# UItron: Foundational GUI Agent with Advanced Perception and Planning

**Zhixiong Zeng**[†], **Jing Huang**[†], **Liming Zheng, Wenkang Han,**
**Yufeng Zhong, Lei Chen, Longrong Yang, Yingjie Chu, Yuzhi He, Lin Ma**[*]
Meituan
zengzhixiong@meituan.com, forest.linma@gmail.com
Project: https://github.com/UITron-hub/UItron

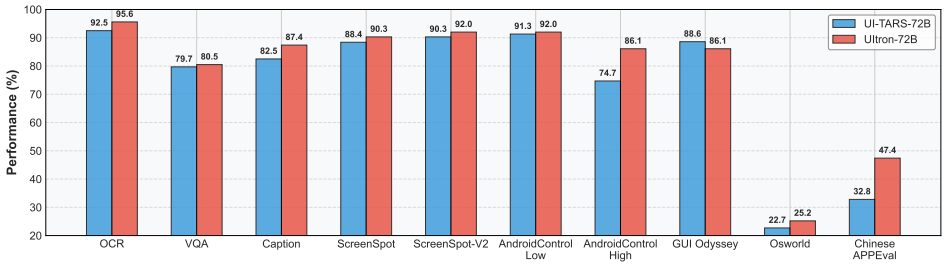


Figure 1: Comparison of UItron and UI-Tars in perception/grounding/planning and Chinese scenarios.

## Abstract

GUI agent aims to enable automated operations on Mobile/PC devices, which is an important task toward achieving artificial general intelligence. The rapid advancement of VLMs accelerates the development of GUI agents, owing to their powerful capabilities in visual understanding and task planning. However, building a GUI agent remains a challenging task due to the scarcity of operation trajectories, the availability of interactive infrastructure, and the limitation of initial capabilities in foundation models. In this work, we introduce UItron, an open-source foundational model for automatic GUI agents, featuring advanced GUI perception, grounding, and planning capabilities. UItron highlights the necessity of systemic data engineering and interactive infrastructure as foundational components for advancing GUI agent development. It not only systematically studies a series of data engineering strategies to enhance training effects, but also establishes an interactive environment connecting both Mobile and PC devices. In training, UItron adopts supervised finetuning over perception and planning tasks in various GUI scenarios, and then develop a curriculum reinforcement learning framework to enable complex reasoning and exploration for online environments. As a result, UItron achieves superior performance in benchmarks of GUI perception, grounding, and planning. In particular, UItron highlights the interaction proficiency with top-tier Chinese mobile APPs, as we identified a general lack of Chinese capabilities even in state-of-the-art solutions. To this end, we manually collect over one million steps of operation trajectories across the top 100 most popular apps, and build the offline and online agent evaluation environments. Experimental results demonstrate that UItron achieves significant progress in Chinese app scenarios, propelling GUI agents one step closer to real-world application.

[†] Equal contribution. [*] Corresponding author.

Preprint.

# 1 Introduction

GUI agents [26, 84, 82, 77, 67, 21, 72, 49, 33, 7] aim to automatically execute complex tasks in various digital environments such as PC and Mobile, satisfying the growing expectations of autonomous decision-making and software control in human-computer interaction. These agents decompose the task instructions into multi-step actions by observing the screen status, then navigate and manipulate the on-screen elements following the human-like interactive manners (*i.e.*, click, scroll). This human-like approach provides visually trackable trajectories with step-by-step task execution process, enabling convenient user interaction and explainable decision-making. Therefore, GUI agents have received a rapidly growing amount of attention, becoming an important research topic toward achieving artificial general intelligence.

The pursuit of automated GUI agent has been going on for a long time. Early methods [15, 24, 29] utilize optical character recognition and icon detectors to parse GUI environments into textual elements (e.g., HTML and AXTree) as input to the LLM, leveraging its powerful reasoning capabilities to plan and generate multi-step executable actions. The rapid advancement of vision-language models catalyze a series of GUI agents (e.g., [77, 67, 21, 72, 49]) that operate directly on visual GUI images, which achieve superior performance within the framework of unified perception and planning in pure vision. A representative work is UI-TARS [49], which achieves leading performance via a large amount of data engineering and a carefully designed iterative training framework. Recently, some R1-style works [44, 41, 86, 54, 73, 16] represented by GUI-R1 [86] designs multimodal reasoning data and explores the typical group relative policy optimization to improve reasoning ability, reporting improved results in grounding benchmarks. To address the limited adaptability in offline environments, ZeroGUI [73] and ARPO [16] propose the online reinforcement learning framework, which adopts VLM-based automatic reward estimation to assess task success and continuously learn from the GUI environments, without hand-crafted evaluation functions.

However, developing automatic GUI agents still remains a highly challenging task due to several limitations: the scarcity of annotated operation trajectories, the availability of interactive infrastructure, and the limitation of initial capabilities in foundation models. As illustrated in Figure 1, training a GUI agent necessitates precise GUI perception, grounding, offline planning, and online planning capabilities, thereby rendering the collection of adequately annotated trajectory data exceedingly difficult. Moreover, in contrast to traditional multimodal tasks, GUI agents face the additional challenge of requiring an interactive environment to execute the model-generated actions that enables multi-round interaction. Additionally, extensive empirical evidence shows that current foundation models typically possess limited performance in GUI scenarios, substantially hindering progress toward effective GUI agent development.

In this paper, we introduce UItron, a powerful open-source foundational model for automatic GUI agents, with powerful GUI perception, task grounding, offline/online planning capabilities. UItron emphasizes the importance of data engineering and interactive infrastructure for developing GUI agents. For data engineering, we significantly expand the available operation trajectories through three key aspects: data unification, trajectory distillation, and manual annotation over different domains. Moreover, we systematically investigated a series of data engineering strategies to enhance training effectiveness, including the utilization of various trajectory elements (*e.g.*, observation, thought and action), the exploration of different reasoning formats, and the incorporation of diverse reflection mechanisms like backtracing. We also find the advantages of integrating multi-task UI-related data and general multimodal data. For interactive infrastructure, we build an interactive environment connecting both Mobile and PC devices. It not only simplifies trajectory data collection by automatically recording screenshots and coordinates, but also provides a foundation for online reinforcement learning (RL) during training.

During training, we employ a three-stage training strategy over several GUI scenarios, which includes GUI perception, planning and RL. Note that the RL stage is specifically designed to enhance complex reasoning and exploration capabilities within online environments. First, UItron adopts a supervised finetuning strategy for GUI perception and planning tasks. The perception task focuses on improving the basic understanding ability of the vision-language model in GUI scenarios, such as grounding, captioning, VQA, and OCR. The planning stage concentrates on forecasting the next action based on historical actions. Then UItron develops a Curriculum Reinforcement Learning (CuRL) framework with group relative policy optimization algorithm on trajectory data. To address the problem of sparse rewards, CuRL first computes dense rewards from the action steps in the offline environment (simple),

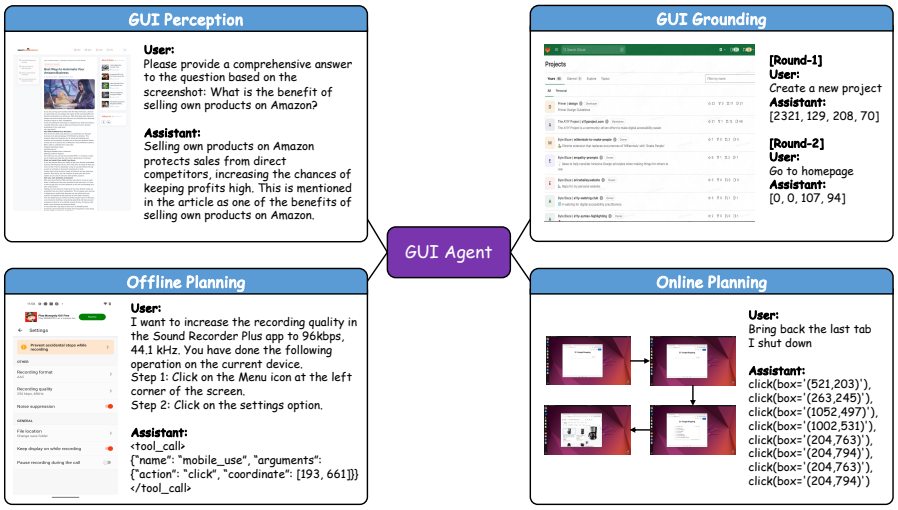


Figure 2: The core capabilities of GUI agent, including GUI perception, grounding, offline planning and online planning.

and then computes the task-level reward for the trajectory data in the online environment (complex). In addition, to improve the credibility of rewards in the RL process, we strictly filter the trajectories that are predicted correctly by multiple scoring models simultaneously.

In particular, UItron emphasizes its ability to interact with top-tier mobile Apps in China, as we find that even state-of-the-art solutions generally underperform in Chinese scenarios for GUI agents. To this end, we meticulously annotate over one million action steps from the top 100 monthly active Apps, covering mainstream interaction scenarios such as social networking, office work, entertainment, and shopping. Based on this, we constructed an offline evaluation dataset to assess the capabilities of GUI agents in Chinese App scenarios, evaluating the performance of different models based on two classical evaluation metrics: single-step success rate and task completion rate. To evaluate the realistic interaction performance of GUI agent in real-world applications, we also build an Android-based cloud real-device environment for online evaluation. Specifically, we develop a rollout method to alternately execute actions and refresh status between GUI agent and Android-based cloud environment. Next, we developed an automated scoring mechanism that leverages multiple VLMs to score the entire task trajectory, and averages the scores to generate evaluation results. Experimental results not only confirm the limitations of existing methods, but more importantly, demonstrate that UItron has achieved substantial progress in Chinese application scenarios, advancing GUI agents toward practical and real-world deployment.

The main contributions of this work are summarized as follows:

- We present a systematic investigation of data engineering and interactive infrastructure that effectively supports the development of foundational GUI agents.
- We develop a curriculum reinforcement learning framework with dense and credible rewards for trajectory data in GUI agents.
- We open-source UItron, achieving superior performance in benchmarks of GUI perception, grounding and offline planning, and competitive results in online agent environments.
- We significantly improves the interactive capabilities of UItron in Chinese scenarios through carefully labeled data and tailored online environments.

## 2 Related Works

### 2.1 MLLM

Large language models (LLMs) have shown strong generalization capabilities and instruction-following abilities. However, they can only process text information, while real-world applica-

tions require models to understand visual information. Thus, Multimodal Large Language Models (MLLMs) such as GPT-4 [1] and LLaVA [38] utilize visual encoders and visual projectors to integrate visual data into large language models. MLLM works on visual encoder mainly focus on improving input resolution, which can be roughly divided into direct scaling and patch-division. Direct scaling, *e.g.*, [5, 36, 26], inputs higher-resolution images into the encoder, usually requiring further fine-tuning of the encoder or the use of a pre-trained encoder with higher resolution. Patch-division, *e.g.*, [35, 32], splits high-resolution images into multiple patches and then reuses the low-resolution encoder, which has gradually become the mainstream choice because it can support dynamic resolution or natural resolution [11, 59, 68, 6, 23], making it suitable for processing varying visual information. MLLM works on projector mainly focuses on the effective mapping of visual information, which can be roughly divided into token-level and feature-level. The token-level projector converts the output features into tokens and concatenates them with text tokens before feeding them into the large language model. Some works use Q-Former, *e.g.*, [81, 14], but most works directly use MLP to bridge the modal gap due its simplicity and generalization, *e.g.*, [38, 52, 36, 59, 6]. The feature-level projector, *e.g.*, [2, 83, 5, 60], inserts additional modules that enable in-depth interaction and fusion between text and visual features. For example, Flamingo [2] inserts additional cross-attention layers into the frozen LLM layers to enhance language features using external visual cues.

Beyond architecture, scaling data is key to improve the performance of MLLMs, and numerous studies have focused on enhancing the data quality for MLLMs. For example, many MLLM studies focus on improving the data quality of instruction fine-tuning, *e.g.*, [14, 83, 61, 18, 45, 12, 37, 9, 23]. In addition, some studies, *e.g.*, [63, 38, 87, 75, 58, 8], collect samples through self-instruction, where they use large language models to generate data that conform to text instructions based on a small number of manually annotated samples. Then, language data is usually mixed with multimodal data to train the model [78, 45, 18], improving the instruction-following ability. In addition, preference alignment is often used in scenarios where the model needs to align with specific human preferences. Reinforcement Learning from Human Feedback (RLHF) [48] and Direct Preference Optimization (DPO) [50] are two main techniques for preference alignment.

### 2.2 GUI Agent

Early GUI agents rely on HTML or AXTree data to describe the screen state of user interactions through textual state descriptions. These methods depend on the structured representation of web page elements, enabling agents to locate targets based on tags, attributes, or text content. Raw HTML or AXTree data usually has redundant or noisy structures. Thus, some methods have studied the extraction of effective information from HTML. For example, Mind2Web [15] uses a fine-tuned language model to sort web page elements and extract important ones. WebAgent [24] uses a dedicated HTML-T5 model to generate HTML fragments for specific tasks. AutoWebGLM [29] designs an algorithm to simplify HTML content. Some methods [25, 74] combine visual information. For instance, Set-of-Mark [74] integrates visual and tree tag information for agent decision-making. However, the quality of HTML or AXTree data can limit the application of the aforementioned methods, considering different standards across platforms. In addition, structured data requires meticulous pre-processing, and in some cases, it also relies on additional heuristic methods or trained models to accurately locate and understand key GUI components.

With the rise of multimodal large language models (MLLMs), an increasing number of pure vision-based methods have been proposed. These methods leverage the strong visual capabilities of multimodal language large models and eliminate the need for manually designing data pre-processing for each task, so they have significant advantages over early GUI agents in terms of generalization. Early pure visual GUI agents focus on using MLLMs to process screenshots to understand GUI components, replacing HTML-based GUI component understanding. For example, CogAgent [26] uses MLLMs to process high-resolution GUIs, achieving performance that surpasses HTML-based methods. Auto-GUI [84] unifies GUI grounding into a text-driven grounding task and proposes action chains, using a series of historical actions to enhance agents; COAT [82] further models the thinking process of "what action should be performed" to improve agents in complex tasks.

Furthermore, researchers have found that scaling data is key to enhancing the performance of GUI agents, thus proposing using synthetic data or video data. For instance, Aria-UI [77] proposes an extensible data synthesis pipeline for generating grounding data, which is used to train MLLMs specialized in GUI grounding. OS-Genesis [77] proposes a trajectory synthesis method to retrospectively

generate tasks from the agent-environment interaction, rather than relying on manual supervision or predefined tasks. GUI-explorer [69] automatically generates function-aware task objectives by analyzing GUI structure information, and achieves low-cost generalization of agents through unsupervised analysis of state transitions in the observation-action-result triples. OS-Atlas [67] releases the first multi-platform GUI data synthesis toolkit, supporting the automatic synthesis of cross-platform GUI grounding data while resolving action naming conflicts. GUI-Xplore [53] enables GUI agents to learn from exploration videos.

Recently, researchers have aimed to make GUI agents more similar to human-like agents. For example, UGround [21] executes actions only through human-like keyboard and mouse operations. Aguvis [72] unifies different GUI action spaces and divides training into grounding and planning, enhancing the action ability after the agent has high GUI grounding performance. GUI-Odyssey [43] presents a large-scale cross-application GUI agent training and evaluation dataset, allowing agents to interleavingly use multiple applications to execute tasks. UI-TARS [49] can perform human-like interactions, which achieves accurate perception of GUI elements by collecting a large amount of screenshot data and enhances agent capabilities through various reasoning modes.

The efficiency of GUI agents is crucial for practical deployment, so some research efforts are also dedicated to this. For example, ShowUI [33] constructs a UI correlation graph to identify redundant UI relationships and selects tokens based on the identification results to improve training efficiency. SimpleAgent [7] masks redundant elements in the current environment and uses consistency constraints to guide the cropping of historical tokens.

However, the aforementioned methods mainly focus on using Supervised Fine-Tuning (SFT) to fit manually annotated action trajectories for enhancing the performance of GUI Agents. In fact, numerous studies have shown that the single SFT method limits the model ability for autonomous exploration, which is crucial in GUI Agent tasks. Early works [46, 44] directly optimize models using a 0-1 reward similar to deepseek-R1 [22]. For example, UI-R1 [44] uses rule-based action rewards and the GRPO algorithm to enhance the performance of GUI agents in unknown scenarios. Further optimizations have focused on issues such as reasoning patterns [41, 86], sparse rewards [80, 54], and online learning [54, 65]. InfiGUI-R1 [41] proposes to first use trajectories with explicit reasoning steps for training, and then apply reinforcement learning to enhance the error correction ability. GUI-G1 [86] uses a fast-thinking template to encourage the model to generate answers directly, reducing excessive reasoning during training, and simultaneously designs a difficulty-aware RL objective to better learn hard samples. SE-GUI [80] calculates continuous rewards using the proximity between the predicted box and the GT box, replacing the 0-1 reward to alleviate the sparse reward problem. GUI-G$^2$ [54] transforms the discrete classification of GUI grounding into the continuous optimization of IoU through Gaussian point rewards, coverage rewards, and an adaptive variance mechanism, addressing the sparse reward. Zero-GUI [73] proposes the ZeroGUI online learning framework, which automatically generates tasks and estimates rewards.

### 2.3 API Agent

Another type of agent, distinguished by its mode of interaction with computers or mobile phones, is the API-centric agent. These agents interact with external tools, functions, or services through pre-defined, well-structured programming interfaces. During interaction, relevant API information (*e.g.*, function names) is included in the LLM prompt. The agent receives natural language requests from users and selects the most appropriate API to execute the task.

Microsoft Copilot is a typical example of an enterprise-level API agent. Through interfaces such as the Microsoft 365 Copilot API, it allows developers to integrate capabilities like data analysis and document generation into custom applications. Related research focuses on issues such as automated generation/updating of tools [19, 64, 56, 57, 57], simplification of tools [79], and patterns of tool usage [17, 34]. CLOVA [19] identifies tools that need updating by analyzing human feedback, automatically collects training data, and uses prompt tuning to update the tools. TroVE [64] constructs a verifiable and efficient function toolbox through generating, using, expanding, and periodically streamlining the toolbox. Appworld [56] builds a high-quality execution environment for agents and creates a set of autonomous agent task sets that require agents to generate cross-application interaction code for processing. STE [57] utilizes large models to simulate reasonable environments for tool usage, then enables large models to interact with tools and learn from environment feedback. Kimi K2 [55] proposes a trajectory synthesis scheme for function calls, relying on a vast tool specification

library constructed from real-world tools and synthesized tools. Then, EasyTool [79] converts tool documents into unified and concise instructions to improve tool usage efficiency. Anytool [17] retrieves APIs to handle user needs and proposes a self-reflection mechanism. Hammer 2.1 [34] improves the model sensitivity to irrelevant functions through enhanced datasets and function masking technology. API agents rely on text-based API calls, which are generally highly reliable, and can complete complex tasks with a single call. However, they are limited to pre-defined APIs, have low transparency, and lack the generalization of human-like actions exhibited by GUI agents.

# 3 UItron

UItron is an open-source foundational GUI agent framework designed to advance automated interaction and reasoning across both mobile and PC environments. The system is built upon two key pillars: a robust data engineering pipeline tailored for GUI agent training, and a unified interactive infrastructure that supports scalable data collection and dynamic training. Leveraging these foundations, UItron delivers core capabilities in perception, grounding, and planning, enabling agents to understand complex interfaces, accurately localize tasks, and execute effective action sequences in diverse real-world scenarios.

## 3.1 Problem Formulation

GUI Agent aims to predict the next action in the $n$-th step based on the task instruction, historical actions and visual environment observation (a GUI image). The action is usually represented by the action type and parameters, such as the click(box) and input(content). Formally speaking, we denote the task instruction as $T$, the historical action as $\{a_1, a_2, ..., a_{n-1}\}$, and the visual environment observation as $o_n$. Therefore, the task goal of GUI agent in the $n$-th step can be formulated as:

$$a_n = \mathbf{M}_\theta(T, (a_1, a_2, ..., a_{n-1}), o_n), \tag{1}$$

here $\mathbf{M}$ represents the GUI agent with trainable paratemers $\theta$.

Note that previous work in this area usually utilizes multiple historical images to augment the input with historical information. It significantly increases the length of the input sequence and the computational cost, which is detrimental to the redundant nature of visual information. In fact, we empirically found that omitting historical images does not result in a significant performance degradation in most benchmarks, as historical action information provides sufficient gains. Therefore, to mitigate computational cost, we did not use historical images in this version.

## 3.2 Data Engineering

As shown in Figure 2, we explore systematic data engineering to improve UItron, including perception data, planning data, and distillation data. Besides, we also organize a small amount of general multimodal data that is beneficial to GUI agent, as well as high-quality manual annotation data for Chinese scenarios.

### 3.2.1 Perception Data

**Multi-turn Conversation.** In practical applications, a single complex screenshot can contain hundreds of UI elements, and open-source grounding datasets typically feature multiple objects within one image. To minimize redundant image loading and decrease training costs, we consolidated various instruction/description-answer pairs associated with the same screenshot into unified multi-turn conversations, thereby constructing multi-turn training samples. Utilizing such multi-turn data for training not only lowers computational overhead but also improves the model's ability to comprehend and distinguish between different elements within a UI scene.

**Multi-task Unification.** To enhance the basic understanding ability in GUI scenarios, we collect a large amount of UI-related perception data instead of just considering the traditional agentic trajectory data. We collect a wealth of image-text multimodal pairs from a wide range of PC/mobile application screenshots, covering tasks in GUI scenarios such as OCR, VQA, and Caption. We then integrate these UI-related perception data and traditional agentic trajectory data into the unified format to support training.

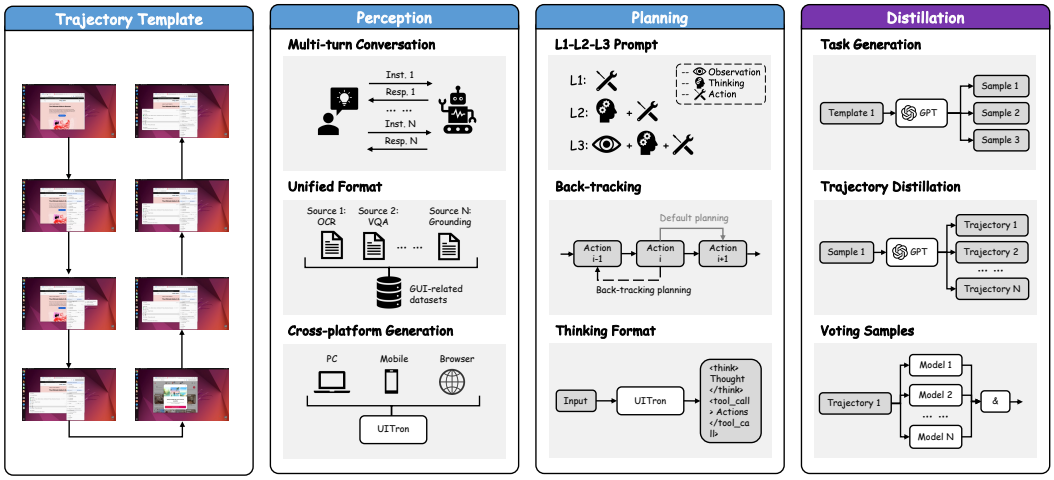


Figure 3: Overall introduction of data engineering.

**Cross-platform Generalization.** Although a substantial amount of grounding data already exists in the GUI agents field, but data collected from different platforms and devices often lacks generalizability, and various tasks employ distinct and isolated data synthesis criteria, making it challenging for these datasets to complement one another. To address the generalization challenge in GUI grounding, we integrated data from diverse sources and synthesis methodologies within the GUI agent domain. By unifying open-source datasets (including Uground [20], Aria-UI [76], Aguvis [71] and OS-Atlas[66]), our approach systematically explores whether diverse synthesis criteria can complement one another, thereby enhancing the generalization capability of agent localization across various scenarios.

### 3.2.2 Planning Data

**L1-L2-L3 Inference.** In addition to the final output action, the execution of a planning task can be enhanced by incorporating multiple levels of perception and reasoning to facilitate action prediction. Following [72], we divide the planning data into several elements including screen observation, reasoning (thinking), action and summarization. The L-1 inference involves only action prediction and summarization, L-2 inference further introduces reasoning, and L-3 inference incorporate screen context to observe and analyze changes in the UI interface. This multi-layered and fine-grained perception strategy enables the model to better adapt to tasks of varying complexity and diverse scenarios. To balance efficiency and accuracy, we utilize L2-level descriptions as historical context prompts for action prediction during inference.

**Back-tracking.** The planning process of a GUI agent can be naturally formulated as a partially observable Markov decision process, in which the model predicts the next action based on historical actions and the current state. However, this approach neglects the model's capacity for reflection and backtracking on previous decisions. Specifically, while the model is aware of its current state, it lacks insight into the sequence of actions that led to that state. Consequently, the model struggles to establish connections between past, present, and future states, which hinders its ability to generate consistent and coherent action predictions. Following [27], we enhance the interaction between GUI agents and their environment by introducing backtracking. Specifically, at each time stept, agent not only predicts the next action based on the current overall goal, but also infers the sequence of historical actions that resulted in the present state.

**Thinking format.** To more precisely distinguish the reasoning process from action prediction during inference optimization, and to facilitate seamless integration with function calls, we employ explicit separators to demarcate different sections of the model output. Specifically, the model's output is structured in the following format:

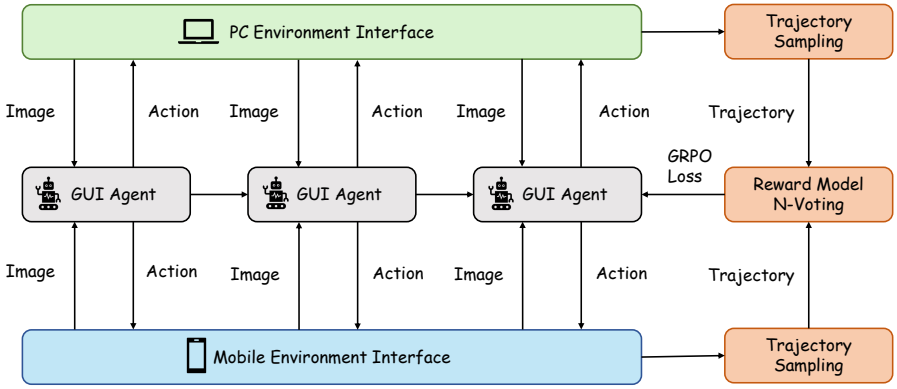


Figure 4: Overall introduction of interactive infrastructure.

```
<observation> Observation </observation>
<think> Thought </think>
<tool_call> Actions </tool_call>
<conclusion> Conclusion </conclusion>
```

### 3.2.3 Distillation Data

Manual labeling of long trajectory data in real scenarios is costly. Therefore, we construct a fully automated trajectory collection process, which includes three stages: (1) Automated task generation guided by real tasks, (2) Automated task execution in simulation environment, and (3) Trajectory result judgment based on VLM voting. After data cleaning and splitting, we finally obtain 500k single-step trajectory data for training.

**Task Generation.** Directly generating tasks with a VLM, without contextual awareness of specific scenarios, often results in unclear or unexecutable tasks. To mitigate this issue, we utilized the initial states of 369 existing tasks in Osworld as prompts to generate additional, related yet distinct tasks using the GPT-4o extension. Furthermore, to prevent task misalignment caused by varying initial states, each generated task is paired with its corresponding initial state.

**Trajectory Distillation.** Building on the multi-domain tasks generated by the VLM, we integrated state-of-the-art GUI agent models into the Osworld simulation environment and implemented a concurrent trajectory distillation pipeline. For each task, the model is allowed up to $n$ attempts, with the reasoning process and specific actions of each step recorded. The complete execution trajectory and task details are then evaluated by a VLM to determine whether the task was successfully completed. Additionally, we tracked the number of attempts for each task: data that succeeded in a single attempt were utilized for supervised fine-tuning (SFT), while data requiring multiple attempts were identified as challenging cases and used for GRPO training.

**Voting Samples.** As illustrated in Figure 2, the trajectory result discriminator is designed according to the following key principles: (1) Visual-Centric Evaluation: GUI agents primarily rely on interactions with the graphical interface. Therefore, changes in the GUI interface serve as the primary indicators of task execution status. (2) Voting Mechanism: Both in supervised fine-tuning (SFT) and GRPO training, even minor variations in training data quality can lead to fluctuations in model prediction accuracy. To ensure robustness, we adopt a stringent voting consensus mechanism, wherein each trajectory is sampled and evaluated multiple times. A trajectory is assigned a positive label only if all evaluations unanimously indicate success. (3) Difficulty Classification: Leveraging the multi-sampling strategy, each task is inferred multiple times by the model. The sample difficulty is then graded based on the number of successful executions across these inferences, enabling targeted application of samples to different training stages.

#### 3.2.4 General Multimodal Data

The general multimodal data serves as a rich repository of fundamental and universal knowledge, intimately interwoven with GUI-related datasets. Recognizing this intrinsic connection, we augment our training regime with image-text pair data sourced from diverse task scenarios such as Optical Character Recognition (OCR), Visual Question Answering (VQA), and Image Captioning. This incorporation aims to bolster the GUI agents' capability to seamlessly comprehend visual content and accurately interpret directive objectives across varied contexts, ultimately fostering a more holistic understanding of task execution dynamics. By leveraging this diverse array of multimodal inputs, we strive to enrich the GUI agents' adaptability and cognitive depth, equipping them to meet increasingly complex interaction demands.

#### 3.2.5 Manual Annotation.

A comprehensive and representative training dataset is essential for developing a robust GUI agent. However, most existing datasets are predominantly focused on English-language applications, leaving a significant gap in coverage for Chinese apps and interfaces. This imbalance limits the agent's ability to generalize and perform effectively in Chinese application scenarios. To address this critical shortcoming, we assembled a dedicated team to manually collect operation trajectories specifically targeting top-tier Chinese mobile applications. Our annotation efforts focused on capturing diverse tasks, complex user interactions, and a wide variety of interface designs unique to the Chinese app ecosystem. Through this targeted data collection, we substantially broadened the scope and diversity of our training data, ensuring that UItron is equipped to excel in both English and Chinese application environments.

### 3.3 Interactive Infrastructure

To facilitate trajectory data collection, online evaluation and RL training, we build an interactive environment connecting both Mobile and PC devices, as shown in Figure 3. Specifically, its significance comes from the following three aspects. First, the Mobile and PC interactive environment provides an automated function for recording screenshots and coordinates, which significantly simplifies the difficulty of manually annotating trajectory data and thus accelerates our efficiency in collecting trajectories for Chinese scenarios. Then, the Mobile and PC interactive environment provides a more realistic evaluation environment, simulating the real interaction process between GUI agents and humans. Finally, the Mobile and PC interactive environment provides the execution results of each action output that facilitate online reinforcement learning for the entire trajectory.

**Mobile Infra.** We build an Android-based cloud real-device environment, which connects multiple genuine Android devices via a server, allowing users to remotely control these smartphones through a web browser. The system is composed of three key components:

- Scrcpy: Responsible for streaming the smartphone's screen content to the browser in real time, similar to live streaming.
- Phone-server: Converts user interactions like clicks and swipes made in the browser into touch commands that the smartphone can understand.
- Device-agent: Serves as the device management center, integrating the functionalities of the previous two components and providing HTTP interfaces for application installation and device information retrieval.

The architecture follows an Agent/Server model, with the server side handling the user interface and device scheduling, while the Agent side manages the specific smartphone devices. Real-time communication is facilitated via WebSocket, and MySQL is used to store device and user data. This solution addresses prevalent issues in mobile application testing, such as insufficient device availability, incomplete model coverage, and the need for remote operations.

**PC Infra.** We utilize the open-source OSWorld environment [70], a scalable real computer setting specifically designed for developing multimodal agents capable of executing a wide array of real computer tasks beyond isolated interfaces and applications. This executable environment allows unrestricted keyboard and mouse control over real computer applications, supporting initial task state configuration, execution-based assessment, and interactive learning across major operating systems

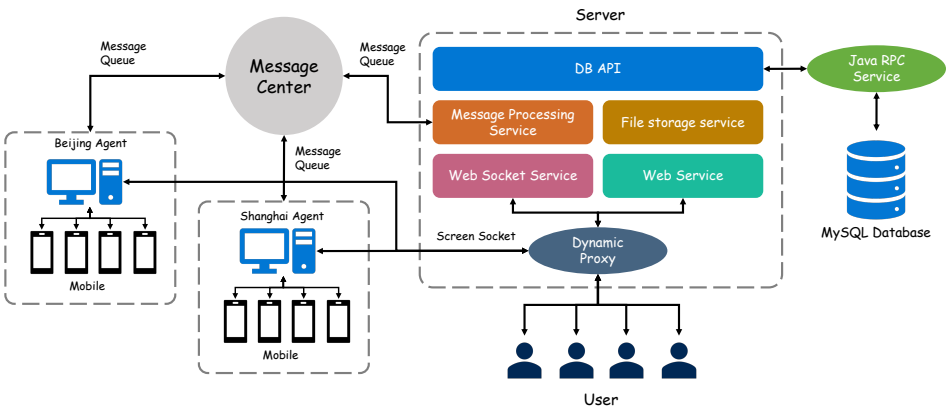


Figure 5: The overall architecture of Mobile infra.

like Ubuntu, Windows, and macOS. Moreover, it provides the capability to evaluate open-ended computer tasks, encompassing activities from image viewing and software feature integration to programming. Hence, OSWorld serves as a unified real computing environment where users can define their agent tasks without the need to construct simulation environments tailored to specific applications or domains.

### 3.4 Training Paradigm

During training, we employ a three-stage training strategy (as shown in Figure 4), in which consists of two SFT stages for perception and planning tasks, as well as a RL stage with curriculum reinforcement learning framework. In the first stage, the perception task focuses on improving the basic understanding ability of the vision-language model in GUI scenarios, such as grounding, captioning, VQA, and OCR. In the second stage, the planning task concentrates on predicting the next action based on historical actions. In the final RL stage, the curriculum reinforcement learning framework aims to improve reasoning and exploration capacity via group relative policy optimization algorithm on trajectory data.

#### 3.4.1 Stage 1: Perception Task

The perceptual abilities of a GUI agent are fundamental for enabling deep understanding and effective interaction with digital interfaces. Modern digital environments are increasingly complex, with user interfaces containing rich visual elements, diverse layouts, and embedded semantic information. Without robust perception capabilities, an agent would struggle to interpret the structure, content, and intent behind various UI components, thereby limiting its effectiveness in real-world applications.

To address this critical challenge, we initially enhance UItron's perceptual ability in the first stage by fine-tuning it on a wide range of GUI perception scenarios. The goal of this fine-tuning is to systematically strengthen UItron's ability to recognize and interpret interface elements, ensuring a deeper and more precise understanding of digital UIs. In particular, we focus on four core perception tasks: grounding, captioning, VQA, and OCR. Grounding enables the agent to accurately localize and associate semantic labels with interface components, establishing a clear mapping between visual regions and their meanings. Captioning facilitates the generation of natural language descriptions for UI layouts and elements, allowing the agent to summarize and communicate interface structures effectively. Furthermore, VQA empowers the agent to answer queries about the interface by integrating both visual and semantic cues, supporting interactive and context-aware understanding. OCR extracts embedded textual information from the interface, ensuring that no detail is overlooked and that all relevant data is accessible for downstream reasoning.

Through mastering these perception tasks, the fine-tuned UItron attains a holistic and nuanced understanding of user interfaces. This comprehensive perceptual foundation not only enhances its ability to interpret complex digital environments, but also lays a solid groundwork for advanced reasoning, planning, and autonomous interaction in subsequent stages.

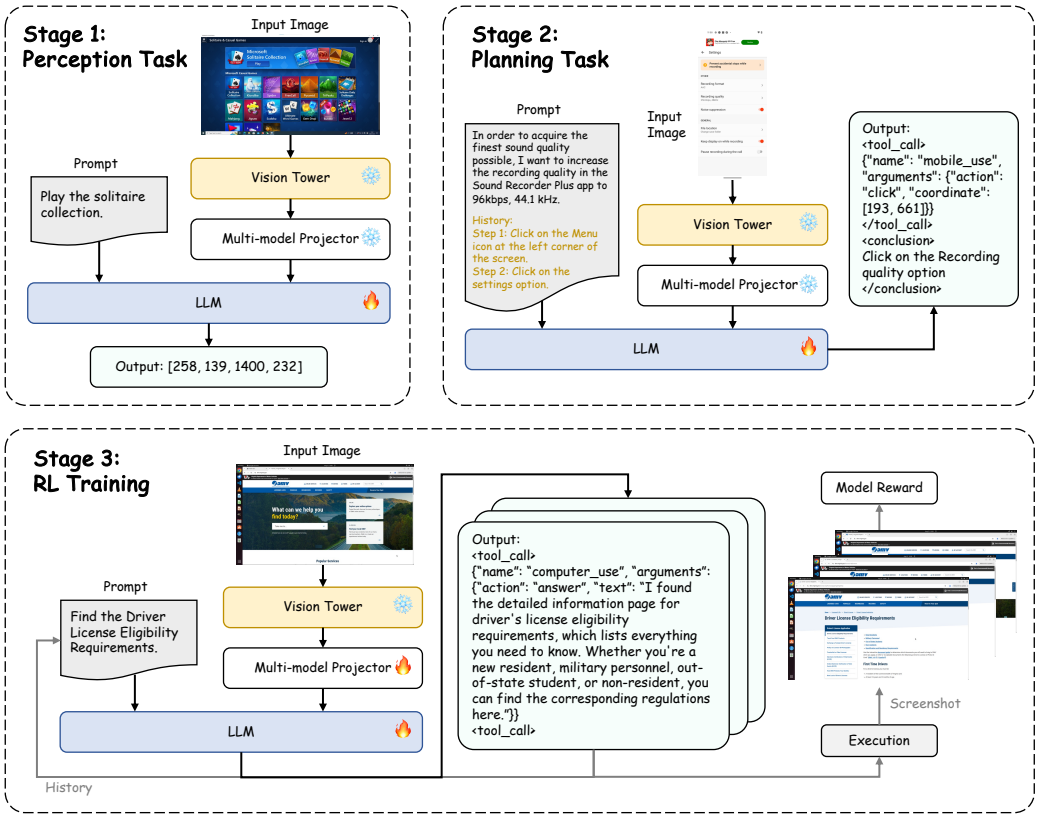


Figure 6: Overall introduction of training paradigm.

### 3.4.2 Stage 2: Planning Task

In this stage, training with planning tasks aims to output the predicted actions defined in Equation (1), then optimize the output using a generative loss in an auto-regressive manner. Effective planning is a key capability that enables a GUI agent to execute purposeful actions and navigate complex digital environments.

The centra idea of planning task is to capture next actions for forward planning and historical actions for backtracking. To this end, we construct two types of training data in the planning task, one for forward planning and the other for backward backtracking. The forward planning follows the message organization approach defined in Equation (1), which inputs historical actions $(a_1, a_2, ..., a_{n-1})$ and the environment observation $o_n$ to output next action $a_n$. In contrast, the backward backtracking follows the message organization approach defined as follows:

$$a_{n-1}, a_n = \mathbf{M}_\theta(T, (a_1, a_2, ..., a_{n-2}), o_{n-1}, o_n), \tag{2}$$

Here the main difference is that the previous action $a_{n-1}$ is not provided in the input, while the agent need to predict $a_{n-1}$.

### 3.4.3 Stage 3: Curriculum Reinforcement Learning

To enhance the reasoning ability, UItron develops a curriculum reinforcement learning framework for performing group relative policy optimization (GRPO) [51]algorithm on trajectory data. It first computes dense rewards from the action steps in the offline environment (simple), and then computes the task-level reward for the trajectory data in the online environment (complex).

**GRPO**

| Benchmarks | Task | Platform | Metric | # Test Samples |
|---|---|---|---|---|
| VisualWebBench | Element Grounding | Web | Prediction Accuracy | 413 |
| | Action Grounding | Web | Prediction Accuracy | 103 |
| | Element OCR | Web | ROUGE-L | 245 |
| | Heading OCR | Web | ROUGE-L | 46 |
| | Web QA | Web | SQuAD-F1 | 314 |
| RefExp | Task Grounding | Web | Accuracy ($IoU \geq 0.5$) | 1000 |
| WidgetCap | Element Caption | Mobile | CIDEr | 1000 |
| WebSRC | WebQA | Web | SQuAD-F1 | 1000 |

Table 1: Details of GUI perception benchmarks. All evaluation data is structured as single-round conversations during experimental.

We adapt the Group Relative Policy Optimization (GRPO) [51] algorithm for RL. For each input $(x, y)$, the policy $\pi_\theta$ samples a group of $G$ candidate responses $\{o_i\}_{i=1}^{G}$.

$$\begin{aligned}\mathcal{J}_{GRPO}(\theta) =& \mathbb{E}_{(x,y)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\min\Bigg(\frac{\pi_\theta(o_i \mid x)}{\pi_{\theta_{\text{old}}}(o_i \mid x)}A_i, \\ & \text{clip}\left(\frac{\pi_\theta(o_i \mid x)}{\pi_{\theta_{\text{old}}}(o_i \mid x)}, 1-\varepsilon, 1+\varepsilon\right)A_i\Bigg) - \beta\mathbb{D}_{KL}(\pi_\theta\|\pi_{\text{SFT}})\Bigg]\end{aligned} \tag{3}$$

where $\varepsilon$ and $\beta$ are hyperparameters, and $\pi_{\text{SFT}}$, $\pi_\theta$, and $\pi_{\theta_{\text{old}}}$ are the model after SFT, the optimized model and the old policy model. The group-normalized advantage for the $i$-th response is:

$$A_i = \frac{r_i - \text{mean}\left(\{r_1, r_2, \cdots, r_G\}\right)}{\text{std}\left(\{r_1, r_2, \cdots, r_G\}\right)} \tag{4}$$

**Offline RL** Since successful trajectories of GUI agents in online environments are usually rare, the offline RL collects rewards for each action step in the offline environment to avoid sparse rewards. For each action prediction, it generates multiple candidate actions for the same input and calculates the GRPO loss to improve reasoning and exploration capabilities.

**Online RL** The online RL is built on the interactive infrastructure shown in Figure 3. It collects rewards for the entire trajectory through the rollout dialogues in an online environment. For each task, online RL allows the model to freely explore all possible action plans until it reaches the maximum number of steps or generates an end signal. To this end, online RL utilizes the advanced vision-language model as a scoring model to evaluate whether the task is completed based on the entire trajectory and task, and outputs a reward signal of 0 or 1. To improve the credibility of rewards in the RL process, we incorporate multiple scoring models from different vision-language models. Besides, we also strictly filter the trajectories that are predicted correctly by multiple scoring models simultaneously. Finally, the online RL calculates the trajectory-level GRPO loss based on multiple sampled trajectories, thereby improving the exploration ability in the online environment.

**Summary** Finally, we produce two versions named UItron and UItron-RL, both of which are based on the Qwen25-VL model structure, but with different parameter weights. The former is obtained after training in stages 1 and 2, while the latter is obtained after reinforcement training in stage 3. In the experiments, we report the results of uitron-RL in all online environments, and report the results of UItron in other offline scenarios.

# 4 Experiments

We carry out extensive experiments covering scenarios including GUI perception, grounding, offline planning, and online planning. In particular, we also built our own Chinese scenario evaluation and conduct experiments to explore the improvement of Chinese capabilities.

## 4.1 Evaluation of GUI Perception

**VisualWebBench.** We evaluate our model's screen perception capabilities on VisualWebBench [40], a comprehensive benchmark containing multiple website-based tasks. For the Grounding Tasks, we

| Benchmarks | Platform | #Test Episodes | #Test Samples | History |
|---|---|---|---|---|
| ScreenSpot | Mobile&Desktop&Web | - | 1272 | |
| ScreenSpot-V2 | Mobile&Desktop&Web | - | 1272 | |
| AndroidContorl-Low | Mobile | 1,000 | 6,585 | |
| AndroidContorl-High | Mobile | 1,000 | 6,585 | ✓ |
| GUI-Odyssey-Random | Mobile | 1,933 | 29,426 | ✓ |
| GUI-Odyssey-App | Mobile | 1,139 | 17,455 | ✓ |
| GUI-Odyssey-Device | Mobile | 1,262 | 18,967 | ✓ |
| GUI-Odyssey-Task | Mobile | 1,016 | 17,920 | ✓ |
| OSWorld | Desktop&Web | 369 | - | ✓ |
| AndroidWorld | Mobile | 116 | - | ✓ |
| MobileMiniWob | Web | 92 | - | ✓ |

Table 2: Details of the agentic benchmarks. "Test episodes" refers to the number of trajectory data used for evaluation, while "Test samples" represents the total number of individual step data contained within all trajectories. "History" indicates whether the historical information of previous actions is provided in the model input.

| Method | Element Grounding | Action Grounding | Element OCR | Heading OCR | Web QA |
|---|---|---|---|---|---|
| GPT-4o | 79.91 | 86.41 | 79.42 | 64.57 | 77.44 |
| Qwen2.5-VL | 82.81 | 77.67 | 95.72 | 66.82 | 80.23 |
| MutiUI | 75.92 | 36.66 | - | - | - |
| UI-TARS | 96.13 | 92.23 | 92.53 | 70.63 | 79.7 |
| UItron-7B | 94.67 | 94.07 | 95.36 | 65.50 | 77.70 |
| UItron-72B | 96.37 | 94.17 | 95.56 | 72.15 | 80.49 |

Table 3: Comparative results on VisualWebBench [40].

measure prediction accuracy by requiring the agent to select correct answers from set of masks (SoM) on screenshots.

**Complicated Perceptual Benchmarks.** To evaluate the model's ability to comprehend abstract instructions, we follow [39] by assessing visual grounding of natural language-described elements through RefExp [4] and testing the reverse task of element captioning on WidgetCap [31]. We additionally evaluate on general Web QA task of WebSRC [10], requiring textual and structural understanding of GUI elements, for further assessing comprehensive perceptual capabilities.

**Baseline Models.** We compare our UItron with SOTA models in both understanding and GUI operation task. Among general VLLMs, we use GPT-4o [28] and Qwen2.5-VL [6] as our baseline for their powerful understanding capabilities in general task understanding; among GUI-related VLLMs, we compare our UItron with MultiUI [39] and UI-TARS [49], the former is specialized in GUI understanding while the later one is the SOTA model in GUI tasks.

| Method | Task Grounding (RefExp) | Element Caption (WidgetCap) | Web QA (WebSRC) |
|---|---|---|---|
| GPT-4o | – | 61.02 | 78.60 |
| Qwen2.5-VL | 6.55 | 58.45 | 91.2 |
| MutiUI | 43.56 | 72.73 | 82.9 |
| UI-TARS | – | 82.48 | 93.6 |
| UItron-7B | 51.40 | 77.55 | 89.7 |
| UItron-72B | 59.20 | 87.44 | 93.24 |

Table 4: Comparative results on RefExp [4],WidgetCap [31] and WebSRC [10]. Note that GPT-4o and UI-TARS are failed to evaluate due to the invalid output format in task grounding (RefExp).

As shown in Tables 3 and 4, our UItron demonstrates superior performance on perceptual tasks, establishing crucial groundwork for subsequent planning and reasoning capabilities essential for GUI task execution. This effectiveness stems from the limited understanding data employed in both training stages 1 and 2, which not only mitigates the spurious forgetting issue [85] that degrades baseline VLLM's original comprehension, but also enhances GUI-specific understanding. This results indicate that maintaining the generalist model's capabilities relevant to the the downstream task while developing specialized skills is critical for creating effective specialist agents. Details of the benchamrks are listed in Table 1.

| Method | | Mobile | | Desktop | | Web | | Avg |
|---|---|---|---|---|---|---|---|---|
| | | Text | Icon/Widget | Text | Icon/Widget | Text | Icon/Widget | |
| *Agent Framework* | | | | | | | | |
| GPT-4o | SeeClick | 81.0 | 59.6 | 69.6 | 33.6 | 43.9 | 26.2 | 52.3 |
| | UGround | 93.4 | 76.9 | 92.8 | 67.9 | 88.7 | 68.9 | 81.4 |
| *Agent Model* | | | | | | | | |
| GPT-4o | | 20.2 | 24.9 | 21.1 | 23.6 | 12.2 | 7.8 | 18.3 |
| Claude | | - | - | - | - | - | - | 83.0 |
| Gemini 2.0 | | - | - | - | - | - | - | 84.0 |
| Qwen2.5-VL-7B | | - | - | - | - | - | - | 84.7 |
| UGround | | 82.8 | 60.3 | 82.5 | 63.6 | 80.4 | 70.4 | 73.3 |
| Aria-UI | | 92.3 | 73.8 | 93.3 | 64.3 | 86.5 | 76.2 | 82.4 |
| OS-Atlas | | 93.0 | 72.9 | 91.8 | 62.9 | 90.9 | 74.3 | 82.5 |
| AGUVIS-7B | | 95.6 | 77.7 | 93.8 | 67.1 | 88.3 | 75.2 | 84.4 |
| AGUVIS-72B | | 94.5 | 85.5 | 95.4 | 77.9 | 91.3 | 85.9 | 89.2 |
| UI-TARS-7B | | 94.5 | 85.2 | 95.9 | 85.7 | 90.0 | 83.5 | 89.5 |
| UI-TARS-72B | | 94.9 | 82.5 | 89.7 | 88.6 | 88.7 | 85.0 | 88.4 |
| UItron-7B | | 94.1 | 83.8 | 94.8 | 73.6 | 92.2 | 81.1 | 87.7 |
| UItron-72B | | 94.5 | 88.2 | 96.9 | 79.2 | 93.0 | 85.4 | 90.3 |

Table 5: Comparison of different baseline methods on ScreenSpot [13].

| Method | | Mobile | | Desktop | | Web | | Avg |
|---|---|---|---|---|---|---|---|---|
| | | Text | Icon/Widget | Text | Icon/Widget | Text | Icon/Widget | |
| *Agent Framework* | | | | | | | | |
| GPT-4o | SeeClick | 85.2 | 58.8 | 79.9 | 37.1 | 72.7 | 30.1 | 63.6 |
| | OS-Atlas-4B | 95.5 | 75.8 | 79.4 | 49.3 | 90.2 | 66.5 | 79.1 |
| | OS-Atlas-7B | 96.2 | 83.4 | 89.7 | 69.3 | 94.0 | 79.8 | 87.1 |
| *Agent Model* | | | | | | | | |
| SeeClick | | 78.4 | 50.7 | 70.1 | 29.3 | 55.2 | 32.5 | 55.1 |
| OS-Atlas-4B | | 87.2 | 59.7 | 72.7 | 46.4 | 85.9 | 63.1 | 71.9 |
| OS-Atlas-7B | | 95.2 | 75.8 | 90.7 | 63.6 | 90.6 | 77.3 | 84.1 |
| UI-TARS-7B | | 96.9 | 89.1 | 95.4 | 85.0 | 93.6 | 85.2 | 91.6 |
| UI-TARS-72B | | 94.8 | 86.3 | 91.2 | 87.9 | 91.5 | 87.7 | 90.3 |
| UItron-7B | | 96.9 | 88.2 | 92.8 | 71.4 | 91.5 | 80.8 | 88.4 |
| UItron-72B | | 95.5 | 90.5 | 99.0 | 80.0 | 94.0 | 87.7 | 92.0 |

Table 6: Comparison of different baseline methods on ScreenSpot-V2 [13].

### 4.2 Evaluation of GUI Grounding

**ScreenSpot.** We use ScreenSpot [13] to assess the fundamental GUI-understanding and element-grounding accuracy of GUI-agent models. The ScreenSpot benchmark comprises more than 600 screenshots and 1,200 instructions, spanning multiple platforms—iOS, Android, macOS, Windows, and web pages. We report separate results for Text and Icon/Widget elements on the Mobile, Desktop, and Web splits of ScreenSpot, together with the micro accuracy aggregated across all platforms.

| Method | | AndroidControl-Low† | | | AndroidControl-High† | | | GUI Odyssey | | | Avg |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Type | Grounding | SR | Type | Grounding | SR | Type | Grounding | SR | |
| *Agent Framework* | | | | | | | | | | | |
| GPT-4o | SeeClick | - | - | 52.8 | - | - | 41.8 | - | - | - | - |
| | UGround | - | - | 62.4 | - | - | 48.4 | - | - | - | - |
| *Agent Model* | | | | | | | | | | | |
| Claude | | 74.3 | 0.0 | 19.4 | 63.7 | 0.0 | 12.5 | 60.9 | 0.0 | 3.1 | 26.0 |
| GPT-4o | | 74.3 | 0.0 | 19.4 | 66.3 | 0.0 | 20.8 | 34.3 | 0.0 | 3.3 | 24.3 |
| InternVL-2 | | 90.9 | 84.1 | 80.1 | 84.1 | 72.7 | 66.7 | 82.1 | 55.5 | 51.5 | 74.2 |
| SeeClick | | 93.0 | 73.4 | 75.0 | 82.9 | 62.9 | 59.1 | 71.0 | 52.4 | 53.9 | 69.3 |
| Aria-UI | | - | 87.7 | 67.3 | - | 43.2 | 10.2 | - | 86.8 | 36.5 | - |
| OS-Atlas | | 93.6 | 88.0 | 85.2 | 85.2 | 78.5 | 71.2 | 84.5 | 67.8 | 62.0 | 79.6 |
| AGUVIS | | - | - | 80.5 | - | - | 61.5 | - | - | - | - |
| UI-TARS-7B | | 98.0 | 89.3 | 90.8 | 83.7 | 80.5 | 72.5 | 94.6 | 90.1 | 87.0 | 87.4 |
| UI-TARS-72B | | 98.1 | 89.9 | 91.3 | 85.2 | 81.5 | 74.7 | 95.4 | 91.4 | 88.6 | 88.5 |
| UItron-7B | | 96.5 | 90.6 | 90.1 | 91.4 | 77.2 | 79.0 | 95.3 | 85.7 | 84.8 | 87.8 |
| UItron-72B | | 96.4 | 93.8 | 92.0 | 93.9 | 85.3 | 86.1 | 94.4 | 86.3 | 86.1 | 90.5 |

Table 7: Comparative results on AndroidControl-Low [30], AndroidControl-High [30] and GUI-Odyssey [42]. † indicates that there is no unified evaluation set. Thus we follow UI-TARS [49] to sample 1000 trajectories for evaluation, ensuring that all test data do not appear in any of the training data.

**ScreenSpot-V2.** Similar with ScreenSpot [13], we also employ ScreenSpot-V2 [67] for evaluation, which is a GUI benchmark that advances from basic recognition to cross-modal reasoning. This enhanced version better reflects real-world complexity through optimized annotations, expanded task types, and improved data diversity. The benchmark contains 1,272 instructional samples paired with 756 images, drawing from data sources similar to ScreenSpot.

The experimental results in Table 4.1 demonstrate that UItron exhibits impressive leading GUI grounding performance across all platforms. This advantage is primarily attributed to UItron's adoption of data engineering specifically tailored for GUI agents, which provides high-quality and well-defined datasets for model training. Furthermore, the parameter scaling experiments of UItron indicate that, with sufficient and high-confidence training data, the model's grounding capability is further enhanced as its scale increases. Compared with state-of-the-art model (*i.e.*, UI-TARS) that additionally utilize internal data, UItron-72B relies solely on open-source data, achieves a 2.1% improvement in micro grounding accuracy.

### 4.3 Evaluation of Offline Planning

**AndroidControl.** AndroidControl [30] is a benchmark for evaluating the planning and action-execution capabilities of GUI agents on Android devices. It contains 15,283 episodes of everyday tasks across 833 distinct applications, making it the most diverse UI-control dataset to date. Following standard practice, we report results under two settings. AndroidControl-Low: At every step the agent receives a screenshot together with a natural-language description of the required action and must predict both the action type and its exact parameters. AndroidControl-High: Only the high-level task goal and the current screenshot are provided at each step. The agent must autonomously plan the entire procedure and output the correct action together with its parameters. Following OS-Atlas [67], we reserve 1,000 episodes as an out-of-domain evaluation set and report the action-type accuracy, grounding accuracy, and average step success rate.

General-purpose LLMs such as GPT-4o and Claude demonstrate reasonable action-type accuracy in the Low setting, but their grounding accuracy is essentially zero and their step success rates are very low, indicating a lack of fine-grained perception and UI localization capability. In contrast, specialized GUI models like SeeClick, Aria-UI, OS-Atlas, AGUVIS, and UI-TARS exhibit a clear advantage, achieving substantially higher scores across all metrics. This demonstrates the superiority of dedicated GUI models in both accurately identifying UI elements and executing precise actions. Notably, our model, UItron, surpasses all others: UItron-72B achieves the highest grounding and step success rates in both Low and High settings, showcasing exceptional performance not only in guided UI action execution but also in autonomous planning. This underscores the critical importance of our

model's unified approach to perception, grounding, and planning, enabling robust and generalizable UI control.

**GUI-Odyssey.** GUI-Odyssey [42] is used for evaluating cross-app navigation agents, surpassing the limitation of other benchmarks that are restricted to a single app. It consists of 7,735 episodes, six types of cross-app tasks, 201 apps, and 1.4k app combinations. GUI-Odyssey-Random/Task/Device/App are four different test subsets, with statistics shown in Table 2. It aims to assess the generalization ability of autonomous GUI agents across different applications, tasks, and device setups. Following OS-Atlas [67], we report the macro average performance across these subsets.

The challenge of cross-app navigation exposes even greater limitations in general-purpose LLMs, with GPT-4o and Claude displaying poor performance in both grounding and step success rate, and action-type accuracy dropping further compared to single-app scenarios. Specialized GUI models again demonstrate their superiority, with SeeClick, Aria-UI, and OS-Atlas showing solid results, and UI-TARS achieving state-of-the-art performance. Notably, UItron remains highly competitive, achieving results close to those of UI-TARS in most metrics. While UItron may perform slightly below UI-TARS on certain cross-app tasks, it consistently demonstrates top-tier results when considering both AndroidControl and GUI-Odyssey benchmarks together, highlighting its overall superiority in comprehensive UI understanding and control. UItron's strong performance across diverse tasks and app combinations underscores its robust generalization ability and reliability, making it one of the most effective agents for complex, real-world UI navigation tasks.

### 4.4 Evaluation of Online Planning

**OSWorld.** We use OSWorld [70] to evaluate the performance of GUI agent models as online agents on personal computer (PC) platforms. OSWorld is a real computer environment that supports multimodal agents in task setup and execution evaluation across multiple operating systems. It includes a benchmark of 369 tasks covering real-world web and desktop applications, OS file I/O, and workflows across applications.

**Baselines.** We compare our method with two types of agentic methods, namely GUI agent and computer-use agent. The GUI agent is a typical method that considers both Mobile and PC scenarios, while the compute-use agents are some recent methods that are specifically designed for PC scenario. For GUI agent, we select several advanced baselines including Augvis-72B [72], UI-TARS-72B [49], UI-TARS-1.5-7B [49] (72B version is closed source). We also compare with Qwen2.5-VL-72B [6] to demonstrate the improvement gains via several training stages. For compute-use agent, we select several advanced baselines including OpenAI CUA [47], Claude CUA [3] and OpenCUA [62]. All methods adopt the same setting of maximum length of 15 steps for fair comparison.

| Model | OSWorld |
|---|---|
| **Compute-Use Agent (CUA)** | |
| OpenAI CUA | 26.0 |
| Claude CUA | 31.2 |
| OpenCUA-32B | 29.7 |
| **GUI Agent** | |
| Qwen2.5-VL-72B | 4.4 |
| Augvis-72B | 10.3 |
| UI-TARS-7B | 18.7 |
| UI-TARS-72B | 22.7 |
| UI-TARS-1.5-7B* | 23.3 |
| UItron-72B | 25.2 |

Table 8: Task Success Rates (SR) on OSWorld [70]. We report results on their official verified environment (*i.e.*, OSWorld-verified) that fix several issues. * denotes our reproduction within the same environment.

**Results.** Table 8 reports the comparative results of UItron and other baseline methods. From the results, we observe that specialized CUA agents generally outperform GUI agents, primarily due to their more singular scenarios and objectives. We can also see that Uitron achieves competitive performance in GUI agents, with only a small gap compared to the state-of-the-art UI-Tars-1.5 method. In addition, the experimental results also show that existing vision-language models such as Qwen25-VL suffers from poor performance, which can be greatly improved through a large amount of targeted training in GUI scenarios.

### 4.5 Evaluation of Chinese Scenario

**Evaluation Data** We evaluate the effectiveness of our method in both offline and online environments. To support comprehensive evaluation, we constructed test data and an Android cloud environment. We manually annotate 545 trajectory steps from 109 universal tasks across several apps, and verify that these test tasks did not overlap with the training tasks. Considering that some tasks in the online environment have some app automatic login risks and failures, we retain 86 tasks that can be completed in the online environment.

**Evaluation Metrics** We design different evaluation metrics for GUI agents in offline and online environments. For evaluation in offline environment, in which each predicted action corresponding to a ground-truth action, we directly calculate accuracy to evaluate the single-step success rate (*i.e.*, Step SR) and task success rate (*i.e.*, Task SR). A task is deemed successful when all execution steps exactly align with the ground-truth action sequence. For evaluation in online environment, the GUI agent freely explores all possible actions according to finish the task without any ground-truth action. Therefore, we evaluate whether the task is completed accurately based on the complete execution trajectory. We leverage an advanced visual-language model (*i.e.*, GPT-4o [28]) to determine whether the task is completed. The result is 1 for completion and 0 for incomplete.

**Offline Results** Table 9 reports the Step SR and Task SR results of UItron and baseline methods. From the results, we can see that both UItron-7B and UItron-72B significantly outperform the baseline methods in all evaluation metrics, demonstrating their superiority in Chinese scenarios. Interestingly, for the Step SR and Task SR indicators, the results indicate that they have a positive correlation, but the difference in Task SR is significantly larger, which is probably because Task SR reflects the more rigorous accumulation of Step SR. Therefore, we can see from the results that different methods are relatively close in terms of Step SR, but have significant differences in terms of Task SR. The advanced performance of UItron primarily stems from learning page organization and interaction logic through extensive data from Chinese scenarios, which exhibit significant differences compared to traditional English contexts.

| Method | Step SR | Task SR |
|---|---|---|
| **Offline Environment** | | |
| UI-TARS-7B | 75.1 | 22.4 |
| UI-TARS-72B | 80.5 | 32.8 |
| UI-TARS-1.5-7B | 77.4 | 29.3 |
| UItron-7B | 82.7 | 40.5 |
| UItron-72B | 84.1 | 47.4 |

Table 9: Offline evaluation results on top-tier Chinese mobile Apps.

**Online Results** Table 10 reports the Task SR results of UItron and baseline methods. The results indicate that UItron outperforms the baseline model with a significant performance advantage, verifying its better interaction and exploration capabilities in online environments. Another noteworthy phenomenon is that the online evaluation results of the same model consistently surpass its offline evaluation results, a trend often overlooked in previous research due to the lack of comparable offline and online tasks. The explanation for this phenomenon lies in the nature of the online environment, which offers GUI agents ample space to explore and recover from errors with relaxed constraints. In this setting, certain erroneous steps can be rectified by returning to the original step. Conversely, in offline evaluations, any failed step inevitably results in task failure.

| Method | Task SR |
|---|---|
| **Online Environment** | |
| UI-TARS-1.5-7B | 38.9 |
| UItron-7B | 44.4 |
| UItron-72B | 54.1 |

Table 10: Online evaluation results on top-tier Chinese mobile Apps.

## 5 Conclusion and Future Work

This paper presents UItron, a pioneering open-source foundational model designed to enhance the capabilities of GUI agents in executing complex tasks across digital environments such as PCs and Mobile devices. UItron conduct sufficient investigation of data engineering and interactive infrastructure to handle the scarcity of annotated trajectory data. It systematically compares various data strategies to improve the training effectiveness. UItron adopts a typical training paradigm of

GUI grounding and planning, and then develops a curriculum reinforcement learning method that improves complex reasoning and exploration in the online environment. In particular, UItron also emphasizes the importance of Chinese interaction capabilities in practical GUI agent deployment. Through comprehensive annotation of over one million action steps from leading Chinese apps, UItron achieves superior results in realistic offline and online evaluation frameworks, bringing GUI agents closer to practical deployment. Experimental results demonstrate that UItron achieves superior performance in benchmarks of GUI perception, task localization and planning, as well as a significant advance in Chinese application scenarios.

In summary, UItron offers an open-source foundation that facilitates the future development of GUI agents. In our future work, we will investigate the intrinsic thinking patterns underlying the interpretive action behaviors of GUI agents, as we have observed frequent ambiguities and inconsistencies between thinking and action outputs in our method. Furthermore, we will study multi-agent collaboration strategies to systematically explore capabilities such as reflection, backtrace, and correction, considering that current single-agent methods often struggle to simultaneously handle both visual and textual aspects. Lastly, we plan to build the unified agentic infrastructure and reinforcement learning environment that integrate capabilities such as coding, tool-use and function-call, spanning both the 2D digital world and the 3D physical realm.

| Environment | Action Space |
|---|---|
| Web | {"name": "computer_use", "arguments": {"action": "key", "keys": ["ctrl", "a"]}}<br>{"name": "computer_use", "arguments": {"action": "left_click", "coordinate": [x, y]}}<br>{"name": "computer_use", "arguments": {"action": "type", "text": "text"}}<br>{"name": "computer_use", "arguments": {"action": "answer", "text": "text"}}<br>{"name": "computer_use", "arguments": {"action": "terminate", "status": ["success"]}}<br>{"name": "computer_use", "arguments": {"action": "wait", "time": "time"}}<br>(Total 15 action types...) |
| Mobile | {"name": "mobile_use", "arguments": {"action": "system_button", "button": "enter"}}<br>{"name": "mobile_use", "arguments": {"action": "click", "coordinate": [x, y]}}<br>{"name": "mobile_use", "arguments": {"action": "type", "text": "text"}}<br>{"name": "mobile_use", "arguments": {"action": "swipe", "coordinate": [x, y], "coordinate2": [x, y]}}<br>{"name": "mobile_use", "arguments": {"action": "type", "text": "text"}}<br>{"name": "mobile_use", "arguments": {"action": "status", "button": "success"}}<br>{"name": "mobile_use", "arguments": {"action": "wait", "time": "time"}}<br>(Total 11 action types...) |

Table 11: Action space specifications for Web and Mobile environments

## References


[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[3] Anthropic. Introducing claude 4.5. *URL https://www.anthropic.com/news/claude-4*, 2025.

[4] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*, 2021.

[5] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 1(2):3, 2023.

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

[7] Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. Less is more: Empowering gui agent with context-aware simplification. In *International Conference on Computer Vision*, 2025.

[8] Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. Allava: Harnessing gpt4v-synthesized data for lite vision-language models. *arXiv preprint arXiv:2402.11684*, 2024.

[9] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions. In *European Conference on Computer Vision*, pages 370–387. Springer, 2024.

[10] Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. Websrc: A dataset for web-based structural reading comprehension. *arXiv preprint arXiv:2101.09465*, 2021.

[11] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *Science China Information Sciences*, 67(12):220101, 2024.

[12] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024.

[13] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*, 2024.

[14] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. *Advances in neural information processing systems*, 36:49250–49267, 2023.

[15] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

[16] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

[17] Yu Du, Fangyun Wei, and Hongyang Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. In *International Conference on Machine Learning*, pages 11812–11829. PMLR, 2024.

[18] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. Llama-adapter v2: Parameter-efficient visual instruction model. *arXiv preprint arXiv:2304.15010*, 2023.

[19] Zhi Gao, Yuntao Du, Xintong Zhang, Xiaojian Ma, Wenjuan Han, Song-Chun Zhu, and Qing Li. Clova: A closed-loop visual assistant with tool usage and update. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13258–13268, 2024.

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

[21] Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[23] Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, Haoqi Fan, Jian Wang, Jianyu Jiang, Jiawei Wang, et al. Seed1. 5-vl technical report. *arXiv preprint arXiv:2505.07062*, 2025.

[24] Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *ICLR*, 2024.

[25] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, 2024.

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

[27] Jing Huang, Zhixiong Zeng, Wenkang Han, Yufeng Zhong, Liming Zheng, Shuai Fu, Jingyuan Chen, and Lin Ma. Scaletrack: Scaling and back-tracking automated gui agents. *arXiv preprint arXiv:2505.00416*, 2025.

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

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

[30] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024.

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

[32] Zhang Li, Biao Yang, Qiang Liu, Zhiyin Ma, Shuo Zhang, Jingxu Yang, Yabo Sun, Yuliang Liu, and Xiang Bai. Monkey: Image resolution and text label are important things for large multi-modal models. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26763–26773, 2024.

[33] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508, 2025.

[34] Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, et al. Robust function-calling for on-device language model via function masking. In *The Thirteenth International Conference on Learning Representations*, 2025.

[35] Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, et al. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. *arXiv preprint arXiv:2311.07575*, 2023.

[36] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[37] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024.

[38] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[39] Junpeng Liu, Tianyue Ou, Yifan Song, Yuxiao Qu, Wai Lam, Chenyan Xiong, Wenhu Chen, Graham Neubig, and Xiang Yue. Harnessing webpage uis for text-rich visual understanding. *arXiv preprint arXiv:2410.13824*, 2024.

[40] Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*, 2024.

[41] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025.

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

[43] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *International Conference on Computer Vision*, 2025.

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

[45] Gen Luo, Yiyi Zhou, Tianhe Ren, Shengxin Chen, Xiaoshuai Sun, and Rongrong Ji. Cheap and quick: Efficient vision-language instruction tuning for large language models. *Advances in Neural Information Processing Systems*, 36:29615–29627, 2023.

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

[47] OpenAI. Operator. *URL https://openai.com/research/operator*, 2025.

[48] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[49] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025.

[50] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[51] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[52] Yixuan Su, Tian Lan, Huayang Li, Jialu Xu, Yan Wang, and Deng Cai. Pandagpt: One model to instruction-follow them all. *arXiv preprint arXiv:2305.16355*, 2023.

[53] Yuchen Sun, Shanhui Zhao, Tao Yu, Hao Wen, Samith Va, Mengwei Xu, Yuanchun Li, and Chongyang Zhang. Gui-xplore: Empowering generalizable gui agents with one exploration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19477–19486, 2025.

[54] Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g $^2$: Gaussian reward modeling for gui grounding. *arXiv preprint arXiv:2507.15846*, 2025.

[55] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

[56] Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076, 2024.

[57] Boshi Wang, Hao Fang, Jason Eisner, Benjamin Van Durme, and Yu Su. Llms in the imaginarium: Tool learning through simulated trial and error. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10583–10604, 2024.

[58] Junke Wang, Lingchen Meng, Zejia Weng, Bo He, Zuxuan Wu, and Yu-Gang Jiang. To see is to believe: Prompting gpt-4v for better visual instruction tuning. *arXiv preprint arXiv:2311.07574*, 2023.

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

[60] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499, 2024.

[61] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36:61501–61513, 2023.

[62] Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

[63] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[64] Zora Zhiruo Wang, Graham Neubig, and Daniel Fried. Trove: inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Proceedings of the 41st International Conference on Machine Learning*, pages 51177–51191, 2024.

[65] Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

[66] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218*, 2024.

[67] Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: Foundation action model for generalist gui agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

[68] Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024.

[69] Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. Gui-explorer: Autonomous exploration and mining of transition-aware knowledge for gui agent. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

[70] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

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

[72] Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. In *International Conference on Machine Learning*, 2025.

[73] Chenyu Yang, Shiqian Su, Shi Liu, Xuan Dong, Yue Yu, Weijie Su, Xuehui Wang, Zhaoyang Liu, Jinguo Zhu, Hao Li, et al. Zerogui: Automating online gui learning at zero human cost. *arXiv preprint arXiv:2505.23762*, 2025.

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

[75] Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007, 2023.

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

[77] Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-UI: Visual grounding for GUI instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22418–22433, 2025.

[78] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023.

[79] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. Easytool: Enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972, 2025.

[80] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

[81] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

[82] Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for gui agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12016–12031, 2024.

[83] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023.

[84] Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3132–3149, 2024.

[85] Junhao Zheng, Xidi Cai, Shengjie Qiu, and Qianli Ma. Spurious forgetting in continual learning of language models. *arXiv preprint arXiv:2501.13453*, 2025.

[86] Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. *arXiv preprint arXiv:2505.15810*, 2025.

[87] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.