InfiGUIAgent: A Multimodal Generalist GUI Agent with Native Reasoning and Reflection

Yuhang Liu[1,3], Pengxiang Li[2], Zishu Wei[1], Congkai Xie[3], Xueyu Hu[1], Xinchen Xu[1], Shengyu Zhang[1]*, Xiaotian Han[4]*, Hongxia Yang[2]*, Fei Wu[1]
[1]Zhejiang University, Hangzhou, Zhejiang, China,
[2]The Hong Kong Polytechnic University, Hong Kong, China,
[3]InfiX.ai, Hong Kong, China,
[4]Independent Researcher
siriusliuyh@gmail.com, sy_zhang@zju.edu.cn, hongxia.yang@polyu.edu.hk

Abstract

Graphical User Interface (GUI) Agents, powered by multimodal large language models (MLLMs), have shown great potential for task automation on computing devices such as computers and mobile phones. However, existing agents face challenges in multi-step reasoning and reliance on textual annotations, limiting their effectiveness. We introduce *InfiGUIAgent*, an MLLM-based GUI Agent trained with a two-stage supervised fine-tuning pipeline. Stage 1 enhances fundamental skills such as GUI understanding and grounding, while Stage 2 integrates hierarchical reasoning and expectation-reflection reasoning skills using synthesized data to enable native reasoning abilities of the agents. *InfiGUIAgent* achieves competitive performance on several GUI benchmarks, highlighting the impact of native reasoning skills in enhancing GUI interaction for automation tasks.

1 Introduction

Graphical User Interface (GUI) Agents have emerged as powerful tools for automating tasks on computing devices, including mobile phones and computers. These agents can understand and interact with GUIs to execute complex operations, significantly enhancing user productivity and expanding the scope of automated task completion (Hu et al., 2024b; Hong et al., 2024; Zhang and Zhang, 2023; Qi et al., 2024; Xie et al., 2024; Vu et al., 2024; Yu et al., 2024; Wen et al., 2023). Recent developments in multimodal large language models (MLLMs) (Bai et al., 2023b; Li et al., 2024c; Team et al., 2024; Dai et al., 2022) have significantly advanced the potential of GUI Agents. MLLMs possess powerful visual understanding capabilities and can reason based on visual information, making them a promising foundation for building sophisticated GUI Agents. These models can interpret

*Corresponding author.

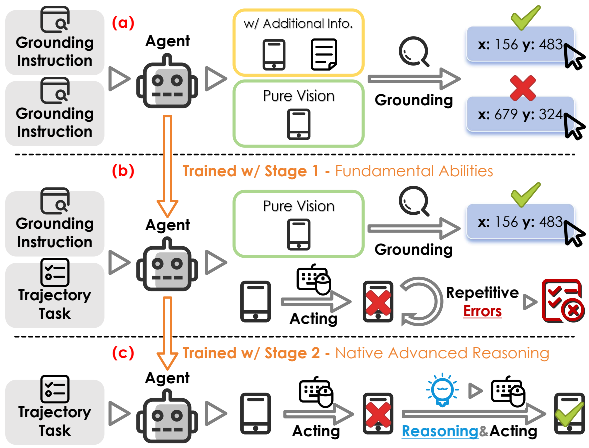


Figure 1: *InfiGUIAgent*, trained via a two-stage SFT pipeline, addresses key GUI agent challenges. **(a)** Baseline grounding is often unreliable, with auxiliary text offering partial gains at the cost of overhead, while pure vision struggles. **(b)** Stage 1 enhances fundamental GUI understanding and grounding, but agents remain prone to repetitive errors in multi-step trajectory tasks. **(c)** Stage 2 integrates native hierarchical and expectation-reflection reasoning, empowering *InfiGUIAgent* for successful complex task automation.

complex interface elements and adapt to a wide range of tasks, leading to more efficient and robust automation (Hong et al., 2024; Jiang et al., 2023; You et al., 2025; Nong et al., 2024; Vu et al., 2024).

Despite their promise, MLLM-based GUI Agents encounter hurdles in achieving robust, autonomous task completion. A primary challenge lies in their interaction with the visual interface. Many agents struggle to connect natural language instructions to the correct GUI elements (such as icons or buttons) within complex layouts. Determining the precise location for an interaction, like a tap, also remains unreliable. This often compels a reliance on additional information, such as accessibility trees or Set-of-Marks (Yang et al., 2023a), to interpret the GUI. However, GUIs are inherently visual; textual augmentation can lead to informa-

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics Volume 1: Long Papers, pages 1035–1051
March 24-29, 2026 ©2026 Association for Computational Linguistics

tion loss, increased computational overhead, and platform inconsistencies, hindering practical deployment.

Beyond these interaction challenges, agents face limitations in their reasoning and planning capabilities (Zhang and Zhang, 2023; Qi et al., 2024; Yu et al., 2024). Many do not effectively leverage insights from previous steps or reflect on past actions, frequently leading to repetitive errors. Separately, agents often make decisions that yield low long-term value or are incorrect, indicating difficulties in decomposing high-level goals into effective sub-steps. These issues, particularly acute in MLLMs with fewer parameters, can cause agents to become trapped in unproductive behavioral loops.

Such behaviors suggest underlying weaknesses: a deficit in robust visual-semantic grounding and spatial awareness at the perceptual level, and at the cognitive level, a lack of intrinsic mechanisms for proactive historical analysis (hindering self-correction) and strategic, multi-scale planning.

Overcoming these limitations requires more than applying existing MLLMs or relying on elaborate prompting. We introduce *InfiGUIAgent*, an MLLM-based GUI Agent developed through a two-stage supervised fine-tuning (SFT) pipeline (Figure 1). This pipeline is architected to: first, instill **foundational GUI understanding and grounding capabilities (Stage 1)** by utilizing diverse datasets covering layout comprehension, grounding tasks, and GUI QA, thereby directly improving the agent's ability to interpret and interact with visual interfaces. Second, cultivate **advanced reasoning skills (Stage 2)**—hierarchical reasoning and expectation-reflection reasoning—using carefully synthesized data from trajectories, enabling these to be performed 'natively' by the agent. This overall approach aims to equip the agent with internalized mechanisms for planning and adaptive learning from its operational history. *InfiGUIAgent* achieves competitive performance on several GUI benchmarks, highlighting the impact of this integrated approach. Our main contributions are threefold:

- We propose a two-stage supervised fine-tuning pipeline to comprehensively improve both the fundamental abilities and advanced reasoning abilities of GUI Agents.
- We synthesize SFT data with two advanced reasoning skills: hierarchical reasoning and expectation-reflection reasoning, enabling the agents to natively perform complex reasoning. This data will be open-sourced to promote community development.
- We build *InfiGUIAgent* by supervised fine-tuning a model using our SFT data and conduct experiments on several GUI benchmarks, demonstrating that our model achieves competitive performance.

2 Related Works

2.1 Multimodal LLMs

Large Language Models (LLMs) (Floridi and Chiriatti, 2020; Touvron et al., 2023; Bai et al., 2023a; Xiao et al., 2021) have significantly enhanced the capabilities of AI systems in tackling a wide range of tasks (Hu et al., 2024c; Li et al., 2024d), thanks to their exceptional ability to process complex semantic and contextual information. The remarkable power of LLMs has also inspired exploration into their potential for processing multimodal data, such as images. Typically, the architecture of Multimodal Large Language Models (MLLMs) consists of three main components: a pre-trained large language model, a trained modality encoder, and a modality interface that connects the LLM with the encoded modality features. Various vision encoders, such as ViT (Dosovitskiy et al., 2021), CLIP (Radford et al., 2021), and ConvNeXt (Liu et al., 2022), extract visual features, which are integrated using techniques like adapter networks (Liu et al., 2023), cross-attention layers (Alayrac et al., 2022), and visual expert modules (Wang et al., 2023). These methods have facilitated the development of high-performing MLLMs, such as Qwen-VL (Bai et al., 2023b), GPT-4 Vision (OpenAI, 2023), BLIP-2 (Li et al., 2023) and InfiMM (Liu et al., 2024), thus opening new avenues for LLMs in processing GUI tasks.

2.2 MLLM-based GUI Agents

Agents are AI systems that perceive their environments, make decisions, and take actions to complete specific tasks. LLMs reaching human-level intelligence have greatly enhanced the ability to build agents. For GUI tasks, LLMs that read HTML code to perceive GUIs are developed (Wen et al., 2023). However, various works have shown that learning to interact with the visual form of the GUIs can show superior performance (Hu et al., 2024b). Therefore, MLLM-based GUI Agents are developed. ILuvUI (Jiang et al., 2023) fine-tuned LLaVA to enhance general GUI understanding, while AppAgent (Zhang et al., 2023) explored

Figure 2: *InfiGUIAgent* is trained in two stages. **Stage 1** cultivates fundamental abilities using diverse datasets covering GUI understanding (element recognition and layout comprehension), question answering, instruction grounding, general knowledge, and tool usage. **Stage 2** introduces native advanced reasoning, employed during both training and inference. This stage follows a cyclical process at each step, consisting of `Reflection`, `Hierarchical Reasoning` (strategic and tactical layers), `Action`, and `Expectation`. Each step receives the overall task, the history of previous screenshots and reasoning, and the current environment as input. `Reflection` assesses the previous action's outcome against its expectation, while `Expectation` predicts the outcome of the current action for subsequent reflection.

app usage through autonomous interactions. CogAgent (Hong et al., 2024) integrated high-resolution vision encoders, and Ferret-UI-anyres (You et al., 2025) employed an any-resolution approach. Building upon these works, our study focuses on developing a more lightweight agent with a simplified architecture for GUI tasks, aiming to improve ease of deployment.

3 Method

In this section, we introduce our two-stage supervised fine-tuning strategy for building *InfiGUIAgent*, as shown in Figure 2. In stage 1, we focus on improving fundamental abilities such as understanding and grounding, particularly considering the complexity of GUIs. In stage 2, we move on to improve the native reasoning abilities of agents for handling complicated GUI tasks.

3.1 Stage 1: Training for Fundamental Abilities

Considering the complexity of GUIs, which involve diverse data formats such as HTML code, high-resolution interfaces cluttered with small icons and text, general MLLMs lack fundamental abilities in both understanding GUI and grounding the actions. To address this, we first collected a range of existing visual-language and GUI datasets for supervised fine-tuning in stage 1. We gathered data covering several GUI tasks from multiple sources to ensure a comprehensive capabilities improvement (see Table 1). The datasets can be categorized into five parts:

- **GUI Understanding.** Datasets focusing on GUI element recognition, layout comprehension, and semantic interpretation, including Screen2Words (Wang et al., 2021) and Screen Annotation (Baechler et al., 2024).
- **Grounding.** Datasets capture various user interaction sequences and operation patterns, including GUIEnv (Chen et al., 2024), RICO Semantic Annotation (Sunkara et al., 2022), SeeClick-Web (Cheng et al., 2024), RICO SCA (Li et al., 2020a), Widget Caption (Li et al., 2020b), UIBert Reference Expression (Bai et al., 2021) and OmniAct-Single Click (Kapoor et al., 2024).
- **Question Answering.** Datasets contain GUI-specific QA tasks, including GUIChat (Chen et al., 2024), ScreenQA (Hsiao et al., 2022) and Complex QA (Yin et al., 2023).
- **General Knowledge.** Multimodal datasets maintain model's general capabilities, including LLaVA-OneVision (Li et al., 2024a) and PixMo (MDeitke et al., 2024).
- **Tool Usage.** Datasets cover general tool using, including Glaive-function-calling (Glaive AI, 2024).

Due to the diversity of our data sources, we im-

Table 1: Training datasets used in stage 1 of supervised fine-tuning.

Dataset	Platform	Category	# of Samples
GUI-related Datasets			
GUIEnv (Chen et al., 2024)	Webpage	Grounding	150,000
RICO Semantic Annotation (Sunkara et al., 2022)	Mobile	Grounding	150,000
SeeClick-Web (Cheng et al., 2024)	Webpage	Grounding	100,000
RICO SCA (Li et al., 2020a)	Mobile	Grounding	100,000
Widget Caption (Li et al., 2020b)	Mobile	Grounding	70,000
GUIChat (Chen et al., 2024)	Webpage	QA	40,000
ScreenQA (Hsiao et al., 2022)	Mobile	QA	17,000
UIBert Reference Expression (Bai et al., 2021)	Mobile & Mobile	Grounding	16,000
Screen2Words (Wang et al., 2021)	Mobile	Understanding	12,000
Complex QA (Yin et al., 2023)	Mobile	QA	11,000
Screen Annotation (Baechler et al., 2024)	Mobile	Understanding	5,400
OmniAct-Single Click (Kapoor et al., 2024)	Webpage & Desktop	Grounding	4,800
Non-GUI Datasets			
LLaVA-OneVision (Li et al., 2024a)	-	General	250,000
PixMo (MDeitke et al., 2024)	-	General	68,800
Glaive-function-calling (Glaive AI, 2024)	-	Tool Usage	5,000

plemented comprehensive format standardization across all datasets. Additionally, we adopted the Reference-Augmented Annotation format (see Section 3.1.2) to enhance the model's ability to ground visual elements with textual descriptions, enabling precise spatial referencing while maintaining natural language flow.

3.1.1 Data Preprocessing and Standardization

Given the diversity of our data sources, we implemented comprehensive preprocessing steps to standardize the data format across all datasets. We normalized the coordinate system by following (Wang et al., 2024), mapping all spatial coordinates to a relative scale of [0, 1000]. This standardization facilitates consistent representation of both point and box annotations in JSON format, with points expressed as $\{"x" : x, "y" : y\}$ and bounding boxes as $\{"x1" : x_1, "y1" : y_1, "x2" : x_2, "y2" : y_2\}$. In this coordinate system, the origin $\{"x" : 0, "y" : 0\}$ is located at the screen's top-left corner, with the x-axis extending rightward and the y-axis downward. The bottom-right corner corresponds to coordinates $\{"x" : 1000, "y" : 1000\}$. To enhance data quality, we implemented two additional preprocessing steps:

Instruction Enhancement. For datasets with ambiguous instructions (e.g., those varying in phrasing or lacking explicit intent across different data sources), we developed standardized instruction templates to establish clear correspondence between commands and their expected outcomes.

Response Refinement. For entries with complex or inconsistent response formats, we utilized Qwen2-VL-72B (Bai et al., 2023b) to reformulate responses while preserving their semantic content. Each reformulation underwent validation to ensure accuracy and consistency.

3.1.2 Reference-Augmented Annotation

To better leverage the spatial information available in our collected datasets and enhance the model's visual-language understanding of GUIs, we implemented a reference-augmented annotation format. This format enables bidirectional referencing between GUI elements and textual responses. Specifically, we adopted the following structured notation:

```
<ref type="box"
    coords={"x1": x1, "y1": y1,
         "x2": x2, "y2": y2}
    note="GUI annotation">
    corresponding text
</ref>
```

The format consists of several key components: the reference type (either "box" for rectangular regions or "point" for specific locations), coordinate specifications (x1, y1, x2, y2 for boxes or x, y for points), optional annotative notes, and the corresponding textual content. To generate training data in this format, we prompted Qwen2-VL-72B (Bai et al., 2023b) to seamlessly integrate GUI spatial information with original responses, maintaining natural language flow while preserving precise spatial references.

3.2 Stage 2: Training for Native Reasoning

Building upon the foundational capabilities such as understanding and grounding, GUI Agents must also master advanced reasoning skills to effectively handle complex tasks. We identify two crucial reasoning skills : (1) Hierarchical reasoning, which enables planning and task decomposition, helping agents structure complex tasks into manageable subtasks and execute them efficiently (Huang and Chang, 2023; Zhang et al., 2024b; Huang et al., 2024), and (2) Expectation-reflection reasoning, which fosters adaptive self-correction and reflection (Shinn et al., 2023; Yao et al., 2023; Hu et al., 2024a), enabling agents to learn from past actions and improve decision-making consistency. These reasoning skills are integrated into the training datasets of agents, so that they can reason with these skills natively without any extra prompting. To achieve this, we generate SFT data incorporating these reasoning skills based on existing trajectory data (see Table 2) and continue fine-tuning the model from stage 1.

3.2.1 Hierarchical Reasoning

Effective execution of GUI tasks demands both overarching strategic planning and meticulous tactical execution. To achieve this, we synthesize trajectory data with a hierarchical reasoning with two distinct layers:

- **Strategic Layer.** Strategic layer is responsible for high-level task decomposition and sub-goal planning. This layer analyzes the overall task objective and determines the sequence of subtasks needed for completion.
- **Tactical Layer.** Tactical layer handles the selection and grounding of concrete actions. Based on the strategic layer's planning, agent select appropriate GUI operations and adjusts their parameters to match the target.

3.2.2 Expectation-Reflection Reasoning

To enhance action consistency and foster autonomous self-correction, we incorporate Expectation-reflection reasoning into the training datasets. This iterative process enhances the agent's ability to adapt and learn from its actions through a structured reflection cycle:

- **Reasoning.** After reflection (except the first step), the agents conduct hierarchical reasoning.
- **Action.** After the reasoning, the agent takes the action.
- **Expectation.** Following each action, the agent generates expected outcomes which are used to be verified at the next step.
- **Reflection.** The agent evaluates whether its actions achieved the expected results and generating a textual summary of the reflection.

3.2.3 Agent-Environment Interface

We formulate the GUI interaction as a process where an agent interacts with a mobile environment. Let $s_t \in \mathcal{S}$ denote the environment state at step t, where $\mathcal{S}$ represents the state space. The agent can observe the state through a screenshot observation o_t and performs actions $a_t \in \mathcal{A}$, where $\mathcal{A}$ is the action space. The environment transitions from s_t to s_{t+1} following $s_{t+1} \sim P(\cdot|s_t, a_t)$, where P represents the transition probability function.

The agent receives a task goal g and maintains access to a history window of size n. At each step t, the agent's input consists of:

- Goal g
- Current observation o_t
- Historical context $H_t = \{(o_i, r_i, a_i)\}_{i=t-n}^{t-1}$, where r_i represents the reasoning process

Based on these inputs, the agent generates a reasoning process r_t and predicts an action a_t. The interaction follows a standard protocol using function calls and responses:

```
Assistant Message:
<tool_call>
{
    "name": "action_name",
    "arguments": {"action_parameters"}
}
</tool_call>

Tool Message:
<tool_response>
{
    "name": "gui_operation",
    "content": {
        "status": "success | failure",
        "current_ui": <image>,
        "current_task": <task_description>
    }
}
</tool_response>
```

3.2.4 Modular Action Space

Given the diverse action spaces across collected datasets, we categorized and standardized the actions by unifying their names and parameters, merg-

Table 2: UI action reasoning datasets used in the training process

Dataset	Platform	# of Samples
GUIAct (Chen et al., 2024)	Webpage & Mobile	10,000
AMEX (Chai et al., 2024)	Mobile	3,000
Android in the Zoo (Zhang et al., 2024a)	Mobile	2,000
Composition: Stage 1-aligned	-	30,000

Category	Operations
Single-point operations	`tap`, `click`, `hover`, `select`
Two-point operations	`swipe`, `select_text`
Directional operations	`scroll`
Text input	`input`, `point_input`
Parameterless operations	`remember`, `enter`, `home`, `back`
State settings	`set_task_status`

Table 3: Categorization of actions in the action space.

ing similar operations where appropriate. The resulting action space $\mathcal{A}$ consists of independent, composable operations that can be flexibly combined based on task requirements, as shown in Table 3. This modular design allows for dynamic action space configuration while maintaining a consistent interface across different platforms and scenarios. The modularity of the action space is implemented by modifying the system prompt. The system prompt we use is shown in Appendix A.2.

3.2.5 Reasoning Process Construction

To construct high-quality reasoning data to stimulate the model's native reasoning capabilities, we leverage more capable MLLMs (e.g. Qwen2-VL-72B) to generate structured reasoning processes based on existing interaction trajectories. The construction process involves several key components:

- **Screenshot Description.** For each observation o_t in the trajectory, we generate a detailed description d_t. This step addresses the limitation that some MLLM models do not support interleaved image-text input formats well. To establish clear correspondence between observations (screenshots) and steps, we generate detailed descriptions to replace the screenshots, which helps facilitate the subsequent reasoning process construction.
- **Reflection.** Given the previous expectation e_{t-1} and current observation o_t, we generate a reflection f_t that evaluates the outcome of the previous action.
- **Strategic Layer.** The strategic reasoning consists of two parts: First, a summary is generated based on the n-step history $H_t = \{(o_i, r_i, a_i)\}_{i=t-n}^{t-1}$ and current observation o_t. Then, the planning component is generated with access to the actual action a_t to ensure alignment with the trajectory.
- **Tactical Layer.** This layer's reasoning is constructed using the generated reflection f_t and strategic layer output. The actual action a_t from the trajectory is incorporated to ensure the tactical reasoning leads to appropriate action selection.
- **Expectation.** For each state-action pair (s_t, a_t), we generate an expectation e_t based on current observation o_t, reasoning process r_t, and action a_t. Notably, we deliberately avoid using the next state s_{t+1} in this generation process. Although using s_{t+1} could improve the agent's accuracy in modeling state transitions, while using s_{t+1} could lead to perfect expectations, such an approach might impair the agent's ability to handle expectation mismatches during deployment.

While we avoid using s_{t+1} in expectation generation to maintain robustness, we also explore the possibility of improving state transition modeling through a parallel next-state prediction task. Using the trajectory data, we construct additional training examples where the agent learns to predict the next state description d_{t+1} given the current observation o_t and action a_t. This auxiliary task helps the agent learn state transition dynamics, while keeping the expectation generation process independent of future states.

4 Experiments

4.1 Experimental Setting

4.1.1 Implementation Details

In stage 1, we sample 1M samples in total as illustrated in Table 1. In stage 2, we synthesized 45K samples based on trajectories from datasets shown in Table 2. We employ a full fine-tuning approach to continually supervised fine-tune Qwen2-VL-2B (Bai et al., 2023c). The hyperparameters used for training were a learning rate of 5×10^{-6}, a batch

Model	Accuracy (%)						Avg.
	Mobile		Desktop		Web		
	Text	Icon	Text	Icon	Text	Icon	
Proprietary Models							
GPT-4o[1] (OpenAI, 2024)	30.5	23.2	20.6	19.4	11.1	7.8	18.8
Gemini-1.5-pro[2] (Team et al., 2024)	76.2	54.1	65.5	39.3	52.2	32.0	53.2
Open-source Models							
Qwen2-VL-2B (Wang et al., 2024)	24.2	10.0	1.4	9.3	8.7	2.4	9.3
Qwen2-VL-7B (Wang et al., 2024)	61.3	39.3	52.0	45.0	33.0	21.8	42.9
Qwen2-VL-72B (Wang et al., 2024)	73.6	65.5	54.1	57.1	53.0	44.2	57.9
CogAgent (Hong et al., 2024)	67.0	24.0	74.2	20.0	70.4	28.6	47.4
SeeClick (Cheng et al., 2024)	78.0	52.0	72.2	30.0	55.7	32.5	53.4
UGround-7B (Gou et al., 2024)	82.8	60.3	<u>82.5</u>	<u>63.6</u>	80.4	**70.4**	73.3
ShowUI-2B (Lin et al., 2024)	**92.3**	**75.5**	76.3	61.1	<u>81.7</u>	<u>63.6</u>	<u>75.1</u>
Ours							
InfiGUIAgent-2B	**92.3**	<u>70.3</u>	**85.6**	**65.7**	**83.0**	<u>63.6</u>	**76.8**

Table 4: Performances on various platforms (Mobile, Desktop, Web) on Screenshot. All experiments were conducted using raw screenshot information. Results marked in **bold** represent the best performance, and those <u>underlined</u> indicate the second-best performance.

Model	Success Rate			
	Easy	Middle	Hard	Overall
Open-source Models				
Qwen2-VL-2B (Wang et al., 2024)	0.00	0.00	0.00	0.00
Qwen2-VL-7B (Wang et al., 2024)	0.00	0.00	0.00	0.00
Qwen2-VL-72B (Wang et al., 2024)	0.08	0.00	0.00	0.04
LLaVa-OV-7B (Li et al., 2024b)	0.00	0.00	0.00	0.00
ShowUI-2B (Lin et al., 2024)	0.18	0.00	0.00	0.09
Ours				
InfiGUIAgent-2B	**0.25**	0.00	0.00	**0.13**

Table 5: Performances on AndroidWorld. Results marked in **bold** represent the best performance

size of 256, and a maximum sequence length of 32k to accommodate long trajectories. The training was conducted for 1 epoch with a warmup ratio of 0.05. To optimize the learning process across the two stages, the vision module was unfrozen during the first stage to facilitate the learning of fundamental UI knowledge, and subsequently frozen during the second stage to allow the model to focus more on reasoning capabilities. We leverage ZeRO (stage 0) (Rajbhandari et al., 2020) technology and FlashAttention-2 (Dao, 2023) to accelerate training and reduce memory consumption, enabling full parameter fine-tuning of the model across 8 A800 80GB GPUs.

4.1.2 Evaluation Benchmarks

ScreenSpot. ScreenSpot (Cheng et al., 2024) is an evaluation benchmark for GUI grounding, consisting of over 1,200 instructions from iOS, Android, macOS, Windows, and Web environments, with annotated element types.

AndroidWorld. AndroidWorld (Rawles et al., 2024) is a fully functional Android environment that provides reward signals for 116 programmatic tasks across 20 real-world Android apps. We find that Android World uses Set-of-Marks (SoM) (Yang et al., 2023b) to enhance the agent's grounding ability. However, when humans operate smartphones, their brains do not label elements on the screen. Over-reliance on SoM can lead to insufficient focus on pixel-level grounding ability. Therefore, in our experiments, agents respond to the raw image rather than the annotated image.

4.2 Main Results

ScreenSpot. Table 4 provides the results of different models across three platforms (Mobile, Desktop and Web) and two element types (Text and Icon) on ScreenSpot (Cheng et al., 2024). InfiGUIAgent-2B achieves highest accuracy of 76.3%, surpassing several strong baselines such as ShowUI (Lin et al., 2024) (75.1%) and UGround-7B (Gou et al., 2024) (73.3%), which is even with larger parameters size.

AndroidWorld. Table 5 compares the success rates of *InfiGUIAgent* with open-source models on AndroidWorld (Rawles et al., 2024). InfiGUIAgent-2B achieves an overall success rate of 0.09, outperforming open-source models of similar size, such as ShowUI-2B (Lin et al., 2024) (0.07), and model with much more parameters such as LLaVa-OV-7B (Li et al., 2024b) (0.00) and Qwen2-VL-72B (Bai et al., 2023b) (0.05).

4.3 Ablation Studies

Configuration	ScreenSpot (% Accuracy)	AndroidWorld (Success Rate)
InfiGUIAgent-2B	**76.8**	**0.13**
Ablations		
w/o Stage 2	76.0	0.00
w/o Stage 1	74.3	0.09
w/o Reasoning	76.6	0.09

Table 6: Ablation study of InfiGUIAgent components. Scores for the full pipeline are taken from the main paper results for InfiGUIAgent-2B.

We performed ablation studies (Table 6) to assess the contributions of InfiGUIAgent's main components. The full InfiGUIAgent-2B model achieves 76.8% accuracy on ScreenSpot and 0.13 SR on AndroidWorld.

Removing Stage 2 training ("w/o Stage 2") causes the AndroidWorld SR to drop to 0.00, highlighting that the advanced reasoning skills from this stage are crucial for complex, multi-step tasks. Conversely, omitting the dedicated Stage 1 training ("w/o Stage 1") degrades ScreenSpot accuracy to 74.3%. While this underscores the importance of Stage 1 for foundational GUI understanding and grounding, the model's residual performance on ScreenSpot in this "w/o Stage 1" setting is likely supported by the Stage 2 training data, which includes a "Stage 1-aligned" component composed of data thematically similar to that used in Stage 1. The "w/o Stage 1" configuration still achieves 0.09 SR on AndroidWorld, suggesting that advanced reasoning can offer some utility even with a less robust dedicated perceptual foundation, but is outperformed by the full model.

The "w/o Reasoning" variant, representing a simplified Stage 2 without the specific hierarchical and expectation-reflection structures, yields 76.6% on ScreenSpot and 0.09 SR on AndroidWorld. This indicates that while any learning in Stage 2 is beneficial for complex tasks compared to removing Stage 2 entirely, the structured reasoning patterns contribute to the full model's superior performance. These results affirm the distinct and vital roles of both Stage 1 and Stage 2.

4.4 Qualitative Analysis

A qualitative analysis of the models' performance on the stopwatch operation task reveals significant disparities in their capabilities. The baseline model (Qwen2-VL-2B) exhibits behavior consistent with random exploration, demonstrating a fundamental lack of understanding regarding the screen state or the overarching task goal. As detailed in Table 7, its sequence of three taps appears arbitrary, failing to engage with crucial UI elements such as the "Stopwatch" tab or the "Start" button. This deficiency in identifying and interacting with relevant components results in an inability to progress, leaving the task uncompleted, and suggests an absence of visual grounding and goal-oriented reasoning.

In stark contrast, InfiGUIAgent demonstrates a structured and effective multi-step approach, visually and descriptively detailed in Table 7. For instance, in its initial step, the agent correctly identifies the need to navigate to the Stopwatch tab and executes the tap, expecting the UI to transition. Subsequent steps involve confirming the UI state, identifying the next sub-goal, locating the relevant UI element for that sub-goal, and performing the action with a clear expectation. Each step showcases its ability to process the visual information from the screen (represented by the included images for each step) and make an informed decision.

This side-by-side qualitative assessment underscores InfiGUIAgent's markedly superior capabilities in robust visual grounding, effective hierarchical task decomposition, and strong goal-oriented reasoning. The agent's explicit articulation of its internal reasoning, planned actions, and expected outcomes at each juncture, supported by the visual context of each step, highlights a sophisticated and transparent understanding of the GUI interaction process, ultimately leading to successful task execution where the baseline model fails. See Table 7 in Appendix for more details.

5 Conclusion

In this work, we propose *InfiGUIAgent*, a novel MLLM-based GUI Agents. By constructing comprehensive training datasets with two-stage supervised fine-tuning, we enhance the model's ability to understand, reason, and interact with GUIs. Our evaluation, conducted using raw screenshots without relying on additional GUI metadata, demonstrates the model's applicability to real-world scenarios. Experimental results show that our model performs well on GUI tasks and surpass several open-source baselines. We markedly enhance the model's core abilities in visual GUI understanding, multi-step task reasoning, and precise interactive control.

Limitations

While our work demonstrates promising advancements in GUI task automation through the proposed two-stage training framework, several limitations remain. First, due to our focus on enabling efficient deployment on edge devices, we primarily explored small-scale models (e.g., Qwen2-VL-2B). While this approach ensures practicality, larger models may offer further performance improvements, particularly in handling more complex tasks. Second, although hierarchical reasoning and expectation-reflection reasoning enhance the agent's ability to decompose tasks and adapt dynamically, real-world GUI interactions often involve unforeseen complexities, such as error recovery, cross-application coordination, or dynamically changing interfaces. Enhancing the agent's robustness to such challenges remains an open research direction.

Acknowledgements

Work by Yuhang Liu was done during an internship at InfiX.ai. This work was supported by the National Natural Science Foundation of China (No. 62402429, U24A20326, 62441236), the Key Research and Development Program of Zhejiang Province (No. 2025C01026, 2024C03270), the Ningbo Yongjiang Talent Introduction Programme (2023A-397-G), the Young Elite Scientists Sponsorship Program by CAST (2024QNRC001), and the Tencent AI Lab Rhino-Bird Focused Research Program. The author gratefully acknowledges the support of Zhejiang University Education Foundation Qizhen Scholar Foundation.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Aguera y Arcas. 2021. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Yu Bowen, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xing Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023a. Qwen technical report. *ArXiv*.

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

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

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

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. *arXiv preprint arXiv:2406.11317*.

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. *arXiv preprint arXiv:2401.10935*.

Yong Dai, Duyu Tang, Liangxin Liu, Minghuan Tan, Cong Zhou, Jingquan Wang, Zhangyin Feng, Fan Zhang, Xueyu Hu, and Shuming Shi. 2022. One model, multiple modalities: A sparsely activated approach for text, sound, image, video and code. *Preprint*, arXiv:2205.06126.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

Glaive AI. 2024. Glaive function calling dataset. https://huggingface.co/datasets/glaiveai/glaive-function-calling. Accessed: 2024-01-08.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. 2024. Navigating the digital world as humans do: Universal visual grounding for gui agents. *arXiv preprint arXiv:2410.05243*.

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

Yu-Chung Hsiao, Fedir Zubach, Gillbune Baechler, Victor Carbune, Jason Lin, Maria Wang, Srinivas Sunkara, Yun Zhu, and Jindong Chen. 2022. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*.

Xueyu Hu, Kun Kuang, Jiankai Sun, Hongxia Yang, and Fei Wu. 2024a. Leveraging print debugging to improve code generation in large language models. *Preprint*, arXiv:2401.05319.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shawn Wang, Xinchen Xu, Shuofei Qiao, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. 2024b. Os agents: A survey on mllm-based agents for general computing devices use. *Preprints*.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024c. Infiagent-dabench: Evaluating agents on data analysis tasks. *arXiv preprint arXiv:2401.05507*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. *Preprint*, arXiv:2212.10403.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Yue Jiang, Eldon Schoop, Amanda Swearngin, and Jeffrey Nichols. 2023. Iluvui: Instruction-tuned language-vision modeling of uis from machine conversations. *arXiv preprint arXiv:2310.04869*.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem Alshikh, and Ruslan Salakutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. *arXiv preprint arXiv:2402.17553*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yan-Liu Li, Ziwei Liu, and Chunyuan Li. 2024a. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03329*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024b. Llava-onevision: Easy visual task transfer. *Preprint*, arXiv:2408.03326.

Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Fan Zhou, Chengen Huang, Yanpeng Li, Chongyan Zhu, Xiaoyi Ren, Chao Li, Yifan Ye, Lihuan Zhang, Hanshu Yan, Guoyin Wang, Bei Chen, and Junnan Li. 2024c. Aria: An open multimodal native mixture-of-experts model. *Preprint*, arXiv:2410.05993.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Linyi Li, Shijie Geng, Zhenwen Li, Yibo He, Hao Yu, Ziyue Hua, Guanghan Ning, Siwei Wang, Tao Xie, and Hongxia Yang. 2024d. Infibench: Evaluating the question-answering capabilities of code large language models. *arXiv preprint arXiv:2404.07940*.

Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020a. Mapping natural language instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*.

Yang Li, Luheng Li, Gangaand He, Jingjie Zheng, Hong Li, and Zhiwei Guan. 2020b. Widget captioning: Generating natural language description for mobile user interface elements. *arXiv preprint arXiv:2010.04295*.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*.

Haogeng Liu, Quanzeng You, Yiqi Wang, Xiaotian Han, Bohan Zhai, Yongfei Liu, Wentao Chen, Yiren Jian, Yunzhe Tao, Jianbo Yuan, Ran He, and Hongxia Yang. 2024. Infimm: Advancing multimodal understanding with an open-sourced visual language model. In *Annual Meeting of the Association for Computational Linguistics*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. 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*.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.

Matt MDeitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bramsom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli van der Bilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Kuo-Hao Gupta, Tanmay sna Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. 2024. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. *arXiv preprint arXiv:2409.17146*.

Songqin Nong, Jiali Zhu, Rui Wu, Jiongchao Jin, Shuo Shan, Xiutian Huang, and Wenhao Xu. 2024. Mobileflow: A multimodal llm for mobile gui agent. *arXiv preprint arXiv:2407.04346*.

OpenAI. 2023. Gpt-4v(ision) system card.

OpenAI. 2024. Gpt-4o. Accessed: 2025-01-03.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, et al. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. *Preprint*, arXiv:1910.02054.

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

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Srinivas Sunkara, Maria Wang, Lijuan Liu, Gilles Baechler, Yu-Chung Hsiao, Abhanshu Sharma, James Stout, et al. 2022. Towards better semantic understanding of mobile interfaces. *arXiv preprint arXiv:2210.02663*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*.

Minh Duc Vu, Han Wang, Jieshan Chen, Zhuang Li, Shengdong Zhao, Zhenchang Xing, and Chunyang Chen. 2024. Gptvoicetasker: Advancing multi-step mobile task efficiency through dynamic interface exploration and learning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–17.

Bryan Wang, Gang Li, Xin Zhou, Zhourong Chen, Tovi Grossman, and Yang Li. 2021. Screen2words: Automatic mobile ui summarization with multimodal learning. *arXiv preprint arXiv:2108.03353*.

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

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. 2023. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2023. Autodroid: Llm-powered task automation in android. *arXiv preprint arXiv:2308.15272*.

Chaojun Xiao, Xueyu Hu, Zhiyuan Liu, Cunchao Tu, and Maosong Sun. 2021. Lawformer: A pre-trained language model for chinese legal long documents. *AI Open*, 2:79–84.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023a. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *Preprint*, arXiv:2310.11441.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023b. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Agent lumos: Unified and modular training for open-source language agents. *arXiv preprint arXiv:2311.05657*.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. 2025. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, pages 240–255. Springer.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Exact: Teaching ai agents to explore with reflective-mcts and exploratory learning. *arXiv preprint arXiv:2410.02052*.

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

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

Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. 2024b. Llm as a mastermind: A survey of strategic reasoning with large language models. *Preprint*, arXiv:2404.01230.

Zhuosheng Zhang and Aston Zhang. 2023. You only look at screens: Multimodal chain-of-action agents. *arXiv preprint arXiv:2309.11436*.

A Prompt Templates

This section details the various prompt templates employed for trajectory and grounding tasks within our study. These prompts are designed to guide the AI agent in understanding its role, the environment, and the specific requirements of each task.

A.1 Prompt Templates for Grounding Tasks

For grounding tasks, which require the agent to identify specific elements or locations on the UI, we use tailored prompt templates. The choice of template depends on whether the output is a single point or a bounding box.

A.1.1 Point Output

When the grounding task requires the output of specific coordinates (a point) related to a given instruction, the following prompt template is used:

Prompt Template for Grounding (Point)

Output the relative coordinates of the icon, widget, or text most closely related to "instruction" in this screenshot, in the format of "{"x": x, "y": y}", where x and y are in the positive directions of horizontal left and vertical down respectively, with the origin at the top left corner, and the range is 0-1000.

Here, `{instruction}` is a placeholder for the natural language instruction describing the target element (e.g., "the 'Login' button", "the user's profile picture").

A.1.2 Bounding Box Output

When the grounding task requires identifying a bounding box for an element related to a given instruction, the following prompt template is employed:

Prompt Template for Grounding (Bounding Box)

Output the relative coordinates of the icon, widget, or text most closely related to "instruction" in this screenshot, in the format of "{"x1": x1, "y1": y1, "x2": x2, "y2": y2}", where x1, y1, x2 and y2 are in the positive directions of horizontal left and vertical down respectively, with the origin at the top left corner, and the range is 0-1000.

Similarly, `{instruction}` serves as a placeholder for the user's instruction. The coordinates `x1`, `y1` represent the top-left corner of the bounding box, and `x2`, `y2` represent the bottom-right corner.

A.2 System Prompt for Trajectory Tasks

For trajectory-based tasks, which involve the agent performing a sequence of actions to achieve a goal on a user interface, we utilize the following comprehensive system prompt:

System Prompt for Trajectory Tasks

You are an AI agent capable of interacting with a user interface through function calling. Your task is to assist users in completing UI-related tasks or answering related questions.

The current UI type is: {ui_type}

When referring to positions on the UI:
- Use integer relative coordinates ranging from 0 to 1000.
- The origin (0, 0) is at the top-left corner of the screen. The x-axis increases to the right, and the y-axis increases downward.

You are provided with function signatures within <tools> </tools> XML tags. You may call one or more functions to assist with the UI-related tasks. Don't make assumptions about what values to plug into functions.
<tools>
{action_space}
</tools>
For each function call return a json object with function name and arguments within <tool_call> </tool_call> tags with the following schema:
<tool_call>
"name": <function-name>, "arguments": <args-dict>
</tool_call>

In this prompt, `{ui_type}` is a placeholder that is replaced with the actual type of the user interface being interacted with. The available actions and their function signatures are provided to the agent within the `tools` and `/tools` tags at runtime.

B Cases

B.1 Stage 1: Fundamental Abilities

We demonstrate the fundamental abilities trained in Stage 1 through three cases: GUI Understanding (Figure 3), Grounding (Figure 4), and Question Answering (Figure 5).

B.2 Stage 2: Native Reasoning

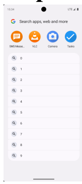
We provide two representative cases to demonstrate the reasoning and interaction process of *InfiGUIAgent*.

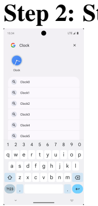
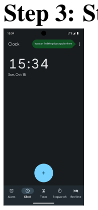
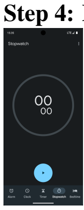
Reply to a Message Figure 6 illustrates a step where the agent needs to reply to a specific message in a messaging application. The reasoning process involves identifying the "Start chat" button and grounding the action to initiate the reply process.

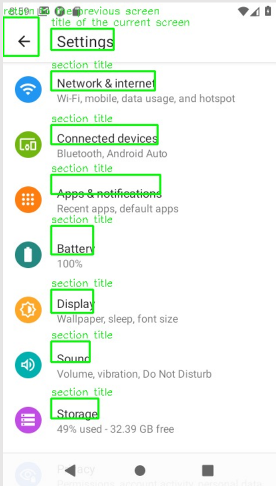
Creating a New Contact Figure 7 and Figure 8 demonstrate sequential steps for creating a new contact. In the first step (Step K), the agent navigates to the "Contacts" section by reasoning and grounding the action to the corresponding tab. In the following step (Step K+1), the agent initiates the contact creation process by identifying and tapping the "Create new contact" button. These sequential steps highlight the agent's hierarchical reasoning and grounding abilities.

Table 7: Qualitative Comparison for Task: `ClockStopWatchRunning` (Illustrating a 4-step InfiGUIAgent process with screen states)

Baseline Output (Qwen2-VL-2B, temp=1.0)	**InfiGUIAgent Output (Stage 1 + 2, temp=1.0)**

Actions:
`tap(x=710, y=55)`
`tap(x=500, y=880)`
`tap(x=240, y=240)`
Provided Observation:
Issues taps without clear reasoning. Fails to locate Stopwatch tab or start button; never reaches running state. No goal conditioning or hierarchical planning evident.
	Step 1: Navigate to Stopwatch Tab
	
	Agent Reasoning (Key Points): Clock app open, tabs visible. Goal: Run stopwatch. Need to:
	• Tap Stopwatch tab.
	• Wait for UI transition.
	Stopwatch tab is rightmost on top bar.
	Action: `tap(x=800, y=120)`
	Expected Outcome: Stopwatch screen loads with a start button.

	Step 2: Start the Stopwatch  *Agent Reasoning (Key Points):* Now on Stopwatch screen. Timer and Start button visible. Goal: Start stopwatch. Need to tap Start button. Start button is large, circular, centered below timer. **Action:** `tap(x=500, y=850)` *Expected Outcome:* Timer starts counting, stopwatch running.
	Step 3: Stop the Stopwatch  *Agent Reasoning (Key Points):* Stopwatch is running. Next goal is to stop it. “Stop” button is now visible (often replaces “Start”). **Action:** `tap(x=500, y=850)` *Expected Outcome:* Timer stops counting. “Reset” and “Lap” buttons might appear.
	Step 4: Reset the Stopwatch  *Agent Reasoning (Key Points):* Stopwatch is stopped. Final goal is to reset. “Reset” button should be visible. **Action:** `tap(x=250, y=850)` *Expected Outcome:* Timer resets to 00:00.00. “Start” button reappears.

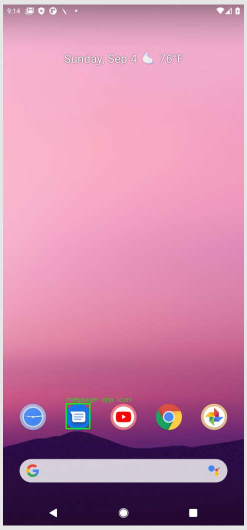


Instruction: Describe the UI in detail. *You should reference components and locations in the UI.*

Answer: This screenshot shows the settings menu of an Android device, displaying various configuration options. The top of the screen features a navigation bar with a back arrow, indicating the ability to return to the previous screen. The main content area lists several settings categories, each with a brief description and icon for easy identification.

Key UI components and elements present on the screen include:
- A <ref type="box" coords={"x1": 0, "y1": 31, "x2": 131, "y2": 110} note="return to the previous screen">back arrow</ref> at the top left corner, allowing the user to navigate back to the previous screen.
- The <ref type="box" coords={"x1": 175, "y1": 54, "x2": 404, "y2": 98} note="title of the current screen">Settings</ref> title at the top center, indicating the current page.
…

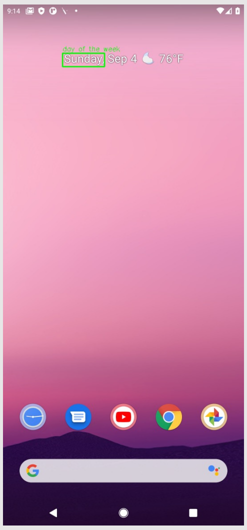
Figure 3: Case of GUI Understanding.



Instruction: Where can I send a message to someone? *You should reference components and locations in the UI.*

Answer: To send a message, you need to access the messaging app. The key components on the home screen are the app icons. Specifically, you should look for the <ref type="box" coords={"x1": 262, "y1": 766, "x2": 360, "y2": 814} note="message app icon">message app icon</ref> located at the bottom left of the screen. This icon is typically used to open the messaging app. Once you tap on this icon, you will be taken to the messaging app where you can send a message.

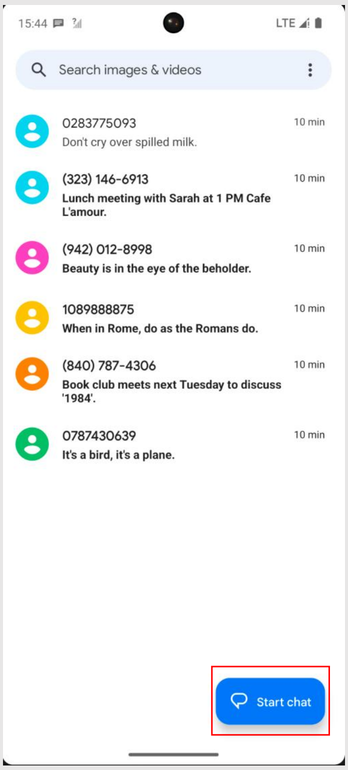
Figure 4: Case of Grounding.



Instruction: What day is it tomorrow? *You should reference components and locations in the UI.*

Answer: To determine the day of the week, we need to look at the date displayed at the top of the screen. The date is shown as <ref type="box" coords={"x1": 248, "y1": 94, "x2": 422, "y2": 119} note="day of the week">Sunday</ref>. This indicates that today is Sunday. Therefore, tomorrow will be Monday.

Figure 5: Case of Question Answering.



Goal: Reply to +18407874306 with message: Yoga class every Tuesday and Thursday at 6 PM. in Simple SMS Messenger.

Reflection: The expectation from the previous step was to open the Simple SMS Messenger app by tapping on its icon. The current screen content shows the Simple SMS Messenger app's ...

Strategic Layer

Summary: The overall task objective is to send the message "Yoga class every Tuesday and Thursday at 6 PM" to +18407874306 ...

Planning: To achieve the final goal, the plan is to: **first**, initiate a new chat; **second**, enter the recipient's phone number (+18407874306); **third**, compose the message ...

Tactical Layer

Reasoning: The core component on the current screen is the "Start chat" button. The key point of the task objective in the current step is to initiate a new chat. Therefore, I need to tap on the "Start chat" button.

Grounding: Based on the reasoning, the current action should be to tap on the "Start chat" button. **The button is located at the bottom right of the screen**, a prominent blue circle with a white chat bubble icon. Therefore, I should tap on the "Start chat" button.

Action:
<tool_call>
{"name": "tap", "arguments": {"point": {"x": 824, "y": 886}}}
</tool_call>

Expectation: The purpose of this action is to open the new chat interface. This action will likely result in a new screen appearing ...

Figure 6: Case of Native Advanced Reasoning. The agent's goal is to reply to a message

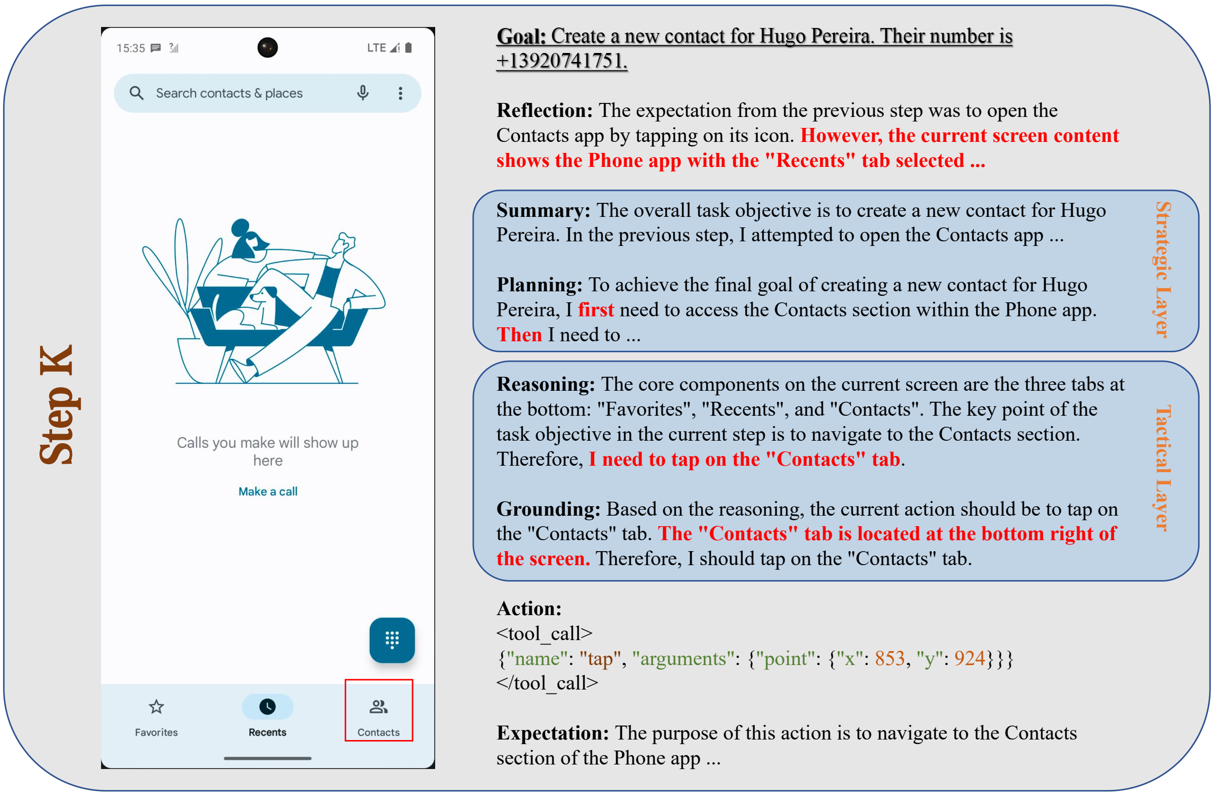


Figure 7: Case of Native Advanced Reasoning. The agent's goal is to create a new contact.

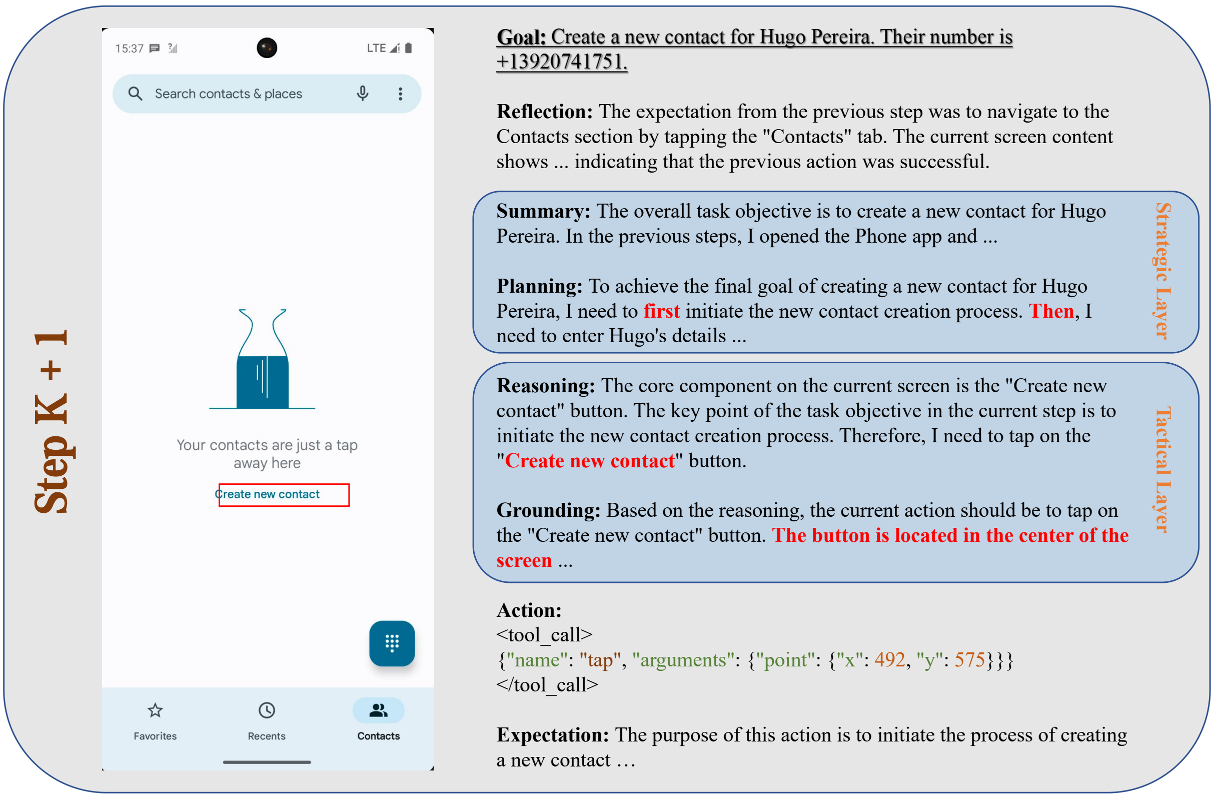


Figure 8: Case of Native Advanced Reasoning. The agent's goal is to create a new contact.