FROM BENCHMARKS TO THE WILD: PRACTICAL DESIGN CHOICES FOR DEPLOYABLE MOBILE GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent years have witnessed rapid advances in large multimodal model (LMM)-based mobile GUI agents that have achieved strong performance on related benchmarks such as ScreenSpot and AndroidWorld. However, it remains challenging to deploy such benchmark-capable agents in real-world scenarios where they must operate under dynamic UI states, diverse user instructions, third-party application variations, and strict latency budgets that existing benchmarks often underrepresent. To investigate how to bridge this gap, we present an empirical study of practical mobile GUI agent design through deployment-oriented experiments across Android and HarmonyOS, covering 50+ mainstream applications. Our analysis focuses on two important dimensions: data construction strategies that make agents robust beyond benchmark-like settings, and latency-aware designs that keep accurate agents responsive as well. For robustness, to handle the wide variation of user inputs and environment states in real use, we propose three complementary data construction schemes that reflect diverse, realistic workloads and transfer GUI agents from fixed benchmarks to dynamic scenarios, improving real-world task success rate by 10.2 percentage points. For efficiency, we find that heavy reasoning or multi-agent workflows emphasized by existing agent paradigms can introduce unacceptable latency; by streamlining agent input/output and workflows, we identify the Pareto-optimal inference-time designs for balancing task accuracy and latency. Together, these findings provide practical guidelines for building mobile GUI agents that are robust, efficient, and deployable in real-world environments.

1 INTRODUCTION

Graphical User Interface (GUI) agents are automation systems that translate user intents into iterative GUI interactions, offering a general way for controlling diverse applications through natural language. Recent advances in large multimodal models (LMMs) and SFT/RL algorithms have substantially accelerated the progress in mobile GUI agents, ranging from app-specific to general-purpose assistants (Wu et al., 2025; Qin et al., 2025; Bai et al., 2024; Lu et al., 2025b). This progress has been reflected in increasingly strong results on standardized benchmarks that evaluate core capabilities such as GUI grounding and navigation, including ScreenSpot, Mind2Web, AndroidWorld, and OSWorld (Cheng et al., 2024; Deng et al., 2023; Xie et al., 2024; Rawles et al., 2025).

However, deploying GUI agents in real-world mobile environments still presents substantial challenges that static or virtual benchmarks often abstract away. Our study is motivated by a large-scale commercial deployment, where our mobile GUI agent has been integrated into flagship smartphones which serve over a million users. The deployment produces anonymized production traces and user feedback that reveal failure modes difficult to anticipate from curated benchmark tasks alone: agents that appear capable in standardized evaluations frequently fail when UI content changes dynamically, user instructions vary in granularity and wording, and third-party applications expose a long tail of platform- or version-specific interface designs. Furthermore, real-world mobile deployment imposes strict constraints on computational overhead and execution latency: in production, service-level objectives (SLOs) require not only high task success rates but also a 1.5-second single-step response time for interactive actions. An agent burdened by high-resolution visual processing or lengthy

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

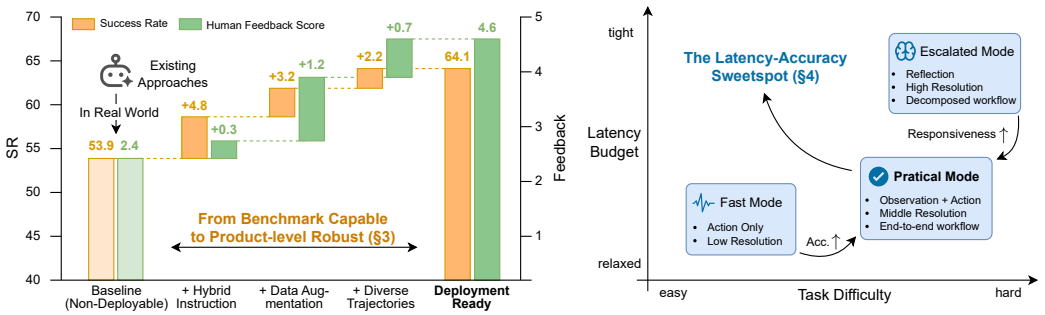


Figure 1: Our contribution: **Left:** turning benchmark-capable GUI agents into deployment-ready ones through robust data construction (Human Feedback Score: 1–5 scale from 1,000+ user questionnaires). **Right:** identifying the practical latency-accuracy sweet spot of GUI agent inference-time design.

multi-step reasoning chains may achieve high benchmark scores, yet prove impractical for actual users due to excessive per-step latency.

To ensure that advances in GUI agent capabilities demonstrated on existing benchmarks further translate into real-world utility, we ask a fundamental question: *how should one make the key design choices to build mobile GUI agents that work reliably in real-world environments?* To this end, we present a systematic empirical study that examines the end-to-end development of such agents under deployment-oriented experiments, spanning Android and HarmonyOS across 54 mainstream applications, and analyze two central tensions: building agents that remain reliable beyond benchmark-like settings (Section 3) and making accurate agents responsive enough for practical interaction (Section 4). As illustrated in Figure 1, our study improves the real-world task success rate by 10.2 percentage points and user rating collected from product feedback by 91.7%; we also identify the optimal inference-time designs to achieve both accuracy and responsiveness.

Our investigation yields a set of actionable insights for mobile GUI agent development:

- **Bridging benchmark capability and production-level robustness:** We identify why agents that perform well in existing benchmarks still fail in real usage: dynamic environment states, diverse user intents, and the long tail of third-party application behavior require training data that reflects real-world variability, rather than only clean, standardized demonstrations. Our study finds that (1) trajectory diversity trades ID accuracy for OOD transfer and robustness, therefore suboptimal but reasonable execution paths should be prioritized in data collection while sufficient canonical and optimal paths should be preserved, (2) injecting randomized contextual noise and runtime anomalies into training data reduces history-action memorization and enhances robustness in dynamic environments, and (3) training with high-level-biased hybrid instruction granularity is crucial as high-level goals require more supervision for flexible knowledge transfer while low-level instructions remain essential for instruction-following scenarios such as skill use (Section 3).

- **Balancing task accuracy and execution latency:** Existing mobile GUI agents are mainly optimized for task accuracy while the runtime overhead is often overlooked. However, for real-world deployment, the inference-time tension between task success and per-step latency must be carefully considered to achieve practical utility, especially when interactive systems must satisfy strict latency SLOs rather than only accuracy targets. Our experiments show that (1) observation-based reasoning offers the best accuracy-latency balance compared to overly simplified action-only reasoning or heavy reflection-style reasoning, (2) screenshot downsampling with a moderate resize factor of 0.4-0.6 preserves most accuracy gains at substantially lower latency, and (3) the end-to-end single-model workflow remains more practical than decomposed multi-agent workflows for most deployment settings (Section 4).

Overall, this paper moves beyond isolated benchmark performance toward deployment-oriented design for mobile GUI agents. We provide a concrete blueprint for building agents that are trained to handle real-world variability and tuned to remain responsive within practical latency budgets.

2 PRELIMINARIES

Before detailing our empirical study, we establish the foundational concepts and experimental framework that underpin our investigation. This section reviews related work and describes the experimental setup, including trajectory collection and the training/evaluation protocols.

2.1 RELATED WORK

GUI agents for interactive environments. Recent work on GUI agents spans web, desktop, and mobile interactive environments. One prominent line treats GUI control as prompted decision making, where large language or multimodal models are guided by demonstrations, memory, or grounding-aware prompting to act on web or computer interfaces (Zheng et al., 2024a; He et al., 2024; Zheng et al., 2024b; Yang et al., 2023; Liu et al., 2024b). Another line emphasizes model training and screenshot-centric perception, aiming to align visual understanding with executable actions through grounded supervision and end-to-end vision-language-action modeling, as illustrated by CogAgent, UI-TARS, ShowUI, and other GUI models (Hong et al., 2024; Qin et al., 2025; Lin et al., 2025; You et al., 2024; Li et al., 2025b; Yang et al., 2025; Xu et al., 2025; Chen et al., 2025; Zhang & Zhang, 2024; Zhou et al., 2025; Gu et al., 2025). Existing systems also vary in system organization and action formulation, adopting modular or decomposed pipelines that separate perception, planning, grounding, memory, or execution, as seen in desktop and mobile agents such as UFO, AppAgent, the Mobile-Agent series, and other GUI automation systems (Zhang et al., 2025a;c; Wang et al., 2024b;a; Wen et al., 2024; Niu et al., 2024; Wu et al., 2024; Liu et al., 2024a; 2025).

Benchmarks and evaluation for GUI agents. Benchmarks for GUI agents differ in their emphasis on grounding or offline/online action prediction. Grounding benchmarks evaluate whether models can localize UI elements from natural language, e.g., ScreenSpot and ScreenSpot-Pro (Cheng et al., 2024; Li et al., 2025a). Offline benchmarks instead focus on next action prediction along recorded traces, including Mind2Web for web tasks, Android in the Wild for Android control, alongside cross-app, dialogue-driven, and instruction-tuned navigation datasets (Deng et al., 2023; Rawles et al., 2023; Deng et al., 2024; Chai et al., 2025; Lu et al., 2025a; Furuta et al., 2024; Lù et al., 2024; Li et al., 2024). Online environments place greater emphasis on interaction in real/simulated environments, as exemplified by WebArena, OSWorld, and AndroidWorld (Kong et al., 2025; Koh et al., 2024; Bonatti et al., 2024; Cao et al., 2024; Yao et al., 2022; Ma et al., 2024; Wang et al., 2024c; Trivedi et al., 2024; Xu et al., 2024). Taken together, these resources provide complementary views of GUI-agent capability, spanning UI grounding, offline action prediction, and fully interactive task execution.

2.2 EXPERIMENTAL SETUP

Trajectory collection. All task trajectories used for training and evaluation are collected using the MobiAgent trajectory recording toolkit (Zhang et al., 2025b), which operates at the OS level to capture complete interaction traces on real devices. The collected trajectories span 54 mainstream applications covering high-frequency categories accounting for approximately 86.7% of users' aggregate application usage time in our production statistics. We adopt a human–LLM hybrid collection protocol: an LLM agent powered by M3A (Rawles et al., 2025) autonomously executes and records simple, short-horizon tasks, while human annotators handle complex, long-horizon tasks that LLMs cannot reliably complete. Each recorded trajectory contains the action type and its parameters (e.g., tap coordinates, typed text, scroll direction), and a screenshot of the device display at every step.

Training protocol. We adopt Qwen3-VL-8B (Bai et al., 2025) as our base vision-language model across all experiments as its parameter size is preferred in commercial deployment for balancing accuracy and inference speed (see Section 5). We adopt a Reinforcement Fine-Tuning (RFT) pipeline with two stages: (1) a 2-epoch SFT stage with a batch size of 64 and a learning rate of 5×10^{-6}, and (2) a 1-epoch offline rule-based RL stage using the DAPO algorithm (Yu et al., 2025) with a learning rate of 1×10^{-6}, a group size of 16, and a batch size of 128. All RFT runs are conducted on NVIDIA A100 GPUs. Unless explicitly stated, each training run involves 8,000 training trajectories with the following settings as default: diverse trajectories, no data augmentation, 2:1 mixed-level instructions, Observation+Action reasoning, a resize factor of 0.5, and end-to-end workflow. While

Table 1: Per-category success rate (SR, %) for consistent and diverse trajectories under ID, OOD-transfer, and OOD-variation evaluation. Best result within each setting is bolded.

Evaluation Setting		Social	E-commerce	Video	Productivity	Travel	Food Delivery	Music	Avg.
ID	Consistent	**68.7**	66.7	**86.2**	66.3	**62.5**	**61.8**	**66.7**	**68.1**
	Diverse	66.5	**69.9**	79.6	**68.9**	62.4	61.1	64.0	67.3
OOD-trans	Consistent	47.9	52.6	49.9	53.6	49.4	50.6	52.1	50.7
	Diverse	**54.1**	**58.2**	**56.0**	**58.7**	**54.7**	**56.2**	**57.4**	**56.3**
OOD-var	Consistent	37.2	40.4	38.5	42.5	37.7	39.9	41.4	39.5
	Diverse	**53.1**	**55.9**	**54.3**	**57.4**	**51.9**	**54.4**	**55.4**	**54.5**

the full production dataset contains over 100,000 trajectories, we pick the 8,000 deduplicated, verified subset for controlled comparisons while the accuracy loss is minor.

Evaluation protocol. For comprehensive evaluation, we employ a hybrid evaluation protocol designed to balance efficiency, reproducibility, and coverage of real-world use cases. For simple and common tasks with relatively fixed completion paths, which account for 80% of the evaluation set, we extend MobiFlow (Feng et al., 2026) using its trajectory-fusion algorithm and our collected trajectory data, enabling efficient and reproducible automatic evaluation. For difficult tasks, tasks with many feasible paths, and highly dynamic scenarios that cannot be adequately covered by the UI-state-transition benchmark, which account for the remaining 20%, we evaluate agents through execution on real devices and human judgment. Specifically, 5 annotators score each task according to a specified final state; each score is assigned by one annotator and checked by another, with disagreements resolved by a third annotator via majority vote. We refer to the automatically evaluated 80% subset as the *easy split* and the human-evaluated 20% subset as the *hard split*, and the full evaluation set contains 295 tasks. Model inference in all experiments is deployed on a single NVIDIA A100 GPU using vLLM 0.18.0 (Kwon et al., 2023). We report the following metrics:

- **Success rate (SR)** measures the percentage of tasks that the agent completes successfully under the hybrid automatic–human evaluation protocol.
- **Per-step latency** measures the average wall-clock time that the agent spends on processing the input and generating the next action, capturing deployment efficiency under different inference-time designs.

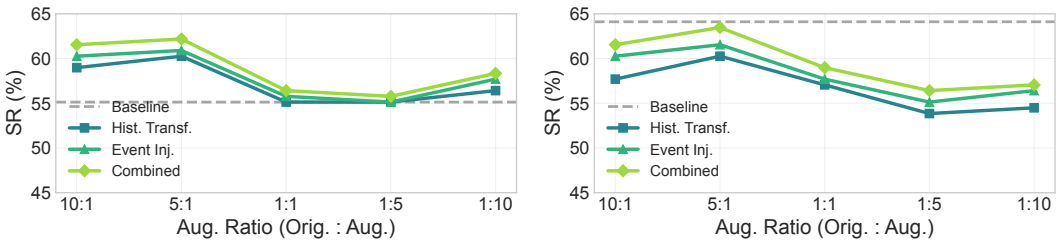
3 Constructing data for robustness beyond benchmarks

To achieve better robustness, a GUI agent must handle unseen task families, changing application contexts, transient UI events, and different forms of user instruction correctly. To develop production-level agents rather than merely benchmark-competent ones, we study three training data construction strategies that directly correspond to these real-world challenges.

3.1 Controlled diversity for out-of-distribution robustness

When collecting task trajectories, practitioners face a trade-off between consistency and diversity. **Consistent trajectories** follow nearly identical execution paths with minimal deviation, typically produced by expert annotators adhering to strict protocols. **Diverse trajectories** span a wide spectrum of execution strategies, including alternative solution paths and exploratory actions that reflect realistic user behavior. Consistent trajectories reduce data noise and produce more stable gradients, enabling faster convergence but risking local optima. Diverse trajectories enhance generalization but introduce noisier supervision that may hinder convergence.

To characterize this trade-off, we evaluate both trajectory types using size-matched datasets (8,000 trajectories). For each task family (tasks sharing the same template but different parameters), we specify an optimal execution protocol, reflecting either the shortest route or the route most consistent with human usage patterns, which is followed by annotators to collect consistent trajectories. For diverse trajectories, we additionally specify two suboptimal but still valid execution protocols, and uniformly select one of the three protocols for the collection of each task instance. We train our base



(a) SR under test-time anomaly injection.

(b) SR without anomaly injection.

Figure 2: Effect of data augmentation on mitigating memorization and enhancing robustness.

model on both datasets and evaluate SR under three conditions: *ID* evaluates on apps covered by training data; *OOD-transfer* evaluates on unseen apps whose categories still appear in training data; and *OOD-variation* evaluates on ID apps with version/platform changes that alter the UI layout or usage patterns (e.g., a search button becoming unavailable in a food-delivery app on HarmonyOS).

As shown in Table 1, consistent trajectories provide a modest ID advantage, improving average SR from 67.3% to 68.1%. This pattern is consistent with the role of cleaner supervision in fitting canonical execution paths. However, diverse trajectories outperform consistent ones under OOD-transfer, raising average SR from 50.7% to 56.3%. This gain suggests that diverse training data leads to stronger generalization across apps that preserve similar task semantics and interaction logic. The SR gap becomes larger under OOD-variation where diverse trajectories improve average SR by 15.0 pp. In this setting, models trained on consistent trajectories appear more tied to a fixed execution pattern, whereas diverse training data makes the agent more likely to identify a feasible path when the nominal route is blocked. Overall, trajectory consistency helps in-distribution fitting with clean canonical demonstrations, whereas controlled diversity (i.e., mixing suboptimal but valid alternatives) is more important for deployment robustness under both app-level transfer and platform-level variation. This suggests that practical data collection should prioritize alternative execution paths while retaining a core set of canonical routes for stable standard-case behavior.

3.2 Noise Injection for State-Aware Agents

During deployment, we identified a failure mode tied to overfitting: agents memorize the mapping from action history to next action, predicting steps based solely on past actions while ignoring the current screenshot. When the screen state deviates from the nominal trajectory due to dynamic content or execution failures, the agent mechanically reproduces the next action associated with the same history prefix in the training data, regardless of the actual screen state. This behavior is more pronounced with consistent trajectories (Section 3.1).

Although collecting a more diverse set of trajectories can partially mitigate this memorization, doing so incurs prohibitive annotation costs. In this section, we develop two data augmentation strategies and study whether simple training data transformations can break the agent's dependence on history and force it to reason about the current screenshot.

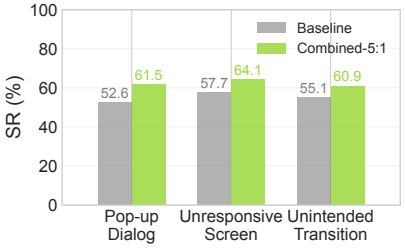


Figure 3: SR of the best augmentation setting under each anomaly type.

History transformation: We randomly perturb the action history during training via *clipping* (removing random actions from the history) and *shifting* (replacing history entries with older screenshot-action pairs), breaking the direct history-to-action mapping.

Event injection: We record screenshots on real devices for three representative anomalies: pop-up dialogs, unresponsive screens, and unintended page transitions (e.g., accidental taps, redirects, or interstitial pages). We pair them with the corresponding recovery actions (close, retry, or navigate back) and randomly splice these event-action pairs into normal trajectories during training.

We fix the training data budget at 8,000 trajectories but vary the *augmentation ratio* (original:augmented data) and compare three strategies (History Transformation, Event Injection, and

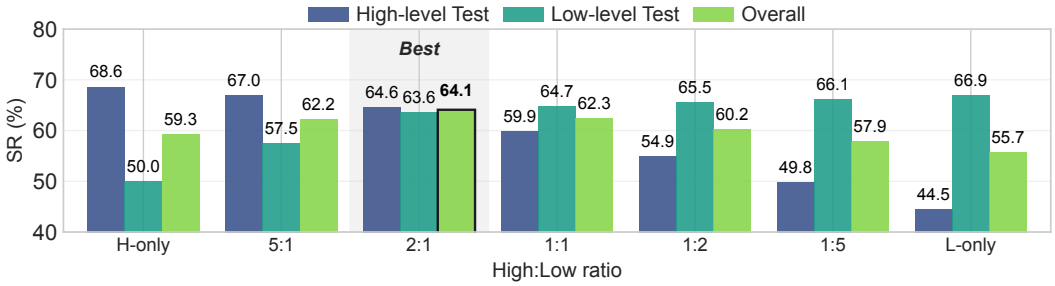


Figure 4: High-level/low-level/overall SR by ratio of high-level:low-level training instructions.

Combined) against an augmentation-free baseline. At test time, we simulate the same anomaly set by dropping actions or replacing input screenshots at a fixed probability, and evaluate the SR.

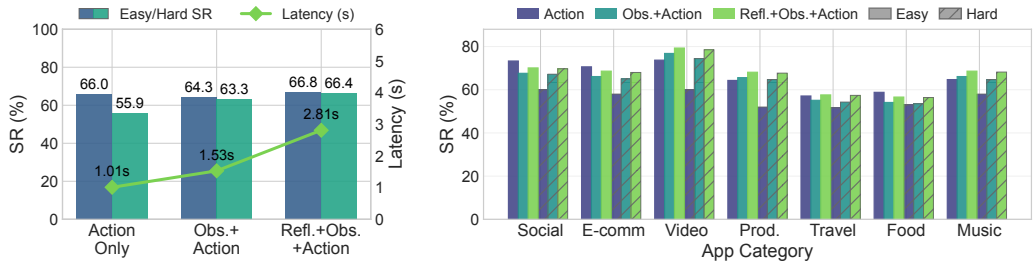
As shown in Figure 2a, the baseline SR drops to 55.1% under test-time anomaly injection. At moderate augmentation ratios (10:1 and 5:1), both augmentation strategies improve SR over the baseline, and their combination yields the largest gains (up to +7.1 pp). However, the benefit saturates as the ratio further increases: SR of 1:1, 1:5, and 1:10 remain near the baseline, indicating that excessive augmentation no longer provides additional robustness. Figure 2b further shows that moderate augmentation preserves performance in anomaly-free scenarios (only -0.6 pp at Combined-5:1), whereas larger augmentation ratios drive SR below the baseline due to the introduced noise. We further evaluate SR of Combined-5:1 under each of the three anomaly types separately. As shown in Figure 3, Combined-5:1 improves robustness across pop-up dialogs, unresponsive screens, and unintended transitions by +8.9, +6.4, and +5.8 pp, respectively. Overall, the History Transformation+Event Injection augmentation at a moderate 5:1 ratio provides the strongest robustness-accuracy trade-off: the augmented samples disrupt the history-to-action shortcut to ensure that the agent focuses on the current UI state, while the retained original trajectories provide sufficient operational knowledge of GUI tasks to preserve the agent's capabilities.

3.3 Tuning goal- and procedure-level supervision

In practice, GUI agents encounter two types of user instructions: **high-level instructions** that specify only abstract goals (e.g., "Send a project status report to my manager") and **low-level instructions** that provide step-by-step procedures (e.g., "Open email, click Compose, type the address, ..., click Send"). A robust agent must handle both instruction types effectively. According to our microbenchmark results and user feedback, general-purpose LLMs perform similarly on both instruction types while UI-automation-specialized models often lack the ability to follow detailed instructions as their training corpora are largely focused on high-level goals. This section investigates how the granularity of training instructions affects model performance.

We construct training sets from the same underlying 8,000 trajectories, differing only in instruction annotation. In addition to high-only and low-only, we vary the high:low training instruction ratio from 5:1 to 1:5. We evaluate all settings with two instruction granularities: high-level test uses high-level goal descriptions, whereas low-level test uses step-by-step instructions for the same tasks.

As shown in Figure 4, high-only training reaches 68.6% SR on high-level test but drops to 50.0% on low-level test, while low-only training reaches 66.9% on low-level test but only 44.5% on high-level test. Mixed training improves the average behavior across both instruction granularities. Among the tested mixtures, the 2:1 high:low ratio achieves the highest overall SR at 64.1% and balanced SR for high-level and low-level tests. Meanwhile, high-level SR drops sharply while low-level SR plateaus after the 2:1 ratio. This asymmetry suggests that high-level competence is more data-demanding. As high-level instructions specify only abstract goals, the agent must infer a less deterministic goal-to-action mapping across diverse user intents and feasible execution paths. Low-level instructions reduce this ambiguity by specifying the procedure and require only a small training share to preserve the base model's instruction-following ability. Nevertheless, procedure-oriented instructions are still crucial for preserving agent capabilities in scenarios such as skill use where agents must follow specified steps. Overall, training instructions for real-world GUI agents should be hybrid but biased toward high-level goals, with the 2:1 high:low ratio providing the best cross-granularity balance, as well as the highest overall accuracy.



(a) SR and step latency of three paradigms. (b) Per-category SR breakdown with easy/hard splits.

Figure 5: The trade-off between execution accuracy and latency influenced by reasoning depth.

4 Taming the Accuracy-Latency Trade-off

In real-world deployment, task accuracy and execution latency jointly determine the utility of a GUI agent. Meanwhile, the strict responsiveness requirements of mobile interaction leave even less room for inefficient inference-time design choices compared to other scenarios. We therefore investigate three inference-time design axes and quantify their effects on accuracy and latency to identify the most cost-effective operating points.

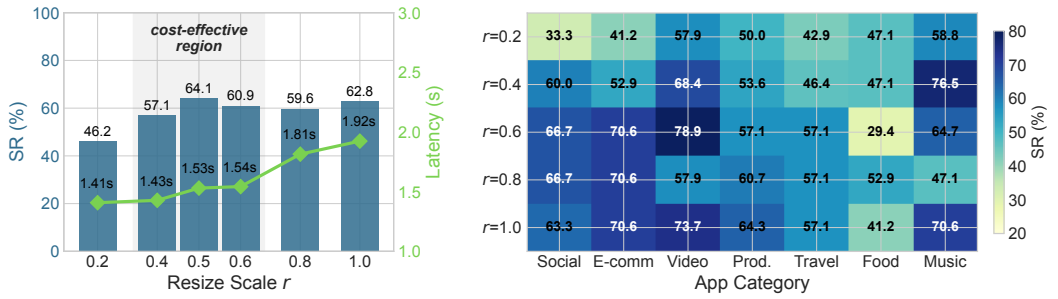
4.1 Calibrating Reasoning Depth

Test-time scaling, which aims to improve model accuracy by consuming additional tokens at inference time, has proven effective across AI applications (Wei et al., 2022; Yao et al., 2023a; Wang et al., 2023; Shinn et al., 2023; Madaan et al., 2023; Guo et al., 2025). For GUI agents, the reasoning at each step can be decomposed into three components: **Reflection** (reasoning about whether the previous action is successful), **Observation** (describing the current GUI state), and **Action** (the semantics of the next action). Combining these yields established reasoning paradigms such as ReAct (Yao et al., 2023b), CoAT (Zhang et al., 2024), and Mobile-Agent-E (Wang et al., 2025). For latency-sensitive mobile interaction, however, the token overhead of complex reasoning chains translates directly into slower per-step responses, degrading user experience. While longer reasoning chains can improve task success rates, they also increase inference time, making the selection of reasoning components to balance accuracy and speed a key design decision for real-world GUI agent deployment.

To empirically evaluate this trade-off, we compare models trained with varying levels of reasoning complexity. Specifically, following ActRe (Yang et al., 2024), we employ an LLM-assisted pipeline to annotate the reasoning part of each action in our trajectory corpus under three styles: **Action Only**, **Observation + Action**, and **Reflection + Observation + Action**, and train a separate model for each. We analyze model performance on easy/hard test splits (introduced in Section 2.2) to examine the usefulness of additional reasoning at a finer granularity. As shown in Figure 5a, on easy tasks, Action Only slightly outperforms Observation+Action, suggesting that simple GUI tasks can be solved by the pre-trained model's inherent ability, which may be instead hindered by training with complex reasoning. This pattern is consistent with recent findings on overthinking in agentic tasks, where excessive reasoning can interfere with effective action selection (Cuadron et al., 2025). In contrast, Observation+Action improves SR from 55.9% to 63.3% on hard tasks, indicating that explicit observation reasoning is most beneficial when tasks are long-horizon, state-dependent, or have non-obvious completion strategies. Adding Reflection provides only marginal additional SR while latency rises sharply, because this paradigm requires two screenshots (before and after the previous action) as input and generates additional reflection tokens. Figure 5b shows a similar trend across categories: lightweight action prediction remains competitive for easy tasks, while reasoning provides advantages on harder tasks. These results suggest that Observation+Action captures most of the benefit of state-aware reasoning while incurring minor token overhead, therefore offering the best accuracy-latency trade-off for most deployment scenarios. Meanwhile, Reflection should be reserved for use cases where tasks are more difficult or latency constraints are relaxed.

4.2 Selecting the Optimal Input Resolution

The resolution of input screenshots is another important axis in the accuracy-latency trade-off. Higher-resolution images preserve fine-grained visual details such as small icons and densely packed UI



(a) SR and step latency across resize scale r.

(b) Per-category SR heatmap across resize scale r.

Figure 6: Characterizing the input resolution trade-off for execution accuracy and latency.

controls, enabling more precise action prediction. However, because the number of visual tokens scales quadratically with the resize scale (i.e., the factor applied to both image width and height relative to the native screen resolution), increasing resolution imposes a steep latency penalty. For mobile GUI agents that must respond within tight interaction latency targets, it is crucial to identify the sweet spot of input image resolution.

To quantify this relationship, we train and evaluate agents across six resolution levels. Our test device has a native screen resolution of 2772×1272; we uniformly downsample both dimensions with resize scales $r \in [0.2, 1.0]$, yielding input sizes that range from roughly 554×254 to the full native resolution. For each resize scale, the same resolution is used during both training and evaluation. As our base model, Qwen3-VL, natively supports dynamic resolution, we freeze the visual encoder and leverage the model's inherent ability to adapt to different resolutions.

As shown in Figure 6a, per-step latency grows consistently from 1.41s to 1.92s with r due to the scaling of input visual tokens. However, the SR exhibits diminishing returns, peaking at 64.1% at r=0.5 and plateauing around 60–63% for $r \geq 0.6$. This saturation arises because additional visual tokens consume a larger share of the context budget while contributing increasingly redundant low-level details. The region $r \in [0.4, 0.6]$, shaded in Figure 6a, offers the most favorable cost-performance balance, capturing the majority of SR gains at a fraction of peak latency. Figure 6b provides a per-category heatmap that reveals nuanced differences: E-commerce and Social apps improve sharply from very low to moderate resolutions, Productivity apps continue to benefit at higher resolution, and Travel apps saturate after r=0.6. These patterns suggest that applications containing fine-grained text or controls benefit from higher fidelity while others with larger visual elements or more repetitive layouts can be handled reliably at smaller resolution. Overall, moderate input image downsampling is a simple yet effective strategy for accelerating mobile GUI agents without sacrificing meaningful accuracy.

4.3 End-to-end vs. decomposed agent workflows

Existing GUI agent workflows fall into two categories. The **end-to-end workflow**, as used in UI-TARS (Qin et al., 2025), produces an executable action including low-level information such as coordinates in a single forward pass. The **decomposed workflow**, as used in SeeAct-V (Gou et al., 2025), splits action prediction into two sequential stages: a planning pass that generates a high-level action description, followed by a grounding pass that maps the description to a specific bounding box, which decreases the complexity of action prediction but requires extra model inference.

To study this trade-off, we train two models with different workflows. The end-to-end model is trained on data that directly maps screenshots to executable actions. The decomposed model is trained on a mixture of planning data (screenshot to action description) and grounding data (screenshot plus action description to coordinates), and performs two sequential forward passes at inference time.

We evaluate both workflows on the easy/hard test splits. As shown in Figure 7, on easy tasks, the end-to-end workflow achieves 64.3% overall SR, outperforming the decomposed workflow at 57.7%. This is because the action description is a lossy interface between two stages of the decomposed workflow, resulting in more grounding errors (e.g., underspecified description grounded to a similar element) which dominate the outcome since easy tasks require straightforward decision-making that both workflows can handle. On hard tasks where planning becomes the bottleneck, the

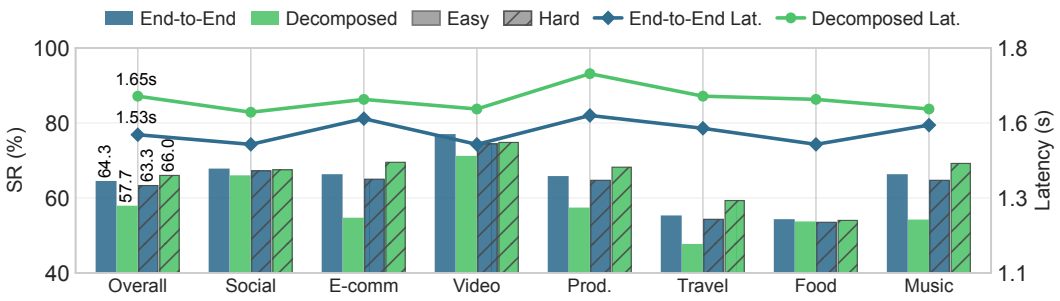


Figure 7: Per-category SR and latency of end-to-end and decomposed workflows.

decomposed workflow improves overall SR from 63.3% to 66.0%. This gain stems from training-time specialization: the decomposed model is supervised only on action reasoning, whereas reasoning and coordinate prediction are co-optimized in the end-to-end model. This joint objective causes gradient interference, thereby degrading the planning accuracy of the end-to-end workflow. The per-step latency of the decomposed workflow increases from 1.53s to 1.65s, which is modest but could matter under stricter latency targets or be amplified by larger model sizes. A consistent pattern can also be observed from the per-category breakdown. These results favor the end-to-end workflow for practical deployment as it avoids an additional model call while preserving competitive SR, making it easier to satisfy latency SLOs under real-world serving constraints. The decomposed workflow may still be useful for hard cases where planning is the dominant bottleneck and the overhead is tolerable.

5 Discussion and Limitations

Base model selection. All experiments use Qwen3-VL-8B as the only base model. This is because its parameter size is preferred in commercial deployment. In vertical domains such as GUI control, larger models exhibit diminishing returns (Hoffmann et al., 2022) while making it difficult to satisfy the strict latency service-level objectives required for interactive phone operation. Conversely, since most GUI agent inference is still served from the cloud, the per-step latency of smaller models is often dominated by communication and hardware underutilization. Consequently, reducing the model size lowers accuracy without yielding improvements in end-to-end response time. In our practice, an 8B-scale model provides the best balance between accuracy, latency, and deployment efficiency.

Evaluation scope. First, our evaluation covers mainstream applications across Android and HarmonyOS, primarily covering the Chinese mobile ecosystem. This is chosen to match the real deployment scenarios that motivate our system. Second, we evaluate screenshot-based agents rather than accessibility tree-based ones. This is because in our use cases, third-party applications developed by heterogeneous frameworks often prevent complete and reliable UI semantics from being exposed in OS-level UI hierarchy/accessibility trees. Screenshot provides the most complete UI semantics, therefore we choose it as the only UI-related input of our agent.

6 Conclusion

We present an empirical study of mobile GUI agents that asks how benchmark-capable agents can be made reliable and responsive in real-world deployment scenarios. Through controlled experiments across Android and HarmonyOS applications, our study shows that the deployability gap is not simply a lack of model capability, but a mismatch between benchmark-centered development and real use: agents must handle varied intents, dynamic UI states, long-tail application behavior, and strict interaction latency budgets. On the data side, robustness should be learned from realistic variation in trajectories, runtime context, and instruction granularity, rather than assumed to emerge from clean demonstrations alone. On the inference side, efficiency should be optimized as part of the agent design: reasoning depth, visual resolution, and agent workflow are useful only when their cost remains compatible with the latency budget. These results provide guidelines for GUI agents in practical use where robustness, responsiveness, and adaptability are jointly necessary.

AI USE STATEMENT

In this work, we used LLMs for short GUI trajectory collection and reasoning annotation, as described in Sections 2.2 and 4.1. We did not use generative AI tools for designing experiments or implementing methods, and other tasks such as formulating theoretical claims and proofs are not applicable to this work. Additionally, we used generative AI tools to polish the manuscript for readability. We take responsibility for the final content of this work, including text and research artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The experimental setup and evaluation protocol are described in Section 2.2. The conditions for the data-construction and inference experiments are detailed in Sections 3 and 4, while the agent prompts and training hyperparameters are provided in Appendices A and B.

REFERENCES

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning, 2024. URL https://arxiv.org/abs/2406.11896.

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, 2025. URL https://arxiv.org/abs/2511.21631.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and Zack Hui. Windows agent arena: Evaluating multi-modal os agents at scale, 2024. URL https://arxiv.org/abs/2409.08264.

Ruisheng Cao, Fangyu Lei, Haoyuan Wu, Jixuan Chen, Yeqiao Fu, Hongcheng Gao, Xinzhuang Xiong, Hanchong Zhang, Yuchen Mao, Wenjing Hu, Tianbao Xie, Hongshen Xu, Danyang Zhang, Sida Wang, Ruoxi Sun, Pengcheng Yin, Caiming Xiong, Ansong Ni, Qian Liu, Victor Zhong, Lu Chen, Kai Yu, and Tao Yu. Spider2-v: how far are multimodal agents from automating data science and engineering workflows? In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2138–2156. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-acl.110/.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, Yuan Yao, Yankai Lin, Zhiyuan Liu, and Maosong Sun. Guicourse: From general vision language models to versatile gui agents, 2025. URL https://arxiv.org/abs/2406.11317.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI grounding for advanced visual GUI agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.505.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks, 2025. URL https://arxiv.org/abs/2502.08235.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, Rui Yan, and Shuo Shang. Mobile-bench: An evaluation benchmark for LLM-based mobile agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8813–8831, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.478. URL https://aclanthology.org/2024.acl-long.478/.

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. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5950bf290a1570ea401bf98882128160-Paper-Datasets_and_Benchmarks.pdf.

Yunfei Feng, Xi Zhao, Cheng Zhang, Dahu Feng, Daolin Cheng, Jianqi Yu, Yubin Xia, and Erhu Feng. Mobiflow: Real-world mobile agent benchmarking through trajectory fusion, 2026. URL https://arxiv.org/abs/2604.09587.

Hiroki Furuta, Kuang-Huei Lee, Ofir Nachum, Yutaka Matsuo, Aleksandra Faust, Shixiang Gu, and Izzeddin Gur. Multimodal web navigation with instruction-finetuned foundation models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 29528–29558, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/7ef7d8359036afd8c2378d82c21058a4-Paper-Conference.pdf.

Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the digital world as humans do: Universal visual grounding for GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, Yue Wen, Jingya Dou, Fei Tang, Jinzhen Lin, Yulin Liu, Zhenlin Guo, Yichen Gong, Heng Jia, Changlong Gao, Yuan Guo, Yong Deng, Zhenyu Guo, Liang Chen, and Weiqiang Wang. Ui-venus technical report: Building high-performance ui agents with rft, 2025. URL https://arxiv.org/abs/2508.10833.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen,

Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

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

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

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. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, pp. 8778–8786, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400720352. doi: 10.1145/3746027.3755688. URL https://doi.org/10.1145/3746027.3755688.

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. In A. Globerson, L. Mackey,

D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 92130–92154. Curran Associates, Inc., 2024. doi: 10.52202/079017-2925. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a79f3ef3b445fd4659f44648f7ea8ffd-Paper-Datasets_and_Benchmarks_Track.pdf.

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui 2: Mastering universal user interface understanding across platforms, 2025b. URL https://arxiv.org/abs/2410.18967.

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

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. Autoglm: Autonomous foundation agents for guis, 2024a. URL https://arxiv.org/abs/2411.00820.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Zibin Liu, Cheng Zhang, Xi Zhao, Yunfei Feng, Bingyu Bai, Dahu Feng, Erhu Feng, Yubin Xia, and Haibo Chen. Beyond training: Enabling self-evolution of agents with mobimem, 2025. URL https://arxiv.org/abs/2512.15784.

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. URL https://arxiv.org/abs/2406.08451.

Xing Han Lù, Zdeněk Kasner, and Siva Reddy. Weblinx: real-world website navigation with multi-turn dialogue. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, and Yueting Zhuang. Ui-s1: Advancing gui automation via semi-online reinforcement learning, 2025b. URL https://arxiv.org/abs/2509.11543.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 74325–74362. Curran Associates, Inc., 2024. doi: 10.52202/079017-2365. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/877b40688e330a0e2a3fc24084208dfa-Paper-Datasets_and_Benchmarks_Track.pdf.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Runliang Niu, Jindong Li, Shiqi Wang, Yali Fu, Xiyu Hu, Xueyuan Leng, He Kong, Yi Chang, and Qi Wang. Screenagent: a vision language model-driven computer control agent. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/711. URL https://doi.org/10.24963/ijcai.2024/711.

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. URL https://arxiv.org/abs/2501.12326.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/bbbb6308b402fe909c39dd29950c32e0-Paper-Datasets_and_Benchmarks.pdf.

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. URL https://arxiv.org/abs/2405.14573.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 2686–2710. Curran Associates, Inc., 2024a. doi: 10.52202/079017-0088. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0520537ba799d375b8ff5523295c337a-Paper-Conference.pdf.

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. *arXiv preprint arXiv:2401.16158*, 2024b.

Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. Mobileagentbench: An efficient and user-friendly benchmark for mobile llm agents, 2024c. URL https://arxiv.org/abs/2406.08184.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

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, 2025. URL https://arxiv.org/abs/2501.11733.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ACM MobiCom '24, pp. 543–557, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704895. doi: 10.1145/3636534.3649379. URL https://doi.org/10.1145/3636534.3649379.

Zhiyong Wu, Chengcheng Han, Zichen Ding, Zhenmin Weng, Zhoumianze Liu, Shunyu Yao, Tao Yu, and Lingpeng Kong. Os-copilot: Towards generalist computer agents with self-improvement, 2024. URL https://arxiv.org/abs/2402.07456.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. OS-ATLAS: Foundation action model for generalist GUI agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=n9PDaFNi8t.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094. Curran Associates, Inc., 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5d413e48f84dc61244b6be550f1cd8f5-Paper-Datasets_and_Benchmarks_Track.pdf.

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, 2024. URL https://arxiv.org/abs/2410.24024.

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, 2025. URL https://arxiv.org/abs/2412.04454.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v, 2023. URL https://arxiv.org/abs/2310.11441.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions, 2025. URL https://arxiv.org/abs/2412.16256.

Zonghan Yang, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. React meets actre: Autonomous annotation of agent trajectories for contrastive self-training. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=0VLBwQGWpA.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in*

Neural Information Processing Systems, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIV*, pp. 240–255, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73038-2. doi: 10.1007/978-3-031-73039-9_14. URL https://doi.org/10.1007/978-3-031-73039-9_14.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. UFO: A UI-focused agent for windows OS interaction. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 597–622, Albuquerque, New Mexico, April 2025a. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.26. URL https://aclanthology.org/2025.naacl-long.26/.

Cheng Zhang, Erhu Feng, Xi Zhao, Yisheng Zhao, Wangbo Gong, Jiahui Sun, Dong Du, Zhichao Hua, Yubin Xia, and Haibo Chen. Mobiagent: A systematic framework for customizable mobile agents, 2025b. URL https://arxiv.org/abs/2509.00531.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA, 2025c. Association for Computing Machinery. ISBN 9798400713941. doi: 10.1145/3706598.3713600. URL https://doi.org/10.1145/3706598.3713600.

Jiwen Zhang, Jihao Wu, Teng Yihua, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. Android in the zoo: Chain-of-action-thought for GUI agents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12016–12031, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.702. URL https://aclanthology.org/2024.findings-emnlp.702/.

Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3132–3149, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.186. URL https://aclanthology.org/2024.findings-acl.186/.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024a.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan,

and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/52f050499cf82fa8efb588e263f6f3a7-Paper-Conference.pdf.

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*, 2025.

A AGENT PROMPT TEMPLATES

We provide the prompt templates used to train the mobile GUI agents in our experiments.

A.1 DEFAULT PROMPT

This prompt template is used for the Observation+Action/Action-Only reasoning paradigms under the end-to-end workflow, which is the default setting.

```
<system>
You are a phone-use AI agent.

### Action Space
Your action space includes:
- Name: click, Parameters: target_element (a high-level
    description of the UI element to click), bbox (a bounding box
    of the target element, [x1, y1, x2, y2]).
- Name: swipe, Parameters: direction (one of UP, DOWN, LEFT, RIGHT
    ), start_coords (the starting coordinate [x, y]), end_coords (
    the ending coordinate [x, y]).
- Name: click_input, Parameters: target_element (a high-level
    description of the UI element to click), text (the text to
    input), bbox (a bounding box of the target element, [x1, y1,
    x2, y2]).
- Name: input, Parameters: text (the text to input).
- Name: open_app, Parameters: app_name (the name of the
    application to open).
- Name: press_home, Parameters: (no parameters, returns to the
    home screen).
- Name: press_back, Parameters: (no parameters, goes back to the
    previous screen).
- Name: wait, Parameters: (no parameters, will wait for 1 second).
- Name: done, Parameters: status (the completion status of the
    current task, one of `success`, `suspended` and `failed`).

### Response Format
Your output should be a JSON object with the following format:
{
  "reasoning": "Your reasoning here",
  "action": "The next action (one of click, click_input, input,
    swipe, open_app, press_home, press_back, wait, done)",
  "parameters": {"param1": "value1", "param2": "value2", ...}
}

<user>
### Current Task
"{task}"
### Action History
The sequence of actions you have already taken:
{history}
### Constraints
- If the screen has not changed after your last action, do not
    repeat the exact same action. Try a different method or
    slightly adjust coordinates.
- If the task is completed, verify the result before outputting '
    done'.

{screenshot}
```

```
Please provide the next action based on the screenshot and your
    action history. You should do careful reasoning before
    providing the action.
```

A.2 REFLECTION PROMPT

The reflection workflow uses the default prompt except for the following changes.

In <system>, the reasoning field in the response format is changed to:

```
"reasoning": "Your reflection and reasoning here"
```

In <user>, the single screenshot placeholder is replaced with:

```
### previous screen
{previous screenshot}

### current screen
{current screenshot}
```

In <user>, the final instruction is changed to:

```
Please provide the next action based on the previous screen,
    current screen, and your action history. You should do careful
     reflection about the previous action and reasoning about the
    next action.
```

A.3 PROMPT FOR THE DECOMPOSED WORKFLOW

The decomposed workflow uses two sequential stages. In stage 1, the planner prompt is the same as the default prompt, except that the coordinate-related parameters are removed from the action space:

```
- Name: click, Parameters: target_element (a high-level
    description of the UI element to click).
- Name: swipe, Parameters: direction (one of UP, DOWN, LEFT, RIGHT
    ).
- Name: click_input, Parameters: target_element (a high-level
    description of the UI element to click), text (the text to
    input).
```

In stage 2, the grounder prompt is:

```
<user>
{image}
Based on the description of the target UI element, locate the
    element in the screenshot.
Target element's description: {description}
Report the bbox coordinates in JSON format.
```

B DETAILED HYPERPARAMETERS

We provide the detailed hyperparameters used in SFT and RL training.

Table 2: SFT hyperparameters.

Hyperparameter	Value
Batch size	64
Cutoff length	16,384
Learning rate	5×10^{-6}
Training epochs	2
Learning-rate scheduler	Cosine Scheduler
Warmup ratio	0.05
Weight decay	0.01

Table 3: RL hyperparameters.

Hyperparameter	Value
RL algorithm	DAPO
Train batch size	128
Maximum prompt length	16,384
Maximum response length	1,024
Learning rate	1×10^{-6}
PPO mini-batch size	16
Group size	16
Sampling temperature	1.0
Top-p	1.0
KL regularization	Disabled
Clip ratio	Low 0.20, high 0.28
Training epochs	1