INREACT: An Inspire-Then-Reinforce Training Framework For Multimodal GUI Agent

Yuanlei Wang[1]*, Liuzhou Zhang[2]*, Haohao Luo[1]*, Ying Shen[1,3]†
[1]Sun Yat-sen University, [2]Central China Normal University,
[3]Guangdong Provincial Key Laboratory of Fire Science and Intelligent Emergency Technology
{wangylei7, luohh5}@mail2.sysu.edu.cn, zlz212@mail.ccnu.edu.cn,
sheny76@mail.sysu.edu.cn

Abstract

Graphical User Interface (GUI) interaction, which aims to develop an intelligent GUI agent that executes user instructions to perform tasks such as installing applications by controlling digital devices, has gained significant attention due to its practical value. Although current advanced multimodal large language models (LLMs) provide GUI agents with robust perception and reasoning capabilities, they often struggle with the precise localization of small elements. To tackle this problem, we propose INREACT, a multimodal GUI agent framework that unifies observing, thinking, and acting for precise and interpretable decision-making. It is trained via a two-stage process: curriculum learning to progressively build perception, grounding, and reasoning abilities, followed by reinforcement learning to refine pixel-level grounding with an outcome-based reward. We introduce a rule-based reward function that jointly optimizes action-type selection and pixel-level localization accuracy. Experimental results on multiple datasets demonstrate the superiority of INREACT in both grounding and navigation tasks.

1 Introduction

Graphical User Interface (GUI) interaction refers to a class of sequential decision-making tasks in which agents interpret user instructions and manipulate digital interfaces, including PCs and mobile devices, to accomplish specific goals. These interactions typically involve executing low-level operations like *Click*, *Type*, or *Scroll* in the correct order to complete complex tasks, such as *searching for information*. A capable GUI agent must possess two core abilities: visual observation and reasoning. Observation is particularly critical in GUI environments like PCs and mobile devices, where visual

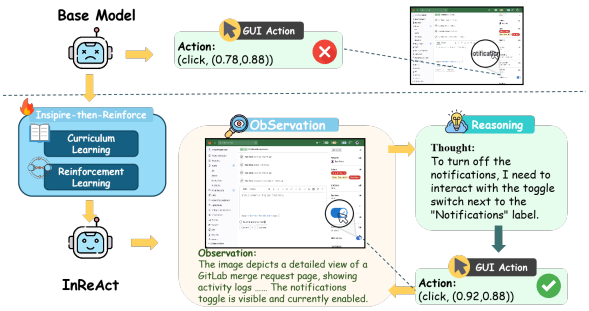


Figure 1: Overview of the INREACT interaction

components are often densely packed and visually similar, making precise localization challenging. Meanwhile, reasoning is essential for complex, goal-directed interactions that involve sequential decision-making. Unlike single-step tasks, GUI interaction requires agents to consider both interface states and past actions. With the emergence of powerful multimodal large language models (MLLMs) (Wang et al., 2024c; Bai et al., 2025; Chen et al., 2023; Luo et al., 2024), recent approaches have enabled screenshot-based GUI control without relying on structured metadata such as HTML or accessibility trees (Zhang et al., 2025; Wang et al., 2024b). However, despite these advances, existing GUI agents still struggle with accurate grounding and coherent reasoning in complex, real-world scenarios.

To address the challenges of imprecise grounding and weak multi-step reasoning, recent efforts have focused on training MLLM-based GUI agents via supervised fine-tuning (SFT) or reinforcement learning (RL). In SFT, models are trained on paired GUI-action datasets to align visual observations with user instructions. Some approaches (Hong et al., 2024; Cheng et al., 2024; Lin et al., 2024; Li et al., 2025a) enhance SFT by incorporating instruction tuning, vision-language pretraining, or synthetic data generation to boost coverage. However, these strategies commonly adopt a flat optimization objective, lacking curriculum or structural

*Equal Contribution
†Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 9148–9160
November 4-9, 2025

guidance, which limits the model's ability to progressively acquire complex capabilities. In contrast, reinforcement learning introduces outcome-driven optimization through environment feedback. Some methods (Lai et al., 2024; Liu et al., 2024) rely on sparse success signals to guide multi-step task completion. Despite this, many RL strategies remain coarse and fail to capture pixel-level grounding precision. Moreover, most RL pipelines lack strong initialization or structured learning stages, often resulting in unstable policies and inconsistent behaviors.

In this work, we propose INREACT, a structured training framework for GUI agents that combines curriculum learning with reinforcement learning. The curriculum learning stage is designed to systematically build core agent capabilities—including perception, grounding, and reasoning through a staged learning process of increasing complexity. Furthermore, we introduce a reinforcement learning strategy with rule-based reward functions that jointly considers action-type correctness and distance-aware spatial accuracy, enabling more precise and reliable interactions in complex GUI environments.

Our contributions are summarized as follows:

- We propose a novel multimodal GUI agent framework, called INREACT, which synergizes *observing*, *thinking*, and *acting* to make more interpretable and goal-directed decisions when handling complex GUI navigation tasks.
- INREACT is trained through a two-stage training framework, including a curriculum-based supervised fine-tuning stage to progressively acquire perception, grounding, and reasoning capabilities and a reinforcement learning stage that reinforces grounding precision with an outcome-based reward to encourage accurate and consistent action predictions.
- Experimental results on four datasets in grounding and navigation tasks show that INREACT outperforms existing methods and exhibits promising generalization. Our code will be released via `https://github.com/wangyuanlei30/InReAct`.

2 Related Work

2.1 GUI Agents

GUI agents allow models to interact with software interfaces via visual inputs and simulated actions. Early works like CogAgent (Hong et al., 2024) introduced MLLMs for GUI tasks. Some methods leverage general-purpose models with prompt engineering (e.g., AppAgent (Zhang et al., 2025), Mobile-Agent (Wang et al., 2024b)), while others fine-tune open-source models on GUI-specific data (e.g., UGround (Gou et al., 2025), ShowUI (Lin et al., 2024)). Recent work like UI-TARS (Qin et al., 2025) and AGUVIS (Xu et al., 2024) combines GUI pretraining with reasoning fine-tuning. Most rely on single-stage supervised fine-tuning, which requires large-scale data and often struggles with generalization, highlighting the need for more structured learning approaches.

2.2 Curriculum Learning

Curriculum learning (Wang et al., 2022) is a training strategy where models are exposed to tasks in increasing order of difficulty, mimicking human learning. It has shown strong effectiveness across NLP and vision tasks by improving learning stability and generalization. Recent works have applied this idea to large-scale models: AutoGLM (Liu et al., 2024) and AutoWebGLM (Lai et al., 2024), for example, design task curricula to enhance model robustness in web navigation tasks. In the context of instruction tuning and agent learning, curriculum learning helps models acquire basic capabilities before handling complex reasoning or planning, offering a structured alternative to flat fine-tuning.

2.3 Rule-Based Reinforcement Learning

Rule-based reinforcement learning has emerged as a scalable alternative to traditional supervised fine-tuning by optimizing models with predefined and verifiable reward functions. Early works such as DeepSeek-R1 (Guo et al., 2025) and o1 (Jaech et al., 2024) applied RFT to language tasks, including mathematical reasoning (Shao et al., 2024) and code generation, where correctness can be evaluated with rule-based signals. More recent studies (Shen et al., 2025; Liu et al., 2025c; Chen et al., 2025; Wang et al., 2024a) extend this paradigm to multimodal models by designing task-specific rewards—such as Intersection-over-Union (IoU) for image grounding and symbolic consistency checks for classification. Building on this line of research, works like UI-R1 (Lu et al., 2025) and GUI-R1 (Luo et al., 2025) have further explored applying rule-based RL to GUI agents, typically leveraging binary hit-or-miss rewards for GUI grounding.

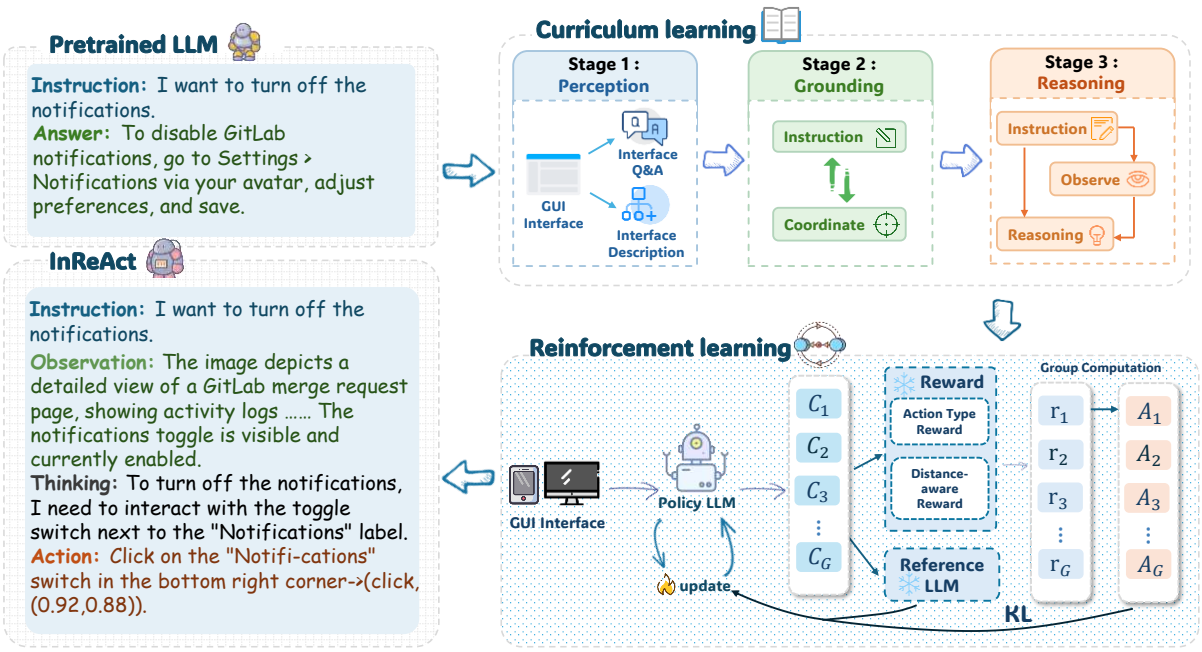


Figure 2: The overall framework of INREACT

3 Method

Given a GUI screenshot S, a user instruction G and the previous interaction history $A_{<t}$ at each time step t, the goal of the agent is to generate a sequence of executable actions $A = \{a_1, a_2, \ldots, a_T\}$, where action a_t at time step t is determined based on G, observation of the current screenshot environment O_t, and previous actions sequence $A_{<t} = [a_1, a_2, ..., a_{t-1}]$.

- **State Space**. At each time step t, the environment state is represented by a GUI screenshot s_t, which captures the current visual interface after executing the previous action. The agent's observation O_t includes the screenshot S_t, the user instruction G, and the action history $A_{<t} = \{a_1, ..., a_{t-1}\}$. This combination provides both static semantic goals and dynamic context necessary for decision-making.
- **Action Space**. We define an action as a structured function call in the form $a_t = (\text{type}_t, \text{args}_t)$, where `type` specifies the operation (e.g., `click`, `scroll`, `input`) and `args` provides necessary arguments such as coordinates, text, or direction. The environment executes actions parsed from the agent's JSON-style outputs. A full specification of supported actions is provided in Table 1.

3.1 GUI Agent Workflow

To guide the agent to continually interact with the environment and complete multi-step complex tasks. Following ReAct (Yao et al., 2023), we construct our agent framework through synergizing *observing*, *thinking* and *acting* in GUI controlling process for agent.

Action Type	Parameters
click,tap,hover,select	(x, y)
swipe,select_text	(x_1, y_1, x_2, y_2)
scroll	`direction`
input	`text`
home,back,enter	None

Table 1: Action types and their parameters

Observation Phase. Based on a visual screenshot S_t of the GUI, an user instruction G and the previous interaction history $A_{<t}$ at each time step t, the agent first generates a natural language observation d_t describing key interface elements and their relevance to the task instead of directly generating action by:

$$d_t = \mathcal{F}(O_t) = \mathcal{F}(S_t, G, A_{<t}) \quad (1)$$

Thinking Phase. In the thinking phase, the generated observation d_t is appended to the original input $O_t = \{S_t, G, A_{<t}\}$ to construct the further input. Then, we feed the updated input to the agent for generating internal reasoning to produce a task-oriented thinking h_t by:

$$h_t = \mathcal{F}(O_t, d_t) \quad (2)$$

Acting Phase. Similarly, the input in the final action phase is constructed by concatenating the generated thinking h_t with the previous input. Subsequently, we feed the modified input to the agent to generate executable actions by

$$a_t = \mathcal{F}(O_t, d_t, h_t) = (\text{type}_t, \text{args}_t) \quad (3)$$

The action is then executed in the environment, resulting in a transition to the next state S_{t+1}.

3.2 Inspire-Then-Reinforce Training

To progressively enhance the reasoning and control capabilities of GUI agents, we propose an inspire-then-reinforce two-stage training paradigm, as illustrated in Figure 2.

In Stage 1, we introduce a curriculum learning strategy that progressively builds the model's perception, grounding, and reasoning capabilities through supervised fine-tuning. By organizing training tasks from simple to complex, the model first learns to understand GUI layouts, then localize targets, and ultimately reason over multi-step goals.

In Stage 2, we perform reinforcement learning to improve action accuracy and grounding robustness. Using a GRPO-based optimization framework, we incorporate a task-specific reward function that evaluates both action-type correctness and spatial precision, enabling the model to generate reliable actions in dense and high-resolution GUI environments.

3.2.1 Stage 1: Curriculum Learning

Drawing inspiration from human learning, we organize agent training into three progressive stages: perception, grounding, and reasoning. Each stage targets a specific skill and increases in task complexity, forming a curriculum learning framework that guides the model from basic comprehension to advanced planning.

Specifically, we adopt supervised fine-tuning and have the same training objective that minimizes the sum of negative log-likelihood loss averaged over tokens in each step:

$$\mathcal{L}(y, \widehat{y}) = -\frac{1}{L}\sum_{l=1}^{L} \widehat{y}_l \log\left(\frac{\exp(y_l)}{\sum_i^l \exp(y_i)}\right), \quad (4)$$

where L is the max length of output sequence, $\widehat{y}_l$ and y_l denote the l-th token in the groundtruth sequence $\widehat{y}$ and generation sequence y, respectively.

Learning for Perception The first stage of the curriculum aims to build the model's visual and semantic understanding of GUI layouts. This foundational perception equips the agent with spatial awareness and interface comprehension, essential for later grounding and reasoning. Specifically, we decompose it into two perception-oriented tasks:

i. GUI-based Question Answering. Given a screenshot s and question q, the model generates an answer a: $\mathcal{M}_{\text{qa}}(s, q) \rightarrow a$, enhancing spatial attention and element recognition.

ii. Interface Description Generation. Given s, the model produces a structured description d: $\mathcal{M}_{\text{desc}}(s) \rightarrow d$, fostering global layout and semantic understanding. The training loss for this stage is computed as:

$$\mathcal{L}_{\text{perception}} = \mathcal{L}(a, \widehat{a}) + \mathcal{L}(d, \widehat{d}), \quad (5)$$

where $\widehat{s}$ and $\widehat{d}$ denote the ground-truth answers and interface descriptions, respectively.

Learning for Grounding The second stage of the curriculum focuses on grounding user instructions to visual actions, a core capability for operating in GUI environments. The model must understand what the user wants and accurately identify the corresponding interface region for execution. We similarly decompose it into two complementary grounding tasks:

i. Instruction-to-Coordinate. Given a GUI screenshot s and instruction g, the model predicts the target coordinate c: $\mathcal{M}_{\text{inst2coord}}(s, g) \rightarrow c$. This task aligns user intent with precise UI locations.

ii. Coordinate-to-Description. Given s and a coordinate c, the model generates a description d of the corresponding UI element: $\mathcal{M}_{\text{coord2desc}}(s, c) \rightarrow d$, reinforcing spatial-semantic understanding. The loss function for this stage is defined as:

$$\mathcal{L}_{\text{grounding}} = \mathcal{L}(c, \widehat{c}) + \mathcal{L}(d, \widehat{d}), \quad (6)$$

where $\widehat{c}$ and $\widehat{d}$ denote the ground-truth coordinate and description, respectively.

Learning for Reasoning The third stage focuses on high-level reasoning and multi-step decision-making, where the model explicitly follows an *observe-think-act* process instead of making one-step predictions. We introduce the following two components to support structured reasoning:

i. Observation Chain. Given a screenshot x and instruction g, the model generates an observation o to identify relevant UI regions: $\mathcal{M}_{\text{obs}}(x, g) \rightarrow o$.

ii. Reasoning Chain. Conditioned on o and g, the model produces a reasoning trace r to guide decision-making: $\mathcal{M}_{\text{reason}}(o, g) \rightarrow r$.

We define the training loss of this stage as follows:

$$\mathcal{L}_{\text{reasoning}} = \mathcal{L}(o, \widehat{o}) + \mathcal{L}(r, \widehat{r}), \tag{7}$$

where $\widehat{o}$ and $\widehat{r}$ denote the ground-truth observation and reasoning trace.

3.2.2 Stage 2: Reinforcement Learning

Although supervised fine-tuning provides a strong initialization, models often suffer from fine-grained prediction errors, especially in dense or high-resolution GUIs where small pixel-level deviations can lead to task failure. To address this, we adopt GRPO Reinforcement fine-tuning.

A critical component of our approach is the design of the reward function. While prior works like UI-R1 (Lu et al., 2025) and GUI-R1 (Luo et al., 2025) also leverage rule-based reinforcement learning for GUI agents, their reward mechanisms for localization typically rely on a coarse, binary signal (i.e., whether the predicted point falls within the target's bounding box). Such a reward, while useful for confirming task success, provides limited gradient for improving fine-grained precision. To overcome this, we introduce a task-specific reward function that better captures the dual objectives of action correctness and spatial precision, directly encouraging the model to enhance its pixel-level accuracy.

Reward Modeling The reward function serves as the primary training signal in reinforcement learning, directly guiding the model's optimization trajectory. To support fine-grained policy improvement in GUI environments, we design a rule-based reward function tailored to the unique demands of screen-based interaction. Unlike traditional RL settings that rely on binary task success or coarse region-level feedback, GUI tasks require both accurate action selection and high-precision localization of target elements. Specifically, our reward function comprises two components: (1) an action-type reward(2) a distance-aware coordinate reward. This composite design provides the model with dense, informative feedback signals essential for precise and interpretable GUI control, which is defined as:

$$R = R_{\text{T}} + R_{\text{D}} \tag{8}$$

Regarding the action type reward component R_{T}, it provides a binary reward signal that evaluates whether the predicted action type T aligns with the ground truth label T'. This mechanism offers a direct and effective evaluation framework for action-type prediction tasks. The mathematical formulation of this component is presented below:

$$R_{\text{T}} = \begin{cases} 1, & \text{if } T' = T \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Additionally, the distance-aware reward R_{D} measures spatial accuracy by combining two components: (1) a binary reward for whether the predicted point lies within the target bounding box, and (2) a precision reward based on the normalized distance to the box center. This encourages both correct region prediction and fine-grained localization:

$$R_D = R_{acc} + R_{dist} \tag{10}$$

i. Accuracy Reward. We assign a base reward calculating whether the predicted point (x', y') falls inside the ground-truth bounding box $B = [(x_1, y_1), (x_2, y_2)]$:

$$R_{\text{acc}} = \begin{cases} 0.5, & \text{if } (x', y') \in B \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

ii. Precision Reward. If the point is within the box, we compute the normalized Euclidean distance $\hat{d}$ from the predicted coordinate (x', y') to the center of the target box (x_c, y_c):

$$(x_c, y_c) = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right) \tag{12}$$

$$\hat{d} = \frac{\sqrt{(x' - x_c)^2 + (y' - y_c)^2}}{\sqrt{(x_2 - x_1)(y_2 - y_1)}} \tag{13}$$

We then define the precision reward based on discrete distance intervals:

$$R_{\text{dist}} = \begin{cases} 0.5, & \hat{d} \leq 0.1 \\ 0.3, & 0.1 < \hat{d} \leq 0.3 \\ 0.2, & 0.3 < \hat{d} \leq 0.5 \\ 0.0, & \hat{d} > 0.5 \end{cases} \tag{14}$$

To provide a clear and stepped gradient for precision improvement, we empirically determine the thresholds and reward values for our reward formulation. This design provides a continuous and informative learning signal that encourages the model to both select the correct action type and localize it precisely, promoting stable convergence and robust behavior in dense GUI layouts.

Method	Development			Creative			CAD			Scientific			Office			OS			Overall		
	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg	Text	Icon	Avg
GPT-4o (OpenAI et al., 2024)	1.3	0.0	0.7	1.0	0.0	0.6	2.0	0.0	1.5	2.1	0.0	1.2	1.1	0.0	0.9	0.0	0.0	0.0	1.3	0.0	0.8
ShowUI-2B (Lin et al., 2024)	16.9	1.4	9.4	9.1	0.0	5.3	2.5	0.0	1.9	13.2	7.3	10.6	15.3	7.5	13.5	10.3	2.2	6.6	10.8	2.6	7.7
AriaUI-3.9B (Yang et al., 2024)	16.2	0.0	8.4	23.7	2.1	14.7	7.6	1.6	6.1	27.1	6.4	18.1	20.3	1.9	16.1	4.7	0.0	2.6	17.1	2.0	11.3
OSAtlas-4B (Wu et al., 2024)	7.1	0.0	3.7	3.0	1.4	2.3	2.0	0.0	1.5	9.0	5.5	7.5	5.1	3.8	4.8	5.6	0.0	3.1	5.0	1.7	3.7
QwenVL-7B (Bai et al., 2023)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1
Qwen2-VL-7B (Wang et al., 2024c)	2.6	0.0	1.3	1.5	0.0	0.9	0.5	0.0	0.4	6.3	0.0	3.5	3.4	1.9	3.0	0.9	0.0	0.5	2.5	0.2	1.6
SeeClick-7B (Cheng et al., 2024)	0.6	0.0	0.3	1.0	0.0	0.6	2.5	0.0	1.9	3.5	0.0	2.0	1.1	0.0	0.9	2.8	0.0	1.5	1.8	0.0	1.1
CogAgent-18B (Hong et al., 2024)	14.9	0.7	8.0	9.6	0.0	5.6	7.1	3.1	6.1	22.2	1.8	13.4	13.0	0.0	10.0	5.6	0.0	3.1	12.0	0.8	7.7
UGround-7B (Gou et al., 2025)	26.6	**2.1**	14.7	27.3	2.8	17.0	**14.2**	1.6	11.1	31.9	2.7	19.3	31.6	**11.3**	27.0	17.8	0.0	9.7	25.0	2.8	16.5
OSAtlas-7B (Wu et al., 2024)	33.1	1.4	17.7	28.8	**2.8**	17.9	12.2	**4.7**	10.3	37.5	7.3	24.4	33.9	5.7	27.4	**27.1**	**4.5**	16.8	28.1	**4.0**	18.9
InReAct	**35.7**	1.4	**19.0**	**33.3**	2.1	**20.2**	13.2	1.6	**10.7**	**41.7**	**10.0**	**27.9**	**42.4**	3.8	**33.5**	23.4	1.1	**17.3**	**31.4**	3.3	**20.7**

Table 2: GUI grounding results on **ScreenSpot-Pro** dataset.

3.3 Inference Procedure

During inference, our agent adopts an interlaced observe-think-action workflow that sequentially executes tasks by observing the interface, thinking through the instruction, and predicting the next action. Given the input $O_t = (S, G, H)$, where S is the GUI screenshot, G is the task instruction, and $A_{<t} = \{a_1, ..., a_{t-1}\}$ denotes the history of executed actions, the model first generates an observation summary $\hat{o}_t$, then infers a reasoning trace $\hat{r}_t$ conditioned on $\hat{o}_t$ and G, and finally predicts the next action $a_t = (T', x', y')$ based on both. This process is repeated until task completion or a maximum number of steps is reached, enabling goal-driven, interpretable decision-making in complex GUI environments.

4 Experiment

4.1 Experiment Setup

Datasets We evaluate our model on two tasks: GUI grounding and navigation.

For Grounding, we use ScreenSpot (Cheng et al., 2024) and ScreenSpot-Pro (Li et al., 2025b). ScreenSpot contains 1,272 samples across mobile, desktop, and web platforms, focusing on common interface elements. ScreenSpot-Pro includes 23 professional applications with high-resolution and complex layouts, offering a more challenging evaluation.

For Navigation, we use: (i) Mind2Web (Deng et al., 2023), a web-based dataset for evaluating generalist agents that perform complex tasks via language instructions. It includes 7,775 training actions and three test splits (task, website, domain), with aligned HTML and screenshots. (ii) AndroidControl (Li et al., 2024), a mobile dataset testing multi-step task execution in realistic Android environments, emphasizing long-term planning and state tracking.

See datasets details in appendix A.

Method	Mobile		Desktop		Web		Average
	Text	Icon	Text	Icon	Text	Icon	
GPT-4 (OpenAI, 2023)	22.6	24.5	20.2	11.8	9.2	8.8	16.7
GPT-4o (OpenAI et al., 2024)	20.2	24.9	21.1	23.6	12.2	7.8	18.1
Gemini-1.5-pro (Team, 2024)	76.2	54.1	65.5	39.2	52.2	32.0	53.2
Qwen2-VL-2B (Wang et al., 2024c)	24.2	10.0	1.4	9.3	8.7	2.41	9.3
Qwen2-VL-7B (Wang et al., 2024c)	61.3	39.3	52.0	45.0	33.0	21.8	42.9
CogAgent-18B (Hong et al., 2024)	67.0	24.0	74.2	20.0	70.4	28.6	47.4
Seeclick-7B (Cheng et al., 2024)	78.0	52.0	72.2	30.0	55.7	32.5	53.4
UGround-7B (Gou et al., 2025)	82.8	60.3	82.5	63.6	80.4	70.4	73.3
ShowUI-2B (Lin et al., 2024)	**92.3**	**75.5**	76.3	61.1	81.7	63.6	75.1
InfiGUIAgent-2B (Liu et al., 2025b)	88.6	74.7	**85.6**	65.0	79.1	64.6	76.3
InReAct	86.1	71.6	84.5	**65.7**	**82.2**	**70.4**	**77.8**

Table 3: GUI grounding results on **ScreenSpot** dataset.

Evaluation Metrics We use the metrics proposed in the corresponding datasets as shown in Table 7. Although they have different names, these metrics are similar: GUI grounding tasks consistently measure the hit rate of predicted bounding boxes, while GUI navigation tasks focus on single-step accuracy.

Click Accuracy. The proportion of test samples where the predicted location falls in the ground truth element's bounding box.

Element Accuracy (Ele.Acc). Comparing the selected element with all acceptable elements. For vision-based methods, it is the same as Click Accuracy.

Operation F1 (Op.F1). Token-level F1 score for the predicted operation.

Step Success Rate (Step SR) and Step Accuracy. The proportion of successful steps. A step is regarded as successful only if both the selected element and the predicted operation are correct.

Baselines We compare the performance of our model with various baselines, including some closed source large models such as GPT-4o (OpenAI et al., 2024), Gemini-1.5-pro (Team, 2024)

Method	Cross-Task			Cross-Website			Cross-Domain		
	Ele.Acc	Op.F1	Step.SR	Ele.Acc	Op.F1	Step.SR	Ele.Acc	Op.F1	Step.SR
GPT-4 (OpenAI, 2023)	41.6	60.6	36.2	35.8	51.1	30.1	37.1	46.5	26.4
OmniParser (Lu et al., 2024)	42.4	87.6	39.4	41.0	84.8	36.5	45.5	85.7	42.0
Qwen2-VL-2B (Wang et al., 2024c)	37.7	86.4	33.2	36.0	79.2	27.6	36.3	81.8	30.7
ShowUI-2B (Lin et al., 2024)	39.9	88.6	37.2	41.6	83.5	35.1	39.4	**86.8**	35.2
SpiritSight-2B (Huang et al., 2025)	51.7	87.2	44.9	44.9	83.6	37.8	42.4	83.5	36.9
SeeClick-7B (Cheng et al., 2024)	28.3	87.0	25.5	21.4	80.6	16.4	23.2	84.8	20.8
CogAgent-18B (Hong et al., 2024)	22.4	53.0	17.6	18.4	42.4	13.4	20.6	42.0	15.5
InReAct	**57.8**	**89.4**	**50.8**	**52.6**	**88.8**	**47.4**	**55.4**	85.4	**49.2**

Table 4: GUI navigation results on **Mind2web** dataset.

Method	Low-Level	High-Level
Claude* (Anthropic, 2024)	19.4	12.5
GPT-4o (OpenAI et al., 2024)	19.4	20.8
Aria-UI-3.9B (Yang et al., 2024)	67.3	10.2
OS-Atlas-4B (Wu et al., 2024)	80.6	67.5
Aguvis-7B (Xu et al., 2024)	80.5	61.5
InReAct	**83.1**	**68.1**

Table 5: GUI navigation results on **AndroidControl** dataset.

and Claude(Anthropic, 2024), as well as numerous open-source GUI agents such as SpiritSight-2B (Huang et al., 2025), AGUVIS-7B (Xu et al., 2024), Uground (Gou et al., 2025), Aria-UI(Yang et al., 2024), OSAtlas (Wu et al., 2024), ShowUI (Lin et al., 2024), InfiGUIAgent (Liu et al., 2025b), UGround (Gou et al., 2025), CogAgent (Hong et al., 2024), SeeClick (Cheng et al., 2024), Qwen2-VL-2B, Qwen2-VL-7B (Wang et al., 2024c).Implementation details refer to appendix B.

4.2 Overall Performance

GUI Grounding Evaluation The evaluation results on the ScreenSpot (Cheng et al., 2024) are presented in Table 3. Our model achieves an average accuracy of 77.8%, outperforming all competing baselines, including both open-source and commercial GUI agents. In particular, Our model surpasses strong 2B-level models such as ShowUI (75.1%) and InfiGUIAgent (76.3%), and even outperforms several significantly larger models such as CogAgent (47.4%, 18B) and UGround (73.3%, 7B).

As in shown in table 2,on the more challenging ScreenSpot-Pro (Li et al., 2025b)dataset, our method achieves an overall accuracy of 20.7%, representing a substantial improvement under complex, high-resolution desktop GUI environments. This dataset covers a wide spectrum of professional software, including CAD tools, scientific platforms, development IDEs, and creative applications, each demanding fine-grained grounding of visually dense and domain-specific interfaces. Despite the increased difficulty, our model consistently outperforms previous strong baselines. It exceeds ShowUI by 13.0% percentage points and surpasses AriaUI by 9.4% points, demonstrating particularly strong capabilities in GUI grounding

GUI Navigation Evaluation On Mind2Web (Deng et al., 2023), the task requires agents to perform realistic web navigation across diverse websites. As shown in Table 4, InReAct achieves strong performance across all three generalization splits—cross-task, cross-website, and cross-domain—outperforming previous methods in terms of element grounding accuracy, operation-level F1, and step-wise success rate. This indicates that our model not only understands the semantics of web elements but also makes consistent and interpretable action decisions across varying environments. The consistent gains across generalization splits demonstrate InReAct's robustness in unseen task goals and layouts.

On AndroidControl (Li et al., 2024), we evaluate the model under two settings: Low-Level, which focuses on simple, mostly single-step interactions (e.g., tapping a button); and High-Level, which involves complex, multi-step instructions with long-horizon dependencies. As shown in Table 5, InReAct achieves 83.1 on the Low-Level set and 68.1 on the High-Level set, outperforming all prior baselines. While strong results in Low-Level indicate precise grounding and action formulation, the improvements in High-Level highlight the model's ability to plan and reason over multi-step interaction sequences.

4.3 Ablation Study

As summarized in Table 6, we conduct ablation studies to assess the impact of the proposed strategies in both curriculum learning (CL) and reinforcement learning (RL) training stages. For Mind2Web and AndroidControl, we only reported the results on cross-task SR and High-level. There are several notable observations as follows:

Curriculum Learning We examine the role of each step in our curriculum learning framework. Removing Step 1, where the model learns basic GUI interactions, leads to a significant performance drop, as the agent fails to acquire essential skills. Similarly, omitting Steps 2 and 3, which introduce grounding and reasoning tasks progressively, also harms performance, showing their importance for understanding and reasoning over GUI elements. Furthermore, replacing curriculum learning with multi-task training across all steps results in worse performance, demonstrating that staged, gradual training enhances learning and generalization more effectively than joint training.

Reinforcement Learning To further evaluate the effectiveness of the reinforcement learning (RL) stage, we first replace it with a standard supervised fine-tuning (SFT) setup. The results show that RL brings consistent performance gains across all datasets. On the two navigation datasets—Mind2Web and AndroidControl—RL yields improvements of +5.1 and +4.3, respectively, demonstrating its ability to produce more accurate actions and more robust decision-making.

Furthermore, to specifically justify our reward function design within the RL framework, we compare our proposed piecewise reward function against two alternatives: a binary (hit-or-miss) reward, which provides an accuracy signal without any dedicated precision component, similar to approaches in works like UI-R1 and GUI-R1, and a continuous linear reward (defined as $R_{dist} = 1 - \hat{d}$, where $\hat{d}$ is the normalized distance to the target's center). This analysis, presented in Table 6, reveals that the design of the reward signal plays an important role. On ScreenSpot, our method (77.8) outperforms both the linear (77.0) and binary (76.8) baselines. On the more challenging ScreenSpot-Pro dataset, our approach (20.7) also provides a modest but consistent improvement over the linear (20.3) and binary (19.8) rewards.

The advantage of our piecewise reward function lies in two aspects. First, it delivers clearer learning signals by introducing reward "steps," which motivate the agent to cross precision thresholds rather than being limited by diminishing returns near the target. Second, it aligns more closely with real-world GUI logic, where clicks have discrete outcomes—successful or not—and central clicks are inherently safer than edge clicks. This design thus captures the practical structure of interaction more faithfully than a simple linear function. It is worth noting that our thresholds are chosen empirically and may not be optimal; further tuning is left for future work.

Impact of Observation and Reasoning Components The results, presented in Table 6, underscore the importance of both components within our observe-think-act cycle. Removing the thinking phase (w/o Thought) leads to a consistent and significant performance drop across all four benchmarks, with the most notable decreases observed on the complex ScreenSpot-Pro (-2.5 points) and AndroidControl (-2.3 points) datasets. This indicates that the explicit reasoning step is crucial for formulating effective plans and making robust decisions in challenging environments.

Similarly, removing the observation phase (w/o Observation) also consistently degrades performance across all tested datasets, such as causing a 1.8-point drop on ScreenSpot. An interesting finding is that the performance degradation from removing the thinking phase is more pronounced than that from removing the observation phase on the three more complex datasets. This suggests that while both components are beneficial, the structured reasoning provided by the "Thinking" step is arguably the most critical factor for the agent's success in complex scenarios. These findings strongly justify the synergistic design of our InReAct framework.

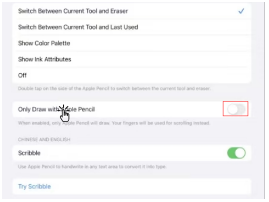
4.4 Case Study

To qualitatively assess the effectiveness of our proposed curriculum learning and reinforcement learning strategies, we conduct a case study on the ScreenSpot dataset. As illustrated in Figure 3, we present two pairs of examples highlighting the contributions of each training stage.

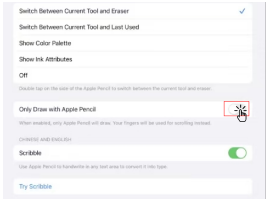
In Figure 3a, without curriculum learning, the agent clicks on the text "Only draw with Apple Pencil" instead of the adjacent switch, showing weak perception and semantic grounding. In contrast,

Method	ScreenSpot	ScreenSpot-Pro	Mind2Web	AndroidControl
InReAct	**77.8**	**20.7**	**50.8**	**68.1**
Stage 1: Curriculum Learning				
- w/o Step 1	76.3	19.2	47.4	65.9
- w/o Step 2	68.6	10.8	37.8	57.4
- w/o Step 3	74.7	17.4	42.5	62.3
- w/o curriculum	75.4	19.5	48.5	66.4
Stage 2: Reinforcement Learning				
- w/ Binary Reward	76.8	19.8	49.2	67.0
- w/ Linear Reward	77.0	20.3	50.2	67.3
- w/ SFT	76.2	16.8	45.7	63.8
Ablation on Agent Components				
- w/o Observation	76.0	19.7	49.2	66.8
- w/o Thought	76.6	18.2	48.6	65.8

Table 6: Ablation study of INREACT. We evaluate the impact of different training stages, reward functions, and core agent components. For Mind2Web, we report the Step SR on the cross-task split, while for AndroidControl, we report results on the High-Level setting.

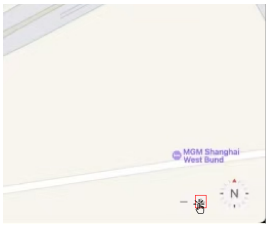


(a) ✗ **Instruction:"Only draw with apple pencil"**. Without curriculum learning



(b) ✓ **Instruction:"Only draw with apple pencil"**. InReAct achieves accurate grounding

(c) ✗ **Instruction:"Zoom in"**. Without reinforcement learning



(d) ✓ **Instruction:"Zoom in"**. InReAct achieves accurate grounding

Figure 3: Case study on **ScreenSpot** dataset.

InReAct (Figure 3b) correctly targets the switch, indicating better alignment between instructions and interface elements.

In Figure 3c, without reinforcement learning, for the instruction "Zoom in," the predicted click point is near the correct bounding box but slightly outside it, indicating general understanding but lacking fine-grained precision. In contrast, as shown in Figure 3d, InReAct achieves significantly more accurate localization, clicking within the target region. This confirms that our reinforcement learning stage enhances fine-grained control.

5 Conclusion

In this paper, we present INREACT, a multimodal GUI agent framework that integrates observation, reasoning, and action for complex GUI interactions. It is trained in two stages: curriculum-learning to build perception, grounding, and reasoning skills, followed by reinforcement learning with a task-specific reward that improves action correctness and spatial precision.

We evaluate INREACT across four datasets: ScreenSpot, ScreenSpot-Pro, Mind2Web, and AndroidControl, and demonstrate consistent improvements over prior methods. Our findings highlight the effectiveness of combining structured learning and outcome-driven optimization for enhancing GUI agent performance, setting a new direction for generalizable and interpretable vision-language agents.

Acknowledgments

This research was supported by Key-Area Research and Development Program of Guangdong Province, Granted No. 2024B1111060004.

Limitations

Limited Action Space Coverage Our current action space, while sufficient for common GUI interactions such as click, type, and scroll, does not fully cover the diversity of real-world GUI operations. Advanced actions like drag-and-drop, multi-touch gestures, system-level controls, or context-menu navigation are not well supported in our framework. Extending the action space to include such capabilities would be essential for achieving broader applicability in real-world software environments.

Inference Speed and Efficiency Although InReAct achieves strong performance on various GUI tasks, its inference speed has room for optimization. The structured reasoning process—particularly the multi-step observation and thinking phases—introduces additional computational overhead. This may affect the agent's ability to perform real-time or near-real-time interactions, which is crucial for applications requiring fast response times. Future work could explore model compression, caching mechanisms, or streamlined reasoning pipelines to improve efficiency.

Limited Handling of Dynamic Interfaces Our experiments are primarily conducted on static interfaces. However, real-world GUI applications often involve dynamic content updates, such as pop-up notifications, auto-refreshing elements, or state transitions, that can disrupt perception and decision-making. Our current framework lacks explicit mechanisms to detect, adapt to, or reason about such changes over time. Incorporating temporal modeling or environment change detection would be a valuable direction for future research.

References

Anthropic. 2024. Developing a computer use model. https://www.anthropic.com/news/developing-computer-use. Accessed: 2025-04-12.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A frontier large vision-language model with versatile abilities. *CoRR*, abs/2308.12966.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923.

Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. 2025. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. https://github.com/Deep-Agent/R1-V. Accessed: 2025-02-02.

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. 2024. Guicourse: From general vision language models to versatile GUI agents. *CoRR*, abs/2406.11317.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. 2023. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *CoRR*, abs/2312.14238.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9313–9332. Association for Computational Linguistics.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samual Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2025. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, and Jie Tang. 2024. Cogagent: A visual language model for GUI agents. In *IEEE/CVF Conference on Computer Vision and*

Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, pages 14281–14290. IEEE.

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, and Jindong Chen. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *CoRR*, abs/2209.08199.

Zhiyuan Huang, Ziming Cheng, Junting Pan, Zhaohui Hou, and Mingjie Zhan. 2025. Spiritsight agent: Advanced GUI agent with one look. *CoRR*, abs/2503.03196.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXVIII*, volume 15126 of *Lecture Notes in Computer Science*, pages 161–178. Springer.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5295–5306. ACM.

Hongxin Li, Jingfan Chen, Jingran Su, Yuntao Chen, Qing Li, and Zhaoxiang Zhang. 2025a. Autogui: Scaling GUI grounding with automatic functionality annotations from llms. *CoRR*, abs/2502.01977.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025b. Screenspot-pro: Gui grounding for professional high-resolution computer use. *Preprint*, arXiv:2504.07981.

Wei Li, William W. Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. 2024. On the effects of data scale on computer control agents. *CoRR*, abs/2406.03679.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for GUI visual agent. *CoRR*, abs/2411.17465.

Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. 2025a. Harnessing webpage uis for text-rich visual understanding. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, and 11 others. 2024. Autoglm: Autonomous foundation agents for guis. *CoRR*, abs/2411.00820.

Yuhang Liu, Pengxiang Li, Zishu Wei, Congkai Xie, Xueyu Hu, Xinchen Xu, Shengyu Zhang, Xiaotian Han, Hongxia Yang, and Fei Wu. 2025b. Infiguiagent: A multimodal generalist GUI agent with native reasoning and reflection. *CoRR*, abs/2501.04575.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. 2025c. Visual-rft: Visual reinforcement fine-tuning. *CoRR*, abs/2503.01785.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024. Omniparser for pure vision based GUI agent. *CoRR*, abs/2408.00203.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. 2025. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *Preprint*, arXiv:2503.21620.

Haohao Luo, Yang Deng, Ying Shen, See-Kiong Ng, and Tat-Seng Chua. 2024. Chain-of-exemplar: Enhancing distractor generation for multimodal educational question generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 7978–7993. Association for Computational Linguistics.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. 2025. Gui-r1 : A generalist r1-style vision-language action model for gui agents. *Preprint*, arXiv:2504.10458.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. UI-TARS: pioneering automated GUI interaction with native agents. *CoRR*, abs/2501.12326.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*, page 20. IEEE/ACM.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *Preprint*, arXiv:2504.07615.

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024a. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024b. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *CoRR*, abs/2401.16158.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024c. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *CoRR*, abs/2409.12191.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2022. A survey on curriculum learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(9):4555–4576.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. Os-atlas: A foundation action model for generalist gui agents. *Preprint*, arXiv:2410.23218.

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

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. 2024. Aria-ui: Visual grounding for gui instructions. *Preprint*, arXiv:2412.16256.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. 2025. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI 2025, YokohamaJapan, 26 April 2025- 1 May 2025*, pages 70:1–70:20. ACM.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Datasets Details

We provide training data used in the curriculum learning stage and reinforcement learning stage, as well as statistical information for the four datasets used in the experiment, as shown in Table 8 and Table 7, respectively.

B Implementation Details

We build our GUI agent upon Qwen2-VL-2B-Instruct (Wang et al., 2024c), a vision-language model with strong general capabilities. It is worth noting that we chose Qwen2-VL-2B instead of Qwen's latest series Qwen2.5-VL-3B (Bai et al., 2025) because 2B is lighter, and most of the methods we compared used Qwen2-VL-2B as the base model or larger parameter models before Qwen2-VL. We validated the effectiveness of our proposed method in a fairer situation. The training process consists of two stages: Curriculum Learning (CL) and Reinforcement Learning (RL). For Curriculum Learning, we employ the LLaMA Factory (Zheng et al., 2024) framework for one epoch at each stage with the visual backbone frozen. We employ AdamW with a cosine annealing learning rate scheduler, setting the initial learning rate to 1e-5 and batch size to 16. For Reinforcement Learning, we use the R1-V (Chen et al., 2025) framework for training over two epochs, setting the initial learning

Datasets	Platforms	Task	Metric	# Test Samples	History ?
ScreenSpot-Pro (Li et al., 2025b)	Web, PC, Mobile	Grounding	clickAcc	1,272	✗
ScreenSpot (Cheng et al., 2024)	Web, PC, Mobile	Grounding	clickAcc	1,581	✗
Mind2Web (Deng et al., 2023)	Web	Navigation	Ele.Acc, Op.F1, Step SR	6,418	✓
AndroidControl-High (Li et al., 2024)	Mobile	Navigation	Step Accuracy	8,444	✓
AndroidControl-Low (Li et al., 2024)	Mobile	Grounding	Step Accuracy	8,444	✗

Table 7: Statistics of GUI datasets.

Usage	**Source**	**Number**
CL	GUIChat(Chen et al., 2024)	40k
	ScreenQA (Hsiao et al., 2022)	17k
	MultiUI (Liu et al., 2025a)	50k
	AutoGUI (Li et al., 2025a)	50k
	SeeClick (Cheng et al., 2024)	100k
	GUIEnv (Chen et al., 2024)	100k
	Aguvis-stage2 (Xu et al., 2024)	35k
RL	OmniAct (Kapoor et al., 2024)	2k
Total		394k

Table 8: Training data statistics from various sources.

rate to 1e-6 and batch size to 16. At each stage, we utilize the strategies of DeepSpeed optimization (Rajbhandari et al., 2020), BF16 format, and gradient checkpointing to save GPU memory. All training is conducted on 2*A100 GPU (80G).