STATE2STATE: ENVIRONMENT-DERIVED MID-TRAINING FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training LLM agents commonly relies on supervised fine-tuning from expert trajectories or online reinforcement learning over human-specified tasks with handcrafted verifiers. Though effective, both remain bottlenecked by externally specified tasks and supervision signals, limiting the scalability and diversity of agent training. We study an *environment learning* paradigm in which agents acquire interaction and manipulation capabilities solely through environment interaction, without externally specified tasks. We propose STATE2STATE, an environment-derived mid-training method that converts explored environment states into training objectives, challenging agents to reach a specified target state. By deriving tasks from environment exploration and verifying success through rule-based state matching, STATE2STATE provides scalable and verifiable training objectives without expert supervision or manual task design. Experiments across textual and mobile GUI environments demonstrate the effectiveness of STATE2STATE as a standalone environment-learning stage, with performance gains observed in both interaction modalities. As initialization for downstream RL, it further improves final performance and learning efficiency, retains a 3.5 to 6.6 point margin over RL given the same total number of update steps, and provides initial evidence of positive transfer across environments.

1 INTRODUCTION

Existing methods for training LLM agents mainly follow two paradigms: supervised fine-tuning (SFT) from expert trajectories and reinforcement learning with verifiable rewards (RLVR). SFT trains agents to imitate trajectories collected from human experts or stronger teacher models (Zeng et al., 2024; Wang et al., 2025a), but such trajectories are costly to obtain, inherit the capability limits of their sources, and constrain learning to the coverage of a limited task distribution. RLVR reduces the reliance on expert demonstrations by allowing agents to learn through interaction (Wang et al., 2025c; Feng et al., 2025; Jin et al., 2025), but it shifts the bottleneck to task and reward construction: effective training still requires externally specified goals of suitable difficulty and manually designed verifiers for judging success.

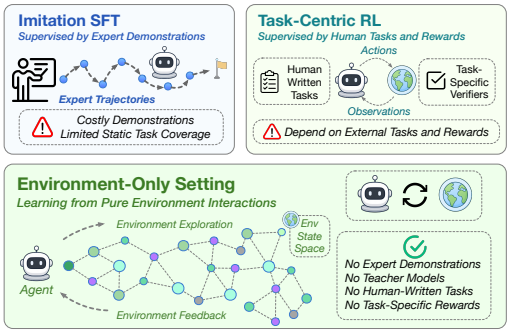


Figure 1: Comparison of supervision sources for LLM agent training. SFT relies on expert trajectories, and task-centric RL relies on human-specified tasks with task-specific verifiers. In contrast, the environment-only setting studied in this work derives learning signals purely from environment itself.

Recent task generation methods (Hu et al., 2025; Shi et al., 2025) attempt to alleviate this bottleneck by expanding the supply of training tasks. For example, recent works (Zhai et al., 2025; Lin et al., 2025) use stronger teacher models to collect interaction trajectories, synthesize task instructions, and construct domain-specific verifiers. While these methods improve the scalability of task acquisition, they still largely treat agent training as learning to solve externally formulated tasks. As a result,

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.

the diversity and depth of training objectives remain tied to human-task distributions, teacher-model capabilities, or domain-specific task construction pipelines.

Viewed through the lens of supervision signals, SFT relies on expert action trajectories, while RLVR relies on externally specified goals and verifiers. However, both paradigms remain centered on human-defined task distributions and external supervision sources. As illustrated in Figure 1, this raises a natural question beyond action-level or goal-level supervision: *Can agents learn useful environment perception and manipulation capabilities directly from environment-only supervision?* In this work, we study this question through an *environment learning* paradigm. Environment learning considers a minimal training setting with only the agent and the environment, where supervision is derived from the environment itself, without external teachers or human-authored tasks. By first learning to understand and manipulate the environment, agents can acquire primitive environment skill priors that later serve as a foundation for RL on human-specified tasks, where these capabilities are aligned with human intent. This progression makes environment learning naturally suited for agent mid-training (Tu et al., 2025), which aims to develop useful capabilities by intermediate training before optimizing specific downstream tasks.

Under this setting, we propose STATE2STATE, a simple instantiation of environment learning that turns reachable environment states into verifiable mid-training objectives. As shown in Figure 2, STATE2STATE begins by exploring the environment with an exploration policy to collect reproducible reachable states. It then filters and samples learnable target states to improve diversity and reproducibility, and pairs each target with a corresponding initial configuration to construct a state-reaching task. In each task, given the initial observation and the target observation, the agent must reason and act through environment interactions to transform the current state into the target state. By sampling from explored states rather than predefined static tasks, STATE2STATE derives task diversity directly from the reachable environment state space, exposing agents to environment dynamics beyond human-designed task distributions. Task success can be verified by matching the resulting state against the target state, yielding rule-based rewards without task-specific test cases. We train agents on these objectives using GRPO-based reinforcement learning (Shao et al., 2024) with dynamic sampling (Yu et al., 2025), which collects informative rollouts for effective and stable training. Overall, STATE2STATE turns environment exploration into verifiable RL objectives, allowing agents to learn from the environment itself without expert demonstrations, teacher models, or human-authored task instructions, while requiring only reproducible environment states and a rule-based state-matching verifier. The resulting environment-aware policy serves as a stronger starting point for subsequent human-task RL, which further aligns the environment skill priors with human intent and achieves better task completion performance.

We evaluate STATE2STATE on ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022), covering household and scientific environments, and further extend it to two mobile GUI benchmarks, MobileWorld (Kong et al., 2025) and MobileGym (Wu et al., 2026a). Across these environments, STATE2STATE improves agent performance before task-specific RL in most settings, showing that verifiable state-reaching objectives can induce useful environment understanding and manipulation capabilities. Serving as a stronger initialization for subsequent RL on human-specified tasks, STATE2STATE further improves final task performance and learning efficiency of downstream RL, demonstrating its effectiveness as an environment-centric mid-training stage. We also observe positive transfer from ScienceWorld to ALFWorld: models mid-trained with STATE2STATE generalize better than those trained on standard ScienceWorld tasks, suggesting that environment-derived objectives potentially support stronger cross-environment generalization. Finally, STATE2STATE only requires a way to match current and target states, making it naturally applicable beyond textual environments. Using screenshot and UI-text similarity for mobile GUI states, task-free mid-training improves both a GUI-specialized agent on MobileWorld and a general vision-language model on MobileGym, without human-specified training tasks.

In summary, our main contributions are:

- We study *environment learning* as a mid-training paradigm for LLM agents, where agents learn environment perception and manipulation capabilities from direct agent-environment interaction without external teachers or manually designed tasks.
- We introduce STATE2STATE, which turns explored reachable states into verifiable state-reaching objectives and trains agents with state-matching rewards, enabling reinforcement learning from the environment itself.

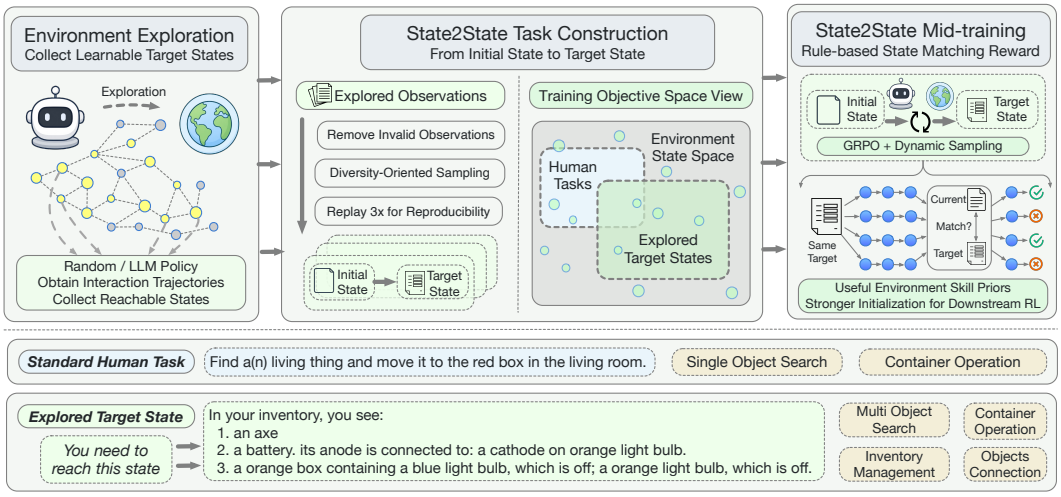


Figure 2: Overview of STATE2STATE, an environment-derived mid-training method. STATE2STATE converts explored environment states into reproducible training objectives and trains agents with rule-based state-matching rewards. By sampling targets from explorations, STATE2STATE aims to expose agents to a different and potentially broader set of environment states, not limited by human-task distributions.

- We evaluate STATE2STATE on textual and mobile benchmarks, demonstrating standalone gains across different interaction modalities, better initialization for subsequent human-task RL under a matched update budget, and initial evidence of positive cross-environment transfer.

2 RELATED WORK

LLM Agent Training. Recent work has improved LLM agents through supervised fine-tuning and reinforcement learning on task-oriented interaction data. Early studies (Zeng et al., 2024; Chen et al., 2023) fine-tune LLMs on expert trajectories to elicit tool use and decision making. More recent efforts (Li et al., 2025; Team, 2025a; Su et al., 2025; Wang et al., 2025a) further scale this paradigm by automatically constructing task instructions and corresponding trajectories. In parallel, agentic RL (Qi et al., 2025; Wang et al., 2025c) optimizes agents through online interactions and verifiable rewards. Other studies further refine long-horizon behavior with more fine-grained optimization signals (Feng et al., 2025; Zhang et al., 2025b). Despite their effectiveness, these methods largely rely on predefined tasks, expert trajectories or manually designed rewards.

Agentic Mid-Training. Agentic mid-training introduces intermediate objectives before final task training. Prior work (Tu et al., 2025) studies general mid-training for LLMs, and this idea has been extended to agent training by introducing intermediate objectives including world modeling (Wu et al., 2026b; Chen et al., 2025a; Yu et al., 2026), reflection and action evaluation (Liu et al., 2026; Xia et al., 2025; Chen et al., 2025b; Yuan et al., 2025) and domain-specific skill-prior learning (Zeng et al., 2026; Yang et al., 2025). As a representative, Agent Early Experience (Zhang et al., 2025a) develops two variants for constructing world-modeling and self-reflection data from reward-free interactions respectively.

Learning from Environment Exploration. A closely related direction leverages environment exploration to generate agent training signals. Recent efforts (Zhai et al., 2025; Lin et al., 2025; Wang et al., 2025b) collect autonomous interaction experience and transform the resulting trajectories into tasks, demonstrations or world knowledge, often with additional teacher annotation. These works demonstrate the value of exploration, but they often require model-based annotation to ensure quality, and their generated objectives remain largely guided by human-task distributions. In contrast, STATE2STATE enables environment learning without external teachers or human-task distributions.

Goal-Conditioned and Hindsight RL. Learning to reach specified states is a long-standing formulation in goal-conditioned RL (Kaelbling, 1993), and hindsight methods relabel achieved outcomes as goals to densify sparse rewards (Andrychowicz et al., 2017; Rauber et al., 2019). Recent work

extends this idea to LLM agents, relabeling achieved or failed trajectories into alternative goals or supervision (Ding, 2026; Li et al., 2026; Pourcel et al., 2025). STATE2STATE adapts this principle to LLM agent mid-training and differs in three ways: it discovers goals through task-free exploration rather than assuming or relabeling task goals, derives supervision directly from the discovered target states, and uses goal reaching as environment mid-training for downstream RL not as the task itself.

3 METHOD

3.1 OVERVIEW

We present STATE2STATE, an environment-centric mid-training method that turns explored environment states into verifiable RL objectives. The core idea is to derive both the task and reward from the environment itself. Rather than relying on expert trajectories or human-specified tasks, it derives both tasks and rewards directly from environment interaction: as shown in Figure 2, exploration collects candidate target states, task construction turns them into reproducible state-reaching objectives, and RL trains the agent to reach these targets. The formulation applies when explored states can be reproduced through replay and observations provide a reliable proxy for verifying state equivalence.

3.2 ENVIRONMENT EXPLORATION

The first stage of STATE2STATE collects reachable states through environment exploration. Because any reached state can serve as a target, STATE2STATE is compatible with a broad range of explorers: the explorer need only visit diverse and sufficiently deep states, rather than solve predefined tasks. To preserve the environment-only setting, we use only explorers available within the agent and environment system itself: random exploration and self-exploration by the agent base model, requiring neither human supervision nor stronger external models.

At each step, the explorer samples an action from the valid action set derived from the environment and executes it. When useful, the sampling distribution can be lightly adjusted with environment-specific priors, such as favoring navigation actions in early steps to enhance exploration breadth and manipulation actions in later steps to prioritize interaction depth. Since no model inference is required, random exploration can be highly parallelized and scaled primarily with environment interaction, enabling the construction of large candidate pools of target states. In GUI environments, where deep application states often require long, coherent action sequences, self-exploration can be combined with random exploration to extend exploration depth (Appendix J.5).

3.3 STATE2STATE TASK CONSTRUCTION

We construct STATE2STATE tasks from trajectories collected during environment exploration. In many interactive environments, the full underlying state is inaccessible or environment-specific, while the agent observes the environment through textual or structured observations. We therefore use observations as an operational representation of state: a target state is specified by a reachable observation, and task success is verified by matching the current observation against the target.

Given the explored observation pool, we first filter out invalid or uninformative observations, such as error messages and meaningless transitions. We then sample target observations with a diversity-oriented strategy that favors broad coverage of unique reachable observations while limiting excessive repetition of the same instance. To ensure that targets define reproducible objectives, we replay the corresponding exploration trajectory from the same initial configuration three times and retain only observations that can be consistently reached.

Each retained target observation is paired with its source environment configuration to form a STATE2STATE task. At training time, the environment is reset to this initial configuration, and the agent is asked to reach the target observation through interaction. Formally, let s denote an underlying environment state and $o(s)$ the observation exposed to the agent, and let $\mathcal{O}_{\text{exp}}$ denote the observations collected during exploration. A STATE2STATE task is specified by an initial state s_0 and a target observation $o^\star \in \mathcal{O}_{\text{exp}}$, which operationally represents the target state. The policy is tasked with driving the environment from s_0 to any state whose observation matches $o^\star$.

We measure target attainment with an observation-matching function $m(o, o^\star) \in [0, 1]$, which also serves as the task verifier. The episode reward is the match score of the final observation:

$$r = m\big(o(s_T), o^\star\big), \tag{1}$$

where T is the final step of the episode, which terminates when the agent stops, or the step budget is exhausted. The function m is environment-specific but shared across all generated tasks, making it a task-agnostic verifier: adding new tasks requires no additional reward design. In deterministic textual environments, m is normalized exact match, yielding binary rewards and terminating the episode once the target is reached. In GUI environments, where equivalent states may have visually different renderings, m combines screenshot similarity with UI-hierarchy text similarity (Appendix J). Both verifiers are rule-based and require neither learned reward models nor LLM judges.

These verifiers are intentionally simple and may miss semantically equivalent states. Human audits of 100 cases per setting find 94% agreement for the textual verifier and a Cohen's κ of 0.64 for the GUI verifier, with most textual mismatches arising from ordering or wording differences (Appendix C).

3.4 STATE2STATE MID-TRAINING

We optimize STATE2STATE tasks with the GRPO algorithm augmented with dynamic sampling (Yu et al., 2025). For each target observation, we sample multiple rollouts from the same initial configuration and optimize them with the GRPO algorithm, which compares successful and failed attempts to provide contrastive rewards. Additionally, we apply dynamic sampling by discarding rollout groups with identical rewards and continuing to sample new groups until the required batch size is reached. This increases the density of informative groups and adaptively focuses training on targets near the current policy frontier, allowing the model to effectively learn from environment-derived targets.

Finally, STATE2STATE serves as an environment-centric mid-training stage before downstream RL on human-specified tasks. After environment learning, we continue training the policy on human-specified tasks using reinforcement learning, where objectives are natural-language instructions and rewards are given by task-specific verifiers. This design separates environment capability acquisition from human-intent alignment: STATE2STATE first provides scalable environment-grounded skill priors, while downstream RL adapts these priors to solve human-specified tasks.

4 EXPERIMENTS

To demonstrate the effectiveness of STATE2STATE, we examine two questions: whether STATE2STATE alone improves agents without any human-specified task, and whether it further improves downstream agent RL beyond direct task-specific training.

4.1 BENCHMARKS AND EVALUATION

We evaluate STATE2STATE on two textual environments, ScienceWorld (Wang et al., 2022) and ALFWorld (Shridhar et al., 2021), and on two mobile GUI benchmarks described in Section 4.5. ScienceWorld focuses on science-oriented environments, where agents must gather observations, perform procedural manipulation, and execute long-horizon action sequences to complete a scientific experiment. ALFWorld focuses on household navigation and manipulation in textual embodied environments. For both benchmarks, we report task success rates on in-distribution (*ID*) and out-of-distribution (*OOD*) partitions. For ScienceWorld, we randomly sample 400 tasks from each partition because of its large size; for ALFWorld, we use the full set. For stable evaluation, we set temperature to 0.1 during model generation. We include more details in Appendix A.

4.2 EXPERIMENT SETUP

We study STATE2STATE as a mid-training stage: the downstream human-task RL budget is held fixed, and STATE2STATE adds an environment-derived stage before it. Because this adds policy-update steps, we also report a compute-aligned baseline that instead spends the same number of additional updates on longer human-task RL. The question is therefore not whether additional training helps, but whether environment-derived objectives convert that budget into downstream capability more effectively than more human-task RL. We conduct experiments with two models of different scales,

Table 1: Main results on ALFWorld and ScienceWorld, reported as task success rates. Human-Specified Task Training indicates whether the model is further trained on standard human-specified tasks in our experiments. RL (compute-aligned) spends the same number of additional update steps as STATE2STATE mid-training on longer human-task RL. STATE2STATE generally improves base models as a standalone environment-learning stage, and the full STATE2STATE + RL pipeline achieves the best performance across both benchmarks and model scales.

Model	Method	Human-Specified Task Training	ALFWorld			ScienceWorld		
			ID	OOD	Average	ID	OOD	Average
Prompting-based	GPT-5.2	✗	75.71	78.36	77.04	50.50	47.50	49.00
	Claude 4.5 Haiku	✗	79.29	81.34	80.32	48.00	48.00	48.00
	Qwen-Plus	✗	80.00	85.07	82.54	44.50	45.50	45.00
	DeepSeek V4 Flash	✗	84.29	87.31	85.80	44.50	39.00	41.75
Qwen3-4B	Base	✗	35.71	31.34	33.53	25.75	21.50	23.63
	STATE2STATE (Ours)	✗	43.57	46.27	44.92	24.50	17.50	21.00
	Agent Early Experience	✓	67.14	76.87	72.01	41.00	41.00	41.00
	Distillation SFT	✓	64.29	67.91	66.10	48.00	44.75	46.38
	RL	✓	86.43	85.82	86.13	51.00	48.25	49.63
	RL (compute-aligned)	✓	85.71	88.81	87.26	51.75	51.00	51.38
	STATE2STATE + RL (Ours)	✓	**90.00**	**91.79**	**90.90**	**59.75**	**51.25**	**55.50**
Qwen3-8B	Base	✗	74.29	76.12	75.21	28.50	34.50	31.50
	STATE2STATE (Ours)	✗	70.00	84.33	77.17	31.75	34.50	33.13
	Agent Early Experience	✓	70.00	72.39	71.20	43.25	44.00	43.63
	Distillation SFT	✓	83.57	88.81	86.19	47.00	49.75	48.38
	RL	✓	91.43	93.28	92.36	56.00	48.25	52.13
	RL (compute-aligned)	✓	92.14	89.55	90.85	56.50	48.50	52.50
	STATE2STATE + RL (Ours)	✓	**97.14**	**97.76**	**97.45**	**59.50**	**52.50**	**56.00**

Qwen3-4B and Qwen3-8B (Team, 2025b), under several training settings. STATE2STATE trains the model solely on environment-derived STATE2STATE tasks, which evaluates the transferable interaction capabilities induced by environment learning alone. RL-only (denoted as RL) directly optimizes the model on downstream human tasks without mid-training. RL (compute-aligned) extends RL-only by 80 additional policy updates on human tasks, matching the total number of policy updates of STATE2STATE + RL. STATE2STATE + RL is our full pipeline, where the model first learns environment perception and manipulation, and then continues training on downstream human tasks. Comparing this setting with the two RL baselines isolates the effect of STATE2STATE from the effect of a larger optimization budget.

For additional reference, we also include two baselines based on expert demonstrations: distillation SFT and agent early experience (Zhang et al., 2025a), described in Appendix H. For SFT, we collect 200 high-quality successful trajectories from Qwen-Plus using rejection sampling. Additionally, we report the performance of several competent LLMs including GPT-5.2 (Singh et al., 2025), Claude 4.5 Haiku (Anthropic, 2025), DeepSeek V4 Flash (DeepSeek-AI, 2026) and Qwen-Plus (Team, 2025b). For ScienceWorld, the prompting agents are tested on 200 samples each split to reduce cost.

4.3 IMPLEMENTATION DETAILS

We use random exploration in both environments. For ScienceWorld, we adopt a two-stage sampling strategy that favors movement and observation early and state-changing scientific actions later. Exploration is run in parallel for hundreds of steps per episode, followed by filtering and diversity-based target sampling with at most five repetitions per observation. Dataset statistics and representative examples are provided in Appendix B.

We train STATE2STATE with GRPO augmented with dynamic sampling for 80 steps and continue downstream training with the best-performing checkpoint on the state-reaching validation set. This mid-training stage accounts for roughly a quarter of the total GPU-hours and a fifth of the total online

Table 2: Standalone STATE2STATE in mobile GUI environments, trained without any human-specified task. Rows above the line are selected entries from the official leaderboards (Appendix J).

MobileGym	
Model	Success Rate
Gemini 3.1 Pro	58.8
UI-Venus-1.5-8B	15.4
GUI-Owl-1.5-8B-Think	15.1
UI-TARS-1.5-7B	13.8
Step-GUI-4B	12.9
Qwen3-VL-4B	10.2
STATE2STATE	16.0

MobileWorld (GUI-only)	
Model	Success Rate
Gemini-3.1-Pro-Preview	58.1
MAI-UI-32B	36.2
MemGUI-Agent-235B	29.1
Doubao-1.5-UI-TARS	26.3
Qwen3-VL-8B	9.4
MAI-UI-8B	27.5
STATE2STATE	30.8

environment interactions of the full pipeline; Appendix D reports hardware, wall-clock time, rollout counts, and throughput per run. We present more details in Appendix I.

4.4 MAIN RESULTS

Standalone environment learning. Table 1 reports the main results on ALFWorld and ScienceWorld. As a standalone environment-learning stage, STATE2STATE improves the average success rate in three of the four model–environment settings, by $+11.39$ (Qwen3-4B) and $+1.96$ (Qwen3-8B) points on ALFWorld and by $+1.63$ points for Qwen3-8B on ScienceWorld. This shows that environment-centric STATE2STATE can induce useful environment cognition and manipulation capabilities even before training on human-specified tasks. The only exception is Qwen3-4B on ScienceWorld with slight degradation, possibly due to the challenging exact state-reaching objectives for smaller models.

Stronger initialization for downstream RL. The full STATE2STATE + RL pipeline achieves the best performance across both environments and model scales. Its gains over direct RL appear not only on ALFWorld, where RL is already strong, but also on the more challenging ScienceWorld benchmark. Together with consistent improvements on both ID and OOD splits, these results suggest that STATE2STATE provides useful environment-level capabilities rather than merely fitting a specific task distribution. The resulting models also surpass the strong prompting LLMs in these environments.

Gains are not explained by extra training steps. Spending the same additional budget on longer human-task RL closes only part of the gap: STATE2STATE + RL still leads the compute-aligned baseline by 3.50 to 6.60 points across the four settings, and longer RL slightly degrades Qwen3-8B on ALFWorld. This indicates that the improvement comes from the environment-derived state-reaching objective rather than from the larger optimization budget alone.

Improved learning efficiency. Figure 3 further shows that STATE2STATE improves not only final performance but also downstream RL efficiency. Once the mid-training steps are charged to STATE2STATE on the horizontal axis, it still reaches the marked reference score well before the baseline, which needs nearly twice as many updates to match it, and stays ahead for the rest of training. This indicates that environment-level mid-training provides useful skill priors that accelerate subsequent learning on human tasks, rather than simply buying a head start.

Latent benefits released by human-task RL. Mid-training helps potentially more after RL than on its own: for Qwen3-8B, STATE2STATE + RL gains $+5.09$ and $+3.87$ over RL on ALFWorld and ScienceWorld, versus $+1.96$ and $+1.63$ for STATE2STATE alone over the base model, suggesting that downstream RL turns latent environment skills into task performance.

4.5 MOBILE GUI ENVIRONMENTS

Mobile GUI environments are a natural target for environment learning: observations are visual, tasks span long action sequences across applications, and human-specified training tasks with reliable verifiers are scarce, with MobileWorld (Kong et al., 2025) providing none. We therefore evaluate STATE2STATE in the environment learning setting: each agent is trained on environment-derived targets alone, without any human-specified task, and evaluated directly on the benchmark tasks. We use two benchmarks with two kinds of base agent: MobileGym (Wu et al., 2026a), a verifiable, highly

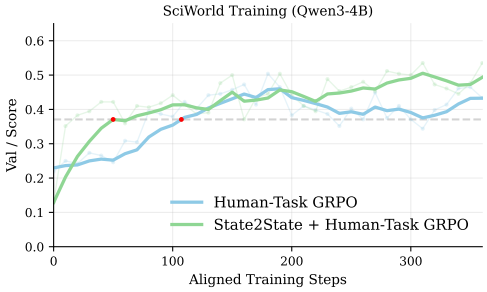


Figure 3: Compute-aligned curves for Qwen3-4B on ScienceWorld. At step 0 both runs have spent the same number of updates on mid-training and human-task RL respectively.

Table 3: Ordering of SFT and STATE2STATE, with STATE2STATE denoted as S2S. SFT + S2S + RL performs best, showing that S2S is most effective as a mid-training stage placed after SFT and before RL.

Method	ID	OOD	Avg.
Base	25.75	21.50	23.63
SFT	48.00	44.75	46.38
S2S + SFT	47.25	48.75	48.00
SFT + S2S	55.75	52.50	54.13
SFT + RL	65.00	51.75	58.38
S2S + SFT + RL	60.75	60.25	60.50
SFT + S2S + RL	**70.75**	**61.00**	**65.88**

parallel simulation platform, with the general vision-language model Qwen3-VL-4B (Team, 2025c), and the GUI-only subset of MobileWorld, a long-horizon, cross-application benchmark, with the GUI-specialized MAI-UI-8B. The policy observes screenshots, and the matching function compares screenshots and UI text instead of strings (Section 3.3; Appendix J).

As shown in Table 2, STATE2STATE lifts Qwen3-VL-4B from 10.2 to 16.0 on MobileGym, overtaking four GUI-specialized models of 4B to 8B parameters, and lifts MAI-UI-8B from 27.5 to 30.8 on MobileWorld, overtaking MemGUI-Agent-235B, all without any human-specified training task. Together with the textual results, this shows that environment-derived objectives alone improve agents in realistic, visually grounded environments.

5 ANALYSIS

We conduct the following analyses to better understand the mechanisms of STATE2STATE, with experiments on ScienceWorld with the Qwen3-4B model unless otherwise specified.

5.1 ORDERING OF SFT AND STATE2STATE

We analyze how STATE2STATE interacts with commonly used SFT method using the same dataset as the distillation SFT baseline. When applying STATE2STATE on the stronger SFT model, we observe more consistent and stable performance gains and extend the STATE2STATE training to 150 steps. As shown in Table 3, applying STATE2STATE after SFT substantially improves over SFT, showing that STATE2STATE objectives derived from environment explorations still provide complementary benefits beyond SFT.

The benefit becomes clearer when followed by downstream RL. SFT + STATE2STATE + RL achieves the best overall and OOD performance, indicating that STATE2STATE remains effective on top of SFT, benefiting from a stronger initial policy. This suggests that expert demonstrations and environment learning are complementary: while SFT imitates human-task trajectories, STATE2STATE trains on state-reaching objectives sampled from a broader reachable environment space. The training order is also important: applying STATE2STATE before SFT brings limited gains, likely because subsequent imitation learning can overwrite part of the exploratory environment priors.

5.2 INTEGRATION WITH DIFFERENT RL BACKBONES

We examine whether STATE2STATE remains effective with a stronger downstream RL backbone algorithm. We use GiGPO (Feng et al., 2025), an RL algorithm with fine-grained rewards designed for agentic RL, and compare each backbone with and without STATE2STATE mid-training.

As shown in Table 4, STATE2STATE + GRPO reaches 55.50 average, outperforming GiGPO alone on average, showing that environment learning with a standard RL backbone can already match or

Table 4: Analysis of different RL algorithms on ScienceWorld. STATE2STATE improves both GRPO and GiGPO, showing that its benefit is complementary to the downstream optimizer.

Method	ID	OOD	Avg.
Base	25.75	21.50	23.63
GRPO	51.00	48.25	49.63
STATE2STATE + GRPO	59.75	51.25	55.50
GiGPO	60.25	49.25	54.75
STATE2STATE + GiGPO	**61.75**	**53.75**	**57.75**

Table 5: Analysis of exploration strategies on ScienceWorld. Random exploration yields the strongest STATE2STATE + RL result, showing that low-cost state discovery can provide effective training targets.

Method	Exploration	ID	OOD	Avg.
Base	–	25.75	21.50	23.63
RL	–	51.00	48.25	49.63
STATE2STATE + RL	LLM	54.75	50.50	52.63
STATE2STATE + RL	Random	**59.75**	**51.25**	**55.50**

exceed a stronger RL algorithm without STATE2STATE. When combined with GiGPO, STATE2STATE further improves the performance on both ID and OOD partitions.

5.3 EFFECT OF EXPLORATION STRATEGY

To study how the choice of explorer affects STATE2STATE, we compare random exploration with a stronger external LLM explorer, Qwen-Plus, in the textual environment. As shown in Table 5, both improve over direct RL, indicating that useful target states can be collected by different explorers.

Notably, random exploration outperforms the stronger LLM explorer. Our observations suggest that the task-oriented priors of LLMs may concentrate exploration on a potentially deeper but narrower region of the state space, whereas random exploration reaches more diverse states without additional inference cost. This result suggests that effective STATE2STATE exploration does not require a stronger external model.

5.4 CROSS-ENVIRONMENT GENERALIZATION

To examine the generalization of STATE2STATE, we evaluate cross-environment transfer in both directions: STATE2STATE mid-training in one environment followed by downstream RL in the other. For the ScienceWorld-to-ALFWorld direction, we also compare against a variant that uses ScienceWorld human-task RL as mid-training, with the same amount of mid-training steps.

As shown in Table 6 and Appendix F, cross-environment STATE2STATE improves over no mid-training in both directions, with a more modest gain when transferring into the more challenging ScienceWorld. Mid-training on ScienceWorld human tasks instead of STATE2STATE objectives transfers much less ($+0.74$), suggesting that the state-reaching objective, rather than additional cross-environment RL alone, is what produces transferable environment capabilities. The bidirectional transfer gains, together with the weaker transfer from human-task RL, suggest that STATE2STATE learns environment capabilities that extend beyond the tasks and environment on which they are acquired.

Table 6: Transfer from ScienceWorld mid-training to downstream RL on ALFWorld, with STATE2STATE denoted as S2S. Only the mid-training stage varies.

Mid-training	ID	OOD	Avg.
None	86.43	85.82	86.13
ScienceWorld RL	86.43	87.31	86.87
ScienceWorld S2S	**88.57**	**90.30**	**89.44**

6 CONCLUSION

We present STATE2STATE, an environment-derived mid-training method that converts explored environment states into verifiable state-reaching objectives. By deriving both objectives and rewards from the environment, STATE2STATE enables agents to acquire environment skill priors without expert supervision or human-specified tasks. Across several environments, STATE2STATE standalone generally improves agent performance, strengthens downstream human-task RL, and shows promising cross-environment generalization.

AI USE STATEMENT

We used generative AI tools only to polish author-written prose, specifically for grammar checking and for improving sentence-level readability. We did not use generative AI tools for research ideation, method design, experimental design or drawing conclusions. All substantive content of this paper, including the research idea, the method, and the experiments, was developed and verified by the authors.

REPRODUCIBILITY STATEMENT

All experiments use publicly available environments and open-weight models for training, and the appendix documents the materials to reproduce the results. Appendix A specifies the environment interfaces, the action spaces, and the reward definitions used for both the standard and the STATE2STATE objectives. Appendix B gives the full dataset construction pipeline, including exploration seeds and episode budgets, the filtering and diversity-sampling rules, the resulting dataset statistics, and representative target states. Appendix I lists the RL hyperparameters, step budgets, and the verbatim training prompts for every textual setting, and Appendix H defines the baselines, including the compute-aligned RL baseline. Appendix G describes the RL algorithms, Appendix D reports the hardware, wall-clock time, and GPU-hours of each run, and Appendix J covers the GUI setting. We also report the reliability of the rule-based verifier in Appendix C and run-to-run variance in Appendix E, so that the size of the reported gains can be weighed against both verifier noise and seed variance. We will release the main code, the generated STATE2STATE datasets, and the training configurations.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5048–5058, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/453fadbd8a1a3af50a9df4df899537b5-Abstract.html.

Anthropic. System card: Claude haiku 4.5, 2025. URL https://www-cdn.anthropic.com/7aad69bf12627d42234e01ee7c36305dc2f6a970.pdf.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *CoRR*, abs/2310.05915, 2023. doi: 10.48550/ARXIV.2310.05915. URL https://doi.org/10.48550/arXiv.2310.05915.

Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. Internalizing world models via self-play finetuning for agentic RL. *CoRR*, abs/2510.15047, 2025a. doi: 10.48550/ARXIV.2510.15047. URL https://doi.org/10.48550/arXiv.2510.15047.

Yihan Chen, Benfeng Xu, Xiaorui Wang, Yongdong Zhang, and Zhendong Mao. Training llm-based agents with synthetic self-reflected trajectories and partial masking. *CoRR*, abs/2505.20023, 2025b. doi: 10.48550/ARXIV.2505.20023. URL https://doi.org/10.48550/arXiv.2505.20023.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026. doi: 10.48550/ARXIV.2606.19348. URL https://doi.org/10.48550/arXiv.2606.19348.

Liang Ding. Agenther: Hindsight experience replay for LLM agent trajectory relabeling. *CoRR*, abs/2603.21357, 2026. doi: 10.48550/ARXIV.2603.21357. URL https://doi.org/10.48550/arXiv.2603.21357.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978, 2025. doi: 10.48550/ARXIV.2505.10978. URL https://doi.org/10.48550/arXiv.2505.10978.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. In Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins (eds.), *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, pp. 496–507. ACM, 2025. doi: 10.1145/3690624.3709321. URL https://doi.org/10.1145/3690624.3709321.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Leslie Pack Kaelbling. Learning to achieve goals. In Ruzena Bajcsy (ed.), *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pp. 1094–1099. Morgan Kaufmann, 1993.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. Mobile-world: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *CoRR*, abs/2512.19432, 2025. doi: 10.48550/ARXIV.2512.19432. URL https://doi.org/10.48550/arXiv.2512.19432.

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 ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, Kuan Li, Liangcai Su, Litu Ou, Liwen Zhang, Pengjun Xie, Rui Ye, Wenbiao Yin, Xinmiao Yu, Xinyu Wang, Xixi Wu, Xuanzhong Chen, Yida Zhao, Zhen Zhang, Zhengwei Tao, Zhongwang Zhang, Zile Qiao, Chenxi Wang, Donglei Yu, Gang Fu, Haiyang Shen, Jiayin Yang, Jun Lin, Junkai Zhang, Kui Zeng, Li Yang, Hailong Yin, Maojia Song, Ming Yan, Peng Xia, Qian Xiao, Rui Min, Ruixue Ding, Runnan Fang, Shaowei Chen, Shen Huang, Shihang Wang, Shihao Cai, Weizhou Shen, Xiaobin Wang, Xin Guan, Xinyu Geng, Yingcheng Shi, Yuning Wu, Zhuo Chen, Zijian Li, and Yong Jiang. Tongyi deepresearch technical report. *CoRR*, abs/2510.24701, 2025. doi: 10.48550/ARXIV.2510.24701. URL https://doi.org/10.48550/arXiv.2510.24701.

Zichao Li, Gang Wu, Zichao Wang, Ruiyi Zhang, Wanrong Zhu, Ryan A. Rossi, Vlad I. Morariu, and Jihyung Kil. Spinning straw into gold: Relabeling LLM agent trajectories in hindsight for successful demonstrations. *CoRR*, abs/2607.04235, 2026. doi: 10.48550/ARXIV.2607.04235. URL https://doi.org/10.48550/arXiv.2607.04235.

Musen Lin, Minghao Liu, Taoran Lu, Lichen Yuan, Yiwei Liu, Haonan Xu, Yu Miao, Yuhao Chao, and Zhaojian Li. Gui-rewalk: Massive data generation for GUI agent via stochastic exploration and intent-aware reasoning. *CoRR*, abs/2509.15738, 2025. doi: 10.48550/ARXIV.2509.15738. URL https://doi.org/10.48550/arXiv.2509.15738.

Weize Liu, Minghui Liu, Sy-Tuyen Ho, Souradip Chakraborty, Xiyao Wang, and Furong Huang. Agentic critical training. *CoRR*, abs/2603.08706, 2026. doi: 10.48550/ARXIV.2603.08706. URL https://doi.org/10.48550/arXiv.2603.08706.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on ARC-AGI. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pourcel25a.html.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on*

Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

Paulo E. Rauber, Avinash Ummadisingu, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg2viA5FQ.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. Taskcraft: Automated generation of agentic tasks. *CoRR*, abs/2506.10055, 2025. doi: 10.48550/ARXIV.2506.10055. URL https://doi.org/10.48550/arXiv.2506.10055.

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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, Zile Qiao, Zhongwang Zhang, Huifeng Yin, Shihao Cai, Runnan Fang, Zhengwei Tao, Wenbiao Yin, Chenxiong Qian, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Scaling agents via continual pre-training. *CoRR*, abs/2509.13310, 2025. doi: 10.48550/ARXIV.2509.13310. URL https://doi.org/10.48550/arXiv.2509.13310.

Kimi Team. Kimi K2: open agentic intelligence. *CoRR*, abs/2507.20534, 2025a. doi: 10.48550/ARXIV.2507.20534. URL https://doi.org/10.48550/arXiv.2507.20534.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025b. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Qwen Team. Qwen3-vl technical report. *CoRR*, abs/2511.21631, 2025c. doi: 10.48550/ARXIV.2511.21631. URL https://doi.org/10.48550/arXiv.2511.21631.

Chengying Tu, Xuemiao Zhang, Rongxiang Weng, Rumei Li, Chen Zhang, Yang Bai, Hongfei Yan, Jingang Wang, and Xunliang Cai. A survey on LLM mid-training. *CoRR*, abs/2510.23081, 2025. doi: 10.48550/ARXIV.2510.23081. URL https://doi.org/10.48550/arXiv.2510.23081.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.775. URL https://doi.org/10.18653/v1/2022.emnlp-main.775.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Haotian Yao, Ziwei Chen, Qizheng Gu, Yipu Wang, Heng Wang, Diyi Yang,

Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents. *CoRR*, abs/2508.09123, 2025a. doi: 10.48550/ARXIV.2508.09123. URL https://doi.org/10.48550/arXiv.2508.09123.

Zhaoyang Wang, Yiming Liang, Xuchao Zhang, Qianhui Wu, Siwei Han, Anson Bastos, Rujia Wang, Chetan Bansal, Baolin Peng, Jianfeng Gao, et al. Adapting web agents with synthetic supervision. *arXiv preprint arXiv:2511.06101*, 2025b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, 2025c. doi: 10.48550/ARXIV.2504.20073. URL https://doi.org/10.48550/arXiv.2504.20073.

Dingbang Wu, Rui Hao, Haiyang Wang, Shuzhe Wu, Han Xiao, Zhenghong Li, Bojiang Zhou, Zheng Ju, Zichen Liu, Lue Fan, and Zhaoxiang Zhang. Mobilegym: A verifiable and highly parallel simulation platform for mobile GUI agent research. *CoRR*, abs/2605.26114, 2026a. doi: 10.48550/ARXIV.2605.26114. URL https://doi.org/10.48550/arXiv.2605.26114.

Mengzhou Wu, Yuzhe Guo, Yuan Cao, Haochuan Lu, Songhe Zhu, Pingzhe Qu, Xin Chen, Kang Qin, Zhongpu Wang, Xiaode Zhang, Xinyi Wang, Wei Dai, Gang Cao, Yuetang Deng, Zhi Gong, Dezhi Ran, Linyi Li, Wei Yang, and Tao Xie. Ui-oceanus: Scaling GUI agents with synthetic environmental dynamics. *CoRR*, abs/2604.02345, 2026b. doi: 10.48550/ARXIV.2604.02345. URL https://doi.org/10.48550/arXiv.2604.02345.

Yu Xia, Yiran Shen, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, Lina Yao, and Julian J. McAuley. SAND: boosting LLM agents with self-taught action deliberation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 3062–3077. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.152. URL https://doi.org/10.18653/v1/2025.emnlp-main.152.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, Yingwei Ma, Yanhao Li, Yue Liu, Zhenxing Hu, Kaitai Zhang, Shuyi Wang, Huarong Chen, Flood Sung, Yang Liu, Yang Gao, Zhilin Yang, and Tianyu Liu. Kimi-dev: Agentless training as skill prior for swe-agents. *CoRR*, abs/2509.23045, 2025. doi: 10.48550/ARXIV.2509.23045. URL https://doi.org/10.48550/arXiv.2509.23045.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jiangfeng Gao, and Zhou Yu. Reinforcement world model learning for llm-based agents. *CoRR*, abs/2602.05842, 2026. doi: 10.48550/ARXIV.2602.05842. URL https://doi.org/10.48550/arXiv.2602.05842.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *CoRR*, abs/2501.11425, 2025. doi: 10.48550/ARXIV.2501.11425. URL https://doi.org/10.48550/arXiv.2501.11425.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, Findings of ACL, pp. 3053–3077. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.181. URL https://doi.org/10.18653/v1/2024.findings-acl.181.

Ji Zeng, Dayuan Fu, Tiantian Mi, Yumin Zhuang, Yaxing Huang, Xuefeng Li, Lyumanshan Ye, Muhang Xie, Qishuo Hua, Zhen Huang, Mohan Jiang, Hanning Wang, Jifan Lin, Yang Xiao, Jie Sun, Yunze Wu, and Pengfei Liu. davinci-dev: Agent-native mid-training for software engineering. *CoRR*, abs/2601.18418, 2026. doi: 10.48550/ARXIV.2601.18418. URL https://doi.org/10.48550/arXiv.2601.18418.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. Agentevolver: Towards efficient self-evolving agent system. *CoRR*, abs/2511.10395, 2025. doi: 10.48550/ARXIV.2511.10395. URL https://doi.org/10.48550/arXiv.2511.10395.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. *CoRR*, abs/2510.08558, 2025a. doi: 10.48550/ARXIV.2510.08558. URL https://doi.org/10.48550/arXiv.2510.08558.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. RLVMR: reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *CoRR*, abs/2507.22844, 2025b. doi: 10.48550/ARXIV.2507.22844. URL https://doi.org/10.48550/arXiv.2507.22844.

A ENVIRONMENTS

A.1 ALFWORLD

ALFWorld (Shridhar et al., 2021) is a text-based embodied household environment built on ALFRED and TextWorld. In the standard human tasks, each episode starts from a household scene and a task instruction, such as finding, cleaning, heating, cooling, examining, or placing objects. At each step, the agent receives a textual observation describing the consequence of its previous actions. The environment also exposes a set of admissible commands, which are used for environment exploration by the random policy. The agent must output a command inside an `<action>` span, and the command is executed by the simulator.

We use the ALFWorld text environment for standard human task training and evaluation. Episodes terminate according to the environment success signal, and the reward returned from the environment is 1.0 for successful completion and 0.0 otherwise.

The STATE2STATE training reuses the same environment as the standard human tasks but replaces the original natural-language task instruction with a target observation as the state-reaching objective. Each task specifies a source game file for environment initialization and a target observation string. After each action, the current textual observation is compared with the target observation after normalization. The episode succeeds and terminates when the normalized observation exactly matches the normalized target. Thus, STATE2STATE provides a binary, rule-based state-matching reward without using the original ALFWorld task success verifier.

Training and evaluation are run with parallel environment workers. For grouped RL, repeated workers in the same group load the same task instance, enabling multiple rollouts from the same initial configuration.

A.2 SCIENCEWORLD

ScienceWorld (Wang et al., 2022) is a text-based interactive environment for elementary-level science tasks. Each episode presents a scientific goal that may require navigation, object inspection, object manipulation, container operations, device activation, and experiment execution. The agent receives a textual observation at each step and must issue one valid action. The action space is built from action templates, possible objects, and valid action-object combinations exposed by the simulator. In the prompt, we present the generic action templates.

In the standard ScienceWorld environment, each task is specified by a task name and variation. The simulator maintains a task score in the range from 0 to 100. In our reward design, we reward only perfect completion of the task: score 100 from the environment will receive a 1.0 for successful completion and 0.0 otherwise. In our main experiments, ScienceWorld evaluation reports success rates on sampled in-distribution and out-of-distribution task partitions.

The STATE2STATE training uses the same ScienceWorld environments but defines the objective as reaching a target observation string, without access to the original environment task instruction. Each STATE2STATE task JSON provides the original task name, task variation, and `task_target_state`. After each action, the current observation is compared against the target observation using exact string equality under normalization. If the strings match, the episode terminates with reward 1; otherwise the reward remains 0. This setting turns explored ScienceWorld observations into verifiable state-reaching objectives.

B STATE2STATE DATASETS

This section describes the construction of the STATE2STATE datasets used for environment-derived mid-training. Both datasets are generated from environment exploration rather than expert demonstrations or human-written task instructions. The resulting examples specify a reproducible environment configuration and a target observation, and success is verified by matching the current observation to the target observation.

B.1 SCIENCEWORLD DATASET

Exploration. The ScienceWorld STATE2STATE dataset is built from random exploration trajectories generated with 2560 exploration episodes. The exploration seed is 42, and each episode allows up to 300 environment steps.

The exploration policy samples from environment actions after several validity and compatibility filters. Invalid actions, non-actionable objects, and semantically incompatible action-object pairs are removed before sampling. Because ScienceWorld has comparatively large maps, exploration uses an explore-then-operate two-phase strategy: a spatial-exploration phase ending at 20% of the episode, and during the remaining 80% of steps we encourage scientific and state-changing actions.

Processing and filtering. The dataset construction process loads all exploration trajectories and collects observations from each trajectory's set of unique observations. For each observation, the pipeline records provenance information including source trajectory, source step, task name, task variation, seed, and exploration strategy. It then removes invalid or uninformative observations using a keyword blacklist and a short-text heuristic for error-like messages.

Sampling. ScienceWorld target selection follows a diversity-first procedure. The first phase ensures that each unique valid observation appears at least once. If more tasks are needed, later phases add additional instances subject to a maximum observation frequency of 5. The target number is 8,000 tasks, and the LLM-exploration variant compared in Section 5.3 uses the same number of targets. Each exported task contains a task identifier, the original ScienceWorld task name and variation, a `task_target_state` field, and metadata describing the source trajectory and step.

Statistics. The final ScienceWorld dataset contains 8,000 tasks, split into 7,872 training tasks and 128 validation tasks. All 8,000 target observations are unique, giving a diversity ratio of 1.0. The largest task-name groups are `test-conductivity` (2,900), `inclined-plane-friction-named-surfaces` (1,630), `find-living-thing` (674), `find-non-living-thing` (574), and `measure-melting-point-known-substance` (540). Target observations have median length 72 characters, 90th percentile length 500, 95th percentile length 811, and maximum length 1,276. The source-step distribution has median 116, 90th percentile 185, 99th percentile 199, and maximum 200.

B.2 ALFWORLD DATASET

Exploration. The ALFWorld STATE2STATE dataset is built from random exploration over the ALFWorld training task list. The production run uses seed 42, 256 parallel environments and 20 episode batches, yielding 5,120 trajectories. Each trajectory allows up to 500 steps. The random explorer samples from the environment-provided admissible commands. Since the distribution of each action verb can be very uneven, we adopt a verb-weighted sampling mechanism: commands are grouped by inferred verb, assigned per-verb weights, and normalized by the number of admissible commands for that verb.

Processing and filtering. Candidate target states are collected from all trajectory steps whose index is at least 10; no upper step bound is applied. Observations are filtered with substring bans for welcome banners and error-like feedback, including messages such as unknown actions, invalid actions, and impossible commands. For each candidate, the selection script infers a verb from observation text using a regular expression matching phrases of the form `You <verb>`.

Deduplication and sampling. To encourage the state-changing actions, we use different deduplication scopes for different verb categories in ALFWorld. For state-changing verbs like `clean`, `cool` and `heat`, duplicate observations are removed only within the same trajectory, preserving similar outcomes reached in different initial games. For all other verbs, duplicate observations are removed globally across trajectories. The final dataset is sampled without replacement from the candidate pool using verb-weighted sampling, with preference towards state-changing actions like heat or cool. Each exported task contains a task identifier, a source game file for environment initialization, and a `task_target_state` field.

Table 7: Summary statistics for the STATE2STATE datasets.

Dataset	Tasks	Train	Val	Traj.	Median Len.	Max Len.
ScienceWorld	8,000	7,872	128	2,560	72	1,276
ALFWorld	4,256	4,000	256	5,120	44	360

Statistics. The final ALFWorld dataset contains 4,256 tasks, split into 4,000 training tasks and 256 validation tasks. The final verb distribution is: `arrive` (733), `clean` (435), `cool` (423), `move` (423), `turn` (396), `pick` (381), `open` (361), `sliced` (320), `see` (280), `are` (206), `heat` (204), `other` (51), and `close` (43). Target observations are shorter than those in ScienceWorld: the median length is 44 characters, the 90th percentile is 104, the 95th percentile is 135, and the maximum is 360.

B.3 REPRESENTATIVE CASES

The following examples show representative STATE2STATE targets sampled from the training targets. They are chosen to cover different task families, observation types, and action effects.

ScienceWorld Case 1: `boil`; room inventory

```
This room is called the greenhouse.
In it, you see:
the agent
a substance called air
a bee hive. The bee hive door is open. In the bee hive is: nothing.
a sink, which is turned on. In the sink is: the ground, a hole (containing a
↪ substance called water).
soil
You also see:
A door to the hallway (that is open)
A door to the outside (that is open)
```

ScienceWorld Case 2: `power-component`; inventory

```
In your inventory, you see:
a blue light bulb, which is off.
its anode is connected to: a terminal 1 on agent.
its cathode is connected to: a terminal 2 on agent.
a green light bulb, which is off.
its anode is connected to: a terminal 1 on door.
its cathode is connected to: a terminal 1 on drain.
a substance called wood
```

ScienceWorld Case 3: `find-animal`; action feedback

```
You dunk the flower pot 9 in the flower pot 5, moving the liquids (water, water,
↪ water, water, water) to the flower pot 9.
```

ALFWorld Case 1: `arrive`; navigation/listing

```
You arrive at countertop 1.
On the countertop 1,
you see a bowl 2, a bread 3,
```

```
a bread 2, a bread 1,
a fork 1, a houseplant 1,
a kettle 2, a knife 2, a knife 1.
```

ALFWorld Case 2: open; container state

```
You open the fridge 1.
The fridge 1 is open.
In it, you see a apple 2, a bread 1,
a egg 2, a egg 1, a mug 1,
a plate 2, a potato 1,
a tomato 2, and a tomato 1.
```

ALFWorld Case 3: heat; state-changing action

```
You heat the potato 1
using the microwave 1.
```

C VERIFIER RELIABILITY

Because STATE2STATE rewards come from a rule-based matching function rather than a task-specific verifier, we audited how well that function agrees with human judgement in both the textual and the GUI setting.

Protocol. For each setting we randomly sampled 100 cases from training rollouts, each consisting of a target state and the state the agent actually reached. Annotators were given the same criterion the verifier uses, namely whether the two correspond to the same environment state. Three annotators participated: two labeled every case independently, and the third adjudicated the cases on which they disagreed.

Textual environments. The normalized exact-match verifier agrees with the adjudicated human label on 94% of the 100 cases. The six disagreements come mainly from ordering and minor wording differences between observations.

GUI environment. The MobileWorld verifier combines UI hierarchy structural similarity with screenshot visual similarity, so it produces a continuous score rather than a binary decision. Against human ratings, the verifier score reaches a Pearson correlation of 0.75 and a Spearman correlation of 0.76. Thresholded into a binary decision, it reaches 82% accuracy and a Cohen's κ of 0.64 against the adjudicated human label. The disagreements mainly involve dynamically changing content or sub-states that the UI hierarchy does not distinguish.

Implications for training. False positives are more harmful because they may reward a state that a human would reject. Nevertheless, the human audits show substantial agreement with both verifiers (94% for textual matching and 82% for the thresholded GUI verifier), suggesting that verifier noise is limited in practice.

D COMPUTE AND INTERACTION COST

Because STATE2STATE is proposed as a scalable mid-training stage, we report what each stage actually costs. Table 8 lists steps, rollouts, online environment interactions, wall-clock time, GPU-hours, and throughput for every run in our main experiments.

Table 8: Compute and interaction cost of each training run. All runs use one node with 8 A100-80GB GPUs. Environment interactions are the environment steps actually consumed, including the extra rollout groups that dynamic sampling collects and discards.

Environment / Model	Phase	Steps	Rollouts	Env. interactions	Wall-clock (h)	GPU-hours	Rollouts per hour	Env. steps per second
ALFWorld / Qwen3-4B	S2S	80	11,364	186,184	24.0	192.2	473.5	2.15
	Downstream RL	300	38,400	963,505	66.3	530.2	579.2	4.04
ALFWorld / Qwen3-8B	S2S	80	11,772	169,886	31.5	251.9	373.7	1.50
	Downstream RL	300	38,400	739,833	86.7	693.0	442.9	2.37
ScienceWorld / Qwen3-4B	S2S	80	11,480	184,939	35.9	287.2	319.8	1.43
	Downstream RL	400	51,200	751,588	104.9	839.2	488.1	1.99
ScienceWorld / Qwen3-8B	S2S	80	14,988	243,952	51.5	412.2	290.9	1.32
	Downstream RL	400	51,200	646,925	135.6	1084.8	377.6	1.32

Table 9: Three independent runs of the full STATE2STATE + RL pipeline for Qwen3-4B on ALFWorld. Table 1 reports the average performance.

Run	ID	OOD	Average
Run 1	90.71	93.28	92.00
Run 2	90.00	90.30	90.15
Run 3	89.29	91.79	90.54
Mean	90.00	91.79	90.90
Std.	0.71	1.49	0.98

Mid-training budget. At the checkpoints used in our reported results, STATE2STATE runs for 80 update steps and consumes 0.17M to 0.24M online environment interactions per setting, against 300 to 400 steps and 0.65M to 0.96M interactions for downstream human-task RL. Over the full two-stage pipeline, mid-training therefore accounts for 25.5% to 27.5% of the GPU-hours and 16.2% to 27.4% of the online environment interactions.

Cost of dynamic sampling. The interaction counts include the rollout groups that dynamic sampling collects and then discards, so they measure the true environment cost rather than the nominal batch size times the number of steps. With a training batch of 16 targets and a group size of 8, 80 update steps would nominally require $10{,}240$ rollouts; the measured $11{,}364$ to $14{,}988$ rollouts correspond to 1.11 to 1.46 generation batches per update on average. Downstream human-task RL does not use dynamic sampling, and its rollout counts match the nominal budget exactly.

Interpretation. These numbers do not claim that STATE2STATE is free. They show that its cost is a minority share of a pipeline that already performs downstream RL, and should be read together with the compute-aligned baseline in Table 1, which spends the same additional update budget on longer human-task RL and gains less.

E RUN-TO-RUN STABILITY

Repeating every setting across seeds was beyond our compute budget, so we repeated one representative setting, Qwen3-4B on ALFWorld, three times with different seeds. Table 9 reports the per-run results.

The mean average success rate is 90.90 ± 0.98, against 86.13 for RL and 87.26 for compute-aligned RL. The margin over both baselines is several times the run-to-run spread: $+4.77$ and $+3.64$ points respectively, versus a standard deviation of 0.98. Every individual run also exceeds both baselines, with the weakest run still 2.89 points above compute-aligned RL, so the improvement does not depend on a single favorable seed.

F REVERSE CROSS-ENVIRONMENT TRANSFER

Table 10 reports the reverse direction of the transfer experiment in Section 5.4: STATE2STATE mid-training on ALFWorld followed by downstream RL on ScienceWorld, with Qwen3-4B. ALFWorld STATE2STATE raises the average success rate from 49.63 to 51.13, mainly through a 4.50-point gain on the ID split.

Table 10: Transfer from ALFWorld mid-training to downstream RL on ScienceWorld.

Mid-training	ID	OOD	Avg.
None	51.00	**48.25**	49.63
ALFWorld S2S	**55.50**	46.75	**51.13**

G RL ALGORITHMS

This section briefly summarizes the RL algorithms used in our experiments. All of them are group-based policy optimization methods: for each task or target state, the policy samples multiple rollouts under the same initial condition, and learning is driven by relative performance within the group.

G.1 GRPO

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is the main RL backbone used in our STATE2STATE and downstream human-task training experiments. Compared with PPO, GRPO removes the learned value function and estimates the baseline from a group of sampled responses or trajectories for the same prompt. In our agent setting, for a task x, we sample G rollouts $\{\tau_i\}_{i=1}^{G}$ from the policy and obtain environment returns $\{R_i\}_{i=1}^{G}$. The group-relative advantage is computed by normalizing returns inside the group:

$$\hat{A}_i = \frac{R_i - \mathrm{mean}(\{R_j\}_{j=1}^{G})}{\mathrm{std}(\{R_j\}_{j=1}^{G})}.$$

The policy is then updated with a clipped importance-ratio objective, with the same rollout-level advantage assigned to the generated tokens/actions in that rollout. This critic-free design is well suited to sparse verifiable rewards, because successful and failed attempts on the same task provide a direct contrastive signal.

G.2 DAPO DYNAMIC SAMPLING

DAPO (Yu et al., 2025) introduces several modifications to GRPO for large-scale RL, and in this work we only use the dynamic sampling mechanism. The motivation is that a rollout group with identical outcomes gives no useful group-relative gradient: if all G rollouts fail or all G rollouts succeed, the group provides no meaningful relative comparison. Dynamic sampling therefore over-samples rollout groups and keeps only informative groups before an update. For binary verifiable rewards, this corresponds to retaining groups satisfying

$$0 < \sum_{i=1}^{G} \mathbf{1}[R_i > 0] < G.$$

In our implementation, STATE2STATE training repeatedly samples groups until enough non-identical outcome groups are collected, up to a maximum generation-batch budget. This increases the density of effective comparisons for sparse state-reaching rewards.

G.3 GIGPO

Group-in-Group Policy Optimization (GiGPO) (Feng et al., 2025) is designed for multi-turn LLM agent training, where a single terminal reward may be too coarse for assigning credit to intermediate actions. GiGPO preserves the episode-level grouping used by GRPO, but adds a second, step-level

grouping mechanism. After collecting multiple trajectories for the same task, it identifies repeated environment states across trajectories as anchor states and groups the actions taken from the same anchor state. This allows the algorithm to compare locally different actions that were chosen under the same state condition, without requiring extra per-state rollouts.

GiGPO combines an episode-level advantage $A^E(\tau_i)$ with a step-level advantage $A^S(a_t^{(i)})$:

$$A(a_t^{(i)}) = A^E(\tau_i) + \omega A^S(a_t^{(i)}),$$

where ω controls the strength of step-level credit assignment. The resulting advantage is used in a clipped policy-gradient objective similar to GRPO. In this paper, GiGPO is used as an alternative downstream RL backbone in the analysis section to test whether STATE2STATE remains beneficial when the optimizer already provides more fine-grained agent credit assignment.

H BASELINES

We include a compute-aligned RL baseline and two trajectory-based baselines that use expert-style supervision rather than online RL from STATE2STATE objectives.

H.1 COMPUTE-ALIGNED RL

RL (compute-aligned) trains the same downstream human-task RL setup for 80 additional update steps, so that its total number of policy updates matches the STATE2STATE + RL pipeline, which uses 80 mid-training steps followed by the standard downstream budget. Concretely, it runs 380 steps on ALFWorld and 480 steps on ScienceWorld, against 300 and 400 steps for RL-only. The reward, prompt, group size, learning rate, and KL coefficient are unchanged, so the only difference from RL-only is the number of updates. The alignment is in policy updates: the baseline does not receive the exploration used to build STATE2STATE targets or the extra rollouts that dynamic sampling collects (Appendix D). This baseline separates the effect of the environment-derived state-reaching objective from the effect of simply optimizing for longer.

H.2 DISTILLATION SFT

The Distillation SFT baseline collects demonstration trajectories by running a strong closed-source LLM Qwen-Plus in ALFWorld and ScienceWorld with the same environment interface used during training. At each step, the model receives the full prompt constructed by the environment manager, including the task description, current observation, recent interaction history, and admissible actions, and returns a response containing thoughts and an `<action>` span. We record the prompt and response for each active step, keep successful trajectories, and convert the resulting step-level pairs into supervised fine-tuning data format.

H.3 AGENT EARLY EXPERIENCE

Agent Early Experience (AEE) (Zhang et al., 2025a) augments expert trajectories with step-level self-reflection supervision. In our experiment, we adopt the self-reflection variant. For each expert step, AEE reconstructs the environment state by replaying the expert prefix, samples K alternative admissible actions excluding the expert action, and executes the expert and alternative actions in parallel environment copies. An LLM then observes the original situation, the expert action and next state, and an alternative action and next state, and writes a self-reflection explaining why the expert action is preferable. The final SFT example trains the model to output the reflection followed by the expert `<action>` tag. In our experiments, AEE serves as a stronger trajectory-based baseline that provides additional reasoning supervision beyond direct action imitation.

I TRAINING SETUPS AND PROMPTS

This section presents the key parameters and prompts used for training. In all settings, the agent is expected to reason and then emit an executable action inside `<action>` and `</action>` tags. Curly-braced fields are filled by the environment manager at rollout time. Our standard training prompts

mainly follow the previous work (Feng et al., 2025) with slight modifications. The STATE2STATE training prompts modify the standard ones by replacing the human instruction with the target state and adding a short description of the state-reaching objective. All evaluations, including those of standalone STATE2STATE models, use the standard prompts with the benchmark's natural-language task instruction.

I.1 KEY RL TRAINING HYPERPARAMETERS

All RL training runs, including STATE2STATE mid-training and downstream standard-task training, share most optimization settings across model sizes and phases. We adopt verl (Sheng et al., 2024) open-source RL framework and vllm (Kwon et al., 2023) inference engine for high-throughput RL training. We optimize the actor with learning rate $1e^{-6}$, enable KL loss with coefficient 0.01, and do not add KL penalties directly into the environment reward. We use GRPO-style RL algorithm in both training phases. We set the train batch size to 16 and the GRPO group size is set to 8. We deploy the environment via ray framework and enable the vllm rollout server to interact with the environments in parallel. For validation rollouts, we use sampling with temperature 0.1 for stability, and for training rollouts, we set the temperature to 1.0 for balanced exploration and exploitation.

For ALFWorld training, we use a maximum prompt length of 4096 and response length of 1024. For ScienceWorld training, we use longer contexts, with maximum prompt length between 12000 and response length 1024. STATE2STATE runs use dynamic group filtering to continually collect informative rollout groups. We set the maximum number of generation batches to 4 for ALFWorld and ScienceWorld Qwen3-8B STATE2STATE training, and to 6 for ScienceWorld Qwen3-4B training. For the STATE2STATE training, we train the model for 80 steps maximum. And for standard task training, we train the model for 400 steps in ScienceWorld and 300 steps in ALFWorld, determined by the convergence steps of the baseline training (direct RL on the environments without STATE2STATE). As for the max_steps in a single episode, we set max_steps to be 40 for ALFWorld and 30 for ScienceWorld.

I.2 STANDARD ALFWORLD TRAINING PROMPT

Standard ALFWorld Training Prompt

```
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.
Once you've finished your reasoning, you should choose an admissible action for
↪ current step and present it within <action> </action> tags.
```

I.3 ALFWORLD STATE2STATE TRAINING PROMPT

ALFWorld STATE2STATE Training Prompt

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your goal is to reach the following TARGET STATE (the environment observation
↪ text must match it):
{target_state}
You need to perform actions to make the environment observation the same as the
↪ given target state. After your each action, the environment observation will
```

```
↪ be updated. Only if the environment observation is exactly the same as the
↪ target state, the task is considered successful.

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}].

The target state is: {target_state}

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.
```

I.4 STANDARD SCIENCEWORLD TRAINING PROMPT

Standard ScienceWorld Training Prompt

```
You are an expert agent operating in the ScienceWorld environment, which is a
↪ text-based virtual environment centered around accomplishing tasks from the
↪ elementary science curriculum.

Your current 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}

Here are the actions you may take:
[
{"action": "open OBJ", "description": "open a container"},
{"action": "close OBJ", "description": "close a container"},
{"action": "activate OBJ", "description": "activate a device"},
{"action": "deactivate OBJ", "description": "deactivate a device"},
{"action": "connect OBJ to OBJ", "description": "connect electrical components"},
{"action": "disconnect OBJ to OBJ", "description": "disconnect electrical
↪ components"},
{"action": "use OBJ [on OBJ]", "description": "use a device/item"},
{"action": "look around", "description": "describe the current room"},
{"action": "look at OBJ", "description": "describe an object in detail"},
{"action": "look in OBJ", "description": "describe a container's contents"},
{"action": "read OBJ", "description": "read a note or book"},
{"action": "move OBJ to OBJ", "description": "move an object to a container"},
{"action": "pick up OBJ", "description": "move an object to the inventory"},
{"action": "put down OBJ", "description": "drop an inventory item"},
{"action": "pour OBJ into OBJ", "description": "pour a liquid into a container"},
{"action": "dunk OBJ into OBJ", "description": "dunk a container into a liquid"},
{"action": "mix OBJ", "description": "chemically mix a container"},
{"action": "go to LOC", "description": "move to a new location"},
{"action": "eat OBJ", "description": "eat a food"},
{"action": "flush OBJ", "description": "flush a toilet"},
{"action": "focus on OBJ", "description": "signal intent on a task object"},
{"action": "wait", "description": "take no action for 10 iterations"},
{"action": "wait1", "description": "take no action for 1 iteration"},
```

```
{"action": "task", "description": "describe current task"},
{"action": "inventory", "description": "list your inventory"}
]

Current available actions:
{available_actions}

Tips:
1. You can use look around to see the surroundings of the current location.
2. All containers in the environment have already been opened, you can directly
↪ get items from the containers.
3. You can explore the environment and find the items you need to complete the
↪ experiment.
**4. Your thinking process should be concise, do not waste time on irrelevant
↪ details.**

Your current task is: {task_description}

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

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

I.5 SCIENCEWORLD STATE2STATE TRAINING PROMPT

ScienceWorld STATE2STATE Training Prompt

```
You are an expert agent operating in the ScienceWorld environment, which is a
↪ text-based virtual environment centered around accomplishing tasks from the
↪ elementary science curriculum.
Your task is to reach a target state(the environment observation). If the target
↪ state is reached, the task is considered successful. If the task is not
↪ completed within a certain number of steps, the task is considered failed.

The target state is: {target_state}

You need to perform actions to make the environment observation the same as the
↪ given target state. After your each action, the environment observation will
↪ be updated. Only if the environment observation is exactly the same as the
↪ target state, the task is considered successful.

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}

Here are the actions you may take:
[
{"action": "open OBJ", "description": "open a container"},
{"action": "close OBJ", "description": "close a container"},
{"action": "activate OBJ", "description": "activate a device"},
{"action": "deactivate OBJ", "description": "deactivate a device"},
{"action": "connect OBJ to OBJ", "description": "connect electrical components"},
{"action": "disconnect OBJ to OBJ", "description": "disconnect electrical
↪ components"},
{"action": "use OBJ [on OBJ]", "description": "use a device/item"},
```

```
{"action": "look around", "description": "describe the current room"},
{"action": "look at OBJ", "description": "describe an object in detail"},
{"action": "look in OBJ", "description": "describe a container's contents"},
{"action": "read OBJ", "description": "read a note or book"},
{"action": "move OBJ to OBJ", "description": "move an object to a container"},
{"action": "pick up OBJ", "description": "move an object to the inventory"},
{"action": "put down OBJ", "description": "drop an inventory item"},
{"action": "pour OBJ into OBJ", "description": "pour a liquid into a container"},
{"action": "dunk OBJ into OBJ", "description": "dunk a container into a liquid"},
{"action": "mix OBJ", "description": "chemically mix a container"},
{"action": "go to LOC", "description": "move to a new location"},
{"action": "eat OBJ", "description": "eat a food"},
{"action": "flush OBJ", "description": "flush a toilet"},
{"action": "focus on OBJ", "description": "signal intent on a task object"},
{"action": "wait", "description": "take no action for 10 iterations"},
{"action": "wait1", "description": "take no action for 1 iteration"},
{"action": "inventory", "description": "list your inventory"}
]

Current available actions:
{available_actions}

Tips:
1. You can use look around to see the surroundings of the current location.
2. All containers in the environment have already been opened, you can directly
↪ get items from the containers.
3. You can explore the environment and find the items you need to complete the
↪ experiment.
**4. Your thinking process should be concise, do not waste time on irrelevant
↪ details.**

The target state is: {target_state}

Now it's your turn to take an action. You should first reason step-by-step about
↪ the current situation(Your thinking process should be concise and focus on
↪ the key information). After you've finished your reasoning, you should choose
↪ an appropriate action for the current step and present it within <action>
↪ </action> tags.
```

J MOBILE GUI EXPERIMENTS

This section provides details for the mobile GUI experiments reported in Table 2, first on MobileWorld and then on MobileGym. In both environments, evaluation uses the benchmark's standard natural-language instruction prompt, and MobileWorld evaluation uses the leaderboard's 50-step budget. The reference rows in Table 2 are taken from the official leaderboards at `https://tongyi-mai.github.io/MobileWorld/` and `https://mobilegym.dev/`, retrieved in September 2026.

J.1 MOBILEWORLD: ENVIRONMENT AND EVALUATION

MobileWorld (Kong et al., 2025) is a mobile GUI benchmark that requires agents to interact with Android applications through visual observations and executable UI actions. Compared with the text-only environments used in our main experiments, MobileWorld introduces richer visual observations, cross-application interaction patterns, and longer-horizon GUI navigation. For our evaluation, we use the GUI-only subset, comprising 117 instances which focus exclusively on GUI operation and omit user interaction and MCP tasks.

We use MAI-UI-8B (Kong et al., 2025) as the base model due to its competent GUI operation capability among open-source models, which provides a good foundation for conducting STATE2STATE training. Given that MobileWorld contains only evaluation tasks and lacks a separate training set, while constructing additional human-annotated training tasks with reliable verifiers would introduce

substantial manual costs, we restrict our methodology to STATE2STATE training in this environment rather than performing downstream RL on human-specified tasks. This setup effectively positions MobileWorld as a stress test to evaluate whether STATE2STATE can yield robust environment-learning signals without relying on human-authored training tasks.

J.2 MOBILEWORLD: STATE2STATE DATA

We adopt a rule-based exploration strategy within MobileWorld. At each step, the strategy randomly selects a clickable XML element, effectively executing a randomized depth-first search (DFS) through the application states. Each collected state comprises a target screenshot and its corresponding UI hierarchy XML, which jointly serve as the reward signal for STATE2STATE training. In total, we collected 365 target states across 14 applications. These were partitioned into a training set of 313 states and a validation set of 52 states, providing a robust dataset for learning from the MobileWorld environment.

J.3 MOBILEWORLD: REWARD AND TRAINING SETUP

During the training phase, the model receives a target screenshot within its task instruction. At each subsequent interaction step, it operates purely on visual inputs, observing only the current screen image. The underlying UI hierarchy XML remains strictly hidden from the policy and is reserved solely for reward computation. In our training, both target screenshots and online screen observations are resized to 540×1200 to keep the visual context length manageable during multi-turn rollouts. In evaluation, we use the original image size for standard evaluation settings.

To determine whether the final GUI state matches the target state, the STATE2STATE reward combines structural and visual similarity signals. For the structural component, we normalize the XML by removing position- and instance-specific attributes, converting it into a compact semantic sequence composed of resource identifiers, text, content descriptions, and boolean states (e.g., selected or checked). We then calculate a fuzzy sequence similarity score between the current and target XML sequences. For the visual component, we directly compute SSIM-based similarity score between two screenshots. The final reward is computed as a weighted combination, heavily favoring the structural signal with an XML similarity weight of 0.8.

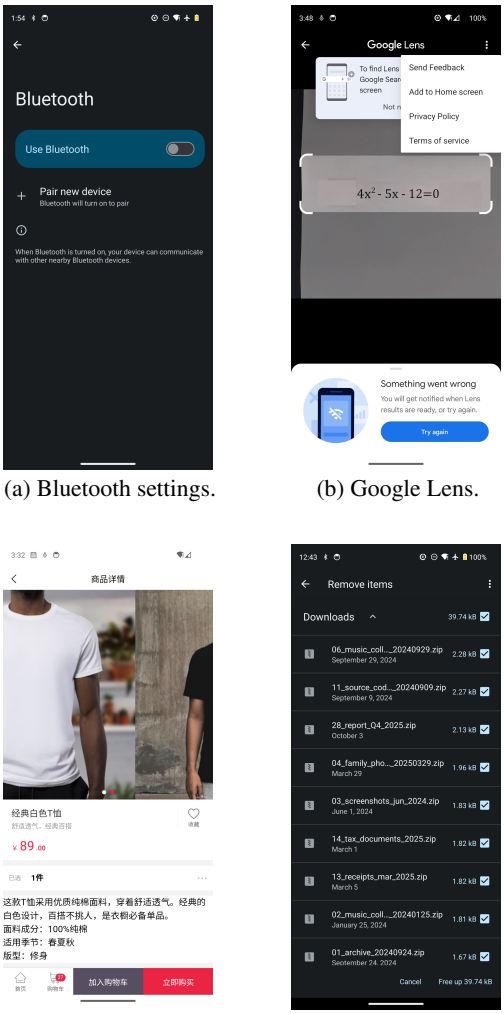
We optimize the policy with GRPO, sampling 6 rollouts per target state with a base batch size of 4 target states. This process is distributed across 24 parallel rollout executions due to computational resource constraints. To enhance training stability, we accumulate gradients over two steps, yielding an effective training batch size of 8 target states. Additionally, we cap the maximum rollout length at 15 interaction steps to prevent excessive sequence lengths and mitigate memory overhead during training; an episode ends when the agent stops or reaches this cap. We set the actor learning rate to 10^{-6}, the KL loss coefficient to 0.001, and rollout temperature to 1.0 with top-$p = 0.9$. The reported STATE2STATE model checkpoint is trained for 80 update steps and evaluated on downstream tasks directly, without an additional MobileWorld human-task RL stage.

J.4 MOBILEWORLD: REPRESENTATIVE TARGET STATES

Figure 4 shows four representative MobileWorld STATE2STATE targets. These examples illustrate the type of state-reaching objective used during training: the agent is not given a human-written task instruction, but must interact with the phone until the current screen matches the target state. The targets include system settings, camera-based search, shopping, and file-management states, covering both built-in Android interfaces and application-specific UI modes. Together, they show that the environment-derived targets span diverse visual layouts, interaction contexts, and application domains.

J.5 MOBILEGYM

MobileGym (Wu et al., 2026a) is a verifiable, highly parallel simulation platform for mobile GUI agents. We evaluate on its 256 test tasks and report success rate. We use Qwen3-VL-4B (Team, 2025c) as the base model, a general-purpose vision-language model rather than a GUI-specialized agent, and train it with STATE2STATE only, without downstream RL on human-specified tasks.



(a) Bluetooth settings. (b) Google Lens.

(c) Shopping product page. (d) File cleanup selection.

Figure 4: Representative MobileWorld STATE2STATE target states used for environment-derived training. The model observes a target screenshot and learns to reach a matching GUI state through multi-turn interaction. The examples cover system settings, visual search, shopping, and file-management interfaces.

We train on 1,200 targets, 60% collected by self-exploration and 40% by random exploration (Section 3.2). Self-exploration rolls out the base Qwen3-VL-4B before any training and without any task instruction, mainly to extend exploration depth. No stronger teacher model is used at any stage: Qwen3-VL-4B is the only model involved. The reward uses the same form of state-matching function as MobileWorld with different weights: screenshot similarity (0.6) and UI-hierarchy text similarity (0.4). We optimize the policy with GRPO using a batch size of 12 target states, 8 rollouts per target, and an actor learning rate of 10^{-6}, and an episode ends when the agent stops. The reported checkpoint is trained for 100 update steps and evaluated on the test tasks directly. Our evaluation of the base Qwen3-VL-4B gives 10.16, slightly above the 9.4 on the leaderboard; we report our own numbers for both the base model and STATE2STATE, so the gain is measured against the higher base score.