
GUI-Reflection: Empowering Multimodal GUI Models with Self-Reflection Behavior

Supplementary Materials

Anonymous Author(s)

Affiliation

Address

email

1 This supplementary appendix includes (1) a comprehensive discussion of related works in Section A;
2 (2) a discussion about limitations in Section B; (3) a discussion about societal impacts in Section C;
3 (4) training details about training data and training hyperparameters in Section D; (5) implementation
4 details including model details of the model designs and evaluation details in Section E; (6) details
5 about the data construction process of the GUI-Reflection Task Suite in Section F; (7) details about
6 the data construction process of the GUI offline SFT stage in Section G; (8) details about the learning
7 environment and the reflection tuning algorithm in online stage in Section H; (9) detailed experiment
8 results on grounding benchmarks in Section I.

9 A Related Works

10 A.1 Mobile GUI Agents

11 Driven by the success of large language models and multimodal large language models, research in
12 mobile GUI automation has seen significant advancements. Current approaches to developing mobile
13 GUI agents can be broadly categorized into agent-based frameworks and end-to-end models, each
14 with distinct characteristics and trade-offs.

15 **Agent-based** By leveraging the advanced reasoning, planning, generalization capabilities, and broad
16 knowledge of foundation LLMs (*e.g.* GPT-4o), agent-based frameworks structure sophisticated
17 agentic workflows. One approach uses foundation models to directly engage with GUI interfaces
18 [42, 17, 33]. Such methods usually depend on accessible device information, particularly accessibility
19 trees, to allow the model to ground actions at the element level. These systems primarily consume
20 textual information from accessibility trees, often enriched with screenshots featuring Set-of-Mark
21 (SoM) augmentations for improved visual understanding. To enhance their operational efficacy, these
22 workflows frequently incorporate components such as memory [33], reflection [17, 33], knowledge
23 documents [42, 17], task decomposition [33], and visual tool integration [23], thereby improving task
24 completion and overall agent robustness.

25 Another agent-based approach [12, 38] combines a powerful foundation model with a specialized
26 GUI grounding model. In this setup, the LLM handles high-level planning and reasoning, while the
27 dedicated GUI grounding model is responsible for accurately identifying and interacting with GUI
28 elements based on a low-level instruction or element description from the LLM. UGround [12] trains
29 a universal GUI grounding model and combines it with a planning model. Aria-UI [38] improves the
30 grounding model by providing more task context, like overall task instruction and history information.

31 **End-to-end** End-to-end GUI models [14, 10, 37, 36, 44, 25] aim to directly map raw GUI inputs
32 (task information and screenshots) to grounded actions within a single model. The common paradigm
33 for these models involves a two-stage training process: 1) GUI-specific pre-training, where the
34 model learns fundamental GUI understanding and accurate grounding; 2) GUI offline SFT, where
35 the pre-trained model is fine-tuned on demonstration trajectories to learn task-specific behaviors.

Some methods [5, 34, 6] also apply reinforcement learning on the fine-tuned model, but experiments are only conducted in relatively simple and repetitive tasks. Beyond only predicting the atomic action, recent methods adopt the CoT idea to train the model to output additional components like the thinking process [43, 37, 25] and low-level action description [43, 37]. InfiGUIAgent [19] adds expectation-reflection components in the training data, but they only use the existing offline data, with most steps being error-free, and the reflection component primarily learns to confirm success rather than to actively diagnose and recover from a wide array of potential failures. UI-TARS [25] introduces an online bootstrapping process to learn the self-reflection and correction behaviors. However, unlike our proposed framework, which employs a fully automated pipeline for generating reflection data and integrates reflection capability enhancement across pre-training, offline SFT, and online stages, UI-TARS’s approach requires considerable human annotation efforts and focuses the learning of these behaviors only in their final online bootstrapping process.

A.2 LLM and MLLM Reasoning and Reflection

The pursuit of enhanced reasoning in Large Language Models (LLMs) has evolved from structured prompting and SFT data construction [35, 39, 40] towards leveraging Reinforcement Learning. While initial RLHF methods [30, 26] showed promise, recent paradigms focusing on outcome-based rewards have demonstrated great potential to intrinsically cultivate complex reasoning and even emergent self-reflection [13]. For the multimodal domains, current research is actively exploring how to adapt similar RL techniques to improve multimodal reasoning [15, 20, 24] involving visual information like images and videos, though challenges related to data and effective training signals remain. Furthermore, recent studies underscore the critical importance of the inherent capabilities and behaviors present in the base models before task-specific fine-tuning or reinforcement learning begins. Research indicates that foundational abilities for verification and reflection are not merely helpful but often prerequisites for successful online learning and significantly influence the ultimate performance ceiling achievable through RL [29, 41, 11]. This highlights a potential vulnerability in current end-to-end GUI model training pipelines, which often rely heavily on offline SFT with near error-free data. Such approaches may inadvertently suppress or fail to cultivate these vital reflective capabilities present in the base MLLM.

B Limitation

In this work, the constructed reflection-related data focuses primarily on visual and action-grounded errors or direct element functioning misunderstanding, potentially neglecting deeper and more complex errors, such as errors in high-level planning or complex task decomposition. Besides, our framework currently mainly focuses on mobile environments. While the underlying principles are generalizable, adapting GUI-Reflection to other platforms such as desktop systems or web-based interfaces may require domain-specific dataset construction and engineering adjustments.

C Societal Impacts

GUI-Reflection has the potential to improve digital accessibility and productivity by enabling more robust and error-tolerant GUI agents. However, these capabilities could also be misused for automated manipulation in malicious contexts. Furthermore, the reliance on synthetic data may introduce biases if not carefully curated, potentially leading to unintended behaviors in sensitive applications. Responsible deployment, transparency in usage, and alignment with human intentions are critical for maximizing societal benefit while minimizing risks.

D Training Details

Besides the reflection-related data we construct in the GUI-Reflection pipeline, the statistics of other data and the corresponding license information we used in the GUI pre-training and offline SFT stages are provided in Table 1 and Table 2.

For the GUI pre-training, we train the model for 1 epoch with a learning rate of 4×10^{-5} . For the SFT stage, we train the pre-trained model for 1 epoch with a learning rate of 3×10^{-5} . In each reflection tuning iteration, we train the model on the collected data in this iteration for 2 epochs with

Table 1: The detailed training data information for the GUI-Pretraining Stage. The total number reported is at the element level, while in implementation, the elements on the same image are grouped as a single training sample in the multi-turn conversation format.

Data Source	Platform	Task Type	Total Samples	License
UI RefExp [4]	Mobile	Grounding	16,660	CC BY 4.0
Widget Captioning [18]	Mobile	Grounding	96,648	CC BY 4.0
SeeClick-Rico [10]	Mobile	Grounding	173,275	CC BY 4.0
RICO Semantics [31]	Mobile	Grounding	31,560	CC BY-SA 4.0
OpenApp [7]	Mobile	Grounding	142,810	BSD-3-Clause license
AMEX [8]	Mobile	Grounding	1,360,595	CC BY 4.0
OS-Atlas [36]	Mobile	Grounding	89,860	Apache-2.0
Wave-UI [1]	Web	Grounding	79,412	MIT
Wave-UI-25K [2]	Web	Grounding	24,978	MIT
SeeClick-Web [10]	Web	Grounding	2,968,695	Apache-2.0
GUIEnv [9]	Web	Grounding	340,477	CC BY 4.0
ScreenQA [3]	Mobile	VQA	62,401	CC BY 4.0

Table 2: The detailed training data information for the offline SFT stage.

Data Source	Platform	Total Steps	License
AITW [28]	Mobile	19,831	CC BY 4.0
AITZ [43]	Mobile	14,686	CC BY 4.0
AMEX [8]	Mobile	39,023	CC BY 4.0
AndroidControl [16]	Mobile	89,603	Apache-2.0
GUI-Odyssey [22]	Mobile	102,202	CC BY 4.0

a learning rate of 1×10^{-5} . For our final model, we randomly sample 51694 samples from the offline SFT data and combine them with 63353 samples collected in the online iterations and finetune the model after offline SFT for 1 epoch with a learning rate 2×10^{-5} . We use AdamW [21] optimizer for all the training. All the training is conducted on 32 H100 GPUs. The pre-training stage takes about 11.5 hours, and the SFT stage takes about 8.5 hours. Each training iteration during the online training stage needs about 2 hours, and the final training stage takes 4 hours.

E Implementation Details

E.1 Model Details

The detailed descriptions of the valid actions for our GUI model are provided below.

Action Space

```

CLICK[[x, y]]. Click the screen at position [x,y].
LONG_PRESS[[x, y]]. Long press the screen at position [x, y].
SCROLL[[x1, y1, x2, y2]]. Scroll from the position [x1, y1] to [x2, y2].
TYPE[text]. Type in the text.
MEMORIZE[summary: text; content: text]. Store information into the memory.
ANSWER[text]. Answer with the text.
PRESS_HOME. Go back to the home screen.
PRESS_BACK. Go back to the previous screen.
OPEN_APP[app_name]. Open the app named app_name.
PRESS_ENTER. Press the enter key.
WAIT. Wait for device response.
TASK_COMPLETE. Indicate the task is completed.
TASK_IMPOSSIBLE. Indicate the task is impossible.

```

The input and output formats of our GUI agent model are shown below.

Input Format of the GUI Agent

```
<image>
<image>
The images are the screenshots from the past 2 steps.
<image>
The image is the current screenshot.
<INSTRUCTION> (user instruction): {goa}
<MEMORY> (stored memory content): {current memory}
<PAST ACTIONS> (past actions): {action history}
Based on the above information, your task is to reason about the next action
and provide your thinking process and the next action. Your output should
follow the following format:
<THOUGHT>: the thinking process
<ACTION DESC>: the description about the next action
<ACTION>: the next action
```

96

Output Format of the GUI Agent

```
<THOUGHT>: {action thought}
<ACTION DESC>: {action description}
<ACTION>: {action}
```

97

98 We define the action descriptions to follow fixed formats, and the formats for different action types
99 are shown below.

Action Description Format

```
CLICK: click the {element} to {purpose}
LONG_PRESS: long press the {element} to {purpose}
SCROLL: scroll {direction} to {purpose}
TYPE: type in the content '{content}'
MEMORIZE: memorize {memory_summary}
ANSWER: answer with the text '{text}'
PRESS_HOME: Go back to the home screen
PRESS_BACK: Go back to the previous screen
OPEN_APP: open the '{app_name}' app
PRESS_ENTER: press enter
WAIT: wait
TASK_COMPLETE: task complete
TASK_IMPOSSIBLE: task impossible
```

100

101 In our model implementation, the screenshot history length n is set to 4. All past screenshots except
102 the one in the last step are downsampled to 448×448 . We also visualized the click point on the past
103 screenshot using a red dot if the corresponding action is a click or long press. The coordinates in the
104 action representations are normalized to integers in the range 0 to 999.

105 E.2 Evaluation Details

106 The original scroll and swipe implementation in AndroidWorld [27] always uses a fixed trajectory,
107 so we modify it to make the scroll action trajectory follow the start and end points predicted by the
108 model. The original type action implementation includes clicking the target element and typing,
109 so we modify it to only include typing the text, and the model needs two actions (click + type) to
110 complete the original type action. We find that in some cases of AndroidWorld, the maximum steps
111 defined are impossible for agents that do not use the UI element information, so we increase the
112 maximum step by 5 for all test cases.

113 F Details of GUI-Reflection Task Suite

114 F.1 Action Verification

115 For the action verification task, we randomly sample step-wise data from AndroidControl [16] and
116 Odyssey [22] datasets. Each data sample consists of a ground truth action, the screenshot before the
117 action, and the screenshot after the action. We only consider action types including CLICK, LONG
118 PRESS, and SCROLL as the purposes of other actions are relatively fixed. Then, we extract the purpose
119 from the action descriptions of the ground truth actions to be the positive purpose. To construct the
120 negative purpose, we use Gemini-2.5-Pro to annotate the corresponding negative purpose for this
121 sample. The prompt for this annotation is shown in Table 5.

122 For the evaluation data of the action verification task, we have 603 positive samples and 603
123 corresponding negative samples from the Odyssey test split. For the training data, we construct 16220
124 paired samples from AndroidControl and 15616 paired samples from Odyssey. The evaluation and
125 training template for this task is shown in Table 6.

126 F.2 Action Reversal

127 The data for this task is constructed in two steps. First, we sample step-wise action data paired
128 with the screenshot before and after the action execution from existing datasets. Gemini-2.5-Pro
129 is instructed to generate the appropriate *undo* action for this data pair as the ground truth. Our
130 model is supposed to learn from this action reversal process. The annotation MLLM is instructed
131 to prioritize app-internal revert actions instead of overly relying on the general Press Back action
132 during this *undo* action generation process. When multiple revert actions are available, we select
133 the most straightforward and efficient option. After obtaining the correct *undo* action, we instruct
134 Gemini-2.0-Flash to generate interference options. The Press Back action is excluded from the
135 interference generation action space, as the functionality of the back button can vary significantly
136 across different applications. For this task, when the current action is CLICK, a semi-transparent red
137 circle is painted at the click location on the first screenshot, serving as a visual cue. We construct the
138 evaluation and training data from AndroidControl and Odyssey. We have 420 samples in total for the
139 evaluation part and 8642 samples for training. The task template for this task is shown in Table 7.

140 F.3 Mistake-informed Reattempt

141 For this task, we construct training data from the Wave-UI [1], AMEX [8], and OS-ATLAS-Desktop
142 [36] datasets and directly evaluate on the grounding benchmark ScreenSpot [10] and ScreenSpot
143 [36]. To obtain the failed attempts in training data, we first train a GUI-Pretrain model with the three
144 target datasets for this task excluded. Then we conduct inference on these three datasets and select
145 samples with failed predictions. We also use the bounding boxes of other elements annotated on the
146 same image as mistake candidates. To construct mistake-informed training data, for each sample,
147 we randomly choose 1 to 5 failed attempts and provide these mistakes in the prompt. The bounding
148 boxes of the mistakes are also drawn using red rectangles on the image. We have 31836 samples in
149 total for the training of this task. The task template is shown in Table 8.

150 G Details of Reflection Data in Offline SFT

151 We use two approaches to construct reflection data in the offline SFT stage. For the first approach, we
152 first adopt Gemini-2.5-Pro [32] (prompt shown in Table 9) to modify the original goal to make the
153 action incorrect. With the modified goal \tilde{G} , we further generate the reflection action at $t + 1$ with the
154 prompt shown in Table 10. If the reflection action \tilde{a}_{t+1} is Press Back, we assume $\tilde{I}_{t+2} = I_t$. Then
155 at step $t + 2$, the agent needs to summarize the previous mistake and make an informed new attempt.
156 To obtain such action annotation, we first generate the correct action \bar{a}_t at step t after the goal is
157 modified to \tilde{G} using the prompt shown in Table 11. The action \bar{a}_t does not contain the reflection part
158 about the mistake, so we further modify the action thought $\tilde{a}_t^{thought}$ to $\tilde{a}_{t+2}^{thought}$ while keeping the
159 action description and grounded action by adding reflection content using the prompt provided in
160 Table 12.

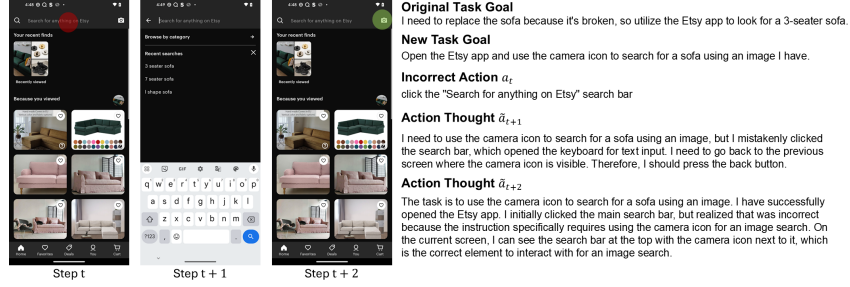


Figure 1: Example of the reflection data generated with the first approach in the offline SFT stage. The click point of the incorrect action is highlighted with a red circle on the first screenshot, and the click point of the action at step $t + 2$ is highlighted with a green circle in the third screenshot.

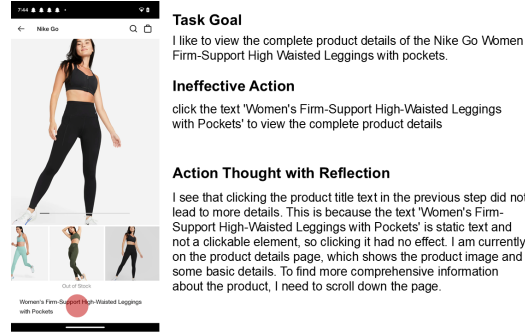


Figure 2: Example of the reflection data generated with the second approach in the offline SFT stage. The click point of the ineffective action is highlighted with a red circle on the first screenshot.

For the second approach, we first create an ineffective incorrect action using the prompt in Table 13. And then we modify the original $a_t^{thought}$ by adding reflection content about additional inserted ineffective action using the prompt in Table 14.

We build reflection data from the AndroidControl dataset and obtain 17557 samples with the first approach and 15394 samples with the second approach in total. Examples of the reflection data generated via these two approaches are provided in Fig 1 and Fig 2.

H Details of Online Iterative Reflection Tuning

The Apps in our online environment with the task statistics and examples are shown in Table 18.

For the MLLM-based verifier, we adopt Gemini-2.0-Flash as the MLLM and provide the complete sequence of screenshots, task goal, task guidance, and action sequence to it for judgment. The prompt for this process is provided in Table 15.

In our online reflection tuning algorithm, we further check the step-wise correctness for those successful trajectories using Gemini-2.0-Flash with the prompt shown in Table 16. For unsuccessful trajectories, we use GPT-4o and the prompt in Table 17 to identify the first error step. The pre-correction and post-reflection annotations are similar to the process used in Sec G with additional task guidance provided for more accurate annotation.

I Detailed Experiment Results

We provide detailed evaluation results on ScreenSpot and ScreenSpotv2 in Table 3 and Table 4.

Table 3: Detailed evaluation results on ScreenSpot.

Model	Mobile		Desktop		Web		Avg
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
InternVL2.5-8B	81.32	52.40	46.39	30.00	43.48	25.24	46.47
InternVL3-8B	91.57	75.11	76.80	52.86	77.39	55.82	71.59
GUI-Pretrain	93.04	75.54	91.24	72.86	89.56	77.67	83.31
GUI-Pretrain-Ref	94.50	79.04	91.24	69.29	89.56	79.61	83.87

Table 4: Detailed evaluation results on ScreenSpotv2.

Model	Mobile		Desktop		Web		Avg
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
InternVL2.5-8B	80.69	57.34	45.36	29.28	35.47	27.58	45.95
InternVL3-8B	92.76	79.15	75.26	52.14	75.21	57.63	72.02
GUI-Pretrain	96.20	79.62	93.30	71.43	91.02	78.33	84.98
GUI-Pretrain-Ref	96.89	82.94	93.81	71.43	91.03	82.27	86.39

Table 5: Prompt for negative purpose annotation

<p>## System Role You are an expert in understanding GUI operations.</p> <p>## Information Provided You will receive the following information:</p> <ol style="list-style-type: none"> 1. Two screenshots: the first screenshot corresponds to the current step, and the outcome screenshot corresponds to the next step. 2. Current Action: the action executed at the current step 3. Current Action Purpose: The purpose of executing the current action <p>The current action successfully achieved the current purpose (can be seen from the outcome of this action, which is the outcome screenshot). Your task is to come up with a new purpose, such that it is not satisfied according to the outcome screenshot. Note that your new purpose should be somewhat difficult and confusing, such that someone who is unfamiliar with the GUI operations or functions might think it is satisfied by the current action executed. But make sure your new purpose has no ambiguity so an expert can determine that it is not satisfied based on the screenshots.</p> <p>Note that the new purpose should be a verb phrase (starting with a verb) describing the overall purpose and it should **not** be some direct low-level action instructions like 'click the xxx'.</p> <p>Make sure your new purpose is NOT satisfied after the execution of the current step.</p> <p>!!! Important</p> <p>Make sure your new purpose can actually be satisfied by doing some **single action** different from the current action on the **first screen**.</p> <p>You also need to provide a simple explanation about what single action on the current step can satisfy your new purpose and why an expert can tell from the outcome screen that your new purpose is not satisfied.</p> <p>## Input: Current Action: {action description} Current Action Purpose: {action purpose}</p> <p>## Output Format: Directly output your results in json format as follows:</p> <pre>{ "new purpose": "your new purpose", "explanation": "your explanation" }</pre>
--

Table 6: Task template for Action Verification

<p><image> <image></p> <p>Given an action purpose and two screenshots (the first screenshot corresponds to the step before a certain action while the second one is the outcome screenshot after the execution of the action). You need to judge whether the action purpose has been satisfied by the action executed between these screenshots based on the screenshots content. Directly answer Yes or No.</p> <p>## Input action purpose: {action purpose}</p> <p>## Output:</p>

Table 7: Task template for Action Reversal

<p><image></p> <p><image></p> <p>You are an expert in evaluating the behavior of GUI agents that interact with Android phone interfaces. Your task is to assist in training intelligent agents by identifying the correct revert operation to undo a previously executed incorrect operation.</p> <p>You will be given:</p> <ul style="list-style-type: none"> - A current action that was performed - A pair of screenshots: <ul style="list-style-type: none"> - Screen A: the UI before the action - Screen B: the UI after the action - A set of six revert operation action choices (A-F) - The agent must choose the one correct revert operation that best reverts Screen B back to Screen A. <p>Valid Action Space:</p> <ul style="list-style-type: none"> - Open app[app]: Open the specified app. - Click: Tap on a specific UI element. - Long Press: Long press on a specific UI element. - Scroll: Perform a scroll gesture on the screen. - Type[text]: Input the specified text into a text field. - Press Home: Return to the home screen. - Press Back: Return to the previous screen. - Press Enter: Confirm input using the enter/return key. <p>Evaluation Criteria:</p> <ul style="list-style-type: none"> - The correct revert operation must be the most effective and reasonable way to return the UI from Screen B to Screen A, based on the change caused by the current action. - Only one option is correct. The remaining five should be plausible but incorrect. - Evaluate the options based on: <ul style="list-style-type: none"> - Whether the action targets the correct UI element - Whether it reverses the effect of the current action <p>Input:</p> <ul style="list-style-type: none"> - Current Action: {action_desc} - Choices: A-F options {undo_options} <p>Output Format: Directly output the option letter only</p>
--

Table 8: Task template for Mistake-informed Reattempt

<p><image></p> <p>You are given a screenshot of a mobile phone screen, a question, and some incorrect answers have already been excluded for you.</p> <p>You need to give a correct answer based on the screenshot. Notice that the correct answer should be different from the incorrect answers.</p> <p>The question is: {grounding instruction}</p> <p>The incorrect answers (also annotated using red bbox in the image) are: {incorrect answer}</p> <p>The correct answer is:</p>
--

Table 9: Prompt for instruction modification.

System Role

You are an expert in understanding the operations from a GUI agent. The agent’s task is to help human users operate an Android phone by completing specific instructions provided by the user. Your goal is to modify the original instruction such that the action taken by the agent becomes incorrect in the context of your modified instruction.

Valid Action Space

The actions that the agent may take to complete the task are as follows:

Open app[app]: Open the ‘app’ APP

Click: Click on the current screen.

Long Press: Long press on the current screen.

Scroll: Scroll on the screen.

Type[text]: Type the ‘text’ into the input field.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something

Task Complete/Task Impossible: The agent indicates the task is completed or impossible.

Information Provided

You will receive the following information:

1. Task Instruction: The original task instruction for the GUI agent.
2. Past Actions: The past actions taken by the agent to complete this task before this step. Empty if it is the first step.
3. Future Actions: The future actions taken by the agent to complete this task after this step. Empty if it is the last step.
4. Current Action: The action taken by the agent at the current step.
5. Screenshots: The screenshot at the current step and the screenshot at the next step after the current action is executed. The click point is highlighted using a red dot in the current screenshot and the scroll positions are visualized using a red arrow in the current screenshot.

Your job is to analyze the provided information to provide a modified instruction. With the modified instruction, the current action becomes incorrect, while the past actions are still correct (compatible with the modified instruction).

Incorrect action means that the current action taken by the agent is clearly wrong or unnecessary for completing the modified instruction or deviates from the correct way.

! Important

Your modified instruction should be natural, reasonable, and also realistic. With your modified instruction, the now-incorrect action (**current action**) becomes an easy or natural mistake a user who is unfamiliar with the App, button functions, or certain operations might make. The user will probably realize the mistake when seeing the execution of this action (the second screenshot).

Useful Guidelines

To make the evaluation process more accurate, please follow these guidelines:

1. Based on the information, judge the feasibility of having a reasonable instruction meeting the mentioned requirements. Ignore the following if not.
2. Provide your modified instruction.
3. Explain why the modified instruction is reasonable and realistic, and does not change the correctness of the past actions.
4. Explain why the **current action** becomes incorrect with your modified action, and explain why it is an easy or natural mistake a user might make.

Note that the red dot and red arrows are just for visualizing the actions; do not mention them in your response.

Note that the future actions are provided to better understand the overall task and context; you do not need to consider them when creating modified instruction.

Note that your modified instruction should clearly be possible for the agent to complete based on the provided information.

Input:

Task Instruction: {task_instruction}

Past Actions: {past_actions}

Future Actions: {future_actions}

Current Action: {action}

Output Format:

Directly output your results in json format as follows:

```
{{
  "Task Feasibility": "Yes or No", if no, the following entries should be empty. "Modified
  Instruction": "Your modified instruction",
  "Explanation 1": "Explain why the modified instruction is good and compatible with action
  history.",
  "Explanation 2": "Explain why the current action becomes a natural mistake for a user not
  familiar with the app and certain operations."
}}
```

Table 10: Prompt for reflection action annotation \tilde{a}_{t+1} of the first approach in offline SFT.

System Role

You are an expert in correcting the step-wise operation of a GUI agent. The agent’s task is to help human users operate an Android phone by completing specific instructions provided by the user but it performs an incorrect action at a certain step. Your task is to make a reflection about the previous incorrect step and reason about the next correct atomic action after this mistake, provide the action thought, and the action type.

Definition of Action Thought

The action thought is the rationale behind the actual action taken at a certain step. The action thought should be a compact paragraph consisting of 3-4 sentences. It could include the following aspects if they are helpful and important for the action reasoning process: - observation: a concise description of the current screenshot, focusing on the task related content and progress.

- reflection: a simple analysis of the unexpected situation caused by the incorrect action in the previous step, admit and analyze the mistakes.

- action rationale: a brief reasoning process to integrate the above information and provide a natural thinking process leading to the actual action from the perspective of the operator. The action thought should be natural and logically fluent. The action thought should *not* be clearly separated into the above aspects and does not need to mention the keywords like observation and reflection.

Information Provided

You will receive the following information:

1. Task Instruction: The overall task instruction for the GUI agent.
2. Past Actions: A list of past actions prior to the previous incorrect step. Empty if it is the first step.
3. Previous Incorrect Action: The incorrect action taken at the previous step.
4. Error Action Analysis: An analysis from an expert explaining why the previous action is incorrect.
5. Screenshots: The first screenshot corresponds to the step before the incorrect action is taken while the second screenshot corresponds to the current step after executing the incorrect action. The click point is highlighted using a red dot in the first screenshot.

Your job is to analyze the provided information and provide the action thought leading the agent should take at the current step after the execution of the previous incorrect action.

Valid Action Space

The actions that the agent may take to complete the task are as follows:

Click: Click at a certain position on the current screen.

Long Press: Long press at a certain position on the current screen.

Scroll: Scroll on the screen, scroll down/up/left/right, where the direction is the opposite direction of the figure movement.

Type[text]: Type the 'text' into the input field.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something.

Task Complete/Task Impossible: Indicate the task is completed or impossible.

Useful Guidelines

To make your reasoning and response more accurate, please follow these guidelines:

1. Analyze the provided information

2. Provide your action thought. In the action thought, use first-person description, that is, using 'I' instead of 'the user' or 'the agent'

3. Provide your action type which must be consistent with your action thought.

The correct action type should be one of the following: [Click, Long Press, Scroll, Type, Press Home, Press Back, Press Enter, Wait, Task Complete, Task Impossible]

Important Notes:

Note that the red dots for click and long press are only for better understanding of the actions; do not mention them in your action thought.

Note that some sub-task might need multiple actions (e.g. typing something needs clicking the text input field and then typing) and you should only give the very first atomic action for the current step.

Note that if the previous step type in some incorrect content, you have to first find ways to clear it before typing the correct one.

Note that when you want to click or press the backspace button, the correct action type is click instead of Press Back (which means going back to the previous page). And in such cases, try to use the word 'click' instead of 'press' in your action thought.

Input:

Task Instruction: {task_instruction}

Past Actions: {past_actions}

Previous Incorrect Action: {incorrect_action}

Error Action Analysis: {error_analysis}

Output Format:

Directly output your results in json format as follows:

```
{{
  "Action Thought": "The action thought"
  "Action Type": "The action type"
}}
```

Table 11: Prompt for correct action annotation \tilde{a}_t of the first approach in offline SFT.

System Role

You are an expert in correcting the step-wise operation of a GUI agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user but it performs an incorrect action at a certain step. Your goal is to reason about the correct action and provide the reasoning process leading to the correct action at this step.

Definition of Action Thought

The action thought is the rationale behind the actual action taken at a certain step. The action thought should be a compact paragraph consisting of 3-4 sentences. It could include the following aspects if they are helpful and important for the action reasoning process:

- observation: a concise description of the current screenshot, focusing on the task related content
 - reflection: a simple analysis of whether the previous step's action is successful and as expected
 - progress analysis: a brief summary of the progress towards the overall goal before taking the action at the current step and a plan about the sub-tasks that still need to be done in order to complete the final goal
 - action rationale: a brief reasoning process to integrate the above information and provide a natural thinking process leading to the actual action from the perspective of the operator
- The action thought should be logically fluent, and it does not necessarily include all the above aspects. The action thought should *not* be clearly separated into the above aspects and does not need to mention the keywords like observation and reflection.

Information Provided

As an evaluator, you will receive the following information: 1. Task Instruction: The overall task instruction for the GUI agent.

2. Past Actions: A list of past actions prior to this step. Empty if it is the first step.

3. Screenshots: The first screenshot corresponds to the previous step while the second screenshot corresponds to the current step. Only the current screenshot is provided if it is the first step. The click point is highlighted using a red dot in the first screenshot.

Your job is to analyze the provided information and provide the correct action thought leading to the correct action that the agent should take.

Valid Action Space

The actions that the agent may take to complete the task are as follows:

Click: Click at a certain position on the current screen.

Long Press: Long press at a certain position on the current screen.

Scroll: Scroll on the screen, scroll down/up/left/right, note that the direction is the opposite direction of the figure movement.

Type[text]: Type the 'text' into the input field.

Open App[app]: Open the 'app' App.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something.

Task Complete: Indicate the task is completed

Task Impossible: Indicate the task is impossible.

Useful Guidelines

To make your reasoning and response more accurate, please follow these guidelines:

1. Based on the provided information, provide the correct action thought. In the action thought, use first-person description, that is, using 'I' instead of 'the user' or 'the agent'

2. Provide the correct action type which must be consistent with your action thought.

The correct action type should be one of the following: [Click, Long Press, Open App, Scroll, Type, Press Home, Press Back, Press Enter, Wait, Task Complete, Task Impossible]

Important Notes

Note that when you want to click or press the backspace button, the correct action type is click instead of Press Back (which means going back to the previous page). And in such cases, try to use the word 'click' instead of 'press' in your action thought.

Input:

Task Instruction: {task_instruction}

Past Actions: {past_actions}

```

## Output Format:
Directly output your results in json format as follows:
{{
  "Action Thought": "The correct action thought"
  "Action Type": "The correct action type"
}}

```

Table 12: Prompt for adding reflection to action thought $\tilde{a}_t^{thought}$ of the first approach in offline SFT.

```

## System Role
You are an expert in correcting the step-wise operation of a GUI agent. The agent’s task is to help human users operate an Android phone by completing specific instructions provided by the user. The agent performs an incorrect action at a certain step, and it needs to execute the press back action to go back to the previous normal state. Your task is to add some reflection content about the previous incorrect action to the correct action thought.

## Definition of Action Thought
The action thought is the rationale behind the actual action taken at a certain step. The action thought should be a compact paragraph consisting of 3-4 sentences. It could include the following aspects if they are helpful and important for the action reasoning process:
- observation: a concise description of the current screenshot, focusing on the task related content
- reflection: a simple analysis of whether the previous step’s action is successful and as expected
- progress analysis: a brief summary of the progress towards the overall goal before taking the action at the current step and a plan about the sub-tasks that still need to be done in order to complete the final goal
- action rationale: a brief reasoning process to integrate the above information and provide a natural thinking process leading to the actual action from the perspective of the operator
The action thought should be logically fluent, and it does not necessarily include all the above aspects. The action thought should *not* be clearly separated into the above aspects and does not need to mention the keywords like observation and reflection.

## Information Provided
You will receive the following information:
1. Task Instruction: The overall task instruction for the GUI agent.
2. Incorrect Action: The previous incorrect action
3. Error Action Analysis: An analysis from an expert explaining why the previous action is incorrect.
4. Correct Action Thought: The actual correct action thought leading to the correct action
5. Screenshot: The screenshot after performing the press back action to go back to the normal status

Your job is to analyze the provided information and modify the correct action thought by adding some reflection and lessons learned about the incorrect action narrated in the first person as if you have performed the previous incorrect action, realize the mistake, and press back to go to the current status.

## Important Notes:
Your modified action thought should still be consistent with the provided correct action thought.

## Input:
Task Instruction: {task_instruction}
Incorrect Action: {incorrect_action}
Error Action Analysis: {error_analysis}
Correct Action Thought: {correct_action_thought}

```

```

## Output Format:
Directly output your results in json format as follows:
{{
"Updated Action Thought": "The updated action thought with reflection added"
}}

```

Table 13: Prompt for creating ineffective incorrect action of the second approach in offline SFT.

```

## System Role
You are an expert in understanding the operations from a GUI agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user. Your goal is to come up with an incorrect ineffective Action that is different from the correct action and would not change the current screen.

## Candidate Actions
You may consider the following actions
- click: click on the current screen.
The complete format of this action is "click <element> to <purpose>", where <element> is a noun phrase starting with 'the' indicating the clicked element, and <purpose> is a verb phrase indicating the direct purpose and expected outcome of clicking this UI element.

- long press: long press on the current screen.
The complete format of this action is "long press <element> to <purpose>", where <element> is a noun phrase starting with 'the' indicating the pressed element, and <purpose> is a verb phrase indicating the direct purpose and expected outcome of long pressing this UI element.

- scroll: scroll on the screen.
The complete format of this action is "scroll <direction> to <purpose>", where <direction> is one of the following directions: ['up', 'down', 'left', 'right'] and the direction is the opposite direction of the movement of the finger. <purpose> is a verb phrase indicating the direct purpose and expected outcome of this scrolling action.

- type: type some 'text' into the input field.
The complete format of this action is "type in the <content>", where <content> is the typed text.

## Information Provided
You will receive the following information: 1. Task Instruction: The original task instruction for the GUI agent.
2. Past Actions: The past actions taken by the agent to complete this task before this step. Empty if it is the first step.
3. Future Actions: The future actions taken by the agent to complete this task after this step. Empty if it is the last step.
4. Current Action: The action taken by the agent at the current step.
5. Screenshot: The screenshot at the current step. The click point is highlighted using a red dot in the current screenshot.

Your job is to analyze the provided information and decide whether it is possible to come up with an incorrect ineffective action.
The requirement of the incorrect ineffective action:
1. The action is incorrect; it is clearly wrong or unnecessary for completing the task or deviates from the correct way.
2. It is an easy or natural mistake a user who is unfamiliar with the App, button functions, or certain operations might make. The user will probably realize the mistake when seeing the execution of this action.

```

3. This action has no effect on the current screen, which means the current screen will remain exactly the same after executing the incorrect action.

Some examples: scroll down while it is already at the bottom, click the entry name instead of the actual text field for entering information, type in something without activating the input field yet.

Useful Guidelines

To make the evaluation process more accurate, please follow these guidelines:

1. Based on the information, judge the feasibility of having an incorrect ineffective action meeting the mentioned requirements. Ignore the following if not.
2. Provide your incorrect ineffective action if possible.
3. Explain why the incorrect ineffective action is incorrect, and explain why it is an easy or natural mistake a user might make.

Note that you have to strictly follow the complete format for your action.

Input:

Task Instruction: {task_instruction}

Past Actions: {past_actions}

Future Actions: {future_actions}

Current Action: {action}

Output Format:

Directly output your results in json format as follows:

```
{ {
  "Task Feasibility": "Yes or No", if no, the following entries should be empty.
  "Incorrect Ineffective Action": "Your incorrect ineffective action",
  "Explanation": "Explain why the action is incorrect and why it is a natural mistake for a
user not familiar with the app and certain operations."
}
```

Table 14: Prompt for adding reflection about the ineffective incorrect action of the second approach in offline SFT.

System Role

You are an expert in correcting the step-wise operation of a GUI agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user. The agent performs an incorrect action at a certain step, and it needs to realize the mistake and perform the correct action. Your task is to add some reflection content about the previous incorrect action to the correct action thought.

Definition of Action Thought

The action thought is the rationale behind the actual action taken at a certain step. The action thought should be a compact paragraph consisting of 3-4 sentences. It could include the following aspects if they are helpful and important for the action reasoning process:

- observation: a concise description of the current screenshot, focusing on the task related content
 - reflection: a simple analysis of whether the previous step's action is successful and as expected
 - progress analysis: a brief summary of the progress towards the overall goal before taking the action at the current step and a plan about the sub-tasks that still need to be done in order to complete the final goal
 - action rationale: a brief reasoning process to integrate the above information and provide a natural thinking process leading to the actual action from the perspective of the operator
- The action thought should be logically fluent, and it does not necessarily include all the above aspects. The action thought should *not* be clearly separated into the above aspects and does not need to mention the keywords like observation and reflection.

Information Provided

You will receive the following information:

1. Task Instruction: The overall task instruction for the GUI agent.
2. Incorrect Action: The previous incorrect action
3. Error Action Analysis: An analysis from an expert explaining why the previous action is incorrect and why it might happen.
4. Correct Action Thought: The actual correct action thought leading to the correct action
5. Screenshot: The screenshot of the current step after executing the incorrect action

Your job is to analyze the provided information and modify the correct action thought by adding some observation and reflection content realizing and acknowledging the incorrect action narrated in the first person as if you have performed the previous incorrect action.

Important Notes:

The previous incorrect action has no effect on the screenshot; that is, the screenshots before and after the incorrect action are the same.

Your modified action thought should still be consistent with the provided correct action thought.

!!! Note that the correct action thought does not include the previous incorrect action, so the 'previous step' in the correct action thought actually corresponds to the step before the previous incorrect action. So you have to ****remove**** the part describing the success of the previous step or description about the progress in the original correction action thought! For example, you should remove parts like 'I have successfully xxx' or 'The previous step has successfully xxx'.

!!! In the first sentence of your thought, you should directly mention that the previous step is unsuccessful or incorrect by observation, and then do a reflection explaining why you made that mistake and the fact learned from this failure.

Input:

Task Instruction: {task_instruction}

Incorrect Action: {incorrect_action}

Error Action Analysis: {error_analysis}

Correct Action Thought: {correct_action_thought}

Output Format:

Directly output your results in json format as follows:

```
{{  
  "Updated Action Thought": "The updated action thought with reflection added"  
}}
```

Table 15: Prompt for the MLLM-based verifier.

System Role

You are an expert in evaluating the performance of a GUI operation agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user. Your goal is to evaluate whether the agent successfully completed the task or not.

Information Provided

As an evaluator, you will receive the following information:

1. Task Instruction: The original task instruction for the GUI agent.
2. Task Guidance: An overall description about how to ****correctly**** complete this task as a reference.
3. Correct Answer: The ground truth answer for this task. Empty if the answer is not available or the task does not require an answer.
4. Operation History: A list of actions (in the form of images and corresponding actions) taken by the agent to execute the Task Instruction.

5. Agent Answer: The answer provided by the agent for the task. Empty if there is no answer action.
6. Final Status: The final task status is indicated by the agent. The status is either Task Complete or Task Impossible.

Your job is to analyze the provided information to determine whether the agent successfully completed the task, based on the alignment between the Task Instruction, the actions performed (shown in screenshots and operation history), and the expected outcome.

Effective Action Space

The actions that the agent may take to complete the task are as follows:

Click[[x, y]]: Click the location [x, y] on the current screen, marked with a red circle in the screenshot.

Long Press[[x, y]]: Click the location [x, y] on the current screen, marked with a red circle in the screenshot.

Scroll[[x1, y1, x2, y2]]: Scroll from position [x1, y1] to [x2, y2], as shown by the arrows in the screenshot, to access information that is not currently visible.

Type[text]: Type the 'text' into the input field.

Memorize[text]: Store some 'text' into an intermediate memory for future reference.

Answer[text]: The agent provides the 'text' as the answer.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something

Task Complete/Task Impossible: The agent indicates the task is completed or impossible.

Task Evaluation Criteria:

SUCCESS: The agent successfully performs all the necessary actions to meet the Task Instruction.

NOT SUCCESS: The agent failed to perform at least one necessary action, or the task could not be completed correctly, based on the screenshots and operation history.

Useful Guidelines

To make the evaluation process more accurate, please follow these guidelines:

1. Analyze the Task Instruction: You should first analyze what critical milestones have to be completed and what necessary outcomes are expected for this task.
2. Analyze the agent's operations, provide a definitive verdict on whether the task has been successfully completed together your reasoning process.
3. Provide your final answer: 'SUCCESS' or 'NOT SUCCESS'

Important Notes

When the task instruction asks a certain question or seeks certain information, the agent has to provide the ****correct**** answer before completing the task, otherwise, the task should be NOT SUCCESS.

When the agent indicates the task is impossible, you should judge whether this task is indeed impossible to complete (e.g. due to network issues). You should output SUCCESS only when both you and the agent indicate the task is impossible.

You should carefully examine the screenshots to double check whether the operations taken by the agents achieve the desired and expected outcome. Any minor input or format issues should be judged as NOT SUCCESS.

Note that the agent might make some wrong attempts during the process, and it should be judged as SUCCESS as long as the agent has successfully completed the task at the end.

Input:

Task Instruction: {task_instruction}

Task Guidance: {task_guidance}

Correct Answer: {correct_answer}

Agent Answer: {agent_answer}

Final Status: {final_status}

Output Format:

Directly output your results in json format as follows:

```
{ {  
  "Task Analysis": "An analysis of the task",  
  "Reasoning": "Reason about whether the agent has successfully completed the task",  
  "Final Result": "SUCCESS" or "NOT SUCCESS"  
}
```

Table 16: Prompt for checking the step-wise correctness of successful trajectories.

System Role

You are an expert in evaluating the step-wise operation correctness of a GUI agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user. Your goal is to evaluate whether the current action taken by the agent is correct in terms of completing the overall task instruction.

Information Provided

As an evaluator, you will receive the following information:

1. Task Instruction: The overall task instruction for the GUI agent.
2. Task Guidance: An overall description of how to correctly complete this task as a reference.
3. GT Answer: The ground truth answer for this task. Empty if the answer is not available or the task does not require an answer.
4. Stored Memory: Information the agent has stored in the intermediate memory from previous steps. Empty if none.
5. Task Progress: A summary describing the current progress of the overall task before taking the action at the current step. The progress is empty if it is the first step.
6. Current Action: The action taken by the agent at the current step.
7. Screenshots: The screenshot at the current step and the screenshot at the next step after the current action is executed. The click point is highlighted using a red dot in the current screenshot and the scroll positions are visualized using a red arrow in the current screenshot.

Your job is to analyze the provided information to determine whether the current action taken by the agent is correct for achieving the overall task goal.

Valid Action Space

The actions that the agent may take to complete the task are as follows: Click: Click on the current screen, the click point is marked with a red circle in the screenshot.

Long Press: Long press on the current screen, the press point is marked with a red circle in the screenshot.

Scroll: Scroll on the screen, the touch point and lift point of the scroll are marked with a red arrow in the screenshot.

Type[text]: Type the 'text' into the input field.

Memorize[text]: Store some 'text' into an intermediate memory for future reference

Answer[text]: The agent provides the 'text' as the answer.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something

Task Complete/Task Impossible: The agent indicates the task is completed or impossible.

Task Evaluation Criteria:

CORRECT: The current action taken by the agent is reasonable without skipping any necessary steps and makes correct progress towards completing the overall task instruction.

INCORRECT: The current action taken by the agent is clearly wrong or unnecessary for completing the task, or deviates from the correct way. You should consider whether the outcome of the current action (from the second screenshot) is as expected or not when making the judgment.

Useful Guidelines

To make the evaluation process more accurate, please follow these guidelines:

1. Based on the provided information, analyze the agent's action, provide a definitive verdict on whether the current action is correct, together with your reasoning process.
2. Provide your conclusion: 'CORRECT' or 'INCORRECT'
3. Based on the current progress and the screenshot for the next step, update the task progress summarizing the overall progress after taking this current action. One or two sentences.

Note that some sub-tasks or expected outcomes might need multiple actions (e.g. typing something needs to click the text input field and then type), and you only need to assess the correctness of the current action.

Note that the 'Current Action' represents the purpose of the agent, which might be inconsistent with the actuation of the action due to its unfamiliarity with certain operations or elements when the action is to click/long press/scroll. In such cases, the 'Current Action' itself may seem correct, but the grounded action is wrong. For example, the agent's action (purpose) is to click App A, but it actually clicks App B as it does not know which icon is App A. So you should also make the judgment based on the actual action and its outcome. !!! Note that an action should be considered correct if it attempts to recover from a previous mistake or redirect the process back toward the final goal, even if earlier steps were incorrect or unnecessary.

Input:

Task Instruction: {task_instruction}

Task Guidance: {task_guidance}

GT Answer: {gt_answer}

Stored Memory: {memory}

Task Progress: {prev_progress}

Current Action: {action}

Output Format:

Directly output your results in json format as follows:

```
{{
  "Reasoning": "Reason about whether the current action is correct",
  "Conclusion": "CORRECT" or "INCORRECT",
  "Updated Progress": "The updated task progress summary after taking this action."
}}
```

Table 17: Prompt for identifying the first error step of the unsuccessful trajectories.

System Role

You are an expert in evaluating the step-wise operation correctness of a GUI agent. The agent's task is to help human users operate an Android phone by completing specific instructions provided by the user. Your goal is to evaluate whether the current action taken by the agent is correct in terms of completing the overall task instruction.

Information Provided

As an evaluator, you will receive the following information:

1. Task Instruction: The overall task instruction for the GUI agent.
2. Task Guidance: An overall description of how to correctly complete this task as a reference.

- 3: GT Answer: The ground truth answer for this task. Empty if the answer is not available or the task does not require an answer.
- 4: Stored Memory: Information the agent has stored in the intermediate memory from previous steps. Empty if none.
- 5. Task Progress: A summary describing the current progress of the overall task before taking the action at the current step. The progress is empty if it is the first step.
- 6. Current Action: The action taken by the agent at the current step.
- 7. Screenshots: The screenshot at the current step and the screenshot at the next step after the current action is executed. The click point is highlighted using a red dot in the current screenshot and the scroll positions are visualized using a red arrow in the current screenshot.

Your job is to analyze the provided information to determine whether the current action taken by the agent is correct for achieving the overall task goal.

Valid Action Space

The actions that the agent may take to complete the task are as follows:

Click: Click on the current screen, the click point is marked with a red circle in the screenshot.

Long Press: Long press on the current screen, the press point is marked with a red circle in the screenshot.

Scroll: Scroll on the screen, the touch point and lift point of the scroll are marked with a red arrow in the screenshot.

Type[text]: Type the 'text' into the input field.

Memorize[text]: Store some 'text' into an intermediate memory for future reference.

Answer[text]: The agent provides the 'text' as the answer.

Press Home: Return to the home page.

Press Back: Go back to the previous page.

Press Enter: Click the enter button.

Wait: Wait for the device to respond or load something.

Task Complete/Task Impossible: The agent indicates the task is completed or impossible.

Task Evaluation Criteria:

CORRECT: The current action taken by the agent is reasonable without skipping any necessary steps and makes correct progress towards completing the overall task instruction.

INCORRECT: The current action taken by the agent is clearly wrong or unnecessary for completing the task, or deviates from the correct way. You should consider whether the outcome of the current action (from the second screenshot) is as expected or not when making a judgment.

Useful Guidelines

To make the evaluation process more accurate, please follow these guidelines:

1. Based on the provided information, analyze the agent's action, provide a definitive verdict on whether the current action is correct, together with your reasoning process.
2. Provide your conclusion: 'CORRECT' or 'INCORRECT'.
3. Based on the current progress and the screenshot for the next step, update the task progress summarizing the overall progress after taking this current action. One or two sentences.

Note that some sub-tasks or expected outcomes might need multiple actions (e.g. typing something needs to click the text input field and then type), and you only need to assess the correctness of the current action.

Note that if the task requires question answering (GT Answer is not empty), the agent must provide the answer before marking the task as complete.

Input:

Task Instruction: {task_instruction}

Task Guidance: {task_guidance}

GT Answer: {gt_answer}

Stored Memory: {memory}
Task Progress: {prev_progress}
Current Action: {action}

Output Format:

Directly output your results in json format as follows:

```
{
  "Reasoning": "Reason about whether the current action is correct",
  "Conclusion": "CORRECT" or "INCORRECT",
  "Updated Progress": "The updated task progress summary after taking this action."
}
```

Table 18: Task examples of our online learning environment.

Apps	Difficulty Level	Number	Example Task
Simple Calendar Pro	Level-1	13	Change the view to {view} in Simple Calendar Pro.
Simple Calendar Pro	Level-2	7	Change the location of event {event1} to be the same as event {event2} in Simple Calendar Pro.
Contacts	Level-1	12	Delete the contact named {name} in the Contacts App.
Contacts	Level-2	9	Find {name} in the Contacts App, change the phone number to {new number} and email to {new email}
Dice	Level-1	10	In the Dice App, roll the dice {n times} times with the default setting.
Dice	Level-2	4	In the Dice App, set Dice to 1 and Sides to {n sides}, and roll the dice twice with this setting. Record the results and answer the sum of numbers.
FitBook	Level-1	11	Open the FitBook App, delete the entry with the highest calories in the diary tab.
FitBook	Level-2	8	Open the FitBook App, find and open the {entry} entry in the Diary tab, then change its date to {n days} days before the original date.
Fossify Clock	Level-1	15	Add a clock in {timezone} timezone in the Fossify Clock App.
Fossify Clock	Level-2	9	Start all timers and then pause them in the Fossify Clock App.
Fossify Messages	Level-1	8	Delete the conversation from {number} in the Fossify Messages App.
Fossify Messages	Level-2	7	Send the same message {n times} times to {number} in the Fossify Messages App with the following content: {message}

LibreOffice	Level-1	10	In the LibreOffice Viewer App, go to the {folder} folder in the sdk_gphone64_x86_64 storage area. How many docs are there? Answer with the number only.
LibreOffice	Level-2	4	Find the numbers logged in {docfile1} and {docfile2} in the sdk_gphone64_x86_64 storage area of the LibreOffice Viewer App. Answer the sum of the numbers.
Markor	Level-1	13	Find all markdown notes in Markor. Answer with their names separated with comma.
Markor	Level-2	6	Check the file sizes of the note {name1} and {name2} in Markor. Then answer the larger file size in Bytes (number only without units), for example: 500.
ProExpense	Level-1	11	What expenses are logged in the Pro Expense App? Answer the names separated by comma.
ProExpense	Level-2	9	Delete all expenses in the Pro Expense App that are higher than {amount}.
Broccoli	Level-1	13	Add the following recipe in the Broccoli APP: {recipe}.
Broccoli	Level-2	9	In the Broccoli APP, check the preparation times of recipes {title1} and {title2}, and answer the longer preparation time. Please directly answer the time in the following format: XX mins/XX hrs.
Chrome (WebShopping)	Level-1	9	Go to {website} using Chrome, search for {product}, what is the overall rating of the first search result? Answer with the number only.
Chrome (WebShopping)	Level-2	9	Search for the following items {product list} on {website} and add them to my cart. What is the total price of all the items in my cart?

References

- [1] AgentSea Team. Wave-UI Dataset. <https://huggingface.co/datasets/agentsea/wave-ui>, 2023.
- [2] AgentSea Team. Wave-UI-25K Dataset. <https://huggingface.co/datasets/agentsea/wave-ui-25k>, 2024.
- [3] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: A vision-language model for ui and infographics understanding. In *IJCAI*, 2024.
- [4] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. In *IJCAI*, 2021.
- [5] Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *arXiv preprint arXiv:2406.11896*, 2024.
- [6] Hao Bai, Yifei Zhou, Li Erran Li, Sergey Levine, and Aviral Kumar. Digi-q: Learning q-value functions for training device-control agents. *arXiv preprint arXiv:2502.15760*, 2025.
- [7] Andrea Burns, Kate Saenko, and Bryan A. Plummer. Tell me what’s next: Textual foresight for generic ui representations. In *ACL Findings*, 2024.
- [8] 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.
- [9] Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*, 2024.
- [10] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *ACL*, 2024.
- [11] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.
- [12] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *ICLR*, 2025.
- [13] 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.
- [14] Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazhen Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *CVPR*, 2024.
- [15] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.
- [16] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. In *NeurIPS*, 2024.
- [17] 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.
- [18] Yang Li, Gang Li, Luheng He, Jingjie Zheng, Hong Li, and Zhiwei Guan. Widget captioning: Generating natural language description for mobile user interface elements. In *EMNLP*, 2020.
- [19] Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchun Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. Infiguiagent: A multimodal generalist gui agent with native reasoning and reflection. *arXiv preprint arXiv:2501.04575*, 2025.

- [20] 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.
- [21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [22] 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.
- [23] Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent, 2024.
- [24] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.
- [25] 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.
- [26] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*, 2023.
- [27] Christopher Rawles, Sarah Clinckemaulle, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyi 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. In *NeurIPS*, 2023.
- [29] Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvareja, Andrew Hojel, Andrew Ma, et al. Rethinking reflection in pre-training. *arXiv preprint arXiv:2504.04022*, 2025.
- [30] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *NeurIPS*, 2020.
- [31] Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Jindong Chen, Abhanshu Sharma, and James Stout. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663*, 2022.
- [32] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [33] 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. In *NeurIPS*, 2024.
- [34] Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye HAO, Jun Wang, and Kun Shao. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agent. In *ICLR*, 2025.
- [35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022.
- [36] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *ICLR*, 2025.
- [37] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024.

- 283 [38] Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li.
284 Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024.
- 285 [39] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik
286 Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *NeurIPS*,
287 2023.
- 288 [40] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is
289 more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- 290 [41] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does
291 reinforcement learning really incentivize reasoning capacity in llms beyond the base model?
292 *arXiv preprint arXiv:2504.13837*, 2025.
- 293 [42] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang
294 Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.
- 295 [43] Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and
296 Duyu Tang. Android in the zoo: Chain-of-action-thought for GUI agents. In *EMNLP Findings*,
297 2024.
- 298 [44] Junlei Zhang, Zichen Ding, Chang Ma, Zijie Chen, Qiushi Sun, Zhenzhong Lan, and Junxian
299 He. Breaking the data barrier – building gui agents through task generalization. *arXiv preprint*
300 *arXiv:2504.10127*, 2025.