Learning While Acting: A Skill-Enhanced Test-Time Co-Evolution Framework for Online Lifelong Learning Agents

Bo Mao[1], **Jie Zhou**[1,2]*, **Yutao Yang**[1], **Xin Li**[2], **Xian Wei**[3], **Qin Chen**[1], **Xingjiao Wu**[1], **Liang He**[1]
[1] School of Computer Science and Technology, East China Normal University,
[2] Shanghai AI Laboratory,
[3] Software Engineering Institute, East China Normal University
{jzhou, qchen, lhe}@cs.ecnu.edu.cn

Abstract

Lifelong learning is essential for Large Language Model (LLM) agents operating in dynamic, interactive environments. However, existing lifelong learning agents for long-horizon tasks typically depend on discrete skill or past experiences retrieval with static parameters during inference, which prevents them from continuously internalizing test-time feedback like human learners. To bridge this gap, we propose Skill-enhanced Test-Time Co-Evolution (`LifeSkill`), a two-stage reinforcement learning framework for Online Lifelong Learning Agents. Specifically, we design Verifier-Guided Skill Learning that addresses the lack of direct supervision for skill extraction by rewarding candidate skills according to the average verifier success of multiple skill-conditioned policy rollouts, encouraging the model to generate skills that are useful for solving tasks rather than merely plausible in text. Furthermore, we introduce Online Skill Internalization, which continuously improves the policy model during test-time interaction by transforming skill-conditioned trajectories into reward signals. This enables the agent to directly internalize reasoning capabilities into its parameters, avoiding the context bloat of experience retrieval. Experiments on LifelongAgentBench show that LifeSkill improves average performance by 7 absolute points by comparing with existing lifelong agent baselines.

1 Introduction

Large language model (LLM) agents have shown promising performance in long-horizon decision making, interactive planning, and sequential problem solving. However, real-world environments are dynamic: task distributions evolve over time, useful strategies emerge through repeated interaction, and agents must continually accumulate knowledge from experience. This makes lifelong learning or continual learning a core requirement for LLM agents (Gao et al., 2025; Yang et al., 2025). Rather than treating each episode as an isolated inference problem, a lifelong agent should continuously improve while interacting with the environment, converting test-time feedback into persistent capability growth (Fang et al., 2025; Cai et al., 2025).

Recent studies have taken initial steps toward this goal, mainly through memory-based experience reuse and test-time self-improvement. Memory-driven agents such as Synapse (Zheng et al., 2024), AWM (Wang et al., 2025), AgentKB (Tang et al., 2025), and CER (Liu et al., 2025) store and retrieve past trajectories or summarized workflows to guide future decisions. In parallel, test-time scaling and self-reflection methods improve performance by exploring multiple reasoning paths and revising failed attempts during inference (Shinn et al., 2023; Madaan et al., 2023). Reinforcement learning with verifiable rewards has further shown strong potential for improving reasoning in domains with deterministic outcomes (DeepSeek-AI, 2025). Despite these advances, existing lifelong learning agents still fall short of genuine continual adaptation.

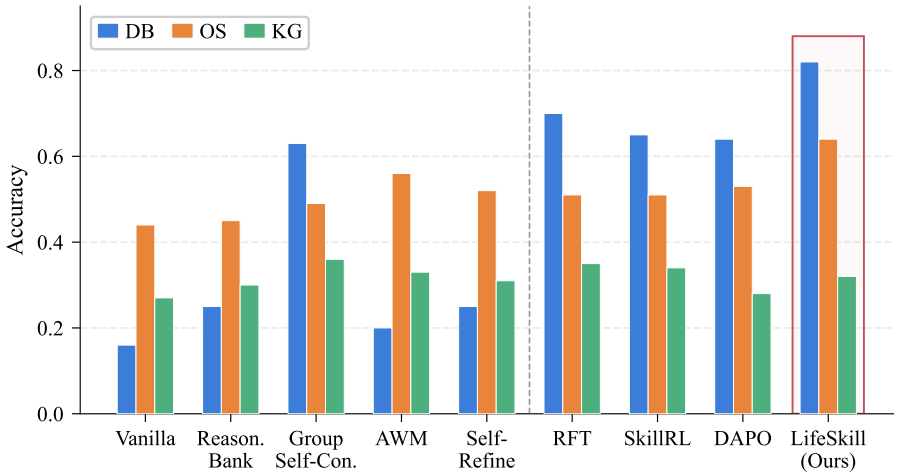


Figure 1: Comparison of all methods on LifelongAgentBench. Left of the dashed line: training-free (memory retrieval) baselines; right: training-based methods. `LifeSkill` (highlighted) achieves the highest accuracy on DB and OS and the best overall average.

In particular, current approaches face three key limitations. **First**, most agents keep model parameters fixed at test time and rely on external memories to improve performance, which prevents them from truly learning during deployment. **Second**, although prior work can summarize trajectories or produce reflections, how to extract *reusable skills* from interaction feedback remains insufficiently explored, especially under reliable and execution-grounded supervision. **Third**, acquired knowledge is usually represented as discrete external text, making it difficult for the model to internalize, compose, and generalize these skills as part of its parametric knowledge. As a result, existing agents can retrieve experience, but cannot effectively transform it into lasting capability.

To address these challenges, we propose `LifeSkill`, a Skill-Enhanced Test-Time Co-Evolution Framework for online lifelong LLM agents. The key idea is to use reinforcement learning to improve both *what skills the agent extracts from failure* and *how those skills are absorbed into the policy*. In the first stage, Verifier-Guided Skill Learning addresses the lack of direct supervision for skill extraction. After a failed attempt, the skill extractor proposes candidate skills, and each skill is evaluated by the average verifier reward of multiple skill-conditioned policy rollouts. This trains the extractor to generate skills that are useful for solving tasks, rather than merely plausible in text. In the second stage, Online Skill Internalization addresses the mismatch between skill-guided exploration and deployment. We remove explicit skill text from successful skill-conditioned trajectories and update the policy under the original task input alone, so that behaviors discovered during interaction are internalized into model parameters instead of remaining as external memory. Together, these two stages form a test-time co-evolution loop: better skill extraction leads to more effective exploration, while improved policy parameters in turn yield stronger trajectories for subsequent skill learning.

We evaluate `LifeSkill` on LifelongAgentBench, a benchmark for continual adaptation in long-horizon interactive tasks. Experimental results (Figure 1) show that `LifeSkill` consistently outperforms strong baselines and improves average performance by 7 absolute points. Further analysis confirms that both verifier-guided skill learning and online skill internalization are critical for sustained lifelong adaptation.

Our main contributions are as follows:

- We propose `LifeSkill`, a two-stage reinforcement learning framework that enables online lifelong LLM agents to improve during deployment, moving beyond static-parameter inference with external memory retrieval.
- We introduce Verifier-Guided Skill Learning to train a skill extractor from execution-grounded feedback, and Online Skill Internalization to convert successful skill-guided trajectories into unscaffolded policy improvements.

- Extensive experiments on LifelongAgentBench show that `LifeSkill` consistently outperforms strong baselines, improving average performance by 7 absolute points.

2 Related Work

Self-evolving LLM Agents. Recent surveys and benchmarks argue that agent capability should be studied as *continual adaptation* rather than single-episode inference (Gao et al., 2025; Fang et al., 2025; Yang et al., 2025; Cai et al., 2025; Wang et al., 2024; Liu et al., 2024; Zhou et al., 2023b; Koh et al., 2024; Zheng et al., 2025). A dominant line augments agents with external memory or experience banks, including trajectory exemplars, workflow memories, contextual replay buffers, cross-domain knowledge bases, and persistent memory systems (Zheng et al., 2024; Wang et al., 2025; Liu et al., 2025; Tang et al., 2025; Ouyang et al., 2025; Packer et al., 2024; Li et al., 2025; Suzgun et al., 2025). Related agents also accumulate reusable reflections or executable routines in libraries (Wang et al., 2023a; Zhao et al., 2024; Park et al., 2023). These approaches show that experience reuse is valuable, but most gains remain *outside* model parameters and therefore incur retrieval overhead, context growth, or memory interference at deployment time.

Test-time Learning. A complementary direction improves performance by allocating more inference-time computation to decomposition, revision, and search (Wei et al., 2022; Wang et al., 2023b; Zhou et al., 2023a; Yao et al., 2023b; Shinn et al., 2023; Madaan et al., 2023; Yao et al., 2023a; Besta et al., 2024; Zhou et al., 2024). Such methods can recover from local mistakes and better handle long-horizon tasks by exploring multiple candidate reasoning paths or action plans. However, their gains are usually transient: the agent often repeats similar search or self-correction on related future tasks because the discovered strategies are not persistently absorbed into the policy. Our work keeps the benefits of test-time exploration, but treats it as supervision for future capability rather than only extra computation for the current episode.

Skill Abstraction. A third line of work represents reusable behavior as skills, tools, programs, or temporally extended options. Temporal abstraction in reinforcement learning formalizes reusable high-level behaviors (Sutton et al., 1999), while LLM agents increasingly externalize procedures as code modules, tool calls, or workflow descriptions (Wang et al., 2023a; Zhao et al., 2024; Schick et al., 2023; Gao et al., 2023; Chen et al., 2023). These formulations improve compositionality and interpretability, but most prior work either assumes manually designed tool interfaces, relies on offline accumulation of successful routines, or keeps the discovered skills as explicit scaffolds during future inference. In contrast, we treat skills as an intermediate learning interface: they guide exploration after failure and then internalize the resulting behavior back into the policy.

Online Continual Learning. At the parameter level, continual learning studies how to retain prior knowledge while acquiring new behavior, using mechanisms such as regularization, replay, and online sample selection (Kirkpatrick et al., 2017; Schwarz et al., 2018; Rolnick et al., 2019; Aljundi et al., 2019). Test-time adaptation further updates models during deployment under distribution shift (Sun et al., 2020; Wang et al., 2021). For LLM post-training, instruction alignment and reasoning self-improvement have been advanced by policy optimization, self-training, and verifier-based learning (Ouyang et al., 2022; Schulman et al., 2017; Zelikman et al., 2022; Yuan et al., 2023; DeepSeek-AI, 2025; Yu et al., 2025). Meanwhile, judge-based reflection pipelines often rely on subjective LLM evaluators (Zheng et al., 2023). Our setting differs in two aspects: adaptation must happen online inside interactive environments, and supervision must come from task verifiers rather than preference labels or free-form judges.

Overall, `LifeSkill` differs from prior lifelong agent methods in three key ways. First, it learns which skills to extract from failures using execution-grounded verifier feedback, rather than heuristic summaries or judge-scored reflections. Second, it performs online parameter updates during deployment instead of relying solely on external memory retrieval. Third, it internalizes successful skill-guided behaviors under the original task input, reducing long-term dependence on retrieved text.

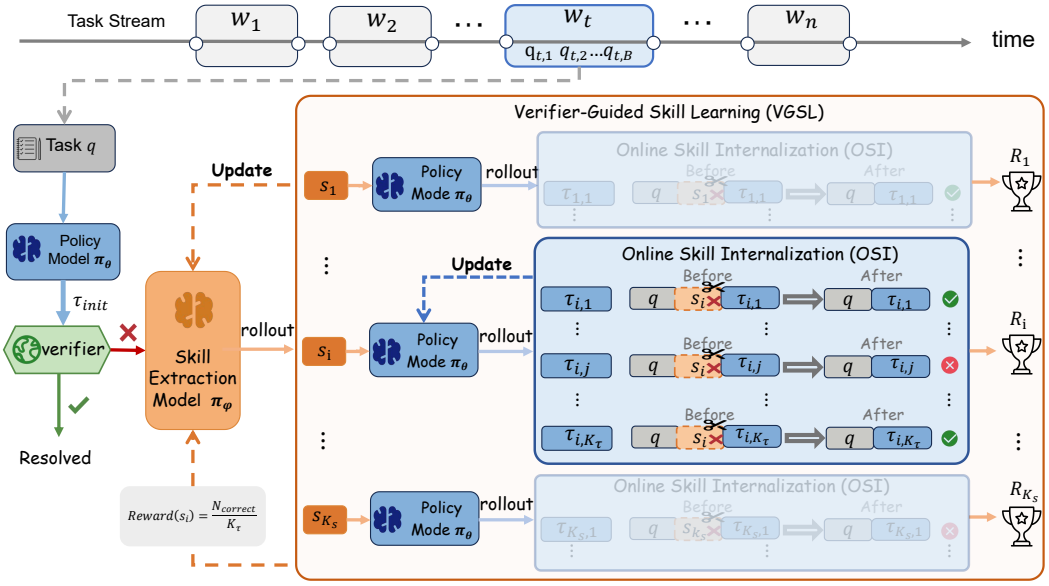


Figure 2: Overview of `LifeSkill`. Verifier-Guided Skill Learning optimizes the extractor using the average verifier success of these skill-conditioned executions, while Online Skill Internalization removes explicit skill text from verified successful trajectories and updates the Policy Model under the original task input alone.

3 Our `LifeSkill` Method

We study online lifelong learning over a stream of tasks and instantiate `LifeSkill` as a skill-enhanced test-time co-evolution framework. Unlike approaches that keep model parameters fixed during deployment or treat adaptation as a separate post-hoc step, `LifeSkill` learns while interacting with the current task stream. It maintains two models with the same backbone architecture but different parameters: a *Policy Model* π_θ for task execution and a *Skill Extraction Model* π_ϕ for distilling reusable skills from failed interactions. The two models are coupled through two reinforcement learning mechanisms. Verifier-Guided Skill Learning (VGSL; §3.3) trains π_ϕ using downstream verifier outcomes, encouraging it to extract skills that improve execution rather than merely sounding plausible. Online Skill Internalization (OSI; §3.4) updates π_θ online using successful skill-conditioned trajectories after removing the explicit skill scaffold from the input, so that useful behaviors are absorbed into the model parameters. Figure 2 illustrates the overall online co-evolution loop.

3.1 Problem Formulation: Online Lifelong Learning

We consider an online stream of tasks

$$\mathcal{D}_{\text{stream}} = (q_1, q_2, \ldots). \tag{1}$$

Following prior lifelong evaluation protocols, we process the stream using non-overlapping windows. At step t, the agent receives a batch

$$\mathcal{W}_t = \{q_{t,1}, \ldots, q_{t,B}\}, \tag{2}$$

where B is the window size. For a task q and an executable trajectory τ, the environment returns a deterministic binary verdict

$$r = \mathcal{V}(q, \tau) \in \{0, 1\}, \tag{3}$$

where $\mathcal{V}$ is the task-specific verifier and $r = 1$ indicates success.

The agent must solve the tasks in $\mathcal{W}_t$ and update its models before moving to $\mathcal{W}_{t+1}$. During this process, it can only access the current and past windows, without future tasks, human

annotations, or judge-model supervision. The goal is to maximize cumulative verified success over the task stream:

$$\max \sum_{t} \sum_{q \in \mathcal{W}_t} \mathbb{E}\left[\mathcal{V}(q, \tau_q)\right]. \tag{4}$$

3.2 Overview of LifeSkill

`LifeSkill` operates in a continual online loop over the task stream. For each task $q \in \mathcal{W}_t$, the Policy Model first attempts the task under the original input:

$$\tau^{(0)} \sim \pi_\theta(\cdot \mid q), \tag{5}$$

with verifier reward

$$r^{(0)} = \mathcal{V}(q, \tau^{(0)}). \tag{6}$$

If $r^{(0)} = 1$, the task is solved. Otherwise, the Skill Extraction Model summarizes reusable guidance from the failed attempt:

$$s \sim \pi_\phi(\cdot \mid q, \tau^{(0)}), \tag{7}$$

and the Policy Model performs an additional skill-conditioned rollout:

$$\tau^{(1)} \sim \pi_\theta(\cdot \mid q, s), \qquad r^{(1)} = \mathcal{V}(q, \tau^{(1)}). \tag{8}$$

These interactions serve two roles simultaneously. On the one hand, skill-conditioned retries improve current-window performance by enabling targeted exploration after failure. On the other hand, the collected verifier signals provide online supervision for updating both models before the agent moves to $\mathcal{W}_{t+1}$. Specifically, VGSL improves the quality of extracted skills, while OSI turns successful skill-guided behavior into unscaffolded parametric improvements. In this way, acting, skill discovery, and parameter adaptation are integrated into one continuous online loop.

3.3 Verifier-Guided Skill Learning

A key difficulty in training the Skill Extraction Model is that skill descriptions are free-form text and therefore lack explicit ground-truth labels. A common workaround is to score them with an auxiliary LLM judge, but such supervision is subjective and only indirectly related to task success. Instead, we supervise π_ϕ using *verifiable* downstream outcomes from online execution. Since the extractor outputs a variable-length skill sequence rather than a discrete label, we optimize it with the standard DAPO objective (Yu et al., 2025).

Consider a failed initial attempt $(q, \tau^{(0)})$ with $\mathcal{V}(q, \tau^{(0)}) = 0$. VGSL evaluates candidate skills by their effect on downstream success. First, the Skill Extraction Model samples K_s candidate skills:

$$s_i \sim \pi_\phi(\cdot \mid q, \tau^{(0)}), \qquad i = 1, \ldots, K_s. \tag{9}$$

For each candidate skill s_i, the Policy Model then samples K_τ skill-conditioned trajectories:

$$\tau_{i,j} \sim \pi_\theta(\cdot \mid q, s_i), \qquad j = 1, \ldots, K_\tau, \tag{10}$$

and obtains verifier rewards:

$$r_{i,j} = \mathcal{V}(q, \tau_{i,j}). \tag{11}$$

We define the utility of skill s_i by its empirical success rate:

$$R_i \;=\; \frac{1}{K_\tau} \sum_{j=1}^{K_\tau} r_{i,j}. \tag{12}$$

Intuitively, R_i measures whether the skill improves executable behavior, rather than whether it merely appears plausible in text. We then use these verifier-based skill utilities to optimize the Skill Extraction Model with DAPO. Concretely, for each failed attempt, the sampled

candidate skills form a comparison group, and we convert their utilities into normalized relative advantages:

$$\hat{A}_i \;=\; \frac{R_i - \mu_R}{\sigma_R + \epsilon}, \qquad \mu_R \;=\; \frac{1}{K_s}\sum_{i=1}^{K_s} R_i, \qquad \sigma_R \;=\; \sqrt{\frac{1}{K_s}\sum_{i=1}^{K_s}(R_i - \mu_R)^2}. \tag{13}$$

For a tokenized skill $s_i = (s_{i,1}, \ldots, s_{i,|s_i|})$, we define the token-level importance ratio as

$$\rho_{i,\ell}(\phi) \;=\; \frac{\pi_\phi(s_{i,\ell} \mid q, \tau^{(0)}, s_{i,<\ell})}{\pi_{\phi_{\text{old}}}(s_{i,\ell} \mid q, \tau^{(0)}, s_{i,<\ell})}. \tag{14}$$

The Skill Extraction Model is then updated with the DAPO token-level clipped objective:

$$\mathcal{L}_{\text{VGSL}}(\phi) \;=\; -\mathbb{E}_{i,\ell}\left[\min\left(\rho_{i,\ell}(\phi)\hat{A}_i,\, \text{clip}\big(\rho_{i,\ell}(\phi),\, 1-\epsilon_{\text{low}},\, 1+\epsilon_{\text{high}}\big)\hat{A}_i\right)\right]. \tag{15}$$

Here, the expectation is taken over all generated skill tokens for failed tasks in the current window. In this way, VGSL preserves the optimization stability of DAPO while grounding the learning signal in downstream verified execution outcomes, so the extractor is rewarded for proposing skills that actually improve policy rollouts.

3.4 Online Skill Internalization

VGSL improves online exploration by discovering useful skill text, but an online lifelong agent must also convert these discoveries into persistent capability improvements during deployment. Otherwise, useful skills would remain external guidance that needs to be regenerated or reintroduced repeatedly, rather than becoming part of the policy itself.

Let $\mathcal{B}_t = \{(q, s, \tau, r)\}$ denote the collection of all skill-conditioned trajectories gathered in window t, including both the online retries and the rollouts used by VGSL.

For each sample $(q, s, \tau, r) \in \mathcal{B}_t$, we remove the explicit skill from the conditioning context and retain only the original task-query/output pair:

$$(q, s, \tau) \;\Rightarrow\; (q, \tau). \tag{16}$$

Importantly, we do not modify the output trajectory itself; we only remove the external scaffold from the input.

Because the conditioning context changes from (q, s) to q, the original sampling probabilities are no longer valid. We therefore recompute the trajectory likelihood under the current Policy Model conditioned on q alone:

$$\log \pi_\theta(\tau \mid q) = \sum_{\ell=1}^{|\tau|} \log \pi_\theta(\tau_\ell \mid q, \tau_{<\ell}). \tag{17}$$

We then update π_θ using the verifier reward associated with each pruned trajectory:

$$\mathcal{L}_{\text{OSI}}(\theta) = -\frac{1}{|\mathcal{B}_t|}\sum_{(q,s,\tau,r)\in\mathcal{B}_t} r \log \pi_\theta(\tau \mid q). \tag{18}$$

Since $r \in \{0, 1\}$, only verified successful trajectories contribute positive learning signal in practice. This update can be viewed as reward-weighted policy optimization under the *unscaffolded* task condition.

OSI complements VGSL: the latter improves the quality of online skill-guided exploration, while the former turns successful online discoveries into lasting parametric knowledge. Together, they enable continual adaptation under the original task input rather than relying on ever-growing external text memories.

Method	Experience	Accuracy ↑			
		DB	OS	KG	Avg.
Training-Free Methods					
Vanilla	0	0.16	0.44	0.27	0.29
Self-Refine (Madaan et al., 2023)	0	0.25	0.52	0.31	0.36
Reasoning Bank (Ouyang et al., 2025)	4	0.25	0.45	0.30	0.33
Group Self-Consistency (Zheng et al., 2025)	4	0.63	0.49	**0.36**	0.49
AWM (Wang et al., 2025)	4	0.20	<u>0.56</u>	0.33	0.36
Training-Based Methods					
RFT (Yuan et al., 2023)	0	<u>0.70</u>	0.51	<u>0.35</u>	<u>0.52</u>
DAPO (Yu et al., 2025)	0	0.64	0.53	0.28	0.48
SkillRL (Xia et al., 2026)	4	0.65	0.51	0.34	0.50
`LifeSkill` (Ours)	0	**0.82**	**0.64**	0.32	**0.59**

Table 1: Main results on LifelongAgentBench across three environments. "Experience" denotes the number of past experiences injected into the prompt at inference time. Best results per column are in **bold**; second-best are <u>underlined</u>.

4 Experiments

4.1 Experimental Setup

Datasets. We conduct experiments on LifelongAgentBench (Zheng et al., 2025), a benchmark for evaluating lifelong learning in LLM agents. It contains three interactive environments: Database (DB), Operating System (OS), and Knowledge Graph (KG). Each environment requires the agent to solve interdependent tasks under deterministic binary feedback.

Baselines. We compare `LifeSkill` with two categories of baselines, all built on the same Llama-3.1-8B-Instruct backbone. (1) **Training-free methods (memory retrieval).** These methods improve performance through context augmentation without updating model parameters. We compare against Vanilla (no retrieval), Reasoning Bank (Ouyang et al., 2025), Group Self-Consistency (Zheng et al., 2025), AWM (Wang et al., 2025), and Self-Refine (Madaan et al., 2023). (2) **Training-based methods.** These methods update model parameters but do not perform online dual-model co-evolution. We compare against Rejection Sampling Fine-Tuning (RFT) (Yuan et al., 2023), SkillRL (Xia et al., 2026), and DAPO (Yu et al., 2025). Since SkillRL was not originally introduced on LifelongAgentBench, we adapt its recursive skill-augmented training pipeline to this benchmark under the same backbone and evaluation protocol.

Implementation Details. All methods use Llama-3.1-8B-Instruct as the backbone. In `LifeSkill`, the Policy Model and the Skill Extraction Model are initialized from the same backbone while maintaining separate LoRA adapters with rank 32 and $\alpha = 64$. Following the online protocol of LifelongAgentBench, we process tasks sequentially in the benchmark's default order, using temperature 1.0 and top-p 0.9 for rollout generation. We update the Skill Extraction Model online with the standard DAPO token-level clipped objective (Yu et al., 2025), using dual clipping with clip ranges $(0.2, 0.28)$. We set the learning rate to 2×10^{-5} and accumulate two samples before each online update. We will release the codes and models on GitHub.

4.2 Main Results

Table 1 and Figure 1 summarize the main results. `LifeSkill` achieves the best average accuracy of 0.59, outperforming the strongest training-free baseline by 10 absolute points and the strongest training-based baseline by 7 points.

Comparison with training-free methods. Training-free methods improve performance by injecting retrieved experiences into the prompt, but they do not update model parameters. Even with up to four retrieved memories, their gains remain limited on challenging environments. On DB, the strongest retrieval-based baseline, Group Self-Consistency, reaches 0.63, which is still 19 points below `LifeSkill` (0.82). Notably, `LifeSkill` achieves this result

without any retrieved context, suggesting that internalizing skills into model parameters is both more effective and more context-efficient than prompt-based retrieval.

Comparison with training-based methods. Among training-based methods, RFT achieves 0.70 on DB, while DAPO reaches 0.53 on OS. Both remain substantially below `LifeSkill`, which attains 0.82 on DB and 0.64 on OS. The gap is especially clear on DB (+0.12 over RFT), where long-horizon tasks require structured reasoning beyond what reward-only trial-and-error optimization can reliably provide. In contrast, the co-evolving Skill Extraction Model supplies more informative guidance for exploration and policy improvement.

Generalization across environments. `LifeSkill` ranks first on DB and OS and achieves the best average performance across all three environments. On KG, however, it does not outperform the strongest training-free baseline (0.32 vs. 0.36 for Group Self-Consistency). A possible reason is that KG tasks typically involve longer trajectories, which makes the binary reward signal much sparser and weakens the learning signal for both skill extraction and policy internalization. Improving performance in sparse-reward, long-horizon settings remains an important direction for future work.

4.3 Continual Learning Analysis

Task	Method	Retention ↑	Forgetting ↓
DB	RFT	0.552	0.156
	DAPO	0.604	**0.042**
	`LifeSkill`	**0.708**	0.104
OS	RFT	0.406	0.104
	DAPO	0.469	**0.042**
	`LifeSkill`	**0.552**	0.083
KG	RFT	**0.293**	0.076
	DAPO	0.250	**0.043**
	`LifeSkill`	0.283	0.065

Table 2: Post-update retention and forgetting. Retention and forgetting are evaluated on previous-window tasks after the full online stream has been processed.

Table 2 provides a post-update retention analysis. After processing the full online stream, we re-evaluate tasks from previous windows and measure both the final retained performance and the corresponding forgetting. On DB and OS, `LifeSkill` achieves the highest Past Retention, suggesting that skill-guided exploration followed by unscaffolded internalization improves retained earlier-window capability. DAPO obtains lower Forgetting, while `LifeSkill` reaches higher retained accuracy, reflecting a stability-adaptation trade-off between conservative updates and stronger online improvement. On KG, `LifeSkill` remains below RFT in retention, which is consistent with the sparse-reward limitation discussed above.

4.4 Ablation Studies

Table 3 shows that each component of `LifeSkill` contributes to the final performance. Replacing Verifier-Guided Skill Learning (VGSL) with LLM-as-Judge supervision reduces accuracy from 0.82/0.64 to 0.70/0.62 on DB/OS, indicating that execution-grounded verifier rewards provide a more reliable signal for skill extraction than subjective scoring. Removing the Skill Extraction Model leads to the largest degradation, dropping performance to 0.64 on DB and 0.53 on OS, which highlights the importance of dual-model co-evolution for discovering useful guidance beyond reward-only policy optimization. Disabling Online Skill Internalization (OSI) also hurts performance, reducing accuracy to 0.79 on DB and 0.60 on OS, showing that skill extraction alone is not sufficient unless the discovered behaviors are further absorbed into the policy parameters.

Component	Configuration	DB	OS
Reward Signal	w/o VGSL (LLM-Judge)	0.70	0.62
Architecture	w/o Skill Extraction	0.64	0.53
Skill Internalization	w/o OSI	0.79	0.60
`LifeSkill` (Ours)	Full Model	**0.82**	**0.64**

Table 3: Results of ablation studies.

Method	DB Avg. ± Std.	OS Avg. ± Std.	Avg.
DAPO	0.640±0.061	0.530±0.050	0.585
`LifeSkill`	**0.820**±0.044	**0.640**±0.056	**0.730**

Table 4: Task-order sensitivity on DB and OS. We report the mean accuracy and sample standard deviation across three different task orders.

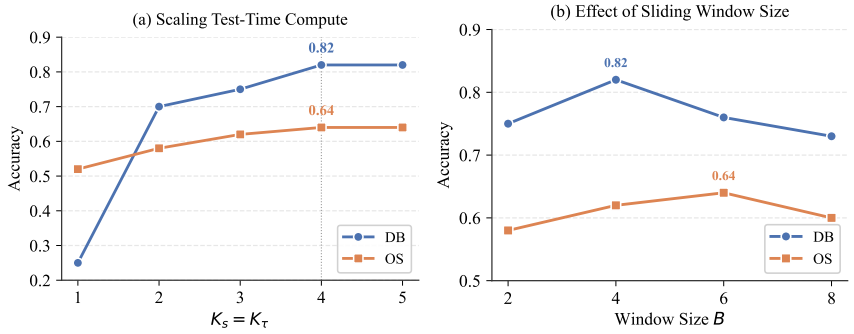
Method	Llama-3.1-8B-Instruct				Qwen2.5-7B-Instruct			
	DB	OS	KG	Avg.	DB	OS	KG	Avg.
Vanilla	0.16	0.44	0.27	0.29	0.74	0.50	0.22	0.49
RFT	0.70	0.51	**0.35**	0.52	0.77	0.51	0.26	0.51
DAPO	0.64	0.53	0.28	0.48	0.76	0.53	0.31	0.53
`LifeSkill`	**0.82**	**0.64**	0.32	**0.59**	**0.83**	**0.65**	**0.38**	**0.62**

Table 5: Backbone ablation results. Qwen2.5-7B-Instruct follows the same online evaluation protocol as the Llama-3.1-8B-Instruct experiments.

We further analyze sensitivity to the online task order in Table 4. `LifeSkill` maintains higher mean accuracy than DAPO on both DB and OS, improving the average of the two environments from 0.585 to 0.730. The standard deviations are comparable across methods, indicating that the relative gain is stable under different stream permutations.

We further conduct a backbone ablation in Table 5, comparing Llama-3.1-8B-Instruct with Qwen2.5-7B-Instruct under the same online protocol for Vanilla, RFT, DAPO, and `LifeSkill`. The Qwen backbone yields much stronger Vanilla performance on DB, a smaller improvement on OS, and lower Vanilla performance on KG, indicating that the effect of backbone choice varies across environments. For `LifeSkill`, switching from Llama to Qwen improves accuracy on all three environments, with the largest gain appearing on KG.

4.5 Hyperparameter Analysis



(a) Sensitivity of K_s and K_τ. (b) Sensitivity of the window size B.

Figure 3: Hyperparameter analysis of `LifeSkill`.

Scaling Test-Time Compute. We study the effect of test-time compute by varying the number of sampled candidate skills K_s and skill-conditioned trajectories K_τ. For simplicity, we set $K_s = K_\tau$ and vary them jointly from 1 to 5. As shown in Figure 3a, increasing the rollout budget consistently improves performance on both DB and OS. On DB, accuracy

Stage	Extracted Skill
Early	Apply filtering in a subquery with `WHERE` before ordering and limiting the results.
Middle	Use `HAVING`, rather than `WHERE`, to filter aggregate results after grouping.
Late	Map strict count constraints such as 'exceeds' and 'fewer than' to strict inequalities in `HAVING`, rather than inclusive `BETWEEN` conditions.

Table 6: Representative extracted skills from early, middle, and late training stages.

Item	Content
Query	For `Fiction` books published after 2000, return yearly counts where the total exceeds 5 but is fewer than 10.
Initial SQL	`SELECT genre, published_year, COUNT(*) AS total_count FROM library_books WHERE genre = 'Fiction' AND published_year > 2000 GROUP BY genre, published_year HAVING COUNT(*) BETWEEN 5 AND 9;`
Extracted skill	Map strict count constraints such as 'exceeds' and 'fewer than' to strict inequalities in `HAVING`.
Corrected SQL	`SELECT genre, published_year, COUNT(*) AS total_count FROM library_books WHERE genre = 'Fiction' AND published_year > 2000 GROUP BY genre, published_year HAVING COUNT(*) > 5 AND COUNT(*) < 10;`

Table 7: A skill-guided correction example on the DB environment of LifelongAgentBench. The extracted skill fixes the semantic interpretation of a strict count constraint.

rises markedly from 0.25 at $K_s{=}K_\tau{=}1$ to 0.82 at 4, and then plateaus at 5. OS shows a similar but milder trend, reaching 0.64 at 4 and remaining stable thereafter. These results suggest that a larger exploration budget generally leads to better performance, while the limited gains beyond 4 indicate a practical trade-off between compute cost and accuracy.

Effect of Window Size. We further analyze the effect of the window size B, which controls the trade-off between update frequency and optimization stability. Smaller windows enable more frequent online updates but may introduce noisier gradients, whereas larger windows provide more stable updates at the cost of slower adaptation. As shown in Figure 3b, DB achieves its best performance at $B{=}4$ (0.82) and gradually declines as the window becomes larger, while OS peaks at $B{=}6$ (0.64). This difference suggests that the two environments favor different adaptation granularities: DB appears to benefit more from faster updates, whereas OS benefits from slightly larger windows.

4.6 Qualitative Analysis

We use two case studies to understand what the Skill Extraction Model learns and how the extracted skills affect downstream execution. Table 6 traces representative skills from different training stages. Table 7 shows a concrete failure-recovery example in which the extracted skill changes the policy's SQL decision and corrects the final answer.

Table 6 shows that the extracted skills become progressively more precise and execution-relevant over training. Early-stage skills are relatively generic and can be partially misleading, while middle-stage skills begin to capture operator-level distinctions such as the different roles of `HAVING` and `WHERE`. By the late stage, the extracted skill aligns more closely with the underlying task semantics, correctly mapping strict natural-language constraints to strict inequalities. This trend suggests that verifier-guided training gradually suppresses generic reflections and encourages compact skills that are more useful for downstream execution.

Table 7 further shows that the extracted skill is not merely descriptive. The initial policy applies an inclusive interpretation to a strict cardinality constraint, which leads to an incorrect answer. After injecting the extracted skill, the retry adopts the correct semantic reading and solves the task successfully. This example indicates that the extracted skill

operates at the level of constraint interpretation, steering the policy toward a corrected execution rather than merely paraphrasing the previous failure.

5 Conclusions and Further Work

In this paper, we proposed `LifeSkill`, a skill-enhanced test-time co-evolution framework for online lifelong LLM agents. By combining Verifier-Guided Skill Learning and Online Skill Internalization, `LifeSkill` enables the agent to extract useful skills from failed interactions and absorb successful skill-guided behaviors into its policy parameters during deployment. Experiments on LifelongAgentBench show that `LifeSkill` consistently outperforms both memory-based and RL-based baselines, demonstrating the effectiveness of coupling execution-grounded skill extraction with online parametric adaptation. For future work, it would be valuable to improve performance in sparse-reward long-horizon settings and to reduce the test-time compute cost of skill-guided exploration.

References

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. *Online continual learning with maximally interfered retrieval*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, et al. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zhen, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, et al. Building self-evolving agents via experience-driven lifelong learning: A framework and benchmark. *arXiv preprint arXiv:2508.19005*, 2025.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL `https://doi.org/10.48550/arXiv.2501.12948`.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 1, 2025.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning*, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, et al. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, et al. Memos: An operating system for memory-augmented generation in large language models. *arXiv preprint arXiv:2505.22101*, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, et al. Agentbench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024.

Yitao Liu, Chenglei Si, Karthik R. Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 14179–14198. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.694/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *CoRR*, abs/2509.25140, 2025. doi: 10.48550/ARXIV.2509.25140. URL https://doi.org/10.48550/arXiv.2509.25140.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2024.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

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

Jonathan Schwarz, Wojciech M. Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, 2018.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2020.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112 (1–2):181–211, 1999.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. Agent KB: leveraging cross-domain experience for agentic problem solving. *CoRR*, abs/2507.06229, 2025. doi: 10.48550/ARXIV.2507.06229. URL https://doi.org/10.48550/arXiv.2507.06229.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, et al. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. 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*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning, 2026. URL https://arxiv.org/abs/2602.08234.

Yutao Yang, Jie Zhou, Xuanwen Ding, Tianyu Huai, Shunyu Liu, Qin Chen, Yuan Xie, and Liang He. Recent advances of foundation language models-based continual learning: A survey. *ACM Computing Surveys*, 57(5):1–38, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.

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

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.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825, 2023. doi: 10.48550/ARXIV.2308.01825. URL https://doi.org/10.48550/arXiv.2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners, 2025. URL https://arxiv.org/abs/2505.11942.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Pc8AU1aF5e.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. *Proceedings of Machine Learning Research*, 235:62138–62160, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, et al. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023a.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023b.