Look Before You Leap: A GUI-Critic-R1 Model for Pre-Operative Error Diagnosis in GUI Automation

Yuyang Wanyan[1,2,3†*], **Xi Zhang**[3*], **Haiyang Xu**[3‡], **Haowei Liu**[3], **Junyang Wang**[3], **Jiabo Ye**[3], **Yutong Kou**[1], **Ming Yan**[3‡], **Fei Huang**[3], **Xiaoshan Yang**[1,2‡], **Weiming Dong**[1,2], **Changsheng Xu**[1,2]
[1]MAIS, Institute of Automation, Chinese Academy of Sciences, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, China
[3]Alibaba Group
wanyanyuyang2021@ia.ac,cn, xiaoshan.yang@nlpr.ia.ac.cn,
{shuofeng.xhy, ym119608}@alibaba-inc.com

Abstract

In recent years, Multimodal Large Language Models (MLLMs) have been extensively utilized for multimodal reasoning tasks, including Graphical User Interface (GUI) automation. Unlike general offline multimodal tasks, GUI automation is executed in online interactive environments, necessitating step-by-step decision-making based on the real-time status of the environment. This task has a lower tolerance for decision-making errors at each step, as any mistakes may cumulatively disrupt the process and potentially lead to irreversible outcomes like *deletions* or *payments*. To address these issues, we introduce a **pre-operative critic** mechanism that provides effective feedback prior to the actual execution, by reasoning about the potential outcome and correctness of actions. Specifically, we propose a Suggestion-aware Group Relative Policy Optimization (**S-GRPO**) strategy to construct our pre-operative critic model GUI-Critic-R1, incorporating a novel suggestion reward to enhance the reliability of the model's feedback. Furthermore, we develop a reasoning-bootstrapping based data collection pipeline to create a **GUI-Critic-Train** and a **GUI-Critic-Test**, filling existing gaps in GUI critic data. Static experiments on the GUI-Critic-Test across both mobile and web domains reveal that our GUI-Critic-R1 offers significant advantages in critic accuracy compared to current MLLMs. Dynamic evaluation on GUI automation benchmark further highlights the effectiveness and superiority of our model, as evidenced by improved success rates and operational efficiency. The code is available at https://github.com/X-PLUG/MobileAgent/tree/main/GUI-Critic-R1.

1 Introduction

Recently, Multi-modal Large Language Models (MLLMs), leveraging their remarkable perception and reasoning capabilities, have demonstrated outstanding capabilities in various domains [2, 41]. Among these, GUI automation, as a practical multi-modal application scenario, is emerging as a significant technological revolution in artificial intelligence interactions [38, 16, 50, 15, 44, 26, 27]. To be specific, given an online GUI device and a natural language instruction, it requires the GUI agent driven by MLLMs to generate a series of precise operations similar to how humans do [48, 32], such as click, type, and scroll.

Unlike traditional offline multimodal tasks such as visual question answering [19] and optical character recognition [14], GUI automation task operates within an online interactive environment

*Equal contribution. †Work done during internship at Tongyi Lab, Alibaba Group. ‡Corresponding Author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-0137

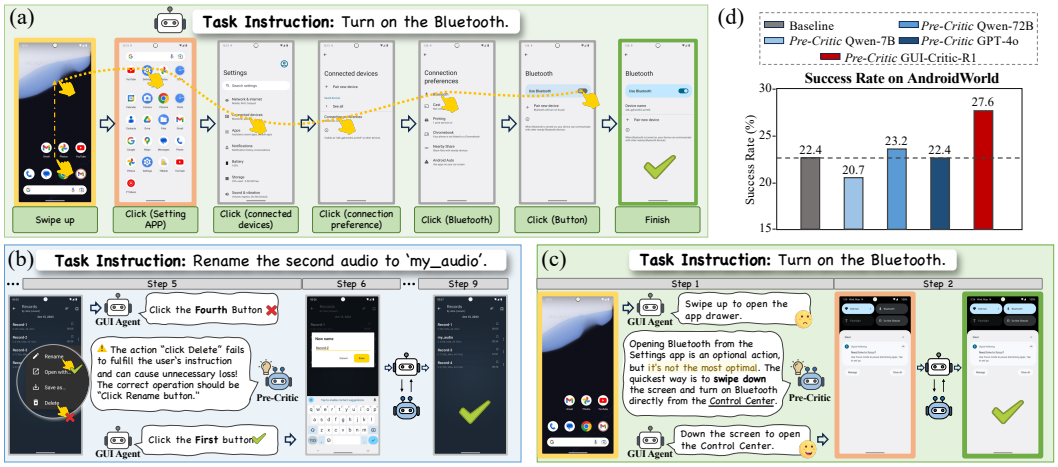


Figure 1: (a) shows an example of GUI automation. Case studies in (b-c) demonstrate how pre-critic prevents erroneous and redundant actions in GUI automation. (d) illustrates the quantitative performance comparison between pre-critic methods and baseline on AndroidWorld [27].

and has some inherent key challenges [32]. To be specific, the agents are required to generate coherent and sequential operations step by step. In this way, an error in one step can have cumulative effects on subsequent operations (*e.g., delete a file shown in Figure 1(b)*), thereby disrupting the entire interactive process. However, constrained by limited reflection capabilities, current MLLMs usually struggle to detect errors independently [34]. Therefore, to ensure the single-step accuracy, it is necessary to provide MLLM-based agents with additional feedbacks that incorporates critic analysis, such as assessments of the action correctness, potential outcomes, and action suggestions. Nevertheless, in the dynamic environment, erroneous operations usually require additional steps to correct (*e.g., refunding after an incorrect payment*), and some dangerous errors may be irreparable (*e.g., deleting files as shown in Figure 1(b)*). Therefore, to prevent these issues from happening, we believe that the critical feedback should be provided to the agent **before the action is actually executed**. Additionally, completing a user instruction on a GUI device typically entails multiple pathways. From an application perspective, the GUI agent is expected to complete instructions with an optimal path that contains the least number of steps. In this way, the prior critical feedback may also prevent the model from selecting a sub-optimal path with more steps as shown in Figure 1(a), thereby enhancing the efficiency of completing instructions.

Considering the above issues, in this paper, we propose a **pre-operative critic** (pre-critic) mechanism for GUI automation. Specifically, before performing an operation in online environments, the pre-critic mechanism first evaluates whether the operation generated by the agent is beneficial to the instruction completion, through comprehending the screenshot and analyzing potential results of the operation. Then, it provides real-time critical feedback to the agent, including the underlying causes of errors and corrective suggestions, to assist in refining its decision-making process. For example, as shown in Figure 1(b), given the instruction *Rename the current audio to 'my_audio'*, the agent initially predicts the action of *click (delete button)* in the 5-th step. Such operation may cause the audio file to be deleted and the task to fail. Fortunately, with the pre-critic mechanism, we can catch the dangerous error before executing, and provide objective feedback to the agent. Besides, Figure 1(c) illustrates a case where the pre-critic identifies a more efficient method to *turn on Bluetooth* (*i.e., enabling Bluetooth via the control center*), thereby reducing the operation steps. We also exhibit some statistic results in Figure 1(d), where the pre-critic helps increase the success rate of baseline from 22.4% to 27.6% in the dynamic GUI automation [27] *. In conclusion, these figures demonstrate that the introduction of pre-critic can effectively alleviate the aforementioned issues of error revision and operational inefficiency in online environments.

An intuitive approach to implementing a pre-critic model is leveraging existing MLLMs. However, closed-source models [13] incur heavy efficiency and cost issues, making them unsuitable for real-time GUI automation. Besides, open-source models [5, 2, 41] struggle as pre-critic models due to inherent

*More detailed experimental results about efficiency can be found in Table 2

https://doi.org/10.52202/085713-0137

limitations in comprehending GUI interface and forecasting interaction outcomes [16, 36, 51]. To this end, we propose a specialized 7B model **GUI-Critic-R1** that harmonizes performance and efficiency for GUI pre-critic. In particular, to enhance the model's GUI reasoning and generalization capabilities, we introduce a Suggestion-aware Group Relative Policy Optimization (**S-GRPO**). In S-GRPO, a suggestion reward is innovatively designed to refine the model's critic reasoning process and ensure it provides reliable suggestions for fixing the error operation. Moreover, since it lacks both training and test datasets for pre-critic in GUI automation, we develop a reasoning-bootstrapping based data collection pipeline to construct a **GUI-Critic-Train** and a **GUI-Critic-Test** dataset. Specifically, GUI-Critic-Train contains 6K high-quality chain-of-thought data about mobile devices for robust training. GUI-Critic-Test includes 1k samples encompassing both mobile and web scenarios, and aims to explicitly evaluate the pre-critic model's diagnostic capabilities when exposed to novel instructions or applications. In experiments, we first compare our GUI-Critic-R1 with advanced MLLMs on the GUI-Critic-Test, and find that our model achieves satisfactory results in the critic accuracy. Then, we also apply the GUI-Critic-R1 to a real-time GUI automation benchmark (*i.e.*, AndroidWorld [27]), the improvements on the success rate further demonstrate the effectiveness of the proposed method.

Our main contributions are summarized as follows:

1. To the best of our knowledge, we are the first to investigate a pre-operative critic mechanism for diagnosing GUI operations. To achieve it, we propose a Suggestion-aware Group Relative Policy Optimization strategy integrating a novel suggestion reward, and develop our pre-critic model GUI-Critic-R1. This model is capable of delivering constructive and insightful feedback to refine the GUI reasoning process.
2. We present a reasoning-bootstrapping based data collection pipeline to construct a GUI-Critic-Train dataset, comprising 6k high-quality chain-of-thought annotations. Additionally, a GUI-Critic-Test dataset is developed to comprehensively evaluate the critic model's performance in both mobile and web domains.
3. Extensive experiments on both GUI-Critic-Test dataset and dynamic GUI automation benchmark validate the efficacy of our GUI-Critic-R1 model in producing reliable judgments and feedback for GUI operations.

2 Related Work

LLM-based GUI Agent. Recently, significant attention has been directed toward Graphical User Interface (GUI) agents for task automation on smart devices designed for the mobile and web environments [50, 15, 36, 49, 56, 45, 10, 52, 20, 30, 39, 40]. Despite these advancements, GUI agents frequently make erroneous decisions. Some research has incorporated reflection modules in GUI automation frameworks to verify operation correctness based on the both current screenshot and the screenshot after execution [**?**, 16, 1]. For example, Mobile-Agent-v2 frameworks [36] employ multi-agent architectures that separate planning, decision-making, and reflection to optimize task tracking, memory, and error correction. But these methods require additional steps to undo actions and pose the risk of irreversible operations, resulting in lower efficiency and accuracy. In this paper, we introduces a pre-critic mechanism for diagnosing potential errors.

Critic Model for LLMs. Large language models (LLMs) do not always generate the best reasoning output when performing challenging inference tasks [34]. To address this limitation, several studies propose self-refinement[21, 31, 24] and self-reflect techniques [29, 9, 35, 47, 55, 6, 7] to reifine the output themselves. However, their effectiveness is largely restricted by their reliance on the inherent capabilities of LLMs, which may hinder the broader application and scalability of these methods [11]. In contrast, several studies [22, 43, 42] explore employing an independent critic model to produce natural language feedback for the evaluation of outputs generated by Large Language Models (LLMs). Critic-V [51] extends this inspiration to the area of VLMs to train a critic vision-language model to locate imperfections in visual content perception and errors in reasoning steps. However, they focus on general offline tasks. Differently, we explore the application of a critic model in more complex scenarios of GUI automation, which presents increased challenges due to its operation in an online environment. In this paper, we perform pre-operative critic to resolve the challenges.

Reinforcement Learning for Reasoning. Recent research has increasingly focused on enhancing the reasoning capabilities of LLMs through Reinforcement Learning (RL), drawing inspiration from

https://doi.org/10.52202/085713-0137

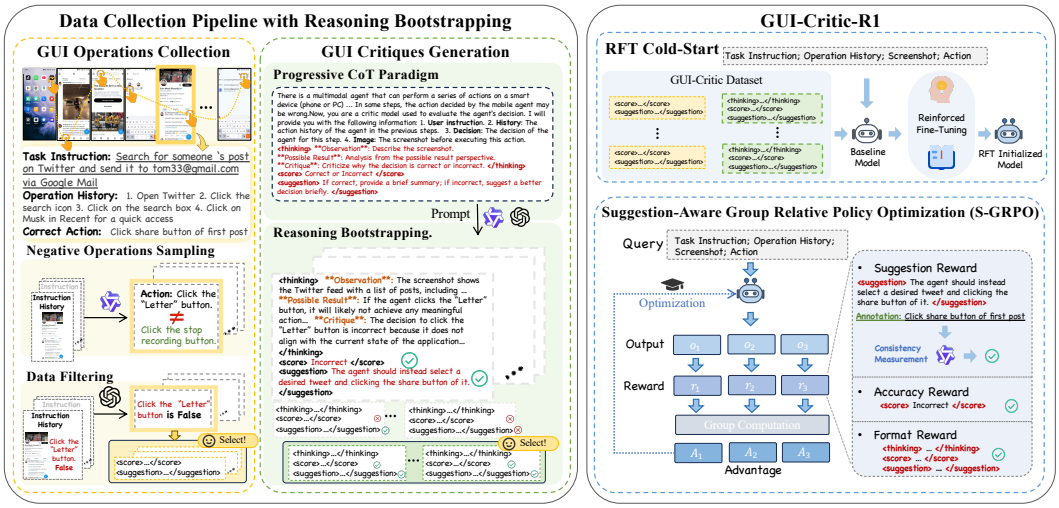


Figure 2: The left shows our reasoning-bootstrapping based data collection pipeline, including the GUI operations collection and GUI critiques generation. Specifically, a progressive CoT paradigm and a reasoning-bootstrapping strategy are employed to ensure the quality of critiques. The right illustrates the training strategy for our GUI-Critic-R1 model. The process begins with a RFT cold-start on the GUI-Critic-Train dataset, and followed by the implementation of our proposed S-GRPO. Besides, a novel suggestion reward is employed to constrain the correctness of suggestions.

models such as DeepSeek-R1 [8] and Kimi-1.5 [33]. These models employ rule-based reward mechanisms to enhance reasoning performance. Several studies attempt to adapt the idea of reinforcement learning with verifiable rewards in multimodal scenarios. For instance, R1-V [53] investigates the application of rule-based RL in geometry problems and object-counting tasks, while Visual-RFT [17] extends this approach to open vocabulary and few-shot detection, reasoning grounding, and fine-grained few-shot classification. Recent studies [12, 25, 23, 53, 46] further generalize the algorithm to address more general tasks such as multimodal mathematical reasoning, decision-making, and planning. In this paper, we extend the reinforcement learning algorithm to train a critic model that diagnoses operations in GUI automation, and propose a S-GRPO strategy with a novel suggestion reward to enhance the model's reasoning capabilities.

3 Method

3.1 Problem Definition of Pre-Operative Critic

The GUI automation task can be formally characterized as a Markov Decision Process: $\mathcal{M} = (\mathcal{E}, \mathcal{A}, \mathcal{P}, \pi_{agent})$. The state of the environment $\epsilon \in \mathcal{E}$ consists of a user instruction, historical interactions, and the current screenshot of the device. The action space $\mathcal{A}$ encompasses all available operations (actions) including *click, long press, type, scroll, home, back*, and *done*. At each step, the MLLM-based agent π_{agent} observes the environment ϵ and selects an action a. Upon executing a, it receives a novel observation ϵ', with the probability given by the state transition function $\mathcal{P}(\epsilon'|\epsilon, a)$. The agent iterates this process until accomplishing the desired instruction or encountering a terminal state.

In this paper, we propose a pre-operative critic model $\pi_{critic}(\epsilon, a)$ to critique the decision-making process of π_{agent}, before the action is actually executed. Specifically, taking state ϵ and action a as inputs, the model produces a correctness score $l \in [0, 1]$ that reflects the rightness of a. Besides, it generates critique c and corrective suggestion s in natural language, with the former explaining the rationale behind the correctness judgment and the latter suggesting a better action if a is incorrect.

3.2 Data Collection Pipeline with Reasoning Bootstrapping

In this section, we detail the data collection pipeline of our proposed GUI-Critic-Train and GUI-Critic-Test datasets. Each sample in the dataset includes the environmental state ϵ of the step, a operation candidate a, as well as annotations for correctness score l and suggestion s. This section

https://doi.org/10.52202/085713-0137

commences by introducing the collection of operations in Section 3.2.1, followed by the presentation of our reasoning bootstrapping method for generating critique c in Section 3.2.2.

3.2.1 GUI Operations Collection

To begin with, we collect successful automation trajectories from publicly accessible datasets, encompassing correct step-level operations across various GUI scenarios. The action in a specific state ϵ is considered as the operation candidate whose correctness score is positive ($l = 1$), and the corresponding action description is considered as the suggestion s.

Negative Operations Sampling. Subsequently, we collect samples with negative score ($l = 0$) based on the states of the above correct operations. To be specific, given these states, open-source MLLM-based agents are first employed to predict a set of operations. Then, these operations are evaluated according to rule-based criteria, and we retain those deemed incorrect. In this way, the obtained negative operations are aligned as closely as possible with the real error distribution in GUI environments, thus ensuring the quality of the dataset. The corrective suggestions for these samples are the same as the corresponding positive samples.

Data Filtering. The data collected using the aforementioned methods may not be entirely reliable. This is because that public datasets could contain erroneous annotations, and the rule-based criteria are not entirely reliable, as they may also erroneously penalize some correct operations. Consequently, further data cleaning is necessary. Specifically, we adopt GPT-4o [13] as a pre-critic model to judge the rightness of the operations in the collected samples, and we retain the samples for which the annotated scores are consistent with the judgment by GPT-4o. Finally, we denote the collected samples as $\mathcal{D}_{c\_action} = \{(\epsilon_n, a_n, l_n, s_n)\}_{n=1}^{N}$.

3.2.2 GUI Critiques Generation

The $\mathcal{D}_{c\_action}$ collected in the above section still lacks the critique c that elucidates why an action is considered as correct or incorrect. In this section, we resort to Chain-of-Thought (CoT) technique to enabling MLLMs to handle the challenging critique generation.

Progressive CoT Paradigm. We design a progressive CoT paradigm to helps MLLMs to perform deliberate, structured, and analytical thinking about the GUI operations. Specifically, our paradigm contains three parts `<thinking>...</thinking>` `<score>...</score>` `<suggestion>...</suggestion>`, which respectively corresponding to intermediate logical thought t, correctness score l, and suggestion s. The reasoning t contains the content of critique c. In particular, the logical reasoning in the <thinking> field is composed of several essential components:

1. `Observation: Analyze the screenshot's state, ignoring user instructions.`
2. `Possible Result: Speculate the most possible result of the action.`
3. `Critique: Assess correctness of the action with reasoning.`

`Observation` enhances the model's perceptual acuity by demanding a comprehensive understanding of spatial and contextual elements present in GUI. `Possible Result` boosts foresight by prompting the model to predict outcomes from current observations. Finally, `Critique` involves a critical analysis of actions to verify their correctness.

Reasoning Bootstrapping. Following the above format, we leverage current MLLMs to generate CoT reasoning progress. However, performing reverse annotation of CoT conditioned on ground-truth score l and suggestion s is harmful. This is because that the MLLMs may overly depend on pre-known annotations, potentially biasing the thought process toward these outcomes rather than reflecting the actual critique reasoning sequence. Therefore, we propose a reasoning bootstrapping strategy to generate high-quality thoughts without prior knowledge of the ground-truth l and s. Specifically, merely provided with environmental states ϵ and action candidates a, it forces the model to simulate a genuine critic reasoning process for max times as follows:

$$\mathbb{E}_{(\epsilon,a,l,s)\sim\mathcal{D}_{c\_action}}(\bar{t}_i, \bar{l}_i, \bar{s}_i)_{i=1}^{\max} = \bar{\pi}(\epsilon, a), \quad \text{Select}\,(\bar{t}_i, \bar{l}_i, \bar{s}_i) \mid (\bar{l}_i = l) \wedge (\bar{s}_i = s), \tag{1}$$

where $\bar{\pi}$ is an excellent MLLM and $\bar{t}$ denotes the generated thought. We select reasoning outputs containing the correctness score and suggestion that match with annotations to compile $\mathcal{D}_{c\_cot} = \{(\epsilon_m, a_m, l_m, s_m, t_m)\}_{m=1}^{M}$.

https://doi.org/10.52202/085713-0137

Finally, the GUI-Critic-Train dataset is constructed by aggregating the collected datasets, which can be represented as $\mathcal{D}_c = \mathcal{D}_{c\_action} \cup \mathcal{D}_{c\_cot}$. For the GUI-Critic-Test dataset that does not require the annotation of reasoning process, we construct it with the pipeline introduced in Section 3.2.1.

3.3 GUI-Critic-R1

Constructing a pre-critic model for GUI automation is not easy since it requires a thorough understanding of GUI knowledge, multimodal processing, and logical reasoning abilities. In this section, to cultivate the deliberative analysis ability of the pre-critic model for complex GUI scenarios, we propose a Suggestion-aware Group Relative Policy Optimization (S-GRPO) strategy for our GUI-Critic-R1. Specifically, we first employ Reinforcement Fine-Tuning Initialization (RFT cold-start) to stabilize the model's reasoning process. Then, the S-GRPO is adopted to further enhance the model's ability for GUI pre-critic.

3.3.1 RFT Cold-Start

We initiate model training with Reinforced Fine-Tuning (RFT) on collected GUI-Critic dataset $\mathcal{D}_c$:

$$\mathcal{L}_{rft} = \mathbb{E}_{(\epsilon,a,l,s)\sim\mathcal{D}_{c\_action},\ (\hat{\epsilon},\hat{a},\hat{l},\hat{s},\hat{t})\sim\mathcal{D}_{c\_cot}} - \log(\pi_{critic}(l,s|\epsilon,a)) - \log(\pi_{critic}(\hat{l},\hat{s},\hat{t}|\hat{\epsilon},\hat{a})). \quad (2)$$

The cold-start stage distills GUI knowledge from human annotations (*i.e.*, correct operations in $\mathcal{D}_{c\_action}$) and distill reasoning experience from existing MLMs (*i.e.*, progressive reasoning processes in $\mathcal{D}_{c\_cot}$), thereby equipping the model with foundational capabilities needed for generating operation critiques and valid feedbacks.

3.3.2 Suggestion-aware Group Relative Policy Optimization

After the RFT cold-start, we enable the pre-critic model to self-improve its reasoning ability via online reinforcement learning. Standard GRPO [8] approach samples a group of generated outputs for each input from the policy model, where each output includes CoT thought and an answer. Subsequently, the model is encouraged to favor better answers with a high reward value within the group. Since the pre-critic model is required to output additional valuable critiques and suggestions of the operation, only employing the typical reward that focus on the correctness of format and answer is not enough. Therefore, we introduce a suggestion-aware GRPO method, incorporating a novel suggestion reward specifically designed for GUI pre-critic.

Suggestion Reward. Given the suggestion annotation s', the suggestion reward r_s evaluates the model's constructive outputs as follows:

$$r_s(o) = \mathbb{I}_{similar}(s, s'), \quad (3)$$

where $o = (l, s, c)$ is the output of pre-critic model π_{critic}. $\mathbb{I}_{similar}(\cdot)$ calculates the similarity between the inputs leveraging large language models, and output a binary judgment. Based on Eq. (3), r_s provides a quantitative and objective metric to assess the correctness of suggestions, thereby facilitating more precise evaluations of π_{critic} efficacy. Moreover, this direct feedback enables models to learn from their outputs iteratively, refining their progressive CoT reasoning process to produce more accurate action critique and corrective suggestions.

Format and Accuracy Reward. Additionally, we adopt the rewards introduced in DeepSeek-R1 [8]: format reward $r_f(o)$ and accuracy reward $r_a(o)$. The former evaluates whether the outputs follow the structure described in Section 3.2.2. The latter evaluates whether the generated correctness score align with ground-truth. These rewards imposes constraints solely on the format and final answer, leaving the thought process unregulated, resulting in stimulating the model's intrinsic ability to reason.

Optmization. The final reward function $r(o)$ can be defined as: $r(o) = \lambda_f \cdot r_f(o) + \lambda_s \cdot r_s(o) + (1 - \lambda_f - \lambda_s) \cdot r_a(o)$, where λ_f and λ_s adjust the importance of the format and suggestion rewards. For each input (ϵ, a), we first sample a group of generated outputs $G = \{o_1, o_2, \cdots, o_G\}$ from the frozen policy model $\pi_{\theta_{old}}$, where each output $o = (c, l, s)$ includes critique c, score l and the suggestion s. Then we maximize the following objective and optimizes the critic model π_{critic}:

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{|G|}\sum_{i=1}^{G} \min\Big(\frac{\pi_{critic}(o_i \mid \epsilon, a)}{\pi_{\theta_{\text{old}}}(o_i \mid \epsilon, a)} A_i, \text{clip}\Big(\frac{\pi_{critic}(o_i \mid \epsilon, a)}{\pi_{\theta_{\text{old}}}(o_i \mid \epsilon, a)}, 1-\varepsilon, 1+\varepsilon\Big) A_i\Big) - \beta\, D\big(\pi_{critic} \,\|\, \pi_{\text{ref}}\big), \quad (4)$$

where ε and β are the PPO clipping hyperparameters and the coefficient controlling the Kullback–Leibler (KL) penalty, respectively. $D\big(\pi_{critic} \,\|\, \pi_{\text{ref}}\big) = \frac{\pi_{\text{ref}}(o_i|\epsilon,a)}{\pi_{critic}(o_i|\epsilon,a)} - \log\Big(\frac{\pi_{\text{ref}}(o_i|\epsilon,a)}{\pi_{critic}(o_i|\epsilon,a)}\Big) - 1$ is the KL divergence. The relative advantage for the i-th response is computed by normalizing the rewards across the group: $A_i = \frac{r(o_i) - \text{mean}(\{r(o_1),\ldots,r(o_G)\})}{\text{std}(\{r(o_1),\ldots,r(o_G)\})}$.

4 Experiment

4.1 Experiment Settings

Dataset and Benchmark. ***(1) GUI-Critic-Train.*** The data in GUI-Critic-Train were sourced from publicly available GUI operation datasets, including AITZ [54], AMEX [3], Odyssey [18], and AITW [28]. We utilize Qwen2.5-VL-7B [2] to generate negative operations, which were simply deemed incorrect based on rule-based criteria. GPT-4o [13] and Qwen2.5-VL-72B [2] are utilized to generate the Chain of Thought (CoT) processes for $\mathcal{D}_{c\_cot}$ outlined in Section 3.2.2. Consequently, GUI-Critic-Train dataset comprises approximately 11k entries, with 6k annotated by high-quality Chain-of-Thought (CoT) processes. ***(2) GUI-Critic-Test.*** To comprehensively evaluate the pre-critic capabilities of MLLMs, we established three main benchmark settings in GUI-Critic-Test: Mobile-Instruction Generalization (**GUI-I**), Mobile-Scenario Generalization (**GUI-S**), and Web-Scenario Generalization (**GUI-W**). Specifically, GUI-I test data are sourced from the AMEX [3], with instructions different from those in the training set. GUI-S test data are drawn from the Odyssey [18], comprising mobile applications that differ from those seen in the training set. Besides, shifting from mobile to web, GUI-Web contains the samples about the web automation, which are randomly sampled from the GUICourse [4]. Each of the settings, GUI-I, GUI-S, and GUI-Web, is manually annotated to ensure label accuracy, resulting in dataset sizes of 656, 114, and 418, respectively. We present critic accuracy and suggestion accuracy as the metrics. The former reflects the ability to assess the correctness of GUI operations, while the latter reflects the suggestion quality by quantifying the similarity between the generated suggestion and the annotation, which is calculated by prompting the Qwen2.5-VL-72B [2].

Implementation Details. We employ Qwen2.5-VL-7B [2] as the backbone. For the RFT cold-start, it is conducted for one epoch on mobile scenarios (GUI-I and GUI-S) and two epochs on web scenarios (GUI-W) via supervised fine-tuning (SFT). Next, we train the model on the $\mathcal{D}_{c\_action}$ by S-GRPO for 10 epochs, with a learning rate of $3e^{-6}$ and a batch size of 128. The suggestion and format reward weights (λ_s, λ_f) are both set to 0.1, while the group size is set to 6. In accordance with the methodology outlined in [57], the KL divergence coefficient is set to $1e^{-2}$ by default. All experiments are conducted on eight NVIDIA A100 Tensor Core GPUs.

4.2 Comparision Results

To analyze the performance of our GUI-Critic-R1 comprehensively, we construct experiments from both static and dynamic aspects. Firstly, we evaluate the model's ability to determine operational correctness and suggest the corrective operation on our static GUI-Critic-Test benchmark. Additionally, a crucial application of GUI-Critic lies in its functionality as a pre-critic model within the dynamic GUI automation processes. To this end, we implement online evaluation experiments on AndroidWorld [27] to validate the effectiveness of our approach.

4.2.1 Static Evaluation

Table 1 illustrates the quantitative results of our GUI-Critic-R1 and a variety of baselines on GUI-Critic-Test benchmark. It showcases that GUI-Critic-R1 performs competitively on different scenarios compared to closed- and open-source MLMs in both critic accuracy and suggestion accuracy. For close source MLMs, Claude-3.5 achieves best performance on most settings. Despite the capabilities of close source MLMs, our GUI-Critic-R1 achieves state-of-the-art (SoTA) performance on the GUI-I test dataset. Compared to the base model, GUI-Critic-R1 achieved a significant improvement in critic accuracy from 54.88% to 69.05%. Compared to the excellent GPT-4o, our model also achieves a 3.19% critic accuracy improvement and a 11.89% advantage in suggestion accuracy. When faced with GUI-S, the novel application scenarios containing complex cross-application instructions from Odyssey [18], our model demonstrated commendable generalization capabilities as well. Although

https://doi.org/10.52202/085713-0137

Table 1: Static evaluation performance comparison of closed-source and open-source MLLMs on our GUI-Critic-Test, including three different settings: GUI-I, GUI-S, and GUI-W.

Model	GUI-I		GUI-S		GUI-W	
	Critic Accuracy	Suggestion Accuracy	Critic Accuracy	Suggestion Accuracy	Critic Accuracy	Suggestion Accuracy
Close Source MLMs						
Claude-3.5	67.26	40.71	64.27	46.11	65.55	37.64
GPT-4o	66.01	40.54	62.28	33.33	68.45	28.27
Gemini-2.0-Flash	66.76	42.98	64.91	38.59	62.85	38.78
Open Source MLMs						
Deepseek-VL2-7B [41]	44.36	0.00	43.85	0.00	10.28	0.00
InternVL2.5-8B [5]	53.96	19.35	52.63	14.91	54.67	24.53
InternVL2.5-8B-MPO [5]	51.06	20.42	51.75	19.30	55.37	24.06
Qwen2.0-VL-7B [37]	52.59	21.04	54.42	19.30	53.50	38.78
Qwen2.0-VL-72B [37]	54.27	30.03	53.51	29.80	51.86	21.96
Qwen2.5-VL-7B [2]	54.88	43.14	57.02	37.72	59.11	36.21
Qwen2.5-VL-72B [2]	56.40	49.08	**59.65**	38.79	60.05	38.79
Ours (GUI-Critic-R1)	**69.20**	**52.43**	58.77	**47.37**	**63.08**	**39.48**
Δ (Ours - Qwen2.5-VL-7B)	+14.32	+9.29	+1.75	+9.65	+3.97	+3.27

our model achieves a relatively insignificant improvement in critic accuracy (1.75%), it performs a 9.65% advantage in suggestion accuracy. It demonstrates that our model has a robust understanding of operations even in a new environment. In the web scenarios, where domain differences are more significant, our model exhibits commendable performance, indicating that it not only enhances the ability to determine the operation correctness but also generates effective suggestions. These results validate the superiority and robustness of our proposed S-GRPO.

4.2.2 Dynamic Evaluation

We further evaluate our model on the AndroidWorld [27] benchmark, which provides a live Android emulator and 116 tasks across 20 mobile apps. Specifically, in this platform, the GUI agent can perform operations on an emulated Android phone to attain human instructions, and the results are evaluated automatically. Our pre-critic model can serve as an error diagnosis module in the framework in a plug-and-play manner. For a fair comparison, we conduct both post-operative and pre-operative critic with existing MLLMs as the baseline. The results in Table 2 illustrate that GUI-Critic-R1 achieves the best success rate on the benchmark, verifying the ability of error correction and suggestion of our model. We observe that both post- and pre-critic can improve the performance of the GUI agent, indicating that pre-critic can avoid some potential mistakes, and post-critic can also remedy some accomplished errors. Despite the capabilities of GPT-4o, we find that it can sometimes produce inaccurate critiques and suggestions to mislead the agent when performing as the pre-critic, primarily due to its insufficient common-sense knowledge of the GUI. GPT-4o outperforms the prior-critic on the post-critic because the prior-critic requires the model to predict the potential outcomes of GUI operations, which GPT-4o lacks the common-sense reasoning ability. Furthermore, we evaluated the efficiency of GUI task execution across different approaches. We introduced a metric called Efficiency Advantage Rate (EAR) for fair comparison, which measures the proportion that the 'baseline + critic' model achieves an efficiency advantage (fewer steps) than the baseline, for tasks where they achieve consistent results. The results shown in Table 2 reveal that our model tends to complete tasks in fewer steps, thereby demonstrating its efficiency. In contrast, post-critic methods typically require a greater number of steps.

Table 2: Dynamic evaluation results on the AndroidWorld [27] benchmark.

Model	SR(↑)	EAR(↑)
Baseline	22.4	-
Post-Critic		
GPT-4o [13]	25.0	18.3
Pre-Critic		
Qwen2.5-vl-7B [2]	20.3	21.8
Qwen2.5-vl-72B [2]	23.2	24.4
GPT-4o [13]	22.4	26.1
Ours	27.6	31.8

https://doi.org/10.52202/085713-0137

Table 3: Ablation study on the dataset collection pipeline.

Model	GUI-I		GUI-S	
	Critic Accuracy	Suggestion Accuracy	Critic Accuracy	Suggestion Accuracy
w/o NOS	50.46	1.22	50.30	5.26
w/o DF	67.23	49.54	54.39	42.98
w/o GCG	67.84	51.22	56.14	42.11
Ours	**69.20**	**52.43**	**58.77**	**47.37**

Table 4: Ablation study on training strategies.

RFT	S-GRPO			GUI-I		GUI-S	
	r_f	r_a	r_s	Critic Accuracy	Suggestion Accuracy	Critic Accuracy	Suggestion Accuracy
✓	✗	✗	✗	63.16	45.61	55.26	34.21
✗	✓	✓	✗	67.98	43.44	54.38	39.47
✗	✓	✓	✓	69.05	49.24	56.14	42.10
✓	✓	✓	✗	66.01	47.71	57.89	40.35
✓	✓	✓	✓	**69.20**	**52.43**	**58.77**	**47.37**

Figure 3: Analysis of suggestion reward weight λ_s and Group Size on the GUI-I setting.

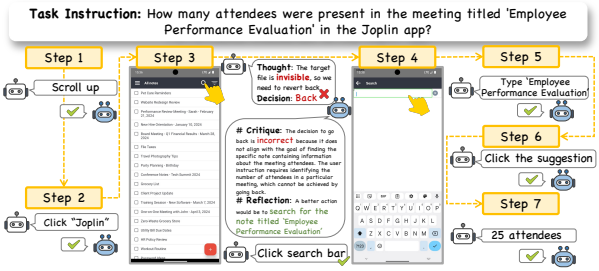


Figure 4: A case of pre-critic on GUI automation.

4.3 Ablation Study

Analysis of Dataset Collection Pipeline. We first conduct ablation studies for the proposed data collection pipeline to verify the contributions of three key components on GUI-I and GUI-S settings. Corresponding to Table 3, we first substitute Negative Operations Sampling (NOS) with a strategy of random decision replacements to acquire negative samples. It underscores that negative samples sampled by random substitution are too naive to undermine the model's capability to detect errors and suggest corrections. Secondly, we substitute the data filtering (DF) phase with a random sample selection approach. Without DF, the quality of training samples may be compromised, potentially degrading the effectiveness of the training process. Finally, we omit the GUI Critique Generation (GCG) process by utilizing only $\mathcal{D}_{c\_action}$ in the RFT cold-start stage. The absence of GCG leads to a 1.36% decline in critic accuracy on GUI-I, highlighting its crucial contribution to the model's generalization and cognitive capabilities. These findings collectively emphasize the pivotal importance of our data collection process, establishing it as an integrated mechanism for boosting model training and operational efficiency.

Analysis of Training Strategy. In order to analyze the impact of training strategies on model performance, we conduct ablation experiments focusing on the RFT cold-start and reward components in the GRPO stage. As depicted in Table 4, the results in the first and third lines indicate that the utilization of RFT cold-start provided a robust initial boost to the model's decision-making capabilities, and S-GRPO further increases the ability. Towards the second line, we observe that employing GRPO alone was insufficient to activate the model's GUI-Critic abilities, as the model does not yet possess the basic GUI critic ability, making it difficult to produce high-quality outputs when sampling randomly. In the fourth line, we ablate the suggestion reward, which led to an obvious decrease in suggestion accuracy, indicating this reward is necessary for the model to suggest correct GUI operations. These observations underscore the crucial significance of the RFT cold-start and the proposed suggestion reward in S-GRPO, incrementally enhancing the capability of our GUI-Critic-R1.

Analysis of parameters. Figure 3 illustrates the impact of different suggestion reward weight λ_s or the group size in the R-GRPO phase. Note that when varying λ_s, the weights for r_a and r_f are maintained at a ratio of 4:1, ensuring that the total sum of weights remains equal to 1. By adjusting λ_s, we observe that excessively low values inadequately constrain the suggestion, while excessively high values may compromise accuracy. Consequently, a value of 0.1 was selected as an optimal parameter. Additionally, the size of the group plays a crucial role in balancing performance with resource utilization. Therefore, a compromise was made by selecting a group size of 6.

https://doi.org/10.52202/085713-0137

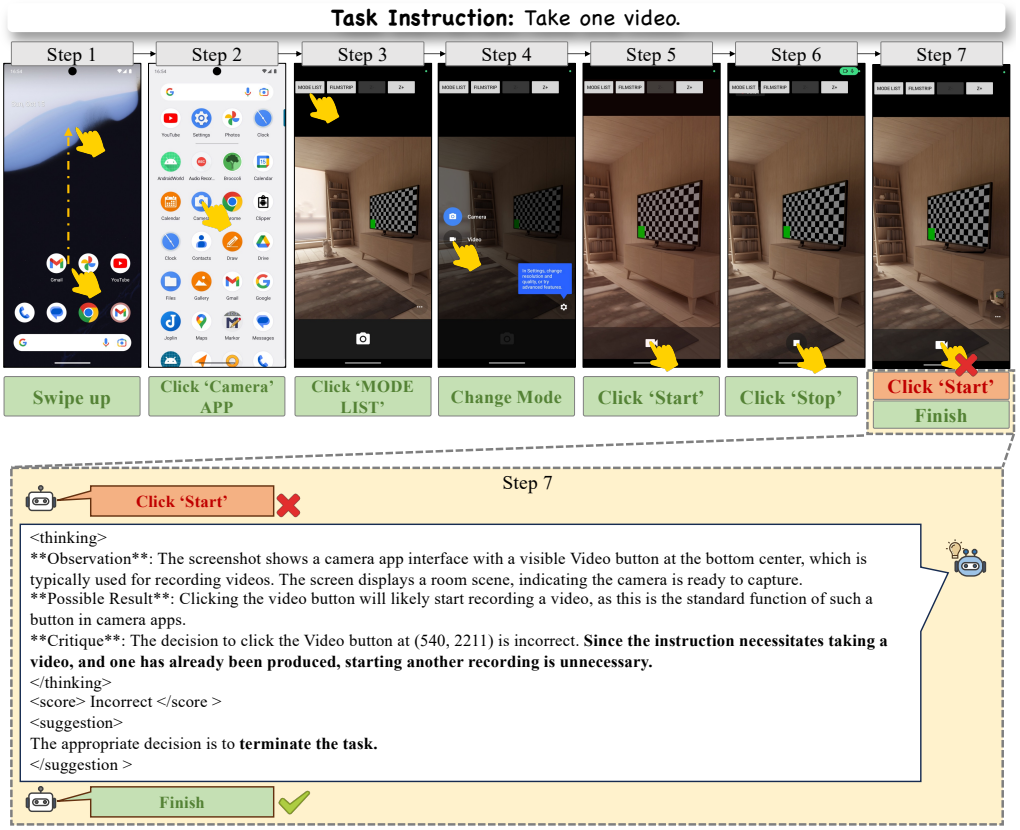


Figure 5: An instance of GUI-Critic-R1 avoiding a redundant action within GUI automation through a pre-operative critique approach.

4.4 Case Study

Figure 4 illustrates an example drawn from the AndroidWorld benchmark [27], demonstrating how our model guides the GUI agent to achieve a correct and effective trajectory. Specifically, the agent is tasked with finding a file in the Joplin app but encounters an interface without the target file visible, prompting it to mistakenly consider rolling back. Our model suggests clicking the search box to locate the target file, thereby assisting the agent in successfully completing the task.

In Figure 5, we depict another example of pre-operative critic for GUI automation. The agent successfully navigates steps 1 through 6, initiating the camera app and completing a video recording. Nonetheless, at step 7, the agent erroneously decides to press the record button again. This action is incorrect, as the instruction clearly stipulates recording only one video. Given that the agent has fulfilled this requirement, the task should be stopped rather than continuing the recording. Our model precisely delineates the state of the current interface and, based on historical operation, identifies the agent's decision as redundant. Consequently, our model advises the agent to terminate the task at this step.

5 Conclusion

In this paper, we introduce a pre-operative critic mechanism aimed at performing error diagnosis in GUI Automation, prior to executing actions within interactive environments. To develop our pre-critic model GUI-Critic-R1, we propose a Suggestion-aware Gradient Policy Optimization (S-GRPO) strategy with a novel suggestion reward. It is designed to ensure effective corrective discrimination for operations and provides reliable feedback for improvement. Futhermore, We develop a reasoning-bootstrapping based data collection pipeline to construct a GUI-Critic-Train and GUI-Critic-Test dataset, addressing existing gaps in GUI critic data. Through extensive static and dynamic experiments, we demonstrate the the efficacy of our approach across diverse GUI scenarios, highlighting its potential to significantly enhance the success rate of automated tasks.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grants 62322212, U23A20387, U25B2070), the Beijing Natural Science Foundation (No. L221013), and the CAS Project for Young Scientists in Basic Research (YSBR-116).

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. *arXiv preprint arXiv:2504.00906*, 2025.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[3] Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Dingyu Zhang, Peng Gao, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*, 2024.

[4] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.

[5] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

[6] Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-rest: Reflection-reinforced self-training for language agents. *arXiv preprint arXiv:2406.01495*, 2024.

[7] Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. Agentrefine: Enhancing agent generalization through refinement tuning. *arXiv preprint arXiv:2501.01702*, 2025.

[8] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[9] Priyanshu Gupta, Shashank Kirtania, Ananya Singha, Sumit Gulwani, Arjun Radhakrishna, Sherry Shi, and Gustavo Soares. Metareflection: Learning instructions for language agents using past reflections. *arXiv preprint arXiv:2405.13009*, 2024.

[10] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[11] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[12] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

[13] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[14] Noman Islam, Zeeshan Islam, and Nazia Noor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, 2017.

[15] Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024.

 https://doi.org/10.52202/085713-0137

[16] Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfeng Yuan, Changsheng Xu, Weiming Hu, et al. Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc. *arXiv preprint arXiv:2502.14282*, 2025.

[17] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025.

[18] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.

[19] Jie Ma, Pinghui Wang, Dechen Kong, Zewei Wang, Jun Liu, Hongbin Pei, and Junzhou Zhao. Robust visual question answering: Datasets, methods, and future challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[20] Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Coco-agent: A comprehensive cognitive mllm agent for smartphone gui automation. *arXiv preprint arXiv:2402.11941*, 2024.

[21] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[22] Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*, 2024.

[23] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, et al. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

[24] Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. *arXiv preprint arXiv:2404.06474*, 2024.

[25] Yingzhe Peng, Gongrui Zhang, Miaosen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang, Xingzhong Xu, Xin Geng, and Xu Yang. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. *arXiv preprint arXiv:2503.07536*, 2025.

[26] 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.

[27] Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024.

[28] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

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

[30] Yunpeng Song, Yiheng Bian, Yongtao Tang, Guiyu Ma, and Zhongmin Cai. 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*, pages 1–17, 2024.

[31] Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245, 2023.

[32] Fei Tang, Haolei Xu, Hang Zhang, Siqi Chen, Xingyu Wu, Yongliang Shen, Wenqi Zhang, Guiyang Hou, Zeqi Tan, Yuchen Yan, et al. A survey on (m) llm-based gui agents. *arXiv preprint arXiv:2504.13865*, 2025.

[33] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[34] Gladys Tyen, Hassan Mansoor, Victor Cărbune, Peter Chen, and Tony Mak. Llms cannot find reasoning errors, but can correct them given the error location. *arXiv preprint arXiv:2311.08516*, 2023.

[35] Haoyu Wang, Tao Li, Zhiwei Deng, Dan Roth, and Yang Li. Devil's advocate: Anticipatory reflection for llm agents. *arXiv preprint arXiv:2405.16334*, 2024.

[36] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *arXiv preprint arXiv:2406.01014*, 2024.

[37] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[38] Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025.

[39] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 543–557, 2024.

[40] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. Droidbot-gpt: Gpt-powered ui automation for android. *arXiv preprint arXiv:2304.07061*, 2023.

[41] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, Xin Xie, Yuxiang You, Kai Dong, Xingkai Yu, Haowei Zhang, Liang Zhao, Yisong Wang, and Chong Ruan. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding, 2024.

[42] Yufei Xiang, Yiqun Shen, Yeqin Zhang, and Nguyen Cam-Tu. Retrospex: Language agent meets offline reinforcement learning critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4650–4666, 2024.

[43] Tianyi Xiong, Xiyao Wang, Dong Guo, Qinghao Ye, Haoqi Fan, Quanquan Gu, Heng Huang, and Chunyuan Li. Llava-critic: Learning to evaluate multimodal models. *arXiv preprint arXiv:2410.02712*, 2024.

[44] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024.

[45] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

[46] Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*, 2025.

[47] Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.

[48] Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Minghua Ma, Guyue Liu, Qingwei Lin, et al. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279*, 2024.

[49] Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*, 2024.

 https://doi.org/10.52202/085713-0137

[50] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.

[51] Di Zhang, Junxian Li, Jingdi Lei, Xunzhi Wang, Yujie Liu, Zonglin Yang, Jiatong Li, Weida Wang, Suorong Yang, Jianbo Wu, et al. Critic-v: Vlm critics help catch vlm errors in multimodal reasoning. *arXiv preprint arXiv:2411.18203*, 2024.

[52] Jiaqi Zhang, Chen Gao, Liyuan Zhang, Yong Li, and Hongzhi Yin. Smartagent: Chain-of-user-thought for embodied personalized agent in cyber world. *arXiv preprint arXiv:2412.07472*, 2024.

[53] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025.

[54] Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*, 2024.

[55] Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. *arXiv preprint arXiv:2402.17574*, 2024.

[56] Jiani Zheng, Lu Wang, Fangkai Yang, Chaoyun Zhang, Lingrui Mei, Wenjie Yin, Qingwei Lin, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Vem: Environment-free exploration for training gui agent with value environment model. *arXiv preprint arXiv:2502.18906*, 2025.

[57] Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. Easyr1: An efficient, scalable, multi-modality rl training framework. https://github.com/hiyouga/EasyR1, 2025.

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 abstract and introduction clearly state the paper's contributions and scope.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations are discussed in the Appendix.

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]

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 have provided the necessary information for reproduction in Section **??**.

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: [No]

   Justification: Justification: The codes for reproducing all experimental results will be released as soon as the paper is accepted by NeurIPS.

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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the training and test details necessary to understand the results in Section 4.1.

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]

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 the compute workers details in Section 4.1, and the amount of compute are illustrated in Appendix.

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?

https://doi.org/10.52202/085713-0137

Answer: [Yes]

Justification: The research conforms with the NeurIPS Code of Ethics in every respect.

10. **Broader Impacts**

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

    Answer: [NA]

    Justification: The paper prioritizes discussing technical methodologies and results rather than societal impacts.

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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper exploring GUI automation on public datasets poses no such risks.

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 have properly credited the used assets. All the datasets introduced in this paper are available for free to researchers for non-commercial use.

13. **New Assets**

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

    Answer: [NA]

    Justification: The codes will be released once upon the paper's acceptance.

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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

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 rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes] .

    Justification: The paper uses LLM and describes in detail the type of model used (Sec 4.1) and prompt in Appendix.

Appendix

A Limitations and Future Works

In this paper, we explore the pre-operative critic mechanism for GUI automation to enhance success rates and decrease the number of operational steps. Future endeavors have the potential to extend our method to lighter models, such as Qwen2.5-VL-3B, to achieve more efficient and improved critic performance. Furthermore, our algorithm is based on single-step GUI visual information and semantic operation history.

Our approach can be enhanced by integrating the trajectory-level critic in future studies, which involves taking a sequence of screenshots as input. It has the potential to provide more comprehensive insights.

B More Dynamic Evaluation

In this paper, we conduct the dynamic evaluation on AndroidWorld [27] and illustrate the experiment results in the main text (Section 4.2.2). In this section, we introduce more details of the implementation of the experiments and more experimental results.

B.1 Supplementary Implementation Detials

We adopt the M3A (Multimodal Autonomous Agent for Android) framework introduced in AndroidWorld [27] as the backbone. In the M3A framework, the decision agent is provided with a list of available action types, guidelines for operating the phone, and a list of UI elements derived from the leaf nodes of the Android accessibility tree. The agent receives the current screenshot along with a Set-of-Mark annotated screenshot, which includes bounding boxes with numeric labels at the top-left corner of each UI element. At this stage, the agent attempts to execute the generated action, referencing specific marks. Prior to action execution, we insert a pre-operative critic to evaluate whether the agent's proposed action is conducive to achieving the instruction. If the action is deemed correct, it proceeds to execution as normal. Conversely, if the action is considered incorrect, we input the critic model's analytical recommendations to the decision agent, prompting it to reassess and formulate a new decision. After action execution, another agent is adopted to deliver a concise summary following the execution of the action. We leverage such semantic summaries to serve as a record of the action history. We follow the Androidworld to configure the UI element detection procedure and the definition of the action space.

B.2 Supplementary Experiments

We extend the dynamic evaluation experiment to incorporate GPT-4o [13] as the decision agent. As evidenced in Table 5, employing GPT-4o [13] as the decision-making agent yields performance benefits over Qwen2.5-VL-72B [2], underscoring GPT-4o's robust perception and reasoning capabilities. The pre-operative critic improves accuracy and exemplifies the efficacy of the baseline in detecting erroneous decisions and offering constructive improvement suggestions. In comparison with other pre-operative models, our GUI-Critic-R1 exhibits superior performance outcomes. When utilizing GPT-4o as the baseline, the advantage of the pre-critic in terms of the Efficient Advantage Rate (EAR) becomes more pronounced. GPT-4o tends to engage in excessive exploration during GUI automation, often opting to attempt further actions rather than terminating early, thereby resulting in a greater number of steps. In contrast, the pre-critic effectively anticipates erroneous or redundant exploration, thereby reducing unnecessary steps and manifesting a superior EAR metric.

Table 5: Supplementary dynamic evaluation results on the AndroidWorld [27] benchmark with GPT-4o [13] as the baseline.

Model	SR(↑)	EAR(↑)
Baselines		
Qwen2.5-VL-72B [2]	22.4	-
GPT-4o [13]	23.9	-
GPT4o+Post-Critic		
GPT-4o [13]	26.4	31.6
GPT4o+Pre-Critic		
Qwen2.5-VL-7B [2]	16.9	32.4
Qwen2.5-VL-72B [2]	25.9	36.8
GPT-4o [13]	24.0	43.1
Ours	**29.4**	**46.2**

https://doi.org/10.52202/085713-0137

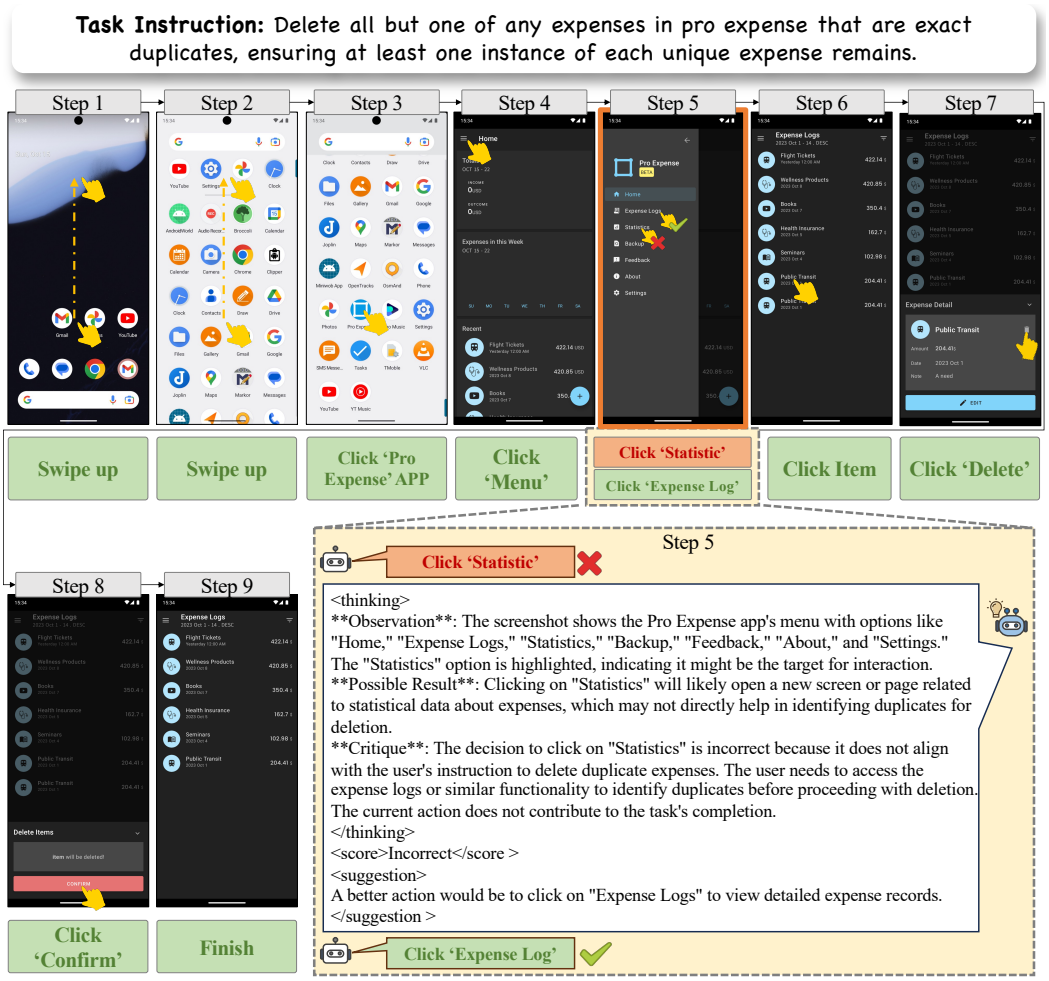


Figure 6: An instance of GUI-Critic-R1 correcting an erroneous decision within GUI automation through a pre-operative critique approach.

B.3 Additional Case Study

Figure 6 illustrates an operational process of GUI automation with pre-operative critic by our GUI-Critic-R1. In this example, the GUI agent initially demonstrates correct behavior by opening the Pro Expense app and accessing the menu. However, at step 5, the agent erroneously decides to click the "statistics" button, which is incorrect. Our GUI-Critic-R1 model successfully identifies this as an incorrect action and analyzes the reason behind the error. The model determines that selecting this button would navigate to a new interface displaying expense statistics, which is irrelevant to the instruction requiring the removal of duplicate expenses. Furthermore, our model provides a suggestion for correction: click on "Expense Logs" to view detailed expense records.

C GUI-Critic Dataset

To construct the GUI-Critic-Train and GUI-Critic-Test datasets, we extract step-level data from publicly available GUI operation datasets, encompassing user instructions, current interface screenshots, text-based operative history, and corresponding correct actions. In the following section, we provide an overview of the datasets employed in our study.

https://doi.org/10.52202/085713-0137

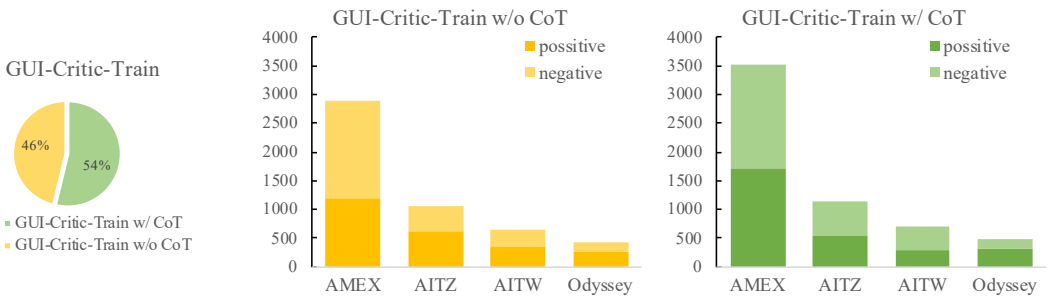


Figure 7: Illustration of the data composition for the GUI-Critic-Train dataset. The left displays the proportion of data without Chain-of-Thought (CoT) annotations versus with CoT. The central part shows the sources of data without CoT annotations, along with the ratio of correct to incorrect actions. The right section depicts the sources of data with CoT annotations, also detailing the ratio of correct to incorrect actions.

C.1 Public Datasets

Firstly, we review the publicly available datasets employed in our research.

Android in the Wild (AITW) [28] dataset comprises human demonstrations of device interactions, encompassing both screen captures and actions, alongside corresponding natural language instructions. It includes a substantial collection of 715,000 episodes covering 30,000 unique instructions, across four versions of the Android operating system (v10–13) and eight different device models ranging from the Pixel 2 XL to Pixel 6, each with distinct screen resolutions. The dataset involves complex multi-step tasks necessitating nuanced semantic understanding of both language and visual context.

Android-In-The-Zoo (AITZ) [54] dataset contains 18,643 screen-action pairs together with chain-of-action-thought annotations, spanning over 70 Android apps, coupled with 4× useful annotations compared with action coordinate labels only. Based on the screen episodes from AITW [28], the authors generate candidate answers for the screen descriptions, action thinkings and next action descriptions. These candidates are further validated and refined by human to guarantee alignment with the screenshots.

GUI Odyssey [18] is a comprehensive dataset for training and evaluating cross-app navigation agents. GUI Odyssey comprises 7735 episodes, meticulously curated from 6 different mobile devices such as Pixel Pro and Tablet. It encompasses 6 types of cross-app navigation tasks spanning from general system tool use to media entertainment, requiring navigating through 201 different apps and 1399 app combos from various fields such as video, music, reading

Android Multi-annotation EXpo (AMEX) [3] is a comprehensive, large-scale dataset designed for generalist mobile GUI-control agents. Their capabilities of completing complex tasks by directly interacting with the graphical user interface (GUI) on mobile devices are trained and evaluated with the proposed dataset. AMEX comprises over 104K high-resolution screenshots from 110 popular mobile applications, which are annotated at multiple levels. AMEX includes three levels of annotations: GUI interactive element grounding, GUI screen and element functionality descriptions, and complex natural language instructions, each averaging 13 steps with stepwise GUI-action chains.

GUICourse [4]. In this paper, the author constructed GUIAct, a GUI navigation dataset in website and android scenarios for enhancing VLMs' knowledge of GUI systems, including 67k single-step and 15k multi-step action instructions.

C.2 GUI-Critic Dataset Components

In this subsection, we delineate the origins, constituent parts, and foundational principles of the construction of our GUI-Critic-Train and GUI-Critic-Test datasets.

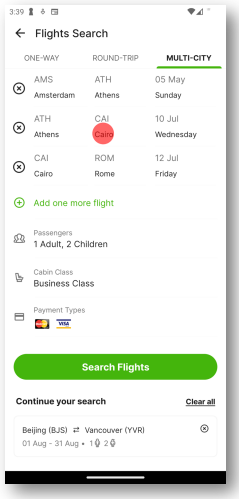
https://doi.org/10.52202/085713-0137

Instruction: Open Wego. Find multi-city flights from Amsterdam to Athens on May 5, from Athens to Cairo on May 8 and from Cairo back to Amsterdam on May 12. 1 adult. First class. Sort by price and filter only book on Wego. The price must be under US$3000. set as May 12.

History: Step 1: Tap on the 'Wego Flights' app icon to open it. Step 2: Tap on the back arrow to return to the main flight search screen. Step 3: Tap on the 'MULTI-CITY' option to change the search type. Step 4: Tap on the first city to change it to Amsterdam. Step 5: Type 'Amsterdam' into the search field. Step 6: Tap on the 'Amsterdam' option to select it. Step 7: Tap on 'GVA' to change it to 'Athens.' Step 8: Type 'Athens' in the search bar to set it as the arrival city. Step 9: Tap on the first option, 'Athens,' to select it as the destination city. Step 10: Tap on the destination city of the second segment and change Madrid to Cairo. Step 11: Swipe up to navigate to the month of May and locate May 8 on the calendar. Step 12: Tap on May 8, 2024, to select it as the departure date from Athens to Cairo. Step 13: Tap to change the destination from Madrid to Cairo in the second segment. Step 14: Select 'Cairo, Egypt' from the list of popular cities. Step 15: Select 'Cairo, Egypt' as the destination.

Action: {"action": "click", "coordinate": [639, 836]}

Screenshot marked with the candidate action.



Generated CoT process:

<thinking>
Observation: The screenshot is of the "Flights Search" interface within the Wego app. The search type is set to "MULTI-CITY." The current itinerary includes the following segments:
1. Amsterdam (AMS) to Athens (ATH), departing May 5, Sunday.
2. Athens (ATH) to Cairo (CAI), departing May 8, Wednesday.
3. Cairo (CAI) to Rome (ROM), departing July 12, Friday.
The red translucent marker is highlighting the city "Cairo" in the second segment of the itinerary. Below are options to add passengers, change cabin class (currently set to "Business Class"), and payment types.
The history so far involves setting the multi-city search and filling in destinations (Amsterdam, Athens, Cairo) for the first two segments. Based on the user instruction, the third flight segment should have Cairo as the departure city and Amsterdam as the destination, with the departure date set to May 12.
Possible Result: If the agent clicks on "Cairo" as marked, it will likely allow editing Cairo as the destination city for the second flight segment. However, this action is redundant since the city "Cairo" is already correctly assigned as the destination for the second segment. The focus should now shift to modifying the third flight segment to Cairo → Amsterdam with the departure date set as May 12.
This click neither promotes progress toward completing the intended user instruction nor aligns with the detailed requirements of modifying the third segment.
Critique: This decision is incorrect. Based on the history, Cairo is already correctly set as the destination for the second segment. The agent should now focus on correcting the third flight segment, currently set as Cairo → Rome on July 12, to reflect Cairo → Amsterdam on May 12 (per user instructions). Clicking "Cairo" in the second segment achieves no necessary action for advancing the core task.
</thinking>
<score> Incorrect </score >
<suggestion>
Change the third segment of the itinerary: Click on "ROM" and type "Amsterdam."
</suggestion>

Figure 8: A data sample in GUI-Critic-Train dataset.

C.2.1 GUI-Critic-Train

Figure 7 provides a visualization of the data sources utilized in our training set. For the construction of our training dataset, we predominantly selected data from AMEX[3] due to its extensive repository of high-quality GUI operation data. The raw data encompasses decision-making processes enriched with reasoning, which contributes to the reliability of the annotated correct actions. Detailed methodologies for the generation of incorrect decisions and the formulation of Chain-of-Thought (CoT) critiques are elaborated in the main text (Section 3.2). We preserved a higher proportion of samples containing incorrect decisions to facilitate the model's acquisition of enhanced error correction competencies. Figure 8 illustrates a data sample with the generated CoT process in GUI-Critic-Train dataset.

C.2.2 GUI-Critic-Test

Mobile-Instruction Generalization (GUI-I). Data in GUI-I originates from the AMEX dataset [3]. We ensure that the instructions in GUI-I are distinct from those in GUI-Critic-Train. After human annotation, GUI-I retains 656 GUI screenshots, each accompanied by a user instruction, operational history, and a set of candidate actions.

Mobile-Scenario Generalization (GUI-S). We collect 114 test data from the Odyssey dataset [18] for GUI-I. We select the data with applications different from those in GUI-Critic-Train. Specifically, the novel applications are:

1. *TikTok*: A popular social media application for sharing and watching short-form videos, fostering a global community of creators and viewers.
2. *Chaton*: An innovative application powered by intelligent conversation models, designed to enhance user interactions through AI-driven dialogues.

3. *ClevCalc*: A multifunctional calculator tool app that offers a wide range of mathematical operations and unit conversions for everyday use.
4. *Remix*: A versatile audio editing application that allows users to mix, cut, and enhance soundtracks with a variety of editing features.
5. *Redfin Houses*: A comprehensive app for real estate transactions, providing listings, pricing, and market insights for prospective home buyers and sellers.
6. *Tripadvisor*: A travel review platform app where users can rate and explore travel destinations, accommodations, and restaurants worldwide.

Web-Scenario Generalization (GUI-W). It contains 418 samples of web automation, which are randomly sampled from the GUICourse [4]. Although the action space in web environments differs from mobile platforms (e.g., limitations on swipe direction and the incorporation of double-tapping), the underlying operational logic largely remains consistent across both scenarios.

D Agent Prompt

In Table 6, Table 7, Table 8, and Table 9, we present the prompts for pre-critic, post-critic, action similarity evaluation, and adding critiques to the GUI agent decision process. Note that, we conduct minimal modifications to the pre-critic prompt, resulting in the post-critic prompt. These modifications include the removal of predictions concerning possible results and the adjustment of requirements from generating corrective suggestions to remedial suggestions in the `<suggestion>...</suggestion>`. The action similarity evaluation prompt is leveraged in calculating the suggestion reward in S-GRPO and the suggestion accuracy.

https://doi.org/10.52202/085713-0137

System

There is a multimodal agent that can perform a series of actions on a smart device (phone or PC) to automate the completion of user instructions.
Possible actions include "click" / "left_click" at (x,y) position, "long press" at (x,y) position, "swipe" from (x1,y1) to (x2,y2), "scroll" down or up, "drag" from (x1,y1) to (x2,y2), "type" (text content), "back", "home", "enter" and so on.
User instructions are usually complex and may include several detailed requirements. In some steps, the action decided by the mobile agent may be wrong.
Now, you are a critic model used to evaluate the agent's decision. I will provide you with the following information:
1. User instruction.
2. History: The action history of the agent in the previous steps.
3. Decision: The decision of the agent for this step.
4. Image: The screenshot before executing this action. If the action contains positional parameters (such as click and swipe), the interaction area is marked with a translucent red circle or red arrow.
Firstly, you need to understand the purpose of the decision. Pay attention to analyzing the interface elements in the screenshot (such as button position, text content, etc.). If there are red marks, focus on the action position. You can take appropriate account of the history information.
Then, based on the given information, carefully analyze the decision given by the agent for the current step:
1. Decision Analysis
(1). Observation: Observe the screenshot and analyze the state without considering the user's instruction.
- Focus on the operable or informative elements related to the operational decision.
(2). Possible Result: Speculate the most possible result of executing this decision.
- Predicts the screenshot change after the operation.
- Whether to promote the progress of core tasks.
(3). Critique: Determine whether the decision is correct and explain why.
- Focus on historical operations.
- Based on the previous analysis and the history, determine if this decision supports the completion of the instruction.
- Only perform actions specified in the instructions.
- Home button is the correct choice for switching apps.
- Both clicking a suggestion and Enter are correct when searching.
2. Based on the above analysis, determine whether this decision is "Correct" or "Incorrect".
3. Reflection: If correct, retell the action; if incorrect, suggest a better action. Propose a one-step action for the current observation, like click, swipe (with direction), type (with information), Home, Back, or Terminate (in 20 words).

Assess the current decision's correctness in the following format:
<thinking>
Observation: Describe the screenshot.
Possible Result: Analysis from the possible result perspective.
Critique: Criticize why the decision is correct or incorrect.
</thinking>
<score>
Correct or Incorrect
</score>
<suggestion>
If correct, provide a brief summary; if incorrect, suggest a better decision briefly.
</suggestion>

User

Below is the information for the current step:
1. User instruction: {User's instruction}
2. History: {Operation History}
3. Decision: {Action}
4. Image is the screenshot of this step. <image>

Table 6: The prompt for the pre-operative critic.

System

There is a multimodal agent that can perform a series of actions on a smart device (phone or PC) to automate the completion of user instructions.
Possible actions include "click" / "left_click" at (x,y) position, "long press" at (x,y) position, "swipe" from (x1,y1) to (x2,y2), "scroll" down or up, "drag" from (x1,y1) to (x2,y2), "type" (text content), "back", "home", "enter" and so on.
User instructions are usually complex and may include several detailed requirements. In some steps, the action decided by the mobile agent may be wrong.
Now, you are a critic model used to evaluate the agent's decision. I will provide you with the following information:
1. User instruction.
2. History: The action history of the agent in the previous steps.
3. Decision: The decision of the agent for this step.
4. Images: The screenshots before and after executing this action.
Firstly, you need to understand the purpose of the decision. Pay attention to analyzing the interface elements in the screenshot (such as button position, text content, etc.). If there are red marks, focus on the action position. You can take appropriate account of the history information.
Then, based on the given information, carefully analyze the decision given by the agent for the current step:
1. Decision Analysis
(1). Observation: Observe the screenshots and analyze the state without considering the user's instruction.
- Focus on the operable or informative elements related to the operational decision.
(2). Critique: Determine whether the decision is correct and explain why.
- Focus on historical operations.
- Ensure compliance with each specific requirement in the instructions.
- Only perform actions specified in the instructions.
- Home button is the correct choice for switching apps.
- Both clicking a suggestion and Enter are correct when searching.
2. Based on the above analysis, determine whether this decision is "Correct" or "Incorrect".
3. Reflection: If correct, retell the action; if incorrect, suggest a remedial action on the screen after executing this action. Propose a one-step action for the current observation, like click, swipe (with direction), type (with information), Home, Back, or Terminate (in 20 words).

Assess the current decision's correctness in the following format:
<thinking>
Observation: Describe the screenshots.
Critique: Criticize why the decision is correct or incorrect.
</thinking>
<score>
Correct or Incorrect
</score>
<suggestion>
If correct, provide a brief summary; if incorrect, suggest a remedial decision briefly.
</suggestion>

User

Below is the information for the current step:
1. User instruction: {User's instruction}
2. History: {Operation History}
3. Decision: {Action}
4. Images are screenshots before and after executing this action.

Table 7: The prompt for the post-operative critic.

The following two sentences describe the operation that a mobile agent needs to perform on mobile in order to complete a certain user instruction:
1. {Annotated Suggestion}
2. {Generated Suggestion}
Determine whether these two sentences describe a similar action? If yes, answer 1, if not 0, no explanation required.

Table 8: The prompt for evaluating the similarity between two action-descriptive sentences.

Note that your last decision may be incorrect! Please make a remedial decision based on the following Critiques, and consider adhering to the Suggestion provided: {Critique}

Table 9: The prompt for integrating critique into the GUI decision-making process.

https://doi.org/10.52202/085713-0137