $\mathcal{V}$ denotes the token vocabulary. In practice, we optimize a sampled policy surrogate of Eq. 4; the update rule, clipping, masking, and loss normalization are detailed in Appendix B.4.

4 ActFirst-OPD

ActFirst-OPD decouples environment interaction from full-response generation, as illustrated in Figure 2. The student advances the environment through actions conditioned on reference next observations, with autonomous next-action prediction once the rollout diverges from the reference trajectory. Meanwhile, full responses are generated asynchronously from collected contexts for distillation, allowing subsequent environment interactions to proceed without waiting for reasoning.

4.1 Reference-Conditioned Inverse-Dynamics Rollout

Given a high-quality offline reference trajectory for training task q, let $o_1^{\mathrm{ref}}, o_2^{\mathrm{ref}}, \ldots$ denote its observations. While the rollout remains aligned with the reference trajectory, the reference next observation o_{t+1}^{ref} serves as a local transition target. Following the inverse-dynamics formulation (Pavse et al., 2020), the student infers an action from its actual interaction context $\boldsymbol{c}_t$ and this target:

$$\tilde{a}_t \sim \pi_{\boldsymbol{\theta}}^{\mathrm{act}}\left(\cdot \mid \boldsymbol{c}_t \oplus o_{t+1}^{\mathrm{ref}}\right), \tag{6}$$

where $\pi_{\boldsymbol{\theta}}^{\mathrm{act}}$ denotes the same student prompted to generate only an action, without preceding reasoning. During interaction, h_t records actual observations and executed fast actions $\tilde{a}_1, \ldots, \tilde{a}_{t-1}$.

Thus, $\boldsymbol{c}_t = q \oplus h_t \oplus o_t$ remains the student's actual interaction context. Executing $\tilde{a}_t$ produces o_{t+1} and advances the rollout.

4.2 Autonomous Next-Action Prediction Fallback

After executing a reference-conditioned action $\tilde{a}_t$, we apply benchmark-specific transition checks (Appendix B.5) to determine whether the resulting observation o_{t+1} aligns with the target o_{t+1}^{ref}. Verification concerns the transition outcome and does not require the generated action to match the reference action. If verification succeeds, reference guidance continues at the next turn.

Once verification fails, reference-conditioned inverse dynamics is disabled for the remainder of the rollout. At each subsequent turn, the student performs autonomous next-action prediction (NAP) using its interaction context alone:

$$\tilde{a}_t \sim \pi_{\boldsymbol{\theta}}^{\mathrm{act}}(\cdot \mid \boldsymbol{c}_t). \tag{7}$$

Interaction continues from the actual state reached by the student, preserving action-only generation while avoiding continued reliance on the inapplicable reference trajectory.

4.3 Asynchronous Full-Response Generation

For each interaction context $\boldsymbol{c}_t$ collected during fast rollouts, the student generates a full think-then-act response $y_t = (z_t, a_t) \sim \pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{c}_t)$ using the standard agent input, without reference next observations. Full-response generation proceeds asynchronously, allowing requests from different turns to overlap with one another and with subsequent environment interactions. The action a_t need not match the executed fast action $\tilde{a}_t$ and is not submitted to the environment.

The training batch consists of student-sampled full responses at contexts visited by the fast rollout. The frozen teacher $\pi_{\boldsymbol{\phi}}$ scores each full response under the same token contexts $\boldsymbol{c}_{t,j}$ defined in Section 3. These turn-level samples are used to optimize the OPD objective in Eq. 4. This preserves full-response supervision while removing full-response generation from the dependency chain of environment transitions. At evaluation time, the student uses standard think-then-act interaction without reference next observations. We discuss the context distribution induced by fast rollouts and its implications for on-policy distillation in Appendix E.

Idealized Rollout Efficiency. We analyze rollout completion time, measured until both environment interaction and generation of all corresponding full responses have finished. Consider a fixed T-turn rollout with constant generation latencies ℓ_{fast} and ℓ_{full} for fast actions and full responses, respectively, including prefill and autoregressive decoding, where $0 < \ell_{\mathrm{fast}} \leq \ell_{\mathrm{full}}$. In the idealized model, both requests start as soon as $\boldsymbol{c}_t$ becomes available. We neglect environment and other non-generation overheads and assume sufficient concurrency so that overlapping requests do not increase their generation latencies.

Proposition 1 (Idealized rollout speedup)**.** *Under the above assumptions, standard think-then-act rollout takes $T\ell_{\mathrm{full}}$, whereas ActFirst-OPD completes in $(T-1)\ell_{\mathrm{fast}} + \ell_{\mathrm{full}}$. The resulting idealized rollout speedup satisfies*

$$S_{\mathrm{ideal}}(T) = \frac{T\ell_{\mathrm{full}}}{(T-1)\ell_{\mathrm{fast}} + \ell_{\mathrm{full}}} \leq \min\left\{T,\ \frac{\ell_{\mathrm{full}}}{\ell_{\mathrm{fast}}}\right\}. \tag{8}$$

The proof of Proposition 1 is provided in Appendix A.1. Longer-horizon rollouts amortize the final full-response generation cost, with $S_{\mathrm{ideal}}(T)$ approaching $\ell_{\mathrm{full}}/\ell_{\mathrm{fast}}$ as T increases (Eq. 20). ActFirst-OPD additionally generates a fast action at each turn, so realized speedup requires sufficient serving concurrency for asynchronous overlap to offset this extra workload (Appendix A.2, Eq. 28).

5 Experiments

5.1 Experimental Setup

Tasks, Models, and Reference Trajectories. We evaluate on ALFWorld (Shridhar et al., 2021) for embodied planning, WebShop (Yao et al., 2022) for web navigation, and ScienceWorld (Wang

Table 1: **Task performance and training cost on ALFWorld.** Overall SR is reported as mean ± standard deviation across three evaluation seeds. Training time is in hours, with speedup relative to Vanilla OPD for the same student size. Among OPD methods, **bold** and underlined values mark the best and second-best results per metric, respectively; shading highlights the shortest training time.

Model		Method	ALFWorld												
Teacher	Student		Pick	Look	Clean	Heat	Cool	Pick2	Seen		Unseen		All		Training Wall-clock Time (Speedup)
									SR↑	Round↓	SR↑	Round↓	SR↑	Round↓	
Qwen3 Series															
-	0.6B	Zero-Shot	0.00	6.45	0.00	0.00	0.00	0.00	0.00	30.00	1.49	29.69	0.73	29.85	N/A
-	1.7B		40.68	16.13	13.79	12.82	8.70	4.88	17.86	27.68	17.16	27.28	17.52	27.48	N/A
-	4B		89.83	77.42	60.34	58.97	28.26	63.41	55.71	21.92	71.64	20.10	63.50	21.03	N/A
8B	-	GiGPO	98.31	90.32	94.83	79.49	86.96	73.17	89.29	10.04	87.31	11.18	88.32	10.59	N/A
8B-GiGPO	0.6B	Vanilla OPD	80.79	55.91	66.67	59.83	52.90	34.96	61.67	17.52	59.20	17.90	60.46 ± 3.15	17.70	4.18 h (×1.00)
		TCOD-F2B	85.88	53.76	67.24	55.56	60.14	32.52	63.81	16.22	59.45	17.41	61.68 ± 1.59	16.80	2.15 h (×1.94)
		TurnOPD	72.88	**60.22**	48.85	39.32	53.62	21.95	51.43	19.18	50.00	19.72	50.73 ± 2.89	19.44	2.87 h (×1.46)
		Ours w/o ID	72.88	55.91	51.15	26.50	34.78	13.01	43.81	20.89	45.02	21.13	44.40 ± 6.38	21.01	2.30 h (×1.82)
		ActFirst-OPD (Ours)	**87.01**	**60.22**	**75.86**	**65.81**	**66.67**	**43.09**	**71.67**	**14.90**	**65.42**	**16.51**	**68.61 ± 0.89**	**15.69**	**1.93 h (×2.17)**
	1.7B	Vanilla OPD	92.66	63.44	74.14	65.81	56.52	39.84	70.48	14.70	64.68	16.79	67.64 ± 1.52	15.72	4.46 h (×1.00)
		TCOD-F2B	92.66	50.54	81.61	64.96	55.80	**57.72**	73.33	14.33	66.92	15.74	70.19 ± 1.38	15.02	2.32 h (×1.92)
		TurnOPD	90.96	63.44	79.31	**67.52**	64.49	54.47	74.76	14.01	69.40	15.30	72.14 ± 3.11	14.64	3.21 h (×1.39)
		Ours w/o ID	74.01	59.14	62.07	52.14	55.07	30.08	57.86	18.10	55.97	18.97	56.93 ± 3.85	18.53	2.48 h (×1.80)
		ActFirst-OPD (Ours)	**93.79**	**82.80**	**85.06**	66.67	**73.19**	54.47	**80.95**	**12.62**	**73.88**	**14.27**	**77.49 ± 1.20**	**13.43**	**1.95 h (×2.28)**
	4B	Vanilla OPD	92.66	80.65	**83.91**	72.65	74.64	68.29	84.05	11.40	75.62	13.55	79.93 ± 0.73	12.46	4.56 h (×1.00)
		TCOD-F2B	92.66	88.17	72.41	58.12	61.59	73.17	79.29	12.65	70.15	15.11	74.82 ± 1.10	13.85	2.72 h (×1.68)
		TurnOPD	93.79	83.87	79.89	**79.49**	72.46	76.42	84.52	11.58	78.36	13.13	81.51 ± 1.47	12.34	3.86 h (×1.18)
		Ours w/o ID	**95.48**	84.95	75.86	65.81	63.77	60.98	80.00	12.28	70.65	14.75	75.43 ± 0.92	13.49	2.59 h (×1.76)
		ActFirst-OPD (Ours)	92.66	**90.32**	78.74	71.79	**81.88**	**78.86**	**86.19**	**11.01**	**78.86**	**12.82**	**82.60 ± 1.12**	**11.90**	**1.93 h (×2.37)**

et al., 2022) for scientific reasoning, with respective interaction limits of 30, 15, and 30 turns. Following prior multi-turn OPD studies (Wang et al., 2026; Zhou et al., 2026b), we train Qwen3-0.6B, 1.7B, and 4B students (Qwen Team, 2025) using a task-specialized Qwen3-8B teacher trained with GiGPO (Feng et al., 2025) for each benchmark. ActFirst-OPD uses offline reference trajectories selected from public demonstrations, model-generated rollouts, and oracle solutions. Data splits are detailed in Appendix C.1; teacher preparation and reference construction are described in Appendix C.2.

Baselines and Implementation. We compare against Vanilla OPD, TCOD-F2B (Wang et al., 2026), and TurnOPD (Zhou et al., 2026b), with results from zero-shot students and task-specialized teachers provided for reference. We also include Ours w/o ID, which replaces reference-conditioned inverse dynamics (ID) with direct action generation without references while retaining asynchronous full-response generation. Within each benchmark–model setting, methods share student initialization, the teacher, and environment settings, while retaining their respective rollout and optimization rules. The main comparison in Section 5.2 uses 250 training updates, rollout batches of 16 tasks, and optimization batches of 64 turn-level full responses. All OPD methods are implemented using Trinity-RFT (Pan et al., 2025), with vLLM (Kwon et al., 2023) for inference and VERL (Sheng et al., 2025) for optimization. Each training run uses eight NVIDIA A100-SXM4-80GB GPUs: four for student rollouts, two for teacher scoring, and two for student optimization.

Evaluation Protocols and Metrics. We evaluate on 140 seen and 134 unseen ALFWorld tasks, 100 held-out WebShop tasks, and 150 held-out ScienceWorld tasks. All models use standard think-then-act interaction without references during evaluation. We report success rate (SR), average interaction turns (Round), and training wall-clock time; WebShop and ScienceWorld additionally report task scores to measure partial completion. Training wall-clock time excludes teacher training, reference construction, and evaluation. Training speedup is Vanilla OPD's training wall-clock time divided by the method's time for the same benchmark and student size. Task-performance means and standard deviations are computed across three evaluation seeds (42, 43, and 44) for the same trained checkpoint.

5.2 Main Results

Across the nine benchmark–model settings, ActFirst-OPD completes 250 updates faster than Vanilla OPD, TCOD-F2B, and TurnOPD, while matching or exceeding Vanilla OPD's mean SR in eight

Table 2: **Task performance and training cost on WebShop and ScienceWorld.** Score and SR are reported as mean ± standard deviation across three evaluation seeds. Training time is in hours, with speedup relative to Vanilla OPD for the same benchmark and student size. Among OPD methods, **bold** and underlined values mark the best and second-best Score, SR, and training time, respectively; shading highlights the shortest training time.

Model		Method	WebShop				ScienceWorld			
Teacher	Student		Score↑	SR↑	Round	Training Wall-clock Time (Speedup)	Score↑	SR↑	Round	Training Wall-clock Time (Speedup)
Qwen3 Series										
-	0.6B	Zero-Shot	49.54	11.00	4.56	N/A	5.59	0.00	19.86	N/A
-	1.7B		51.10	15.67	5.22	N/A	22.28	6.89	19.65	N/A
-	4B		45.37	13.33	6.75	N/A	44.36	20.67	19.19	N/A
8B	-	GiGPO	71.54	43.33	4.28	N/A	55.58	35.33	15.40	N/A
8B-GiGPO	0.6B	Vanilla OPD	**69.29 ± 1.23**	36.67 ± 2.08	4.32	2.81 h (×1.00)	35.35 ± 0.91	10.00 ± 1.33	20.45	10.31 h (×1.00)
		TCOD-F2B	67.79 ± 1.57	36.33 ± 1.15	4.23	2.42 h (×1.16)	35.39 ± 1.02	10.00 ± 1.33	18.52	6.30 h (×1.64)
		TurnOPD	67.61 ± 1.63	36.33 ± 2.31	4.31	2.88 h (×0.97)	35.45 ± 0.60	10.22 ± 0.39	19.58	6.31 h (×1.63)
		Ours w/o ID	67.93 ± 1.11	36.00 ± 1.00	4.15	**1.40 h (×2.01)**	**39.16 ± 0.27**	11.11 ± 1.39	15.04	2.15 h (×4.79)
		ActFirst-OPD (Ours)	68.35 ± 0.44	**38.00 ± 1.00**	4.76	1.46 h (×1.93)	37.06 ± 1.47	**12.67 ± 1.77**	18.15	**2.01 h (×5.12)**
	1.7B	Vanilla OPD	69.50 ± 0.43	**40.33 ± 0.58**	4.04	3.10 h (×1.00)	45.47 ± 1.71	22.89 ± 2.14	15.24	10.37 h (×1.00)
		TCOD-F2B	68.83 ± 0.59	38.33 ± 1.53	4.15	2.67 h (×1.16)	46.06 ± 0.26	22.45 ± 0.39	15.77	6.53 h (×1.59)
		TurnOPD	69.50 ± 0.64	39.67 ± 1.15	3.99	2.77 h (×1.12)	43.61 ± 1.62	20.44 ± 0.77	15.50	6.37 h (×1.63)
		Ours w/o ID	68.44 ± 1.93	38.67 ± 2.08	4.06	**1.70 h (×1.82)**	43.38 ± 1.61	20.00 ± 2.41	15.05	2.56 h (×4.06)
		ActFirst-OPD (Ours)	**69.76 ± 1.11**	39.33 ± 1.53	3.99	1.72 h (×1.80)	**46.57 ± 2.38**	**24.00 ± 1.76**	15.50	**2.16 h (×4.80)**
	4B	Vanilla OPD	**69.90 ± 1.12**	**42.00 ± 1.00**	4.23	3.96 h (×1.00)	50.74 ± 0.13	29.34 ± 1.15	16.49	12.40 h (×1.00)
		TCOD-F2B	69.25 ± 1.15	**42.00 ± 1.00**	4.12	3.51 h (×1.13)	53.47 ± 2.14	30.22 ± 1.39	17.25	8.47 h (×1.46)
		TurnOPD	69.64 ± 1.76	41.33 ± 2.52	4.17	3.52 h (×1.13)	51.64 ± 1.53	30.00 ± 3.06	16.49	7.93 h (×1.56)
		Ours w/o ID	69.07 ± 0.49	41.67 ± 1.15	4.10	**2.15 h (×1.84)**	51.16 ± 1.60	27.78 ± 2.53	17.12	3.25 h (×3.82)
		ActFirst-OPD (Ours)	69.34 ± 0.45	**42.00 ± 1.00**	4.18	2.32 h (×1.71)	**53.56 ± 1.25**	**31.33 ± 1.34**	16.29	**2.63 h (×4.72)**

(Tables 1 and 2). Averaging the per-size training speedups over Vanilla OPD gives $2.3\times$, $1.8\times$, and $4.9\times$ on ALFWorld, WebShop, and ScienceWorld, respectively.

ActFirst-OPD achieves the highest mean SR among the compared OPD methods at all three student sizes on ALFWorld and ScienceWorld. On ALFWorld, its gains over Vanilla OPD are 8.15, 9.85, and 2.67 percentage points for 0.6B, 1.7B, and 4B, respectively. These SR gains may partly stem from reference-conditioned inverse dynamics promoting task progress during rollout collection and providing more useful contexts for distillation, consistent with the ablation and rollout-quality results (Table 3; Figure 4). On WebShop, ActFirst-OPD achieves a $1.80\times$ training speedup over Vanilla OPD with only a 1.00-percentage-point decrease in mean SR for Qwen3-1.7B.

While Tables 1 and 2 compare performance after 250 training updates, we also compare evaluation SR under comparable training-time budgets using performance–time curves on ALFWorld (Figure 5 in Appendix F). At approximately 1.5 hours of training, ActFirst-OPD achieves the highest mean evaluation SR among the compared OPD methods at all three student sizes, with particularly large gains over Vanilla OPD and TurnOPD.

5.3 Ablation Studies

We evaluate three ablation designs. **Ours w/o ID** replaces reference-conditioned inverse dynamics with direct action generation without reference next observations. **w/o NAP Fallback** terminates rollouts after failed transition consistency checks instead of continuing with autonomous next-action prediction (NAP). **Reference-Prefix Replay** executes either the first 50% of each reference action sequence, followed by NAP, or the full sequence. These variants examine reference conditioning, continuation after divergence, and the use of reference actions for rollout collection, respectively.

Table 3 reports results averaged across student sizes. ActFirst-OPD has the highest mean SR and lowest mean SR rank on all three benchmarks. Removing ID lowers mean SR by 17.31, 1.00, and 3.04 percentage points on ALFWorld, WebShop, and ScienceWorld, respectively; training time increases from 1.94 to 2.45 hours on ALFWorld and from 2.27 to 2.65 hours on ScienceWorld. Removing NAP fallback lowers mean SR by 1.05, 3.11, and 2.82 percentage points, respectively, suggesting that contexts collected after divergence remain useful for distillation.

Both reference-replay variants also lower mean SR on all three benchmarks. Compared with ActFirst-OPD, the 50% variant increases training time from 1.83 to 2.18 hours on WebShop and

Table 3: **Ablation results averaged across Qwen3-0.6B, 1.7B, and 4B.** Red subscripts show SR drops from ActFirst-OPD in percentage points. Time denotes training wall-clock time. Rank averages per-size SR ranks, using average ranks for ties. **Bold** and underlined values mark the best and second-best results per benchmark and metric, respectively.

Method	ALFWorld			WebShop			ScienceWorld		
	Avg. SR↑	Time (h)↓	Rank↓	Avg. SR↑	Time (h)↓	Rank↓	Avg. SR↑	Time (h)↓	Rank↓
w/o ID	$58.92_{\downarrow 17.31}$	2.45	4.67	$38.78_{\downarrow 1.00}$	1.75	3.17	$19.63_{\downarrow 3.04}$	2.65	3.33
w/o NAP Fallback	$75.18_{\downarrow 1.05}$	1.25	2.33	$36.67_{\downarrow 3.11}$	2.09	4.50	$19.85_{\downarrow 2.82}$	2.33	3.00
Reference-Prefix Replay (50%)	$68.94_{\downarrow 7.29}$	1.90	4.33	$39.11_{\downarrow 0.67}$	2.18	2.67	$18.89_{\downarrow 3.78}$	2.87	3.67
Reference-Prefix Replay (100%)	$75.87_{\downarrow 0.36}$	**1.11**	2.00	$37.89_{\downarrow 1.89}$	**1.66**	3.17	$18.45_{\downarrow 4.22}$	**1.73**	4.00
ActFirst-OPD (Ours)	**76.23**	1.94	**1.67**	**39.78**	1.83	**1.50**	**22.67**	2.27	**1.00**

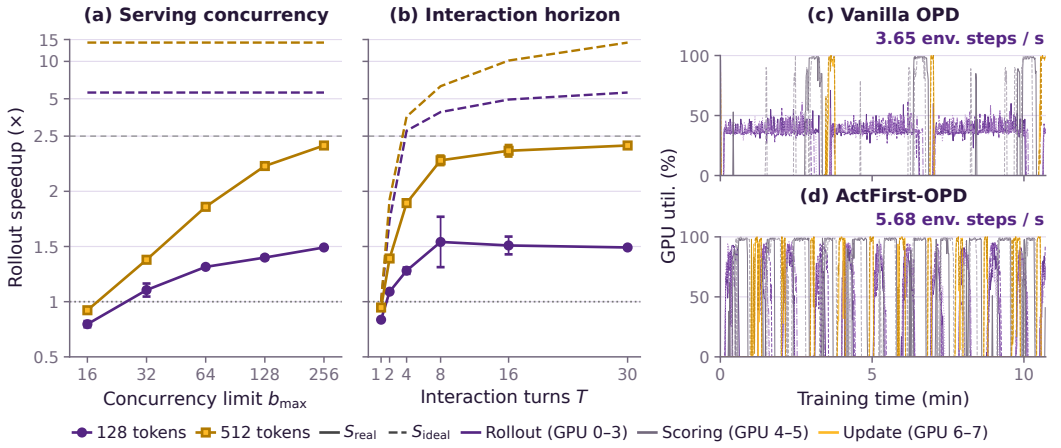


Figure 3: **Rollout efficiency on ALFWorld with Qwen3-1.7B.** (a,b) Measured rollout speedup (solid; mean ± standard deviation over three paired repeats) and idealized estimates (dashed), varying $b_{\max}$ at $T = 30$ in (a) and T at $b_{\max} = 256$ in (b). Speedup axes are linear up to 2.5 and logarithmic above. (c,d) GPU utilization for Vanilla OPD and ActFirst-OPD; annotations show full-training environment-transition throughput.

from 2.27 to 2.87 hours on ScienceWorld. Full replay reduces mean training time on all three benchmarks but lowers mean SR by 0.36 percentage points on ALFWorld, with larger drops of 1.89 and 4.22 points on WebShop and ScienceWorld, respectively. The larger gaps on WebShop and ScienceWorld may reflect a greater benefit from learning to handle deviations from reference trajectories. Full replay follows fixed reference actions, whereas ActFirst-OPD continues after divergence, collecting interaction contexts that support distillation on feedback from student-generated actions.

5.4 Rollout Efficiency and Quality Analysis

Where Does the Speedup Come From? The observed training speedup comes from both higher environment-transition throughput and fewer environment transitions. Asynchronous full-response generation (Section 4.3) can increase throughput, while reference-conditioned inverse dynamics can improve task progress and reduce inefficient interactions.

To isolate the benefit of asynchronous generation, we use fixed ALFWorld interaction contexts with Qwen3-1.7B, 16 tasks per batch, 16-token fast actions, and 128- or 512-token full responses. We vary the cap $b_{\max}$ on concurrent generation requests, shared by fast-action and full-response requests, and the number of virtual turns per task T, timing each rollout until all requests finish. Rollout speedup is the think-then-act completion time divided by the ActFirst-OPD completion time. Figure 3(a) shows that increasing $b_{\max}$ from 16 to 256 raises speedup from $0.80\times$ to $1.49\times$ for 128-token responses and from $0.92\times$ to $2.42\times$ for 512-token responses. At $b_{\max} = 16$, both response lengths yield speedups below one, indicating that overlap does not offset the additional

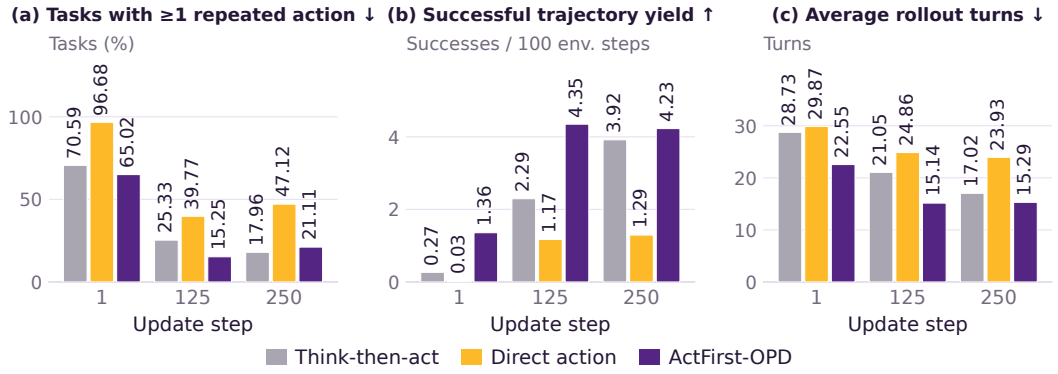


Figure 4: **Rollout quality on ALFWorld.** At each update, all strategies use the same frozen Qwen3-1.7B checkpoint on 3,553 training tasks with a 30-turn maximum rollout horizon.

fast-action generation cost in this setting. Increasing $b_{\max}$ enables speedups in this experiment, but the required serving concurrency depends on the workload and hardware. The larger full-response to fast-action token ratio (32 versus 8) provides more overlap opportunities and reduces the relative decoding overhead of fast actions. At $b_{\max} = 256$, speedup initially grows with T and then levels off (Figure 3(b)). The initial increase and diminishing returns are consistent with amortizing the final full-response cost in Proposition 1. Limited serving concurrency can further constrain speedup and contribute to the gap from the idealized estimates (Appendix A.2).

In 250-update training runs with matched rollout batch size and serving configuration, ActFirst-OPD increases environment-transition throughput from 3.65 to 5.68 transitions per second ($1.56\times$; Figure 3(c,d)). Across the full-training measurement windows, it also executes 30.45% fewer transitions (38,616 versus 55,525; Table 9). Since elapsed time equals transition count divided by throughput, the speedup over these windows is $1.56/(1 - 0.3045) \approx 2.24$. This is close to the $2.28\times$ training speedup for Qwen3-1.7B in Table 1, measured from process launch to exit in separate runs.

How Does ActFirst-OPD Affect Rollout Quality? We hold the student checkpoint fixed and vary only the rollout strategy: think-then-act, direct action without reference next observations, or ActFirst-OPD's reference-conditioned interaction with autonomous fallback. At each of updates 1, 125, and 250, all strategies use the same frozen Qwen3-1.7B Vanilla OPD checkpoint on all 3,553 ALFWorld training tasks with a 30-turn limit. We measure the fraction of tasks with at least one unproductive action repetition, successful trajectories per 100 environment transitions (successful trajectory yield), and mean rollout turns. Measurement protocols are detailed in Appendix C.5.

Compared with direct action, ActFirst-OPD improves rollout quality at every checkpoint, with a lower fraction of tasks exhibiting unproductive action repetition, higher successful trajectory yield, and fewer rollout turns (Figure 4). Compared with think-then-act, it yields more successful trajectories per 100 transitions and uses fewer turns, with broadly comparable task-level repetition. Based on these results, reference-conditioned inverse dynamics may improve rollout quality.

6 Conclusion

We presented ActFirst-OPD, which combines reference-conditioned inverse dynamics, autonomous next-action prediction, and asynchronous full-response generation. Across three benchmarks and three Qwen3 student sizes, it reduces training time relative to Vanilla OPD while matching or exceeding all compared OPD baselines in mean success rate across eight of nine settings. These results highlight that reasoning need not block acting during multi-turn agent distillation.

Despite these gains, our method assumes access to high-quality offline reference trajectories; settings where such references are difficult to obtain are beyond the scope of this work. Reference next observations currently serve only as rollout guidance, which is disabled after divergence. Future work could use reference transitions for auxiliary supervision or realign rollouts with their reference trajectories after divergence.

AI USE STATEMENT

The authors used OpenAI language-model systems, including Codex, as interactive assistants during this project. These tools assisted with literature discovery, experimental design, code development, data processing and visualization, mathematical derivations and proof checking, and manuscript drafting and revision. The human authors made the final research and methodological decisions and take full responsibility for the manuscript, experimental results, and theoretical claims.

ETHICS STATEMENT

This work studies efficient training of language agents using established benchmarks: ALFWorld, WebShop, and ScienceWorld. Our experiments involve benchmark environments and pretrained language models, without human-participant studies or real-world deployment. Reducing training costs can broaden access to agent research, but may also lower barriers to developing agents for harmful uses. Student models may inherit biases, errors, or undesirable behaviors from their teachers and reference trajectories. Applications beyond the evaluated benchmarks therefore require appropriate safety evaluation and controls on environment and tool access.

REPRODUCIBILITY STATEMENT

Section 5 describes the benchmarks, models, computational resources, and evaluation protocols. The assumptions and proofs underlying our efficiency analysis are provided in Appendix A. Appendix B provides the prompt templates, history management, thinking-budget decoding, OPD optimization, and transition consistency checks. Appendix C describes benchmark configurations, teacher training and reference construction, training and evaluation hyperparameters, and rollout measurement protocols. The accompanying anonymous code repository is available at `https://anonymous.4open.science/r/ActFirst-OPD`.

REFERENCES

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

Anthropic. Effective harnesses for long-running agents. 2025a. https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents.

Anthropic. Claude 3.7 Sonnet and Claude Code. 2025b. https://www.anthropic.com/news/claude-3-7-sonnet.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. TextWorld: A learning environment for text-based games. In *Computer Games*, volume 1017 of *Communications in Computer and Information Science*, pp. 41–75, 2019.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 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.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video Prediction Policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267, pp. 24328–24346, 2025.

Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. DeepSpeed Ulysses: System optimizations for enabling training of extreme long sequence transformer models. *arXiv preprint arXiv:2309.14509*, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023.

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

Sizhe Lester Li, Evan Kim, Xingjian Bai, Tong Zhao, Tao Pang, Max Simchowitz, and Vincent Sitzmann. Turning video models into generalist robot policies. *arXiv preprint arXiv:2605.27817*, 2026b.

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

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

Jun Liu, Zhenglun Kong, Peiyan Dong, Changdi Yang, Tianqin Li, Yanyue Xie, Yifan Gong, Xuan Shen, Pu Zhao, Hao Tang, Geng Yuan, Wei Niu, Wenbin Zhang, Xue Lin, Dong Huang, and Yanzhi Wang. Structured agent distillation for large language model agents. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 3676–3685, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

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

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, Lingjie Chen, Yanjun Zhao, Ke Yang, Bingxuan Li, Cheng Qian, Gaotang Li, Xiao Lin, Zhichen Zeng, Ruizhong Qiu, Sirui Chen, Yifan Sun, Xiyuan Yang, Ruida Wang, Rui Pan, Chenyuan Yang, Dylan Zhang, Liri Fang, Zikun Cui, Yang Cao, Pan Chen, Dorothy Sun, Ren Chen, Mahesh Srinivasan, Nipun Mathur, Yinglong Xia, Hong Li, Hong Yan, Pan Lu, Lingming Zhang, Tong Zhang, Hanghang Tong, and Jingrui He. Code as agent harness. *arXiv preprint arXiv:2605.18747*, 2026.

OpenAI. Introducing Codex. 2025. https://openai.com/index/introducing-codex/.

Xuchen Pan, Yanxi Chen, Yushuo Chen, Yuchang Sun, Daoyuan Chen, Wenhao Zhang, Yuexiang Xie, Yilun Huang, Yilei Zhang, Dawei Gao, Weijie Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. Trinity-RFT: A general-purpose and unified framework for reinforcement fine-tuning of large language models. *arXiv preprint arXiv:2505.17826*, 2025.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pp. 2778–2787, 2017.

Brahma S. Pavse, Faraz Torabi, Josiah Hanna, Garrett Warnell, and Peter Stone. RIDM: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *IEEE Robotics and Automation Letters*, 5(4):6262–6269, 2020.

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

Miao Rang, Zhenni Bi, Hang Zhou, Kai Han, Xuechun Wang, An Xiao, Xinghao Chen, Yunhe Wang, and Hanting Chen. Near-policy: Accelerating on-policy distillation via asynchronous generation and selective packing. *arXiv preprint arXiv:2605.05940*, 2026.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the 20th European Conference on Computer Systems (EuroSys)*, pp. 1279–1297, 2025.

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

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

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4950–4957, 2018.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. In *Proceedings of the 3rd Conference on Language Modeling (COLM)*, 2026.

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

Yecheng Wu, Song Han, and Hai Cai. Lightning OPD: Efficient post-training for large reasoning models with offline on-policy distillation. *arXiv preprint arXiv:2604.13010*, 2026.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *International Conference on Learning Representations (ICLR)*, 2026.

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)*, volume 35, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, Qiaobo Chen, Yinyuting Yin, Hao Zhang, Tengfei Shi, Liang Wang, Qiang Fu, Wei Yang, and Lanxiao Huang. Mastering complex control in MOBA games with deep reinforcement learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 6672–6679, 2020.

Weichen Yu, Xiaomin Li, Yizhou Zhao, Xiaoze Liu, Ruowang Zhang, Haixin Wang, Yinyi Luo, Chen Henry Wu, Gaurav Mittal, Matt Fredrikson, and Yu Hu. Multi-rollout on-policy distillation via peer successes and failures. *arXiv preprint arXiv:2605.12652*, 2026.

Yuhang Zhou, Lizhu Zhang, Yifan Wu, Mingyi Wang, Bo Peng, Jiayi Liu, Xiangjun Fan, and Zhuokai Zhao. SAGE-OPD: Selective agent-guided intervention for multi-turn on-policy distillation. *arXiv preprint arXiv:2606.19659*, 2026a.

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

A Theoretical Analysis

A.1 Proof of Proposition 1

Proof. We measure time from the availability of the initial context $\boldsymbol{c}_1$. Throughout the proof, $T \geq 1$ and $0 < \ell_{\text{fast}} \leq \ell_{\text{full}}$. Under the assumptions of Proposition 1, environment transitions and other non-generation operations take negligible time, and overlapping requests do not increase their generation latencies.

In standard think-then-act interaction, the next context becomes available only after the current full response has been generated and its action executed. Since each response takes ℓ_{full}, completing T turns requires

$$C_{\text{Vanilla}} = T\ell_{\text{full}}. \tag{9}$$

For ActFirst-OPD, let s_t denote the time at which context $\boldsymbol{c}_t$ becomes available. Both the fast-action request and the full-response request start at s_t. Only the fast action is required to obtain the next context, giving

$$s_1 = 0, \qquad s_{t+1} = s_t + \ell_{\text{fast}} \quad (1 \leq t < T). \tag{10}$$

It follows that

$$s_t = (t-1)\ell_{\text{fast}}. \tag{11}$$

Environment interaction finishes when the final fast action is executed, at time $T\ell_{\text{fast}}$. The full response for turn t finishes at $s_t + \ell_{\text{full}}$. Thus, completing both interaction and all full responses requires

$$\begin{aligned} C^{\text{ideal}}_{\text{ActFirst}} &= \max\left\{T\ell_{\text{fast}}, \max_{1\leq t\leq T}\left(s_t + \ell_{\text{full}}\right)\right\} \\ &= \max\left\{T\ell_{\text{fast}}, (T-1)\ell_{\text{fast}} + \ell_{\text{full}}\right\} \\ &= (T-1)\ell_{\text{fast}} + \ell_{\text{full}}, \end{aligned} \tag{12}$$

where the last equality follows from $\ell_{\text{full}} \geq \ell_{\text{fast}}$.

Taking the ratio of Eqs. 9 and 12 yields

$$S_{\text{ideal}}(T) = \frac{T\ell_{\text{full}}}{(T-1)\ell_{\text{fast}} + \ell_{\text{full}}}. \tag{13}$$

Since $T \geq 1$, the denominator satisfies

$$(T-1)\ell_{\text{fast}} + \ell_{\text{full}} \geq \ell_{\text{full}}, \tag{14}$$

which gives

$$S_{\text{ideal}}(T) \leq \frac{T\ell_{\text{full}}}{\ell_{\text{full}}} = T. \tag{15}$$

Moreover, the assumption $\ell_{\text{full}} \geq \ell_{\text{fast}}$ implies

$$(T-1)\ell_{\text{fast}} + \ell_{\text{full}} \geq (T-1)\ell_{\text{fast}} + \ell_{\text{fast}} = T\ell_{\text{fast}}. \tag{16}$$

Therefore,

$$S_{\text{ideal}}(T) \leq \frac{T\ell_{\text{full}}}{T\ell_{\text{fast}}} = \frac{\ell_{\text{full}}}{\ell_{\text{fast}}}. \tag{17}$$

The denominator in Eq. 13 is also at most $T\ell_{\text{full}}$, so $S_{\text{ideal}}(T) \geq 1$. Combining this lower bound with the upper bounds in Eqs. 15 and 17 gives the following range for the idealized rollout speedup:

$$1 \leq S_{\text{ideal}}(T) \leq \min\left\{T, \frac{\ell_{\text{full}}}{\ell_{\text{fast}}}\right\}. \tag{18}$$

Finally, rewriting Eq. 13 gives

$$S_{\text{ideal}}(T) = \left(\frac{\ell_{\text{fast}}}{\ell_{\text{full}}} + \frac{1 - \ell_{\text{fast}}/\ell_{\text{full}}}{T}\right)^{-1}. \tag{19}$$

Since $0 < \ell_{\text{fast}}/\ell_{\text{full}} \leq 1$, the expression inside parentheses is nonincreasing in T. Hence, $S_{\text{ideal}}(T)$ is nondecreasing in T, with

$$\lim_{T\to\infty} S_{\text{ideal}}(T) = \frac{\ell_{\text{full}}}{\ell_{\text{fast}}}. \tag{20}$$

□

A.2 FINITE-CONCURRENCY CONSTRAINTS

We analyze how serving concurrency, full-response length, and rollout horizon constrain realized rollout speedup under fixed generation workloads. Consider N tasks, each containing T turns, served by a fixed pool of inference resources. Let $b_{\max}$ denote the cap on concurrent generation requests, shared by fast-action and full-response requests across this pool. All initial contexts are available at time zero, and environment transitions and other non-generation operations take negligible time. At each turn, Vanilla OPD generates one full response before acting. ActFirst-OPD generates one fast action for environment interaction and asynchronously generates one full response from the interaction context.

Each admitted request occupies one concurrency slot until completion. We assume that $0 < \ell_{\text{fast}} \leq \ell_{\text{full}}$ lower-bound the corresponding times spent holding a slot, including prefill and decoding but excluding waiting for admission. Both methods use the same resources and concurrency cap. All completion times below correspond to the given $b_{\max}$; this dependence is suppressed in the notation.

Dependency constraint. Let $s_{i,t}$ denote the availability time of the context for task i at turn t, with $s_{i,1} = 0$. The next context requires completion of the current fast action, so

$$s_{i,t+1} \geq s_{i,t} + \ell_{\text{fast}}, \qquad s_{i,t} \geq (t-1)\ell_{\text{fast}}. \tag{21}$$

The final full response cannot finish before $s_{i,T} + \ell_{\text{full}}$. Thus, the batch completion time, measured until both environment interaction and all full responses have finished, satisfies

$$C_{\text{ActFirst}} \geq (T-1)\ell_{\text{fast}} + \ell_{\text{full}}. \tag{22}$$

Concurrency constraint. Let $n(u)$ denote the number of admitted, unfinished requests at time u, so $n(u) \leq b_{\max}$. ActFirst-OPD generates NT fast actions and NT full responses. Their total time occupying concurrency slots therefore satisfies

$$NT(\ell_{\text{fast}} + \ell_{\text{full}}) \leq \int_0^{C_{\text{ActFirst}}} n(u)\, du \leq b_{\max} C_{\text{ActFirst}}. \tag{23}$$

Consequently,

$$C_{\text{ActFirst}} \geq \frac{NT(\ell_{\text{fast}} + \ell_{\text{full}})}{b_{\max}}. \tag{24}$$

Combining Eqs. 22 and 24 gives

$$C_{\text{ActFirst}} \geq \max\left\{(T-1)\ell_{\text{fast}} + \ell_{\text{full}}, \frac{NT(\ell_{\text{fast}} + \ell_{\text{full}})}{b_{\max}}\right\}. \tag{25}$$

Implications for speedup. Let C_{Vanilla} denote the completion time of the corresponding Vanilla OPD rollout batch. The realized rollout speedup satisfies

$$S_{\text{real}} = \frac{C_{\text{Vanilla}}}{C_{\text{ActFirst}}} \leq \min\left\{\frac{C_{\text{Vanilla}}}{(T-1)\ell_{\text{fast}} + \ell_{\text{full}}}, \frac{b_{\max} C_{\text{Vanilla}}}{NT(\ell_{\text{fast}} + \ell_{\text{full}})}\right\}. \tag{26}$$

This bound retains the actual baseline completion time under the same concurrency cap.

When Vanilla OPD attains the ideal completion time $C_{\text{Vanilla}} = T\ell_{\text{full}}$, Eq. 26 simplifies to

$$S_{\text{real}} \leq \min\left\{S_{\text{ideal}}(T), \frac{b_{\max}}{N(1 + \ell_{\text{fast}}/\ell_{\text{full}})}\right\}. \tag{27}$$

Under this baseline assumption, a necessary condition for $S_{\text{real}} > 1$ is

$$b_{\max} > N\left(1 + \frac{\ell_{\text{fast}}}{\ell_{\text{full}}}\right). \tag{28}$$

This condition is not sufficient for speedup. The first term in Eq. 27 captures the dependence on rollout horizon T, while the second limits the gain under a given concurrency cap $b_{\max}$. Both depend on the latency ratio $\ell_{\text{fast}}/\ell_{\text{full}}$. Under this baseline assumption, limited concurrency can prevent asynchronous overlap from offsetting the additional fast-action generation cost. Increasing $b_{\max}$ relaxes this constraint, but does not guarantee speedup.

Full-response length affects these bounds through ℓ_{full}; no proportionality between token counts and generation latencies is assumed. These bounds motivate the controlled analysis in Section 5.4, which varies $b_{\max}$, full-response length, and rollout horizon T, with $N = 16$ tasks per batch.

B Implementation Details

This appendix details prompt construction, history management, thinking-budget decoding, OPD optimization, and benchmark-specific transition consistency checks. Agent inputs are constructed from the task instruction q, history h_t, and current observation o_t, using benchmark-specific templates for full-response generation, reference-conditioned inverse dynamics, and autonomous next-action prediction. Reference next observations are included only in inverse-dynamics requests.

B.1 Prompt Details

For each benchmark, we show the user templates with history for three generation modes: full-response generation, next-action prediction, and reference-conditioned inverse dynamics. Each template includes the task instruction, retained observation–action history, current observation, and benchmark-specific admissible action information. Only the inverse-dynamics template includes a reference next observation.

ALFWorld.

ALFWorld: Full-response generation

```
You are an expert agent operating in the ALFRED Embodied Environment.
 Your task is to: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}
Your admissible actions of the current situation are: [{
admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation.
This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action> tags
.
```

ALFWorld: Next-action prediction

```
You are an expert agent operating in the ALFRED Embodied Environment.
 Your task is to: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}
Your admissible actions of the current situation are: [{
admissible_actions}].

Now it's your turn to take an action.
You should choose an admissible action for current step and present
it within <action> </action> tags.
```

ALFWorld: Reference-conditioned inverse dynamics

```
You are an expert agent operating in the ALFRED Embodied Environment.
 Your task is to: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
```

ALFWorld: Reference-conditioned inverse dynamics (continued)

```
You are now at step {current_step} and your current observation is: {
current_observation}

Complete the missing current action in this reference transition:
Current observation --[MISSING CURRENT ACTION]--> reference next
observation.

The reference next observation is the environment state immediately
after the missing current action. It may be a terminal state.
Reference next observation:
{reference_next_observation}

The reference next observation is the direct result of exactly one
action. Do not insert an intermediate or preparatory action.

Your admissible actions of the current situation are: [{
admissible_actions}].

Now it's your turn to take one action for the current step. Predict
the MISSING CURRENT ACTION that leads to the reference next
observation.
You should choose one admissible action and present it within <action
> </action> tags.
```

WebShop.

WebShop: Full-response generation

```
You are an expert autonomous agent operating in the WebShop e-
commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation,
then think carefully which admissible action best advances the
shopping goal. This reasoning process MUST be enclosed within <think>
 </think> tags.
Once you've finished your reasoning, you should choose an admissible
action for current step and present it within <action> </action> tags
.
```

WebShop: Next-action prediction

```
You are an expert autonomous agent operating in the WebShop e-
commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s).
Below are the recent {history_length} observations and the
corresponding actions you took:
{action_history}
```

WebShop: Next-action prediction (continued)

```
You are now at step {current_step} and your current observation is:
{current_observation}

Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should choose an admissible action for current step and present
it within <action> </action> tags.
```

WebShop: Reference-conditioned inverse dynamics

```
You are an expert autonomous agent operating in the WebShop e-
commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s).
Below are the recent {history_length} observations and the
corresponding actions you took:
{action_history}

You are now at step {current_step} and your current observation is:
{current_observation}

Complete the missing current action in this reference transition:
Current observation --[MISSING CURRENT ACTION]--> reference next
observation.

The reference next observation is the environment state immediately
after the missing current action. It may be a terminal Score page.
Reference next observation:
{reference_next_observation}

If the missing current action is search[...], reconstruct the query
most likely to produce the reference next observation. Do not add
price or budget terms to the query.

Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step. Predict
the MISSING CURRENT ACTION that leads to the reference next
observation.
You should choose one admissible action and present it within <action
> </action> tags.
```

ScienceWorld.

ScienceWorld: Full-response generation

```
Your ScienceWorld task is: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}
Available action commands: [{action_templates}]
Available objects you can interact with: [{objects}]
```

ScienceWorld: Full-response generation (continued)

```
Now it's your turn to take an action. Combine an action command with
appropriate object(s) to form a valid action.
You should first reason step-by-step about the current situation.
This reasoning process MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose a valid action
 for the current step and present it within <action> </action> tags.
```

ScienceWorld: Next-action prediction

```
Your ScienceWorld task is: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}
Available action commands: [{action_templates}]
Available objects you can interact with: [{objects}]

Now it's your turn to take an action. Combine an action command with
appropriate object(s) to form a valid action.
You should choose a valid action for the current step and present it
within <action> </action> tags.
```

ScienceWorld: Reference-conditioned inverse dynamics

```
Your ScienceWorld task is: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {
current_observation}

Complete the missing current action in this reference transition:
Current observation --[MISSING CURRENT ACTION]--> reference next
observation.

The reference next observation is the environment state immediately
after the missing current action. It may be a terminal state.
Reference next observation:
{reference_next_observation}

Available action commands: [{action_templates}]
Available objects you can interact with: [{objects}]

Now it's your turn to take one action for the current step. Predict
the MISSING CURRENT ACTION that leads to the reference next
observation.
Combine an action command with appropriate object(s), then present
exactly one action within <action> </action> tags.
```

B.2 History Management

We store the full interaction history h_t as chronological observation–action pairs. During ActFirst-OPD rollout, these pairs are $(o_i, \tilde{a}_i)$. Past reasoning z_i and unexecuted actions a_i from asynchronous full responses are excluded. The task instruction q and current observation o_t enter the schematic input $\boldsymbol{c}_t = q \oplus h_t \oplus o_t$ separately.

Each generation mode renders its own request from this interaction record, with o_{t+1}^{ref} added only for inverse dynamics. Each full response $y_t = (z_t, a_t)$ is generated asynchronously from a snapshot of

the interaction context $\boldsymbol{c}_t$; subsequent history updates do not change that request. Different templates can retain different history suffixes because their token lengths differ.

For each request, we begin with a copy of h_t and count tokens after applying the chat template. If the prompt exceeds its budget, we repeatedly remove the oldest complete observation–action pair and render the prompt again, until it fits or no history remains. The corresponding no-history template is used when the retained history is empty. This selection leaves the stored h_t unchanged and does not shorten the current observation, benchmark-specific admissible action information, or reference next observation. We use no learned summaries or semantic history compression.

B.3 Thinking Budget

We use budgeted decoding for full think-then-act responses $y_t = (z_t, a_t)$ generated by $\pi_{\boldsymbol{\theta}}$ from the interaction context $\boldsymbol{c}_t$ during both training and evaluation. The same procedure applies to asynchronous full-response generation for distillation in ActFirst-OPD. Following the Qwen3 thinking-budget approach,[1] we use at most two generation requests to leave room for action generation when reasoning is long. Fast actions $\tilde{a}_t$ follow a separate action-only decoding path.

If the first request finishes normally, we use its output directly. If it reaches its token budget before action generation starts, we insert an early-stop instruction adapted to the action format:

Inserted thinking-budget continuation

```
Considering the limited time by the user, I have to give the action
based on the thinking directly now.
</think>

<action>
```

If reasoning has already ended, the inserted text omits `</think>`; if action generation has started, no text is inserted. For an unfinished response with remaining budget, a second request continues from $\boldsymbol{c}_t$, the first-stage output, and any inserted tokens. Across all three benchmarks, the first-stage and total response limits are 384 and 512 tokens during training, and 1,920 and 2,048 during evaluation. The second request generates at most 128 tokens, subject to the remaining total budget after counting both sampled and inserted tokens. The first-stage limit bounds generated tokens rather than fixing the length of z_t, since reasoning may end before this limit.

Actions are extracted from the `<action>` field. In think-then-act interaction, responses marked as truncated are treated as empty actions. The budgeted decoder does not mark a response as truncated solely because it reaches the length limit after producing `</action>`. During ActFirst-OPD training, the asynchronously generated full response y_t is used for distillation, while its action a_t is not executed; the environment advances through $\tilde{a}_t$. At evaluation time, the student follows standard think-then-act interaction and submits the extracted action a_t to the environment.

For OPD training, manually inserted tokens remain in the shared student–teacher context but are excluded from supervision and loss normalization. The corresponding masks and optimization details are given in Appendix B.4.

B.4 OPD Optimization

Following prior multi-turn OPD studies (Wang et al., 2026), we optimize a sampled policy surrogate of the reverse-KL objective in Eq. 4. An optimization batch $\mathcal{B}$ contains turn-level full responses $y_b = (z_b, a_b)$ indexed by b, with token contexts $\boldsymbol{c}_{b,j}$ defined as in Section 3. Let $\boldsymbol{\theta}_{\text{old},b}$ denote the student parameters used to generate response b. Turn-level full responses from completed rollout batches are stored in a first-in, first-out (FIFO) buffer. Each update consumes responses generated by the current student version or the immediately preceding version, discarding older samples during batch construction. Remaining responses stay buffered for subsequent updates, subject to the same version constraint; consumed responses are not reused. The frozen teacher scores the generated tokens under the same contexts.

[1] https://qwen.readthedocs.io/en/stable/getting_started/quickstart.html#thinking-budget

Let $m_{b,j} = 1$ for a model-generated response token and 0 for manually inserted tokens or padding. Both generated reasoning and action tokens are supervised, including model-generated formatting tokens. The inserted tokens described in Appendix B.3 remain in the student–teacher context but contribute neither to the loss nor to its normalization. At supervised positions, the token advantage is

$$A_{b,j} = \mathrm{sg}\left[\log \frac{\pi_{\boldsymbol{\phi}}(x_{b,j} \mid \boldsymbol{c}_{b,j})}{\pi_{\boldsymbol{\theta}_{\mathrm{old},b}}(x_{b,j} \mid \boldsymbol{c}_{b,j})}\right], \tag{29}$$

where sg denotes stop-gradient; masked positions have zero advantage. The distillation coefficient is set to one. Token advantages are computed without environment rewards or advantage normalization.

Using stored sampling log-probabilities for the behavior-policy denominator, we form the numerically bounded ratio

$$\rho_{b,j}(\boldsymbol{\theta}) = \exp\left[\mathrm{clip}\left(\log \frac{\pi_{\boldsymbol{\theta}}(x_{b,j} \mid \boldsymbol{c}_{b,j})}{\pi_{\boldsymbol{\theta}_{\mathrm{old},b}}(x_{b,j} \mid \boldsymbol{c}_{b,j})}, -20, 20\right)\right]. \tag{30}$$

The log-probability ratio is clipped to $[-20, 20]$ for numerical stability before exponentiation. We use the proximal policy optimization (PPO) surrogate (Schulman et al., 2017) with dual clipping (Ye et al., 2020). With clipping width $\delta = 0.2$ and dual-clipping constant $\kappa = 3$, the minimized token loss is

$$\begin{aligned} \ell_{\mathrm{clip}}(\rho, A) &= -\min\{\rho A,\ \mathrm{clip}(\rho, 1-\delta, 1+\delta)A\}, \\ \ell_{\mathrm{DC}}(\rho, A) &= \begin{cases} \min\{\ell_{\mathrm{clip}}(\rho, A), -\kappa A\}, & A < 0, \\ \ell_{\mathrm{clip}}(\rho, A), & A \geq 0. \end{cases} \end{aligned} \tag{31}$$

Here, ℓ_{DC} denotes the dual-clipped PPO token loss.

We aggregate this loss using `seq-mean-token-mean`:

$$\mathcal{L}_{\mathrm{sur}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{\sum_{j=1}^{L_b} m_{b,j}\, \ell_{\mathrm{DC}}(\rho_{b,j}(\boldsymbol{\theta}), A_{b,j})}{\sum_{j=1}^{L_b} m_{b,j} + \epsilon}, \tag{32}$$

where L_b is the response length and $\epsilon = 10^{-8}$. Thus, losses are averaged over supervised tokens within each full response, and the resulting response losses are averaged across the batch.

All three generation modes share the student parameters $\boldsymbol{\theta}$, which are updated by backpropagating through $\mathcal{L}_{\mathrm{sur}}$. Sampling log-probabilities, teacher scores, and advantages remain fixed during each update, and the teacher parameters $\boldsymbol{\phi}$ remain frozen throughout training. The main method adds no entropy penalty, separate reference-policy KL loss, or auxiliary loss on the fast actions $\tilde{a}_t$.

B.5 Transition Consistency Checks

The checks in Section 4.2 compare the student's actual observations and available action information with their offline reference counterparts. They use observable representations rather than simulator-state equality. Retaining reference guidance requires a valid action, agreement of the environment's termination flag with the reference, and the benchmark-specific conditions below. The generated action need not equal the reference action.

We normalize observation text before comparison by applying Unicode NFKC normalization and case folding, standardizing line endings, collapsing whitespace within each line, and removing empty lines. The remaining line order is preserved except in ScienceWorld, as specified below. Normalized observations are compared for exact equality, except for the WebShop search-result rule described below.

ALFWorld. An action is valid if its normalized text belongs to the current admissible-action set, excluding `help`. We require the normalized next observation o_{t+1} to equal o_{t+1}^{ref}. For nonterminal transitions, the next admissible-action sets must also match after text normalization and deduplication; this comparison is omitted for terminal transitions.

Table 4: **Benchmark configurations.** Counts refer to task instances; the WebShop version denotes its base repository commit.

Benchmark	Version	Training	Evaluation	Max. turns
ALFWorld	0.4.2	3,553	140 seen / 134 unseen	30
WebShop	`64fa2a5`	4,096	100	15
ScienceWorld	1.2.3	2,294	150	30

WebShop. A parsed `search[query]` action is valid when its query is nonempty and a search bar is available; a parsed `click[target]` action is valid when its lowercased target appears among the current clickable elements. Observation comparison additionally removes click-notification lines of the form `You have clicked ...`. We compare normalized next observations and, for nonterminal transitions, the available-action lists after case folding, whitespace normalization, and sorting. The repeated instruction block is retained in these comparisons; its removal applies only to the reference observation shown in the inverse-dynamics prompt. We permit different nonterminal search results when both the student and reference actions are searches and the product identifier in the reference's immediately following click action remains clickable. The search-result mismatch is also allowed at the next turn, provided that the reference product remains clickable; the subsequent click transition must satisfy the ordinary comparison rules. Terminal transitions require matching normalized observations but no available-action comparison.

ScienceWorld. An action is valid if its normalized text appears in the current admissible-action list. We compare normalized next observations and action-template sets, including for terminal transitions. Observation normalization additionally sorts lines and object enumerations inside non-nested `(containing ...)` expressions and after the last `is:` on each line. Enumeration parsing ignores commas inside parentheses and retains clauses starting with `which` or `that` with their preceding item. Action templates use the same text normalization, with duplicates removed and order ignored. Possible-object aliases are excluded from matching because their enumeration can vary across resets. The full admissible-action list is used for action validation, not set equality; separate room descriptions, inventory, reward, and score are not compared.

Across all three benchmarks, a failed check permanently disables reference conditioning for the remainder of the rollout, without resetting the environment. An invalid reference-conditioned action proposal triggers one autonomous next-action prediction request from the unchanged interaction context before environment execution. Reference actions are used internally for the stated checks but are not supplied to the student's prompt.

C Experiment Details

C.1 Benchmarks

We use text observations and actions across all three benchmarks. Table 4 summarizes environment versions, task counts, and interaction limits. Training and evaluation task instances are disjoint.

ALFWorld. ALFWorld (Shridhar et al., 2021)[2] contains six household task types, reported in Table 1 as Pick (pick and place), Look (examine an object under a light), Clean (clean and place), Heat (heat and place), Cool (cool and place), and Pick2 (pick two objects and place them). The prompt lists the current admissible actions, excluding `help`, with object and receptacle arguments already specified, and instructs the student to select one. We use the `json_2.1.1` game files with TextWorld (Côté et al., 2019) 1.7.0, training on `train` and evaluating separately on `valid_seen` and `valid_unseen`. The seen split uses rooms encountered during training with new object configurations; the unseen split uses held-out rooms with different layouts and receptacles. Both splits cover the same six task types.

[2] `https://github.com/alfworld/alfworld`

WebShop. WebShop (Yao et al., 2022)[3] requires agents to search for and purchase products satisfying language instructions. The prompt lists the current `click[target]` actions and, when search is available, a `search[query]` template whose query is generated by the student. Following prior multi-turn OPD studies (Wang et al., 2026), we use its text interface with the full product catalog and a custom split: goal IDs 0–4095 for training and 4096–4195 for evaluation. The goal-construction seed is fixed at 233.

ScienceWorld. ScienceWorld (Wang et al., 2022)[4] provides interactive science tasks covering physical, chemical, and biological processes. To limit prompt length, we provide action templates and possible object referents separately for the student to construct commands. The environment's admissible-action list is used for action validation. We use the `easy` environment preset and a custom split: training uses the first half of the variations from 17 task types, while evaluation uses the last five variations from each of all 30 task types. The evaluation set therefore contains 85 instances from task types covered during training and 65 from the remaining 13 types.

Evaluation Metrics. For N evaluation tasks, let $u_i \in \{0, 1\}$ indicate task success, $\text{score}_i \in [0, 100]$ denote the task score, and T_i denote the number of interaction turns used. We report

$$\text{SR} = \frac{100}{N} \sum_{i=1}^{N} u_i, \qquad \text{Score} = \frac{1}{N} \sum_{i=1}^{N} \text{score}_i, \qquad \text{Round} = \overline{T} = \frac{1}{N} \sum_{i=1}^{N} T_i. \tag{33}$$

SR is expressed as a percentage; Score is reported only for WebShop and ScienceWorld to capture partial task completion.

ALFWorld success requires satisfying the task goal within the turn limit; task-type and overall results pool the seen and unseen tasks. WebShop assigns partial credit for matching the requested product attributes, options, and price, weighted by product-type match. Its task score is the terminal purchase reward scaled to $[0, 100]$, with zero assigned if no purchase is made. ScienceWorld assigns partial credit for completing task subgoals; we use the highest environment score reached during the episode, clipped to $[0, 100]$. Success on WebShop and ScienceWorld requires a full score, implemented as $\text{score}_i \geq 99.999$.

Round averages interaction turns over both successful and failed episodes, including invalid action attempts. On ALFWorld, episodes end upon task success or at the 30-turn limit, so failed episodes contribute 30 turns. On WebShop and ScienceWorld, unsuccessful purchases or terminal failures can end an episode early; fewer turns therefore do not necessarily indicate better task performance.

C.2 Teacher Training and Reference Construction

Teacher Training. Each benchmark uses a task-specialized Qwen3-8B teacher, shared across student sizes and frozen throughout OPD. The ALFWorld teacher is trained by SFT on successful trajectories from the GiGPO-Qwen2.5-7B model, followed by GiGPO (Feng et al., 2025). The WebShop teacher uses score-filtered SFT on trajectories from the corresponding GiGPO-Qwen2.5-7B model, followed by GiGPO (Feng et al., 2025). The ScienceWorld teacher is trained directly with GiGPO.

Reference Trajectories. To obtain high-quality references, we select trajectories based on task success or score, validate them through environment replay, and favor shorter action sequences. For ALFWorld, candidates include public demonstrations from Hugging Face, hand-coded solutions, and ten rollouts per training task from each of GiGPO-Qwen2.5-7B and our Qwen3-8B teacher. We select the shortest candidate marked successful and validate it. For WebShop, we combine public demonstrations from Hugging Face, ten teacher rollouts per training task, and oracle solutions constructed from training-task goals. Candidates are matched to the corresponding training-task instructions and replayed; selection prioritizes higher score and then fewer actions. For ScienceWorld, we combine ten teacher rollouts per training task, simulator-generated gold paths, and task-specific scripted solutions. Candidates are shortened while preserving successful replay. We replay selected action sequences offline under the training environment configuration and cache the initial

[3] `https://github.com/princeton-nlp/WebShop`

[4] `https://github.com/allenai/ScienceWorld`

Table 5: **Offline preparation costs.** Times are in hours, with eight GPUs per job. Teacher training time includes all listed stages. Ten candidates are generated per task for each source model.

	Teacher training			Candidate collection	
Benchmark	Time (h)	SFT steps	GiGPO steps	Time (h)	Candidates
ALFWorld	4.18	250	250	8.50	71,060
WebShop	22.05	250	2,200	4.81	40,960
ScienceWorld	18.27	—	144	19.88	22,940

and subsequent observations. The cached next observations serve as local transition targets o_{t+1}^{ref} for reference-conditioned inverse dynamics. The resulting caches cover all training tasks across the three benchmarks (100% coverage), with one selected trajectory per task.

Offline Cost. Teacher training and reference construction are completed before student training and excluded from the reported training wall-clock time. For a fixed task set and environment, the resulting teachers and reference caches can be reused across student sizes and repeated training runs, amortizing their one-time preparation costs. Table 5 reports teacher training and candidate rollout collection separately.

C.3 Training Protocols and Hyperparameters

Table 6 summarizes student training configurations across benchmarks. Within each benchmark–model setting, methods share the initial Qwen3 checkpoint and frozen teacher. In the training pipeline, rollout collection and student optimization proceed asynchronously. The OPD loss, token masks, and sample staleness control are detailed in Appendix B.4.

Training Time. Training wall-clock time in Tables 1 and 2 spans training-process launch to exit, including initialization, rollout collection, teacher scoring, student updates, checkpoint saving, and shutdown. Offline teacher training, reference construction, and subsequent evaluation are excluded.

C.4 Evaluation Protocols and Hyperparameters

All methods use standard think-then-act interaction without reference information, with the settings in Table 7. Task splits and metric definitions are provided in Appendix C.1; budgeted decoding and action extraction follow Appendix B.3.

For a fixed checkpoint, let M_r denote a task-performance metric evaluated with seed $r \in \mathcal{S} = \{42, 43, 44\}$. We report the mean and sample standard deviation (Tables 1 and 2; Figure 5):

$$\overline{M} = \frac{1}{3}\sum_{r\in\mathcal{S}} M_r, \qquad \mathrm{SD}(M) = \sqrt{\frac{1}{3-1}\sum_{r\in\mathcal{S}}\left(M_r - \overline{M}\right)^2}. \tag{34}$$

C.5 Rollout Efficiency and Quality Measurements

Rollout Efficiency. For the rollout efficiency experiment in Figure 3(a,b), we use the initial Qwen3-1.7B model and 1,024 initial ALFWorld contexts, sampled proportionally across the six training task types with seed 42. Full responses use the standard think-then-act prompt; fast actions use the action-only prompt without reference information. We vary the cap $b_{\max}$ on concurrent generation requests, shared by fast-action and full-response requests, and the virtual interaction horizon T defined in Section 5.4. Table 8 summarizes the settings.

Under Vanilla OPD scheduling, each task's next turn waits for its full response. Under ActFirst-OPD scheduling, fast-action and full-response requests are submitted together, and fast-action completion releases the next turn. Tasks run concurrently within each batch, which ends only after all requests finish. Timing includes queueing, prefill, decoding, and request scheduling, but excludes initialization, warmup, input tokenization, and output detokenization. Each paired repeat uses the same engine, contexts, and full-response sampling seeds, with strategy order alternated across repeats.

Table 6: **Training hyperparameters across all benchmarks.**

Hyperparameter	Value
Algorithm and Optimization	
Distillation objective	Student-to-teacher reverse KL
PPO clipping	0.2
Dual-clipping constant	3.0
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Learning rate	10^{-6}
Learning-rate schedule	Constant, no warmup
Adam coefficients (β_1, β_2)	$(0.9, 0.999)$
Weight decay	0.01
Gradient norm clipping	1.0
Sample staleness control	Version lag ≤ 1
Loss aggregation	`seq-mean-token-mean`
Training	
Training updates	250
Rollout batch	16 tasks
Rollout trajectories per task	1
Optimization batch	64 turn-level full responses
Optimization epochs per batch	1
Weight synchronization interval	Every update
Models and Generation	
Student models	Qwen3-0.6B, 1.7B, 4B
Teacher model	Qwen3-8B
Prompt limit	10,240 tokens
Full-response limit	512 tokens
Thinking budget (first stage)	384 tokens
Action continuation budget	Up to 128 tokens
Sampling temperature	1.0
Top-p	1.0
Top-k filtering	Disabled
Thinking mode	Enabled (`enable_thinking=True`)
Rollout engine seed	42
Distributed Training	
Hardware	8 A100-SXM4-80GB GPUs
Rollout GPUs	4
Teacher-scoring GPUs	2
Student parameter optimization GPUs	2
Rollout engines per GPU	1
Runners per rollout engine	4
Total rollout runners	16
Inference tensor parallelism	1
Training sequence parallelism	2 (Ulysses (Jacobs et al., 2023))
Precision	`bfloat16`
Dynamic batching token budget	16,384 tokens per GPU
vLLM GPU memory fraction	0.7
Prefix caching	Disabled
Environment-Specific Turn Limits	
ALFWorld	30
WebShop	15
ScienceWorld	30

Let B denote a batch of 16 task indices, and let $C^{(k)}_{\text{Vanilla},B}$ and $C^{(k)}_{\text{ActFirst},B}$ denote its measured completion times in paired repeat k. The rollout speedup in each repeat is

$$S^{(k)}_{\text{real}} = \frac{\sum_B C^{(k)}_{\text{Vanilla},B}}{\sum_B C^{(k)}_{\text{ActFirst},B}}, \qquad k \in \{1, 2, 3\}, \tag{35}$$

Table 7: **Evaluation hyperparameters across all benchmarks.**

Hyperparameter	Value
Evaluation Protocol	
Interaction mode	Think-then-act
Reference information	None
Evaluation seeds	42, 43, 44
Trajectories per task and seed	1
Evaluation batch	64 tasks
Generation	
Prompt limit	20,480 tokens
Full-response limit	2,048 tokens
Thinking budget (first stage)	1,920 tokens
Action continuation budget	Up to 128 tokens
Sampling temperature	0.4
Top-p	1.0
Top-k filtering	Disabled
Thinking mode	Enabled (`enable_thinking=True`)
Distributed Inference	
Hardware	8 A100-SXM4-80GB GPUs
Inference engines per GPU	1
Inference tensor parallelism	1
Precision	`bfloat16`
vLLM GPU memory fraction	0.82
Prefix caching	Disabled
Chunked prefill	Enabled
Benchmark-Specific Runners per Engine	
ALFWorld	8
WebShop	4
ScienceWorld	4
Environment-Specific Turn Limits	
ALFWorld	30
WebShop	15
ScienceWorld	30

where the sums cover all 64 batches. We report the mean and sample standard deviation of these three ratios.

Idealized speedup estimates use independent single-request measurements, including prefill and decoding. For the context of task i, let $\widehat{\ell}_{\mathrm{fast},i}$ and $\widehat{\ell}_{\mathrm{full},i}$ denote the median fast-action and full-response latencies over three measurements. Treating these calibrated latencies as constant across T virtual turns, the estimated batch completion times under unrestricted overlap are

$$\begin{aligned} \widehat{C}_{\mathrm{Vanilla},B} &= \max_{i\in B} T\widehat{\ell}_{\mathrm{full},i}, \\ \widehat{C}_{\mathrm{ActFirst},B} &= \max_{i\in B}\left[(T-1)\widehat{\ell}_{\mathrm{fast},i} + \max\left\{\widehat{\ell}_{\mathrm{fast},i}, \widehat{\ell}_{\mathrm{full},i}\right\}\right], \\ \widehat{S}_{\mathrm{ideal}}(T) &= \frac{\sum_B \widehat{C}_{\mathrm{Vanilla},B}}{\sum_B \widehat{C}_{\mathrm{ActFirst},B}}. \end{aligned} \tag{36}$$

The maximum over tasks reflects that a batch waits for its slowest task. The inner maximum for ActFirst-OPD accounts for both requests at the final turn. When $\widehat{\ell}_{\mathrm{fast},i} \le \widehat{\ell}_{\mathrm{full},i}$, its per-task estimate reduces to the expression in Proposition 1. These calibrated estimates extend the idealized analysis to task batches and are not hardware upper bounds.

Environment-Transition Throughput. For Qwen3-1.7B on ALFWorld, both methods use 16 tasks per rollout batch, four rollout engines, and four runners per engine. Let N_{env} count completed

Table 8: **Rollout efficiency experiment settings.**

Parameter	Value
Workload	
Tasks per batch	16
Fast-action output length	16 tokens
Full-response output length	128 or 512 tokens
Sampling temperature	1.0
Top-p	1.0
Top-k filtering	Disabled
Inference	
Hardware per worker	1 A100-SXM4-80GB GPU
Tensor parallelism	1
Precision	`bfloat16`
vLLM GPU memory fraction	0.85
Maximum tokens per scheduling step	16,384
Prefix caching	Disabled
Chunked prefill	Enabled
CUDA graphs	Enabled
Scans and Repeats	
Concurrency scan: $b_{\max}$	$\{16, 32, 64, 128, 256\}$
Fixed horizon for concurrency scan	$T = 30$
Horizon scan: T	$\{1, 2, 4, 8, 16, 30\}$
Fixed concurrency for horizon scan	$b_{\max} = 256$
Paired timing repeats	3

Table 9: **Environment-transition throughput on ALFWorld with Qwen3-1.7B.** Measurements use training runs separate from those reported in Table 1. For each method, both windows use the same run and start when rollout collection begins. The full-training window ends when update 250 completes; the early window covers the first 600 seconds. Initialization and final shutdown are excluded.

Window	Method	Transitions	Window time (s)	Env. steps / s
Full training	Vanilla OPD	55,525	15,202	3.65
	ActFirst-OPD	38,616	6,796	5.68
First 10 min	Vanilla OPD	1,289	600	2.15
	ActFirst-OPD	3,803	600	6.34

environment transitions within a measurement window of Δt seconds. Throughput is

$$\text{Throughput} = \frac{N_{\text{env}}}{\Delta t}, \tag{37}$$

measured in environment transitions per second (env. steps / s).

The full-training window starts at the first training-sample request and ends when update 250 finishes; the early window covers the first 600 seconds from the same origin. Both include rollout, scoring, optimization, and waiting time, but exclude initialization and shutdown. Only transitions completed within each window are counted. Table 9 reports the measurements.

Rollout Quality. We compare think-then-act, direct action without reference next observations, and ActFirst-OPD's reference-conditioned interaction with autonomous fallback. At each of updates 1, 125, and 250, all strategies use the same frozen Qwen3-1.7B Vanilla OPD checkpoint and generate one trajectory per ALFWorld training task (3,553 tasks). Checkpoints are analyzed separately. All strategies use temperature 1.0, top-$p = 1.0$, no top-k filtering, and a 30-turn limit, with sampling seeds matched across strategies for each task and turn. Think-then-act uses the training thinking budget in Appendix B.3; all action-only requests stop at `</action>` or 512 generated tokens. ActFirst-OPD rollouts retain the training transition checks and persistent autonomous fallback, but omit asynchronous full-response generation, teacher scoring, and parameter updates.

An executed action is valid if parsing succeeds and the action, after stripping surrounding whitespace, exactly matches an entry in the admissible-action list. Repetition is determined from environment state–action pairs. For this offline measurement, state equality requires identical sets of positive environment facts and the same task-success flag. Actions are normalized for pair comparison.

A valid action counts as an unproductive repetition if the same state–action pair was previously executed validly, no new normalized observation has appeared since the end of that pair's most recent valid execution, and the current action also produces no new observation. An observation is new if its normalized text has not appeared earlier in the rollout. Text normalization uses Unicode NFKC, case folding, and whitespace normalization. The observation record includes o_1 and feedback from invalid attempts, is maintained independently of prompt-history truncation, and excludes reference observations and model outputs.

For each checkpoint–strategy pair, let $r_i \in \{0, 1\}$ indicate whether task i contains at least one repetition as defined above, $u_i \in \{0, 1\}$ indicate success (Appendix C.1), and T_i denote its executed interaction turns (Appendix C.1). For $N = 3{,}553$ tasks, the metrics in Figure 4(a–c) are computed as

$$\begin{aligned} &\text{(a) Tasks with } \geq 1 \text{ repeated action:} \quad \text{Repeat } (\%) = \frac{100}{N} \sum_{i=1}^{N} r_i, \\ &\text{(b) Successful trajectory yield:} \qquad \text{Yield} = \frac{100 \sum_{i=1}^{N} u_i}{\sum_{i=1}^{N} T_i}, \\ &\text{(c) Average rollout turns:} \qquad \overline{T} = \frac{1}{N} \sum_{i=1}^{N} T_i. \end{aligned} \tag{38}$$

Repeat is the percentage of tasks containing at least one repetition; Yield counts successful trajectories per 100 environment transitions; $\overline{T}$ is the mean number of executed turns across all tasks. Turn counts include invalid attempts and failed trajectories. All failed trajectories in this collection reach 30 turns, so $\overline{T}$ also equals completion cost with a 30-turn failure penalty.

D Extended Related Work

We extend Section 2 by comparing how related OPD and agent distillation methods collect or reuse trajectories, provide teacher supervision, and improve training efficiency. ReOPD (Liao et al., 2026) samples a single-turn student response conditioned on each selected teacher-trajectory prefix and trains the student with teacher supervision, without online environment interaction. Our Reference-Prefix Replay (100%) ablation (Table 3) similarly uses reference trajectories to construct student training contexts, but executes the reference actions in the environment. This ablation evaluates the effect of replacing student-generated actions with reference actions within our framework. It does not reproduce ReOPD's off-environment training or prefix-sampling schedule. Guided-OPD (Li et al., 2026a) mixes teacher- and student-generated turns under a curriculum that gradually removes teacher intervention. SAGE-OPD (Zhou et al., 2026a) combines turn-level intervention based on environment feedback and teacher judgments with teacher-confidence weighting of the distillation loss. Structured agent distillation (SAD) (Liu et al., 2026) segments ReAct trajectories into reasoning and action spans for separate supervision. Multi-rollout on-policy distillation (MOPD) (Yu et al., 2026) conditions the teacher on successful and failed peer rollouts to improve token-level supervision. Near-policy distillation (Rang et al., 2026) decouples student-response generation from optimization, using precomputed teacher logits and sequence packing for efficient training and sample filtering to address policy lag and sample noise. Lightning OPD (Wu et al., 2026) precomputes teacher log-probabilities on rollouts from an SFT-initialized student and reuses them during distillation, eliminating live teacher serving during optimization. These works study supervision quality and distillation efficiency from different perspectives. Our work focuses on the efficiency of online experience collection in multi-turn agent OPD, specifically reducing reasoning-induced interaction delays while maintaining rollout quality. Reference next observations guide fast student-generated actions, with autonomous next-action prediction after divergence, while the student asynchronously generates a full response from each interaction context for teacher supervision. This retains online environment interaction and full-response distillation while removing reasoning from the critical path of environment transitions.

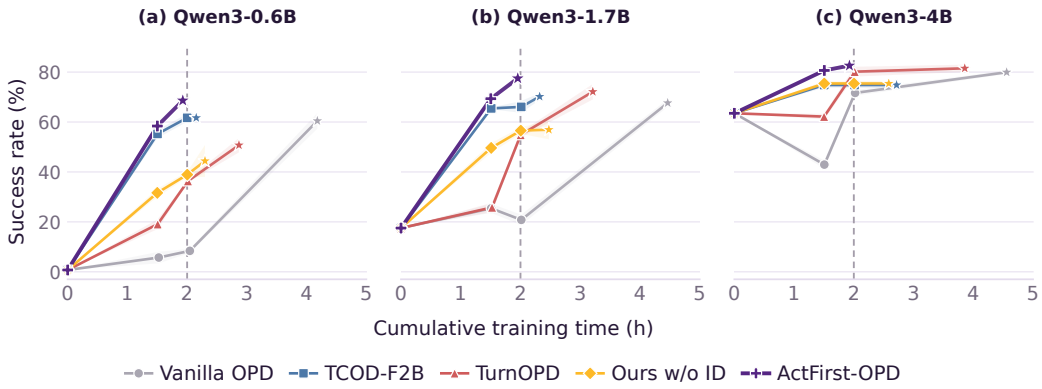


Figure 5: **Performance–time trade-off on ALFWorld for Qwen3-0.6B, 1.7B, and 4B students.** SR aggregates seen and unseen tasks; shading indicates ± 1 standard deviation across three evaluation seeds. Stars mark the final checkpoints in Table 1; the dashed line marks two hours of training.

E Additional Discussion

Scope of On-Policy Distillation. In ActFirst-OPD, we distinguish two aspects of on-policy distillation: the distribution of rollout contexts in the full interaction trajectory, and the on-policy sampling of turn-level full responses used for distillation, conditional on those contexts. The environment advances through fast actions $\tilde{a}_t$, using reference-conditioned inverse dynamics before divergence and autonomous next-action prediction afterward. At each collected interaction context $\boldsymbol{c}_t$, the student generates a full response $y_t = (z_t, a_t)$ without conditioning on the reference next observation. The frozen teacher supervises its tokens, while a_t is not executed. Thus, turn-level full responses are sampled on-policy conditional on the collected contexts, while their context distribution need not match that induced by the student's think-then-act policy.

Context Distribution and Exposure Bias. Early OPD methods, including MiniLLM (Gu et al., 2024) and GKD (Agarwal et al., 2024), address train–inference mismatch in response prefixes by incorporating student-generated sequences into distillation. This mismatch concerns prefixes within a response, conditional on a given input. In our setting, these are token prefixes $\boldsymbol{x}_{t,<j}$ within each full response y_t, distinct from the distribution of interaction contexts $\boldsymbol{c}_t$ across interaction turns. Reference-conditioned inverse dynamics may improve task progress, as suggested by the rollout-quality results in Figure 4(b,c). At the same time, reference guidance may reduce exposure to states that could arise from accumulated errors in think-then-act interaction. Autonomous next-action prediction continues interaction from actual deviations, but remains action-only and need not induce the same context distribution as think-then-act interaction at inference time. The results in Tables 1 and 2 support the practical value of this design in the evaluated settings, without establishing that exposure bias at inference time is eliminated or uniformly reduced.

F Additional Experimental Results

Performance–Time Trade-off. Figure 5 plots mean evaluation SR against cumulative training time on ALFWorld for three Qwen3 student sizes. ActFirst-OPD completes 250 updates within two hours at all three sizes. At approximately 1.5 hours, the 0.6B, 1.7B, and 4B students reach SRs of 58.39%, 69.34%, and 80.66%, respectively, compared with 5.72%, 25.43%, and 42.94% under Vanilla OPD. These comparisons show higher SR under comparable training-time budgets, complementing the fixed-update results in Table 1.

Reference Alignment During Training. We quantify reference alignment before fallback at the first-action, trajectory, and turn levels. Figures 6–8 report rates computed from counts aggregated over the latest 10 completed rollout batches at each update, using all available batches when fewer than 10 have completed. Endpoint labels show the final-window rates at update 250; Table 10 reports full-training cumulative rates.

Table 10: **Cumulative reference alignment over 250 training updates.** Cumulative rates are computed by aggregating the corresponding counts across complete rollouts in rollout batches finished by update 250. First-action and full-trajectory alignment use the number of rollout tasks with reference guidance as the denominator; turn coverage uses all executed turns, including fallback turns.

Benchmark	Student	First-action alignment (%)	Full-trajectory alignment (%)	Aligned-turn coverage (%)
ALFWorld	0.6B	84.86	32.36	18.19
	1.7B	89.38	49.75	33.59
	4B	93.39	85.63	64.05
WebShop	0.6B	49.33	11.80	18.77
	1.7B	56.89	21.24	23.50
	4B	60.85	49.87	51.77
ScienceWorld	0.6B	72.56	27.74	10.44
	1.7B	62.57	40.44	14.99
	4B	94.67	67.87	33.27

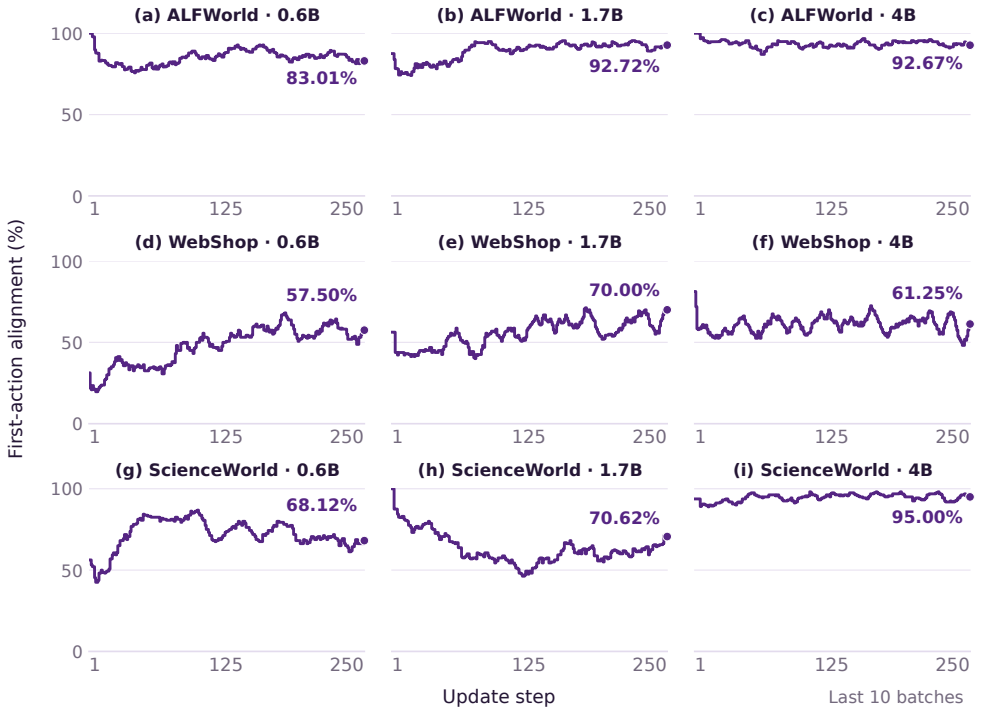


Figure 6: **First-action reference alignment during training.** Fraction of rollout tasks with reference guidance whose first transition passes the transition consistency check.

Larger students achieve higher cumulative full-trajectory alignment and aligned-turn coverage within each benchmark (Table 10). First-action alignment consistently exceeds full-trajectory alignment, indicating that a matching initial transition does not guarantee complete reference following. Aligned-turn coverage also captures reference-aligned prefixes in rollouts that later diverge. Windowed alignment rates do not increase monotonically during training (Figures 6–8).

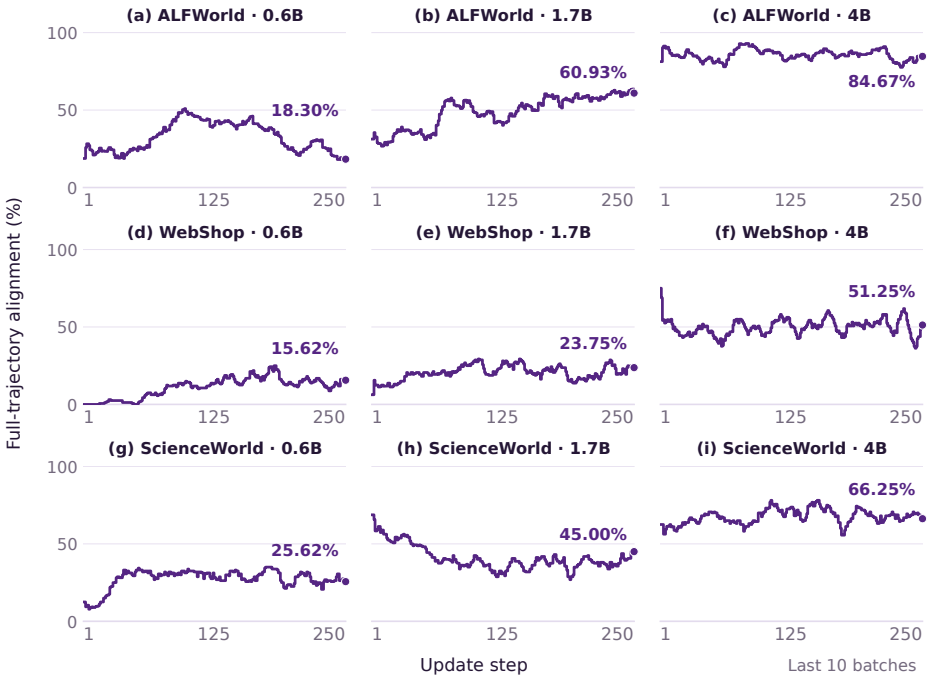


Figure 7: **Full-trajectory reference alignment during training.** Fraction of rollouts with reference guidance that complete the reference trajectory and terminate without triggering fallback.

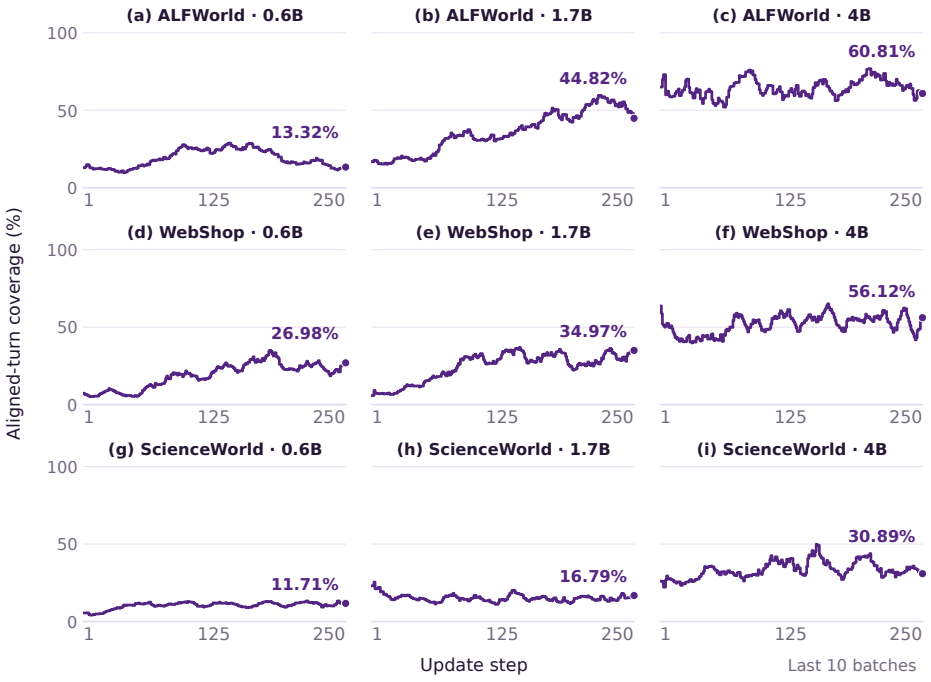


Figure 8: **Reference-aligned turn coverage during training.** Reference-aligned turns as a fraction of all executed turns, including fallback turns.