ENVACE: INTERNALIZING ENVIRONMENT DYNAMICS VIA WORLD REHEARSAL FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training large language model agents for long-horizon tool use typically relies on interactions with real or synthesized executable environments, whose construction and verification are costly, or on external simulators that are difficult to ground. We introduce EnvACE, an agentic reinforcement learning method that replaces external environment interaction during training with *world rehearsal*. The policy alternates between acting and rehearsal: it first generates a tool call, then plays the role of the environment to produce the response induced by that action, and conditions subsequent decisions on the rehearsed response. Both roles are jointly optimized end-to-end using task-success rewards. Through world rehearsal, the policy internalizes the relationship between actions and their environment responses in its parameters, yielding an agent world model that directly supports decision making. Across BFCL-v4, τ^2-Bench, VitaBench, and FinMCP-Bench, EnvACE achieves strong and transferable performance, outperforming environment-scaling baselines in the overall evaluation. Controlled studies further show that world rehearsal consistently improves policy learning across model scales. At test time, the internalized world model enables private rehearsal before committed execution, yielding further gains under a moderate rehearsal budget without additional external interaction. Our findings establish world rehearsal as a new path toward scaling LLM agent training beyond the constraints of external environments. Our code is available at `https://anonymous.4open.science/r/envace-anon-262A`.

1 INTRODUCTION

Large language models (LLMs) are increasingly expected to serve as agents in real-world applications, interacting with users to gather information and invoking tools to query or update the state of a specific environment (Z.ai, 2026; OpenAI, 2025; Team et al., 2026). Success in such settings requires effectively combining dialogue and tool use, adapting actions based on environment feedback, and operating over long interaction horizons. To develop such ability, we expect to expose LLMs to a sufficiently diverse range of tools and user requests while providing effective optimization signals.

To expose LLMs to rich and diverse tool-interactive environments, early work collected trajectories of tool use and user interaction for supervised fine-tuning (Zeng et al., 2024; Liu et al., 2025; Qin et al., 2024). Following the success of reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024; Guo et al., 2025a), recent work has automatically synthesized executable environments in which agents generate on-policy trajectories through interaction and learn from verifiable task rewards (Dong et al., 2026; Tu et al., 2026), To provide more informative training signals, some methods additionally train the policy to predict the observations resulting from its own actions as an auxiliary objective (Lu et al., 2026a; Guo et al., 2025b). However, scaling such environments relies on costly and complex synthesis pipelines, while their correctness becomes harder to verify as complexity grows. LLM-based simulators provide a less costly and more flexible alternative by directly generating environment responses, including tool outputs and user feedback, without requiring a fully executable environment (Li et al., 2025b; Ruan et al., 2024). However, their responses may be inaccurate or inconsistent, and grounding them still requires supervision from real environments.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

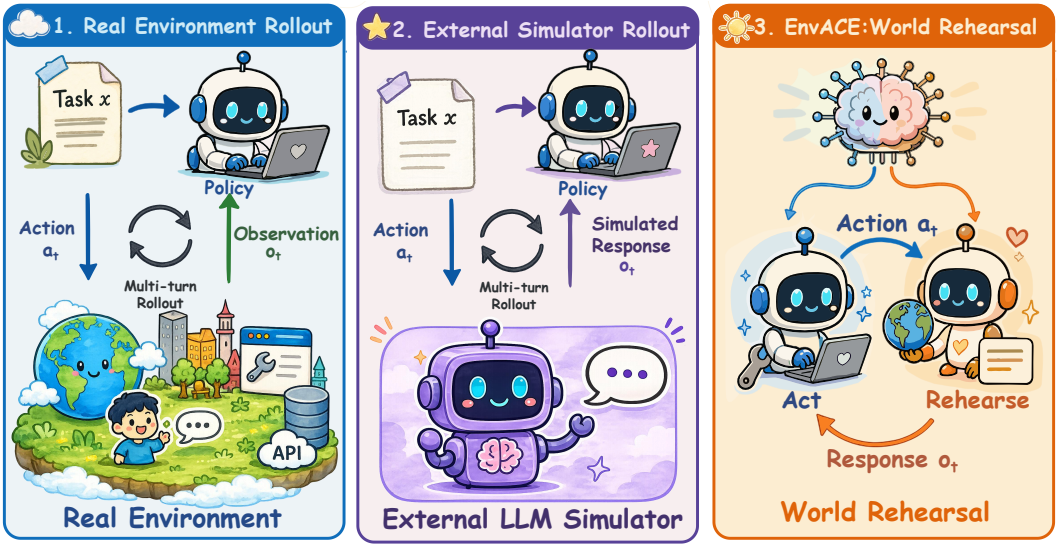

Figure 1: Comparison of three agent rollout paradigms: real-environment rollout, external-simulator rollout, and EnvACE world rehearsal. Existing approaches obtain environment responses from external sources, whereas EnvACE internalizes environment responses within the policy through world rehearsal.

Consequently, existing approaches remain fundamentally dependent on access to real environments, either for generating executable interactions or for grounding and validating simulated ones.

As illustrated in Figure 1, existing training paradigms obtain the responses needed to continue a rollout from either an external environment or a separate simulator. In both cases, the policy learns to act on externally supplied responses, leaving environment modeling outside the acting policy. We argue that an effective agent policy should not only act to accomplish the task but also model how the environment responds to its actions. Inspired by the success of learned world models (Zuo et al., 2026; Chen et al., 2025), we realize this idea by assigning the policy two roles: an actor that interacts with the environment and an environment that provides feedback to the actor, without external real-environment interaction. Rather than merely consuming or predicting externally supplied observations, the policy itself enacts the environment's responses, thereby internalizing the relationship between actions and their induced responses into its own parameters.

Toward this end, we propose EnvACE, an agentic RL method that trains a single policy to both act and rehearse its environment. At each turn, the policy first issues a tool call, then rehearses the environment response that the tool call would induce, and conditions its next action on this rehearsal. A trajectory is therefore no longer a dialogue with an external environment, but a process unfolded by the policy itself. The acting and rehearsal behaviors are jointly optimized end-to-end using a task-success reward. We call this process world rehearsal: the world modeling capability that prior work assigns to a separate simulator is absorbed into the policy and executed inline, enabling the policy to internalize how its actions shape environment responses while acting and world modeling unfold as a unified process.

Extensive experiments across a wide range of agentic benchmarks, including BFCL-v4, τ^2-Bench, VitaBench, and FinMCP-Bench, show that EnvACE consistently outperforms strong baselines. Through the joint optimization of acting and rehearsal, EnvACE captures the relationship between actions and the environment responses they induce, internalizing these dynamics within the policy to support more generalizable decision making. At test time, this internalized world model allows the policy to rehearse candidate actions before real execution, anticipate their outcomes, avoid costly mistakes, and gather useful experience without additional interaction with the external environment. Our experiments further examine how performance varies with the number of rehearsal attempts and show that a moderate rehearsal budget improves test-time performance.

Overall, our contributions can be concluded as follows:

- We introduce **world rehearsal**, in which a policy plays the role of the environment by generating the response induced by an agent action, allowing environment dynamics to be internalized without querying an external environment.
- We propose **EnvACE**, an agentic RL method that interleaves acting with world rehearsal in self-unfolded trajectories. EnvACE uses a shared policy for both roles and jointly optimizes their behaviors end-to-end using task-success rewards.
- We evaluate EnvACE across BFCL-v4, τ^2-Bench, VitaBench, and FinMCP-Bench, where it consistently outperforms strong baselines. Controlled analyses further demonstrate the benefits of jointly learning acting and rehearsal across model scales. At test time, world rehearsal provides additional performance gains.

2 Related Work

2.1 Agentic Reinforcement Learning

Increasingly realistic benchmarks have been developed to evaluate language agents through long-horizon interactions with real-world environments (Xie et al., 2024; Jimenez et al., 2024; Chen et al., 2026b). Reinforcement learning has substantially improved language-model reasoning, with recent work further advancing its efficiency and extending it to search and tool-augmented reasoning (Guo et al., 2025a; Yao et al., 2026; Jin et al., 2025; Feng et al., 2025a; Li et al., 2025a). Building on interfaces that interleave reasoning, actions, and observations (Yao et al., 2022), agentic RL extends outcome optimization across web navigation, multi-turn tool use, GUI interaction, and software engineering (Qi et al., 2025; Singh et al., 2025; Lu et al., 2026b; Wei et al., 2025; Chen et al., 2026a). The resulting long-horizon trajectories have also motivated work on training stability, credit assignment, and role-specific advantage estimation (Feng et al., 2025b; 2026; Lu et al., 2026c).

2.2 Environment Modeling for LLM Agents

Interactive agent training requires an environment that maps actions to responses and task outcomes. Fixed executable environments provide grounded transitions but are costly to construct and scale (Lu et al., 2025; Yao et al., 2024). Recent work instead synthesizes executable tools, databases, tasks, and evaluators (Song et al., 2026; Wang et al., 2026b; Tu et al., 2026; Dong et al., 2026; Xu et al., 2026), or uses language models to simulate environment feedback (Li et al., 2025b; Xiao et al., 2026). These approaches expand the available interaction experience while keeping environment dynamics external to the policy. Learned world models further support agents through lookahead and planning (Chae et al., 2025; Zeng et al., 2026; Liu et al., 2026), or by incorporating environment-modeling objectives into agent training (Yu et al., 2026; Lu et al., 2026a; Wang et al., 2026a; Cai et al., 2026). Whereas these signals generally serve as auxiliary or planning signals, EnvACE directly optimizes rehearsed environment responses as part of its on-policy trajectories.

3 Preliminaries

We formulate a tool-interactive task as a finite-horizon partially observable Markov decision process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, \mathcal{R})$. Here, $\mathcal{S}$ is the space of environment states; $\mathcal{A}$ is the action space, including structured tool calls and final answers; $\mathcal{O}$ is the observation space, including tool outputs and user responses; P specifies the environment dynamics that govern state transitions and the observation returned after an action; and $\mathcal{R}$ is the reward function that evaluates the task outcome of a completed trajectory.

Each task instance specifies an instruction x and an available tool set $\mathcal{T}$. At step t, the policy observes the interaction history h_t and generates an action $a_t \in \mathcal{A}$. For a tool call, the environment returns an observation $o_t \in \mathcal{O}$. Conditioning the response dynamics induced by P on the observable interaction history gives

$$a_t \sim \pi_\theta(\cdot \mid h_t), \qquad o_t \sim P(\cdot \mid h_t, a_t). \tag{1}$$

The history is updated as $h_{t+1} = h_t \oplus (a_t, o_t)$, and a rollout terminates with trajectory $\tau = (x, a_1, o_1, \ldots, a_T, o_T)$.

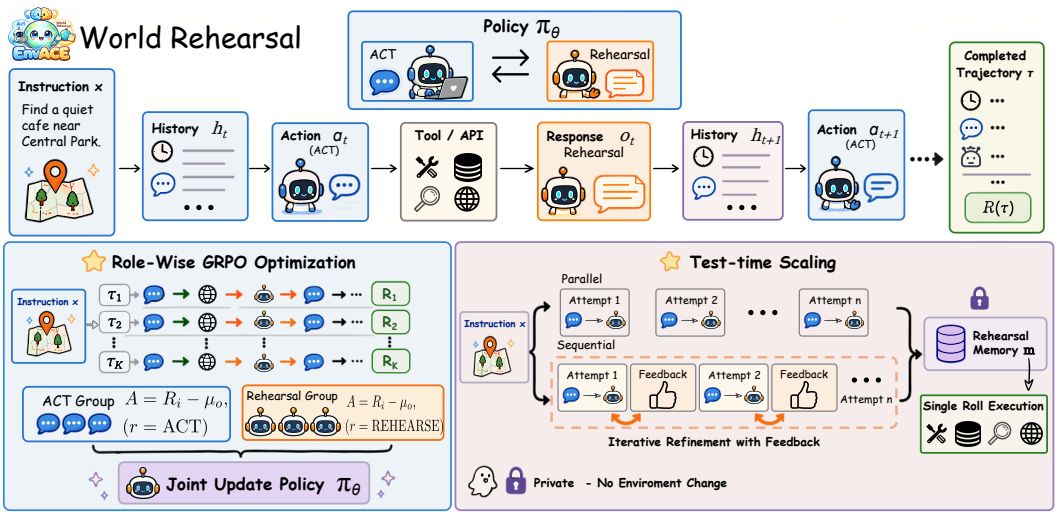

Figure 2: **Overview of EnvACE and world rehearsal.** Unlike conventional agentic RL, where an external environment provides observations after each action, EnvACE internalizes the interaction loop into a single policy. At each turn, the policy first produces an action, then rehearses the corresponding environment response, and conditions its next decision on the self-generated observation. This unified act–rehearse process enables the policy to learn environment dynamics during training and to simulate multiple candidate actions at test time before executing the most promising one in the real environment.

Upon termination, the trajectory receives a scalar reward $R(\tau)$ that evaluates task performance. The policy is optimized to maximize

$$\max_{\theta} \mathcal{J}(\theta) = \mathbb{E}_{\tau \sim (\pi_\theta, P)}[R(\tau)]. \tag{2}$$

We optimize this objective with Group Relative Policy Optimization (GRPO) (Guo et al., 2025a), which estimates advantages by normalizing each trajectory reward against other rollouts sampled for the same instruction.

This formulation exposes the conventional interaction boundary: the policy produces actions, whereas the environment supplies the observations that condition subsequent decisions. EnvACE revises this boundary by assigning observation generation itself to a rehearsal role of the policy.

4 Method

Figure 2 presents an overview of EnvACE, which comprises world rehearsal, role-wise GRPO optimization, and test-time scaling. During training, the policy alternates between acting and rehearsal, appending each generated environment response to the interaction history so that the trajectory unfolds without an external environment. Role-wise GRPO uses separate baselines for the two roles while jointly updating the policy, enabling it to internalize how actions shape environment responses. At test time, the policy performs private rehearsals in parallel or sequentially, summarizes them into a rehearsal memory, and uses this memory to guide a committed execution in the external environment.

4.1 World Rehearsal

A conventional agent rollout can proceed only when an external environment or simulator provides a response after each action. Consequently, the scalability and quality of policy learning remain tied to an external response provider, while the ability to model environment responses remains outside the acting policy. World rehearsal changes this division of labor by making environment response generation an explicit role of the policy itself. This allows the policy to unfold its own training trajectories without querying an external environment.

Concretely, EnvACE assigns two roles to a shared policy π_θ: an acting role and a rehearsal role. Given the interaction history h_t, the acting role first generates an environment-facing action:

$$a_t \sim \pi_\theta(\cdot \mid h_t, \textsc{Act}). \tag{3}$$

Conditioned on the history and the generated action, the rehearsal role then generates the corresponding environment response:

$$\hat{o}_t \sim \pi_\theta(\cdot \mid h_t, a_t, \textsc{Rehearse}). \tag{4}$$

The generated response is appended to the interaction history,

$$h_{t+1} = h_t \oplus (a_t, \hat{o}_t), \tag{5}$$

from which the acting role makes its next decision. Acting and rehearsal alternate in this manner until termination, so subsequent actions are conditioned on the environment responses generated by the policy itself. Through repeated rehearsal, the relationship between actions and their environment responses is absorbed into θ, enabling the policy to internalize environment dynamics as an agent world model.

4.2 Role-Wise GRPO Optimization

For each instruction x, EnvACE samples a group of K rollouts $\{\tau_i\}_{i=1}^{K}$, each receiving a trajectory-level reward $R_i = R(\tau_i)$. Depending on the task, this reward can be provided by either a verifiable outcome evaluator or a checklist-based LLM judge. Let $\mathcal{Y}_i = \{y_{i,m}\}_{m=1}^{M_i}$ denote all policy outputs generated in rollout τ_i, where each output is associated with a role $r_{i,m} \in \{\textsc{Act}, \textsc{Rehearse}\}$. Every policy output in the same rollout inherits its trajectory reward R_i.

For each role r, we collect all policy outputs generated under that role across the K rollouts:

$$\mathcal{G}_{x,r} = \{y_{i,m} \mid i = 1, \ldots, K,\ r_{i,m} = r\}. \tag{6}$$

EnvACE computes a separate reward baseline for each role and defines the role-wise advantage $A_{i,m}$ of output $y_{i,m}$ relative to this baseline:

$$\mu_{x,r} = \frac{1}{|\mathcal{G}_{x,r}|} \sum_{y_{j,n} \in \mathcal{G}_{x,r}} R_j, \qquad A_{i,m} = R_i - \mu_{x,r_{i,m}}. \tag{7}$$

Thus, outputs from the same trajectory receive the same reward, while their advantages are computed relative to other outputs generated under the same role.

The shared policy is optimized using the clipped GRPO objective:

$$\max_\theta\ \mathcal{J}(\theta) = \mathbb{E}_{x,i,m,\ell}\left[\min\left(\rho_{i,m,\ell}(\theta)A_{i,m}, \mathrm{clip}\left(\rho_{i,m,\ell}(\theta), 1-\epsilon, 1+\epsilon\right) A_{i,m}\right)\right], \tag{8}$$

where $\rho_{i,m,\ell}(\theta)$ is the standard GRPO likelihood ratio for the ℓ-th token of $y_{i,m}$. Although the reward baselines are computed separately for the two roles, outputs from both roles jointly update the shared policy parameters θ.

4.3 Test-Time Scaling

After training, EnvACE uses world rehearsal to scale inference-time computation before interacting with the external environment. Let Π_θ denote the rollout distribution induced by the act–rehearse loop. Given a new instruction x, EnvACE performs N private rehearsal attempts, each starting from the same initial task context. An attempt produces an imagined trajectory $\tilde{\tau}^{(n)}$, which the policy then evaluates to obtain feedback $f^{(n)}$ comprising an assessment and a suggestion for revision.

We consider two modes that differ in how information is shared across rehearsal attempts:

$$\begin{aligned} \textsc{Parallel}: &\quad \tilde{\tau}^{(n)} \sim \Pi_\theta(\cdot \mid x), \\ \textsc{Sequential}: &\quad \tilde{\tau}^{(n)} \sim \Pi_\theta\left(\cdot \mid x, \{(\tilde{\tau}^{(j)}, f^{(j)})\}_{j<n}\right). \end{aligned} \tag{9}$$

In the parallel mode, all attempts are generated independently from the same context; their feedback is retained for final aggregation but is not exposed to other attempts. In the sequential mode, each

new attempt observes the previous rehearsal trajectories together with their assessments and revision suggestions, allowing it to refine earlier decisions and avoid repeated failures.

After completing the rehearsal attempts, EnvACE summarizes all trajectories and self-evaluations into a compact rehearsal memory m_x. The acting role conditions on m_x during a single committed execution in the external environment. The rehearsals themselves remain private and do not alter the external environment.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks We evaluate EnvACE on four complementary agentic benchmarks: BFCL-v4 (Patil et al., 2025), τ^2-Bench (Barres et al., 2025), VitaBench (He et al., 2025), and FinMCP-Bench (Zhu et al., 2026). BFCL-v4 evaluates function calling and tool use across single-turn, multi-turn, and agentic tasks. τ^2-Bench and VitaBench evaluate agents in stateful, realistic service environments: the former covers Retail, Telecom, and Airline, while the latter includes food delivery, in-store consumption, online travel, and cross-domain scenarios. FinMCP-Bench evaluates financial agents on real-world tool-use tasks through the Model Context Protocol (MCP). We follow the official evaluation protocol of each benchmark.

Baselines We compare EnvACE against Qwen3 models at different scales (1.7B, 4B, and 8B), as well as Qwen3 models trained with standard GRPO. We also include representative tool-use training methods based on environment simulation or synthesis. Simulator-8B (Li et al., 2025b) uses reasoning models to simulate environment feedback, while TOUCAN-7B (Xu et al., 2025) synthesizes large-scale trajectories from real-world MCP environments. EnvScaler-8B (Song et al., 2026) programmatically synthesizes tool-interactive environments, AWM-8B/14B (Wang et al., 2026b) builds code-driven environments with database-backed states, and ScaleEnv-8B (Tu et al., 2026) constructs interactive environments and verifiable tasks from scratch.

Implementation Details We conduct experiments with Qwen3 backbones at three model scales: 1.7B, 4B, and 8B (Yang et al., 2025). All models are trained on the dataset introduced by CM2 (Zhang et al., 2026). We optimize the model for 470 training steps with a learning rate of 1×10^{-6}, a batch size of 16, and four rollouts per prompt. The KL coefficient is set to 1×10^{-4}, and the entropy coefficient is set to 0.0. At each training step, we sample 64 instances. The maximum input and response lengths are set to 12,000 and 8,000 tokens, respectively, and each agent trajectory is allowed up to 30 interaction turns. During training, we use Qwen3-30B-A3B as the LLM judge. For non-TTS experiments, we report Avg@4 results averaged over four independent runs. Due to the high computational cost of TTS experiments, we report results from a single run only. For the experiments in Table 3, both the acting role and the rehearsal role use a sampling temperature of 1.0 and a top-p of 1.0 to encourage reasoning diversity and broader exploration of the action space. For the remaining roles, we use a lower temperature of 0.01 to improve robustness and reproducibility. For all other experiments outside Table 3, the acting role uses a sampling temperature of 0.6 and a top-p of 0.95. Training is implemented with the verl framework and conducted on 16 NVIDIA H20 GPUs.

5.2 MAIN RESULTS

Table 1 presents the main results across BFCL V4, τ^2-Bench, and VitaBench. EnvACE achieves an Overall score of 32.91%, outperforming all environment-scaling baselines with complete results on the three benchmarks. It surpasses EnvScaler-8B and AWM-14B by 0.99% and 0.37%, respectively. This strong overall result reflects consistent performance across heterogeneous environments. On BFCL V4, EnvACE reaches 46.04%, outperforming Qwen3-8B and AWM-8B by 2.00% and 1.75%, respectively, while remaining within 1.03% of EnvScaler-8B. On τ^2-Bench, it obtains the second-highest average of 36.7%, exceeding EnvScaler-8B, AWM-8B, and AWM-14B by 3.8%, 5.5%, and 6.0%, respectively. On VitaBench, EnvACE achieves 16.0%, the best average among all 7B–8B methods, outperforming EnvScaler-8B and ScaleEnv-8B by 0.2% and 1.0%.

Method	BFCL V4							τ^2-Bench				VitaBench					Overall
	Web	Mem.	Multi	NoLive	Live	Irrel.	Avg.	Retail	Telecom	Airline	Avg.	Cross	Deliv.	Inst.	OTA	Avg.	
Open-Source Foundation Models (1.7B–8B)																	
Qwen3-1.7B	1.88	12.58	15.75	82.94	74.06	75.71	30.89	4.6	11.6	15.0	10.4	0.0	3.7	3.3	0.7	1.9	14.41
Qwen3-4B	8.25	20.86	37.25	88.16	81.55	78.33	41.80	32.7	13.2	38.0	27.9	1.8	13.7	18.2	4.8	9.6	26.43
Qwen3-8B	10.63	22.15	42.09	88.09	80.90	79.58	44.04	41.2	20.8	28.0	30.0	1.2	15.5	24.8	4.0	11.4	28.48
Open-Source Environment Scaling Methods (7B–14B)																	
Simulator-8B	9.38	5.65	1.47	32.46	44.30	86.54	19.78	49.3	15.1	51.0	38.5	0.0	1.2	4.8	1.0	1.8	20.03
TOUCAN-7B	14.00	15.97	22.59	76.11	72.61	76.89	35.33	30.0	10.1	27.0	22.4	0.5	6.9	3.0	0.8	2.8	20.18
EnvScaler-8B	15.38	19.52	52.25	86.40	81.70	76.03	47.07	48.0	16.2	34.5	32.9	6.0	20.8	29.5	7.0	15.8	31.92
AWM-8B	15.00	19.46	42.81	88.47	80.64	76.41	44.29	41.3	18.2	34.0	31.2	1.5	16.0	19.2	4.2	10.2	28.56
AWM-14B	14.88	21.88	49.06	89.46	81.41	81.59	47.32	41.9	19.1	31.0	30.7	8.0	24.8	33.8	11.8	19.6	32.54
ScaleEnv-8B	–	–	–	–	–	–	–	50.9	27.2	37.5	38.5	3.0	26.3	23.8	7.0	15.0	–
EnvACE-1.7B	3.75	15.70	14.38	83.13	74.76	78.14	31.81	7.9	14.5	23.5	15.3	0.0	7.5	5.0	0.3	3.2	16.77
EnvACE-4B	12.75	24.74	29.63	87.86	82.27	84.79	41.88	36.8	12.7	36.0	28.5	1.5	18.0	19.0	4.5	10.8	27.06
EnvACE-8B	12.25	24.03	45.29	87.59	81.20	83.19	46.04	48.9	17.3	44.0	36.7	6.0	24.0	27.0	7.0	16.0	32.91

Table 1: Benchmark results across BFCL V4, τ^2-Bench, and VitaBench. Overall is the arithmetic mean of the BFCL V4 Avg., τ^2-Bench Avg., and VitaBench Avg. Blue cells indicate the highest result in each column, while red cells indicate the second-highest distinct result.

Method	TR (%)	TP (%)	TF1 (%)
Qwen3-8B	43.18	39.47	41.24
AWM-8B	46.43	39.18	42.50
Simulator-8B	11.36	26.72	15.95
EnvScaler-8B	49.35	39.18	43.68
EnvACE-8B	41.23	54.04	46.78

Table 2: Performance comparison on FinMCP-Bench using TR, TP, and TF1 metrics. Values are reported as percentages (%). Blue cells indicate the highest result in each column, while red cells indicate the second-highest result.

Table 2 reports the results on FinMCP-Bench. EnvACE achieves the best TF1 score of 46.78%, outperforming EnvScaler-8B and AWM-8B by 3.10% and 4.28%, respectively. It also obtains the highest tool precision of 54.04%. Although its tool recall is not the highest, the resulting TF1 demonstrates the strongest balance between precision and recall. Taken together, these results demonstrate that world rehearsal provides a more effective and scalable training paradigm than representative methods relying on real-environment interaction or external simulators. By jointly learning to act and generate environment responses within a single policy, EnvACE acquires transferable tool-use capabilities across heterogeneous and specialized scenarios.

5.3 Effectiveness of World Rehearsal

World Rehearsal Improves Policy Learning Figure 3 provides a controlled comparison with standard GRPO at the 8B scale. On τ^2-Bench, EnvACE improves the average score from 31.2% to 36.7%, a gain of 5.5%. This improvement is particularly relevant for stateful, multi-turn interactions, where each action depends on the environment responses observed in previous turns. By jointly learning to act and rehearse these responses, the policy can better anticipate interaction dynamics and make more effective subsequent decisions.

Internalizing Environment Dynamics Figure 3 also compares EnvACE with Per-role Policy, a variant that assigns separate policies to acting and rehearsal. Both variants generate environment responses, but only EnvACE shares parameters between the two roles. On τ^2-Bench, parameter sharing improves the average score from 35.5% to 36.7%, a gain of 1.2%. Because acting and rehearsal update the same parameters, knowledge acquired during rehearsal about how the environment responds to actions is directly incorporated into the acting policy. This gain supports our hypothesis

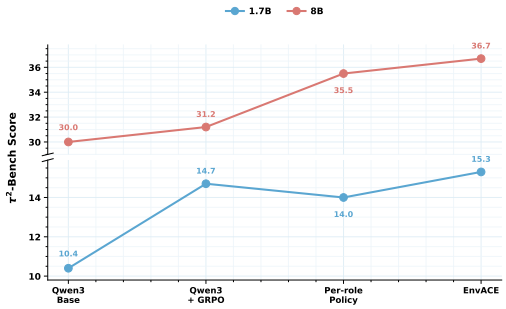

Figure 3: **Ablation results on τ^2-Bench.** EnvACE achieves the best performance at both the 1.7B and 8B scales.

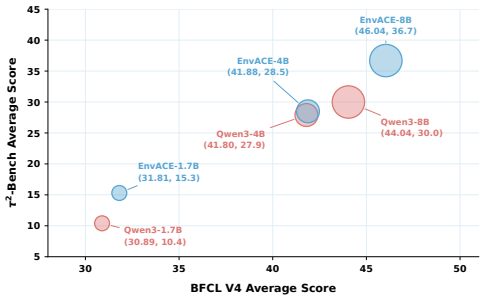

Figure 4: **EnvACE and base-model performance across model scales.** EnvACE (blue) and Qwen3 (red) at 1.7B, 4B, and 8B on both benchmarks.

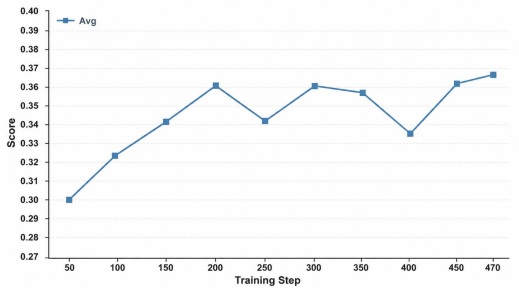

Figure 5: Evaluation performance of EnvACE-8B throughout RL training on τ^2-bench.

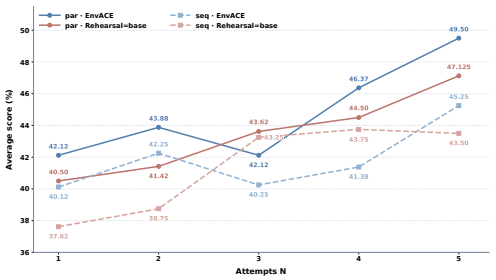

Figure 6: TTS scaling performance on BFCL Multi-Turn.

that parameter sharing enables the policy to internalize environment dynamics as an agent world model and use this knowledge for better decision making.

Training Dynamics Figure 5 further illustrates the optimization behavior of EnvACE. The offline evaluation score exhibits an overall upward trend during training, increasing from 30.0% at step 50 to 36.7% at step 470. Although performance fluctuates at intermediate checkpoints, it recovers in the later stage and reaches its highest observed score at the final checkpoint. This overall trend indicates that world rehearsal provides a sustained learning signal during RL training and progressively strengthens the policy's task-solving capability.

5.4 Effects across Model Scales

We apply the same training recipe to the 1.7B, 4B, and 8B backbones to examine whether EnvACE scales with model capacity. As shown in Figure 4, scaling from 1.7B to 8B improves the BFCL V4 average from 31.81% to 46.04%, a gain of 14.23%, and the τ^2-Bench average from 15.3% to 36.7%, a gain of 21.4%. Figure 3 further shows that EnvACE outperforms standard GRPO at both scales, with the improvement becoming more pronounced at 8B. These results demonstrate that world rehearsal remains effective as model capacity increases and can scale to stronger backbones while delivering larger performance gains.

5.5 Test-Time Scaling with World Rehearsal

World Rehearsal Improves Test-Time Performance Table 3 reports a representative comparison with the rehearsal budget fixed to $N = 2$. We evaluate both parallel and sequential modes, using either the base model or EnvACE for rehearsal. Parallel rehearsal with EnvACE achieves the best Overall score of 40.9%, improving the Non-TTS result of 36.7% by 4.2%. The improvement is consistent across both benchmarks: the average score increases from 31.4% to 38.0% on τ^2-Bench

Method	Mode	Rehearsal	τ^2-bench (%)				BFCL Multi-Turn (%)					Overall (%)
			Airline	Retail	Telecom	Avg.	Base	Miss Func.	Miss Param.	Long Ctx.	Avg.	
Non-TTS	–	–	38.0	42.3	14.0	31.4	47.5	45.5	35.0	39.5	41.9	36.7
TTS	Par.	Base	38.0	45.5	13.2	32.2	50.5	46.0	30.0	39.2	41.4	36.8
TTS	Par.	EnvACE	42.0	54.4	17.5	38.0	55.5	49.0	31.0	40.0	43.9	40.9
TTS	Seq.	Base	36.0	49.1	8.0	31.0	48.0	45.0	26.0	36.0	38.8	34.9
TTS	Seq.	EnvACE	36.0	50.9	17.5	34.8	51.5	50.5	30.0	37.0	42.3	38.5

Table 3: Test-time scaling with $N = 2$ on τ^2 and BFCL multi-turn. Overall is the arithmetic mean of the τ^2-Bench Avg. and BFCL Multi-Turn Avg. ■ Blue cells indicate the highest result in each column, while ■ red cells indicate the second-highest result.

and from 41.9% to 43.9% on BFCL Multi-Turn. Sequential rehearsal with EnvACE also improves the Overall score to 38.5%, showing that world rehearsal benefits final execution under both modes.

The policy used for rehearsal is important. In both modes, using EnvACE for rehearsal consistently outperforms using the base model. The base model yields only a marginal improvement under the parallel mode and underperforms Non-TTS under the sequential mode. This comparison suggests that the gains do not arise from additional inference compute alone, but depend on the environment-response knowledge internalized through world rehearsal. These results demonstrate that world rehearsal provides an effective mechanism for test-time scaling without additional interaction with the external environment.

Effect of Rehearsal Budget We further study the effect of the rehearsal budget on BFCL Multi-Turn performance in Figure 6. As the number of rehearsal attempts increases, the overall performance shows an upward trend. Parallel rehearsal with EnvACE exhibits stronger scaling gains, with the average score increasing from 42.12% at $N = 1$ to 49.50% at $N = 5$; sequential rehearsal also improves from 40.12% to 45.25%. Overall, rehearsal with EnvACE achieves higher performance than rehearsal with the base model, suggesting that more accurate environment rehearsal provides more useful information for the final execution. These results further show that increasing the rehearsal budget improves test-time performance, while higher-quality world rehearsal enables more effective use of additional test-time computation. Similar gains are observed on τ^2-Bench, as shown in Figure 9.

6 Conclusion

We introduced EnvACE, an agentic RL method built around world rehearsal. By assigning acting and rehearsal to a single shared policy and optimizing both roles with role-wise GRPO, EnvACE learns not only to select actions but also to generate the corresponding environment responses. Through repeated rehearsal, this knowledge is absorbed into the policy parameters, enabling the policy to internalize environment dynamics as an agent world model and use them to guide subsequent decisions. Across BFCL V4, τ^2-Bench, VitaBench, and FinMCP-Bench, EnvACE achieves strong and transferable tool-use performance, outperforming existing environment-scaling baselines in the overall evaluation. Controlled analyses further demonstrate the effectiveness of world rehearsal, the importance of sharing parameters between acting and rehearsal, and the generalization of these benefits across model scales. Finally, the resulting agent world model enables test-time scaling through private rehearsal, improving final execution without additional interaction with the external environment. Together, these results establish world rehearsal as an effective and scalable alternative to agent training based on real-environment interaction or separate external simulators.

7 Limitation

Due to computational constraints, we evaluate EnvACE only up to the 8B scale, leaving its effectiveness on larger models for future investigation. In addition, our current evaluation focuses primarily on tool-interactive tasks, and extending world rehearsal to a broader range of agentic settings remains an interesting direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools solely for writing assistance. Specifically, we used large language model based assistants to polish the language of the manuscript, including grammar correction, improving the clarity and conciseness of sentences we had already drafted, and suggesting alternative phrasings for titles and section headings.

We did not use generative AI tools for research ideation, the design of the proposed method, experimental design, data analysis, or the generation of scientific claims or results; these were carried out by the authors. No AI-generated text was incorporated into the manuscript without being reviewed, rewritten where necessary, and verified by the authors. All numbers, tables, figures, and experimental results were produced by our own experimental pipeline. Generative AI tools were not used in the review or evaluation of other researchers' work in connection with this submission, and they are not part of the proposed method's contribution beyond the language models explicitly described as trained policies and baselines in the paper.

ETHICS STATEMENT

This work trains and evaluates language model agents for tool use on publicly available models and benchmarks, and does not involve human subjects, personally identifiable information, or the release of new user data. All benchmarks we use (τ^2-Bench, VitaBench, BFCL-v4, and FinMCP-Bench) are existing public benchmarks used under their intended terms, and their simulated service environments contain synthetic rather than real customer records. Because our method lets a policy rehearse environment responses internally, a rehearsed response is a model prediction and not a verified fact about the world; we therefore emphasize that rehearsal is used for training and for pre-execution deliberation, and that any deployment in a consequential domain (for example the financial tool-use setting we evaluate) should keep real execution and verification in the loop. We see no other ethical concerns specific to this work.

REPRODUCIBILITY STATEMENT

We describe the world rehearsal formulation and the role-wise optimization objective in Section 4 with the complete training procedure, and report the datasets, backbone models, baselines, and evaluation protocol of every benchmark in Section 5.1. All benchmarks are public and we follow each benchmark's official evaluation protocol without modification, so that reported numbers are directly comparable to previously published results. Full training and evaluation hyperparameters, including the learning rate, rollout configuration, sequence lengths, and the maximum number of interaction turns, are reported in Section 5.1, together with qualitative rehearsal trajectories in Appendix A that illustrate the exact prompt format used for the actor and the environment roles. Our code, prompts, and configuration files are released at the anonymized repository linked in the abstract.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Guangfeng Cai, Kaibing Yang, Shuo He, Yu Li, Shengtian Yang, Jiaqi Lv, and Lei Feng. Beyond next-observation prediction: Agent-authored world modeling for sequential decision making. *arXiv preprint arXiv:2606.25421*, 2026.

Hyungjoo Chae, Namyoung Kim, Kai Ong, Minju Gwak, Gwanwoo Song, Jihoon Kim, Sunghwan Kim, Dongha Lee, and Jinyoung Yeo. Web agents with world models: Learning and leveraging environment dynamics in web navigation. In *International Conference on Learning Representations*, volume 2025, pp. 63707–63738, 2025.

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. *arXiv preprint arXiv:2510.15047*, 2025.

Yuxin Chen, Xiaodong Cai, Junfeng Fang, Zhuowen Han, Yu Wang, Yaorui Shi, Yi Zhang, Qi Gu, Xunliang Cai, Xiang Wang, et al. Learning to act under noise: Enhancing agent robustness via noisy environments. *arXiv preprint arXiv:2605.27209*, 2026a.

Yuxin Chen, Yi Zhang, Zhengzhou Cai, Yaorui Shi, Zhiyuan Yao, Chenhang Cui, Jingnan Zheng, Yaqi Huo, Xi Su, Qi Gu, et al. Vitabench 2.0: Evaluating personalized and proactive agents in long-term user interactions. *arXiv preprint arXiv:2605.27141*, 2026b.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025a.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 46375–46408. Curran Associates, Inc., 2025b. doi: 10.52202/085713-1544. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. mas: Stable reinforcement learning for multi-agent llm systems. *arXiv preprint arXiv:2602.08847*, 2026.

Daya Guo et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Shangmin Guo, Omar Darwiche Domingues, Raphaël Avalos, Aaron Courville, and Florian Strub. World modelling improves language model agents. *arXiv preprint arXiv:2506.02918*, 2025b.

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi Gu, Chengcheng Han, Dengchang Zhao, Hui Su, Kefeng Zhang, et al. Vitabench: Benchmarking llm agents with versatile interactive tasks in real-world applications. *arXiv preprint arXiv:2509.26490*, 2025.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025a.

Yuetai Li, Huseyin A Inan, Xiang Yue, Wei-Ning Chen, et al. Simulating environments with reasoning models for agent training. *arXiv preprint arXiv:2511.01824*, 2025b.

Weiwen Liu, Xu Huang, Xingshan Zeng, xinlong hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 41359–41381, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/663865ea167425c6c562cb0b6bcf76c7-Paper-Conference.pdf.

Youwei Liu, Jian Wang, Hanlin Wang, and Wenjie Li. Comap: Co-evolving world models and agent policies for llm agents. *arXiv preprint arXiv:2606.02372*, 2026.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, et al. Policy and world modeling co-training for language agents. *arXiv preprint arXiv:2606.02388*, 2026a.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026b.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026c.

OpenAI. Introducing gpt-5. 2025. URL https://openai.com/index/introducing-gpt-5/.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.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 *International Conference on Learning Representations*, 2025.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris Maddison, and Tatsunori Hashimoto. Identifying the risks of lm agents with an lm-emulated sandbox. In *International Conference on Learning Representations*, volume 2024, pp. 27031–27098, 2024.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis. *arXiv preprint arXiv:2601.05808*, 2026.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, et al. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, et al. Scaleenv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training. *arXiv preprint arXiv:2602.06820*, 2026.

Xucong Wang, Ziyu Ma, Shidong Yang, Tongwen Huang, Pengkun Wang, Yong Wang, and Xiangxiang Chu. Role-agent: Bootstrapping llm agents via dual-role evolution. *arXiv preprint arXiv:2606.10917*, 2026a.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026b.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, LINGMING ZHANG, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 78500–78525. Curran Associates, Inc., 2025. doi: 10.52202/085713-2629. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/7107d4d2e837bde2171c6b71b5bde954-Paper-Conference.pdf.

Zikai Xiao, Jianhong Tu, Chuhang Zou, Yuxin Zuo, Zhi Li, Peng Wang, Bowen Yu, Fei Huang, Junyang Lin, and Zuozhu Liu. Webworld: A large-scale world model for web agent training. *arXiv preprint arXiv:2602.14721*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 2024.

Minrui Xu, Zilin Wang, Mengyi Deng, Zhiwei Li, Zhicheng Yang, Xiao Zhu, Yinhong Liu, Boyu Zhu, Baiyu Huang, Chao Chen, et al. Envfactory: Scaling tool-use agents via executable environments synthesis and robust rl. *arXiv preprint arXiv:2605.18703*, 2026.

Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. Toucan: Synthesizing 1.5 m tool-agentic data from real-world mcp environments. *arXiv preprint arXiv:2510.01179*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Zhiyuan Yao, Yi-Kai Zhang, Yuxin Chen, Yueqing Sun, Zishan Xu, Yu Yang, Tianhao Hu, Qi Gu, Hui Su, and Xunliang Cai. Coba-rl: Capability-oriented budget allocation for reinforcement learning in llms. *arXiv preprint arXiv:2602.03048*, 2026.

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. *arXiv preprint arXiv:2602.05842*, 2026.

Z.ai. GLM-5.2: Built for long-horizon tasks. 2026. URL https://z.ai/blog/glm-5.2.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024.

Xingshan Zeng, Lingzhi Wang, Weiwen Liu, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, and Qun Liu. Artis: Agentic risk-aware test-time scaling via iterative simulation. *arXiv preprint arXiv:2602.01709*, 2026.

Zhen Zhang, Kaiqiang Song, Xun Wang, Yebowen Hu, Weixiang Yan, et al. Cm2: Reinforcement learning with checklist rewards for multi-turn and multi-step agentic tool use, 2026. URL https://arxiv.org/abs/2602.12268.

Jie Zhu, Yimin Tian, Boyang Li, Kehao Wu, Zhongzhi Liang, Junhui Li, Xianyin Zhang, Lifan Guo, Feng Chen, Yong Liu, et al. Finmcp-bench: Benchmarking llm agents for real-world financial tool use under the model context protocol. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 19782–19786. IEEE, 2026.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, et al. Qwen-agentworld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026.

A CASE STUDY

Figure 7 presents a representative example comparing EnvACE with EnvScaler8B and Vanilla agents. In this example, EnvACE anticipates a potential tool-call failure and repairs the invalid parameters before execution. In contrast, the baseline agents issue an invalid tool call and require additional interaction steps to recover from the resulting failure. This case illustrates how world rehearsal enables more reliable and efficient tool use.

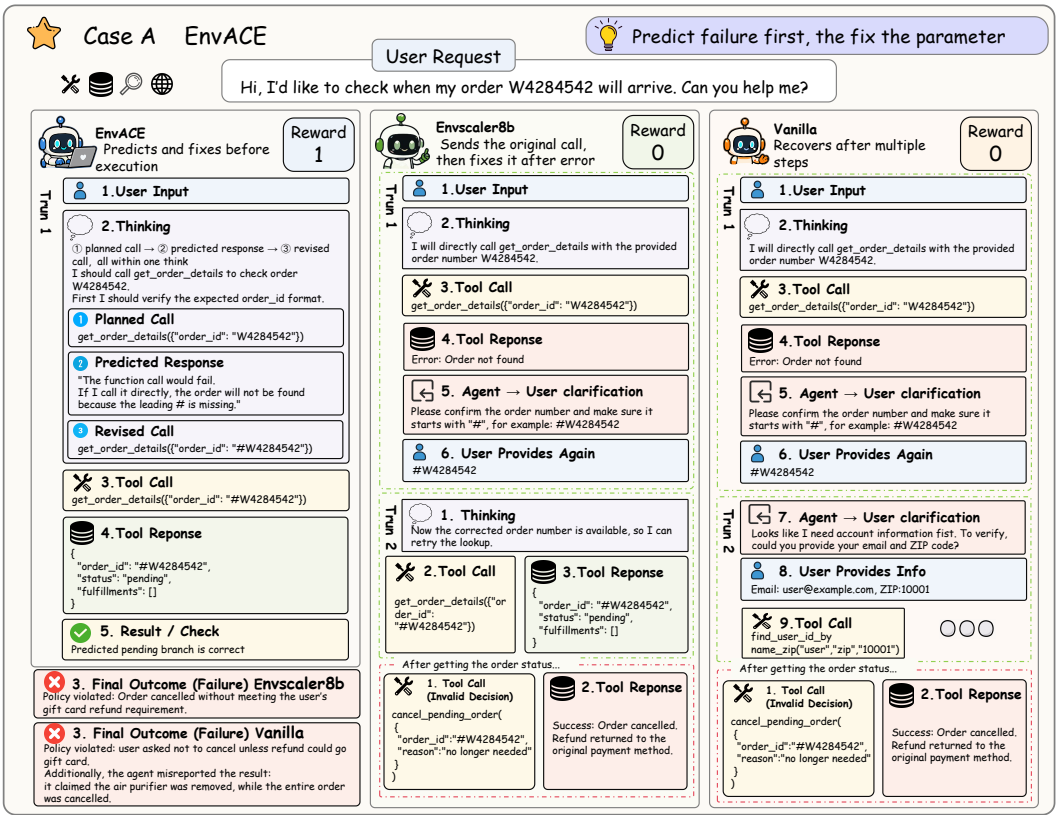

Figure 7: Case study of EnvACE compared with EnvScaler8B and Vanilla agents. EnvACE predicts potential tool-call failures before execution and automatically repairs invalid parameters, while baseline agents require additional recovery steps after failure.

As shown in Figure 8, this case illustrates how EnvACE avoids an invalid write operation through world rehearsal. The user requests to modify a flight reservation with a basic economy ticket. Although updating the reservation appears to be a reasonable next step, such a write operation is not allowed under the given environment constraints. Before execution, EnvACE performs internal world rehearsal by simulating the planned tool call and predicting its potential response. It recognizes that the required information for executing `update_reservation_flights` is incomplete and that issuing the write operation would lead to an invalid action. Therefore, EnvACE proactively replaces the unsafe action with a read-only query to obtain reservation details, avoiding any forbidden modification. In contrast, EnvScaler8B and Vanilla agents fail to anticipate the consequence of tool calls and eventually execute the invalid write operation, resulting in failure. This example demonstrates that world rehearsal enables EnvACE to anticipate tool consequences and make safer decisions before interacting with the external environment.

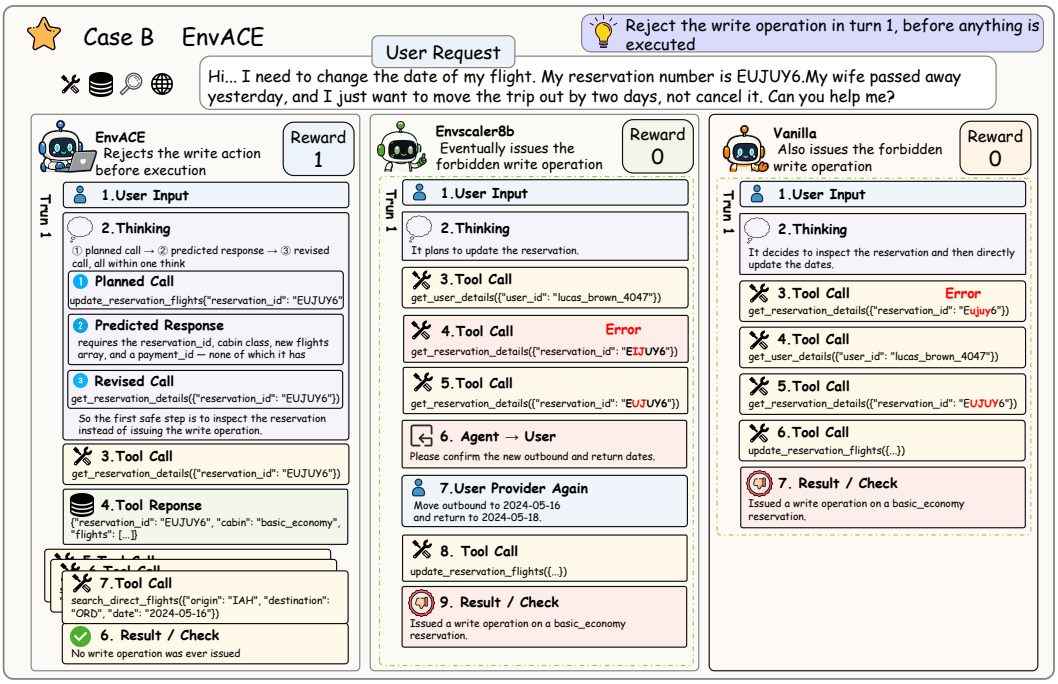

Figure 8: Case study of EnvACE compared with EnvScaler8B and Vanilla agents. EnvACE performs world rehearsal before tool execution, predicting that the planned write operation would fail and proactively replacing it with a safe read-only query. In contrast, baseline agents fail to anticipate the consequence of tool calls and eventually execute invalid write operations.

B TEST-TIME SCALING PROCEDURE

Algorithm 1 summarizes the *rehearse–evaluate–summarize* procedure used for test-time scaling. Given a task x, the policy π performs N private rehearsal attempts before interacting with the external environment. In the parallel mode, attempts are generated independently, whereas in the sequential mode, each attempt conditions on previous rehearsal trajectories and feedback for iterative refinement. All rehearsal results are summarized into a compact rehearsal memory m_x, which guides a single committed execution in the real environment. Only this final execution is officially scored. In our experiments, the acting policy is EnvACE-8B, and the rehearsal model M is either Qwen3-8B or EnvACE-8B.

Algorithm 1 Test-Time Scaling with World Rehearsal

Require: Task x, acting policy π, rehearsal model M, rehearsal budget N, mode $m \in \{\textsc{Parallel}, \textsc{Sequential}\}$
Ensure: Final trajectory $\tau^\star$ and official score r

Initialize rehearsal history $\mathcal{H} \leftarrow [\,]$
for $n = 1, \ldots, N$ **do**
  $c^{(n)} \leftarrow \begin{cases} \emptyset, & m = \textsc{Parallel}, \\ \{(\tilde{\tau}^{(j)}, f^{(j)})\}_{j<n}, & m = \textsc{Sequential} \end{cases}$
  $\tilde{\tau}^{(n)} \leftarrow \textsc{Rehearse}(\pi, M, x, c^{(n)})$
  $f^{(n)} \leftarrow \textsc{SelfEvaluate}(\pi, x, \tilde{\tau}^{(n)})$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\tilde{\tau}^{(n)}, f^{(n)})\}$
end for
$m_x \leftarrow \textsc{Summarize}(\pi, x, \mathcal{H})$
Initialize $\mathcal{E}_{\text{real}}$ from the task's initial context
$\tau^\star \leftarrow \textsc{Execute}(\pi, x, m_x, \mathcal{E}_{\text{real}})$
$r \leftarrow \textsc{OfficialEvaluate}(x, \tau^\star)$
return $(\tau^\star, r)$

C TEST-TIME SCALING ON τ^2-BENCH.

We further evaluate the effect of the rehearsal budget on τ^2-Bench in Figure 9. Performance improves overall as the number of rehearsal attempts increases under both parallel and sequential modes. With EnvACE rehearsal, the parallel score increases from 31.85% to 40.64%, while the sequential score increases from 31.80% to 38.40%. Across all evaluated budgets, EnvACE outperforms the corresponding base-model rehearsal, indicating that its internalized environment-response knowledge enables more effective test-time rehearsal.

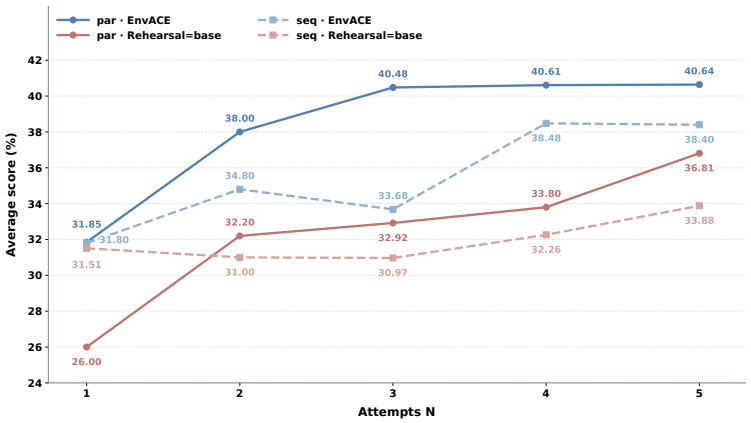

Figure 9: TTS scaling performance on τ^2-Bench.

D PROMPTS

The following prompt templates instantiate the main components of the test-time scaling procedure described in Section 4.3 and Appendix B. The world-rehearsal prompt specifies how environment responses are generated conditioned on the current interaction context and agent action. The self-evaluation prompt produces an assessment and revision feedback for each rehearsed trajectory, while the summarization prompt aggregates the resulting trajectories and feedback into a compact rehearsal memory. For sequential rehearsal, an additional history template organizes previous trajectories and feedback as context for subsequent attempts.

World Rehearsal Prompt for Agentic Tool Execution

You are a precise tool simulator.
Your task: validate the RL model's tool call against the candidate tools whitelist and the JSON schema, then generate the tool execution result.

Input Information

You will be given:

- The candidate tools whitelist (list of tool schemas this sample is allowed to call).
- Previous tool calls and results (few-shot, ground-truth reference data; may be `<unavailable>` when no prior calls exist).
- Current conversation history.
- The RL model's latest response (containing `<tool_call>...</tool_call>` tags).
- The current tool to simulate: name, JSON schema, parsed arguments.

Validation Procedure

1. Tool Call Format Check

The RL model's latest response must wrap the tool call inside `<tool_call>...</tool_call>` tags, with valid JSON:

`{"name": "...", "arguments": {...}}`

On failure (tool_call wrapper absent, or JSON malformed), output:

```
<execution_result>
{
 "error_tool_call":{
   "code":"INVALID_TOOL_CALL_FORMAT",
   "message":"<concise reason>"
 }
}
</execution_result>
```

2. Tool Whitelist Check

The tool name (shown under "Current tool to simulate") must appear in the candidate tools whitelist (either the explicit list, or the `<tools>` block inside the caller's system prompt).
On failure, output:

```
<execution_result>
{
 "error_tool_call":{
   "code":"TOOL_NOT_AVAILABLE",
   "message":"tool name '<name>' is not found. Please check your function call
     and use provided tools."
 }
}
</execution_result>
```

3. Argument Schema Check
Arguments must be a JSON object satisfying the input schema:

- Required fields must be present.
- Primitive types must match.
- Enum values must match.

On failure, output:

```
<execution_result>
{
 "error_tool_call":{
   "code":"INVALID_PARAMETERS",
   "message":"<concise reason>",
   "details":{
      "errors":[...]
   }
 }
}
</execution_result>
```

4. Tool Response Generation
Only when steps 1, 2, and 3 all pass:

- If a related call appears in the few-shot section with matching name and similar arguments, reuse its execution result content.
- Otherwise, generate a reasonable execution result that strictly satisfies the input schema and stays factually consistent with the few-shot patterns.
- May need to fix some type or error in the examples.

Output Rules
Apply to both success and error paths:

- Wrap the result in `<execution_result>...</execution_result>` XML tags.
- Wrapped content must be valid JSON: a non-empty object or an array.
- Return ONLY the wrapped JSON; no commentary, no markdown, no code fences.
- Act as a silent function executor: no explanations, no suggestions, no hints, no references to examples or comparisons.

Format:

```
<execution_result>
<JSON object or array>
</execution_result>
```

Self-Evaluation Prompt

You are an agentic reasoning supervisor.
Your job is to evaluate the agent's last simulated action in a given environment and provide actionable suggestions for improvement.
You must be strict, concise, and goal-oriented. Always assume the agent is allowed to revise its plan.

Input Information
You will be given:

- The environment description:
  {tool_documents}
- The conversation history:
  {history}
- The agent's simulated action and returned result:
  {simulation}

Evaluation Procedure
Your tasks:

1. Judge whether the agent's action is correct, helpful, and aligned with the user's task goal.
2. Identify any mistakes or weaknesses in the action.
3. Explain why the action failed or is suboptimal, based strictly on the environment feedback.
4. Return the exact evaluation result within the <Result> and </Result> tags:
   - Output 1 if the action is correct.
   - Output 0 if the action is incorrect.
5. If the action is incorrect, provide a revised action or an improved action plan for the next simulation attempt.

Output Format

```
<Evaluation>
(Detailed judgement and explanation)
</Evaluation>

<Result>
1/0
</Result>

<Suggestion>
(Detailed suggestion)
</Suggestion>
```

Summarization Prompt for Final Execution Recommendation

You are a senior execution advisor.
Your job is to summarize multiple simulated attempts and evaluations into a single, reliable execution recommendation.
You do not simulate tools or evaluate correctness.
You produce a final actionable recommendation that will be executed once in a real environment.
Input Information
You will receive:

- The environment description:
  {tool_documents}
- The conversation history:
  {history}

- Multiple simulated attempts and feedback: {Simulation_history}

Each simulated attempt may include:

- The action taken.
- The simulated tool return.
- Optional self-evaluation and suggestions.

Recommendation Procedure
Your tasks:

1. Identify the most reliable execution strategy based on all simulated attempts.
2. Extract key lessons from failures and partial successes.
3. Determine the safest and most likely-to-succeed action plan.

Important Rules

- Do not repeat the full history.
- Do not output multiple alternative plans.
- Produce ONE final recommendation using natural language.
- Be conservative: prefer actions that are robust and verified in simulation.
- You may include a step-by-step action sequence, but clearly state that it is FOR REFERENCE ONLY (not to be followed blindly) and explain why each step is taken.

Output Format

```
{
"recommendation": "...",
"rationale": "..."
}
```

Previous Attempt History Prompt

Previous attempts that you can refer to:

```
<Attempt>
<Action>{action}</Action>
<Evaluation>{evaluation}</Evaluation>
<Suggestion>{suggestion}</Suggestion>
</Attempt>
```