UI-Venus-1.5 Technical Report

Venus Team, Ant Group

GUI agents have emerged as a powerful paradigm for automating interactions in digital environments, yet achieving both broad generality and consistently strong task performance remains challenging. In this report, we present **UI-Venus-1.5**, a unified, end-to-end GUI Agent designed for robust real-world applications. The proposed model family comprises two dense variants (2B and 8B) and one mixture-of-experts variant (30B-A3B) to meet various downstream application scenarios. Compared to our previous version, UI-Venus-1.5 introduces three key technical advances: (1) a comprehensive Mid-Training stage leveraging 10 billion tokens across 30+ datasets to establish foundational GUI semantics; (2) Online Reinforcement Learning with full-trajectory rollouts, aligning training objectives with long-horizon, dynamic navigation in large-scale environments; and (3) a single unified GUI Agent constructed via Model Merging, which synthesizes domain-specific models (grounding, web, and mobile) into one cohesive checkpoint. Extensive evaluations demonstrate that UI-Venus-1.5 establishes new state-of-the-art performance on benchmarks such as ScreenSpot-Pro (**69.6%**), VenusBench-GD (**75.0%**), and AndroidWorld (**77.6%**), significantly outperforming previous strong baselines. In addition, UI-Venus-1.5 demonstrates robust navigation capabilities across a variety of Chinese mobile apps, effectively executing user instructions in real-world scenarios.

Code: https://github.com/inclusionAI/UI-Venus

Model: https://huggingface.co/collections/inclusionAI/ui-venus

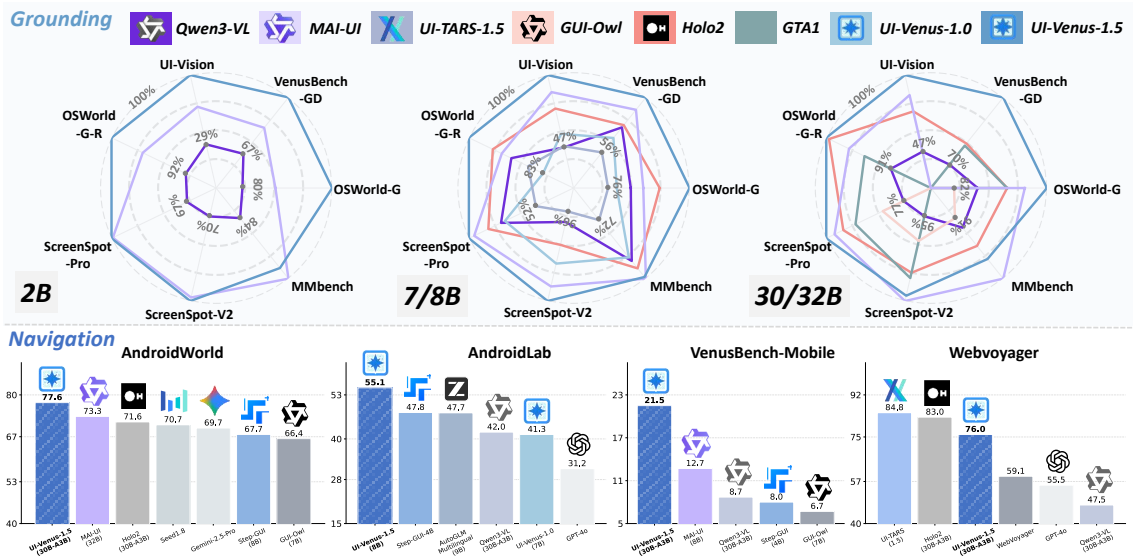


Figure 1 UI-Venus-1.5 achieves SOTA performance across multiple GUI grounding and navigation benchmarks. Note that in the three radar charts of grounding, we have normalized the results of the top-performing model to 100% to more clearly differentiate comparisons among various baselines.

1 Introduction

The pursuit of creating intelligent systems capable of autonomously operating digital devices has long been a central goal in artificial intelligence. With the rapid evolution of Multimodal Large Language Models (MLLMs) Anthropic (2024); Bai et al. (2025b); Zhu et al. (2025); Zhipu-AI (2025); Seed (2025a); Bai et al. (2025a); Ling-Team et al. (2025a); Inclusion-AI et al. (2025); Ling-Team et al. (2025b), GUI Agents Gu et al. (2025); Tang et al. (2025a); Ye et al. (2025); Yan et al. (2025); Zhou et al. (2025b); Wang et al. (2025b); Liu et al. (2024); H-Company (2025); Zhang et al. (2026) have emerged as a promising solution to bridge the gap between human instructions and digital execution. Unlike traditional automation tools that rely on rigid APIs, these agents leverage visual perception to interact directly with graphical interfaces, effectively mimicking human behavior to navigate web and mobile environments.

Currently, this field is experiencing a period of intense development. The research community is actively exploring various dimensions of agent construction, ranging from the curation of large-scale GUI datasets Zhou et al. (2025b,a); Li et al. (2025a); Rawles et al. (2025); Li et al. (2024); Gu et al. (2023); He et al. (2024); Lu et al. (2025a) to the optimization of training paradigms Chen et al. (2025). While early works focused on basic feasibility like purely Supervised Finetuning(SFT), recent studies have shifted toward more sophisticated approaches, such as designing rule-based rewards for offline/online reinforcement learning and optimizing token usage for better efficiency. Despite this progress, building an agent that is both universally capable and easy to deploy remains a significant challenge. Building on this momentum, we firstly released UI-Venus-1.0 Gu et al. (2025) not long ago. By relying solely on reinforcement learning, UI-Venus-GD and UI-Venus-Navi achieved previous state-of-the-art (SOTA) results in grounding and mobile navigation tasks, respectively.

While the GUI agent landscape has expanded rapidly, the rapid influx of new GUI agents has escalated performance standards, challenging the dominance of UI-Venus-1.0. Beyond pure performance metrics, we observe a critical discrepancy between step-level and trace-level accuracy during both SFT and offline reinforcement learning phases. This mismatch is largely due to the sparsity of rewards in individual steps and the inherent domain shift between training data and real-world benchmarks. Moreover, we posit that an ideal agent suitable for daily usage should be an **end-to-end** system that adheres to a *simple yet effective* design philosophy. To address these challenges, we present **UI-Venus-1.5** in this report, a substantially enhanced version of our previous system. Compared to UI-Venus-1.0, UI-Venus-1.5 introduces three key technical advances that jointly improve the final performances:

- **Mid-Training Stage:** Unlike the previous approach, we have added a comprehensive mid-training stage before the reinforcement learning phase. This stage utilizes an extensive corpus of GUI data, comprising 30+ datasets and a total of 10B tokens. By incorporating this step, we equip the base model with robust inherent GUI knowledge, enabling it to effectively solve GUI-related VQA, grounding, and simple navigation tasks even before entering the reinforcement learning stage.
- **Scaled Online Reinforcement Learning:** Recognizing that Online Reinforcement Learning is a highly effective method for training GUI Agents, we have integrated it into UI-Venus-1.5 specifically for mobile and web scenarios. Inspired by T-GRPO Chen et al. (2025), we perform full trajectory rollouts and reward calculations across different environments, which also contributes to address the challenging step-trace accuracy mismatch problem during GUI

Agent finetuning. By scaling up the interaction devices, we have further improved the model's performance in complex navigation tasks.

- **Unified Single-Agent via Model Merging:** A major distinction from UI-Venus-1.0 is that UI-Venus-1.5 is a purely end-to-end model, which greatly simplifies deployment for users. To achieve this, we first conduct finetuning for grounding, web, and mobile domains by using domain-specific reward functions, data, and prompts. Once we obtain these three specialized models, we apply a model merge strategy. This allows us to combine them into a single, unified model with minimal performance loss across individual domains.

We extensively evaluate UI-Venus-1.5 on diverse benchmarks to verify its versatility and robustness. In terms of GUI Grounding, our model establishes a new state-of-the-art on challenging datasets like ScreenSpot-Pro Li et al. (2025a) and VenusBench-GD Zhou et al. (2025a), achieving accuracies of **69.6%** and **75.0%** respectively, which substantially surpasses existing strong baselines like Seed1.8 Seed (2025a), Holo2 H-Company (2025) and MAI-UI Zhou et al. (2025b). Furthermore, in dynamic Navigation tasks, UI-Venus-1.5 proves the efficacy of our proposed training pipeline. On the highly competitive AndroidWorld Rawles et al. (2025) benchmark, it attains a success rate of **77.6%**, outperforming Mobile-Agent-v3 Ye et al. (2025), MAI-UI Zhou et al. (2025b) and StepGUI Yan et al. (2025). When extended to web-based interaction on WebVoyager He et al. (2024), UI-Venus-1.5 achieves 76.0% accuracy, closely matching the performance of leading prior baselines. Even when compared to significantly larger models, UI-Venus-1.5 delivers superior efficiency and decision-making accuracy across both grounding and navigation domains.

Beyond achieving state-of-the-art results on standard GUI benchmarks, we place a strong emphasis on the practical utility of UI-Venus-1.5. To this end, we have specifically optimized the model for 40+ **Chinese mobile ecosystem**, ensuring it can handle a wide array of complex, real-world tasks within third-party applications. These capabilities include, but are not limited to, ticket booking, purchase of goods, and automated conversation management. By mastering these diverse and high-demand scenarios, UI-Venus-1.5 moves closer to becoming a truly helpful digital assistant for everyday life.

2 Methodology

System overview: UI-Venus-1.5 is an end-to-end multimodal agent designed to bridge high-level user intentions with concrete GUI interactions across mobile and web platforms. As shown in Figure 2, the system operates via a closed-loop perception-action mechanism: given a natural language command, the model interprets visual screenshots, grounds semantic intentions into executable actions, and iteratively interacts with the environment until task completion. This unified architecture eliminates the need for handcrafted intermediate representations or API integrations, enabling seamless deployment on heterogeneous interfaces. Beyond state-of-the-art benchmark performance, UI-Venus-1.5 demonstrates robust practical applicability, having been validated across a diverse array of real-world applications ranging from media streaming to complex e-commerce platforms. Whether manipulating native mobile apps or navigating dynamic web content, the model exhibits consistent robustness in handling complex, context-dependent workflows, positioning it as a versatile solution for automating daily digital tasks.

Action Spaces: Building upon the foundational action space of UI-Venus-1.0, we expand the model's capabilities to encompass web-specific interactions. Specifically, we introduce three additional primitives: *Hover*, *DoubleClick*, and *Hotkey*. This augmented action space (Table 8) unifies mobile

Figure 2 System Overview of UI-Venus-1.5. It operates as an end-to-end GUI Agent that interprets user instructions, perceives interface states through screenshots, and executes interactive actions (e.g., clicking, typing, scrolling) to accomplish tasks across diverse executable environments.

and web interaction modalities, allowing the end-to-end model to execute precise operations across diverse environments.

Overall Training Pipeline: Subsequently, we elaborate on the model's training process, which is divided into four distinct stages as shown in Figure 3: (1) a Mid-Training phase for knowledge injection using large-scale GUI data in Section 2.1; (2) task-specific Offline-RL training for each of the three objectives in Section 2.2; (3) Online-RL to further enhance the agent's navigation capabilities in complex, real-world scenarios in Section 2.3; and (4) a model merge strategy to unify the specialized models in Section 2.4.

2.1 Mid-Training

2.1.1 Motivation and Data Collection

The Mid-Training phase is designed to bridge the semantic gap between general visual perception and the fine-grained structural understanding required by GUI Agents. Specifically, general-purpose vision-language models often lack the granularity needed to capture the structural nuances of user interfaces. In the subsequent reinforcement learning phase, this deficiency severely hinders effective exploration, leading to sparse reward signals and preventing policy improvement from bootstrapping in complex interaction scenarios. This limitation is mainly due to the scarcity of GUI-specific structural modeling in standard pre-training corpora. Consequently, rather than relying solely on capability elicitation, we shift our objective toward **foundational representation building**. This phase (Mid-Training) enables the model to encode diverse GUI layouts and interaction logic, providing a robust initialization for subsequent policy optimization, which can also be experimentally verified in Section 3.3.

To support the Mid-Training phase, we constructed a unified corpus by aggregating over 30 diverse

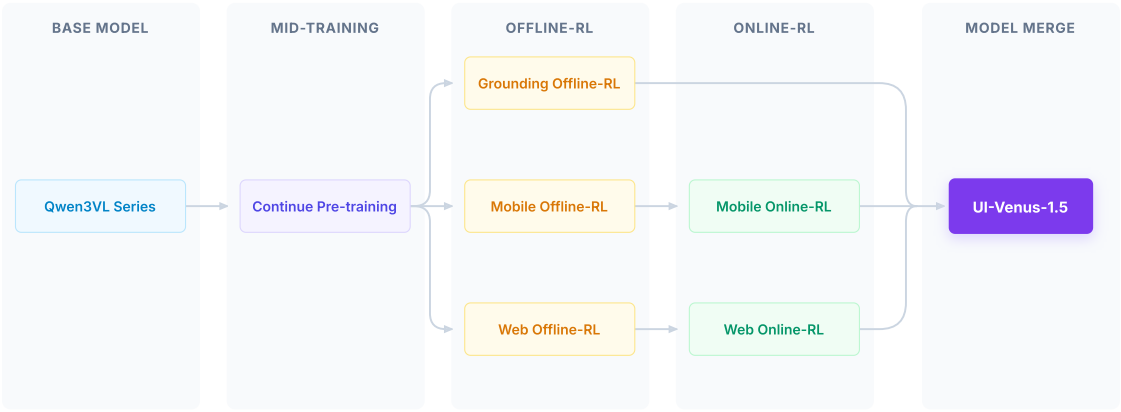
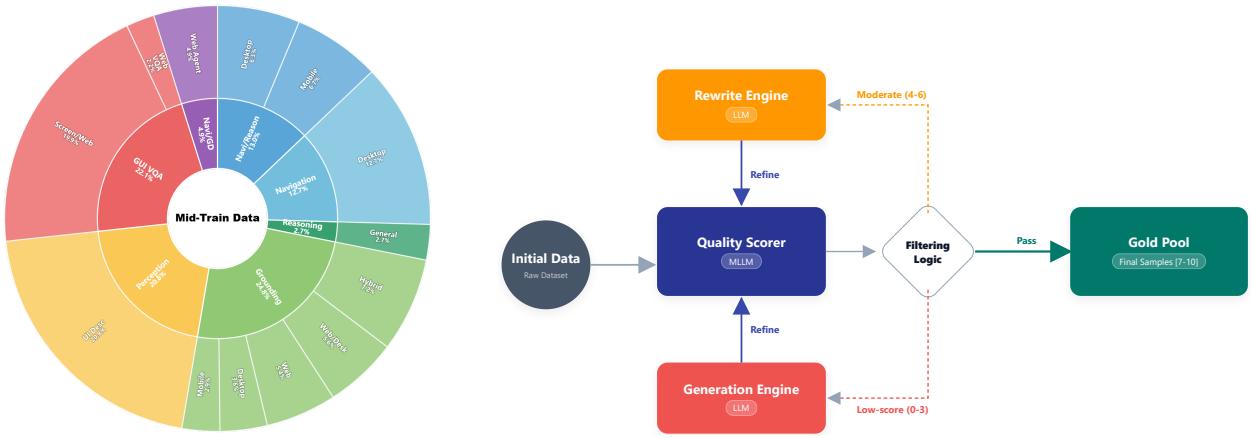


Figure 3 The Four-Stage Pipeline of UI-Venus-1.5. Starting from Qwen3-VL Series, the model progresses through a multi-stage curriculum: (1) Mid-Training on large-scale GUI data for domain knowledge injection; (2) Offline-RL for task-specific optimization across grounding, mobile, and web objectives; (3) Online-RL to enhance navigation in complex, real-world settings; and (4) Model Merge, which unifies the specialized models into the final UI-Venus-1.5.



(a) Mid-Training Corpus **(b) Iterative Data Refinement Pipeline**

Figure 4 (a) The inner part represents the functional task categories (*e.g.*, GUI-VQA, Grounding, Perception), while the outer one details the distribution of specific data sources and target platforms (Web, Desktop, Mobile); (b) Iterative data refinement pipeline with teacher scoring, trace rewriting/reconstruction, and manual verification.

sources, including Mind2Web Deng et al. (2023), ShowUI Lin et al. (2024), AITW Rawles et al. (2023) and so on. The hierarchical distribution of this corpus is detailed in Figure 4a. This 10B-token dataset is strategically stratified to ensure functional diversity: semantic perception (20.8%) and GUI-VQA (22.1%) provide the representational foundation, while grounding (24.8%) and hybrid navigation-reasoning tasks ensure execution robustness.

Based on this unified corpus, the Mid-Training data supports four complementary supervision objectives covering perception, reasoning, and action alignment. Specifically, the dataset provides supervisions for:

- **Navigation & Grounding:** Learning the precise alignment between natural language instructions and executable agent actions.
- **Sequential Reasoning:** Generating Chain-of-Thought (CoT) traces that decompose high-level goals into intermediate steps.

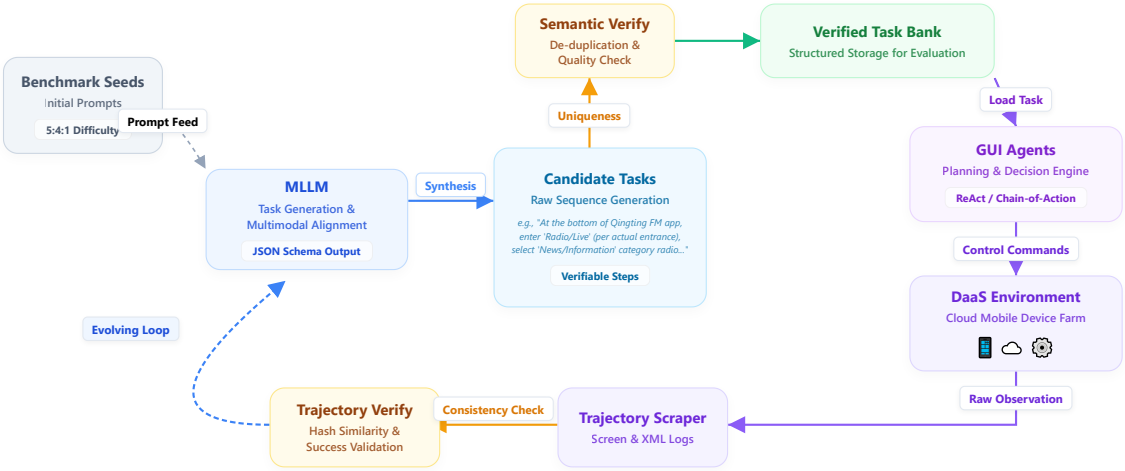


Figure 5 Data generation loop via DaaS environment. By iteratively performing this pipeline, the success rate of total trace generation raises from 17.9% to over 70%.

- **GUI-VQA:** Providing semantic interpretations of GUI components, functional descriptions, and layout logic.
- **Fine-Grained Perception:** Capturing detailed attributes of visual elements, including icon recognition, widget state detection, and OCR-free dense captioning.

2.1.2 Iterative Data Refinement

With the large scale data we collected for Mid-Training, the next step is to clean and refine the low-quality navigation traces since some open-source datasets often contain noise that can limit performance gains. To address this, we propose a teacher-based quality refinement module to rank, select, and rewrite the Mid-Training data. Specifically, we first utilize Qwen3-VL-235B-A22B Bai et al. (2025a) to evaluate and rank the input data with numerical quality assessments as illustrated in Figure 4b following the LLM-as-a-judge fashion. We chose this model because of its superior generalization and reasoning capabilities among open-source models. Mid-Training samples are scored from 0 to 10 based on action-visual alignment and task reachability according to our carefully designed prompts. Among the results, high-quality traces(score $\geq$ 7) are retained in gold pool since the goal is accomplished; mid-quality traces(4-6) are routed to a rewriting model to refine the instruction according to the last state; and low-quality samples(0-3) are totally reconstructed or just discarded.

By performing recursive refinements, the proportion of high-fidelity samples in our dataset increased from 69.7% to 89.7%. Finally, we conduct subsequent manual verification of sampled trajectories from the gold pool and ensure that the training signal remains both dense and accurate.

2.1.3 Data Generation Loop

To further improve robustness in real-world environments, we augment the Mid-Training corpus with interaction trajectories collected from real-device execution. Compared with static open-source datasets, real-device interaction data better captures execution failures, GUI dynamics, and environment-dependent behaviors that GUI agents must handle in practice. We therefore build a data generation loop on top of our DaaS system (Section 2.3.2). As illustrated in Figure 5, an open-source MLLM first generates candidate task prompts from seed instructions. After semantic verification

based on embedding similarity, valid and non-duplicate prompts are stored in a task bank and executed by GUI agents on cloud-hosted devices. The system then performs GUI trajectory scraping followed by multi-annotator verification to collect high-quality interaction trajectories. A key feature of this pipeline is its iterative generation loop. Verified trajectories are fed back to the MLLM as in-context examples for subsequent task generation, enabling progressively more executable and realistic task prompts. As a result, the success rate of the trajectory generation pipeline improves from 17.9% to over 70% after several iterations. In total, this real-device generation loop produces more than 30,000 verified interaction trajectories.

2.2 Offline Reinforcement Learning

2.2.1 Grounding

Reward: Following our previous work, the reward function is composed of two components formally: We first check whether the predicted answer string conforms to a predefined syntax as the format reward R_{format}. Next, given a screenshot and the instruction, the model must predict a center point that localizes the element. We use the commonly used point-in-box reward noted as $R_{\text{point-in-box}}$ to train the model.

Combining all components, the final action-wise reward is computed as:

$$R = R_{\text{format}} \cdot w_1 + R_{\text{poin-in-box}} \cdot w_2, \tag{1}$$

where w_1 and w_2 control the relative importance of format correctness and location precision.

Refusal Samples: A major departure from UI-Venus-1.0 is the modification of our grounding prompts to incorporate what we define as refusal capability. Specifically, when faced with an instruction that refers to an element or icon, not present in the image, the model is trained to return a fixed output of $[-1, -1]$, refer to our prompt as shown in A.2. Compared to models that strictly output coordinates regardless of the instruction's validity, this refusal-aware approach is significantly more intelligent and effectively mitigates hallucinations during user interactions.

Although the introduction of refusal prompts may lead to a marginal performance trade-off on benchmarks lacking refusal examples (such as ScreenSpot-Pro Li et al. (2025a)), UI-Venus-1.5 nevertheless maintains its state-of-the-art (SOTA) standing. Furthermore, on benchmarks that explicitly include refusal tasks—such as VenusBench-GD Zhou et al. (2025a), and OSWorld-G-Refine Xie et al. (2025b), our model achieves new SOTA results, particularly demonstrating superior accuracy on refusal-specific samples.

2.2.2 Navigation

In this section, we detail the Offline-RL formulation and empirical observations across Mobile and Web navigation tasks. In reinforcement learning, a well-designed and robust reward system is essential for stable policy optimization. Despite the differences between Mobile and Web platforms, GUI agents typically share a substantially overlapping action space allowing for a unified reward design across both domains.

Reward: To ensure the agent model generates structured, executable outputs, we build upon our prior UI-Venus-1.0 Gu et al. (2025) framework and adopt a decoupled reward system comprising two primary components: (i) a format reward and (ii) an action reward.

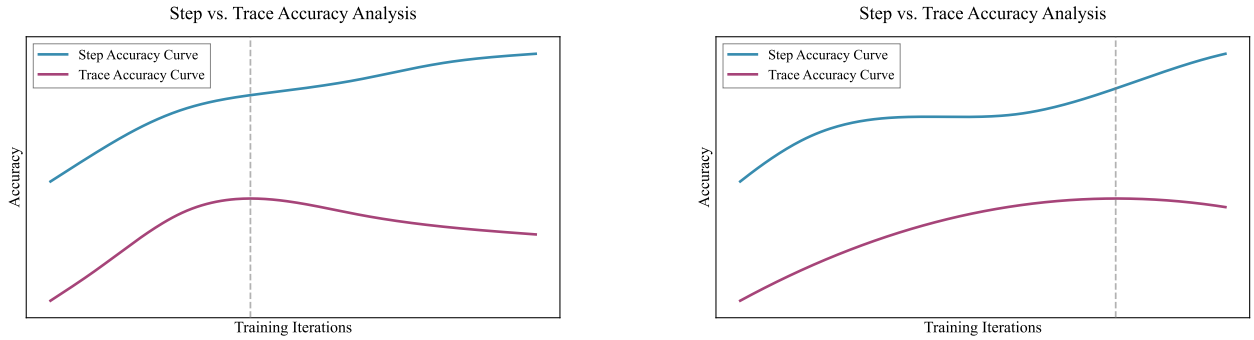
- **Format reward** R_{format}: This component ensures the agent model follows a predefined XML-based template. Specifically, the agent is rewarded for enclosing its response within `<think>`, `<action>`, and `<conclusion>` tags, which represent the reasoning process, the GUI action, and a concise action summary, respectively.
- **Action reward** R_{action}: The primary objective of the action reward is to encourage the model to predict valid and contextually appropriate actions for the current step. It is decomposed into two parts: an action-type reward R_{type} and either a content-related reward R_{content} or a coordinate-related reward R_{coord}. Specifically, R_{type} is a binary reward determined by whether the predicted action type matches the ground-truth type. R_{content} is applied to text-based actions and is computed as the token-level F1-score between the predicted and ground-truth content. For R_{coord}, we adopt a hierarchical reward strategy that gradually relaxes the tolerance on coordinate errors, smoothing the reward scale and reducing the difficulty of policy optimization.

Compared to the UI-Venus-1.0 baseline, we enhance navigation performance in web environments by introducing several specialized GUI actions, such as `Hover` and `Hotkey`. Furthermore, we implement domain-specific constraints for the `Scroll` action to align with the different operational logics of each platform: in mobile tasks, the model must predict precise start and end coordinates, whereas in web tasks, it is only required to specify the scrolling direction.

Overall, the total reward R is formulated as:

$$R = w_1 \cdot R_{\text{format}} + w_2 \cdot R_{\text{action}}, \tag{2}$$

where w_1 and w_2 control the trade-off between the format reward and the action reward.



(a) Mobile Scenario **(b) Web Scenario**

Figure 6 Step vs. trace success rates during Offline-RL training in (a) Mobile and (b) Web scenarios. We show the performance of training iterations on mobile and web benchmarks with two curves. The dashed line marks the peak trace-level success rate.

Discrepancy Between Step and Trace Success Rates: We use the above reward system for stable Offline-RL. However, we observe a notable trend during Offline-RL training. As shown in Figure 6, while step-level success rates increase steadily, trace-level success rates eventually peak and then decline. We attribute this behavior to an inherent limitation of Offline-RL: step-level rewards only optimize individual actions and fail to guide the successful composition of a full multi-step trace. To improve the deployable performance of the model, we therefore append an online reinforcement learning stage after the Offline-RL, explicitly optimizing for trace-level rewards and substantially enhancing the model's end-to-end task completion.

2.3 Online Reinforcement Learning

2.3.1 Motivation

While SFT and Offline-RL provide a solid initialization for GUI agents, their effectiveness is inherently constrained by existing static datasets and predefined interaction distributions. In real-world GUI environments, agents are required to navigate dynamic GUI states, stochastic system behaviors, and extended decision-making horizons, where real-time feedback during execution is critical to performance. Pure offline training is insufficient to address these challenges. Online Reinforcement Learning(Online-RL) responds to these limitations by enabling the agent to learn and adapt in real time through direct interaction with the environment. This allows for rich and immediate feedback and better handling of uncertainties in dynamic settings especially in long interaction sequences. Consequently, policies trained solely offline often exhibit brittleness when deployed in novel GUI layouts or unexpected intermediate states. These limitations are further corroborated by the inconsistencies in step-trace accuracy observed in Section 2.2.

To address these limitations, we introduce online reinforcement learning as a complementary training paradigm. By enabling direct environmental interaction, Online-RL enables the agent to collect trajectories that reflect the actual deployment distribution and to iteratively refine and update its policy based on observed feedback. This leads to the design of a dedicated online learning framework, which comprises a robust execution infrastructure called DaaS (Section 2.3.2), comprehensive task generation and reward formulations (Section 2.3.3), and a stable RL training algorithm (Section 2.3.4).

2.3.2 Device as a Service (DaaS)

Training and deploying a GUI Agent capable of generalizing across heterogeneous devices imposes stringent demands on the underlying execution environment in terms of uniformity, extensibility, and performance. Unlike simulators designed for a single device category, such an environment must accommodate a wide variety of device types, offer a unified abstraction over diverse interaction protocols, and—under strict network-isolation policies, securely and efficiently expose large-scale device resources to upstream training and deployment frameworks. Therefore, we build a unified Device-as-a-Service (DaaS) layer (Figure 7) to meet these requirements. It consists of two core components: the Group Control Gateway (GCGW) and the Unified Client SDK, whose design and implementation are detailed next.

Group Control Gateway (GCGW). The GCGW serves as a high-performance centralized reverse proxy and the orchestration core of the DaaS layer. It abstracts heterogeneous control protocols across diverse platforms—including Android (ADB), Chrome (CDP), and Linux containers (SSH)—enabling extensible support through a protocol-centric abstraction.

To ensure system stability and performance under high-throughput workloads, the GCGW integrates several key architectural optimizations. First, to handle stateful protocols like ADB and CDP which rely on long-lived connections, the GCGW employs an internal secondary hash routing algorithm. This mechanism ensures that all requests for a specific device are consistently routed to the same gateway node, effectively preventing the "$M \times N$ connection explosion" problem(M gateway nodes and N devices) and significantly reducing the connection overhead per node. To support this routing strategy without sacrificing performance, the gateway utilizes streaming transmission and zero-copy I/O for internal forwarding between nodes, ensuring near-zero additional latency. Furthermore, the entire GCGW architecture is built on a high-concurrency coroutine model. This design is specifically

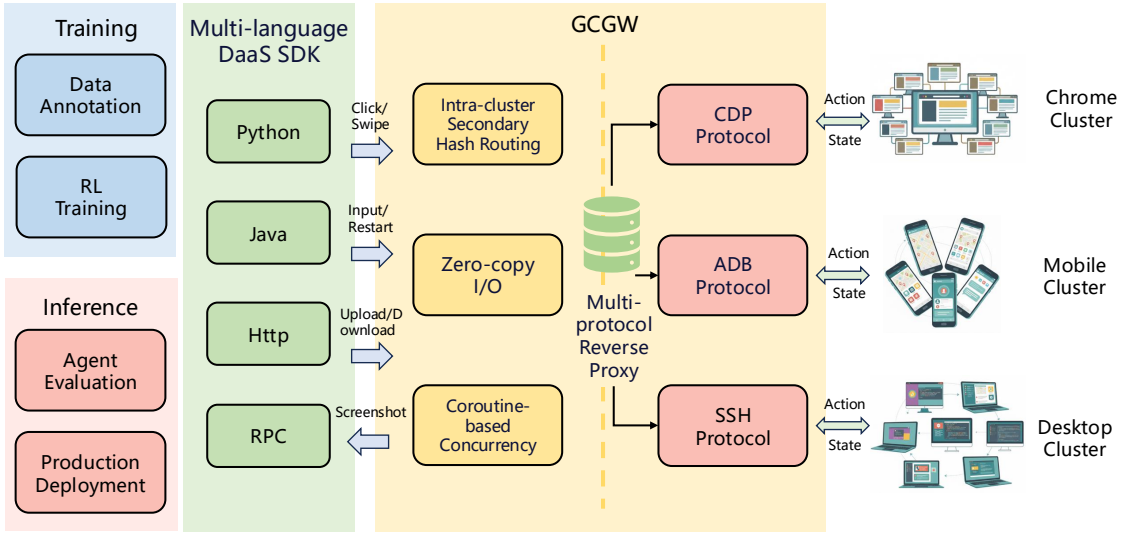


Figure 7 Architecture of the Unified Device-as-a-Service (DaaS) layer. This framework bridges upstream tasks (Training and Inference) with downstream heterogeneous device clusters (Chrome, Mobile, and Desktop). The Multi-language DaaS SDK provides a unified abstraction for diverse interaction protocols, while the Group Control Gateway (GCGW) ensures high-performance and secure resource exposure through secondary hash routing, zero-copy I/O, and a multi-protocol reverse proxy.

tailored for the "high-concurrency, low-frequency" access patterns of device operations, allowing the system to maintain hundreds of thousands of concurrent connections with minimal memory overhead.

Unified Client SDK. To further shield downstream users from the complexities of infrastructure management, a multi-language Unified Client SDK was developed that acts as a high-level API layer atop the GCGW. This SDK encapsulates several critical functions into a streamlined workflow. Specifically, it automates device lifecycle management, including device preemption, heartbeat maintenance, and resource release. Furthermore, it provides a unified semantic interaction interface that standardizes communication across various internal protocols. By abstracting these low-level operations, the SDK significantly lowers integration barriers, allowing downstream teams to concentrate on building and operating the end-to-end training pipeline for large-scale online reinforcement learning of GUI-specialized models, as well as the production deployment of those GUI-focused models—rather than dealing with protocol-specific technicalities.

By implementing these architectural optimizations, the DaaS layer achieves the following performance benchmarks:

- **Scale and Throughput:** The system successfully integrates thousands of heterogeneous devices, establishing a resilient architecture that processes millions of operation requests daily.
- **Resource Allocation Efficiency:** Device resource allocation and scheduling achieve millisecond-level responsiveness, enabling rapid elastic provisioning.
- **High-Concurrency Support:** The system has successfully supported large-scale reinforcement learning training tasks involving hundreds to thousands of concurrent devices, which demonstrated superior stability and extensibility under high-load conditions.

2.3.3 Task Formulation and Reward Design

Ahead of the training principles and pipeline of online-RL, we introduce task formulation and reward design which are fundamental to the success of online-RL. The quality, diversity, and calibrated difficulty of the input tasks define the potential ceiling of policy optimization, while the specialized reward function serves as the primary driver for training efficiency and stability.

Task Generation and Stratified Sampling: The performance ceiling of online-RL is fundamentally governed by the diversity and quality of its task pool $\mathcal{T}$. To this end, we employ a hybrid generation strategy combining static heuristics with dynamic evolution:

- **Static Task Library via LLM:** For a predefined set of applications $\mathcal{A}$ and target websites $\mathcal{W}$, we leverage Large Language Models to extract functional maps and generate common tasks covering core user flows.
- **Dynamic Trajectory-based Generation via MLLM:** To capture long-tail interaction patterns, we randomly sample screenshots s_t from offline trajectories τ and use MLLMs to infer plausible task query q' from the observed state. To maintain task uniqueness, each newly generated goal is filtered by a deduplication function $\psi(\cdot)$:

$$\mathcal{T}_{new} = \{q' \mid \psi(q', \mathcal{T}_{pool}) < \epsilon\}, \tag{3}$$

  where ϵ is a semantic similarity threshold that promotes uniform coverage of the task space.
- **Stratified Sampling:** We stratify tasks by difficulty, which correlates positively with the minimum steps to completion. Tasks are bucketed based on expected step count N_{steps} into: **Easy** ($N_{steps} \leq 10$), **Medium** ($10 < N_{steps} \leq 20$), and **Hard** ($N_{steps} > 20$). During each training iteration, batches are sampled proportionally from these three buckets to support structured curriculum learning.

Reward: To guide the agent effectively in complex GUI environments, we design a composite reward function R. For a given execution trajectory $\tau = (a_0, a_1, \ldots, a_T)$ with T steps, the total reward consists of a task completion reward R_{comp}, an action constraint penalty R_p, and a trace length decay coefficient $\eta \in (0, 1]$:

$$R(\tau) = \mathbb{1}_{success} \cdot R_{comp} \cdot \eta^{\frac{T - T_{\min}}{T_{\min}}} + \sum_{t=0}^{T} R_p(a_t), \tag{4}$$

where $T_{\min}$ is the minimum number of steps to success among a group of trajectories collected during the online RL process. Specifically, to encourage the agent to learn the shortest operational path, we introduce a decay coefficient η. A larger step count T results in a lower final reward, thereby suppressing redundant or circular actions during policy gradient optimization. Another important factor of GUI Agent is to generate the correct actions in the predefined action pool, *i.e.*, the agent's output must adhere to specific syntactic specifications. If the agent's generated response cannot be recognized by the parser as a valid action, a negative penalty λ is assigned at the current step. The penalty term is defined as:

$$R_p(a_t) = \begin{cases} -\lambda, & \text{if } a_t \text{ is unparseable} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

By incorporating R_p, we significantly reduce the number of invalid attempts during the online exploration phase, thereby improving sample efficiency.

After that, we implement a dual-track verification mechanism to determine task success ($\mathbb{1}_{success}$):

- **Rule-based Verification:** For tasks with clear system-side outcomes (*e.g.*, URL redirection, specific file generation, or system setting changes), success is verified deterministically by querying low-level system APIs.
- **MLLM-as-a-Judge:** For semantically ambiguous tasks where visual feedback is prominent, the initial task q and the final keyframe screenshot s_i are fed into an MLLM to judge whether the logical intent has been satisfied.

2.3.4 Training Algorithm

We employ the Group Relative Policy Optimization (GRPO) algorithm. Unlike the conventional Actor-Critic framework, GRPO estimates relative advantages directly from a group of sampled trajectories, thereby bypassing the need for a separate value function network. This approach substantially reduces computational complexity and effectively addresses the convergence challenges posed by sparse reward signals in GUI-based tasks.

In each training step, for a task q sampled from the task pool, the agent generates a group of G complete interaction trajectories $\{\tau_i\}_{i=1}^{G}$ using the current policy $\pi_{\theta_{old}}$. The GRPO loss function $L_{GRPO}(\theta)$ is defined as follows:

$$L_{GRPO}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\min\left(r_{i,t}(\theta)\hat{A}_i, \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\right), \tag{6}$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t}|s_{i,t},q)}{\pi_{\theta_{old}}(a_{i,t}|s_{i,t},q)}$ denotes the importance sampling ratio between the new and old policies at the action (or token) level.

Trajectory-level Advantage Calculation and Assignment: Given the long execution horizons of GUI tasks and the difficulty in identifying critical actions, we forgo step-wise reward signals in favor of evaluating the overall quality of each complete trajectory τ_i. The trajectory-level advantage $\hat{A}_i$ is derived by normalizing relative scores within the sampled group:

$$\hat{A}_i = \frac{R(\tau_i, q) - \text{mean}(\{R(\tau_j, q)\}_{j=1}^{G})}{\text{std}(\{R(\tau_j, q)\}_{j=1}^{G}) + \epsilon}, \tag{7}$$

where $R(\tau_i, q)$ is the composite reward defined in Section 2.3.3 (incorporating task success rewards and invalid action penalties). The calculated $\hat{A}_i$ is **uniformly assigned** to all action steps within the trajectory. By relying on competition within the sampled group, this approach filters out environmental stochasticity and supplies a stable credit assignment signal, thereby supporting stable policy updates in long-horizon decision-making.

Training Stability and Exploration Enhancement: Rewards in GUI navigation are both sparse and costly to verify, which can lead to policy collapse during extended online training. To mitigate these issues, we implement two complementary regularization mechanisms:

- **Adaptive KL Constraint:** To prevent the model from losing the fundamental GUI manipulation capabilities (*e.g.*, basic clicking and swiping logic) acquired during SFT or Offline-RL, we incorporate a KL divergence penalty between the current policy π_θ and reference policy π_{ref}:

$$L_{KL}(\theta) = \beta\mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}). \tag{8}$$

To prevent the reference policy from becoming a stationary constraint that limits progress, we update it adaptively. When the current policy outperforms π_{ref} on a held-out validation set by a margin δ, we smoothly blend the two policies:

$$\pi_{ref} \leftarrow (1-\alpha)\pi_{ref} + \alpha\pi_{\theta}. \tag{9}$$

This allows the KL penalty to dynamically track the policy's improvement, preserving stability while allowing continued optimization.

- **Annealed Entropy Regularization:** After SFT or Offline-RL, policies often become overly deterministic, hampering exploration early in online training. We encourage action diversity via an entropy term:

$$L_{entropy}(\theta) = -\lambda_t \mathbb{H}(\pi_\theta(\cdot|s,q)). \tag{10}$$

To avoid divergence from sustained high entropy, we anneal the coefficient $\lambda_t$exponentially with training steps k:

$$\lambda_t = \lambda_0 \cdot \sigma^k, \quad \sigma \in (0,1).$$

By maintaining a high λ_t to trigger exploration in the early stages and gradually reducing the weight to converge toward an optimal deterministic policy, this method effectively balances exploration and exploitation.

The final total optimization objective for the Online-RL stage is:

$$J(\theta) = L_{GRPO}(\theta) - L_{KL}(\theta) + L_{entropy}(\theta). \tag{11}$$

2.4 Model Merge

After offline-RL and online-RL phases, we implement a model merge Li et al. (2023); DeepResearch et al. (2025); Ling-Team et al. (2025a) strategy to consolidate the specialized expertise of our task-specific models into a single, unified GUI Agent. This approach leverages the principle that models fine-tuned from a common foundational ancestor occupy a shared parameter space, allowing for their weights to be integrated through strategic interpolation. Specifically, we explored two distinct merging paradigms as follows: Linear Merge Li et al. (2023) and TIES-Merge Yadav et al. (2023).

Linear Merge: We take the checkpoints optimized for grounding, web, and mobile navigation, and synthesize them into a global parameter set θ_{linear} using a weighted combination:

$$\theta_{linear} = \sum_{i=1}^{3} w_i \cdot \theta_i, \quad \text{subject to} \quad \sum_{i=1}^{3} w_i = 1, \tag{12}$$

where θ_i denotes the parameters of the i-th specialized model and w_i represents its relative importance in the fusion process.

TIES-Merge: Compared to standard linear merging, TIES-Merge reduces parameter interference through two key steps. First, it calculates task vectors (differences between fine-tuned models and the base model) and prunes low-magnitude updates to retain only the most significant changes. Second, before merging, it elects a dominant sign direction for each parameter and aggregates only updates aligned with that direction. By pruning noisy updates and resolving sign conflicts, TIES-Merge achieves markedly lower performance regression than simple interpolation.

Performance comparison: According to our experiments in Section 3.4, *TIES-Merge always performs better than Linear Merge*. Take our UI-Venus-1.5-30B-A3B for example, it achieves 71.0%

and 75.5% accuracy on ScreenSpot-Pro and AndroidWorld before Model Merging, respectively. By adopting Linear Merge, the performances drop 2.9%↓ and 2.3%↓. Refer to the experiment results of TIES-Merging in Table 7, *i.e.* 69.6%(1.4%↓) and 77.6%(2.1%↑), it significantly outperforms Linear Merge in the context of cross-task fusion. Note that although model merging may lead to performance drop of some tasks compared to domain-specific models, the resulting UI-Venus-1.5 achieves a harmonious balance across all three domains, delivering robust performance without the computational overhead of training a multi-task model from scratch.

3 Experiments

Models	Grounding Benchmarks						
	VenusBench-GD	ScreenSpot-Pro	ScreenSpot-V2	MMbench	OSworld-G-R	OSworld-G	UI-Vision
General VLMs							
Seed1.8 (Seed, 2025a)	-	64.3	-	-	-	-	-
Qwen3-VL-2B* (Bai et al., 2025a)	45.2	40.6	85.6	69.5	60.6	47.7	13.1
Qwen3-VL-8B* (Bai et al., 2025a)	55.1	52.7	92.1	81.4	67.0	57.5	21.9
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	52.4	53.7	91.7	83.7	69.3	61.2	25.6
GUI-specific Models							
OpenCUA-7B (Wang et al., 2025b)	48.2	50.0	92.3	-	-	55.3	29.7
OpenCUA-32B (Wang et al., 2025b)	50.1	55.3	93.4	-	70.2	59.6	33.3
OpenCUA-72B (Wang et al., 2025b)	-	60.8	92.9	-	-	-	37.3
GTA1-7B (Yang et al., 2025)	46.4	50.1	92.4	-	67.7	60.1	-
GTA1-32B (Yang et al., 2025)	58.8	63.6	95.2	-	72.2	65.2	-
GUI-Owl-7B (Ye et al., 2025)	-	54.9	92.8	80.5	-	55.9	-
GUI-Owl-32B (Ye et al., 2025)	-	58.0	93.1	83.0	-	58.0	-
UI-TARS-1.5-7B (Seed, 2025b)	40.7	35.7	91.6	64.3	64.2	52.8	22.3
UI-Venus-1.0-7B (Gu et al., 2025)	49.0	50.8	94.1	79.9	61.7	54.6	26.5
UI-Venus-1.0-72B (Gu et al., 2025)	70.2	61.9	95.3	86.3	69.5	62.2	36.8
Holo2-8B (H-Company, 2025)	56.4*	58.9	93.2	84.5*	70.1	63.5*	35.1*
Holo2-30B-A3B (H-Company, 2025)	59.5*	66.1	94.9	86.8*	76.1	65.2*	40.9*
Step-GUI-4B (Yan et al., 2025)	54.6*	60.0	93.6	84.0	66.9	60.5*	30.0*
Step-GUI-8B (Yan et al., 2025)	-	62.6	95.1	85.6	70.0	-	-
MAI-UI-2B (Zhou et al., 2025b)	55.4*	57.4	92.5	82.6	63.5	52.0	30.3
MAI-UI-8B (Zhou et al., 2025b)	65.2*	65.8	95.2	88.8	68.6	60.1	40.7
MAI-UI-32B (Zhou et al., 2025b)	-	67.9	**96.5**	**91.3**	73.9	67.6	47.1
Ours							
UI-Venus-1.5-2B	67.3	57.7	92.8	80.3	65.6	59.4	44.8
UI-Venus-1.5-8B	72.3	68.4	95.9	88.1	74.1	69.7	46.5
UI-Venus-1.5-30B-A3B	**75.0**	**69.6**	96.2	88.6	**76.4**	**70.6**	**54.7**

Table 1 Performance comparison on various **Grounding Benchmarks**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. "*" indicates results that may require verification with original sources.

3.1 Experimental Setup

3.1.1 Implementation details

UI-Venus-1.5 incorporates the Qwen3-VL Bai et al. (2025a) architecture as its core backbone, leveraging its advanced multimodal processing capabilities to interpret complex visual interfaces. Note that we map all spatial targets into a normalized $[0, 1000]$ coordinate space following Qwen3-VL and unifies diverse events into a shared set of action space as shown in Table 8.

Note that methods marked with * indicate our own reproductions with official prompts, some of which achieve superior performance compared to their original reported results. Moreover, all methods evaluated in this section **follow the end-to-end design** without any *zoom-in or agent-framework* strategies except specially mentioned.

3.1.2 Benchmarks

Grounding: We evaluate the model's grounding capabilities with seven complementary benchmarks: VenusBench-GD Zhou et al. (2025a) for its assessment of high-level reasoning and refusal capabilities, ScreenSpot-Pro Wu et al. (2024) focuses on high-resolution, fine-grained professional layouts, UI-Vision Nayak et al. (2025) assesses reasoning abilities (*e.g.*, spatial and functional) in diverse applications, MMBench-GUI L2 Wang et al. (2025c) evaluates hierarchical-instruction following and compositional reasoning, OSWorldG and OSWorld-G Refine Xie et al. (2025b) jointly assess comprehensive skills such as layout understanding, widget matching, and fine-grained manipulation, and ScreenSpot-V2 Wu et al. (2024) serves to broaden coverage across different operating systems.

Navigation: We first evaluated UI-Venus-1.5 on two widely-adopted online mobile benchmarks: AndroidWorld Rawles et al. (2025) and Androidlab Xu et al. (2025), based on a live Android emulator with various applications and tasks. In addition, we also performed experiments on VenusBench-Mobile, a more challenging benchmark whose tasks are more related to real-world human applications. To further validate the cross-domain versatility of UI-Venus-1.5, we extended our evaluation to real-world web environments. We evaluate our model on a representative subset of WebVoyager He et al. (2024), as the full benchmark is time-consuming to evaluate and some time-sensitive tasks are no longer valid.

3.2 Main Results

3.2.1 Grounding Benchmarks

In the experiments, we compare UI-Venus-1.5 models against various state-of-the-art baselines across two different model categories: **(1) General VLMs**: Seed1.8 Seed (2025a) and widely used Qwen3-VL Bai et al. (2025a). **(2) GUI-specific Models**: OpenCUA Wang et al. (2025b), UI-TARS-1.5 Seed (2025b), GTA1 Yang et al. (2025), Gui-Owl Ye et al. (2025), Holo2 H-Company (2025), Step-GUI Yan et al. (2025) and MAI-UI Zhou et al. (2025b).

VenusBench-GD. VenusBench-GD is a comprehensive grounding benchmark spanning web, desktop, and mobile UIs, covering both basic localization and advanced, reasoning-heavy cases, and further includes *refusal grounding* to test whether agents can correctly reject infeasible instructions. As shown in Table 1, UI-Venus-1.5 achieves strong and scalable performance, with our 30B-A3B model reaching 75.0% and outperforming competitive GUI-specialized baselines.

ScreenSpot-Pro. ScreenSpot-Pro focuses on high-resolution professional software interfaces (*e.g.*, CAD, development tools, creative suites, and office applications), where dense layouts and small icons make fine-grained grounding particularly challenging. In Table 1, UI-Venus-1.5-30B-A3B achieves the best overall accuracy at 69.6%, exceeding the strongest reported baseline MAI-UI-32B (67.9%) and showing clear gains over smaller UI-Venus-1.5 variants (57.7% for 2B, 68.4% for 8B).

ScreenSpot-V2. ScreenSpot-V2 is a broad cross-platform grounding benchmark covering mobile, web, and desktop UIs with both text and icon/widget targets, reflecting everyday GUI interaction scenarios. As shown in Table 1, UI-Venus-1.5 achieves strong performance, with 30B-A3B reaching 96.2% (second-best, 0.3% behind MAI-UI-32B) and 8B reaching 95.9%, demonstrating robust generalization even in a near-saturated benchmark.

MMBench-GUI L2. MMBench-GUI L2 evaluates instruction-following grounding with both *Basic* (low-level attributes) and *Advanced* (goal-oriented, compositional) instructions, testing a model's ability to align implicit user intent with the correct UI element. In Table 1, UI-Venus-1.5 remains

competitive, reaching 88.6% with 30B-A3B.

OSWorld-G. OSWorld-G(and its refined split OSWorld-G-R) contains fine-grained desktop grounding tasks that require diverse skills such as text matching, widget recognition, layout understanding, and precise manipulation. Table 1 shows UI-Venus-1.5-30B-A3B achieves state-of-the-art performance on both settings, reaching 76.4% on OSWorld-G-R and 70.6% on OSWorld-G, surpassing strong baselines such as MAI-UI-32B (73.9% / 67.6%) and GTA1-32B (72.2% / 65.2%).

UI-Vision. UI-Vision is a license-permissive benchmark for desktop GUI grounding that emphasizes real-world applications and fine-grained reasoning (*e.g.*, spatial and functional understanding). As shown in Table 1, UI-Venus-1.5-30B-A3B achieves the strongest result at 54.7%, substantially outperforming prior GUI-specific baselines (*e.g.*, MAI-UI-32B at 47.1%).

In summary, our evaluation reveals several key insights:

- **Strong Overall Grounding:** UI-Venus-1.5-30B-A3B achieves state-of-the-art results on most benchmarks, leading VenusBench-GD (75.0%), ScreenSpot-Pro (69.6%), OSWorld-G-R (76.4%), OSWorld-G (70.6%), and UI-Vision (54.7%), while remaining highly competitive on ScreenSpot-V2 (96.2%, second-best, 0.3% behind MAI-UI-32B).
- **Consistent Scaling Gains:** Increasing model scale yields steady improvements across all benchmarks (*e.g.*, ScreenSpot-Pro: 57.7%→68.4%→69.6% for 2B/8B/30B-A3B).
- **Broad Generalization Across Tasks:** UI-Venus shows robust performance on diverse grounding settings, from refusal-aware evaluation in VenusBench-GD to fine-grained professional UI layouts in ScreenSpot-Pro, and remains competitive on instruction-intensive MMBench (88.6%).

Models	Params.	Success Rate
General VLMs		
Qwen3-VL-2B (Bai et al., 2025a)	2B	36.4
Qwen3-VL-8B (Bai et al., 2025a)	8B	47.6
Qwen3-VL-30B-A3B (Bai et al., 2025a)	30B	54.3
GLM-4.6V (V-Team et al., 2025)	106B	57.0
Gemini-2.5-Pro (Comanici et al., 2025)	-	69.7
Seed1.8 (Seed, 2025a)	-	70.7
GUI-specific Models		
UI-TARS-1.5-7B (Seed, 2025b)	7B	30.0
GUI-Owl-7B (Ye et al., 2025)	7B	66.4
UI-TARS-72B (Qin et al., 2025)	72B	46.6
UI-Venus-1.0-7B (Gu et al., 2025)	7B	49.1
UI-Venus-1.0-72B (Gu et al., 2025)	72B	65.9
Step-GUI-4B (Yan et al., 2025)	4B	63.9
Step-GUI-8B (Yan et al., 2025)	8B	67.7
Holo2-8B (H-Company, 2025)	8B	60.4
Holo2-30B-A3B (H-Company, 2025)	30B	71.6
MAI-UI-2B (Zhou et al., 2025b)	2B	49.1
MAI-UI-8B (Zhou et al., 2025b)	8B	70.7
MAI-UI-32B (Zhou et al., 2025b)	32B	73.3
Ours		
UI-Venus-1.5-2B	2B	55.6
UI-Venus-1.5-8B	8B	73.7
UI-Venus-1.5-30B-A3B	30B	**77.6**

Table 2 Performance comparison on **AndroidWorld** for end-to-end models.

Models	Params.	Success Rate
General VLMs		
Gemini-1.5-Pro (Gemini-Team et al., 2024)	-	16.7
GLM4-9B-ft (GLM et al., 2024)	9B	21.0
LLaMA3.1-ft (Grattafiori et al., 2024)	8B	23.9
GPT-4o (Hurst et al., 2024)	-	31.2
Qwen3-VL-2B* (Bai et al., 2025a)	2B	33.3
Qwen3-VL-8B* (Bai et al., 2025a)	8B	43.5
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	30B	42.0
GUI-specific Models&Frameworks		
V-Droid (Dai et al., 2025)	8B	38.3
UI-Genie (Xiao et al., 2025)	72B	41.2
MobileUse (Li et al., 2025b)	72B	44.2
UI-Venus-1.0-7B (Gu et al., 2025)	7B	41.3
UI-Venus-1.0-72B (Gu et al., 2025)	72B	49.3
AutoGLM-Mobile (Liu et al., 2024)	9B	46.8
AutoGLM-Multilingual (Liu et al., 2024)	9B	47.7
Step-GUI-4B* (Yan et al., 2025)	4B	47.8
Ours		
UI-Venus-1.5-2B	2B	36.2/44.2†
UI-Venus-1.5-8B	8B	**55.1**/68.1†
UI-Venus-1.5-30B-A3B	30B	52.9/68.1†

Table 3 Performance comparison on **AndroidLab**. Note that * indicates that the results are evaluated by us; † denotes results that have been manually verified by humans.

3.2.2 Navigation Benchmarks

In addition to evaluating GUI grounding capabilities, we further assess the UI-Venus-1.5 series on four navigation benchmarks spanning both mobile and web environments, including Android World Rawles et al. (2025), Android Lab Xu et al. (2025), VenusBench-Mobile and WebVoyager He et al. (2024). These benchmarks are challenging, fully dynamic online suites that require GUI agents to perform multi-turn adaptive perception, reasoning, and action in evolving environments, thus providing a more reliable assessment of the model's navigation capabilities.

Android World. AndroidWorld is a comprehensive online evaluation environment for GUI agents. It includes 116 programmatic tasks across 20 real-world Android applications and is widely used as a benchmark for assessing GUI agent performance. As shown in Table 2, the Venus-1.5 model family achieves state-of-the-art (SOTA) performance among models of comparable scale. Specifically, our 2B / 8B / 30B-A3B variants reach accuracies of 55.6% / 73.7% / 77.6%, outperforming existing domain-specific and general-purpose baselines. Relative to the strongest existing contender, MAI-UI-32B, UI-Venus-1.5-30B-A3B secures an absolute margin of 4.3%.

Android Lab. Android Lab is another dynamic, online evaluation benchmark comprising 138 tasks across 9 Android applications. Our model uses only raw screen screenshots as input, yet it outperforms both a range of GUI-specialized models and general-purpose models, some of which take both screenshots and XML information as inputs. As shown in Table 3, our UI-Venus-1.5 series models (2B, 8B, and 30A3B) achieve 36.2%, 55.1%, and 52.9% on AndroidLab, respectively. It is worth noting that, due to bugs in the official AndroidLab evaluation code, we additionally report human-verified results (marked with a †) in the Table 3. The corrected results show that our UI-Venus-1.5-30A3B model does not exhibit any performance degradation on AndroidLab compared with the UI-Venus-1.5-8B model (68.1% vs. 68.1%). Compared with our UI-Venus-1.0-72B model, the best model in the 1.5 series yields up to 5.8% improvement. Notably, even UI-Venus-1.5-8B significantly outperforms other state-of-the-art models.

Models	Params.	Success Rate
General VLMs		
Qwen3-VL-8B (Bai et al., 2025a)	8B	6.7
Qwen3-VL-30B-A3B (Bai et al., 2025a)	30B	8.7
GUI-specific Models		
GUI-Owl-7B (Ye et al., 2025)	7B	6.7
UI-Venus-1.0-7B (Gu et al., 2025)	7B	8.1
UI-Venus-1.0-72B (Gu et al., 2025)	72B	15.4
Step-GUI-4B (Yan et al., 2025)	4B	8.0
MAI-UI-2B (Zhou et al., 2025b)	2B	6.7
MAI-UI-8B (Zhou et al., 2025b)	8B	12.7
Ours		
UI-Venus-1.5-2B	2B	8.7
UI-Venus-1.5-8B	8B	16.1
UI-Venus-1.5-30B-A3B	30B	**21.5**

Table 4 Performance comparison on **VenusBench-Mobile** for end-to-end models. Our UI-Venus-1.5 achieves state-of-the-art performance on this challenging benchmark.

Models	Params.	Success Rate
General VLMs		
GPT-4o (Hurst et al., 2024)	-	55.5
Claude-3.7 (Anthropic, 2025a)	-	84.1
Qwen3-VL-2B* (Bai et al., 2025a)	2B	35.2
Qwen3-VL-8B* (Bai et al., 2025a)	8B	45.2
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	30B	47.5
GUI-specific Models		
WebVoyager (He et al., 2024)	-	59.1
OpenAI-CUA (OpenAI, 2025a)	-	**87.0**
UI-TARS-1.5 (Qin et al., 2025)	-	84.8
Holo2-4B (H-Company, 2025)	4B	80.2
Holo2-8B (H-Company, 2025)	8B	80.2
Holo2-30B-A3B (H-Company, 2025)	30B	83.0
Ours		
UI-Venus-1.5-2B	2B	56.4
UI-Venus-1.5-8B	8B	70.8
UI-Venus-1.5-30B-A3B	30B	76.0

Table 5 Performance comparison on **Webvoyager** existing GUI Agents. Note that * indicates that the results are evaluated by us.

VenusBench-Mobile. VenusBench-Mobile is a challenging benchmark designed to evaluate the end-to-end performance of GUI agents in complex mobile environments. As illustrated in Table 4, the UI-Venus-1.5 model family achieves state-of-the-art (SOTA) performance across all scales.

Specifically, our 2B, 8B, and 30B-A3B variants reach success rates of 8.7%, 16.1%, and 21.5%, respectively, consistently outperforming both general-purpose VLMs and specialized GUI models. Notably, our 8B model (16.1%) already surpasses the much larger UI-Venus-1.0-72B (15.4%), while our 30B-A3B variant sets a new record with a substantial 6.1% absolute margin over the previous best-performing model.

WebVoyager. WebVoyager is a comprehensive end-to-end benchmark for evaluating web agents on 15 real-world websites including e-commerce, travel, and social platforms. It employs an automatic evaluation protocol leveraging MLLM to assess task completion rates, measuring agents' abilities to autonomously navigate and interact with dynamic web environments through visual screenshots and textual elements. As shown in Table 5, the UI-Venus-1.5 model family achieves comparable performance among models of comparable scale. Specifically, our 2B/8B/30B-A3B variants reach success rates of 56.4%/70.8%/76.0%, outperforming WebVoyager He et al. (2024) and general VLMs (*e.g.*, GPT-4o, Qwen3-VL).

In summary, our evaluation reveals several key insights:

- **Superior End-to-End Performance:** UI-Venus-1.5-30B-A3B achieves state-of-the-art or comparable results across a diverse range of GUI agent benchmarks, including Android World (77.6%), Android Lab (55.1%/68.1%†), VenusBench-Mobile (21.5%), and WebVoyager (76.0%). It consistently outperforms both specialized GUI models (*e.g.*, MAI-UI-32B) and leading general-purpose VLMs (*e.g.*, GPT-4o, Qwen3-VL), establishing a new performance ceiling for autonomous agents.
- **Significant Architectural Efficiency and Scaling:** Increasing the model scale leads to consistent performance gains across all benchmarks. Notably, the UI-Venus-1.5 family exhibits remarkable efficiency; our 8B model already surpasses the previous generation's 72B variant on both Android Lab (up to 5.8% improvement) and VenusBench-Mobile (16.1% vs. 15.4%), demonstrating the effectiveness of our updated training methodology.
- **Robust Cross-Platform Generalization:** UI-Venus demonstrates exceptional adaptability across different operating systems and input modalities. It excels not only in programmatic Android environments but also in dynamic web navigation (WebVoyager). Furthermore, the models show strong visual-only reasoning capabilities, outperforming XML-augmented baselines in Android Lab even when relying solely on raw screenshots.

3.3 The Influence of Mid-Training

To verify the effectiveness of our Mid-Training strategy, we conduct the qualitative latent space analysis as shown in Figure 8. Specifically, we analyzed the latent representations using t-SNE visualization to quantify the impact of Mid-Training. By comparing the base model with our model after Mid-Training, we observe several key observations:

- **Cluster Separability:** After Mid-Training, GUI-specific features exhibit a transition to high-density clusters. The Silhouette Score reached 0.315, a relative increase of 34.0% over the base model ($0.235 \rightarrow 0.315$).
- **Feature Sensitivity:** The 11.6% decrease in Intra-class Consistency ($0.448 \rightarrow 0.396$) indicates enhanced discriminative power. It demonstrates that the model is now capable of capturing fine-grained functional and structural variances, allowing for a more granular characterization of GUI elements that were previously treated as uniform.

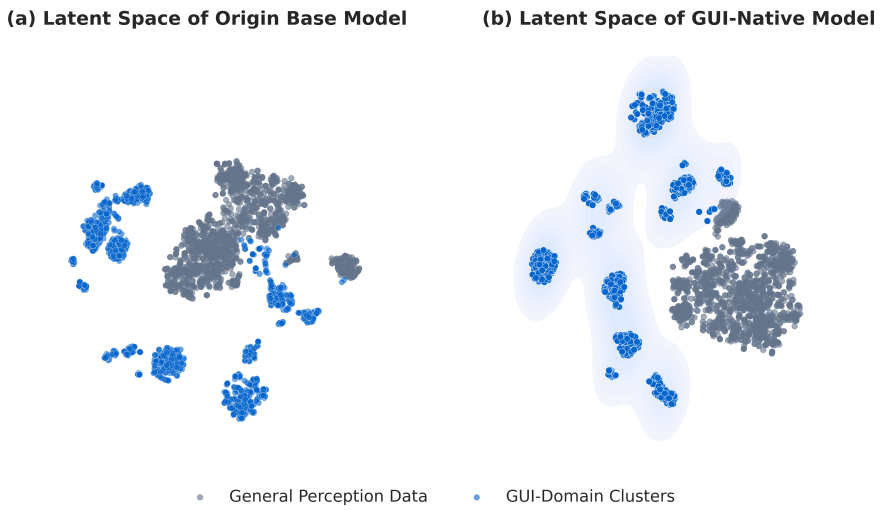


Figure 8 Latent space visualization of (a) base model and (b) model with GUI knowledge. The emergence of distinct clusters indicates that our Mid-Training has successfully enriched the model with GUI domain knowledge. This increased discriminative power between GUI and general data provides a robust structural basis for the reinforcement learning stage.

- **Global Space Stability:** The Inter-class Similarity remains stable (only 1.4% increase), confirming that GUI-specific knowledge does not cause representation collapse.

Metric	Qwen3VL	Model After Mid-Training	Change
Silhouette Score	0.235	0.315	+34.0% (↑)
Intra-class Consistency	0.448	0.396	-11.6% (↓)
Inter-class Similarity	0.220	0.223	+1.4% (≈ stable)

Table 6 Latent space metrics of the model before and after Mid-Training.

3.4 Ablation Studies

Models	Mid-Training		Offline-RL		Online-RL		Model Merge	
	SS-Pro	AW	SS-Pro	AW	SS-Pro	AW	SS-Pro	AW
UI-Venus-1.5								
2B	52.3	39.0	**59.0**(+6.7↑)	45.3(+6.3↑)	-	**59.8**(+14.5↑)	57.7(-1.3↓)	55.6 (-4.2↓)
8B	63.1	57.0	**70.0**(+6.9↑)	63.5(+6.5↑)	-	72.7(+9.2↑)	68.4(-1.6↓)	**73.7**(+1.0↑)
30B-A3B	65.2	67.1	**71.0**(+5.8↑)	68.0(+0.9↑)	-	75.5(+7.5↑)	69.6(-1.4↓)	**77.6**(+2.1↑)

Table 7 Ablation studies of UI-Venus-1.5 on ScreenSpot-Pro (denote as "SS-Pro") and AndroidWorld (denote as "AW").

In this section, we will show the performance gains of every step in the UI-Venus-1.5 pipeline, including Mid-Training, Offline-RL, Online-RL and Model Merge. As shown in Table 7, we can conclude following insights:

- **Offline-RL: Building Foundation for Grounding and Navigation.** The transition from Mid-Training to Offline-RL yields consistent improvements across all scales and tasks. Specifically, ScreenSpot-Pro scores increase by approximately 6–7%, while AndroidWorld (AW) performance also sees a significant boost (up to +6.5% for the 8B model). This confirms that

GRPO on diverse, task-specific offline data effectively aligns the model's visual perception with GUI-specific action spaces.

- **Online-RL: The Catalyst for Complex Navigation.** Online-RL serves as the most critical stage for enhancing autonomous navigation capabilities. We observe a substantial leap in AndroidWorld success rates, with the 2B model showing a remarkable +14.5% absolute gain. By interacting with dynamic environments and learning from exploration, the models overcome the limitations of static datasets, significantly improving their ability to handle long-horizon tasks and error recovery in real-world scenarios.
- **Model Merge: Balancing Specialization and Generalization.** The final model merge stage aims to unify specialized capabilities. While it leads to a minor, acceptable trade-off in fine-grained grounding (a drop of $\sim$1.4% on ScreenSpot-Pro), it further stabilizes and enhances navigation performance for larger models. Notably, UI-Venus-1.5-30B-A3B gains an additional 2.1% on AndroidWorld after the merge, suggesting that the unification process helps the model leverage cross-task knowledge to solve complex GUI sequences more effectively.

4 Related Works

GUI agents can automatically execute a series of operations on GUI screens based on given instructions. In early works, the system typically relied on predefined rules, which exhibited limited scalability. With the emergence and advancement of LLMs, it is possible to adopt a single model to handle diverse tasks as an intelligent GUI agent.

GUI Grounding. Some researches focus on GUI grounding that aims to map natural language descriptions or instructions to precise positions of GUI elements on screens, enabling autonomous agents to identify widgets with diverse functionalities and select the appropriate one to interact by an equipped planner. Early methods Cheng et al. (2024); Lin et al. (2024); Xu et al. (2024); Wu et al. (2024); Gu et al. (2023); Gou et al. (2024); Wang et al. (2024c); Wu et al. (2025); Xie et al. (2025b); Tang et al. (2025b) usually adopt supervised fine-tuning (SFT) to train grounding models, which leverages labeled data rapidly and produces various grounding models that are able to recognize and locate diverse elements on common GUI scenarios. As models evolve rapidly, the accuracies on some grounding benchmarks Cheng et al. (2024); Wu et al. (2024) has hit a ceiling. To further evaluate the grounding capabilities of models in complex scenarios, more benchmarks like Screenspot-Pro Li et al. (2025a) and VenusBench-GD Zhou et al. (2025a) have been introduced to explore the limit of performance, which additionally requires the understanding of professional software and comprehensive visual reasoning. In parallel, the training paradigms are shifting. Inspired by DeepSeek-R1 DeepSeek-AI (2025), recent works have incorporated reinforcement learning (RL) into training process, aiming to enhance model generalization across unseen scenarios with limited labeled data Lu et al. (2025b); Luo et al. (2025); Liu et al. (2025); Zhou et al. (2025c); Yuan et al. (2025); Tang et al. (2025c,a); Zhang et al. (2026); Zhou et al. (2025b); Qin et al. (2025); Seed (2025b); Yan et al. (2025). In addition, some works Zhang et al. (2025); Jiang et al. (2025) focus on post-processing to improve the test-time performance of the model. The mutual evolution between benchmarks and models drives the continuous advancement of the grounding research.

End-to-End GUI Agent. Also, some researchers attempt to tackle navigation tasks end-to-end with a unified model. At early stage, the agents were trained on foundation models directly, which served as preliminary attempts at autonomous GUI agents and produced promising results Hong et al. (2024); Lin et al. (2024); Cheng et al. (2024); Deng et al. (2023); Wu et al. (2024), but there

remained a gap between the actual performance and the requirement of practical deployment. Driven by further practicality, with the incorporation of RL techniques like Direct Preference Optimization (DPO) Rafailov et al. (2023) and Group Relative Policy Optimization (GRPO) Shao et al. (2024), improved pipeline of data generation and increased computational resources, many research groups have developed more powerful agents Qin et al. (2025); Zeng et al. (2025); Sun et al. (2025b); Yang et al. (2024); Sun et al. (2025a); Qiu et al. (2026). More recently, as the alignment between real-world and training environments receives widespread focus, more end-to-end GUI agents exhibit robust practical deployment capabilities Wang et al. (2025a); Liu et al. (2024); Yan et al. (2025), which advances the feasibility of GUI agents in real-life scenarios.

GUI Agent Framework. As a collaboration paradigm that distributes the extensive context across sub-agents with various functionalities, the GUI agent framework fully leverages the base model's understanding and reasoning capabilities to analyze the task progress and GUI screens, and subsequently takes a correct action. Agent S Agashe et al. (2024) introduces experience-augmented hierarchical planning strategy to improve task execution with several role-specific agents, and Agent S2 Agashe et al. (2025) upgrades the strategy as proactive hierarchical planning to dynamically refine actions based on real-time observations. Besides, began with Mobile-Agent Wang et al. (2024b), Mobile-Agent-v2 Wang et al. (2024a) incorporates a multi-agent collaboration architecture for long-step navigation, which includes planning, decision, reflection agents and a memory unit to retain focus content. Moreover, Mobile-Agent-E Wang et al. (2025d) implements a self-evolving hierarchical framework to store long-term memory and learn from the past, and Mobile-Agent-v3 Ye et al. (2025) further advances the framework by improved base model and training strategies based on its predecessors. More studies have also explored GUI agent frameworks dro (2025); Xie et al. (2025a). These agent frameworks probably possess a higher capacity ceiling, enabling them to perform navigation tasks that require intricate reasoning and analysis. Nevertheless, the data flow in the framework usually necessitates multiple rounds of LLM input and output, which leads to substantial computational costs and obvious operation latency.

5 Conclusion

In this work, we presented UI-Venus-1.5, a comprehensive advancement in the development of practical and reliable GUI agents. To address the limitations of previous iterations and current baselines, we introduced a three-tiered training paradigm: a large-scale Mid-Training stage for robust GUI knowledge injection, a task-specific Offline-RL phase unified by an efficient model merge strategy, and a scaled Online-RL framework to master complex navigation. Furthermore, the unified capability of UI-Venus-1.5 is achieved through strategic model merging, which effectively consolidates specialized domain expertise—including grounding, web, and mobile navigation—into a single end-to-end agent while preserving robust performance across tasks.

Experimental results demonstrate that UI-Venus-1.5 establishes a new state of the art across a wide spectrum of benchmarks, including GUI grounding and navigation. Beyond academic metrics, we have optimized the model for real-world utility within the 40+ Chinese third-party app ecosystem, enabling seamless automation for tasks such as ticket booking, and shopping. Collectively, these contributions on GUI Agents mark a significant step towards a truly autonomous and user-centric digital assistant.

6 Contributions

All contributors of UI-Venus-1.5 are listed in alphabetical order by their last names.

6.1 Core Contributors

Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen[†], Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, Xingran Zhou

6.2 Contributors

Weizhi Chen, Sunhao Dai, Jingya Dou, Yichen Gong, Yuan Guo, Zhenlin Guo, Feng Li, Qian Li, Jinzhen Lin, Yuqi Zhou, Linchao Zhu

6.3 Supervisors

Liang Chen, Zhenyu Guo, Changhua Meng[†], Weiqiang Wang

[†]Corresponding Authors: Shuheng Shen(shuheng.ssh@antgroup.com), Changhua Meng(changhua.mch@antgroup.com).

References

Droidrun. https://github.com/droidrun/droidrun, 2025.

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

Anthropic. Claude computer use. Available at: https://www.anthropic.com/news/developing-computer-use, 2024.

Anthropic. Claude-3-7-sonnet. https://www.anthropic.com/news/claude-3-7-sonnet, 2025a.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. https://arxiv.org/abs/2502.13923.

Zengjue Chen, Runliang Niu, He Kong, Qi Wang, Qianli Xing, and Zipei Fan. Tgrpo: Fine-tuning vision-language-action model via trajectory-wise group relative policy optimization. *arXiv preprint arXiv:2506.08440*, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. https://arxiv.org/abs/2401.10935.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuanchun Li, Yuqing Yang, Rui Tan, Mo Li, and Lili Qiu. Advancing mobile gui agents: A verifier-driven approach to practical deployment. *arXiv preprint arXiv:2503.15937*, 2025.

Tongyi-Team DeepResearch, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. https://arxiv.org/abs/2501.12948.

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.

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

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

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, 2024. https://arxiv.org/abs/2410.05243.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Zhangxuan Gu, Zhuoer Xu, Haoxing Chen, Jun Lan, Changhua Meng, and Weiqiang Wang. Mobile user interface element detection via adaptively prompt tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11155–11164, 2023.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025.

H-Company. Holo2 - open foundation models for navigation and computer use agents, 2025.

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. *arXiv preprint arXiv:2401.13919*, 2024.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxuan Zhang, Juanzi Li, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Cogagent: A visual language model for gui agents, 2024. https://arxiv.org/abs/2312.08914.

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.

Inclusion-AI, Bowen Ma, Cheng Zou, Canxiang Yan, Chunxiang Jin, Chunjie Shen, Chenyu Lian, Dandan Zheng, Fudong Wang, Furong Xu, et al. Ming-flash-omni: A sparse, unified architecture for multimodal perception and generation. *arXiv preprint arXiv:2510.24821*, 2025.

Zhiyuan Jiang, Shenghao Xie, Wenyi Li, Wenqiang Zu, Peihang Li, Jiahao Qiu, Siqi Pei, Lei Ma, Tiejun Huang, Mengdi Wang, et al. Zoom in, click out: Unlocking and evaluating the potential of zooming for gui grounding. *arXiv preprint arXiv:2512.05941*, 2025.

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

Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, and Weinan Zhang. Mobileuse: A gui agent with hierarchical reflection for autonomous mobile operation. *arXiv preprint arXiv:2507.16853*, 2025b.

Wei Li, William Bishop, Alice Li, Chris Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents, 2024. https://arxiv.org/abs/2406.03679.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*, 2023.

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

Ling-Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, Changxin Tian, Chao Huang, et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation. *arXiv preprint arXiv:2510.22115*, 2025a.

Ling-Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025b.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.

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. 2025. https://arxiv.org/abs/2504.14239.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices, 2025a. https://arxiv.org/abs/2406.08451.

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. 2025b. https://arxiv.org/abs/2503.21620.

Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1 : A generalist r1-style vision-language action model for gui agents. 2025. https://arxiv.org/abs/2504.10458.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Rabiul Awal, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, et al. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. *arXiv preprint arXiv:2503.15661*, 2025.

OpenAI. Computer using agent. https://platform.openai.com/docs/guides/tools-computer-use, 2025a.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang,

Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. https://arxiv.org/abs/2501.12326.

Xinyu Qiu, Heng Jia, Zhengwen Zeng, Shuheng Shen, Changhua Meng, Yi Yang, and Linchao Zhu. Unified generation and self-verification for vision-language models via advantage decoupled preference optimization. *arXiv preprint arXiv:2601.01483*, 2026.

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.

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

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents, 2025. https://arxiv.org/abs/2405.14573.

Bytedance Seed. Seed1. 8 model card: Towards generalized real-world agency, 2025a. https://lf3-static.bytednsdoc.com/obj/eden-cn/lapzild-tss/ljhwZthlaukjlkulzlp/research/Seed-1.8-Modelcard.pdf.

ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025b.

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

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis, 2025a. https://arxiv.org/abs/2412.19723.

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, 2025b. https://arxiv.org/abs/2503.17709.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. Gui-g^2: Gaussian reward modeling for gui grounding, 2025a. https://arxiv.org/abs/2507.15846.

Fei Tang, Yongliang Shen, Hang Zhang, Siqi Chen, Guiyang Hou, Wenqi Zhang, Wenqiao Zhang, Kaitao Song, Weiming Lu, and Yueting Zhuang. Think twice, click once: Enhancing gui grounding via fast and slow systems. 2025b. https://arxiv.org/abs/2503.06470.

Jiaqi Tang, Yu Xia, Yi-Feng Wu, Yuwei Hu, Yuhui Chen, Qing-Guo Chen, Xiaogang Xu, Xiangyu Wu, Hao Lu, Yanqing Ma, Shiyin Lu, and Qifeng Chen. Lpo: Towards accurate gui agent interaction via location preference optimization, 2025c. https://arxiv.org/abs/2506.09373.

V-Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, Baoxu Wang, Bin Chen, Boyan Shi, Changyu Pang, Chenhui Zhang, Da Yin, Fan Yang, Guoqing Chen, Jiazheng Xu, Jiale Zhu, Jiali Chen, Jing Chen, Jinhao Chen, Jinghao Lin, Jinjiang Wang, Junjie Chen, Leqi Lei, Letian Gong, Leyi Pan, Mingdao Liu, Mingde Xu, Mingzhi Zhang, Qinkai Zheng, Sheng Yang, Shi Zhong, Shiyu Huang, Shuyuan Zhao, Siyan Xue, Shangqin Tu, Shengbiao Meng, Tianshu Zhang, Tianwei Luo, Tianxiang Hao, Tianyu Tong, Wenkai Li, Wei Jia, Xiao Liu, Xiaohan Zhang, Xin Lyu, Xinyue Fan, Xuancheng Huang, Yanling Wang, Yadong Xue, Yanfeng Wang, Yanzi Wang, Yifan An, Yifan Du, Yiming Shi, Yiheng Huang, Yilin Niu, Yuan Wang, Yuanchang Yue, Yuchen Li, Yutao Zhang, Yuting Wang, Yu Wang, Yuxuan Zhang, Zhao Xue, Zhenyu Hou, Zhengxiao Du, Zihan Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025. https://arxiv.org/abs/2507.01006.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration, 2024a. https://arxiv.org/abs/2406.01014.

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

Ke Wang, Tianyu Xia, Zhangxuan Gu, Yi Zhao, Shuheng Shen, Changhua Meng, Weiqiang Wang, and Ke Xu. E-ant: A large-scale dataset for efficient automatic gui navigation, 2024c. https://arxiv.org/abs/2406.14250.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, Zhennan Shen, Zhuokai Li, Ryan Li, Xiaochuan Li, Junda Chen, Boyuan Zheng, Peihang Li, Fangyu Lei, Ruisheng Cao, Yeqiao Fu, Dongchan Shin, Martin Shin, Jiarui Hu, Yuyan Wang, Jixuan Chen, Yuxiao Ye, Danyang Zhang, Dikang Du, Hao Hu, Huarong Chen, Zaida Zhou, Yipu Wang, Heng Wang, Diyi Yang, Victor Zhong, Flood Sung, Y. Charles, Zhilin Yang, and Tao Yu. Opencua: Open foundations for computer-use agents, 2025b. https://arxiv.org/abs/2508.09123.

Xuehui Wang, Zhenyu Wu, JingJing Xie, Zichen Ding, Bowen Yang, Zehao Li, Zhaoyang Liu, Qingyun Li, Xuan Dong, Zhe Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. *arXiv preprint arXiv:2507.19478*, 2025c.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks, 2025d. https://arxiv.org/abs/2501.11733.

Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, et al. Gui-actor: Coordinate-free visual grounding for gui agents. *arXiv preprint arXiv:2506.03143*, 2025.

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

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, et al. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents. *arXiv preprint arXiv:2505.21496*, 2025.

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. *arXiv preprint arXiv:2505.16827*, 2025a.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis, 2025b. https://arxiv.org/abs/2505.13227.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. Androidlab: Training and systematic benchmarking of android autonomous agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2166, 2025.

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. 2024. https://arxiv.org/abs/2412.04454.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, et al. Step-gui technical report. *arXiv preprint arXiv:2512.15431*, 2025.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. Gta1: Gui test-time scaling agent, 2025. https://arxiv.org/abs/2507.05791.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions, 2024. https://arxiv.org/abs/2412.16256.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*, 2025.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, and Bo Li. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. 2025. https://arxiv.org/abs/2505.12370.

Zhixiong Zeng, Jing Huang, Liming Zheng, Wenkang Han, Yufeng Zhong, Lei Chen, Longrong Yang, Yingjie Chu, Yuzhi He, and Lin Ma. Uitron: Foundational gui agent with advanced perception and planning. *arXiv preprint arXiv:2508.21767*, 2025.

Le Zhang, Yixiong Xiao, Xinjiang Lu, Jingjia Cao, Yusai Zhao, Jingbo Zhou, Lang An, Zikan Feng, Wanxiang Sha, Yu Shi, Congxi Xiao, Jian Xiong, Yankai Zhang, Hua Wu, and Haifeng Wang. Omegause: Building a general-purpose gui agent for autonomous task execution, 2026. https://arxiv.org/abs/2601.20380.

Yunzhu Zhang, Zeyu Pan, Zhengwen Zeng, Shuheng Shen, Changhua Meng, and Linchao Zhu. Mvp: Multiple view prediction improves gui grounding. *arXiv preprint arXiv:2512.08529*, 2025.

Zhipu-AI. Glm-4.5v. Available at: https://docs.z.ai/guides/vlm/glm-4.5v, 2025.

Beitong Zhou, Zhexiao Huang, Yuan Guo, Zhangxuan Gu, Tianyu Xia, Zichen Luo, Fei Tang, Dehan Kong, Yanyi Shang, Suling Ou, et al. Venusbench-gd: A comprehensive multi-platform gui benchmark for diverse grounding tasks. *arXiv preprint arXiv:2512.16501*, 2025a.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025b.

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. 2025c. https://arxiv.org/abs/2505.15810.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. https://arxiv.org/abs/2504.10479.

A Action Space and Prompt Templates

A.1 Action Space

Action	Definition
Click(box=(x, y))	Click at coordinates (x, y).
Drag(start=(x1, y1), end=(x2, y2))	Drag from (x1, y1) to (x2, y2).
Scroll(start=(x1, y1), end=(x2, y2), direction='')	Scroll from (x1, y1) to (x2, y2) with specified direction.
Type(content='')	Type the specified content.
Launch(app='')	Launch the specified app.
Wait()	Wait for loading.
Finished(content='')	Finish the task, with optional information.
CallUser(content='')	Conclude the answer for information-retrieval.
LongPress(box=(x, y))	Long press at coordinates (x, y).
PressBack()	Press the 'back' button.
PressHome()	Press the 'home' button.
PressEnter()	Press the 'enter' button.
PressRecent()	Press the 'recent' button.
Hover(box=(x,y))	Move the mouse cursor to coordinates (x, y) without clicking.
DoubleClick(box=(x,y))	Perform a double-click at coordinates (x, y).
Hotkey(keys=['ctrl', 'c'])	Press the specified key combination (e.g., Ctrl+C for copy).

Table 8 All actions and their definitions used in **UI-Venus-1.5**. We unify the action space and map all the actions in the existing open-source dataset to this space.

A.2 Grounding

Grounding Prompt

Output the center point of the position corresponding to the following instruction: `{problem}`. The output should just be the coordinates of a point, in the format [x,y]. Additionally, if the task is infeasible (e.g., the task is not related to the image), the output should be [-1,-1].

A.3 Mobile

Mobile Prompt

You are a GUI Agent.
Your task is to analyze a given user task, review current screenshot and previous actions, and determine the next action to complete the task.

Available Actions
You may execute one of the following functions:
- Click(box=(x1,y1))
- Drag(start=(x1,y1), end=(x2,y2))
- Scroll(start=(x1,y1), end=(x2,y2))
- Type(content='')
- Launch(app='')
- Wait()
- Finished(content='')

- CallUser(content='')
- LongPress(box=(x1,y1))
- PressBack()
- PressHome()
- PressEnter()
- PressRecent()

User Task
{problem}

Previous Actions
{previous_actions}

Output Format
<think> your thinking process </think>
<action> the next action </action>
<conclusion> the conclusion about the next action </conclusion>

Instruction
- Make sure you understand the task goal to avoid wrong actions.
- Make sure you carefully examine the the current screenshot. Sometimes the summarized history might not be reliable, over-claiming some effects.
- For requests that are questions (or chat messages), remember to use the 'CallUser' action to reply to user explicitly before finishing! Then, after you have replied, use the Finished action if the goal is achieved.
- Consider exploring the screen by using the 'scroll' action with different directions to reveal additional content.
- To copy some text: first select the exact text you want to copy, which usually also brings up the text selection bar, then click the 'copy' button in bar.
- To paste text into a text box, first long press the text box, then usually the text selection bar will appear with a 'paste' button in it.

A.4 Web

Web Prompt

You are a GUI Agent.
Your task is to analyze a given user task, review current screenshot and previous actions, and determine the next action to complete the task.

Available Actions
You may execute one of the following functions:
- Click(box=(x1,y1))
- Drag(start=(x1,y1), end=(x2,y2))
- Scroll(direction='down or up')
- Type(content='')
- Launch(url='')
- Wait()
- Finished(content='')

- CallUser(content='')
- LongPress(box=(x1,y1))
- PressBack()
- PressHome()
- PressEnter()
- PressRecent()
- Hover(box=(x1,y1))
- DoubleClick(box=(x1,y1))
- Hotkey(keys=['ctrl', 'c']) # Split keys with comma and wrap each key in single quotes. Do not use more than 3 keys in one Hotkey action.

User Task
{problem}

Previous Actions
{previous_actions}

Output Format
<think> your thinking process </think>
<action> the next action </action>
<conclusion> the conclusion about the next action </conclusion>

Instruction
- Make sure you understand the task goal to avoid wrong actions.
- Make sure you carefully examine the the current screenshot. Sometimes the summarized history might not be reliable, over-claiming some effects.
- For requests that are questions (or chat messages), remember to use the 'CallUser' action to reply to user explicitly before finishing! Then, after you have replied, use the Finished action if the goal is achieved.
- Consider exploring the screen by using the 'scroll' action with different directions to reveal additional content.

A.5 Chinese APPs Prompt

Chinese APP Prompt

你是一个手机图形界面智能体代理
你的任务是根据历史操作和当前设备状态去执行一系列操作来完成用户的任务。

你可以用的操作以及对应功能如下: - Click(box=(x1,y1))
»点击操作，点击屏幕上的指定位置。坐标区间从左上角(0,0)到右下角(999,999)。
- Drag(start=(x1,y1), end=(x2,y2))
»拖动操作，从起始坐标长按数秒之后拖动到结束坐标。用于调整app布局，滑动滑块验证码等。
- Scroll(start=(x1,y1), end=(x2,y2))
»滑动操作，从起始坐标拖动到结束坐标。用于滚动查找内容，切换选项卡，下拉通知栏等。坐标区间从左上角(0,0)到右下角(999,999)。
- Type(content='')
»输入操作，在当前激活的输入框输入指定内容。

- Launch(app='')
»启动目标app。当目标app在当前界面不可见时，可以使用该动作打开app。
- Wait()
»等待页面加载。
- Finished(content='')
»任务结束，退出设备接管。
- CallUser(content='')
»回答用户的问题或者当前界面有多个符合要求的选项时需要用户接管。
- LongPress(box=(x1,y1))
»长按操作，在指定位置长按一定的时间。该操作可以触发更多功能选项，例如复制、转发消息，删除等。坐标区间从左上角(0,0)到右下角(999,999)。
- PressBack()
»返回上一个界面，一般用于错误回退或继续执行剩余任务。
- PressHome()
»返回系统桌面，一般用于跨app任务中快速打开下一个app或遇到严重错误时回退到系统桌面。
- PressEnter()
»回车操作，用于换行或者在搜索框中输入内容之后执行搜索操作。
- PressRecent()
»打开系统后台界面。

用户任务
{problem}

先前的动作和推理过程
{previous_actions}

输出格式
<think>你的思考过程</think>
<action>执行的操作</action>
<conclusion>总结当前操作</conclusion>

Instruction
-输入内容之前，确保输入框已经被激活（出现键盘或者 'ADB Keyboard ON'字样代表输入框已经激活）。
-在app内找不到任务要求的入口时，尝试使用搜索功能，或者如果当前页面上方存在选多个项卡，尝试使用Scroll操作查看。
-如果在执行任务的过程中进入到和任务无关的界面，使用PressBack进行回退。
-任务结束之前，确保已经完整准确地完成用户的任务，如果存在漏做、错做的内容，需要返回重新执行。

B Experiment Details of All Grounding Benchmarks

Note that in OSWorld-G, we re-calculate the performance of UI-Venus-1.0 with the refusal task, and thus, its results are different from our previous report.

C Experiment Details of All Navigation Benchmarks

Models	Basic Tasks				Advanced Tasks				Overall
	Element	Visual	Spatial	Avg	Reasoning	Functional	Refusal	Avg	
General VLMs									
Qwen3-VL-2B* (Bai et al., 2025a)	66.6	79.7	61.3	68.2	12.6	41.6	0.0	15.9	45.2
Qwen3-VL-8B* (Bai et al., 2025a)	73.9	83.8	75.5	76.8	22.6	61.3	6.8	27.3	55.1
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	68.0	84.3	69.2	72.3	19.9	58.6	11.3	27.0	52.4
GUI-specific Models									
OpenCUA-7B (Wang et al., 2025b)	62.23	84.39	67.44	69.15	21.32	49.14	0.00	21.43	48.20
OpenCUA-32B (Wang et al., 2025b)	65.49	78.55	68.80	69.64	29.09	51.00	0.00	25.08	50.08
GTA1-7B (Yang et al., 2025)	63.73	76.64	57.05	64.87	23.31	51.14	0.00	22.75	46.38
GTA1-32B (Yang et al., 2025)	75.36	88.08	76.77	78.87	38.84	67.14	0.00	33.25	58.84
UI-Venus-1.0-7B (Gu et al., 2025)	64.30	78.78	67.15	68.66	24.39	53.85	0.00	23.90	49.01
UI-Venus-1.0-72B (Gu et al., 2025)	81.58	91.30	78.81	83.12	46.16	68.86	51.33	53.75	70.23
Holo2-8B* (H-Company, 2025)	71.4	85.8	77.9	76.8	34.0	63.1	0.0	30.2	56.4
Holo2-30B-A3B* (H-Company, 2025)	78.1	89.7	81.0	81.8	32.2	68.7	0.0	31.0	59.5
Step-GUI-4B* (Yan et al., 2025)	73.9	81.8	77.9	77.0	26.1	59.0	0.0	25.9	54.6
MAI-UI-2B* (Zhou et al., 2025b)	72.8	87.1	76.6	77.4	27.3	62.3	0.0	27.3	55.4
MAI-UI-8B* (Zhou et al., 2025b)	81.3	90.8	84.5	84.5	**55.4**	69.1	0.0	40.5	65.2
Ours									
UI-Venus-1.5-2B	79.4	85.8	80.9	81.4	22.0	57.3	**76.3**	49.2	67.3
UI-Venus-1.5-8B	84.2	93.1	84.9	86.6	38.1	**70.1**	61.6	54.2	72.3
UI-Venus-1.5-30B-A3B	**85.1**	**93.2**	**86.4**	**87.5**	41.8	68.1	73.1	**59.0**	**75.0**

Table 9 Performance comparison on **VenusBench-GD**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Model	CAD		Dev		Creative		Scientific		Office		OS		Avg.		
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	**Avg.**
General VLMs															
Seed1.8 (Seed, 2025a)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	64.3
Qwen3-VL-2B* (Bai et al., 2025a)	27.9	10.9	57.1	9.7	58.1	16.8	62.5	22.7	73.4	34.0	55.1	19.1	55.0	17.4	40.6
Qwen3-VL-8B* (Bai et al., 2025a)	56.9	10.9	75.3	22.8	68.2	16.1	78.5	32.7	80.8	39.6	71.0	20.2	71.1	22.8	52.7
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	51.8	15.6	76.0	24.8	69.2	20.3	76.4	27.3	80.8	37.7	75.7	38.2	70.6	26.3	53.7
GUI-specific Models															
OpenCUA-7B (Wang et al., 2025b)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	50.0
OpenCUA-32B (Wang et al., 2025b)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	55.3
OpenCUA-72B (Wang et al., 2025b)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	60.8
GTA1-7B (Yang et al., 2025)	66.9	20.7	62.6	18.2	53.3	17.2	76.4	31.8	82.5	50.9	48.6	25.9	65.5	25.2	50.1
GTA1-32B (Yang et al., 2025)	**83.1**	37.9	72.2	25.9	70.1	31.3	84.7	39.1	89.3	64.2	76.6	51.7	78.9	38.9	63.6
GUI-Owl-7B (Ye et al., 2025)	64.5	21.9	76.6	31.0	59.6	27.3	79.1	37.3	77.4	39.6	59.8	33.7	-	-	54.9
GUI-Owl-32B (Ye et al., 2025)	62.4	28.1	84.4	39.3	65.2	18.2	82.6	39.1	81.4	39.6	70.1	36.0	-	-	58.0
UI-Venus-1.0-7B (Gu et al., 2025)	60.4	21.9	74.7	24.1	63.1	14.7	76.4	31.8	75.7	41.5	49.5	22.5	67.1	24.3	50.8
UI-Venus-1.0-72B (Gu et al., 2025)	66.5	29.7	84.4	33.1	73.2	30.8	84.7	42.7	83.1	60.4	75.7	36.0	77.4	36.8	61.9
Holo2-8B (H-Company, 2025)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	58.9
Holo2-30B-A3B (H-Company, 2025)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	66.1
Step-GUI-4B (Yan et al., 2025)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	60.0
Step-GUI-8B (Yan et al., 2025)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	62.6
MAI-UI-2B (Zhou et al., 2025b)	61.4	23.4	76.6	32.4	69.2	21.7	81.2	34.5	85.9	39.6	68.2	41.6	-	-	57.4
MAI-UI-8B (Zhou et al., 2025b)	72.6	35.9	83.8	52.4	76.3	33.6	79.9	37.3	88.7	60.4	76.6	49.4	-	-	65.8
MAI-UI-32B (Zhou et al., 2025b)	70.1	**45.3**	86.4	40.7	**82.8**	37.8	**91.7**	46.4	90.4	**71.7**	78.5	34.8	-	-	67.9
Ours															
UI-Venus-1.5-2B	54.3	32.8	70.1	43.4	63.6	28.7	76.4	38.2	81.9	47.2	73.8	51.7	69.1	39.4	57.7
UI-Venus-1.5-8B	75.1	31.2	85.7	54.5	75.3	32.9	86.1	44.5	**92.7**	66.0	82.2	52.8	**82.4**	45.9	68.4
UI-Venus-1.5-30B-A3B	70.6	40.6	**87.7**	**57.9**	75.8	**41.3**	84.0	**47.3**	89.8	69.8	**83.2**	**56.2**	81.2	**51.0**	**69.6**

Table 10 Performance comparison on **ScreenSpot-Pro**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Models	Mobile		Desktop		Web		Avg
	Text	Icon/Widget	Text	Icon/Widget	Text	Icon/Widget	
General VLMs							
Qwen3-VL-2B* (Bai et al., 2025a)	94.1	80.6	94.8	74.3	89.7	72.9	85.6
Qwen3-VL-8B* (Bai et al., 2025a)	**99.7**	87.7	94.8	83.6	95.3	85.7	92.1
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	99.0	87.7	95.4	82.9	95.3	83.7	91.7
GUI-specific Models							
OpenCUA-7B (Wang et al., 2025b)	-	-	-	-	-	-	92.3
OpenCUA-32B (Wang et al., 2025b)	-	-	-	-	-	-	93.4
OpenCUA-72B (Wang et al., 2025b)	-	-	-	-	-	-	92.9
GTA1-7B (Yang et al., 2025)	99.0	88.6	94.9	89.3	92.3	86.7	92.4
GTA1-32B (Yang et al., 2025)	**99.7**	90.5	99.0	**94.3**	95.7	90.1	95.2
GUI-Owl-7B (Ye et al., 2025)	99.0	92.4	96.9	85.0	93.6	85.2	92.8
GUI-Owl-32B (Ye et al., 2025)	98.6	90.0	97.9	87.8	94.4	86.7	93.2
UI-Venus-1.0-7B (Gu et al., 2025)	99.0	90.0	97.0	90.7	96.2	88.7	94.1
UI-Venus-1.0-72B (Gu et al., 2025)	**99.7**	93.8	95.9	90.0	96.2	92.6	95.3
Holo2-8B (H-Company, 2025)	-	-	-	-	-	-	93.2
Holo2-30B-A3B (H-Company, 2025)	-	-	-	-	-	-	94.9
Step-GUI-4B (Yan et al., 2025)	-	-	-	-	-	-	93.6
Step-GUI-8B (Yan et al., 2025)	-	-	-	-	-	-	95.1
MAI-UI-2B (Zhou et al., 2025b)	99.3	87.2	97.4	88.6	94.0	84.7	92.5
MAI-UI-8B (Zhou et al., 2025b)	99.3	89.1	99.0	92.1	97.9	91.1	95.2
MAI-UI-32B (Zhou et al., 2025b)	99.0	92.9	**99.5**	93.6	97.4	**94.6**	**96.5**
Ours							
UI-Venus-1.5-2B	98.6	91.0	93.3	92.9	93.2	85.7	92.8
UI-Venus-1.5-8B	99.3	92.9	96.4	92.9	**98.3**	93.1	95.9
UI-Venus-1.5-30B-A3B	99.3	**94.8**	95.9	**94.3**	97.9	93.1	96.2

Table 11 Performance comparison on **ScreenSpot-V2**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Model	Windows		MacOS		Linux		iOS		Android		Web		Avg.
	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	
General VLMs													
Qwen3-VL-2B* (Bai et al., 2025a)	82.3	41.2	79.1	45.4	67.5	44.4	92.0	68.2	91.6	69.6	85.8	52.9	69.5
Qwen3-VL-8B* (Bai et al., 2025a)	88.6	62.5	86.1	66.8	72.8	57.1	95.9	83.9	95.8	84.8	94.8	72.7	81.4
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	87.8	69.9	85.2	68.8	78.0	60.7	96.5	84.5	96.3	88.5	96.5	78.6	83.7
GUI-specific Models													
GUI-Owl-7B (Ye et al., 2025)	86.4	61.8	81.7	64.5	74.4	61.7	94.9	83.0	95.8	83.7	93.2	72.7	80.5
GUI-Owl-32B (Ye et al., 2025)	85.6	65.1	84.9	67.1	77.0	63.3	95.2	85.5	96.1	87.0	95.5	80.8	83.0
Holo2-8B (H-Company, 2025)	90.8	70.2	87.5	71.4	78.5	60.2	96.2	88.2	96.3	87.9	95.5	77.9	84.5
Holo2-30B-A3B (H-Company, 2025)	91.9	72.8	88.1	74.9	84.3	67.3	96.5	89.7	96.3	90.1	96.5	82.5	86.8
Step-GUI-4B (Yan et al., 2025)	-	-	-	-	-	-	-	-	-	-	-	-	84.0
Step-GUI-8B (Yan et al., 2025)	-	-	-	-	-	-	-	-	-	-	-	-	85.6
MAI-UI-2B (Zhou et al., 2025b)	84.9	64.0	89.3	72.5	75.4	60.2	95.2	85.2	96.3	84.2	92.9	76.0	82.6
MAI-UI-8B (Zhou et al., 2025b)	92.3	74.3	90.7	86.4	81.2	67.3	**97.1**	90.0	97.5	92.7	95.8	86.0	88.8
MAI-UI-32B (Zhou et al., 2025b)	**93.0**	**78.7**	**92.8**	**87.6**	**86.9**	**77.6**	**97.1**	92.4	**98.0**	93.2	96.1	**92.5**	**91.3**
Ours													
UI-Venus-1.5-2B	88.2	61.8	82.3	65.6	77.0	60.7	92.7	80.3	93.3	83.9	94.8	72.1	80.3
UI-Venus-1.5-8B	92.6	74.6	86.1	82.1	84.3	67.3	**97.1**	89.4	96.9	92.4	**96.8**	86.0	88.1
UI-Venus-1.5-30B-A3B	91.5	76.5	88.1	76.6	85.9	69.9	96.5	**93.0**	97.2	**93.8**	96.5	87.7	88.6

Table 12 Performance comparison on **MMbench-GUI-L2**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Models	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Refusal	Avg
General VLMs						
Qwen3-VL-2B* (Bai et al., 2025a)	60.9	49.7	57.3	38.9	0.0	47.7
Qwen3-VL-8B* (Bai et al., 2025a)	71.6	59.4	61.3	49.7	1.9	57.4
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	73.9	65.2	67.2	51.0	5.6	61.2
GUI-specific Models						
OpenCUA-7B (Wang et al., 2025b)	-	-	-	-	-	55.3
OpenCUA-32B (Wang et al., 2025b)	-	-	-	-	-	59.6
GTA1-7B (Yang et al., 2025)	42.1	65.7	62.7	56.1	0.0	55.1
GTA1-32B (Yang et al., 2025)	63.2	**78.4**	73.3	**65.2**	0.0	65.2
GUI-Owl-7B (Ye et al., 2025)	64.8	63.6	61.3	41.0	-	55.9
GUI-Owl-32B (Ye et al., 2025)	67.0	64.5	67.2	45.6	-	58.0
UI-TARS-1.5-7B (Seed, 2025b)	-	-	-	-	-	52.8
UI-Venus-1.0-7B (Gu et al., 2025)	74.6	60.5	61.5	45.5	-	54.6
UI-Venus-1.0-72B (Gu et al., 2025)	**82.1**	71.2	70.7	64.4	-	62.2
Holo2-8B* (H-Company, 2025)	74.3	68.2	67.6	59.1	0.0	63.5
Holo2-30B-A3B* (H-Company, 2025)	77.0	68.8	70.0	59.7	0.0	65.2
Step-GUI-4B* (Yan et al., 2025)	70.1	65.8	67.6	53.0	0.0	60.5
MAI-UI-2B (Zhou et al., 2025b)	62.8	56.7	59.3	40.3	-	52.0
MAI-UI-8B (Zhou et al., 2025b)	72.0	63.3	66.0	51.0	-	60.1
MAI-UI-32B (Zhou et al., 2025b)	73.6	72.4	73.9	57.7	-	67.6
Ours						
UI-Venus-1.5-2B	67.4	66.1	66.4	44.3	7.4	59.4
UI-Venus-1.5-8B	79.7	76.1	72.3	60.4	**22.2**	69.7
UI-Venus-1.5-30B-A3B	80.1	76.1	**75.1**	61.1	9.3	**70.6**

Table 13 Performance comparison on **OS-World-G**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Models	Text Matching	Element Recognition	Layout Understanding	Fine-grained Manipulation	Refusal	Avg
General VLMs						
Qwen3-VL-2B* (Bai et al., 2025a)	73.2	64.2	70.8	47.0	0.0	60.6
Qwen3-VL-8B* (Bai et al., 2025a)	78.2	71.5	72.3	55.7	1.9	67.0
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	77.8	75.8	74.7	54.4	5.6	69.3
GUI-specific Models						
OpenCUA-32B (Wang et al., 2025b)	63.2	79.9	**84.9**	62.1	7.4	70.2
GTA1-7B (Yang et al., 2025)	63.2	82.1	74.2	**70.5**	0.0	67.7
GTA1-32B (Yang et al., 2025)	63.2	**83.6**	84.4	**70.5**	0.0	72.2
UI-Venus-1.0-7B (Gu et al., 2025)	74.6	60.5	61.5	45.5	-	58.8
UI-Venus-1.0-72B (Gu et al., 2025)	82.1	71.2	70.7	64.4	-	70.4
Holo2-8B* (H-Company, 2025)	-	-	-	-	-	70.1
Holo2-30B-A3B* (H-Company, 2025)	-	-	-	-	-	76.1
Step-GUI-4B* (Yan et al., 2025)	-	-	-	-	-	66.9
MAI-UI-2B (Zhou et al., 2025b)	70.9	69.1	72.7	47.7	-	63.5
MAI-UI-8B (Zhou et al., 2025b)	77.4	73.0	78.3	55.7	-	68.6
MAI-UI-32B (Zhou et al., 2025b)	79.7	79.4	81.0	61.7	-	73.9
Ours						
UI-Venus-1.5-2B	75.1	70.0	73.5	52.3	7.4	65.6
UI-Venus-1.5-8B	82.4	81.5	80.2	59.7	**22.2**	74.1
UI-Venus-1.5-30B-A3B	**83.1**	82.1	83.4	65.8	9.3	**76.4**

Table 14 Performance comparison on **OS-World-G-Refine**. For each benchmark, the best and second-best performing models are indicated in **bold** and underlined, respectively. Asterisk (*) indicates results that may require verification with original sources.

Models	Basic	Functional	Spatial	Avg
General VLMs				
Qwen3-VL-2B* (Bai et al., 2025a)	16.4	19.1	4.6	13.1
Qwen3-VL-8B* (Bai et al., 2025a)	27.8	29.6	9.4	21.9
Qwen3-VL-30B-A3B* (Bai et al., 2025a)	31.2	31.9	14.6	25.6
GUI-specific Models				
OpenCUA-7B (Wang et al., 2025b)	-	-	-	29.7
OpenCUA-32B (Wang et al., 2025b)	-	-	-	33.3
OpenCUA-72B (Wang et al., 2025b)	-	-	-	37.3
UI-Venus-1.0-7B (Gu et al., 2025)	36.1	32.8	11.9	26.5
UI-Venus-1.0-72B (Gu et al., 2025)	45.6	42.3	23.7	36.8
Holo2-8B* (H-Company, 2025)	43.6	43.5	19.7	35.1
Holo2-30B-A3B* (H-Company, 2025)	51.0	50.1	23.2	40.9
Step-GUI-4B* (Yan et al., 2025)	39.2	36.5	15.7	30.0
MAI-UI-2B (Zhou et al., 2025b)	41.0	41.2	10.4	30.3
MAI-UI-8B (Zhou et al., 2025b)	51.7	49.6	22.5	40.7
MAI-UI-32B (Zhou et al., 2025b)	59.1	<u>57.1</u>	26.9	<u>47.1</u>
Ours				
UI-Venus-1.5-2B	<u>63.5</u>	51.5	21.6	44.8
UI-Venus-1.5-8B	56.3	52.4	<u>32.0</u>	46.5
UI-Venus-1.5-30B-A3B	**69.0**	**59.3**	**37.4**	**54.7**

Table 15 Performance comparison on **UI-Vision**. For each benchmark, the best and second-best performing models are indicated in **bold** and <u>underlined</u>, respectively.Asterisk (*) indicates results that may require verification with original sources.

Table 16 Performance comparison on **VenusBench-Mobile**. The best performing model is indicated in **bold**.

Agent	FA	CF	VA	MR	GSA	GUIM	HGB	NR	BC	Total
General VLMs										
Qwen3-VL-8B Bai et al. (2025a)	18.2	4.6	18.8	0.0	0.0	0.0	0.0	6.3	10.0	6.7
Qwen3-VL-30B-A3B Bai et al. (2025a)	22.7	4.6	18.8	0.0	0.0	0.0	5.9	6.3	10.0	8.7
GUI-specific Models										
UI-Venus-7B Gu et al. (2025)	13.6	4.6	25.0	0.0	10.0	0.0	0.0	18.8	0.0	8.1
UI-Venus-72B Gu et al. (2025)	22.7	4.6	12.5	0.0	10.0	0.0	**17.7**	**50.0**	0.0	15.4
GUI-Owl-7B Ye et al. (2025)	13.6	0.0	18.8	0.0	0.0	11.1	2.9	12.5	0.0	6.7
MA3(GUI-Owl-7B) Ye et al. (2025)	18.2	9.1	6.3	0.0	0.0	0.0	11.8	31.3	**20.0**	12.1
MAI-UI-2B Zhou et al. (2025b)	9.1	0.0	18.8	0.0	0.0	0.0	0.0	25.0	10.0	6.7
MAI-UI-8B Zhou et al. (2025b)	9.1	**13.6**	25.0	0.0	10.0	11.1	5.9	31.3	10.0	12.8
Ours										
UI-Venus-1.5-2B	22.7	0.0	12.5	0.0	10.0	11.1	2.9	18.8	0.0	8.7
UI-Venus-1.5-8B	22.7	0.0	25.0	0.0	0.0	**22.2**	8.8	**50.0**	**20.0**	16.1
UI-Venus-1.5-30B-A3B	**40.9**	0.0	**37.5**	**10.0**	**20.0**	**22.2**	14.7	43.8	0.0	**21.5**