GAIA: A Data Flywheel System for Training GUI Test-Time Scaling Critic Models

Shaokang Wang *[1,2], Pei Fu[1], Ruoceng Zhang[1], Shaojie Zhang[1], Xiuwen Xi[1], Jiahui Yang[1], Bin Qin[1], Ying Huang[1], Zhenbo Luo ✉[1], and Jian Luan[1]

[1] MiLM Plus, Xiaomi Inc, Beijing, China
[2] School of Computer Science, Peking University, Beijing, China
skwang6272@stu.pku.edu.cn, {fupei1, luozhenbo}@xiaomi.com

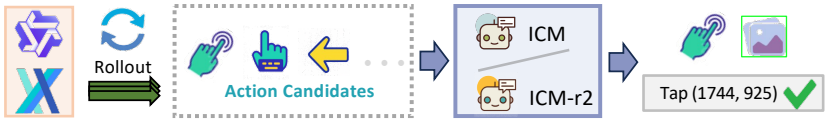
Abstract. While Large Vision-Language Models (LVLMs) have significantly advanced GUI agents' capabilities in parsing textual instructions, interpreting screen content, and executing tasks, a critical challenge persists: the irreversibility of agent operations—where a single erroneous action can trigger catastrophic deviations. To address this, we propose the **G**UI **A**ction Cr**i**tic's Dat**a** Flywheel System (GAIA), a training framework that enables the models to have iterative critic capabilities, which are used to improve the Test-Time Scaling (TTS) of basic GUI agents' performance. Specifically, we train an **Intuitive Critic Model** (ICM) using positive and negative action examples from a base agent first. This critic evaluates the immediate correctness of the agent's intended actions, thereby selecting operations with higher success probability. Then, the initial critic guides agent actions to collect refined positive/negative samples, initiating the self-improving cycle. The augmented data then trains a second-round critic with enhanced discernment capability. We conduct experiments on various datasets and demonstrate that the proposed ICM can improve the test-time performance of various closed-source and open-source models, and the performance can be gradually improved as the data is recycled. The code, dataset, and accompanying datasheet will be publicly released at https://github.com/SeerRay-Lab/GAIA.

Keywords: VLM · GUI Agent · Critic Model

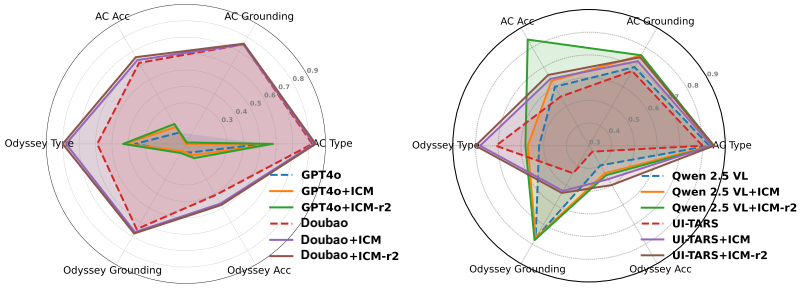
1 Introduction

The automation of Graphical User Interface (GUI) interactions represents a critical frontier in developing intelligent digital assistants [12, 27, 41]. Recent breakthroughs in Large Vision-Language Models (LVLMs) [1, 40], leveraging advanced post-training techniques, have substantially enhanced agents' capabilities in interpreting natural language commands, perceiving visual elements, and executing multi-step tasks [6, 11]. Within this rapidly evolving landscape, the development of robust GUI agents has largely converged on two primary methodological paradigms. The first approaches [20, 31, 46, 50] train models

* This paper was completed during the internship at Xiaomi. ✉ Corresponding author.



(a) The promotion process of the critic model to the GUI Agent during testing.



(b) Comparison of high-level task performance improvements on closed-source and open-source GUI Agents.

Fig. 1: Intuitive results. ICM guides agents' action during testing, as shown in (a), thereby improving the agent's accuracy, and is continuously improved by the data flywheel, as shown in (b).

through Supervised Fine-Tuning (SFT) to directly align their behavior with predefined task objectives. The second approaches employ Reinforcement Fine-Tuning (RFT) [21,24,47], which significantly enhances generalization in complex tasks by adopting a reasoning format.

Despite these advances, the dynamic and continuous nature of real-world GUI tasks means that agents can still produce ambiguous or incorrect action proposals at any step. **A single mis-click or mis-typed output can be irreversible**, derailing the entire workflow and leaving the system in an unrecoverable state. This high-stakes environment imperatively demands **a mechanism for pre-execution validation.**

To avoid irreversible errors in execution and improve the performance of basic GUI agents during testing, previous studies have designed action verifiers for GUI agents [45, 48, 51], which are used to judge and filter GUI agents' actions. However, these existing implementations suffer from two primary limitations. First, training a correctness verifier requires defining positive and negative action samples. Existing work on defining negative samples relies on heuristic algorithms, such as randomly selecting click locations on the current screenshot [48], which fails to capture the realistic action distribution and leads to suboptimal judgment performance. Second, the reasoning-based verifiers [43] implemented in existing work violate the intuitive properties of binary judgments. For an intuitive correctness judgment problem, biological research suggests that higher-level judgment pathways are often more adept than performing extensive multi-step reasoning [8, 18, 29], which indicates that excessive reasoning can be less effective [2,39]. Furthermore, reasoning-based judgment outputs more tokens, thereby reducing the efficiency of test-time scaling.

To fully leverage pre-execution evaluation to enhance GUI agent capabilities and execution correctness, we developed a **G**UI **A**ction Cr**i**tic's Dat**a** Flywheel System (GAIA). This system comprises two core phases: the initialization phase (Phase 1) and the iteration phase (Phase 2), yielding the **Intuitive Critic Model** (ICM). In **Phase 1**, we use real GUI agents to act on an existing dataset to collect positive and negative action data that are random but consistent with the behavior distribution. Using this binary-labeled action dataset, we train ICM to assess action correctness given environmental context. In **Phase 2**, as illustrated in Figure 1(a), ICM employs a Best-of-N approach to select the highest-probability correct actions from agent rollouts. While ICM guidance significantly improves action accuracy, challenging samples persist and produce errors. These difficult cases are annotated and fed back into the data flywheel. Through iterative data augmentation, the flywheel continuously incorporates new action samples, progressively covering challenging scenarios within the action space. Driven by this enriched dataset, we train an enhanced critic—Intuitive Critic Model on Round Two (ICM-r2)—which achieves higher discriminative accuracy for more precise behavioral guidance. This establishes a self-evolutionary virtuous cycle between the data flywheel and critic models, continuously improving GUI agent action accuracy.

Leveraging the proposed system GAIA, ICM achieves SOTA performance in action critique. Naturally, we integrate it into Test-Time Scaling (TTS) [5, 30, 34, 35, 38, 42] during inference, where ICM evaluates stochastically generated actions from the TTS process, releasing only high-confidence operations surpassing predetermined thresholds for execution. To validate the framework's general applicability, we conduct joint experiments using mainstream GUI Agents (including GPT-4o [13] and UI-TARS [31]) on several GUI Agent benchmarks. As shown by the comparative results in Figure 1 (b), the guidance from our iteratively evolved critic models (ICM and ICM-r2) leads to significant performance improvements in basic GUI agents, including GUI operation task planning and grounding capabilities.

Overall, the main contributions are summarized as follows:

1. We introduce GAIA—a novel Data Flywheel System designed for training GUI action-critic models. By iteratively curating positive and negative samples from real-world action data, GAIA continuously boosts model performance and robustness.
2. We propose the ICM for GUI interaction tasks, a critic model trained on data curated by our data flywheel. The ICM enhances the performance of existing GUI agents by employing a best-of-N approach to select the most probable correct action with TTS. This initial boost is then continuously refined as the ICM's discriminatory accuracy is iteratively improved by the data flywheel.
3. We comprehensively demonstrate across multiple datasets that ICM trained with our proposed GAIA system significantly enhances the overall performance of both closed-source and open-source GUI agents.

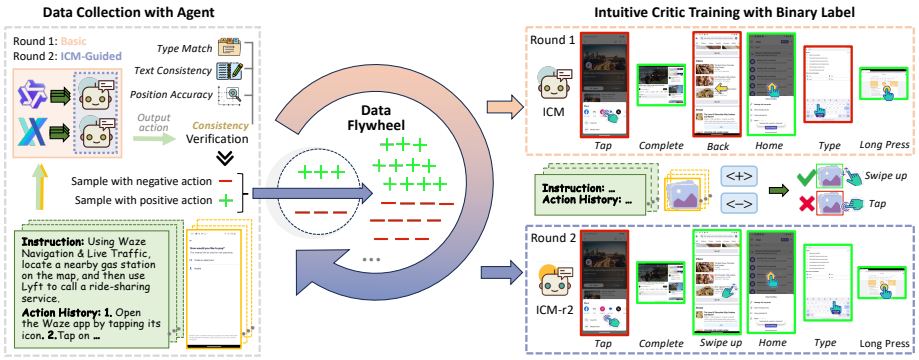


Fig. 2: Data flywheel curation pipeline for GAIA. A sample dataset is constructed using GUI agent interactions. The positive and negative labels are marked by comparing the ground truth actions to train an action correctness discrimination model. After the critic model guides the GUI agent, it further expands the dataset, pushing the data flywheel to cover more action distributions, thereby promoting the iterative improvement of model performance.

2 Related Work

2.1 GUI Agent

The development of autonomous agents powered by LLMs and LVLMs has significantly advanced interactive functionalities within digital environments. Early GUI systems primarily leveraged LLMs to interpret structured representations [11, 28, 36]. The development of LVLM simplifies the paradigm, allowing GUI agents to receive raw visual signals from the screenshots [7, 9, 12, 19, 33, 37, 46, 56]. Recent efforts, such as Aguvis [50] and UI-TARS [31], have advanced autonomous GUI navigation by integrating explicit planning, sophisticated reasoning, and GUI-specific pretraining to handle complex digital environments. Concurrently, the advent of rule-based Reinforcement Learning (RL) approaches [10, 14] has further enhanced GUI agent capabilities. These RFT methods improve reasoning and generalization by enabling models to learn universal action strategies from high-quality samples [21, 22, 24, 32, 47]. While fine-tuning and model scaling can enhance GUI agent capabilities, these methods are often prohibitively resource-intensive. This highlights a clear need for test-time enhancements that can offer universal performance improvements across various agent models without costly retraining.

2.2 Critic Model

To solve the problem of suboptimal single-shot model output [4, 25, 44, 54], research has gradually focused on improving the performance of the basic model during testing with the help of the critic model [15, 16, 26, 49, 53]. This concept has

been expanded to the GUI domain with notable works like GUI-Genie [48], GUI-Actor [45], GTA1 [51], and GUI-Critic-R1 [43]. However, existing GUI critics often rely on synthetic data generated by heuristic algorithms, such as randomly selecting click locations [45], cross-task substitution, or early truncation [48]. This approach fails to accurately simulate the complex behavior of real GUI agents across the full action space, thereby preventing the critic from learning faithful discrimination criteria. Furthermore, while some approaches use RL to inject reasoning capabilities into the critic [43], this often contradicts the very motivation for intuitive judgment [18,39] and introduces delays due to extended output token generation.

3 Method

In this section, we detail the design of our data flywheel-driven GAIA system for the GUI agent shown in Figure 2. We begin in Section 3.1 by introducing the general definition of the GUI agent task and the crucial role of the critic model. Section 3.2 delves into the design and application of our data flywheel system within the initial round of the evaluation process. In Section 3.3, we present the model training in the second round, which builds upon the outcomes from the first iteration and forms a virtuous cycle.

3.1 Preliminaries

The interaction between a GUI agent and its environment can be formulated as a Markov Decision Process (MDP), denoted by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{T}, \mathcal{O} \rangle$. Here, $\mathcal{S}$ defines the state space of possible screen states, while $\mathcal{A}$ encompasses the action space, including interaction types like clicking, typing, and scrolling. The observation space $\mathcal{Z}$ captures inputs such as screenshots or structured UI representations. The state transition probability is given by $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, mapping a state and action to a new state. Similarly, $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ describes the likelihood of observing a particular output given a state and an action. During GUI task execution, at each discrete time step t, the agent receives an input tuple (z_t, u, h), comprising the current screen observation $z_t \in \mathcal{Z}$, the global task instruction u, and the accumulated interaction history h. The agent's decision-making process for GUI actions is then formalized by a structured policy function $\mathcal{F}$:

$$\mathcal{F}(z_t, u, h) \to o_t = \{a_t, c_t\}, \tag{1}$$

where o_t represents the agent output at time t, consisting of the action type a_t (e.g., *click*, *swipe*, and *type*) and its corresponding parameters c_t (e.g., click coordinates, text content for typing). After a_t is executed, the environment transitions to a new state z_{t+1}, and this iterative process continues until the task is successfully completed or a predefined termination condition is met.

The proposed ICM, building upon the same observations and the GUI agent's current proposed action o_t, outputs a judgment j_t regarding the correctness of

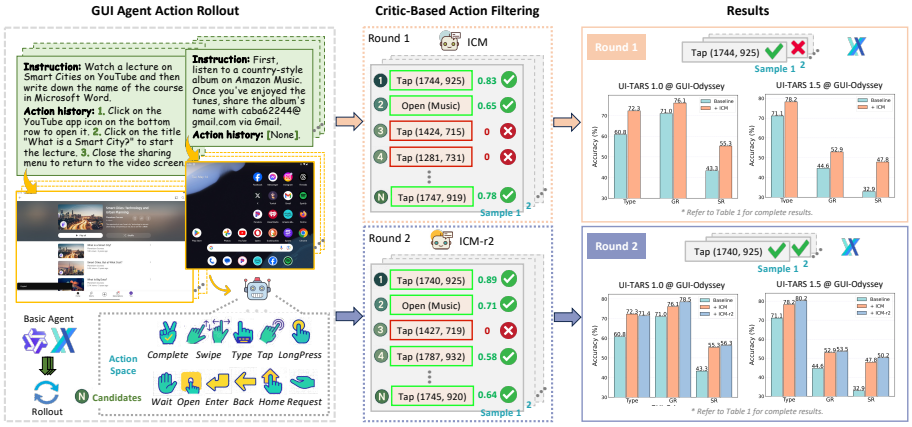


Fig. 3: Test-Time scaling pipeline. Through best-of-N rollout, multi-candidate actions of GUI agents are given, and the correct action with the highest probability is selected after ICM evaluation.

that action:

$$\mathcal{J}(o_t|(z_t, u, h)) \rightarrow l_t = \{j_t, p_t\}, \tag{2}$$

where j_t is a binary indicator, "*correct*" for correct actions and "*wrong*" for incorrect, p_t represents the probability of the judgment, which supports finding the correct action with the highest confidence. By enabling the sampling of multiple candidate actions and prioritizing them based on their respective correctness probabilities, ICM ensures that a more optimal action for the current state is selected and executed, significantly enhancing the agent's actual success rate.

3.2 Action Decision with Intuitive Critic

Data Curation To enable the judgment model to distinguish the correctness of real actions, we meticulously define both positive and negative samples of GUI agent actions. We begin by having existing GUI agents $\Pi = \{\pi_1, \pi_2, ..., \pi_i\}$ [31] interact with and traverse publicly accessible datasets [17, 23], allowing us to collect authentic, step-level operations across various GUI scenarios. For each action executed in a specific state (z, u, h), we then leverage ground truth labels to determine its correctness. An action is designated positive (with a correctness judge $j =$ "*correct*") if it aligns with the GT.

Conversely, we identify negative samples (with a correctness score $j =$ "*wrong*") based on states where the agent's action deviates from the GT. This approach ensures that our collected negative operations are closely aligned with the actual error distribution observed in real GUI environments, significantly enhancing the quality and realism of our training dataset. To prevent bias during ICM training, we balance the collected positive and negative samples, ensuring an equal 50% split for each. This carefully curated dataset, denoted as $\mathcal{D} = \{j_k|z_k, u_k, h_k, o_k^{\pi_i}\}_{k=1}^{K}$, forms the foundation of our data flywheel GAIA.

Table 1: Data distribution of the flywheel. $\mathcal{D}$ and $\mathcal{D}^+$ respectively represent the data of the first and the second round of GAIA.

Category	Source Dataset	Positive Sample	Negative Sample
$\mathcal{D}$	AndroidControl	68.2k	69.9k
	GUI-Odyssey	65.4k	66.8k
$\mathcal{D}^+$	AndroidControl	(68.2+15.1)k	(69.9+14.0)k
	GUI-Odyssey	(65.4+26.1)k	(66.8+26.3)k

ICM Training and Guidance Based on the dataset $\mathcal{D}$, ICM is trained to intuitively judge the correctness of actions. Specifically, the input of ICM includes the screen observation z_k, the instruction description u_k, the action history h_k, and the given agent action $o_k^{\pi_i}$. We implement ICM using LVLM and use standard cross-entropy loss to supervise the ICM's output tokens. For each sample in our dataset D, the model's output is a token representing either "*correct*" or "*wrong*". The training process aims to minimize the discrepancy between the model's predicted probability and the ground truth label:

$$\begin{aligned}\mathcal{L}_{\text{CE}} = -\frac{1}{K}\sum_{k=1}^{K}\Big[& j_k \log\left(P_{\theta_c}(\text{“}correct\text{”} \mid z_k, u_k, h_k, o_k^{\pi_i})\right) \\ & +(1-j_k)\log\left(1-P_{\theta_c}(\text{“}correct\text{”} \mid z_k, u_k, h_k, o_k^{\pi_i})\right)\Big],\end{aligned} \tag{3}$$

where $P_{\theta_c}(\text{“}correct\text{”} \mid z_k, u_k, h_k, o_k^{\pi_i})$ represents the probability assigned by the critic model θ_c to the "*correct*" token.

During test-time, a GUI agent π_i generates N candidate actions $\mathcal{O} = \{o_1, \ldots, o_N\}$ through N-rollout sampling. ICM evaluates these candidates by assigning each action a correctness judge j_n and a confidence score s_n. Leveraging the best-of-N filtering strategy, we select the optimal action o^* from the subset of correct candidates $\mathcal{O}_{\text{correct}}$ that has the highest confidence score, which is formalized as:

$$o^* = \begin{cases} \arg\max\limits_{o_n \in \mathcal{O}_{\text{correct}}} s_n, & \text{if } \mathcal{O}_{\text{correct}} \neq \varnothing \\ o_1. & \text{otherwise} \end{cases} \tag{4}$$

This approach effectively guides the agent to bypass single-shot output failures and select the most promising action, thereby significantly boosting its overall execution accuracy.

3.3 Data Flywheel and Critic Scaling

Guided by Equation 4, the execution accuracy has been significantly improved. However, some difficult action samples require more precise judgment. Considering that ICM and test-time scaling performance can be further enhanced with data, we collect agent actions guided by ICM and, after filtering for positive and negative balance, add them to the data flywheel to form

$\mathcal{D}^+ = \{j_k | z_k, u_k, h_k, o_k^{\pi_i}, \theta_c\}_{k=1}^{K'}$. $\mathcal{D}^+$ further covers the distribution of actions, providing a foundation for performance scaling.

Based on the challenging samples in this enriched dataset, we train the ICM on Round Two (ICM-r2), using the same cross-entropy loss as defined in Equation 3. This new dataset, which is specifically curated to expose the critic's most significant blind spots, allows ICM-r2 to acquire a more nuanced and accurate discriminative ability. Consequently, as illustrated in Figure 3, ICM-r2 provides more precise guidance for the agent's action selection, thereby fundamentally strengthening the critic's overall judgment and significantly improving the agent's performance on the most difficult tasks. Together with ICM, ICM-r2 demonstrates the power of a data flywheel-driven approach to stimulate the performance of GUI agents during testing.

4 Experiment

4.1 Implementation Details

Experimental Setup. We use UI-TARS 1.0 [31] and UI-TARS 1.5 [31] for inference on the AndroidControl [17] and GUI-Odyssey [23] training sets, and compare the real actions with GT to build $\mathcal{D}$ and $\mathcal{D}^+$. On the corresponding data, we develop the ICM and ICM-r2 based on Qwen2.5 VL 7B [1] and adopt the ms-swift [55] framework for training. All action judgments followed the high-level approach, providing only global instructions to the ICM and ICM-r2, not single-step instructions. The distribution of the data flywheel is shown in Table 1. The critic model guides the agents in the N-rollout process with $N = 8$. To allow the base agent to sample a reasonable range of potential actions, its temperature coefficient, top_k, and top_p are set to 1.0, 30, and 0.8, respectively. All experiments are conducted on 8 NVIDIA H100-80G GPUs.

Action Space. We define a unified action space including *Click*, *Swipe*, *Type*, *Open*, *Home*, *Back*, *Enter*, and *Wait*. Parameterized actions (*Click*, *Swipe*, *Type*, *Open*) are associated with structured arguments such as coordinates, directions, text content, or app identifiers.

Binary critic labels are obtained by comparing normalized agent actions with benchmark ground truth following dataset protocols [17, 23]. Specifically, *Click* actions have a coordinate error within a 14% screen-distance threshold to the ground truth, *Swipe* matches direction, *Type* and *Open* require normalized text or app-name matching, and system actions (*Home*, *Back*, *Enter*, *Wait*) are matched according to states.

To better understand the distribution and characteristics of the collected data, we sample 300 auto-labeled negatives from $\mathcal{D}$ for analysis. *Click* accounts for 64%, followed by *Open* (13%), *Swipe* (9%), and *Type* (4%), with the remaining actions comprising the rest. Detailed dataset documentation is provided in the supplementary material.

Evaluation. To evaluate the performance of the agent after being guided by the critic model, we evaluate the agent's task understanding, grounding, and

Table 2: GUI planning accuracy on AndroidControl. † represents the closed-source UI-TARS 1.5. * represents a reproduced agent. ↑ and ↓ represent changes relative to base agents.

Model	Method	Size	AndroidControl-Low			AndroidControl-High		
			Type	GR	SR	Type	GR	SR
OS-Atlas-Base	ZS	7B	73.0	73.4	50.9	57.4	54.9	29.8
SeeClick	SFT	9.6B	93.0	73.4	75.0	82.9	62.9	59.1
Aria-UI	SFT	3.9B	–	87.7	67.3	–	43.2	10.2
Aguvis	SFT	7B	–	–	80.5	–	–	61.5
UI-R1	RFT	3B	79.2	82.4	66.4	57.9	55.7	45.4
GUI-R1	RFT	3B	83.7	81.6	64.4	58.0	56.2	46.6
GPT4o		–	78.8	8.0	20.4	52.4	3.6	13.2
	+ ICM		82.4 ↑ 3.6	9.2 ↑ 1.2	24.8 ↑ 4.4	58.0 ↑ 5.6	5.3 ↑ 1.7	17.0 ↑ 3.8
	+ ICM-r2		81.6 ↑ 2.8	8.5 ↑ 0.5	23.8 ↑ 3.4	57.6 ↑ 5.2	6.1 ↑ 2.5	18.8 ↑ 5.6
Doubao†		–	97.0	86.4	86.2	82.0	75.0	62.4
	+ ICM		97.0 –	86.6 ↑ 0.2	86.4 ↑ 0.2	83.2 ↑ 1.2	75.1 ↑ 0.1	64.4 ↑ 2.0
	+ ICM-r2		96.6 ↓ 0.4	86.6 ↑ 0.2	86.0 ↓ 0.2	84.2 ↑ 2.2	75.5 ↑ 0.5	66.0 ↑ 3.6
Qwen 2.5 VL		7B	94.4	85.6	81.8	83.0	70.9	60.6
	+ ICM		95.4 ↑ 1.0	86.0 ↑ 0.4	81.2 ↓ 0.6	84.0 ↑ 1.0	75.9 ↑ 5.0	63.4 ↑ 2.8
	+ ICM-r2		94.6 ↑ 0.2	85.4 ↓ 0.2	81.8 –	84.4 ↑ 1.4	75.9 ↑ 5.0	63.8 ↑ 3.2
UI-TARS 1.0*		7B	90.0	85.1	75.4	80.8	68.9	58.2
	+ ICM		90.5 ↑ 0.5	87.6 ↑ 2.5	80.4 ↑ 5.0	82.3 ↑ 1.5	79.5 ↑ 10.6	67.5 ↑ 9.3
	+ ICM-r2		90.0 –	86.8 ↑ 1.7	80.2 ↑ 4.8	82.7 ↑ 1.9	78.9 ↑ 10.0	67.1 ↑ 8.9
UI-TARS 1.5*		7B	86.4	82.4	72.2	80.2	68.1	55.8
	+ ICM		90.3 ↑ 3.9	85.0 ↑ 2.6	79.0 ↑ 6.8	84.2 ↑ 4.0	73.3 ↑ 5.2	64.5 ↑ 8.7
	+ ICM-r2		90.1 ↑ 3.7	85.5 ↑ 3.1	79.2 ↑ 7.0	84.6 ↑ 4.4	74.7 ↑ 6.6	65.6 ↑ 9.8

planning capabilities on the AndroidControl and GUI-Odyssey test sets. For grounding ability, we additionally evaluate performance on ScreenSpotV2 [6]. According to the input, the settings on AndroidControl can be divided into low-level tasks and high-level tasks. High-level tasks only input the global instruction to the agent, while low-level tasks will additionally input the single-step action plan.

Importantly, our critic models are trained exclusively under the high-level setting, without access to single-step action plans. This setting is more challenging yet realistic, as it relies solely on global instructions and interaction history. For consistency, the critic operates under the same high-level setting even when evaluating low-level agents on AndroidControl. GUI-Odyssey follows the high-level setting by default, and both the agent and critic are evaluated accordingly.

Comparison. To verify the effectiveness of the proposed evaluation model in guiding existing agent models during testing, we selected a wide range of models. Among the closed-source models, we selected GPT4o [13] and Doubao (UITARS 1.5) [3], and implemented action acquisition through API calls. For open source models, we reproduced Qwen 2.5 VL [1], UI-TARS 1.0 [31], and UI-TARS 1.5 [31], where Qwen 2.5 VL is a general multimodal understanding model and UI-TARS is a model fine-tuned for GUI agent tasks. We used the official prompt template to reproduce basic performance and test the superposition evaluation model. The original UI-TARS requires historical actions and up to 5 images of historical steps as input. To simplify the operation, we only let the model refer

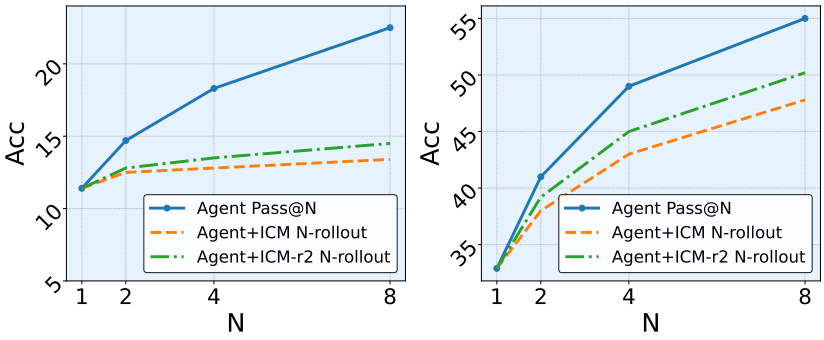
Table 3: GUI planning accuracy on GUI-Odyssey. † represents the closed-source UI-TARS 1.5. * represents a reproduced agent. ↑ and ↓ represent changes relative to base agents.

Model	Method	Size	GUI-Odyssey		
			Type	GR	SR
OS-Atlas-Base	ZS	7B	60.4	39.7	27.0
SeeClick	SFT	9.6B	71.0	52.4	53.9
Aria-UI	SFT	3.9B	–	86.8	36.5
Aguvis	SFT	7B	–	–	–
UI-R1	RFT	3B	52.2	34.5	32.5
GUI-R1	RFT	3B	54.8	41.5	41.3
GPT4o		–	36.6	11.6	11.4
	+ ICM		42.9 ↑ 6.3	10.0 ↓ 1.6	13.4 ↑ 2.0
	+ ICM-r2		43.2 ↑ 6.6	11.4 ↓ 0.2	14.5 ↑ 3.1
Doubao†		–	67.1	67.3	43.8
	+ ICM		70.1 ↑ 3.0	68.4 ↑ 1.1	46.9 ↑ 3.1
	+ ICM-r2		71.6 ↑ 4.5	68.0 ↑ 0.7	47.9 ↑ 4.1
Qwen 2.5 VL		7B	52.7	77.2	40.2
	+ ICM		57.3 ↑ 4.6	77.2 –	43.7 ↑ 3.5
	+ ICM-r2		58.1 ↑ 5.4	78.3 ↑ 1.1	44.8 ↑ 4.6
UI-TARS 1.0*		7B	60.8	71.0	43.3
	+ ICM		72.3 ↑ 11.5	76.1 ↑ 5.1	55.3 ↑ 12.0
	+ ICM-r2		71.4 ↑ 10.6	78.5 ↑ 7.5	56.3 ↑ 13.0
UI-TARS 1.5*		7B	71.1	44.6	32.9
	+ ICM		78.2 ↑ 7.1	52.9 ↑ 8.3	47.8 ↑ 14.9
	+ ICM-r2		80.2 ↑ 9.1	53.5 ↑ 8.9	50.2 ↑ 17.3

Table 4: GUI grounding accuracy on ScreenSpotV2. * indicates reproduced open-source agent performance. ↑ and ↓ respectively represent the performance changes relative to the base agents.

Model	Method	Size	Mobile		Desktop		Web		Avg.
			Text	Icon	Text	Icon	Text	Icon	
GPT-4o	ZS	–	30.5	23.2	20.6	19.4	11.1	7.8	18.8
OS-Atlas-Base	ZS	7B	93.0	72.9	91.8	62.9	90.9	74.3	82.5
SeeClick	SFT	9.6B	78.0	52.0	72.2	30.0	55.7	32.5	53.4
Aguvis	SFT	7B	95.6	77.7	93.8	67.1	88.3	75.2	84.4
Qwen 2.5 VL		7B	84.8	59.7	72.1	52.1	69.2	46.3	65.0
	+ ICM		87.9 ↑ 3.1	70.1 ↑ 10.4	79.4 ↑ 7.3	57.1 ↑ 5.0	74.7 ↑ 5.5	49.2 ↑ 2.9	70.4 ↑ 5.4
	+ ICM-r2		89.7 ↑ 4.9	68.2 ↑ 8.5	78.9 ↑ 6.8	54.3 ↑ 2.2	76.9 ↑ 7.7	51.2 ↑ 4.9	71.1 ↑ 6.1
UI-TARS 1.0*		7B	93.1	82.4	94.8	76.4	91.8	84.2	88.1
	+ ICM		94.5 ↑ 1.3	83.1 ↑ 0.7	93.2 ↓ 1.6	77.9 ↑ 1.5	93.5 ↑ 2.0	84.7 ↑ 0.5	88.7 ↑ 0.6
	+ ICM-r2		94.2 ↑ 1.1	83.7 ↑ 1.3	95.7 ↑ 0.9	78.7 ↑ 2.3	92.1 ↑ 0.3	84.7 ↑ 0.5	89.0 ↑ 0.9
UI-TARS 1.5*		7B	96.2	84.3	94.3	84.2	94.4	86.6	90.8
	+ ICM		96.5 ↑ 0.3	85.3 ↑ 1.0	95.4 ↑ 1.1	84.3 ↑ 0.1	94.2 ↓ 0.2	85.2 ↓ 1.4	90.2 ↓ 0.6
	+ ICM-r2		97.3 ↑ 1.1	84.2 ↓ 0.1	92.4 ↓ 1.9	84.4 ↑ 0.2	95.5 ↑ 1.1	87.7 ↑ 1.1	91.0 ↑ 0.2

to the text description of the historical steps and discard the excessive historical image input. For detailed prompt and inference scripts, please refer to the supplementary material.



(a) GPT4o on GUI-Odyssey **(b)** UI-TARS 1.5* on GUI-Odyssey

Fig. 4: Performance improvements of Pass@N and N-rollout.

As a performance comparison, we selected Zero Shot (ZS) model OS-Atlas-Base [46], SFT-tuned SeeClick [6], Aria-UI [52], and Aguvis [50], and RFT-tuned UI-R1 [24] and GUI-R1 [47].

Evaluation Metrics. For planning tasks, in line with OS-Atlas [46], we report action type prediction accuracy (Type), click point prediction accuracy (GR), and step success rate (SR). Specifically: **Type** measures the exact-match accuracy between predicted and ground-truth action types (e.g., *Click* vs. *Swipe*). **GR** evaluates grounding performance via click point prediction accuracy in specific action types (e.g., *Click*). **SR** is the step-wise success rate: a step is counted as successful only if both the predicted action and its associated arguments (e.g., click coordinates or input text) match the ground truth. For grounding tasks, we use click point prediction accuracy as our evaluation metric.

4.2 Experimental Results

As shown in Table 2 and 3, the proposed ICM and ICM-r2 achieve extensive performance improvements for both zero-shot and fine-tuned GUI agents. On the AndroidControl-High test, ICM can improve agents' SR performance by up to 9.3%, while ICM-r2 can further improve it by an average of 1.32%. The same trend is also observed on GUI-Odyssey, demonstrating that existing models have the potential to correctly infer actions. ICM can leverage this potential to effectively improve existing agents during testing, and data flywheels can further amplify these improvements. For agents with lower basic performance, ICM can significantly improve their performance to an advanced level, which also demonstrates the potential of the model itself and the stimulation ability of the critic model. In terms of generalization, GPT4o, Doubao, and Qwen 2.5 VL were not included in the construction of the GAIA $\mathcal{D}$ and $\mathcal{D}^+$, but ICM and ICM-r2 still achieved significant performance improvements, demonstrating the inherent consistency of real action space data. The real action sampling in the proposed data flywheel effectively covers this space, providing effective support for critic training.

Table 5: The relationship between computation cost and the number of N. Time is measured in seconds (s).

Metric	N			
	1	2	4	8
Actor	1.0143	1.0874	1.1316	1.2617
Critic	0.4739	0.6746	1.1018	1.9897
Total	1.4882	1.7620	2.2334	3.2514

Table 6: Critic comparison. The performance is calculated as (Qwen2.5 VL 7B w/ critic) - (Qwen2.5 VL 7B w/o critic).

Model	N	AndroidControl-High		
		Δ Type	Δ GR	Δ SR
UI-Genie-RM	10	–	–	0.3
ICM	8	1.0	5.0	2.8
ICM-r2	8	1.4	5.0	3.2

Both our ICM and ICM-r2 use a high-level approach to judge correctness and guide action, meaning the critic model is not aware of the current action plan. This setting is more consistent with practical applications, where only global instructions are given, and the agent must independently reason about each step's plan and action. For AndroidControl-Low, the agent is aware of the current action plan, resulting in higher baseline performance. Despite this, our ICM still achieves a certain degree of performance improvement, demonstrating the effectiveness of our proposed approach.

Table 4 shows the improvement of ICM and ICM-r2 on the grounding ability of agents on ScreenSpotv2. The ScreenSpotv2 data is not included in the proposed GAIA, and as a single-step operation, its environmental information is not completely consistent with the aforementioned datasets. Even so, our evaluation model still improves the performance of agents, which is sufficient to prove the validity of the data flywheel definition.

4.3 Ablation Study

While utilizing ICM and ICM-r2, we used N-rollout to improve the test-time performance of existing GUI agents, where N is set to 8 by default. To measure the impact of N on final performance, we selected GPT4o and UI-TARS 1.5* as representative closed-source and open-source models, respectively, and compared their SR at different N values on GUI-Odyssey. We also measured the models' Pass@N during the rollout process to reflect the model's performance ceiling. As shown in Figure 4, the increase in Pass@N accuracy reveals the potential of the agents themselves, while the evaluation model approaches this upper limit through N-rollout. The improvement in ICM-r2 and the gap between the upper limit provide potential performance gains for further cycles of GAIA.

4.4 Computation Efficiency

We conduct an efficiency analysis to verify that the critic-guided TTS does not introduce prohibitive computational overhead. For open-source models, we leverage vLLM for actor inference and multi-GPU parallelism for critic scoring. As shown in Table 5, the total latency at N=8 is only approximately 2.2× that

Table 7: Impact of differences in critic model attributes on accuracy and guidance. The tests are conducted on the GUI-Odyssey dataset using UI-TARS 1.5* as the base model.

Model	Critic Acc	GUI-Odyssey		
		Type	GR	SR
UI-TARS 1.5*	–	71.1	44.6	32.9
+ RCM	70.82%	75.6	49.2	44.1
+ ICM	83.19%	78.2	52.9	47.8
+ ICM-r2	83.56%	80.2	53.5	50.2

of N=1, indicating that the time complexity scales sub-linearly with N due to effective parallel execution. Notably, the dominant cost comes from base-agent rollouts rather than critic inference, as the critic operates as a lightweight binary evaluator with batched evaluation over N candidates.

For closed-source models, since API calls are inherently parallelizable, varying N from 1 to 8 introduces negligible additional latency, which is also adopted by existing GUI agent frameworks such as GTA1 [51]. In practice, a single rollout typically involves 1,000 – 2,000 input tokens and about 30 output tokens. The corresponding API cost can be estimated based on the pricing of the deployed model. These results confirm that the proposed method can be deployed with acceptable overhead in practical settings.

4.5 Qualitative Experiment

Critic Model Comparison. To evaluate the effectiveness of the proposed discriminant model, we compared the accuracy of the Qwen 2.5 VL using a best-of-N approach to guide inference on AndroidControl-High with UI-Genie-RM [48]. As shown in Table 6, due to the use of real action data, the proposed ICM significantly improves the accuracy of the base model, and ICM-r2 can further expand the advantage.

Intuitive and Reasoning Critic. To verify that the intuitive judgment proposed in this article is superior, this section implements a critic model based on reinforcement learning design. Specifically, the input of the Reasoning Critic Model (RCM) is consistent with ICM, and the output includes `<thinking>...</thinking>` and `<critic>...</critic>`, which are supervised by format reward and critic reward. The training of RCM is achieved through Group Relative Policy Optimization (GRPO). Considering the property of reinforcement learning, which is that it can stimulate model capabilities with less data, we randomly sampled 30k data from $\mathcal{D}^+$ to train RCM. This data includes samples from two rounds of GAIA and has the same distribution as the training data for ICM-r2. To intuitively compare the discriminative performance of different critic models, we collected the GAIA test set in a high-level manner on the AndroidControl and GUI-Odyssey test sets in the same way as we collected the training data.

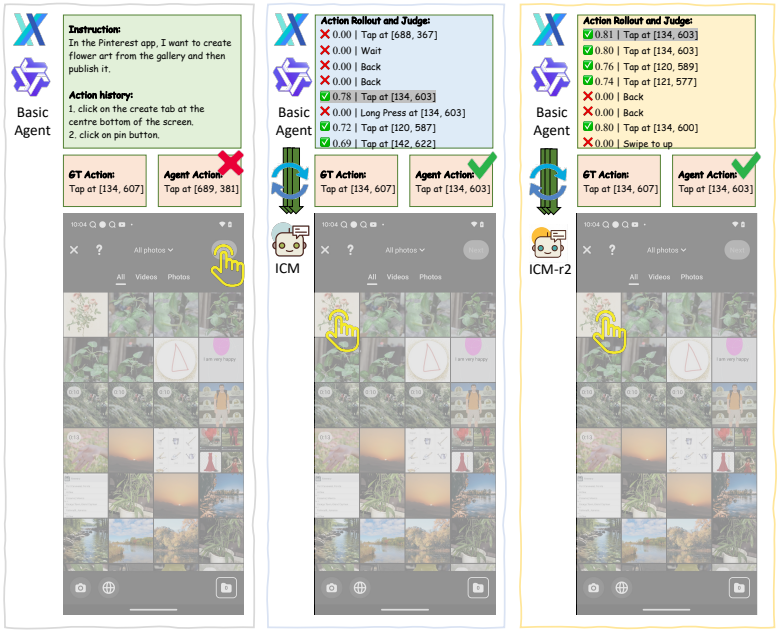


Fig. 5: Visualization result. The basic agent selects the wrong action. Based on the action rollout, both ICM and ICM-r2 select the correct action from the candidates.

Table 7 shows that the proposed ICM achieves an accuracy of 83.19% for correctness assessment, providing a foundation for action guidance. ICM-r2, benefiting from improved data quality, further achieves an accuracy of 83.56%. In contrast, RCM's classification accuracy is 70.82%, indicating that the thinking component fails to significantly contribute to the final assessment. In terms of action guidance accuracy, while UI-TARS 1.5* under RCM guidance outperforms the original model, it still falls short of ICM. This experimental result demonstrates that intuitive judgment outperforms reasoning for improving agents using a critic model.

Please refer to the supplementary material for more experimental results.

4.6 Case Study

To provide an intuitive understanding of the critic-guided TTS, we present qualitative examples in Figure 5 and Figure 6. In Figure 5, the basic agent selects an incorrect action, while both ICM and ICM-r2 successfully identify the correct one from the rollout candidates. Figure 6 illustrates a more challenging case where ICM fails to correct the erroneous action, yet ICM-r2 still guides the agent toward the correct selection. These cases demonstrate not only the effectiveness of the proposed critic mechanism but also the improved robustness of ICM-r2 in handling difficult scenarios.

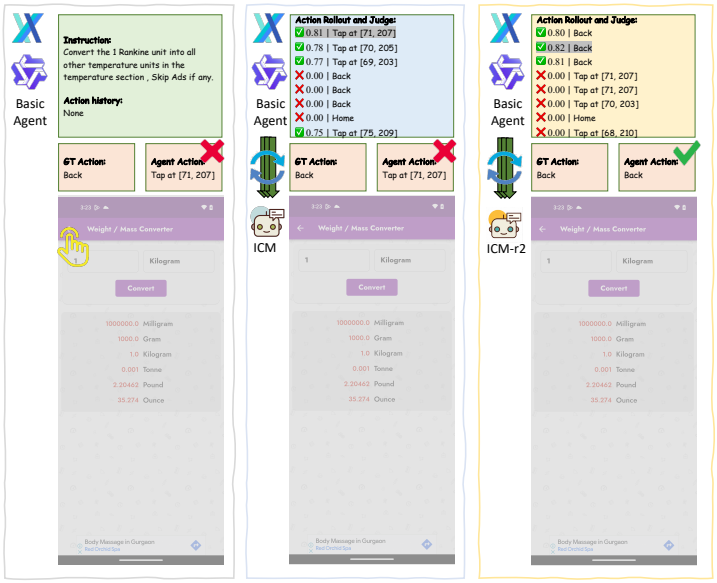


Fig. 6: Visualization result. ICM fails to select the correct one from the rollout candidates, while the enhanced ICM-r2 guides the correct selection.

However, the failure of ICM in Figure 6 does not stem from an inherently incorrect action, but from the inability to distinguish between semantically equivalent yet differently defined actions under the benchmark (clicking a Back icon vs. triggering the system Back action). This corresponds to a multi-path scenario, where multiple valid actions achieve the same goal, suggesting a direction for improving the data flywheel. More visualizations are available in the supplementary material.

5 Conclusion

In this work, we addressed the critical challenge of irreversible errors in GUI agents by proposing GAIA, a framework that unleashes their latent potential at test time. GAIA comprises a data flywheel that iteratively curates realistic action samples and the Intuitive Critic Model (ICM) that evaluates action correctness, establishing a self-evolutionary cycle where enriched data trains an increasingly powerful critic (ICM-r2). By leveraging a Best-of-N strategy, ICM enables agents to select more reliable actions without resource-intensive retraining. Experiments on both closed-source and open-source agents demonstrate significant performance gains, presenting a scalable solution for building more robust GUI agents. In future work, we plan to unify high-level and low-level guidance methods and collect richer data through online testing to continuously iterate the data flywheel.

References

1. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al.: Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923 (2025)
2. Bilalić, M., McLeod, P., Gobet, F.: Why good thoughts block better ones: The mechanism of the pernicious Einstellung (set) effect. Cognition **108**(3), 652–661 (2008)
3. ByteDance: Doubao. https://www.volcengine.com/product/doubao (2025), accessed: 2025-8-1
4. Chen, W., Yuan, J., Qian, C., Yang, C., Liu, Z., Sun, M.: Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system. In: Findings of the Association for Computational Linguistics: ACL 2025. pp. 11534–11557 (2025)
5. Chen, Z., Wang, W., Cao, Y., Liu, Y., Gao, Z., Cui, E., Zhu, J., Ye, S., Tian, H., Liu, Z., et al.: Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. arXiv preprint arXiv:2412.05271 (2024)
6. Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., Wu, Z.: SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. pp. 9313–9332 (2024)
7. Christianos, F., Papoudakis, G., Coste, T., Hao, J., Wang, J., Shao, K.: Lightweight neural app control. In: International Conference on Learning Representations. pp. 74161–74178 (2025)
8. Doyon, J., Benali, H.: Reorganization and plasticity in the adult brain during learning of motor skills. Current opinion in neurobiology **15**(2), 161–167 (2005)
9. Gou, B., Wang, D.R., Zheng, B., Xie, Y., Chang, C., Shu, Y., Sun, H., Su, Y.: Navigating the digital world as humans do: Universal visual grounding for GUI agents. In: International Conference on Learning Representations. pp. 30851–30883 (2025)
10. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)
11. Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., et al.: CogAgent: A visual language model for GUI agents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14281–14290 (2024)
12. Hu, X., Xiong, T., Yi, B., Wei, Z., Xiao, R., Chen, Y., Ye, J., Tao, M., Zhou, X., Zhao, Z., et al.: OS Agents: A survey on MLLM-based agents for general computing devices use. arXiv preprint arXiv:2508.04482 (2025)
13. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: GPT-4o system card. arXiv preprint arXiv:2410.21276 (2024)
14. Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al.: OpenAI o1 system card. arXiv preprint arXiv:2412.16720 (2024)
15. Ji, Z., Yu, T., Xu, Y., Lee, N., Ishii, E., Fung, P.: Towards mitigating LLM hallucination via self reflection. In: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 1827–1843 (2023)
16. Kalyanpur, A., Saravanakumar, K.K., Barres, V., Chu-Carroll, J., Melville, D., Ferrucci, D.: LLM-ARC: Enhancing LLMs with an automated reasoning critic. arXiv preprint arXiv:2406.17663 (2024)

17. Li, W., Bishop, W., Li, A., Rawles, C., Campbell-Ajala, F., Tyamagundlu, D., Riva, O.: On the effects of data scale on computer control agents. arXiv preprint arXiv:2406.03679v2 (2024)
18. Liu, T., Pleskac, T.J.: Neural correlates of evidence accumulation in a perceptual decision task. Journal of Neurophysiology **106**(5), 2383–2398 (2011)
19. Liu, X., Qin, B., Liang, D., Dong, G., Lai, H., Zhang, H., Zhao, H., Iong, I.L., Sun, J., Wang, J., et al.: AutoGLM: Autonomous foundation agents for GUIs. arXiv preprint arXiv:2411.00820 (2024)
20. Liu, Y., Li, P., Wei, Z., Xie, C., Hu, X., Xu, X., Zhang, S., Han, X., Yang, H., Wu, F.: InfiGUIAgent: A multimodal generalist GUI agent with native reasoning and reflection. arXiv preprint arXiv:2501.04575 (2025)
21. Liu, Y., Li, P., Xie, C., Hu, X., Han, X., Zhang, S., Yang, H., Wu, F.: InfiGUI-R1: Advancing multimodal GUI agents from reactive actors to deliberative reasoners. arXiv preprint arXiv:2504.14239 (2025)
22. Liu, Z., Sun, Z., Zang, Y., Dong, X., Cao, Y., Duan, H., Lin, D., Wang, J.: Visual-RFT: Visual reinforcement fine-tuning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2034–2044 (2025)
23. Lu, Q., Shao, W., Liu, Z., Du, L., Meng, F., Li, B., Chen, B., Huang, S., Zhang, K., Luo, P.: GUIOdyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 22404–22414 (2025)
24. Lu, Z., Chai, Y., Guo, Y., Yin, X., Liu, L., Wang, H., Xiong, G., Li, H.: UI-R1: Enhancing action prediction of GUI agents by reinforcement learning. arXiv preprint arXiv:2503.21620 (2025)
25. Martino, A., Iannelli, M., Truong, C.: Knowledge injection to counter large language model (LLM) hallucination. In: European Semantic Web Conference. pp. 182–185. Springer (2023)
26. McAleese, N., Pokorny, R.M., Uribe, J.F.C., Nitishinskaya, E., Trebacz, M., Leike, J.: LLM critics help catch LLM bugs. arXiv preprint arXiv:2407.00215 (2024)
27. Nguyen, D., Chen, J., Wang, Y., Wu, G., Park, N., Hu, Z., Lyu, H., Wu, J., Aponte, R., Xia, Y., et al.: GUI agents: A survey. In: Findings of the Association for Computational Linguistics: ACL 2025. pp. 22522–22538 (2025)
28. Nong, S., Zhu, J., Wu, R., Jin, J., Shan, S., Huang, X., Xu, W.: MobileFlow: A multimodal LLM for mobile GUI agent. arXiv preprint arXiv:2407.04346 (2024)
29. Poldrack, R.A., Sabb, F.W., Foerde, K., Tom, S.M., Asarnow, R.F., Bookheimer, S.Y., Knowlton, B.J.: The neural correlates of motor skill automaticity. Journal of Neuroscience **25**(22), 5356–5364 (2005)
30. Prabhudesai, M., Ke, T.W., Li, A., Pathak, D., Fragkiadaki, K.: Diffusion-TTA: Test-time adaptation of discriminative models via generative feedback. In: Advances in Neural Information Processing Systems. pp. 17567–17583 (2023)
31. Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., et al.: UI-TARS: Pioneering automated GUI interaction with native agents. arXiv preprint arXiv:2501.12326 (2025)
32. Shen, H., Liu, P., Li, J., Fang, C., Ma, Y., Liao, J., Shen, Q., Zhang, Z., Zhao, K., Zhang, Q., et al.: VLM-R1: A stable and generalizable R1-style large vision-language model. arXiv preprint arXiv:2504.07615 (2025)
33. Shen, H., Liu, C., Li, G., Wang, X., Zhou, Y., Ma, C., Ji, X.: Falcon-UI: Understanding GUI before following user instructions. arXiv preprint arXiv:2412.09362 (2024)

34. Snell, C., Lee, J., Xu, K., Kumar, A.: Scaling LLM test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:2408.03314 (2024)
35. Snell, C.V., Lee, J., Xu, K., Kumar, A.: Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In: International Conference on Learning Representations (2025)
36. Song, Y., Bian, Y., Tang, Y., Ma, G., Cai, Z.: VisionTasker: Mobile task automation using vision based UI understanding and LLM task planning. In: Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology. pp. 1–17 (2024)
37. Tang, F., Shen, Y., Zhang, H., Chen, S., Hou, G., Zhang, W., Zhang, W., Song, K., Lu, W., Zhuang, Y.: Think twice, click once: Enhancing GUI grounding via fast and slow systems. arXiv preprint arXiv:2503.06470 (2025)
38. Tian, X., Zhao, S., Wang, H., Chen, S., Ji, Y., Peng, Y., Zhao, H., Li, X.: Think twice: Enhancing LLM reasoning by scaling multi-round test-time thinking. arXiv preprint arXiv:2503.19855 (2025)
39. Wan, X., Nakatani, H., Ueno, K., Asamizuya, T., Cheng, K., Tanaka, K.: The neural basis of intuitive best next-move generation in board game experts. Science **331**(6015), 341–346 (2011)
40. Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191 (2024)
41. Wang, S., Liu, W., Chen, J., Zhou, Y., Gan, W., Zeng, X., Che, Y., Yu, S., Hao, X., Shao, K., et al.: GUI agents with foundation models: A comprehensive survey. arXiv preprint arXiv:2411.04890 (2024)
42. Wang, Y., Ji, P., Yang, C., Li, K., Hu, M., Li, J., Sartoretti, G.: MCTS-judge: Test-time scaling in LLM-as-a-judge for code correctness evaluation. arXiv preprint arXiv:2502.12468 (2025)
43. Wanyan, Y., Zhang, X., Xu, H., Liu, H., Wang, J., Ye, J., Kou, Y., Yan, M., Huang, F., Yang, X., et al.: Look before you leap: A GUI-Critic-R1 model for pre-operative error diagnosis in GUI automation. arXiv preprint arXiv:2506.04614 (2025)
44. Wen, M., Wan, Z., Wang, J., Zhang, W., Wen, Y.: Reinforcing LLM agents via policy optimization with action decomposition. In: Advances in Neural Information Processing Systems. pp. 103774–103805 (2024)
45. Wu, Q., Cheng, K., Yang, R., Zhang, C., Yang, J., Jiang, H., Mu, J., Peng, B., Qiao, B., Tan, R., et al.: GUI-Actor: Coordinate-free visual grounding for GUI agents. arXiv preprint arXiv:2506.03143 (2025)
46. Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P.P., et al.: OS-ATLAS: A foundation action model for generalist GUI agents. In: International Conference on Learning Representations. pp. 5090–5108 (2025)
47. Xia, X., Luo, R.: GUI-R1: A generalist R1-style vision-language action model for GUI agents. arXiv preprint arXiv:2504.10458 (2025)
48. Xiao, H., Wang, G., Chai, Y., Lu, Z., Lin, W., He, H., Fan, L., Bian, L., Hu, R., Liu, L., et al.: UI-Genie: A self-improving approach for iteratively boosting MLLM-based mobile GUI agents. arXiv preprint arXiv:2505.21496 (2025)
49. Xiong, T., Wang, X., Guo, D., Ye, Q., Fan, H., Gu, Q., Huang, H., Li, C.: LLaVA-Critic: Learning to evaluate multimodal models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 13618–13628 (2025)

50. Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., Xiong, C.: AGUVIS: Unified pure vision agents for autonomous GUI interaction. arXiv preprint arXiv:2412.04454 (2024)
51. Yang, Y., Li, D., Dai, Y., Yang, Y., Luo, Z., Zhao, Z., Hu, Z., Huang, J., Saha, A., Chen, Z., et al.: GTA1: GUI test-time scaling agent. arXiv preprint arXiv:2507.05791 (2025)
52. Yang, Y., Wang, Y., Li, D., Luo, Z., Chen, B., Huang, C., Li, J.: Aria-UI: Visual grounding for GUI instructions. In: Findings of the Association for Computational Linguistics: ACL 2025. pp. 22418–22433 (2025)
53. Zhang, D., Lei, J., Li, J., Wang, X., Liu, Y., Yang, Z., Li, J., Wang, W., Yang, S., Wu, J., et al.: Critic-V: VLM critics help catch VLM errors in multimodal reasoning. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 9050–9061 (2025)
54. Zhang, Z., Wang, C., Wang, Y., Shi, E., Ma, Y., Zhong, W., Chen, J., Mao, M., Zheng, Z.: LLM hallucinations in practical code generation: Phenomena, mechanism, and mitigation. Proceedings of the ACM on Software Engineering **2**, 481–503 (2025)
55. Zhao, Y., Huang, J., Hu, J., Wang, X., Mao, Y., Zhang, D., Jiang, Z., Wu, Z., Ai, B., Wang, A., et al.: SWIFT: A scalable lightweight infrastructure for fine-tuning. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 29733–29735 (2025)
56. Zheng, J., Wang, L., Yang, F., Zhang, C., Mei, L., Yin, W., Lin, Q., Zhang, D., Rajmohan, S., Zhang, Q.: VEM: Environment-free exploration for training GUI agent with value environment model. arXiv preprint arXiv:2502.18906 (2025)