Autonomous Continual Learning for Environment Adaptation of Computer-Use Agents

Tianci Xue[1], **Zeyi Liao**[1]*, **Tianneng Shi**[2*], **Zilu Wang**[1], **Kai Zhang**[1], **Dawn Song**[2], **Yu Su**[1], **Huan Sun**[1]

[1]The Ohio State University, [2]University of California, Berkeley

{xue.681, sun.397}@osu.edu

Abstract

Real-world digital environments are highly diverse and dynamic. These characteristics cause agents to frequently encounter unseen environments and distribution shifts, making continual learning in such environments essential for computer-use agents (CUAs). However, a key challenge lies in obtaining high-quality and environment-grounded training data without relying on costly human annotation. In this work, we introduce ACuRL, an **A**utonomous **Cu**rriculum **R**einforcement **L**earning framework that continually adapts agents to specific environments with zero human data. The agent first explores an environment to acquire initial experiences. During subsequent iterative training, a curriculum task generator leverages these experiences together with feedback from the previous iteration to synthesize new tasks tailored for the agent's current capabilities. To provide reliable reward signals, we introduce CUAJudge, a robust automatic evaluator for CUAs that achieves 93% agreement with human judgments. Empirically, our method effectively enables both intra-environment and cross-environment continual learning, yielding 3–29% absolute performance gains on the target environments without catastrophic forgetting on others. We also show that it can mitigate performance degradation under environment changes (e.g., version updates, platform migration, and resolution shifts). Further analyses show highly sparse updates (e.g., only 20% parameters), which helps explain the effective and robust adaptation. Our data and code are available at https://github.com/OSU-NLP-Group/ACuRL.

1 Introduction

Driven by rapid advances in Large Language Models (LLMs), autonomous agents capable of interacting with digital worlds, such as browsing the web [14, 81, 78] or using a computer [69, 68, 3, 63], have emerged as a central and highly promising research direction. With active investments, generalist computer-use agents, such as Claude Computer Use [3, 4] and UI-TARS [48, 63], have shown remarkable improvement on established benchmarks like OSWorld [69] and Mind2Web [14, 71].

However, strong benchmark performance does not necessarily translate into robustness in real-world applications. First, real-world digital environments are diverse with millions of desktop and web applications, which renders comprehensive coverage during training infeasible. As a result, agents frequently encounter unfamiliar or completely new environments where the functionalities and interaction logic lie beyond their internal knowledge. For example, Claude-3.7's performance drops from 37% on OSWorld to 10% on the later released ScienceBoard environments [59]. Second, real-world digital environments are inherently dynamic [45, 71]: they evolve continuously, with underlying functionality and user interface being updated over time. Such changes cause distribution shifts for agents trained under static assumptions [25]. As shown in Figure 2, variations such as software updates, platform migration, and resolution changes consistently degrade performance,

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

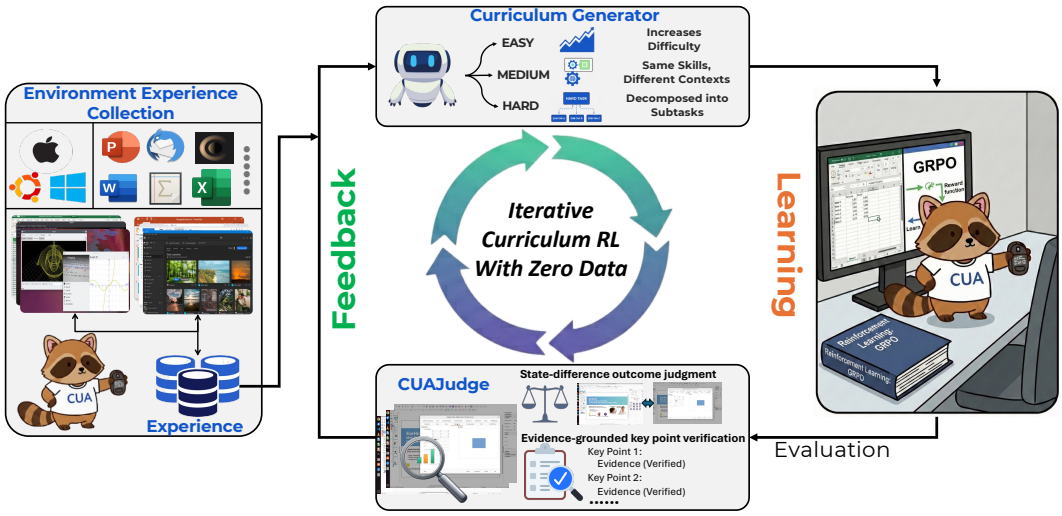


Figure 1: The overview of our ACuRL framework. The agent first autonomously interacts with the environment to collect initial environment experience, and then undergoes iterative RL training to continually learn and adapt to target environments through curriculum tasks with difficulty levels tailored to the agent's current capabilities based on feedback from CUAJudge.

with relative drops of up to 51%. These challenges highlight *continual learning in specific target environments* as a central requirement for real-world applicability, which remains unexplored.

In this work, we make the first effort to fill the gap and introduce ACuRL (Figure 1), an autonomous curriculum reinforcement learning framework that effectively enables agents to continually learn in target environments without any human supervision. The framework begins with the agent autonomously exploring target environments under diverse web-crawled contexts to gather initial environment experiences, including contents, interfaces, and functionalities. The agent is then trained through an iterative reinforcement learning (RL) to progressively learn environment-related knowledge. At the end of each iteration, an automatic evaluator will assess the current agent's capabilities, and a curriculum generator uses this feedback alongside accumulated experiences to synthesize a new set of tasks. The difficulty level of the new tasks is tailored to the agent's current capability. To provide reliable evaluation reward signals, we further introduce CUAJudge, an automatic evaluator for CUAs, which achieves 93% agreement with human judgments. CUAJudge performs a difference analysis between initial and final environment states and verifies each key point for task completion based on supporting screenshots and actions, enabling reliable and interpretable outcome assessment. Finally, to address the challenges of scaling RL to large-scale, realistic, and dynamic digital environments, we introduce a lightweight unified environment management protocol that supports batched environment creation and deletion with fault tolerance. Combined with asynchronous environment preloading and evaluation, our design significantly accelerates the training process.

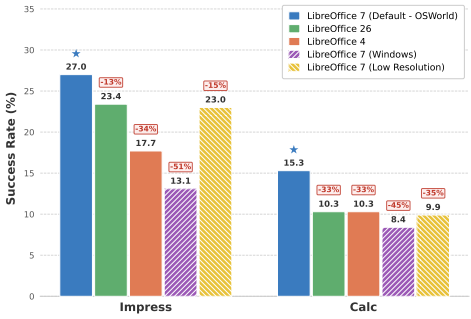


Figure 2: Impact of environmental dynamics, including software updates, platform migration, and resolution changes. For software updates, we evaluate both the latest version (LibreOffice 26) and a version with major UI changes (LibreOffice 4). For platform migration, we evaluate on Windows with the same default version as OSWorld. For resolution variation, we test a low resolution (1280×800). Here, default OSWorld refers to the benchmark's default environment configuration on Ubuntu.

Empirical results across six representative environments demonstrate the effectiveness of our approach for both intra-environment and cross-environment continual learning scenarios, yielding ab-

solute performance improvements of 3–29% across iterations on target environments without catastrophic forgetting in others. Furthermore, we evaluate ACuRL under realistic environment changes in Figure 4 and show that it effectively mitigates their impact, achieving relative performance gains of up to 145% under such shifts. To better understand how the agent evolves during continual learning, we conduct a detailed analysis of the agent's internal parameter updates. We find that only about 20% of the parameters are substantially updated, consistent with recent findings on RL [41]. Moreover, the update patterns differ markedly between the LLM backbone and the vision encoder. Specifically, parameter updates in the LLM backbone are primarily concentrated in the higher layers, whereas updates in the vision encoder gradually diminish as depth increases. This analysis helps explain why the agent can continually learn in target environments while preserving performance in other environments, thereby mitigating catastrophic forgetting.

2 Continual Learning of CUAs and Related Work

Real-world digital environments are diverse and evolving; as a result, agents frequently encounter unfamiliar or completely new environments where little or zero data are seen during training. Prior work relies on fine-tuning with human-annotated, domain-specific data, which is costly, unscalable, and generalizes poorly to unseen environments [64, 35, 23]. Moreover, both environment dynamics (e.g., software updates and platform migration) and user needs (e.g., new environments and more complex goals) evolve sequentially over time, rendering the static-data assumption inadequate and making continual learning essential. Building on this, we formulate the problem as *continual learning of computer-use agents*, where a base agent with agentic capabilities (e.g., UI grounding and planning) continually learns from a sequential set of tasks $\mathcal{U} = (T^{(1)}, T^{(2)}, \ldots, T^{(N)})$ over time. We consider two continual learning scenarios [80] that differ in the scope of the environment: (i) ***intra-environment continual learning***, where tasks come from a single environment with shared interface and dynamics, but exhibit different distributions across different iterations, where they progressively vary in goals, required skills, and complexity and require the agent to continually expand its capabilities over time. (ii) ***cross-environment continual learning***, where the agent *sequentially* learns from multiple heterogeneous environments with distinct interfaces and dynamics. This requires the agent to continually adapt and transfer knowledge across environments while maintaining performance on previously learned ones without catastrophic forgetting. Our formulation of continual learning for CUAs is consistent with the general continual learning literature [31, 80], where an agent incrementally learns from a sequence of tasks or environments, adapts to evolving distributions, and retains previously acquired knowledge without catastrophic forgetting. Additional discussion of related work is provided in Appendix A.

Challenges. One fundamental challenge in continual learning of CUAs lies in acquiring high-quality, environment-grounded agent data in new environments. While traditional continual learning methods such as Elastic Weight Consolidation [26], PackNet [39], and Progressive Networks [50] have been explored, they primarily focus on mitigating catastrophic forgetting and assume access to high-quality training data, without addressing how such data can be acquired in complex, real-world environments. Recent methods based on synthetic tasks and RL [27, 24, 33] break free from human effort and have shown promise for math and coding tasks. However, they are not directly applicable to CUAs due to two key challenges: (1) ***Task generation.*** Computer-use tasks must be grounded in dynamic, multimodal environmental contexts (e.g., different initial states of software) that are rarely present in pretraining data, making environment-agnostic generation unreliable [27, 77]. Moreover, as the agent improves, generating tasks that match its evolving competence remains an additional challenge for effective continual learning. (2) ***Trajectory evaluation.*** Unlike domains with verifiable rewards readily available such as math and coding [77, 22], computer-use task completion cannot be reliably determined from the final state alone but needs to refer to a long trajectory of multimodal observations and actions that often introduces the long context issue, making it difficult to obtain reliable reward signals during RL.

3 ACuRL

To address the aforementioned challenges, we introduce ACuRL, an **A**utonomous **Cu**rriculum **R**einforcement **L**earning framework. Specifically, for the *task generation* challenge, the agent first interacts with target environments under diverse contexts to accumulate environment-specific expe-

rience, thereby grounding task generation in the target environmental contexts (§3.1). To support effective continual learning, we further design a curriculum task generation scheme (§3.2) that synthesizes tasks tailored to the agent's evolving capabilities. For the *trajectory evaluation* challenge, we introduce `CUAJudge` (§3.3) to accurately evaluate task completion over long-horizon trajectories.

3.1 Computer-Use Task Generation

To ground generated CUA tasks in dynamic, multimodal environmental contexts, we introduce *environment exploration* and *context review* stages to acquire some environment experience, ensuring that generated tasks are realistic, executable, and grounded in diverse environmental contexts.

Environment Exploration. This stage aims to collect environment-specific experience for the task generator G within a target environment $\mathcal{E}$, including its interface and functionalities, so that it can synthesize high-quality and valid tasks. We let the agent M autonomously explore the environment $\mathcal{E}$ without any constraints, interacting with diverse interface elements and functions. The resulting action sequences $A = (a_1, a_2, \ldots, a_i)$ and observations $O = (o_1, o_2, \ldots, o_i)$ are recorded as *exploration trajectories* $\tau^{\text{exp}} = (o_1, a_1, o_2, a_2, \ldots, o_i, a_i)$, which serve as preliminary experience for subsequent task generation by G.

Context Review. For each target environment $\mathcal{E}$, we first collect diverse real-world contexts $C = (c_1, c_2, \ldots, c_i)$ via web crawling, where each context corresponds to different user-created artifacts (e.g., documents, emails or slides; examples are provided in Appendix Q). The agent M then actively interacts with the environment to review these contexts (e.g., navigating through each slide). The resulting interaction sequences are recorded as *context trajectories* τ^{ctx}. Conditioning the task generator G on these observed context trajectories significantly increases task diversity and better captures the complexity of real-world user requests. See Appendix F.3 for more details.

3.2 Iterative RL with Curriculum Learning

Traditional RL paradigms typically rely on a fixed task pool or filtering strategies to select informative ones [73, 79, 2], without adapting to the agent's evolving capabilities. Consequently, overly difficult tasks remain unsolved due to limited intermediate supervision, while trivial tasks continue to be sampled instead of being replaced by more challenging ones, hindering effective continual learning [29, 15]. This issue is particularly pronounced in the CUA domain, where agents usually perform poorly in target environments. To address this, we introduce an iterative curriculum RL framework that *dynamically adjusts task difficulty during the agent's learning progress*. Specifically, the agent is trained for N iterations, each with x optimization steps. After each iteration, we perform *capability evaluation* on the current tasks, and use the feedback to guide G in generating curriculum tasks with adjusted difficulty for the next iteration.

Capability Evaluation. At the end of iteration n, the agent is evaluated on task set $T^{(n)}$. For each task $t_k^{(n)} \in T^{(n)}$, we conduct m rollouts and quantify performance as the mean success rate across these rollouts. $y_{k,j}^{(n)}$ denotes whether the agent successfully completes task $t_k^{(n)}$ in the j-th rollout.

$$s_k^{(n)} = \frac{1}{m} \sum_{j=1}^{m} I\left(y_{k,j}^{(n)} = 1\right), \tag{1}$$

Curriculum Task Generation. Drawing on the feedback from the capability evaluation at iteration n, we generate a new set of tasks $T^{(n+1)}$ with different skills tailored to the agent's current capability. Here, we use the term skill to refer to a recurring, reusable, and semantically meaningful sub-goal or interaction pattern required to complete a task. Formally, given the previous task set $T^{(n)}$ and evaluation feedback $\{s_k^{(n)}\}$, context trajectories τ^{ctx}, exploration trajectories τ^{exp}, the generator produces a new task set for the next training iteration $T^{(n+1)} \sim G(\cdot \mid \tau^{\text{ctx}}, \tau^{\text{exp}}, T^{(n)}, \{s_k^{(n)}\})$. Specifically, tasks in $T^{(n)}$ are partitioned into three difficulty categories based on thresholds δ_{high} and δ_{low} to guide the adaptive task generation process: (i) **Easy Level ($s_k^{(i)} > \delta_{\text{high}}$):** As the agent masters the core skills required for these tasks, the generator will synthesize more challenging variants by increasing complexity, such as incorporating additional skill requirements or extending the task

horizon. (ii) **Medium Level ($\delta_{\text{low}} \leq s_k^{(i)} \leq \delta_{\text{high}}$):** Since the agent has not yet fully mastered the required skills, the generator will promote diversity while preserving learnability by synthesizing tasks that share core skills but vary in scenarios or personas. (iii) **Hard Level ($s_k^{(i)} < \delta_{\text{low}}$):** For tasks that remain difficult, the generator applies hierarchical decomposition, breaking them into meaningful subtasks to isolate essential skills and enable intermediate learning. Specifically, the generator first identifies underlying sub-steps and required skills (e.g., invoking functions or manipulating objects), then creates simpler, skill-focused but realistic tasks that are grounded in user-created artifacts.

Through this progressive task generation, the curriculum generator dynamically adjusts task difficulty with the agent's evolving capabilities, enabling effective continual learning. See Appendix F for details. To assess the validity of the generated tasks, we conduct a human evaluation on 144 tasks, uniformly sampled across iterations and six environments. Results show that 94% are valid (i.e., meaningful and executable), demonstrating the effectiveness of our method. Beyond validity, we further analyze how the synthesized tasks evolve across iterations and under different distributions. As shown in Figure 6, both average task length and the number of agent execution steps increase steadily, serving as a proxy for growing task complexity. To further corroborate this trend, we qualitatively examine representative examples. Additional details are provided in Appendix J with examples in Table 24. Together, these results indicate that our method co-evolves learning signals with the agent's capabilities, enabling effective continual learning for CUAs.

3.3 Automatic Evaluator

Another fundamental challenge in computer-use agent learning lies in obtaining reliable reward signals. To this end, we propose `CUAJudge`, a reliable automatic evaluator for CUA trajectories. Our design builds on the WebJudge [71] pipeline with three stages: key point identification, key screenshot identification, and outcome judgment. Specifically, key point identification extracts task-critical requirements from the task description; key screenshot identification selects informative screenshots from the trajectory; and outcome judgment determines task completion by reasoning over the task description, key points, key screenshots, and action history. However, WebJudge is primarily tailored to web tasks, which can be viewed as a subset of the broader computer-use tasks. In general, computer-use tasks are more complex, requiring fine-grained examination of internal system and software states [19, 1], whereas web tasks often only involve verifying surface-level effects, such as whether a specific filter has been applied. Accordingly, `CUAJudge` extends WebJudge with two key modifications to address the unique challenges of general computer-use tasks and improve evaluation reliability:

- **State difference analysis.** Many computer-use tasks (e.g., editing slides or generating plots) can be reliably verified by comparing differences between the initial and final environment states. Accordingly, we augment outcome judgment with explicit state-difference analysis that highlights these changes and incorporates them into the final decision. Gonzalez-Pumariega et al. [19] also introduce a state-difference module based on step-wise screenshot changes to reduce noise in long trajectories. In contrast, we directly compare initial and final states to determine whether the final state satisfies task requirements. Moreover, their method is used to select a trajectory from a candidate set, whereas `CUAJudge` assigns a binary reward to each individual trajectory.
- **Evidence-grounded key point verification.** `CUAJudge` explicitly evaluates the completion of each key point and requires concrete evidence for every decision, including the supporting screenshots and actions. This evidence-grounded design substantially improves evaluation precision.

3.4 Optimization

Following Wei et al. [67], we extend the Group Relative Policy Optimization (GRPO) algorithm [54] to a multi-turn setting by decomposing each trajectory into a sequence of step-level action groups and optimizing them jointly. For a given task I, we sample a group of G trajectories in the environment $\{\tau^{(i)}\}_{i=1}^{G}$. Each trajectory $\tau^{(i)} = \{(s_t^{(i)}, a_t^{(i)})\}_{t=1}^{T^{(i)}}$ consists of multiple interaction steps, where $s_t^{(i)}$ denotes the observation at step t and $a_t^{(i)}$ denotes an action generated autoregressively. We optimize the policy by minimizing the objective:

$$\mathcal{L}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{T^{(i)}}\sum_{t=1}^{T^{(i)}}\frac{1}{|a_t^{(i)}|}\sum_{k=1}^{|a_t^{(i)}|}\Big[\tilde{A}_{t,k}^{(i)} - \beta\, D_{\text{KL}}(\theta)\Big], \tag{2}$$

where $\tilde{A}_{t,k}^{(i)} = \min\{r_{t,k}^{(i)}(\theta)\,\hat{A}^{(i)}, \text{clip}\big(r_{t,k}^{(i)}(\theta), 1-\epsilon, 1+\epsilon\big)\,\hat{A}^{(i)}\}$ is the clipped surrogate objective term for the k-th token in the action $a_t^{(i)}$ of trajectory $\tau^{(i)}$ and the importance sampling ratio for the k-th token of the action at step t is defined as $r_{t,k}^{(i)}(\theta) = \frac{\pi_\theta(a_{t,k}^{(i)}|I, s_t^{(i)}, a_{t,<k}^{(i)})}{\pi_{\theta_{\text{old}}}(a_{t,k}^{(i)}|I, s_t^{(i)}, a_{t,<k}^{(i)})}$. Following GRPO, we compute a trajectory-level normalized advantage using the group of sampled trajectories: $\hat{A}^{(i)} = \frac{R^{(i)} - \text{mean}(\{R^{(j)}\}_{j=1}^{G})}{\text{std}(\{R^{(j)}\}_{j=1}^{G})}$, where $R^{(i)}$ denotes the reward for trajectory $\tau^{(i)}$ produced by `CUAJudge`. We assign the same trajectory-level advantage $\hat{A}^{(i)}$ to all actions within the trajectory.

4 Experiments

4.1 Experiment Setup

Benchmarks and Metrics. We consider six representative environments: LibreOffice Impress, LibreOffice Writer, and LibreOffice Calc from OfficeWorld [30] and OSWorld [69] for office productivity tools (we refer to them as Impress, Writer, and Calc for simplicity); Thunderbird from OSWorld for email management; and Celestia and KAlgebra from ScienceBoard [59] as specialized scientific software. Office tools represent some of the most commonly used software, where users expect agents to adapt effectively to support everyday productivity tasks. Thunderbird, while also a real-world email client, is less prevalent in pretraining data and thus presents a less familiar environment. In contrast, Celestia and KAlgebra are professional scientific applications for astronomical visualization and symbolic mathematics, which are niche and serve as testbeds for continual learning and adaptation to new environments. For evaluation, we use tasks from OSWorld, OfficeWorld, and ScienceBoard, and report success rates based on each benchmark's rule-based protocol. Table 20 in Appendix L presents the task statistics for each environment. To mitigate the effects of network instability, environmental variability and inference randomness, we report the average performance over three runs. To evaluate our automatic evaluator, we consider three benchmarks: 1,444 public OSWorld trajectories generated by Claude-Sonnet-4, Claude-Sonnet-4.5, O3, and OpenCUA-7B; AgentRewardBench, comprising 1,302 trajectories collected from five web agent benchmarks; and CUAStepBench, spanning 177 applications across six platforms (Web, Ubuntu, Windows, macOS, Android, and iOS). We report agreement, precision, recall, and F1 score against ground-truth labels. To assess reliability during RL training, we further sample 432 trajectories across six environments and compute the same metrics against human judgments.

Implementation Details. We adopt UI-TARS-1.5-7B [53] and Qwen3-VL-8B-Instruct [7] as base agents. As UI-TARS-1.5-7B predates ScienceBoard, environments like Celestia and KAlgebra are likely unseen, thereby enabling us to evaluate the effectiveness of `ACuRL` in novel environments. For our task generator, we use GPT-5 [43]. For each environment, we generate 256 tasks per iteration from web-crawled contexts. For CUAJudge, we use GPT-5-mini to reduce evaluation cost. See Appendix L for more implementation details.

4.2 Infrastructure

Building stable, efficient infrastructure for CUAs poses unique challenges. Each rollout requires an isolated Linux environment to prevent cross-trajectory interference, leading to high resource overhead. Moreover, environment initialization and response latency directly affect training throughput. To support scalable, high-throughput training, we introduce a series of infrastructure-level optimizations that accelerate training speed by 3–5 times based on our preliminary exploration.

- **Unified environment management protocol.** We introduce a lightweight API protocol that enables batch environment creation and deletion via IP-based requests, with built-in failure detection and automatic restarts. This design greatly simplifies environment orchestration and facilitates efficient large-scale environment scaling.
- **Environment preloading and asynchronous evaluation.** Environment initialization and trajectory evaluation are two major bottlenecks to training speed. Each step requires reinitializing environments, which typically takes 5–10 minutes. As a result, GPUs remain idle and stall training. To address this issue, we introduce environment preloading: while the agent is being updated, a separate worker asynchronously initializes environments for the next step and deletes previous environments. Furthermore, instead of synchronously evaluating trajectories after all rollouts

complete, we deploy a separate evaluation worker that continuously monitors and asynchronously evaluates trajectories upon completion, thereby further improving overall training efficiency.

4.3 Intra-Environment Continual Learning

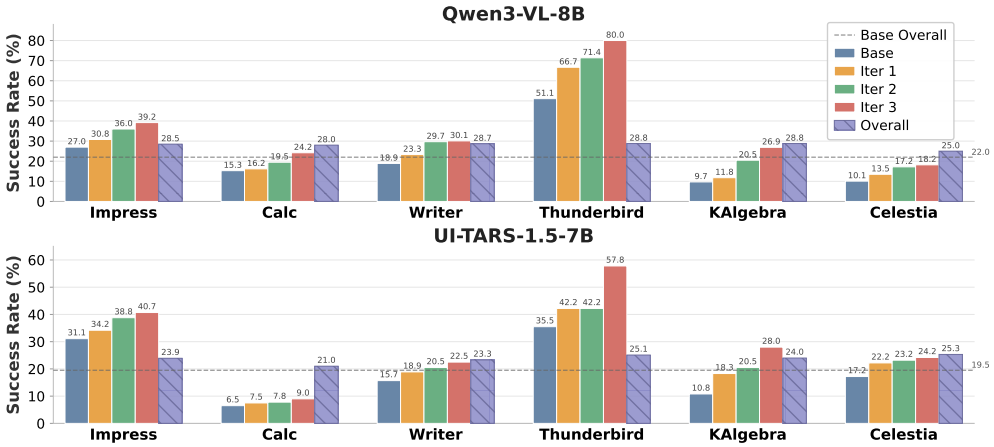


Figure 3: Average success rate across six environments for the base agent and agents that continually learn in specific environments via ACuRL. The x-axis denotes the training environment, and the y-axis reports the corresponding evaluation performance. The overall score is the average performance across all six environments, with the gray dashed line indicating the base agent's overall performance. "Iter" denotes the number of training iterations.

As shown in Figure 3, the agent achieves progressive gains of 3–29% on target environments across iterations, demonstrating effective intra-environment continual learning with autonomous skill acquisition on increasingly complex tasks. Importantly, performance on other environments is also preserved or even improved, yielding overall gains of 2–7% without catastrophic forgetting. Detailed results on each environment after continual learning are provided in Appendix O. We attribute these gains to positive cross-environment knowledge transfer, where agents acquire reusable interaction skills and decision patterns that generalize across environments (e.g., Impress to Writer). We provide a detailed analysis of these phenomena in Section 4.7 and Appendix K. Moreover, since the effectiveness of ACuRL may depend on the base agent's initial capabilities, we evaluate it across agents with varying capability levels. Appendix G.1 shows that ACuRL consistently improves agents across a broad capability range, while requiring only a minimal level of basic GUI competence.

4.4 Cross-Environment Continual Learning

Table 1: Performance of cross-environment continual learning, where agents sequentially adapt to multiple environments. Rows denote training environments and columns denote evaluation environments. Best results are **bolded**. The overall score is the average across all six environments.

Model	Agent	Impress	Calc	Writer	Thunderbird	KAlgebra	Celestia	Overall
UI-TARS-1.5-7B	Base Agent	31.1	6.5	15.7	35.5	10.8	17.2	19.5
	Impress	**40.7**	5.9	20.9	40.0	12.9	23.2	23.9
	Calc	33.2	**9.0**	16.9	37.8	16.1	13.1	21.0
	KAlgebra	30.8	5.3	16.9	48.9	**28.0**	14.2	24.0
	Impress → Calc	37.0	8.7	**23.7**	44.5	16.1	19.2	24.9
	Impress → KAlgebra	36.7	6.8	21.7	46.7	18.3	20.2	25.1
	Impress → KAlgebra → Calc	33.9	6.6	23.3	**51.1**	16.1	**24.2**	**25.9**
Qwen3-VL-8B	Base Agent	27.0	15.3	18.9	51.1	9.7	10.1	22.0
	Impress	**39.2**	14.0	24.9	64.5	17.2	11.1	28.5
	Calc	33.0	**24.2**	26.5	62.2	10.8	11.1	28.0
	Writer	27.7	17.1	**30.1**	68.9	12.9	15.2	28.7
	KAlgebra	30.1	17.1	28.5	62.2	**26.9**	8.1	28.8
	Celestia	26.4	12.2	18.1	62.2	12.9	**18.2**	25.0
	Impress → Calc	35.9	15.5	22.9	70.0	17.8	13.7	29.3
	Impress → KAlgebra	35.8	15.9	19.7	**71.1**	20.5	11.1	29.0
	Impress → KAlgebra → Calc	35.4	19.7	18.1	68.9	22.6	14.2	29.8
	Impress → KAlgebra → Calc → Celestia	33.9	19.3	22.5	**71.1**	25.8	16.2	31.5
	Impress → KAlgebra → Calc → Celestia → Writer	37.3	18.7	25.7	66.7	24.7	17.2	**31.7**

To evaluate the generalization of ACuRL to the cross-environment continual learning scenario, we conduct experiments where agents sequentially learn 2-5 distinct environments. Specifically, we initialize an agent already adapted to one environment and then perform an additional round of continual learning to enable adaptation to different target environments. As shown in Table 1, it remains effective as the number of environments increases, improving target performance and achieving the highest overall score. To more directly characterize knowledge transfer and retention, we also report standard continual learning metrics in Appendix E. As shown in Table 5, both agents exhibit positive forward transfer with limited forgetting. We conjecture that gains across seemingly unrelated environments arise from transferable decision patterns and shared interface-level knowledge, such as exploration strategies, avoiding redundant actions, and common UI conventions. Overall, these results demonstrate that our curriculum-based continual learning strategy facilitates effective knowledge accumulation while mitigating catastrophic forgetting. Moreover, it generalizes beyond single-environment specialization and scales to multiple environments, highlighting its suitability for realistic continual learning scenarios. Although ACuRL achieves substantial improvements, the agent's performance remains far below human level. To better understand this gap, we conduct a detailed failure case analysis. We find that the majority of errors stem from knowledge and planning limitations, premature termination, and repetitive action loops. See Appendix N for more details.

4.5 Effectiveness of ACuRL in Mitigating Environment Changes

As illustrated in Figure 2, real-world environment changes such as platform migration, software updates, UI modifications, and resolution shifts lead to substantial performance drops of up to 51% even on the same test set. This observation suggests that current CUAs are highly sensitive to environmental dynamics and underscores the necessity of continual learning for adaptation. To demonstrate that ACuRL can effectively mitigate these effects, we perform three iterations of continual training with ACuRL. Results show that ACuRL substantially improves performance in post-change environments. For example, it achieves a 145% improvement under platform migration from Ubuntu to Windows, as well as gains of up to 145% under version updates and 77% under resolution shifts. Overall, these findings indicate that ACuRL enables effective adaptation to diverse dynamic changes in real-world digital environments, supporting the practical deployment of CUAs.

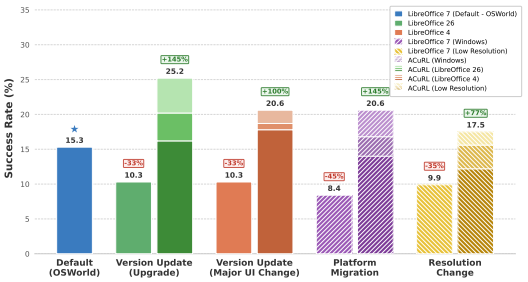


Figure 4: Effectiveness of ACuRL under environmental dynamics on Calc. It mitigates performance drops and improves performance across software updates, platform migration, and resolution changes. Color intensity indicates iterations (darker = earlier, lighter = later).

4.6 Ablation Study

To assess the contribution of each component, we conduct ablation studies that remove iterative training and curriculum learning during training. For iterative RL training, we train the agent for the same number of optimization steps using only tasks from the first iteration (75, 150 and 225 steps corresponding to Iter 1, 2 and 3). To ablate curriculum learning, we replace our task generator with random task generation at each iteration, without using evaluation feedback from the previous iteration. As shown in Table 2, removing either iterative RL training or curriculum-based task generation substantially degrades performance. Without iterative training, performance peaks at 75 steps (Iter 1) and then declines as training on a fixed task pool causes overfitting and reduces generalization to unseen tasks. Replacing curriculum-based task generation with random generation also yields smaller gains. These results highlight the importance of dynamically aligning task difficulty with the agent's evolving capabilities for effective continual learning. These components are also important in the cross-environment setting, where removing either degrades performance (Appendix G.3). We also compare ACuRL with recent curriculum-based methods and find that it achieves competitive or superior performance across environments (Appendix M). While GPT-5 is used as the generator for its strong instruction-following ability, the gains are not attributable to a stronger external model,

Table 2: Ablation study on the impact of iterative training and curriculum learning in the LibreOffice Impress environment.

Method	Iter 1	Iter 2	Iter 3
Iterative Curriculum RL	**34.2**	**38.8**	**40.7**
- w/o Iterative Training	34.2 (-0.0)	34.0 (-4.8)	32.6 (-8.1)
- w/o Curriculum Learning	29.8 (-4.4)	33.6 (-5.2)	36.4 (-4.3)

Table 3: `CUAJudge`'s precision, recall, and agreement with human judgments.

Iteration	Precision	Recall	Agreement
Iter 1	98.7	97.4	97.8
Iter 2	93.2	88.5	89.9
Iter 3	94.2	92.9	93.5
Overall	95.4	92.9	93.7

as GPT-5 lacks GUI navigation capabilities. Replacing it with Qwen3-VL-8B still yields substantial improvements (Appendix F.5). Similarly, `ACuRL` remains effective when Qwen3-VL-8B itself serves as the reward evaluator, enabling self-evaluation without a stronger external model (Appendix H.2).

4.7 Further Analysis

The Reliability of CUAJudge. The effectiveness of continual learning through RL critically depends on the reliability of the automatic evaluator. To this end, we compare CUAJudge with WebJudge across three evaluator benchmarks: Public OSWorld Trajectories, AgentRewardBench, and CUAStepBench. As shown in Table 12, `CUAJudge` achieves a high agreement of 87.7% with the rule-based evaluator. Furthermore, compared with the state-of-the-art evaluator [71, 38], `CUAJudge` consistently improves precision by 3.2–4.7% across the three benchmarks, substantially reducing false positives. As the state difference analysis module is a key component of `CUAJudge`, we conduct an ablation study to evaluate its importance; see Appendix H.1 for details. To further verify that `CUAJudge` can provide reliable reward signals for training tasks and on-policy trajectories, we conduct a human evaluation on 432 trajectories, with 72 per environment evenly sampled across three iterations. As shown in Table 3, `CUAJudge` consistently exhibits high agreement with human judgments (93.7%). Notably, as training progresses across iterations, precision and agreement remain consistently around 90%, suggesting that `CUAJudge` avoids exploitable shortcuts for reward hacking and remains well aligned with true task completion. Beyond the training environments, `CUAJudge` also generalizes effectively across web, desktop, and mobile environments with consistently high agreement; see Appendix H.3 for details. Overall, these findings suggest that `CUAJudge` could serve as a robust and reliable automatic evaluator of CUA trajectories across diverse environments and learning paradigms, including Rejection Sampling Fine-Tuning (RFT) [74], Reflection [56, 46, 70], and RL.

Table 4: Comparison of CUAJudge and WebJudge across three benchmarks.

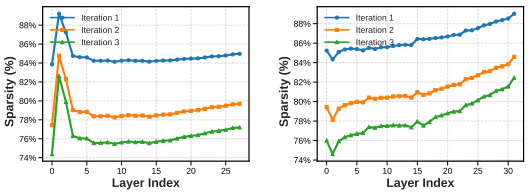
Evaluator	Agreement	Precision	Recall	F1
Public OSWorld Trajectories				
WebJudge	86.2	70.4	78.4	74.2
CUAJudge	**87.7**	**73.6**	**80.8**	**77.0**
AgentRewardBench				
WebJudge	81.5	73.2	48.8	58.5
CUAJudge	**83.2**	**76.6**	**53.4**	**62.9**
CUAStepBench				
WebJudge	68.0	71.9	63.5	67.4
CUAJudge	**74.8**	**76.6**	**74.5**	**75.5**

The Sparsity of Agent Updates during Continual Learning. As shown in Table 22 and Table 1, the agent achieves substantial gains on target environments under both intra- and cross-environment continual learning, while maintaining stable or even improved performance on other environments without catastrophic forgetting. Although [41, 10] observe sparse parameter updates during RL, their studies are limited to language models, leaving it unclear how parameters update for multimodal CUAs, where decision-making is heavily driven by visual perception from dense, information-rich screenshots. Therefore, to better understand how the model's internal representations evolve during continual adaptation, we analyze model sparsity across different iterations. Formally, we quantify it using *update sparsity* [41], which measures the fraction of parameters that remain effectively unchanged. Let $\theta^0, \theta^1 \in R^n$ denote the model parameters before and after fine-tuning, respectively, where n is the total number of trainable parameters. Given a tolerance ε to ignore negligible numerical variations, we set $\varepsilon = 1 \times 10^{-5}$ following Mukherjee et al. [41], and

define *update sparsity* as

$$\text{UpdateSparsity}_{\varepsilon}(\theta^0, \theta^1) = 1 - \frac{1}{n}\sum_{i=1}^{n} I\left(|\theta_i^1 - \theta_i^0| > \varepsilon\right), \tag{3}$$

As shown in Figure 5, although sparsity gradually decreases over training, both the LLM backbone and vision encoder remain highly sparse, with around 80% of parameters unchanged. This helps explain why the agent can continually learn in target environments while preserving performance on others. Moreover, the two components exhibit distinct layer-wise patterns: the LLM backbone maintains relatively uniform sparsity across middle and upper layers (Layers 3–27), whereas the vision encoder becomes increasingly sparse with depth. We also observe a striking contrast in the early layers of the two modules. The first one to two layers of the LLM backbone exhibit unusually high sparsity, whereas the first layer of the vision encoder shows much lower sparsity. This asymmetry suggests that early vision layers, responsible for low-level visual perception, require stronger adaptation in new environments, while early LLM layers are less involved in environment-specific perception or decision making. We leave understanding why RL yields sparse updates and how this naturally mitigates forgetting to future work.



(a) LLM backbone (b) Vision encoder

Figure 5: Sparsity of parameter updates across RL iterations, averaged over six environments.

5 Conclusion

In this work, we propose a fully autonomous curriculum reinforcement learning framework (ACuRL) that enables CUAs to continually adapt to specific environments in a zero-data setting. By grounding task generation in accumulated environment experience and progressively adjusting task difficulty to match the agent's evolving capabilities, ACuRL consistently improves performance in both intra- and cross-environment continual learning across six representative environments, without catastrophic forgetting. Moreover, it effectively mitigates performance degradation under diverse environmental dynamics, including version updates, platform migration, and resolution changes. Further analysis reveals that such adaptation is driven by highly sparse parameter updates, offering insights into how agents acquire new skills while preserving prior knowledge. Overall, these results establish ACuRL as a scalable and effective paradigm for continual learning in complex computer-use environments.

Acknowledgments

The authors would thank colleagues from the OSU NLP group for constructive feedback. This research was sponsored in part by NSF CAREER #1942980, NSF CAREER #2443149, the Alfred P. Sloan Research Fellowship, and gifts from Cisco and Amazon. We also appreciate computational resources provided by the Ohio Supercomputer Center [42]. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

[1] Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=zg5is4GJ3R.

[2] Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for

scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

[3] Anthropic. Developing a computer use model. *Anthropic News*, Oct 2024. URL https://www.anthropic.com/news/developing-computer-use.

[4] Anthropic. Claude 3.7 sonnet and claude code. *Anthropic News*, Feb 2025. URL https://www.anthropic.com/news/claude-3-7-sonnet.

[5] Ahmed Awadallah, Yash Lara, Raghav Magazine, Hussein Mozannar, Akshay Nambi, Yash Pandya, Aravind Rajeswaran, Corby Rosset, Alexey Taymanov, Vibhav Vineet, et al. Fara-7b: An efficient agentic model for computer use. *arXiv preprint arXiv:2511.19663*, 2025.

[6] Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning, 2024. URL https://arxiv.org/abs/2406.11896.

[7] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

[8] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[9] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[10] Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting, 2025. URL https://arxiv.org/abs/2510.18874.

[11] Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Phil Cuvin, Hao Zhu, and Diyi Yang. Decepticon: How dark patterns manipulate web agents. *arXiv preprint arXiv:2512.22894*, 2025.

[14] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

[15] Leander Diaz-Bone, Marco Bagatella, Jonas Hübotter, and Andreas Krause. DISCOVER: Automated curricula for sparse-reward reinforcement learning. In *The Exploration in AI Today Workshop at ICML 2025*, 2025. URL https://openreview.net/forum?id=0kjOajnJQM.

[16] Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, Kaixuan Chen, Mingli Song, and Dacheng Tao. Serl: Self-play reinforcement learning for large language models with limited data. *arXiv preprint arXiv:2505.20347*, 2025.

[17] Aarash Feizi, Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Kaixin Li, Rabiul Awal, Xing Han Lù, Johan Obando-Ceron, Juan A Rodriguez, Nicolas Chapados, et al. Grounding computer use agents on human demonstrations. *arXiv preprint arXiv:2511.07332*, 2025.

[18] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QXEhBMNrCW.

[19] Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. The unreasonable effectiveness of scaling agents for computer use. *arXiv preprint arXiv:2510.02250*, 2025.

[20] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

[21] Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your llm secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025.

[22] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small LLMs can master math reasoning with self-evolved deep thinking. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5zwF1GizFa.

[23] Yanheng He, Jiahe Jin, and Pengfei Liu. Efficient agent training for computer use. *arXiv preprint arXiv:2505.13909*, 2025.

[24] Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*, 2025.

[25] Md Farhan Ishmam and Kenneth Marino. Timewarp: Evaluating web agents by revisiting the past. *arXiv preprint arXiv:2603.04949*, 2026.

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[27] Jakub Grudzien Kuba, Mengting Gu, Qi Ma, Yuandong Tian, Vijai Mohan, and Jason Chen. Language self-play for data-free training. *arXiv preprint arXiv:2509.07414*, 2025.

[28] Hynek Kydlíček. Math-verify: Math verification library, 2025. URL https://github.com/huggingface/Math-Verify.

[29] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[30] Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025.

[31] Shane Legg. Machine super intelligence. 2008.

[32] Zeyi Liao, Yadong Lu, Boyu Gou, Huan Sun, and Ahmed Awadallah. Beyond clicking: A step towards generalist gui grounding via text dragging. *arXiv preprint arXiv:2601.06031*, 2025.

[33] Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2506.24119.

[34] Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning. *arXiv preprint arXiv:2511.19900*, 2025.

[35] Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025.

[36] Ziwei Liu, Borui Kang, Hangjie Yuan, Zixiang Zhao, Wei Li, Yifan Zhu, and Tao Feng. Continual gui agents. *arXiv preprint arXiv:2601.20732*, 2026.

[37] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[38] Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stanczak, Peter Shaw, Christopher Pal, and Siva Reddy. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=fQcUZMPIvu.

[39] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[40] Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

[41] Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tür, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0NdS4xCngO.

[42] Ohio Supercomputer Center. Ohio supercomputer center, 1987. URL https://ror.org/01apna436.

[43] OpenAI. Introducing gpt-5. *OpenAI Blog*, Aug 2025. URL https://openai.com/index/introducing-gpt-5/.

[44] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[45] Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, and Zhengyang Wu. Webcanvas: Benchmarking web agents in online environments. In *Agentic Markets Workshop at ICML 2024*, 2024. URL https://openreview.net/forum?id=O1FaGasJob.

[46] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1100–1126, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.67/.

[47] Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oVKEAFjEqv.

[48] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

[49] Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective. *Advances in neural information processing systems*, 38:59162–59222, 2026.

[50] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[51] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

[53] ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

[54] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[55] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[56] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652, 2023.

[57] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484–489, 2016.

[58] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362 (6419):1140–1144, 2018.

[59] Qiushi Sun, Zhoumianze Liu, Chang Ma, Zichen Ding, Fangzhi Xu, Zhangyue Yin, Haiteng Zhao, Zhenyu Wu, Kanzhi Cheng, Zhaoyang Liu, Jianing Wang, Qintong Li, Xiangru Tang, Tianbao Xie, Xiachong Feng, Xiang Li, Ben Kao, Wenhai Wang, Biqing Qi, Lingpeng Kong, and Zhiyong Wu. Scienceboard: Evaluating multimodal autonomous agents in realistic scientific workflows, 2025. URL https://arxiv.org/abs/2505.19897.

[60] Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience. *arXiv preprint arXiv:2508.04700*, 2025.

[61] Yang Wan, Zhenhao Zhang, Jierui Wang, Xiangxin Zhou, and Linchao Zhu. Seekjudge: A practical reward framework for reinforcement learning in computer-use agents, 2026. URL https://arxiv.org/abs/2607.23263.

[62] Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, et al. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *arXiv preprint arXiv:2512.13607*, 2025.

[63] Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, Yujia Qin, Yuwen Xiong, Yuxin Song, Zhiyong Wu, Aoyan Li, Bo Li, Chen Dun, Chong Liu, Daoguang Zan, Fuxing Leng, Hanbin Wang, Hao Yu, Haobin Chen, Hongyi Guo, Jing Su, Jingjia Huang, Kai Shen, Kaiyu Shi, Lin Yan, Peiyao Zhao, Pengfei Liu, Qinghao Ye, Renjie Zheng, Shulin Xin, Wayne Xin Zhao, Wen Heng, Wenhao Huang, Wenqian Wang, Xiaobo Qin, Yi Lin, Youbin Wu, Zehui Chen, Zihao Wang, Baoquan Zhong, Xinchun Zhang, Xujing Li, Yuanfan Li, Zhongkai Zhao, Chengquan Jiang, Faming Wu, Haotian Zhou, Jinlin Pang, Li Han, Qi Liu, Qianli Ma, Siyao Liu, Songhua Cai, Wenqi Fu, Xin Liu, Yaohui Wang, Zhi Zhang, Bo Zhou, Guoliang Li, Jiajun Shi, Jiale Yang, Jie Tang, Li Li, Qihua Han, Taoran Lu, Woyu Lin, Xiaokang Tong, Xinyao Li, Yichi Zhang, Yu Miao, Zhengxuan Jiang, Zili Li, Ziyuan Zhao, Chenxin Li, Dehua Ma, Feng Lin, Ge Zhang, Haihua Yang, Hangyu Guo, Hongda Zhu, Jiaheng Liu, Junda Du, Kai Cai, Kuanye Li, Lichen Yuan, Meilan Han, Minchao Wang, Shuyue Guo, Tianhao Cheng, Xiaobo Ma, Xiaojun Xiao, Xiaolong Huang, Xinjie Chen, Yidi Du, Yilin Chen, Yiwen Wang, Zhaojian Li, Zhenzhu Yang, Zhiyuan Zeng, Chaolin Jin, Chen Li, Hao Chen, Haoli Chen, Jian Chen, Qinghao Zhao, and Guang Shi. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning, 2025. URL `https://arxiv.org/abs/2509.02544`.

[64] Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Haotian Yao, Ziwei Chen, Qizheng Gu, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents, 2025. URL `https://arxiv.org/abs/2508.09123`.

[65] Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

[66] Yuxiang Wei, Zhiqing Sun, Emily McMilin, Jonas Gehring, David Zhang, Gabriel Synnaeve, Daniel Fried, Lingming Zhang, and Sida Wang. Toward training superintelligent software agents through self-play swe-rl. *arXiv preprint arXiv:2512.18552*, 2025.

[67] Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

[68] Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement. In *ICLR Workshop on Large Language Model (LLM) Agents*, 2024.

[69] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Jing Hua Toh, 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*, 37:52040–52094, 2024.

[70] Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. Rcot: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought, 2023. URL `https://arxiv.org/abs/2305.11499`.

[71] Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=6jZi4HSs6o`.

[72] Yuhao Yang, Zhen Yang, Zi-Yi Dou, Anh Nguyen, Keen You, Omar Attia, Andrew Szot, Michael Feng, Ram Ramrakhya, Alexander Toshev, et al. Ultracua: A foundation model for computer use agents with hybrid action. *arXiv preprint arXiv:2510.17790*, 2025.

[73] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

[74] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

[75] Charlie Zhang, Graham Neubig, and Xiang Yue. On the interplay of pre-training, mid-training, and rl on reasoning language models, 2025. URL https://arxiv.org/abs/2512.07783.

[76] Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, et al. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*, 2025.

[77] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=neZSGqhxDa.

[78] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

[79] Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. URL https://openreview.net/forum?id=23W5YZHGh9.

[80] Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. Lifelong learning of large language model based agents: A roadmap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

[81] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Table of Contents in Appendix

A Related Work **19**

A.1 GUI Agent 19

A.2 Autonomous Learning without Human Data 19

B Limitations **19**

C Broader Impacts **20**

D Background **20**

E Continual Learning Metrics and Analysis **20**

F Details of Curriculum Task Generation **21**

F.1 Iteration 0 21

F.2 Medium Level Adjustment 22

F.3 Context Reviews 22

F.4 Task Evolving Across Different Iterations 22

F.5 The impact of curriculum generator 22

G Generalization of ACuRL **23**

G.1 Generalization to Models with Different Initial Capabilities 23

G.2 Generalization to New Environments 24

G.3 Cross-Environment Ablation Study 24

H Details of CUAJudge **24**

H.1 Component Effectiveness and Efficiency 24

H.2 Effect of Self-Evaluation 25

H.3 Generalization to Diverse Scenarios 26

I Detailed Reliability Analysis of CUAJudge **26**

J Task Validity Analysis **27**

K The Parameter Update Overlap across Agents **27**

L Experimental Details **28**

M Comparison with Existing Methods **29**

N Failure Case Analysis **29**

O Detailed results for each environment **31**

P Prompts **33**

Q Examples of Context Across Different Environments **38**

A Related Work

A.1 GUI Agent

GUI agents operate directly on digital environments to accomplish user-specified tasks by integrating perception [20], planning [21], and reasoning [48]. Existing work has explored a wide range of interaction paradigms, spanning from DOM-based web manipulation [81, 14, 67, 47] to pixel-level control in desktop environments [3, 4, 48, 32]. Building on these paradigms, prior studies have significantly advanced the capabilities of GUI agents. However, they either heavily rely on large-scale human-annotated data [64, 35, 14, 63, 17, 23], which is costly and unscalable, or synthesize data from a limited set of environments [60, 47, 67, 72, 5], failing to capture the diversity of millions of desktop and web applications in the real-world scenarios. Moreover, they model the digital world as static, despite its inherently dynamic nature. In practice, interfaces and functionalities continually evolve [45, 71], which can lead to substantial performance degradation when such changes are not captured by prior training data [40, 13]. Although Liu et al. [36] studies real-world CUA scenarios where data is naturally in flux, its focus is primarily on grounding rather than end-to-end CUA task execution. Consequently, it does not capture the planning and multi-step decision-making required to solve realistic CUA tasks, which is a key distinction from our work. Moreover, they consider only two scenarios: platform migration (mobile, desktop, and web) and resolution changes. In contrast, our work addresses a broader and more realistic range of dynamics, including cross-environment continual learning scenarios and environment updates such as software version upgrades. Overall, we move beyond the static environment assumption and explicitly model real-world digital environments as dynamic and evolving, better reflecting practical deployment scenarios for end-to-end computer-use agents.

A.2 Autonomous Learning without Human Data

To reduce reliance on human effort, reinforcement learning [44, 52] has emerged as a popular paradigm, as it enables agents to learn from environment interactions via reward feedback without explicit human supervision [57, 58, 51]. While existing approaches show promise, transferring them to the computer-use domain remains challenging, as they target weakly grounded or environment-agnostic tasks that rely primarily on internal knowledge for synthesis. [34, 24, 77, 66, 33, 16]. In contrast, computer-use tasks must be grounded in interactive environments. Such environment-specific experiences are largely absent from pretraining data, making it difficult to synthesize high-quality tasks based on the model's internal knowledge alone. Moreover, these domains can easily obtain verifiable rewards, such as exact answer matching [12, 28] and deterministic program execution [77, 66, 65]; even open-ended instruction-following tasks can be alleviated via large-scale preference data [44, 8, 9]. By contrast, obtaining reliable reward signals in computer-use tasks is substantially more challenging [76], as task completion cannot be inferred from the final state alone, and trajectories typically span long horizons with rich multimodal observations (e.g., hundreds of screenshots), rendering trajectory evaluation itself a nontrivial problem [71]. Although SEAgent [60] explores autonomous learning for computer-use agents, it is developed under a simplified setting that deviates from real-world requirements. First, it does not account for environment dynamics, such as software updates, platform migration, and resolution changes, which introduce substantial distribution shifts in real-world interfaces. Second, it focuses on specialist agents within a single environment, overlooking cross-environment continual learning. In practice, both environments and user needs (e.g., new environments and increasingly complex goals) evolve over time, making cross-environment continual learning essential. Third, it adopts single-turn RL rather than end-to-end, multi-step RL. As a result, it fails to capture the emergent self-refinement and exploration behaviors that are essential for computer-use agents. In this work, we systematically study continual learning for CUAs, bridging the gap to diverse and dynamic digital environments in the real world, where agents continually learn in both intra-environment and cross-environment scenarios, while adapting to environment shifts (e.g., software updates, platform migration, and resolution changes) through a fully autonomous, end-to-end curriculum RL framework.

B Limitations

Although `ACuRL` achieves substantial gains in both intra- and cross-environment continual learning autonomously without human supervision, it still depends on some proprietary models (i.e.,

GPT-5-mini for evaluation and GPT-5 for task generation). While replacing these components with the open-source Qwen3-VL-8B also yields modest improvements, ACuRL still fails to enable fully autonomous self-evolution for agents with weaker instruction-following capabilities, such as UI-TARS. We leave this for future work, jointly enhancing GUI navigation and instruction-following, enabling agents to better support task generation and achieve true self-evolution.

C Broader Impacts

Autonomous computer-use agents have the potential to enhance productivity by assisting users in complex digital environments, such as office software and specialized applications. In this work, we study how such agents can continually learn and adapt to specific target environments without relying on human-annotated data. All experiments are conducted in controlled settings explicitly designed to prevent harmful actions on real-world systems. At the same time, computer-use agents are capable of executing multi-step actions that modify environment states, which may lead to unintended or potentially harmful behaviors if deployed without adequate safeguards, such as accidentally modifying or deleting user files. These risks raise important safety and ethical considerations, particularly for real-world deployment. We therefore believe that future research should further explore safety-aware evaluation protocols and effective risk mitigation mechanisms. Our code and models are released solely for research purposes, and we strongly discourage any use that could harm users or violate privacy.

D Background

We model a computer-use agent task execution as a Partially Observable Markov Decision Process (POMDP), where $\mathcal{S}$ denotes the environment state space, $\mathcal{O}$ the observation space, $\mathcal{A}$ the executable action space, $\mathcal{T}\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the environment transition function, and $\mathcal{R}$ the reward function. At step t, the agent receives a partial observation $o_t \in \mathcal{O}$, which may consist of a natural language task description together with observations such as screenshots. Based on the interaction history $h_t = (o_{\leq t}, a_{<t})$, the agent generates an executable action $a_t \in \mathcal{A}$, e.g., clicking a specific screen location or issuing a keyboard command, which transitions the environment to a new state $s_{t+1} \in \mathcal{S}$ and yields a new partial observation $o_{t+1} \in \mathcal{O}$. The interaction loop continues until the agent produces a termination action or reaches a predefined maximum number of steps. Finally, task completion is determined by an evaluator, who assigns an outcome $\mathcal{R}(\tau) \in [0, 1]$ based on the entire interaction trajectory τ. A value of $\mathcal{R}(\tau) = 1$ indicates successful task completion, while $\mathcal{R}(\tau) = 0$ indicates failure.

E Continual Learning Metrics and Analysis

To more directly quantify forgetting and cross-environment transfer, we further report standard continual learning metrics, including backward transfer (BWT), forward transfer (FWT), and forgetting relative to the best checkpoint on each earlier environment. We define these metrics following the standard continual learning formulation of Lopez-Paz and Ranzato [37]. Let N denote the total number of environments in the adaptation sequence, $R_{i,j}$ denote the performance on environment j after sequential adaptation through environment i, and b_j denote the base agent performance on environment j before adaptation.

$$\mathrm{BWT} = \frac{1}{N-1} \sum_{j=1}^{N-1} \left(R_{N,j} - R_{j,j} \right), \tag{4}$$

$$\mathrm{FWT} = \frac{1}{N-1} \sum_{j=2}^{N} \left(R_{j-1,j} - b_j \right), \tag{5}$$

$$\mathrm{Forgetting} = \frac{1}{N-1} \sum_{j=1}^{N-1} \left(\max_{i \in \{j,\ldots,N\}} R_{i,j} - R_{N,j} \right). \tag{6}$$

Higher BWT and FWT indicate stronger backward and forward transfer, respectively, while lower forgetting indicates better retention. We compute all metrics over the longest continual learning sequence for each base agent trained with ACuRL.

Table 5: Results of BWT, FWT, and forgetting over the longest continual learning sequence for each base model.

Metric	Impress	KAlgebra	Calc	Celestia	Writer	Avg.
UI-TARS-1.5-7B / Impress → KAlgebra → Calc						
BWT ↑	-6.8	-2.2	–	–	–	**-4.5**
FWT ↑	–	+2.1	+0.3	–	–	**+1.2**
Forgetting ↓	6.8	2.2	–	–	–	**4.5**
Qwen3-VL-8B / Impress → KAlgebra → Calc → Celestia → Writer						
BWT ↑	-1.9	+4.2	-1.0	+1.0	–	**+0.58**
FWT ↑	–	+7.5	+0.6	+4.1	+3.6	**+3.95**
Forgetting ↓	1.9	1.1	1.0	0.0	–	**1.00**

As shown in Table 5, UI-TARS-1.5-7B achieves an average BWT of -4.5, FWT of $+1.2$, and forgetting of 4.5. Qwen3-VL-8B exhibits better continual learning performance, with an average BWT of $+0.58$, FWT of $+3.95$, and only 1.00 point of forgetting. These results suggest that cross-environment continual learning of ACuRL can largely retain previously acquired capabilities while facilitating positive transfer across environments.

Table 6: Parameter-update sparsity for different base models.

Model	LLM Sparsity	Vision Encoder Sparsity
UI-TARS-1.5-7B	76.41%	78.34%
Qwen3-VL-8B	88.77%	83.80%

The greater forgetting observed for UI-TARS-1.5-7B may be associated with its lower parameter-update sparsity compared with Qwen3-VL-8B, suggesting that denser updates may induce greater interference with previously acquired capabilities. However, despite its sparser updates, Qwen3-VL-8B achieves a larger overall improvement than UI-TARS-1.5-7B (+9.7 vs. +6.4 points), indicating that greater update sparsity does not necessarily limit the acquisition of new capabilities. The relationship among parameter updates, knowledge retention, and skill acquisition may therefore also depend on model-specific factors such as architecture and pretraining data. However, we emphasize that these observations provide only plausible explanations rather than evidence of a verified causal mechanism. Overall, these results demonstrate that ACuRL can mitigate catastrophic forgetting while supporting effective cross-environment continual learning.

F Details of Curriculum Task Generation

F.1 Iteration 0

Notably, under our zero-data setting, no tasks are available at the beginning of training to assess the agent's initial capability. To address this, we let the generator G synthesize an initial task set based on context and exploration trajectories. We then conduct a capability evaluation on this task set to obtain feedback, which is subsequently used to generate the task set for Iteration 1, explicitly tailored to the current agent. We refer to this initialization phase as Iteration 0. During Iteration 0, the task generator focuses on synthesizing a set of atomic tasks that cover essential and frequently used atomic actions in target environments, thereby providing a foundational skill set for subsequent curriculum learning. Tasks from iteration 0 are not used for training; they are only used to assess the agent's initial capabilities and guide task synthesis in later iterations.

F.2 Medium Level Adjustment

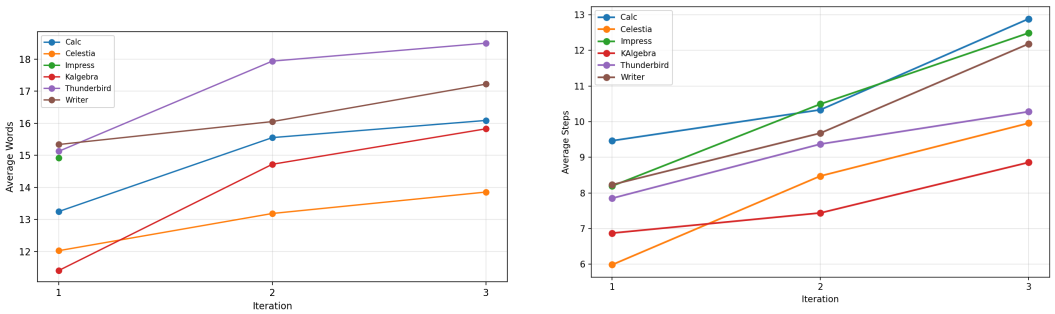
Tasks at a medium level suggest that the agent has not yet fully mastered the underlying skills required and can therefore continue to learn from them, expanding its edge of competence [75]. Accordingly, to maintain task diversity while supporting continual learning and skill consolidation, we let the curriculum generator produce tasks that exercise the same or similar skills across varied contexts, user personas, and task goals, but are realistic (e.g., completing the same document-editing operation under different constraints, such as layout requirements or formatting standards, or plotting charts based on different data.).

F.3 Context Reviews

For each environment, we collect diverse and substantially different contextual materials from the web. For example, in the LibreOffice Impress and Writer environments, we gather 1–3 contexts from multiple domains, including art, technology, social sciences, education, and scientific research. For environments where contexts are difficult to obtain or do not exist (e.g., KAlgebra), we instead let the agent explore the environment directly to acquire environment experience. From each context, we generate 32 tasks. The prompt for context review is "*You are given access to a software application in its current state. Your task is to actively and read-only explore the application to understand the full user context, including multiple distinct content areas, views, or sections, not just a single default view. You must navigate into and inspect representative content from each major content-bearing area (e.g., different folders, tabs, panels, lists, or sections) to verify what information exists and how it differs across areas. Do not modify, create, delete, send, save, or trigger any actions that persist changes. You must not finish after exploring only one section or view; only finish after you have reasonably covered the breadth of the application's content space.*".

F.4 Task Evolving Across Different Iterations

To better illustrate the quality of the synthesized tasks and how they evolve across iterations, we analyze both the average task length and the number of steps taken by the agent to complete tasks throughout training. As shown in Figure 6a and 6b, both metrics exhibit a consistent increasing trend over iterations, indicating that the generated tasks become progressively longer and, to some extent, more complex. This behavior aligns with our expectations: as iterative training improves the agent's capabilities, the curriculum generator produces increasingly challenging tasks that are better tailored to the agent's current ability. We further conduct a qualitative analysis and present representative examples in Table 24. These examples demonstrate that task complexity increases across iterations, evolving from operations on a single object to interactions involving multiple objects, and from requiring a single skill to composing multiple skills within a single task.



(a) Average instruction length across iterations.

(b) Average execution steps across iterations.

Figure 6: Task complexity increases across iterations in different environments.

F.5 The impact of curriculum generator

Some agents are specialized for GUI navigation and thus lack strong instruction-following capabilities, which prevents them from generating valid tasks or supporting curriculum-based learning

Table 7: Performance comparison of different generators on Impress at iteration 1, which involves curriculum learning from iteration 0 to 1, with Qwen3-VL-8B-Instruct as the base agent.

Generator	Impress
GPT-5	**30.8**
Qwen3-VL-8B	29.5

(e.g., UI-TARS). In ACuRL, we instead adopt GPT-5 as the generator due to its strong instruction-following ability. However, this does not imply that our performance gains rely on a more powerful external agent, as GPT-5 itself does not possess GUI navigation capabilities. To further assess the impact of the generator, we replace GPT-5 with Qwen3-VL-8B, an open-source model with strong instruction-following ability. As shown in Table 7, ACuRL remains highly effective, demonstrating that the observed gains do not stem from the frontier capabilities of GPT-5.

G Generalization of ACuRL

G.1 Generalization to Models with Different Initial Capabilities

To assess the applicability of ACuRL across different levels of initial agent capability, we evaluate agents with varying base performance and characterize the minimum competence required for effective autonomous learning. ACuRL begins with atomic tasks that involve only basic operations in the target environment rather than complex workflows. Therefore, as long as the base agent possesses basic GUI competence and can successfully complete a sufficient subset of these tasks, it can obtain informative learning signals that support effective continual learning.

Table 8: Initial capabilities of different agents on LibreOffice Impress and Iteration-0 atomic tasks.

Agent	Impress SR	Atomic SR	Difficulty Distribution (%)		
			Easy	Medium	Hard
UI-TARS-1.5-7B	**31.1**	**60.1%**	53.5%	16.0%	30.5%
Qwen3-VL-8B-Instruct	27.0	48.0%	37.5%	22.3%	40.2%
OpenCUA-7B	17.1	42.1%	26.2%	30.9%	43.0%
Qwen2.5-VL-7B-Instruct	0.0	0.5%	0.0%	0.0%	100.0%

Table 8 shows the performance of agents with different initial capabilities on LibreOffice Impress and the corresponding 256 Iteration-0 atomic tasks. UI-TARS-1.5-7B, Qwen3-VL-8B-Instruct, and OpenCUA-7B successfully complete meaningful subsets of the atomic tasks. In contrast, Qwen2.5-VL-7B-Instruct achieves only 0.5% average success on the atomic-task set and 0.0% on Impress, indicating insufficient basic GUI competence even for atomic operations. These results suggest that ACuRL does not require strong initial task performance, but does require the base agent to possess a minimum level of GUI competence to make progress on the initial curriculum.

Table 9: Performance of ACuRL with OpenCUA-7B on LibreOffice Impress.

Method	Impress
Base (OpenCUA-7B)	17.1
ACuRL (Iteration 1)	**22.0**

To further examine whether ACuRL remains effective in a lower-capability regime, we apply it to OpenCUA-7B, which has substantially lower initial performance on LibreOffice Impress than UI-TARS-1.5-7B and Qwen3-VL-8B-Instruct, while still completing a meaningful subset of the Iteration-0 atomic tasks. After one iteration, ACuRL improves OpenCUA-7B from 17.1% to 22.0% on LibreOffice Impress, an absolute gain of 4.9 percentage points. This result shows that ACuRL can effectively improve relatively weak base agents, as long as they possess sufficient capability to complete a nontrivial subset of the initial atomic tasks. Consistent with this finding, ACuRL also

yields clear gains when initial performance in the target environment is low, improving Qwen3-VL-8B-Instruct on KAlgebra from 9.7% to 26.9% and UI-TARS-1.5-7B on Calc from 6.5% to 9.0%. Together, these results demonstrate that ACuRL remains effective across a broad range of initial agent capabilities.

G.2 Generalization to New Environments

Given the diversity of environments encountered in real-world scenarios, we assess whether ACuRL remains effective beyond the environments considered in our main experiments (Figure 3). To this end, we evaluate it on GIMP, a specialized image-editing environment. As shown in Table 10, a single iteration of ACuRL improves Qwen3-VL-8B from 38.5% to 50.0% task success, corresponding to an absolute gain of 11.5 percentage points. These results highlight the potential of ACuRL to generalize across a broader range of diverse environments.

Table 10: Performance of ACuRL on the GIMP environment.

Method	GIMP
Base (Qwen3-VL-8B)	38.5 ± 0.0
ACuRL (Iteration 1)	$\mathbf{50.0 \pm 0.0}$

G.3 Cross-Environment Ablation Study

To examine the effectiveness of the key components of ACuRL in cross-environment continual learning, we conduct an ablation study on the LibreOffice Impress → KAlgebra sequence using Qwen3-VL-8B-Instruct. We compare ACuRL with variants that remove curriculum learning and both curriculum learning and iterative training under the same computational budget.

Table 11: Ablation of key components of ACuRL on the Impress → KAlgebra continual learning sequence.

Method	Impress	KAlgebra
ACuRL	**36.1**	**30.1**
w/o Curriculum Learning	31.7	16.1
w/o Curriculum Learning & Iterative Training	28.9	12.9

As shown in Table 11, removing curriculum learning reduces performance from 36.1% to 31.7% on Impress and from 30.1% to 16.1% on KAlgebra. Removing both curriculum learning and iterative training further reduces performance to 28.9% and 12.9%, respectively. These results further validate the importance of curriculum-driven task generation and iterative training as key components of ACuRL.

H Details of CUAJudge

Table 12: Overall comparison of WebJudge and CUAJudge on 1,444 OSWorld trajectories from different agents. Results are reported as mean ± standard deviation (%) over three runs.

Evaluator	Precision	Recall	F1	Agreement
WebJudge	70.4 ± 1.6	78.4 ± 0.4	74.2 ± 0.8	86.2 ± 0.5
CUAJudge	$\mathbf{73.6} \pm 0.8$	$\mathbf{80.8} \pm 1.2$	$\mathbf{77.0} \pm 1.0$	$\mathbf{87.7} \pm 0.3$

H.1 Component Effectiveness and Efficiency

Since the number of API calls in the key screenshot identification stage of CUAJudge scales linearly with the number of screenshots in a trajectory, the associated cost increases substantially. To

reduce the overall cost during reinforcement learning, we therefore compare the performance and cost trade-offs of using GPT-5-mini and Qwen3-VL-8B. As shown in Table 13, using Qwen3-VL-8B for the key screenshot identification stage achieves higher precision while reducing the cost by approximately 5×, incurring only $0.40 for evaluating 107 trajectories during RL training.

Table 13: Comparison of precision and cost across GPT-5-mini and Qwen3-VL-8B models for the key screenshot identification module.

Model	Precision	Cost ($)
Qwen3-VL-8B-Instruct	**70.3**	**0.4**
GPT-5-mini	65.6	2.0

We also conduct an ablation study to assess the role of the state difference analysis module. Specifically, we remove the component that compares the initial and final states in `CUAJudge` and exclude it from the outcome judgment. As shown in Table 14, removing this module leads to clear declines in both recall and overall agreement, highlighting its importance.

Table 14: Ablation of the state-difference module. **W/o state difference** omits state-difference analysis during outcome judgment. Results are reported as mean ± standard deviation (%) over three runs.

Evaluator	**Precision**	**Recall**	**F1**	**Agreement**
CUAJudge	73.6 ± 0.8	$\mathbf{80.8} \pm 1.2$	$\mathbf{77.0} \pm 1.0$	$\mathbf{87.7} \pm 0.3$
- w/o state difference	$\mathbf{74.1} \pm 1.0$	72.3 ± 0.6	73.2 ± 0.8	86.3 ± 0.4

H.2 Effect of Self-Evaluation

To investigate the potential of ACuRL for recursive self-improvement and assess whether its effectiveness stems from the framework itself rather than from reliance on a stronger external evaluator, we consider a self-evaluation setting in which Qwen3-VL-8B serves as both the policy model and the evaluator.

Table 15: Comparison of CUAJudge with different evaluator models.

Method	**Evaluator Model**	**Agreement**	**Precision**	**Recall**	**F1**
CUAJudge	GPT-5-mini + Qwen3-VL-8B	**87.69**	**73.63**	**80.77**	**77.03**
CUAJudge	Qwen3-VL-8B	85.53	69.07	79.62	73.97

We first compare the default and self-evaluation configurations of `CUAJudge` on 1,444 public OSWorld trajectories. In the default configuration, GPT-5-mini performs key-point identification and outcome judgment, while Qwen3-VL-8B performs key-screenshot identification. In the self-evaluation configuration, Qwen3-VL-8B performs all three stages. As shown in Table 15, the self-evaluation configuration achieves 85.53% agreement with human judgments, comparable to the 87.69% agreement achieved by the default configuration that incorporates the stronger GPT-5-mini model. This small gap suggests that the policy model itself can serve as a reliable evaluator and provide informative learning signals for self-improvement.

Table 16: ACuRL performance with different reward evaluators.

Policy	**Reward Evaluator**	**LibreOffice Impress**
Qwen3-VL-8B	Qwen3-VL-8B + GPT-5-mini	30.8
Qwen3-VL-8B	Qwen3-VL-8B	**31.7**

We next examine whether these self-evaluation signals can translate into effective policy improvement under ACuRL. Specifically, we train the agent for one additional iteration using ACuRL under the self-evaluation setting, where Qwen3-VL-8B serves as both the policy model and the reward evaluator, and compare it with the default `CUAJudge` configuration. As shown in Table 16, the

self-evaluation setting achieves a task success rate of 31.7% on LibreOffice Impress, compared with 30.8% when using the stronger evaluator. The slightly higher score may arise because a more accurate reward model does not necessarily yield greater policy improvement, while a less accurate model may facilitate more effective RL optimization by inducing greater reward variance, as observed by Razin et al. [49]. Alternatively, the small difference may also be due to stochastic variation in rollout quality during training. Overall, these results show that ACuRL can effectively improve the policy with the evaluation signals generated by the policy model itself, further supporting its potential for recursive self-improvement without relying on a stronger external evaluator.

We choose GPT-5-mini in the default `CUAJudge` setting primarily for computational efficiency. During the outcome judgment stage, `CUAJudge` may need to process trajectories containing up to 25 screenshots, while deploying an additional open-source model with comparable capability, such as Qwen3-VL-8B, would incur extra GPU overhead and increase the latency of the RL training pipeline. GPT-5-mini therefore provides a practical balance between evaluation quality, inference efficiency, and computational cost.

H.3 Generalization to Diverse Scenarios

We further evaluate the generalizability of `CUAJudge` across diverse environments, covering different task domains, applications, operating systems, and device platforms.

Table 17: Comparison of CUAJudge and WebJudge on AgentRewardBench. Results are reported as mean $\pm$ standard deviation over three runs.

Method	Agreement	Precision	Recall	F1	p-value
CUAJudge	**83.15 $\pm$ 0.29**	**76.59 $\pm$ 1.18**	**53.38 $\pm$ 0.00**	**62.91 $\pm$ 0.40**	**0.022 $<$ 0.05**
WebJudge	81.49 $\pm$ 0.58	73.17 $\pm$ 2.06	48.76 $\pm$ 0.78	58.52 $\pm$ 1.05	–

We first evaluate `CUAJudge` on AgentRewardBench, which contains 1,302 trajectories collected from five widely used web agent benchmarks: WebArena, VisualWebArena, AssistantBench, WorkArena, and WorkArena++. As shown in Table 17, `CUAJudge` consistently outperforms WebJudge across all evaluation metrics, achieving an agreement of 83.15%, compared with 81.49% for WebJudge. This improvement is statistically significant under a two-sided paired t-test across the five environments ($p = 0.022$), demonstrating that `CUAJudge` generalizes robustly across diverse web domains.

Table 18: Comparison of different evaluators on CUAStepBench.

Method	Accuracy	Precision	Recall	F1	Cost
Codex (GPT-5.5)	**79.10**	**87.10**	70.10	**77.70**	$\approx$53$\times$
WebJudge	67.99	71.88	63.45	67.40	$\approx$1$\times$
CUAJudge	74.82	76.60	**74.48**	75.52	1$\times$

To assess generalization beyond web environments, we further evaluate `CUAJudge` on CUAStepBench [61], which spans 177 applications across Web, Ubuntu, Windows, macOS, Android, and iOS. As shown in Table 18, `CUAJudge` substantially outperforms WebJudge across all four metrics and achieves performance comparable to Codex (GPT-5.5) while incurring over 50$\times$ lower evaluation cost. Together, these results demonstrate that `CUAJudge` generalizes effectively across diverse task domains, applications, operating systems, and device platforms.

I Detailed Reliability Analysis of CUAJudge

To assess the reliability of `CUAJudge`, we conduct a human evaluation on 432 trajectories uniformly sampled across six environments and training iterations. Table 19 reports agreement, false positives (FP), false negatives (FN), precision, and recall with respect to human judgments. Overall, `CUAJudge` achieves 93.7% agreement, 95.4% precision, and 92.9% recall, demonstrating strong consistency with human evaluation. Across environments, agreement ranges from 91.0% to 100.0%,

Table 19: Human evaluation of CUAJudge reliability across environments and training iterations. Results are reported against human judgments.

	Agreement ↑	FP ↓	FN ↓	Precision ↑	Recall ↑
Environment					
Writer	91.0%	5	2	87.2%	94.4%
Impress	93.2%	0	5	100.0%	85.7%
Calc	95.9%	2	1	93.9%	96.9%
Celestia	92.1%	1	5	97.2%	87.5%
KAlgebra	92.8%	2	3	96.4%	94.7%
Thunderbird	100.0%	0	0	100.0%	100.0%
Training Iteration					
Iteration 1	97.8%	1	2	98.7%	97.4%
Iteration 2	89.9%	5	9	93.2%	88.5%
Iteration 3	93.5%	4	5	94.2%	92.9%
Overall	**93.7%**	**10**	**16**	**95.4%**	**92.9%**

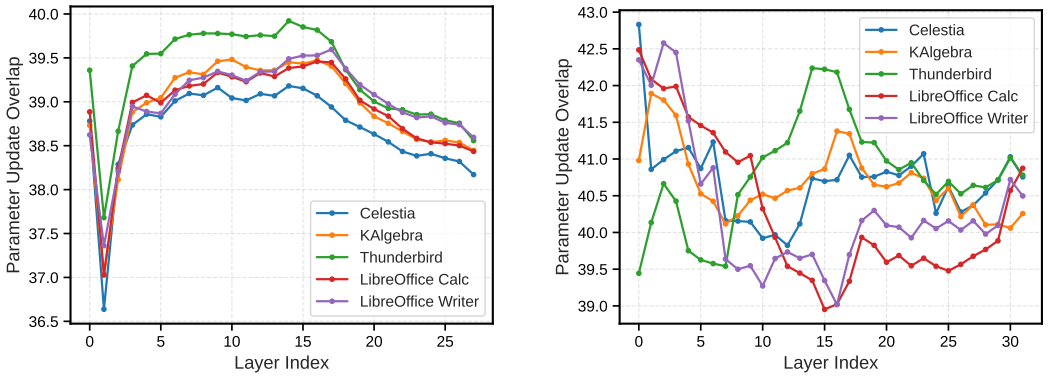
with no environment exhibiting substantial degradation, indicating that CUAJudge remains reliable across diverse application domains. Moreover, CUAJudge produces only 10 false positives across all 432 trajectories. Since false positives correspond to unsuccessful trajectories being incorrectly rewarded, their low frequency suggests a limited risk of reinforcing spurious or incomplete behaviors during RL. Reliability also remains high across training iterations, with agreement of 97.8%, 89.9%, and 93.5% in Iterations 1–3, respectively. Although agreement decreases in Iteration 2, it recovers in Iteration 3 rather than progressively deteriorating as training proceeds, providing no evidence that reward-hacking behavior increases in later training iterations.

J Task Validity Analysis

To quantify the quality of the synthesized tasks, we randomly sample 144 tasks across three training iterations from six different environments. We find that only 8 tasks are invalid, while 94% of the tasks are valid and explicitly grounded in environment-specific contexts. These results demonstrate that our task synthesis pipeline can scalably generate high-quality, environment-grounded tasks for target environments without relying on any human supervision. The eight invalid tasks primarily arise from ambiguous task descriptions, incomplete information, or cases where the environment does not provide the required functionality to support the specified task.

K The Parameter Update Overlap across Agents

To better understand why continual learning in one target environment also improves performance in others, we conduct a fine-grained analysis of the similarity in internal parameter updates across agents undergoing continual learning in different environments. Specifically, we compute the pairwise overlap ratio of significantly updated parameters between the LibreOffice agent and agents trained on other environments. This overlap ratio quantifies the proportion of parameters that are significantly updated in both agents, serving as an indicator of shared internal adaptation patterns across environments. As shown in Figure 7, both the LLM and vision encoder exhibit a 35–45% parameter overlap across different specialist agents. This high degree of overlap provides a plausible explanation for why adapting to a specific environment can also benefit other environments: despite differences in surface-level interfaces, these environments share underlying decision patterns and reusable skills that enable effective transfer. We also observe distinct layer-wise overlap patterns between the LLM and the vision encoder. For the LLM, the overlap increases steadily across the early layers, peaks in the middle layers, and then gradually declines in the higher layers. This trend suggests that the learning of common skills and underlying decision-making patterns is primarily concentrated in the middle layers. In contrast, the vision encoder exhibits a more irregular and less structured relationship between overlap and layer depth.



(a) LLM backbone

(b) Vision encoder

Figure 7: Overlap ratio of significantly updated parameters between LibreOffice Impress and other environments across different layers.

L Experimental Details

Training The number of API calls for key screenshot identification in `CUAJudge` scales linearly with the trajectory length. Therefore, we adopt Qwen3-VL-8B [7] for this component, which not only further reduces cost but also achieves higher precision. See Appendix H for details. For iterative reinforcement learning, we employ strict on-policy training with three iterations on 8 × H100 GPUs and use a server with 96 CPU cores and 384 GB memory for hosting the environments. Each iteration is trained for 75 steps using a batch size of 16 and a group size of 8, and a learning rate of 1×10^{-6}. For the capability evaluation stage, we maintain consistency with training by sampling 8 rollouts with 1.0 temperature and measuring the average success rate as feedback. The final checkpoint of each iteration is used to initialize the next iteration. Following Yu et al. [73] and Wang et al. [62], we remove the KL divergence and adopt a higher clipping ratio of 0.28. To mitigate context length limitations, we retain only the most recent two images in the action history of each trajectory. Our RL framework is implemented based on VeRL [55, 18]. For GRPO, we sample 8 trajectories per group with a temperature of 1.0. For the evaluator, following Xue et al. [71], we set the threshold for filtering key screenshots to 3. For categorizing task difficulty, we set δ_{low} and δ_{high} to 30% and 70%, respectively: tasks with an average success rate below 30% are considered difficult for the agent, those above 70% indicate tasks that the agent has largely mastered, and the remaining tasks are classified as medium difficulty. Our threshold choice follows Chen et al. [11]. In GRPO, advantages are computed relatively within a group, and learning is maximized when the success rate is around 50%. Therefore, we aim to keep task difficulty near this level during training. With a group size of 8, 3/8 and 5/8 are the closest practical approximations. As task difficulty increases alongside the agent's evolving capabilities, we iteratively increase both the maximum interaction steps and the maximum sequence length. Specifically, we use 15 steps with a maximum length of 13,000 tokens for iteration 1, 20 steps with 15,000 tokens for iteration 2, and 25 steps with 17,000 tokens for iteration 3. Regarding cost, we train the agent on 8×H100 GPUs. For environment hosting, we only require 96 CPU cores and 384 GB RAM to stably host 128 virtual machines. The generator cost is about $10 per iteration (i.e., $30 per environment for 3 iterations). For evaluation, we replace the intermediate steps in CUAJudge with Qwen3-VL-8B, reducing the cost by 5×. It costs only $0.4 to evaluate 107 trajectories, corresponding to about $0.48 per training step. With 225 steps in total (75 per iteration), the overall evaluation cost is approximately $108 per environment.

Evaluation During evaluation, we uniformly set the maximum number of environment interaction steps to 25. For the ScienceBoard evaluations on KAlgebra and Celestia, we make a minor adjustment. Specifically, some tasks require the agent to produce answers in a strict format (e.g., prefixed with ANS:). We observe that in many cases the model provides a correct answer but is marked incorrect solely due to formatting mismatches. While the official evaluation protocol addresses this issue by adding extra instructions to explicitly enforce the required output format, we intentionally

avoid introducing such additional instructions in order to strictly align with the OSWorld setting and evaluate the agent's performance without auxiliary guidance, where the agent receives only the task description and no additional guidance. To address this issue, we replace this small subset of format-sensitive tasks with an LLM-as-a-judge evaluation powered by GPT-5-mini, which is given the model's prediction and the ground-truth answer and determines correctness. We manually inspect a portion of these cases and find that the judgments are consistently accurate. Table 20 presents the task statistics for each environment. For human evaluation, we provide evaluators with the full trajectory, including all agent actions and screenshots. The evaluator explicitly flags any uncertain cases; for such instances, a second evaluator is introduced, and the final label is determined through discussion.

Table 20: Task statistics for each environment.

Environment	Task Source	#Tasks
LibreOffice Impress	OSWorld	47
	OfficeWorld	60
LibreOffice Calc	OSWorld	47
	OfficeWorld	60
LibreOffice Writer	OSWorld	23
	OfficeWorld	60
Thunderbird	OSWorld	15
Celestia	ScienceBoard	33
KAlgebra	ScienceBoard	31

M Comparison with Existing Methods

We compare ACuRL with recent curriculum-based approaches, including SEAgent [60], WebRL [47], and DigitalRL [6]. As shown in Table 21, ACuRL achieves competitive or superior performance across environments, with particularly large gains on Impress (+12.6%) and Celestia (+9.0%). Notably, ACuRL achieves these improvements with substantially lower computational cost. For instance, SEAgent requires supervised fine-tuning on synthesized trajectories followed by approximately 1,000 RL steps, whereas ACuRL attains superior performance using only 225 RL steps.

Table 21: Performance comparison with curriculum-based baselines on OSWorld and ScienceBoard, including DigiRL [6], WebRL [47], and SEAgent [60].

Method	Impress	Writer	KAlgebra	Celestia
DigiRL	19.6	19.1	-	-
WebRL	20.4	21.7	-	-
SEAgent	31.9	**43.5**	**29**	15.2
ACuRL	**44.5**	**43.5**	28	**24.2**

N Failure Case Analysis

Although ACuRL achieves substantial improvements in both intra-environment and cross-environment continual learning settings, a significant gap remains in achieving reasonable performance, with success rates in some environments still below 50%. To better understand these limitations, we conduct a detailed human analysis of our best-performing agent, which is continually trained across five environments to shed light on future development and improvement of CUAs. We categorize agent failures into six mutually exclusive types based on root-cause analysis: (1) Knowledge & Planning Limitations, where the agent lacks sufficient environmental knowledge of available UI elements and functionalities, or fails to formulate a coherent plan for compound multi-step tasks; (2) Premature Termination, characterized by the agent declaring success without verifying that the

resulting state satisfies the task specification; (3) Visual Grounding Error, arising from misidentification or mislocation of target UI elements during interaction; (4) Repetitive Action Loops, where the agent repeatedly executes the same action without producing any state change; (5) Task Misunderstanding, involving misinterpretation of the natural-language instruction's semantics; and (6) Others, covering miscellaneous failures that do not fall into any of the above categories. As shown in Figure 8, the failure distribution is dominated by knowledge and planning limitations (37%) and premature termination (30.4%), which together account for 67.4% of all observed failures. Visual grounding errors (11.9%) and repetitive action loops (11.5%) contribute comparably, with the latter highlighting the agent's inability to adaptively recover from unsuccessful actions. Task misunderstanding (7.0%) and other miscellaneous causes (2.2%) constitute a long tail of edge cases. Overall, these results indicate that the dominant bottlenecks lie in environment knowledge, task planning, and self-verification, rather than in low-level perception or instruction comprehension.

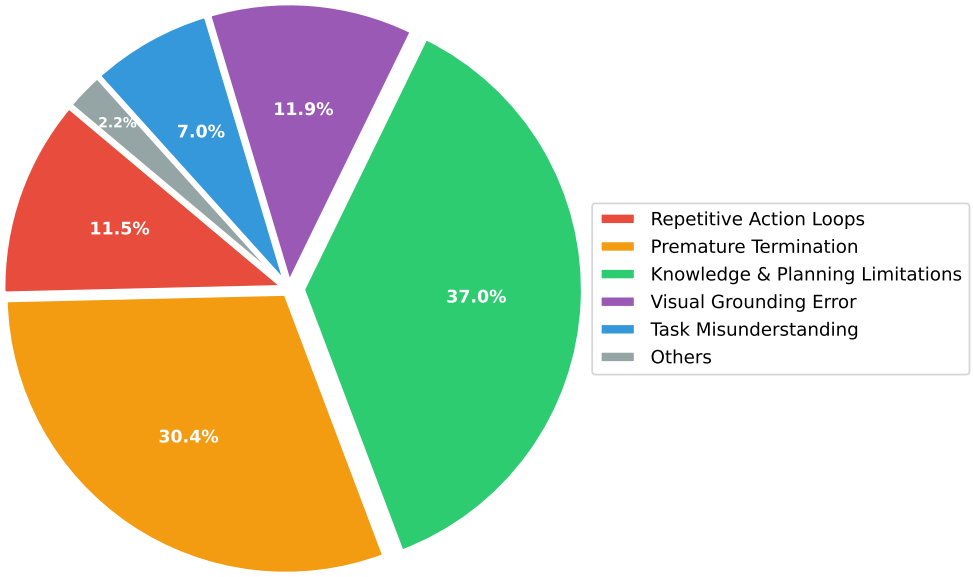


Figure 8: Distribution of failure types based on root cause analysis.

O Detailed results for each environment

To assess whether the improvement in overall performance is consistent across runs, we regress the Overall score on the number of completed environments for each independent run and perform a two-sided one-sample t-test on the resulting slopes. For UI-TARS-1.5-7B, the Overall score increases from 23.9 to 25.9, with a mean slope of $+0.95$ points per environment ($p = 0.008$). For Qwen3-VL-8B, it increases from 28.5 to 31.7, with a mean slope of $+0.90$ ($p = 0.008$). Both trends are statistically significant, indicating consistent performance improvement as the continual learning sequence expands.

Table 22: Average success rate (± standard deviation) across six environments for the base agent and agents that continually learn in specific environments via ACuRL. Rows correspond to agents continually learning in a specific environment and columns to testing environments. The overall score is the mean performance over all six environments. **Bold** denotes the best performance.

Agent	Iteration	Impress	Calc	Writer	Thunderbird	KAlgebra	Celestia	Overall
UI-TARS-1.5-7B (Base agent)	–	31.1±1.0	6.5±1.6	15.7±2.0	35.5±3.9	10.8±1.8	17.2±1.7	19.5±0.9
Impress	Iter 1	34.2±0.5	5.9±0.6	18.9±0.7	44.5±3.9	14.0±1.8	19.2±1.7	22.8±0.6
	Iter 2	38.8±0.6	8.1±0.5	21.7±0.0	51.1±3.8	14.0±1.8	20.2±1.7	**25.7**±0.3
	Iter 3	**40.7**±1.0	5.9±1.1	20.9±1.4	40.0±6.7	12.9±0.0	23.2±4.6	23.9±0.8
Calc	Iter 1	32.3±1.4	7.5±0.0	19.3±1.3	44.5±3.9	12.9±3.2	16.2±1.7	22.1±0.3
	Iter 2	31.7±1.0	7.8±1.1	18.5±0.7	46.7±0.0	19.4±3.3	13.2±3.5	22.9±1.2
	Iter 3	33.2±2.4	**9.0**±0.5	16.9±1.3	37.8±3.9	16.1±3.3	13.1±1.8	21.0±0.6
Writer	Iter 1	30.7±2.5	6.9±1.1	18.9±1.8	44.5±3.9	14.0±1.8	17.2±1.7	22.0±0.3
	Iter 2	32.9±0.5	5.6±0.9	20.5±1.2	51.1±3.8	10.8±1.8	17.2±4.6	23.0±1.5
	Iter 3	30.8±1.9	7.4±1.6	**22.5**±1.8	44.4±7.7	17.2±4.9	17.2±1.7	23.3±0.7
Thunderbird	Iter 1	31.1±1.9	6.6±1.0	17.7±0.7	42.2±3.9	10.8±3.7	16.2±1.7	20.8±1.6
	Iter 2	35.4±1.0	5.6±0.0	18.1±4.4	42.2±3.9	8.6±4.9	20.2±4.6	21.7±1.7
	Iter 3	35.1±1.9	7.5±1.0	13.7±0.7	**57.8**±3.9	14.0±3.7	22.2±4.6	25.1±0.4
KAlgebra	Iter 1	32.3±2.8	6.5±2.8	19.3±2.1	51.1±3.8	18.3±1.9	22.2±6.2	25.0±1.4
	Iter 2	29.8±2.5	7.1±1.1	16.8±1.3	46.7±0.0	20.5±1.8	19.2±4.6	23.4±0.4
	Iter 3	30.8±1.6	5.3±1.0	16.9±2.5	48.9±3.8	**28.0**±4.9	14.2±1.8	24.0±1.0
Celestia	Iter 1	33.6±2.4	5.6±0.0	22.1±0.7	48.9±3.8	11.8±1.8	22.2±1.7	24.0±0.2
	Iter 2	31.7±1.6	5.0±1.4	22.1±1.8	42.2±3.9	15.0±1.8	23.2±1.7	23.2±0.3
	Iter 3	34.2±1.6	7.5±1.0	22.1±0.7	48.9±3.8	15.0±1.8	**24.2**±3.1	25.3±0.4
Qwen3-VL-8B (Base agent)	–	27.0±1.6	15.3±1.9	18.9±2.5	51.1±3.8	9.7±3.2	10.1±1.7	22.0±0.7
Impress	Iter 1	30.8±2.5	14.4±1.1	21.3±0.7	66.7±0.0	11.8±1.8	6.1±0.0	25.2±0.6
	Iter 2	36.0±1.1	14.0±0.0	22.9±1.2	68.9±3.8	17.2±1.9	11.1±1.7	28.4±0.7
	Iter 3	**39.2**±1.0	14.0±1.9	24.9±1.8	64.5±3.9	17.2±1.9	11.1±1.7	28.5±1.2
Calc	Iter 1	25.2±1.6	16.2±0.5	20.5±1.3	66.7±0.0	10.8±1.8	10.1±1.7	24.9±0.7
	Iter 2	27.3±1.1	19.5±2.7	22.5±0.7	66.7±0.0	11.8±1.8	13.1±1.8	26.8±0.5
	Iter 3	33.0±1.9	**24.2**±2.5	26.5±1.2	62.2±3.9	10.8±1.8	11.1±1.7	28.0±0.4
Writer	Iter 1	26.4±0.5	13.4±0.5	23.3±0.7	68.9±3.8	10.8±1.8	10.1±1.7	25.5±1.3
	Iter 2	26.1±1.9	14.4±1.1	29.7±1.8	71.1±3.8	18.3±1.9	11.1±1.7	28.5±0.8
	Iter 3	27.7±0.5	17.1±0.6	**30.1**±3.2	68.9±3.8	12.9±3.2	15.2±3.1	28.7±1.0
Thunderbird	Iter 1	27.0±1.0	12.5±1.1	22.9±2.4	66.7±0.0	9.7±0.0	11.1±1.7	25.0±0.3
	Iter 2	29.5±1.4	9.4±1.0	20.1±1.4	71.4±3.2	11.8±1.8	8.1±1.7	25.1±0.7
	Iter 3	33.0±1.1	11.8±1.1	24.1±1.2	**80.0**±0.0	12.9±3.2	11.1±1.7	**28.8**±0.4
KAlgebra	Iter 1	27.3±4.4	19.3±2.0	21.7±1.2	42.2±3.9	11.8±1.8	6.1±0.0	21.4±0.3
	Iter 2	30.4±1.9	15.3±0.5	25.3±1.2	55.5±3.9	20.5±1.8	10.1±1.7	26.2±0.5
	Iter 3	30.1±2.1	17.1±1.4	28.5±0.7	62.2±3.9	**26.9**±1.8	8.1±1.7	**28.8**±0.6
Celestia	Iter 1	25.4±0.6	14.6±1.9	19.3±1.2	60.0±6.7	9.7±3.2	13.5±2.4	23.8±0.9
	Iter 2	27.0±1.0	14.3±1.1	20.9±0.7	58.1±8.3	10.8±1.8	17.2±1.7	24.7±1.8
	Iter 3	26.4±0.5	12.2±1.0	18.1±0.0	62.2±3.9	12.9±3.2	**18.2**±3.0	25.0±1.6

Table 23: Cross-environment continual learning performance of ACuRL. Results are reported as mean ± standard deviation over three runs.

Model	Agent	Impress	Calc	Writer	Thunderbird	KAlgebra	Celestia	Overall
UI-TARS-1.5-7B	Base	31.1±1.0	6.5±1.6	15.7±2.0	35.5±3.9	10.8±1.8	17.2±1.7	19.5±0.9
	Impress	**40.7±1.0**	5.9±1.1	20.9±1.4	40.0±6.7	12.9±0.0	23.2±4.6	23.9±0.8
	Calc	33.2±2.4	**9.0±0.5**	16.9±1.3	37.8±3.9	16.1±3.3	13.1±1.8	21.0±0.6
	KAlgebra	30.8±1.6	5.3±1.0	16.9±2.5	48.9±3.8	**28.0±4.9**	14.2±1.8	24.0±1.0
	Impress→Calc	37.0±1.2	8.7±1.4	**23.7±2.8**	44.5±3.9	16.1±0.0	19.2±1.7	24.9±0.2
	Impress→KAlgebra	36.7±1.1	6.8±1.4	21.7±2.4	46.7±6.7	18.3±1.9	20.2±4.6	25.1±0.0
	Impress→KAlgebra→Calc	33.9±0.5	6.6±0.1	23.3±1.8	**51.1±3.8**	16.1±0.0	**24.2±3.1**	**25.9±1.1**
Qwen3-VL-8B	Base	27.0±1.6	15.3±1.9	18.9±2.5	51.1±3.8	9.7±3.2	10.1±1.7	22.0±0.7
	Impress	**39.2±1.0**	14.0±1.9	24.9±1.8	64.5±3.9	17.2±1.9	11.1±1.7	28.5±1.2
	Calc	33.0±1.9	**24.2±2.5**	26.5±1.2	62.2±3.9	10.8±1.8	11.1±1.7	28.0±0.4
	Writer	27.7±0.5	17.1±0.6	**30.1±3.2**	68.9±3.8	12.9±3.2	15.2±3.1	28.7±1.0
	KAlgebra	30.1±2.1	17.1±1.4	28.5±0.7	62.2±3.9	**26.9±1.8**	8.1±1.7	28.8±0.6
	Celestia	26.4±0.5	12.2±1.0	18.1±0.0	62.2±3.9	12.9±3.2	**18.2±3.0**	25.0±1.6
	Impress→Calc	35.9±0.6	15.5±0.6	22.9±3.5	70.0±4.7	17.8±2.3	13.7±2.2	29.3±1.3
	Impress→KAlgebra	35.8±1.0	15.9±1.6	19.7±0.7	**71.1±3.8**	20.5±1.8	11.1±1.7	29.0±0.6
	Impress→KAlgebra→Calc	35.4±1.6	19.7±1.0	18.1±3.2	68.9±3.8	22.6±3.2	14.2±1.8	29.8±0.3
	Impress→KAlgebra→Calc→Celestia	33.9±0.6	19.3±1.9	22.5±4.5	**71.1±3.8**	25.8±3.2	16.2±1.7	31.5±1.2
	Impress→KAlgebra→Calc→Celestia→Writer	37.3±1.6	18.7±0.9	25.7±0.7	66.7±0.0	24.7±1.8	17.2±1.7	**31.7±0.2**

Table 24: Task examples from different environments across 3 iterations.

Iteration	Task Description
	LibreOffice Impress
1	Open Slide Properties for slide 5 and change the slide name to "Future Trends".
2	On slide 4, append a numbered list with exactly these items: 'Smart home devices', 'Personalized shopping', 'Healthcare services'.
3	On slide 5, add a straight connector from the apex of the pyramid to the text "Human-AI Collaboration".
	LibreOffice Writer
1	Insert a filled rectangle below the body text and type "Scan here" inside it.
2	Present the document in two columns while keeping the shaded block at the top as a full-width section above the columns.
3	After the line "How to learn this vocabulary", insert a cross-reference that displays the page number where "Graduate" appears.
	LibreOffice Calc
1	Apply a light gray background fill to the header cells A1:F1.
2	Create a filtered copy of all rows where Transmission (column E) equals "Manual" into a new sheet named Manual Cars.
3	Sort the table so Areas follow the custom order North, Midlands, Scotland, South-East, South-West while preserving the row integrity of each record.
	Thunderbird
1	Create a local folder named Reports 2025 and move both messages from the Bills folder into it.
2	Insert the clickable mail link mailto:assistant@outlook.com on its own line in the message body.
3	Create a Saved Search folder named "Recent Attachments" that lists messages with attachments from Inbox and Local Folders > Bills received within the last 60 days, and ensure this saved search appears in Favorites.
	KAlgebra
1	Compute $sin(0.5)^2$ and report the numeric value.
2	Visualize the 3D surface $z = x^2 + y^2$ on the 3D graph.
3	Show two 3D surfaces simultaneously: $z = x^2 - y^2$ and $z = sin(x) + sin(y)$.
	Celestia
1	Turn off the Deep Sky Objects layer so only stars and planets remain visible.
2	Jump to 2020-01-01 00:00 in local time, enable light-travel delay, and keep the simulation paused at that moment.
3	Display only the Galactic grid while keeping the Equatorial, Ecliptic, Horizontal grids and the ecliptic line hidden.

P Prompts

Curriculum Generator

You are a professional instructional designer specializing in software learning task design. Your objective is to generate exactly new {total tasks} high-quality, realistic learning tasks based on the previous feedback.

You will be provided with:
1. A record of random software exploration, including several actions and corresponding screenshots.
- This shows the possible functionality of the software.
2. An initial state of the software (with its own screenshots).
- This represents the exact starting point from which learners will begin.
3. A summary of learner performance on earlier tasks.
- This indicates which tasks were difficult, what the learner can already solve, and what gaps or weaknesses remain.

Software Exploration
Actions: {exploration actions}

The corresponding {len(screenshots)} screenshots across the actions: {screenshots}

Initial State
The current initial state is shown in {len(screenshots)} additional screenshots. These screenshots show the starting point for the software.

The initial state screenshots: {screenshots}

Task Generation Requirements
Based on the random exploration actions and screenshots, combined with the initial state screenshots, generate exactly {total tasks} specific tasks that:
1. **High-Level User Goals (NO Step-by-Step Instructions Allowed)**
- Write tasks as **natural user goals**, NOT tutorials or procedures.
- Express tasks the way a real user would describe what they want to achieve.
- Absolutely **do NOT describe UI navigation paths** (e.g., “click X”, “open Y”, “go to View ¿ Animation”).
- No operational verbs tied to UI mechanics (e.g., “open”, “select”, “navigate to”, “choose”).
- Use **intent-oriented phrasing**, not action-oriented phrasing. - Each task must be **one or two concise sentences** describing the desired outcome or goal.

2. **Anchor in Initial State**
- Each task must assume the learner starts from the provided initial state screenshots.
- Do not invent extra elements not visible in the initial state.

3. **Leverage Exploration Evidence**
- Use the random exploration actions/screenshots to infer what the software can do.
- Design tasks that are plausible and valuable given those capabilities.

4. **Realistic User Value**
- Tasks should reflect meaningful activities a real user of this software might want to accomplish, not artificial button-clicking exercises.

5. **Distinct Coverage**
- Each of the task should focus on a different aspect, function, or workflow of the software.
- Avoid overlap. Aim for variety (e.g., editing, organizing, exporting, formatting).

6. **Diversity**
- Avoid creating tasks similar to the existing ones from previous feedback.
- Generate creative variations and novel approaches.
- Ensure variety in complexity, action types, and user scenarios.
- Think of different user personas and use cases.

Curriculum Generator

7. **Clarity and Specificity (Anti-Ambiguity Rule)**
All regenerated tasks must be stated with full clarity and without ambiguity.
To ensure this:
- Do not use vague or illustrative expressions such as “for example”, “such as”, “any”, “multiple”, or other open-ended terms.
- Do not introduce optionality, alternatives, or flexible ranges (e.g., “A or B”, “choose whichever”, “a few”, “some”).
- Do not reference hypothetical or representative instances.
- Every task must define a **single, explicit target** and a **single, unambiguous goal**.
- The task must be specific enough that a system can determine **exact completion criteria** without guessing.

8. **Feedback Integration**
You must generate a substantively different new task based on the agent’s SR.
The new task must not be a minor revision, paraphrase, or parameter change of the original task.

If SR $>$ 70% (High performance)
The agent performs the original task reliably. You must therefore generate a **fully new** task that is **significantly more challenging**, with **no structural, semantic, or phrasing resemblance** to the original.

The new task must satisfy all of the following:

1. Fully New Scenario
- Use a **completely different real user motivation**, context, and type of content.
- Do NOT share the original task’s goal type, UI element category, action verbs, workflow pattern, or problem structure.

2. Higher Complexity
The new task must be **meaningfully more complex**, for example by: - involving **multiple related sub-goals**,
- introducing **conditional or constraint-based goals**,
- requiring **cross-region or cross-document coordination**,
- adding **explicit constraints, requirements, or trade-offs**,
- or requiring **richer interpretation or reasoning** about the interface/content.

3. Strict Non-Similarity
The new task must NOT:
- resemble the original in structure or step pattern,
- reuse similar keywords, verbs, UI components, or operations,
- operate in the same functional space as the original (e.g., if original used formatting, avoid formatting),
- pursue any version of the same underlying objective.

The final task must feel like an **entirely new, high-value, realistic scenario** that a different user might genuinely perform.

If 30% $\leq$ SR $\leq$ 70% (Medium performance)
The agent shows partial competence. You must generate a **different task of similar overall difficulty**, but with a **distinct scenario and distinct user intent**.

You may:
- explore a **new real user scenario**, or
- create a **lateral variation** that practices similar reasoning skills without repeating the original pattern.

Ensure meaningful variation by changing at least one of:
- the **user persona**,
- the **purpose or motivation**,
- the **content being worked on**,
- or the **overall context** (document type, audience, situation).

The new task must **not** be a softened or strengthened rewrite of the original.

If SR $<$ 30% (Low performance)
When the agent struggles significantly, generate a **scaffolded prerequisite task** that focuses on the fundamental skill(s) required to perform the original task.

This scaffolded task must:
- isolate **one essential underlying capability** needed to accomplish the original task,
- be formulated as an **independent, authentic user goal**, not a simplified or partially completed version of the original task,
- introduce a **distinct scenario, purpose, or context**, while still remaining within the same broad capability domain,
- and explicitly target the foundational competencies the agent lacks (e.g., identifying relevant elements, applying targeted adjustments, interpreting context, selecting appropriate options, or manipulating core components).

The new task should reflect a **core sub-skill** that the user must master before attempting any higher-level or more complex versions of the original task.

{feedback section}
Return the tasks in the following JSON structure:
"'json {{ "tasks": [{{ "task_id": "task_1", "original_task_description": "The original task description from the previous feedback", "reasoning": "Explain why and how this task was modified given the agent's SR and feedback context.", "new_task_description": "High-level yet specific user goal derived from the initial state and refined through feedback." }},
{{ "task_id": "task_2", "original_task_description": "The original task description from the previous feedback", "reasoning": "Explain why and how this task was modified given the agent's SR and feedback context.", "new_task_description": "High-level yet specific user goal derived from the initial state and refined through feedback." }},
{{ "task_id": "task_3", "original_task_description": "The original task description from the previous feedback", "reasoning": "Explain why and how this task was modified given the agent's SR and feedback context.", "new_task_description": "High-level yet specific user goal derived from the initial state and refined through feedback." }},
{{ "task_id": "task_4", "original_task_description": "The original task description from the previous feedback", "reasoning": "Explain why and how this task was modified given the agent's SR and feedback context.", "new_task_description": "High-level yet specific user goal derived from the initial state and refined through feedback." }},
.....] }} "'

CUAJudge - Outcome Judgement

You are an expert evaluator of computer use agents. Your task is to rigorously assess whether an agent has successfully completed a given task by examining the initial state, final state, action history, and intermediate screenshots.

Evaluation Process

Follow this systematic evaluation process:

Step 1: Understand the Task Requirements
- Read the task description carefully and identify what needs to be accomplished
- Review the key points that define successful completion
- Examine the initial screenshot(s) to understand the starting state

Step 2: Compare Initial vs Final State
- **Critical**: Compare the initial screenshot(s) with the final screenshot side-by-side
- Identify what specific changes occurred in the interface
- Determine if these changes align with the task requirements
- Check if the final state demonstrates completion of all required modifications

Step 3: Verify Each Key Point
For EACH key point listed, systematically check:
- Is this requirement explicitly addressed in the action history?
- Is the result visible in the final screenshot or intermediate screenshots?
- Is the implementation correct (not just superficially similar)? - Are the exact specifications met (correct values, correct targets, correct methods)?

Step 4: Apply Strict Evaluation Criteria
1: The agent must only make changes that are directly necessary to complete the task. Any unrelated edits or system/software modifications that are not explicitly required for task completion will result in failure.
2: Interpret the user's instructions in the context of the current software. Professional or common software terms have precise meanings in their respective applications. An action that superficially matches the wording but targets the wrong function or interface region of the software should be considered a failure.
3: You must carefully check whether these snapshots and action history meet these key points. Ensure that specific filter conditions, such as "best," "highest," "cheapest," "latest," "most recent," "lowest," "closest," "highest-rated," "largest," and "newest" are correctly applied using the filter function (e.g., sort function).
4: If the agent loops through a sequence of actions that do not make progress toward the goal (including failing to click "Save" or "Submit," etc.), it should be considered a failure.

Step 5: Make Final Judgment

SUCCESS requires:
- ALL key points are satisfied
- ALL evaluation criteria are met
- Final screenshot shows correct end state
- No extraneous or incorrect modifications

FAILURE if ANY of these are true:
- Even one key point is not satisfied
- Any evaluation criterion is violated
- Final state does not match requirements
- Evidence of incorrect or incomplete execution

Response Format

Provide your evaluation in exactly two lines:

Thoughts: [Systematically walk through Steps 1-5 above. For each key point, explicitly state whether it's satisfied and cite evidence from screenshots/actions. Compare initial and final states. Note any violations of criteria.]
Status: "success" or "failure"

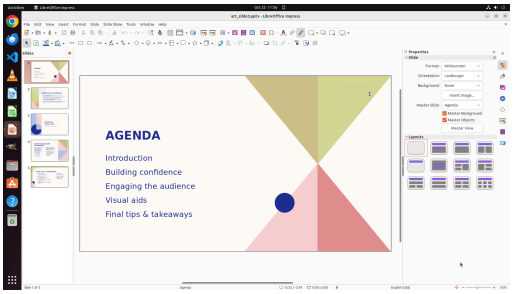
Important: Be thorough but concise. Your thoughts should demonstrate you checked each key point and criterion.

Task to Evaluate: {task}

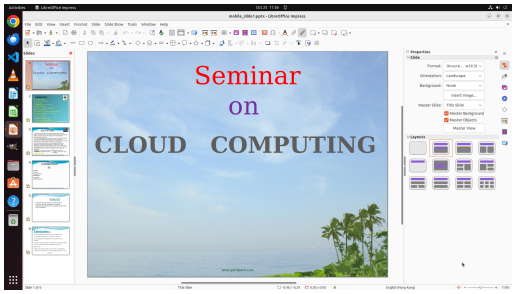
Key Points for Successful Completion: {key points}

Agent's Action History: {action history}

Q Examples of Context Across Different Environments

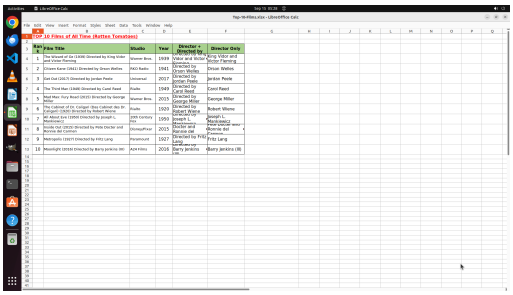


(a) Example 1

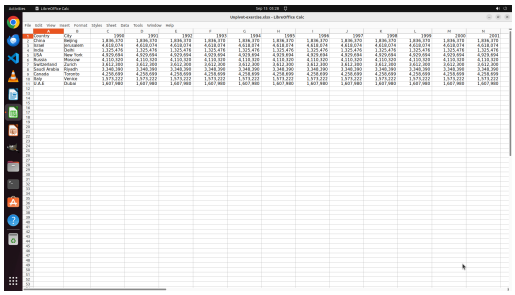


(b) Example 2

Figure 9: Examples of LibreOffice Impress context.



(a) Example 1



(b) Example 2

Figure 10: Examples of LibreOffice Calc context.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction accurately reflect our contributions and scope in continual learning for computer-use agents. We propose ACuRL, an autonomous curriculum reinforcement learning framework that enables effective intra- and cross-environment continual learning, while substantially mitigating the impact of real-world environmental dynamics, such as version updates, platform migration, and resolution changes. In §4, we evaluate ACuRL on two base agents and observe absolute improvements of 3–29% across both intra- and cross-environment continual learning settings, without catastrophic forgetting, all in a fully autonomous setting without human supervision.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a limitation section in Appendix B that discusses ACuRL's reliance on proprietary models.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We do not introduce novel theoretical results requiring formal proofs. Instead, it builds upon standard reinforcement learning formulations and well-established optimization objectives. For completeness and clarity, we explicitly present all relevant formulations, assumptions, and training objectives used in our method. Since our contributions are primarily methodological and empirical, no additional theoretical assumptions or proofs are required.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all necessary experimental details, including model settings, training configurations, environment setups, and evaluation protocols, in the main paper and supplementary material to enable reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code, training scripts, and evaluation pipelines are included in the supplementary material. We provide detailed instructions for environment setup, training configurations, and evaluation procedures, along with all necessary hyperparameters, enabling full reproduction of the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Appendix L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean and standard deviation over 3 runs for the main results in Table 22.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide details of the computational resources used for our experiments, including hardware configurations (e.g., GPUs and CPU resources), environment infrastructure, and execution setup in the Appendix L.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The submission conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have included a Broader Impacts section in Appendix C.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly cite all external assets used in this work, including datasets, models, and software environments. We follow the corresponding licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: All details are comprehensively documented in the README included in the supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work involved no crowdsourcing experiments or research with human subjects. The human evaluation procedure was conducted by authors. We provide details in the Appendix L, including instructions and labeling protocols.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research did not involve human subjects. The data used in experiments consisted entirely of simulated data and publicly released datasets. Accordingly, IRB review was not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: We clearly describe the use of LLMs in our method, including their roles in task generation and evaluation, in the main paper and supplementary material.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.