Step-GUI Technical Report

GELab-Team, StepFun

Homepage: https://opengelab.github.io/
Github: https://github.com/stepfun-ai/gelab-zero
Huggingface: Step-GUI Collections

Abstract

Recent advances in multimodal large language models unlock unprecedented opportunities for GUI automation. However, a fundamental challenge remains: how to efficiently acquire high-quality training data while maintaining annotation reliability? We introduce a self-evolving training pipeline powered by the **Calibrated Step Reward System**, which converts model-generated trajectories into reliable training signals through trajectory-level calibration, achieving >90% annotation accuracy with 10~100× lower cost. Leveraging this pipeline, we introduce Step-GUI, a family of models (4B/8B) that achieves state-of-the-art GUI performance (8B: 80.2% AndroidWorld, 48.5% OSWorld, 62.6% ScreenShot-Pro) while maintaining robust general capabilities. As GUI agent capabilities improve, practical deployment demands standardized interfaces across heterogeneous devices while protecting user privacy. To this end, we propose **GUI-MCP**, the first Model Context Protocol for GUI automation with hierarchical architecture that combines low-level atomic operations and high-level task delegation to local specialist models, enabling high-privacy execution where sensitive data stays on-device. Finally, to assess whether agents can handle authentic everyday usage, we introduce **AndroidDaily**, a benchmark grounded in real-world mobile usage patterns with 3146 static actions and 235 end-to-end tasks across high-frequency daily scenarios (8B: static 89.91%, end-to-end 52.50%). Our work advances the development of practical GUI agents and demonstrates strong potential for real-world deployment in everyday digital interactions.

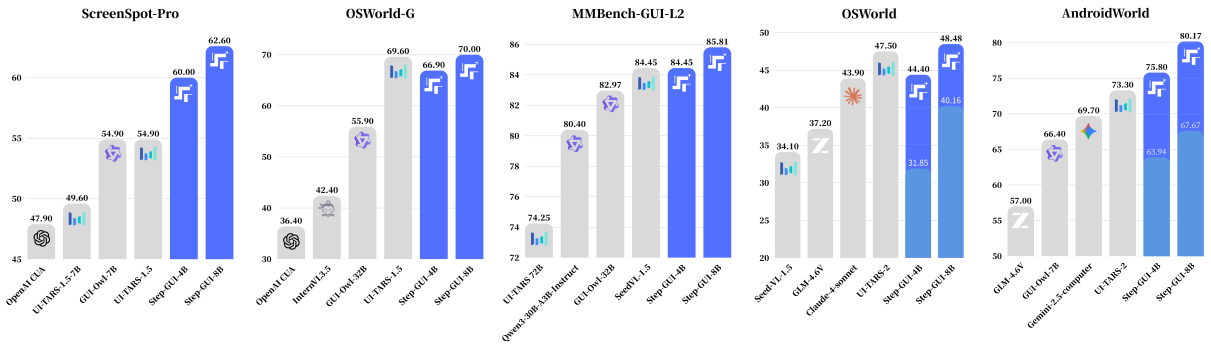

Figure 1 | Performance Overview across Heterogeneous GUI Benchmarks. We compare Step-GUI (4B/8B) against leading baselines on five diverse benchmarks covering grounding (ScreenSpot-Pro, OSWorld-G, MMBench-GUI-L2) and end-to-end agentic tasks (OSWorld, AndroidWorld). End-to-end results use pass@3 metric to mitigate non-model-related failures (e.g., CAPTCHA, VM crashes). Pass@1 results are also shown in a different shade of blue. The results demonstrate that Step-GUI-8B achieves state-of-the-art performance, outperforming existing open-source and proprietary agents, even surpassing models with much larger parameter counts.

1. Introduction

The advent of Large Language Models (LLMs) unlocks unprecedented capabilities in natural language understanding and generation, fundamentally transforming how machines process and generate human language. Recent breakthroughs in multimodal large language models Anthropic (2025c); Bai et al. (2025a); Comanici et al. (2025); Guo et al. (2025b); OpenAI (2025a) further extend these capabilities to visual perception and action planning, providing the foundational abilities necessary for GUI automation. These advances enable models to understand complex visual interfaces and reason about appropriate interactions, opening new possibilities for autonomous GUI agents that can assist users in their daily digital tasks.

Building on these foundational capabilities, the research community makes significant progress in GUI agent development. Various approaches explore methods for acquiring GUI interaction data Gao et al. (2024); Liu et al. (2025b); Rawles et al. (2023); Wang et al. (2025c), establishing critical data infrastructure for training GUI agents. Several powerful GUI agent models Qin et al. (2025); Wang et al. (2025a); Ye et al. (2025); Zeng et al. (2025) demonstrate impressive capabilities in navigating and manipulating graphical interfaces. However, a critical challenge remains: *how to efficiently obtain high-quality trajectory and knowledge data to improve agent performance in target domains?* Traditional annotation methods suffer from subjectivity and prohibitive costs, limiting the scalability of GUI agent development.

To address this challenge, we introduce a self-evolving training pipeline centered on the Calibrated Step Reward System (CSRS). While prior work explores self-improvement through data generation via rationalization, self-critique, quality filtering, and offline reinforcement learning Gulcehre et al. (2023); Jung et al. (2023); Madaan et al. (2023); Zelikman et al. (2022), these approaches primarily target single-turn tasks where model-generated annotations can be reliably verified. For multi-turn GUI agents where task execution spans multiple steps, direct model-generated annotations are prone to factual errors and hallucinations in the absence of verifiable ground truth. Unlike existing methods that rely solely on model introspection or post-hoc filtering, CSRS anchors LLM-generated dense step-level reasoning to trajectory-level evaluation signals, either automated verification scripts or human annotations, thereby ensuring annotation quality while maintaining scalability. Through trajectory-level calibration and LLM-powered knowledge extraction, CSRS converts model-generated trajectories into high-quality training data, achieving >90% annotation accuracy with 10-100× cost reduction compared to traditional step-level annotation. Our progressive three-stage training paradigm orchestrates parallel data flows for novel exploration and strategic knowledge filtering, continuously improving model capabilities across multiple training rounds.

On top of this pipeline, we train **Step-GUI**, a family of GUI specialist models (4B, 8B) built upon the Qwen3-VL backbone, and Step-GUI-8B achieves state-of-the-art performance: 80.2% on AndroidWorld, 48.5% on OSWorld, 65% on ScreenShot-Pro, and 89.91%/52.50% on our AndroidDaily benchmark (static/end-to-end). Remarkably, our compact 4B model delivers competitive performance while remaining deployable on consumer-grade hardware, enabling local execution without cloud dependencies.

As GUI agents gain enhanced capabilities in visual understanding and autonomous task execution, two fundamental challenges emerge: *standardized communication across heterogeneous devices* and *user privacy protection when processing sensitive data*. The fragmentation of development efforts and high integration costs historically suppress innovation in tool ecosystems Qu et al. (2025). To address the standardization challenge, the Model Context Protocol (MCP) Anthropic (2023) is proposed as a universal standard that defines how agents and tools should

communicate, enabling an interoperable "plug-and-play" ecosystem. Recent work explores how agents can effectively retrieve and select appropriate MCP services from large pools Gan and Sun (2025) and how to convert existing software projects into MCP format Ouyang et al. (2025). Building on these foundations, we propose **GUI-MCP** (Graphical User Interface - Model Context Protocol), the first MCP implementation specifically designed for GUI automation that addresses both standardization and privacy protection. GUI-MCP provides a standardized, cross-platform protocol that abstracts device capabilities into a small set of atomic and composite tools. Its hierarchical dual-layer architecture combines a *Low-level MCP* that offers fine-grained operations (e.g., clicks, swipes, text input) with a *High-level MCP* that delegates entire tasks to locally deployed GUI specialist models such as Step-GUI-4B. This design allows the main LLM to focus on high-level planning while it offloads routine GUI operations to local models. Critically, GUI-MCP supports a high-privacy execution mode in which raw screenshots and sensitive states stay on the device and only semantic summaries flow to external LLMs, which effectively protects user privacy while it leverages cloud-based reasoning capabilities.

As the community makes progress on these technical challenges, attention is shifting toward a more pressing concern: *how to evaluate whether GUI agents can handle real-world everyday usage beyond synthetic setups?* Existing work develops diverse evaluation benchmarks, ranging from static benchmarks Burns et al. (2022); Cheng et al. (2024); Li et al. (2025b,c); Liu et al. (2024a); Rastogi et al. (2021); Wu et al. (2024); Zhang et al. (2024) that assess element grounding and task planning through prediction-annotation matching, to interactive benchmarks Rawles et al. (2024); Xie et al. (2024) that enable task execution in realistic operating system environments. However, these efforts either concentrate on single-step prediction accuracy or lack coverage of high-frequency Chinese mobile applications, leaving a critical gap: *can agents reliably handle the high-frequency, everyday tasks that constitute real-world mobile usage?* To address this gap, we introduce **AndroidDaily**, a benchmark explicitly grounded in empirical analysis of authentic mobile usage patterns. Rather than pursuing maximal application coverage, AndroidDaily focuses on ubiquitous daily scenarios (Transportation, Shopping, Social Media, Entertainment, Local Services) where agent deployment has immediate practical impact. The benchmark employs a two-tier evaluation strategy: a *Static Benchmark* with 3146 actions for efficient single-step action prediction, and an *End-to-End Benchmark* with 235 tasks that span multiple dimensions (scenario, task type, complexity, and ambiguity) and evaluate autonomous task completion in fully functional environments. Step-GUI-8B demonstrates strong performance on AndroidDaily, which highlights both its current capabilities and remaining challenges in realistic deployment scenarios.

In summary, our contributions establish a comprehensive framework for building practical GUI agents:

- A self-evolving training pipeline with CSRS that achieves 10-100× cost reduction while it continuously improves model capabilities through closed-loop data refinement.
- Step-GUI models that achieve state-of-the-art performance across multiple benchmarks (8B: 80.2% on AndroidWorld, 48.5% on OSWorld, 62.6% on ScreenShot-Pro, 89.91%/ 52.50% on AndroidDaily static/end-to-end), with the 4B variant that enables consumer-grade local deployment.
- GUI-MCP, the first standardized protocol for LLM-device interaction that balances execution efficiency with privacy protection through hierarchical architecture.
- AndroidDaily, an ecologically valid benchmark grounded in real-world usage patterns with dual evaluation modes (3146 static actions, 235 end-to-end tasks) and multi-dimensional taxonomies for targeted capability analysis.

Together, these contributions address the complete pipeline from model training to standardized deployment interfaces and authentic evaluation, which paves the way for GUI agents that can assist users in their daily mobile interactions.

2. Step-GUI

2.1. Data

2.1.1. Mid-train Data

We aim to build a multimodal foundation model that retains **general-purpose capabilities**, inherits rich world knowledge, and acquires **strong agentic competence**. Instead of training a specialized planner or a domain-restricted policy model, we develop a multimodal foundation model capable of understanding diverse visual environments, following complex protocols, and executing multi-step tasks.

To bridge generic pretrained models and agent-specific training, we introduce a mid-train stage that equips the model with **foundational agent capabilities**. Specifically, the mid-trained model should: (i) retain and consolidate broad knowledge, including multimodal understanding, commonsense reasoning, and world knowledge; (ii) acquire fundamental visual competencies such as visual grounding, dense captioning, and fine-grained region understanding essential for GUI-based interactions; (iii) parse agent-style formats, including trajectory representations, action schemas, and observation structures; (iv) form initial instruction-to-action mappings that ground natural-language instructions into executable action sequences.

We construct a mid-train data mixture that balances general knowledge retention with agent-specific skill acquisition, comprising two complementary categories:

- **General Multimodal & Knowledge Data**, which preserves broad capabilities and builds foundational visual understanding:
  - High-quality text and multimodal data (1.9M): curated general-purpose samples to maintain language fluency and multimodal reasoning;
  - Knowledge-intensive data (2M): including dense captions and structured knowledge to reinforce world understanding;
  - Grounding data (2.7M): region-level annotations for visual grounding, enabling precise localization required by GUI agents.
- **Agent-Oriented Data**, which introduces action understanding and trajectory reasoning:
  - Action alignment data (170K): samples that align natural-language instructions with atomic actions, bootstrapping the instruction-to-action mapping;
  - Trajectory data (4M): multi-step interaction sequences annotated under diverse protocols by different annotators, exposing the model to varied action formats and planning patterns;
  - Environment-specific data (420K): incorporating cross-platform trajectory samples from Android, Ubuntu, Windows, and macOS systems, offering comprehensive agent experiences across heterogeneous mobile and desktop environments.

Through this balanced data mixture, the mid-trained model consolidates broad world knowledge, acquires essential visual competencies, learns to parse agent-style formats, and forms initial instruction-to-action mappings.

2.1.2. Cold-start Data

Building upon the mid-trained model's foundational agent capabilities, including visual grounding, agent-style format understanding, and initial instruction-to-action mappings, the cold-start stage focuses on knowledge injection and execution refinement. Rather than continuing broad exposure, this stage addresses knowledge gaps that cause execution failures and refines agent behavior through curated trajectory data. Specifically, the cold-start stage: (i) patches knowledge gaps by injecting missing knowledge identified from trajectory execution errors, enabling the model to overcome failure modes; (ii) refines agent behavior using curated trajectory samples to shape execution patterns; and (iii) preserves general competence by maintaining multimodal reasoning and visual grounding abilities. Agent failures often stem from knowledge deficiencies rather than insufficient behavioral examples. We therefore adopt an error-driven knowledge injection strategy: diagnosing execution failures and converting the missing knowledge into VQA pairs to directly target the model's weaknesses. The trajectory data serves as a behavioral scaffold, aligning outputs with agent formats while the enriched knowledge base enables robust generalization.

The cold-start data mixture (~1.67M samples) comprises:

- **Knowledge Data** (864K, 52%): constructed by analyzing execution errors from trajectory rollouts. When the model fails during trajectory execution, we identify the underlying missing knowledge (e.g., UI semantics, application behaviors, domain facts) and convert it into VQA-format samples. This targeted injection directly addresses the model's knowledge blind spots rather than providing generic world knowledge;
- **Trajectory Data** (404K, 24%): high-quality multi-step interaction sequences serving as behavioral demonstrations;
- **General Multimodal Data** (284K, 17%): high-quality samples to retain broad multimodal reasoning capabilities;
- **Grounding Data** (122K, 7%): curated localization samples essential for GUI-based interactions.

Table 1 presents the data composition across mid-train and cold-start stages. While mid-train employs a large-scale mixture (~11.2M samples) covering diverse data types including action alignment and environment-specific data, cold-start adopts a more focused mixture (~1.67M samples) with a higher proportion of knowledge data (52%) to address identified execution failures:

Table 1 | Data composition comparison between mid-train and cold-start stages.

Data Type	**Mid-Train**	**Cold-Start**
General Multimodal	1.9M	284K
Grounding	2.7M	122K
Knowledge	2.0M	864K
Action Alignment	170K	–
Trajectory	4.0M	404K
Environment-Specific	420K	–
Total	**~11.2M**	**~1.67M**

The cold-start stage uses a smaller data volume (~1.67M vs. ~11.2M), reflecting a shift from

broad exposure to focused refinement. This stage emphasizes knowledge-driven optimization: identifying and patching execution failures through error-driven VQA knowledge injection, while using curated trajectory data to refine interaction patterns and execution behaviors.

2.1.3. Grounding Data

While grounding is traditionally framed as a perception-language alignment problem, this formulation becomes insufficient in the GUI domain, where the model is expected to understand and act within structured virtual environments. A graphical interface is effectively a microcosm of the world: it comprises entities, spatial hierarchies, causal affordances, and state transitions. Supervision that merely aligns text spans with visual regions fails to capture these richer semantics and often leads to brittle, appearance-driven behavior. This observation motivates a shift in how GUI grounding is formulated and trained. To succeed in GUI grounding, a model must move beyond surface-level visual matching and acquire capabilities analogous to a lightweight world model for virtual environments. In our formulation, this entails addressing three fundamental requirements:

- **Functional semantics beyond appearance.** The model must learn that a gear icon affords settings and a trash icon affords deletion, rather than relying on surface-level visual similarity.
- **Latent world state.** The model should maintain a latent representation of what is visible, what is actionable, and how the interface state evolves under candidate actions.
- **World knowledge of HCI conventions.** Knowledge of human–computer conventions, layouts, and symbolic meanings enables reasoning about unseen or partially observed interfaces.

GUI grounding datasets face a fundamental challenge: annotations frequently contain errors (noise) and fail to accurately correspond to the semantic meaning of the interface elements they describe (misalignment). Directly scaling such data often amplifies noise rather than improving generalization. To address this, we design an iterative grounding-cleaning pipeline that progressively filters, corrects, and refines supervision using model feedback.

1. **Initial grounding training.** We first train an initial model on raw open-source grounding data, general multimodal data, and knowledge-augmented annotations to establish basic perceptual alignment.
2. **Pass-rate labeling with complexity scoring.** The trained model performs multiple independent rollouts per sample. Each sample receives a **pass-rate label** reflecting supervision quality. Additionally, an **LLM-based complexity scorer categorizes** tasks into simple localization, functional understanding, and intent-alignment levels, decoupling failures from annotation noise versus genuinely complex semantics.
3. **Curriculum-based reliable data training.** High pass-rate samples serve as reliable supervision and are organized by complexity for difficulty-aware curriculum training: simple localization tasks stabilize early-stage grounding, while functional and intent-alignment tasks are progressively introduced via curriculum SFT and reinforcement learning.
4. **Early exclusion of noisy cases.** Zero pass-rate samples which mix noisy annotations and genuinely hard cases, are excluded from early training to preserve learning signal quality.
5. **Hard-case refinement.** Excluded samples are revisited at later stages: failed executions are rewritten with step-by-step knowledge and enriched annotations, then reintroduced as high-quality supervision.

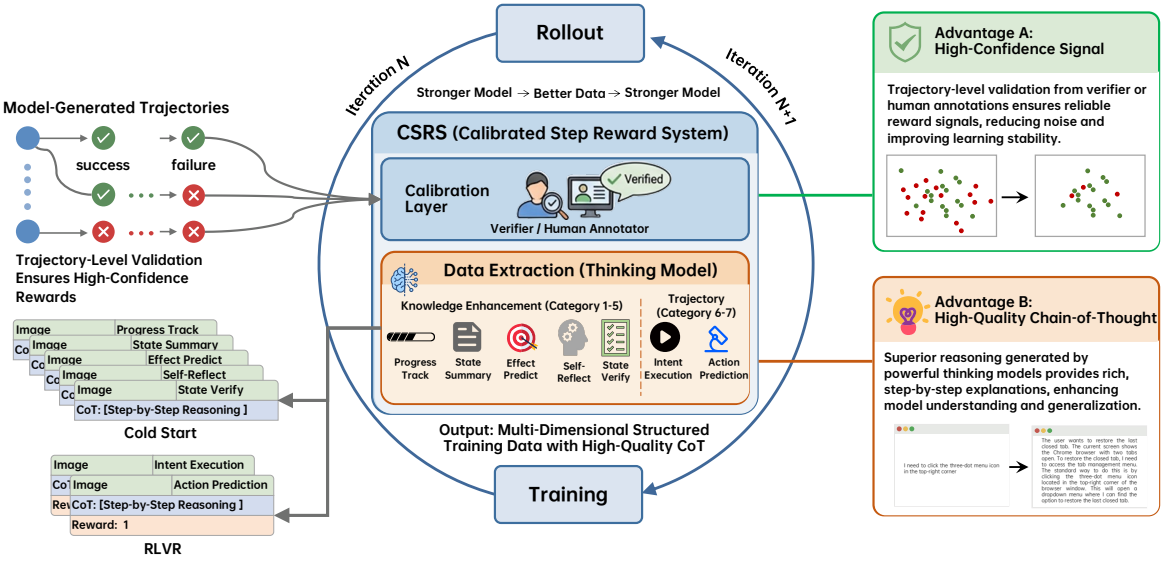

Figure 2 | **Calibrated Step Reward System (CSRS) Architecture.** The system consists of a Calibration Layer that performs trajectory-level validation (success/failure) and a Data Extraction module powered by thinking models that generates seven categories of structured training data. Model-generated trajectories flow through CSRS in an iterative loop: rollout generates trajectories, CSRS processes them into high-quality training data, and training produces stronger models for the next iteration. *Advantage A*: Trajectory-level validation provides high-confidence reward signals, ensuring learning stability. *Advantage B*: LLM-generated chain-of-thought provides rich reasoning that enhances model understanding. Success trajectories yield all seven data types while failed trajectories contribute only knowledge-related data (categories 1-6), implementing a selective learning strategy.

Through this closed-loop process of pass-rate labeling, data filtering, curriculum learning, and hard-case refinement, the model gradually transitions from sparse click-based supervision to explicit modeling of interface structure, dynamics, and affordances. This grounding loop substantially improves robustness on complex grounding and agentic tasks, while relying exclusively on open-source corpora and systematic scaling of model capacity, data volume, and compute.

2.1.4. Trajectory Data

To enable continuous model improvement through data flywheel iteration, we propose the Calibrated Step Reward System (CSRS), a novel data processing framework that converts model-generated trajectories into high-quality, multi-dimensional training data. CSRS serves as the critical bridge in the "Rollout → CSRS → Training" loop, ensuring data reliability while maximizing information extraction from each trajectory. CSRS introduces two synergistic mechanisms that address fundamental challenges in reward system design (see Figure 2): (1) *trajectory-level calibration* providing high-confidence reward signals, and (2) *LLM-powered data extraction* generating superior chain-of-thought reasoning. Unlike traditional step-level annotation approaches that suffer from subjectivity and high costs, our trajectory-level validation achieves >90% accuracy with 10-100× lower annotation costs by focusing on objectively verifiable task outcomes.

System Architecture. As illustrated in Figure 2, CSRS consists of two main components.

The *Calibration Layer* employs verifiers or human annotators to perform binary success/failure validation at the trajectory level, establishing a reliable quality anchor. The *Data Extraction Module*, powered by sophisticated thinking models, then generates seven categories of training data: (1) progress tracking, (2) state summary, (3) effect prediction, (4) self-reflection, (5) state verification, (6) intent execution, and (7) action prediction. This design ensures that all generated fine-grained data is anchored by high-confidence trajectory-level labels.

Selective Learning Strategy. CSRS intelligently handles trajectories of different qualities. For successful trajectories, all seven data types are extracted, encompassing both knowledge enhancement (categories 1-5) and action prediction (categories 6-7). For failed trajectories, only knowledge-related data (categories 1-6) are retained, following the principle of "learning knowledge from failures, but not learning erroneous actions." This selective strategy maximizes data utility while preventing the propagation of incorrect behaviors.

LLM-Generated Knowledge Superiority. A key advantage of CSRS lies in leveraging powerful LLMs to automatically generate training data. Compared to human annotators, LLMs produce: (i) significantly richer chain-of-thought reasoning with detailed multi-step analysis, (ii) consistent quality across all samples without individual variation, (iii) comprehensive domain knowledge about GUI operations and application functionalities, and (iv) 80-90% cost reduction through automation. For instance, while human annotators might simply label "click center button", CSRS generates detailed reasoning: "The text is already selected. The next step is to apply center alignment formatting. I can see the alignment buttons in the toolbar, and I will click the 'Align Center' button. After clicking it, the heading should move to the center of the document."

CSRS enables continuous model improvement through iterative training. In iteration N, model M_n generates rollout trajectories, which are processed by CSRS to produce high-quality training data, resulting in an improved model M_{n+1}. As the model becomes stronger, rollout quality increases, leading to more successful trajectories and richer training data. This self-reinforcing cycle drives progressive performance enhancement: from initial success rates of 30-40% to expert-level performance exceeding 85% after multiple iterations. The combination of trajectory-level calibration and LLM-powered generation achieves an optimal balance of quality, cost, and scalability. (*Advantage A*) Trajectory-level validation from verifiers or human annotations ensures reliable reward signals, reducing noise in the learning process and improving training stability. (*Advantage B*) Superior reasoning generated by powerful thinking models provides rich, step-by-step explanations, enhancing model understanding and generalization. This "coarse-grained high-confidence labels + fine-grained high-quality content" paradigm represents a significant advancement over traditional step-level annotation methods, establishing CSRS as a key infrastructure for building high-performance GUI agents.

2.2. Training

We employ the Qwen3-VL Bai et al. (2025a) series (spanning 4B, 8B parameters) as our visual-linguistic backbone. To bridge the gap between general multimodal capabilities and expert GUI agency, we propose a progressive three-stage training paradigm: Mid-Training, Cold-Start Fine-Tuning, and Reinforcement Learning with Verifiable Rewards (RLVR).

2.2.1. Formulation

We formulate GUI automation as a sequential decision-making problem over the tuple $(\mathcal{Q}, \mathcal{A}, \mathcal{S}, \mathcal{O})$:

- **Task Space** $\mathcal{Q}$: The set of natural language instructions that describe user intents (e.g., "Check the battery level and send it to social media."). Each task instance is denoted as $q \in \mathcal{Q}$.
- **Action Space** $\mathcal{A}$: GUI-based actions including click, scroll, swipe, type, and other interface interactions. These actions are executed purely through visual understanding without relying on structured accessibility information.
- **State Space** $\mathcal{S}$: The visual state of the GUI environment, represented by screen captures at each time step.
- **Observation Space** $\mathcal{O}$: Screenshots of the current interface along with the task instruction and historical interaction logs.

We model the agent as a parameterized policy $\pi_\theta : \mathcal{O} \times \mathcal{Q} \to \mathcal{A}$, where θ denotes the model parameters. Given a task $q \in \mathcal{Q}$ and visual observation $s \in \mathcal{S}$, the policy generates a sequence of outputs $o = (o_1, o_2, \ldots, o_T)$ representing the action trajectory. The objective is to optimize θ such that the policy successfully executes tasks through vision-based GUI interaction.

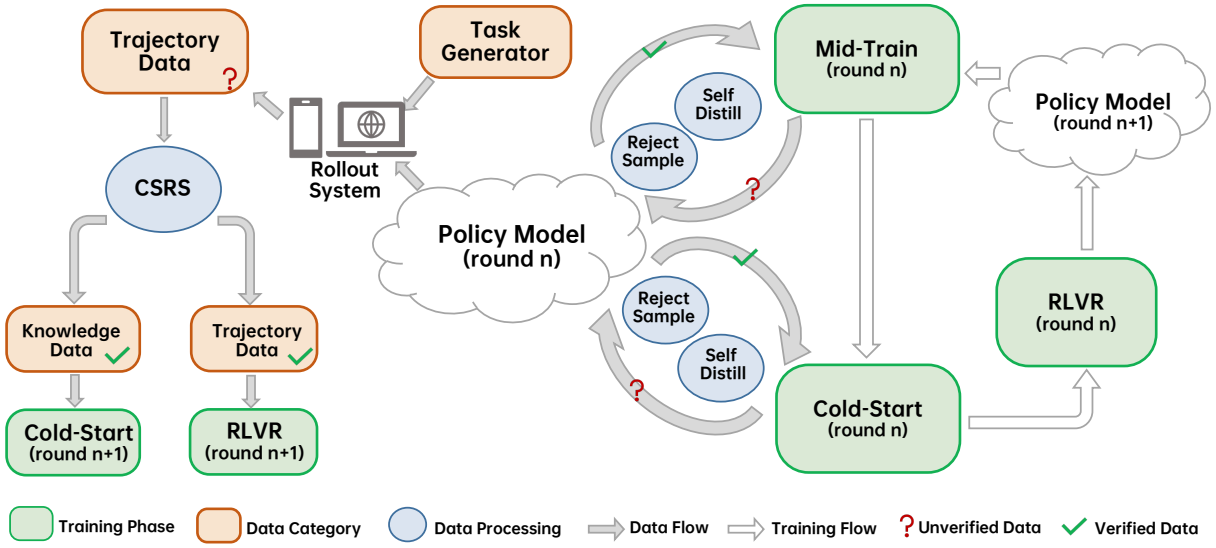

Figure 3 | Self-Evolving Training Pipeline with Closed-Loop Data Refinement. The pipeline consists of three progressive training stages (Mid-Train, Cold-Start, and RLVR) and two parallel data flows. **Generation Data Flow:** The Policy Model generates new trajectories via Task Generator, which are verified through the CSRS to produce high-quality Knowledge Data and Trajectory Data for the next training round. **Refinement Data Flow:** Existing trajectory data undergo dual-path filtering through Self-Distillation and Rejection Sampling. This iterative loop continuously enhances data quality and model capability across rounds.

2.2.2. *Data Flow*

Central to our approach is a self-evolving closed-loop data engine illustrated in Figure 3, which orchestrates two parallel data flows to continuously enhance both model capability and data quality across training rounds:

Generation Data Flow. The current Policy Model (round n) interactively executes newly generated tasks from the Task Generator within the Rollout System. During execution, the model produces raw trajectory data capturing its step-by-step interactions with the GUI environment. These raw trajectories are then processed by CSRS, which verifies action correctness and assigns calibrated rewards to filter and refine the data into two high-quality categories: (1) *Knowledge*

Data containing distilled task-solving insights and reasoning patterns, and (2) *Trajectory Data* capturing verified complete multi-step execution paths. These high-quality synthetic data are channeled into Cold-Start and RLVR stages of the next round ($n+1$), enabling the model to learn from its own explorations.

Refinement Data Flow. In parallel, existing trajectory data undergo a dual-path filtering mechanism combining Self-Distillation and Rejection Sampling. This process partitions the data into two categories: (1) *Accepted Set*—stable, high-confidence samples that consistently pass quality thresholds, recycled to Mid-Train and Cold-Start to reinforce foundational capabilities; and (2) *Rejected Set*—challenging samples near the decision boundary that expose model weaknesses, exclusively routed to Cold-Start for targeted capability enhancement.

This dual-flow architecture ensures that each training round benefits from both *novel high-quality exploration* (Generation Data Flow) and *strategic refinement of existing knowledge* (Refinement Data Flow). The iterative cycle progressively amplifies the model's grounding accuracy, general multimodal understanding, and complex reasoning capabilities in GUI scenarios.

2.2.3. Reinforcement Learning with Verifiable Rewards

To specialize the agent in precise reasoning and execution within the GUI domain, we employ Group Relative Policy Optimization (GRPO). The objective function is defined as:

$$J_{GRPO}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(O|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{k=1}^{|o_i|}\min\left(r_{i,k}A_i, \text{clip}(r_{i,k}, 1-\varepsilon, 1+\varepsilon)A_i\right) - \beta\mathbb{D}_{KL}\right] \quad (1)$$

where G is the group size, A_i is the advantage computed by normalizing rewards within the group, q is the task sampled from $P(Q)$, $\pi_{\theta_{old}}$ is the reference policy, o_i is the trajectory generated by $\pi_{\theta_{old}}$, $r_{i,k} = \frac{\pi_\theta(o_{i,k}|q,o_{i,<k})}{\pi_{\theta_{old}}(o_{i,k}|q,o_{i,<k})}$ is the importance sampling ratio, ε is the clipping parameter, and β controls the KL divergence regularization.

Fine-Grained Hybrid Reward Specification. We construct a composite reward function $R(o,s)$ by integrating three distinct signal categories: verifiable spatial metrics, action-semantic validity, and model-based capability assessment.

1) Spatial-Geometric Dense Rewards. To ensure pixel-level precision, we implement a dense reward mechanism based on tolerance-normalized high-order decay. Let τ_x, τ_y denote the predefined pixel-level tolerance thresholds.

Point-based Reward: For coordinate interactions (e.g., grounding, CLICK, LONGPRESS), we define the normalized deviation $\hat{\delta}_k = |k_{pred} - k_{gt}|/\tau_k$. The reward r_{point} employs a quartic decay function to impose steep penalties for deviations beyond the tolerance zone:

$$r_{point} = \exp\left(-\left(\hat{\delta}_x^4 + \hat{\delta}_y^4\right)\right) \quad (2)$$

BBox-based Reward: For region grounding, we decouple the bounding box into center coordinates (c_x, c_y) and dimensions (w,h). We construct a composite geometric energy term E_{geom} that down-weights dimensional errors by a factor of λ:

$$E_{geom} = \hat{\delta}_{cx}^4 + \hat{\delta}_{cy}^4 + \lambda\hat{\delta}_w^4 + \lambda\hat{\delta}_h^4 \quad (3)$$

where $\lambda = 0.5$. To simultaneously optimize for alignment accuracy and area overlap, the final bounding box reward R_{bbox} is a weighted fusion of the geometric score and Intersection over

Union (IoU):

$$R_{bbox} = \alpha \cdot \exp(-E_{geom}) + (1 - \alpha) \cdot \text{IoU}(b_{pred}, b_{gt}) \tag{4}$$

where we set $\alpha = 0.8$ to prioritize geometric centrality while retaining IoU as a shape-consistency regularization.

2) Action-Semantic Mixed Rewards. For the GUI action space, we decouple the reward signal into action type and action value.

- **Sparse Action Type:** The validity of the chosen operation (e.g., WAIT vs. COMPLETE) is modeled as a binary sparse reward $\mathbb{I}(\hat{a}_{type} = a^*_{type})$.
- **Adaptive Value Modeling:** The reward for action parameters varies by type. For trajectory-based vectors (e.g., SLIDE), we compute the cosine similarity between the predicted vector $\mathbf{v}_{pred}$ and the ground truth $\mathbf{v}_{gt}$, mapping the alignment to a dense $[0, 1]$ interval. For semantic actions requiring information retrieval (e.g., INFO, TYPE), we utilize an external LLM to verify the content, returning a scalar score $s \in [0, 1]$.

3) Soft Capability Rewards (LLM-as-a-Judge). For abstract qualities where deterministic rules are inapplicable, we employ an LLM-as-a-Judge mechanism. This module evaluates the generated trajectory based on intent consistency, fluency, and reasoning quality, providing a complementary soft signal that aligns the policy with human-preferred interaction patterns.

Semi-Online Exploration with Hindsight. Exploration in long-horizon GUI tasks is notoriously difficult due to sparse rewards. To mitigate this, we introduce a Semi-Online training strategy. For rollout groups that fail to complete the task, we inject Ground-Truth Hints into the prompt to guide the model through the correct reasoning path during a second pass. This allows the model to experience high-reward trajectories that were previously out of its capability range, effectively converting "failed explorations" into "guided successful samples" with high advantage scores.

Stability & Efficiency Enhancements. To ensure stable convergence and maximize data utility, we integrate several algorithmic enhancements: **1) Dynamic Exploration (ε_{high}):** We introduce a dynamic parameter ε_{high} to modulate the clipping range, providing greater flexibility for low-probability actions. This expands the exploration space without destabilizing the policy update. **2) Sample Reuse via Importance Sampling:** As illustrated in the efficiency module, data generation is computationally expensive. We employ Importance Sampling to reuse collected trajectories for multiple gradient updates. The policy is updated for K iterations per rollout batch. The importance sampling ratio $r_t(\theta) = \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{old}(o_t|q,o_{<t})}$ accounts for the distributional shift between the current evolving policy and the data-collecting policy, significantly improving sample efficiency while maintaining trust-region constraints.

Gradient Preservation for Clipping. The gradient of Equation (1) without $\mathbb{D}_{KL}$ can be formulated as:

$$\nabla_\theta J_{GRPO}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(O|q)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{k=1}^{|o_i|} f(r_{i,k}, A_{i,k}) \nabla_\theta \log \pi_\theta(o_{i,k}|q, o_{i,<k}) \right],$$
$$\text{where } f(r_{i,k}, A_{i,k}) = \begin{cases} A_i \cdot r_{i,k} & \text{if } 1-\varepsilon < r_{i,k} < 1+\varepsilon, \\ \beta_1 \cdot A_i \cdot (1+\varepsilon) & \text{if } r_{i,k} > 1+\varepsilon, \\ \beta_2 \cdot A_i \cdot (1-\varepsilon) & \text{if } r_{i,k} < 1-\varepsilon, \end{cases} \tag{5}$$

where the gradient modulation depends explicitly on the importance sampling ratio $r_{i,k}$. A critical analysis of Equation (5) reveals an inherent instability for tokens with low baseline

probabilities. For tokens where the reference probability $\pi_{\theta_{\text{old}}}$ is minimal, even marginal positive perturbations in the current policy π_θ result in a disproportionate amplification of the ratio $r_{i,k}$. Consequently, these low-probability tokens are highly susceptible to ratio explosion, prematurely triggering the upper clipping threshold $(1+\varepsilon)$. In standard clipping mechanisms, this saturation effectively zeros out the gradient, preventing the model from reinforcing potentially valuable but currently rare actions, a phenomenon that severely hinders the exploitation of sparse rewards. To mitigate this issue and maintain effective optimization, we implement a Gradient Preservation strategy Su et al. (2025). As defined in the piecewise function $f(r_{i,k}, A_{i,k})$, we retain a scaled gradient even when the ratio exceeds the trust region, thereby ensuring continuous learning signal flow for low-probability tokens.

3. GUI-MCP: An Efficient Protocol for Secure LLM-Device Interaction

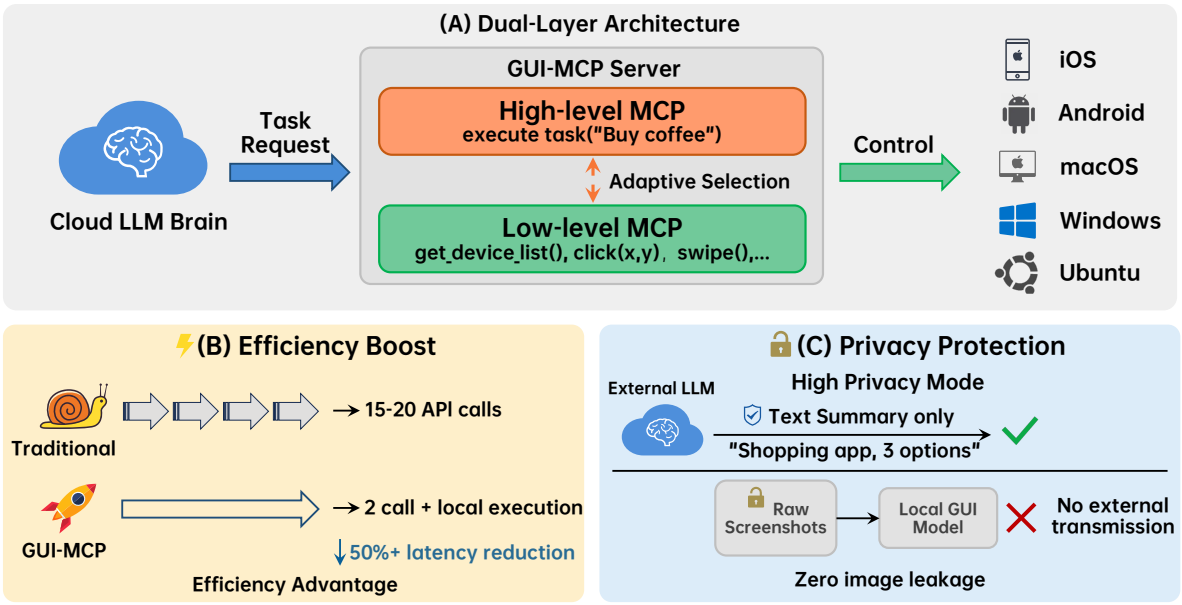

Figure 4 | Overview of GUI-MCP architecture. The dual-layer design includes Low-level MCP (providing atomic device operations) and High-level MCP (delegating tasks to a local GUI specialist model). This hierarchical approach enables efficient task execution while preserving user privacy through local processing.

Despite the remarkable progress in LLMs, their application to GUI automation remains challenging due to the lack of standardized interfaces for cross-platform device control. Existing solutions are often platform-specific and require substantial engineering effort to integrate with different LLMs and devices. To address this limitation, we propose **GUI-MCP** (Graphical User Interface - Model Context Protocol), the first MCP implementation specifically designed for GUI operation tasks. As shown in Figure 4, GUI-MCP provides a standardized toolkit that seamlessly connects various LLMs with multiple device platforms (Ubuntu, macOS, Windows, Android, iOS), enabling LLMs to control mobile and desktop devices through a unified protocol for executing GUI operation tasks.

3.1. Architecture Design

GUI-MCP adopts a hierarchical design that stratifies functionality into two distinct levels: **Low-level MCP** and **High-level MCP**, as illustrated in Figure 4.

3.1.1. Low-level MCP

The low-level MCP focuses on atomic device operations, providing fine-grained control interfaces. This layer exposes the following categories of primitives:

Device Management. The interface `get_device_list()` retrieves all connected devices, enabling multi-device orchestration.

State Perception. The interface `get_screenshot()` captures the current device screen state, providing visual feedback for decision-making.

Basic Operations. A comprehensive set of interaction primitives as shown in Table 2.

Category	Operations
Click	`click`, `double_click`, `triple_click` `right_click`, `middle_click`
Gesture	`swipe`, `long_press`
Mouse	`move_to`, `drag_to`
Input	`input_text`, `hotkey`
Application	`awake`

Table 2 | Low-level MCP basic operations.

These atomic interfaces provide maximum flexibility for the main LLM to perform fine-grained planning and control based on the current state and task requirements.

3.1.2. High-level MCP

The high-level MCP focuses on abstract task execution by encapsulating complete task execution logic. The primary interface is:

`execute_task(task_description)`: This interface accepts natural language task descriptions and automatically completes the task. For example:

- `execute_task("Click the first element")`
- `execute_task("Buy a cup of coffee")`
- `execute_task("Search for white canvas shoes, size 37, under $100, and favorite the first result")`

Internally, the high-level MCP integrates a locally deployed GUI specialist model (e.g., Step-GUI-4B) that has been specifically optimized for GUI operation tasks. This specialist model can autonomously complete tasks within its capability scope. The system prompt of the main LLM explicitly describes the capability boundaries of the GUI specialist model, helping the main LLM determine when to delegate tasks to the high-level MCP.

3.2. Design Advantages

3.2.1. Improved Execution Efficiency

The dual-layer architecture enables the main LLM to flexibly select control strategies based on task complexity and current state:

Low-level MCP Usage Scenarios:

- Rapid acquisition of current device state

- Tasks requiring fine-grained step-by-step planning
- Tasks exceeding the GUI specialist model's capabilities
- Scenarios requiring multi-turn user interaction for task clarification

High-level MCP Usage Scenarios:

- Clear task descriptions within the GUI specialist model's scope
- Desire to reduce inference overhead and API calls to the main LLM
- Tasks with strong independence that can be completed in one shot

By appropriately allocating tasks, the main LLM can delegate simple, repetitive GUI operations to the local specialist model while focusing on high-level task planning and decision-making, thereby significantly improving overall execution efficiency.

3.2.2. Enhanced Privacy Protection

In an era where privacy protection is increasingly critical, many users have concerns about transmitting screenshots and device information to external cloud LLM service providers. GUI-MCP provides a **High Privacy Mode** with the following characteristics:

Data Anonymization Mechanism: External cloud LLM brains cannot directly access raw screenshots and detailed device information. Instead, they only receive state summaries processed by the local GUI model. These summaries contain only the key semantic information necessary for task completion, excluding sensitive visual details.

Local Execution: Operations that require screenshot analysis for planning are performed by the local GUI model. All image data and sensitive information are processed exclusively on the local device, with the external cloud LLM only responsible for high-level task decomposition and decision-making.

Flexible Privacy Levels: Users can configure privacy protection levels based on their trust preferences and task requirements. The system supports multi-level configurations ranging from fully open (direct screenshot transmission) to fully private (text summaries only).

This design allows GUI-MCP to fully leverage the reasoning capabilities of powerful cloud-based LLMs while effectively protecting user privacy data, achieving an optimal balance between functionality and privacy.

Through its innovative dual-layer architecture, GUI-MCP provides a systematic solution for LLM-driven GUI automation across both efficiency and privacy dimensions. This protocol not only lowers the technical barrier for extending LLM capabilities to the GUI operation domain but also provides a viable path for building AI assistants that are both powerful and privacy-preserving.

4. AndroidDaily: A Dynamic Benchmark for Agentic Tasks in Daily Life

While recent benchmarks like AndroidWorld make significant strides in evaluating GUI agents on Android platforms, a critical gap remains: the disconnect between evaluated tasks and actual daily usage patterns. Existing benchmarks predominantly focus on productivity or utility apps that, while technically diverse, do not necessarily represent the tasks users perform most frequently in their everyday lives. AndroidDaily addresses this gap by grounding task selection in empirical analysis of actual mobile usage patterns rather than application catalogs. Based on

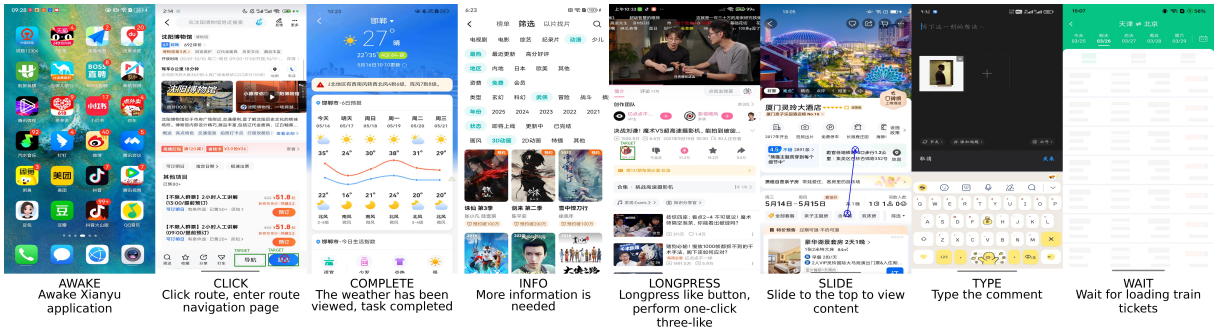

Figure 5 | AndroidDaily Static Benchmark Action Taxonomy. Eight action types for Android task automation are illustrated: AWAKE, CLICK, COMPLETE, INFO, LONGPRESS, SLIDE, TYPE, and WAIT (left-to-right, top-to-bottom). Each example shows a task description and annotated ground truth actions with parameters. Multi-solution cases are supported, e.g., the CLICK example (second panel) shows two valid target regions highlighted in red boxes.

usage frequency data and download statistics, we curate apps spanning key daily-life scenarios including transportation, shopping, social media, entertainment, and local services, prioritizing high-frequency tasks such as food delivery ordering, ride-hailing, short-form video consumption, and mobile payments. The selected tasks involve real-world consequences (e.g., financial transactions, service bookings), multi-step decision-making (e.g., comparing options with multiple criteria), and tight integration with physical services. These characteristics fundamentally differentiate AndroidDaily from utility-focused benchmarks, enhancing the representativeness of scenarios where agent deployment has immediate practical impact. AndroidDaily employs a dual evaluation strategy that balances evaluation efficiency with real-world fidelity, including a static benchmark and an end-to-end benchmark.

Static Benchmark (3146 actions). The static benchmark provides task descriptions alongside step-by-step screenshots, evaluating agent performance by comparing predicted actions against ground-truth single-step actions. As illustrated in Figure 5, the benchmark encompasses eight action types essential for Android task automation: *AWAKE*, *CLICK*, *COMPLETE*, *INFO*, *LONGPRESS*, *SLIDE*, *TYPE*, and *WAIT*. Each action is annotated with precise parameters that define execution targets and values. Importantly, the benchmark explicitly supports multi-solution ground truth annotations, for instance, CLICK actions may have multiple valid target regions when several UI elements can accomplish the same goal, reflecting the reality that mobile interfaces often offer alternative interaction pathways. We compute accuracy for both action types and action values. This approach offers efficient testing without requiring complex engineering infrastructure, enabling rapid iteration during model development.

End-to-End Benchmark (235 tasks). The end-to-end benchmark evaluates agents on complete task workflows, employing an LLM-based judger to determine task completion and measuring overall success rates. Though more demanding in terms of infrastructure, this setup provides ecologically valid evaluation that closely mirrors real-world task execution scenarios.

As shown in Figure 6, the 235 end-to-end tasks are primarily organized by scenario distribution, spanning five core categories: *Transportation* (78 tasks, 33.19%), *Shopping* (61 tasks, 25.96%), *Social Media* (43 tasks, 18.30%), *Entertainment* (37 tasks, 15.74%), and *Local Services* (16 tasks, 6.81%). Beyond scenario categorization, tasks are further characterized by cognitive operation types (information filtering, querying, and analysis), structural complexity (atomic, composite, and conditional/loop tasks), and instruction ambiguity levels (low, medium, and high ambigu-

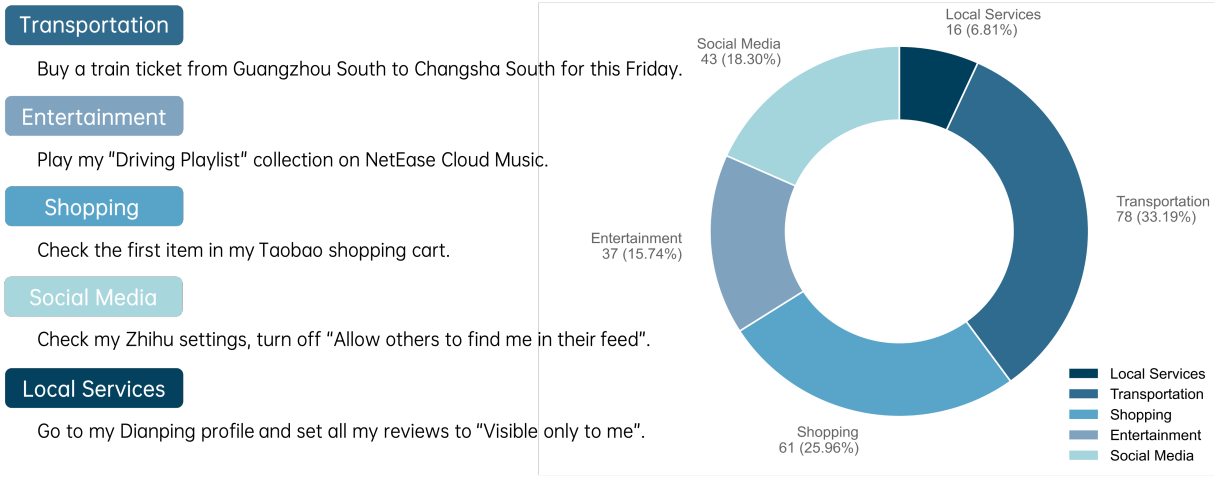

Figure 6 | AndroidDaily End-to-End Benchmark Scenario Distribution. The 235 tasks are organized across five daily-life scenarios: Transportation, Shopping, Social Media, Entertainment, and Local Services. The pie chart (right) shows the quantitative distribution across these categories, with Transportation and Shopping comprising the majority of tasks. Representative task examples for each scenario are provided (left), demonstrating the diversity and practical relevance of the benchmark.

ity). This multi-dimensional organization enables targeted analysis of agent capabilities and systematic identification of failure modes, offering actionable insights for model improvement.

5. Experiments

5.1. Experimental Setup

Grounding Benchmarks. We evaluate the grounding capabilities of our model on five challenging benchmarks: ScreenSpot-Pro Li et al. (2025c), ScreenSpot-v2 Wu et al. (2024), OSWorld-G Wu et al. (2024),MMBench-GUI-L2 Wang et al. (2025d) and VisualWebBench Liu et al. (2024a). These benchmarks assess the model's ability to accurately perceive and locate UI elements across common GUI pages with varying levels of difficulty, providing a comprehensive evaluation of fundamental visual grounding skills in graphical user interfaces.

Computer Use. OSWorld Xie et al. (2024) provides a comprehensive evaluation suite consisting of 369 real-world tasks with detailed environment configurations and automated evaluation scripts. These tasks cover diverse desktop applications and require multi-step reasoning and interaction capabilities.

Phone Use. AndroidWorld Rawles et al. (2024) offers 116 tasks across 20 mobile applications within a live Android emulator environment. The benchmark includes dynamic task variations generated via randomized parameters, enabling robust evaluation of agent performance.

AndroidDaily. As described in Section 4, AndroidDaily evaluates agent performance on commonly used Chinese applications with tasks categorized into three difficulty levels (L1-L3), providing insights into model capabilities on region-specific mobile interfaces.

General Multimodal Benchmarks. We further evaluated the generalizability of our model on mainstream benchmarks across other multimodal domains which contains V* Wu and Xie (2024), PhyX Shen et al. (2025), OmniOCR OmniAI (2025), OCRBench Liu et al. (2024c),

Table 3 | Model Performance on Multiple GUI-grounding Benchmarks. Best results are in **bold**, second best are underlined.

Model	ScreenSpot-Pro	ScreenSpot-v2	OSWorld-G	MMBench-GUI-L2	VisualWebBench
OpenAI CUA	23.4	87.9	36.4	-	-
UI-TARS-1.5	61.6	94.2	-	-	-
SeedVL-1.5	60.9	**95.2**	-	84.4	87.3
InternVL3.5-30B-A3B-Instruct	-	87.3	42.4	-	-
UI-TARS-1.5-7B	35.7	91.6	47.5	64.3	-
UI-TARS-72B-DPO	38.1	90.3	-	74.2	82.8
Qwen2.5-VL-72B	43.6	87.1	-	41.8	-
GUI-Owl-7B	54.9	92.8	55.9	80.5	-
GUI-Owl-32B	58.0	93.2	58.0	83.0	-
Qwen3-VL-4B-Instruct	59.5	94.0	58.2	-	-
Qwen3-VL-8B-Instruct	54.6	94.4	58.2	-	-
Qwen3-30B-A3B-Instruct	60.5	94.7	61.0	79.7	79.7
Step-GUI-4B	60.0	93.6	66.9	84.0	**90.7**
Step-GUI-8B	**62.6**	95.1	**70.0**	**85.6**	89.7

LogicVista Xiao et al. (2024), MMBench_en Liu et al. (2024b), MMBench_cn, MMStar Chen et al. (2024),MathVista Lu et al. (2023),CharXiv Wang et al. (2024) and Simplevqa Cheng et al. (2025).

Baselines. We compare our approach against state-of-the-art foundation models and specialized GUI agents: Claude-4-sonnet Anthropic (2025b), Claude-4.5-sonnet Anthropic (2025a), OpenAI CUA-o3 (Computer Use Agent) OpenAI (2025b), Gemini-2.5-Computer Use Google (2025), UI-TARS-1 Qin et al. (2025), UI-TARS-2 Wang et al. (2025a), InternVL3.5 Wang et al. (2025b), MobileRL-9B Xu et al. (2025), AutoGLm-9B Z.ai (2025a), GLM4.6V Z.ai (2025b), Qwen2.5VL Bai et al. (2025b), Qwen-OWL&Mobile-Agent-v3 Ye et al. (2025), Qwen3VL Bai et al. (2025a). These baselines represent diverse paradigms including general-purpose vision-language models, specialized computer-use models, and dedicated GUI automation agents.

5.2. Main Results

Performance on Grounding Benchmarks. Table 3 presents a comprehensive comparison of our method against state-of-the-art baselines across five GUI grounding benchmarks. Our models, Step-GUI-4B and Step-GUI-8B, demonstrate consistently strong performance across diverse evaluation settings.

ScreenSpot-Pro. On ScreenSpot-Pro, which evaluates professional-level GUI grounding, Step-GUI-8B achieves the highest score of 62.6, outperforming UI-TARS-1.5 (61.6), SeedVL-1.5 (60.9), and Qwen3-30B-A3B (60.5). Step-GUI-4B also attains a competitive 60.0, surpassing models with significantly larger parameter counts such as Qwen2.5-VL-72B (43.6).

ScreenSpot-v2. On ScreenSpot-v2, Step-GUI-8B reaches 95.1, approaching the top-performing SeedVL-1.5 (95.2) and exceeding UI-TARS-1.5 (94.2).

OSWorld-G. For OSWorld-G, which measures grounding in realistic desktop environments, our models achieve substantial improvements: Step-GUI-8B attains 70.0 and Step-GUI-4B reaches 66.9, outperforming the next best Qwen3-30B-A3B (61.0) by a significant margin of +9.0 and +5.9 points respectively.

MMBench-GUI-L2. On MMBench-GUI-L2, Step-GUI-8B achieves the best result of 85.6, followed by Step-GUI-4B at 84.0, both surpassing SeedVL-1.5 (84.4) and GUI-Owl-32B (83.0).

VisualWebBench. For VisualWebBench, Step-GUI-4B achieves the highest score of 90.7,

Table 4 | Performance on End-to-End Environment Benchmarks. Best results are in **bold**, second best are underlined. Note: We report both pass@1 and pass@3 metrics. Pass@3 (maximum 3 attempts per task) is used to mitigate environment instability issues including occasional CAPTCHA verification, virtual machine crashes, and other non-model-related failures, providing a more reliable evaluation of actual model capabilities.

Model	OSWorld-Verified	AndroidWorld
Claude-4-sonnet	43.9	-
Claude-4.5-sonnet	**61.4**	-
OpenAI CUA o3	23	-
Gemini-2.5-Computer Use	-	69.7
SeedVL-1.5	34.1	62.1
UI-TARS-1.5	42.5	64.2
UI-TARS-2	47.5	73.3
UI-TARS-1.5-7B	27.4	-
UI-TARS-72B-DPO	24.0	46.6
GUI-Owl-7B	34.9	66.4
Mobile-Agent-v3	37.7	73.3
Qwen3-VL-4B-Instruct	26.2	45.3
Qwen3-VL-8B-Instruct	33.9	47.6
Qwen3-30B-A3B-Instruct	30.3	54.3
GLM-4.6V	37.2	57.0
MobileRL-9B	-	**80.2**
Step-GUI-4B (Pass@1)	31.9	63.9
Step-GUI-8B (Pass@1)	40.2	67.7
Step-GUI-4B (Pass@3)	40.4	75.8
Step-GUI-8B (Pass@3)	48.5	**80.2**

exceeding SeedVL-1.5 (87.3) by +3.4 points.

These results validate the effectiveness of our multi-stage training pipeline: mid-training establishes robust visual grounding and agent-style format parsing, cold-start knowledge injection addresses task-critical gaps, and the iterative grounding-cleaning pipeline ensures high-quality supervision throughout training. Notably, our compact 4B and 8B models consistently match or outperform substantially larger models (30B–72B), demonstrating strong parameter efficiency.

Performance on End-to-End Benchmarks.

We evaluate our models on two challenging end-to-end environment benchmarks: OSWorld-Verified and AndroidWorld. Table 4 presents a comprehensive comparison with state-of-the-art closed-source and open-source models.

OSWorld-Verified. Xie et al. (2024) Due to the inherent instability of the OSWorld-Verified testing environment (including frequent VM crashes, substantial loading delays, and CAPTCHA interruptions), we employ the Pass@3 metric to mitigate failures caused by infrastructure issues rather than model limitations. On the OSWorld-Verified benchmark, our Step-GUI-8B achieves a score of 48.5, ranking second only to the closed-source Claude-4.5-sonnet (61.4) and significantly outperforming other competitive baselines. Notably, our model surpasses the powerful closed-source model OpenAI CUA o3 (23.0) by a substantial margin of +25.5 points, demonstrating superior reasoning and interaction capabilities in complex operating system environments. Our approach also outperforms specialized GUI understanding models such as SeedVL-1.5 (34.1), UI-TARS-1.5 (42.5), and UI-TARS-2 (47.5), as well as open-source alternatives like GUI-Owl-

Table 5 | Performance on AndroidDaily (Static) Benchmark. Best results are in **bold**, second best are underlined.

Model	CLICK	TYPE	SLIDE	AWAKE	INFO	COMPLETE	WAIT	LONG_PRESS	AVG
GPT-4o	12.65	52.41	31.17	10.84	0.00	44.85	5.88	0.00	17.73
Claude-4.5-sonnet	5.56	17.77	50.65	1.33	0.00	53.48	21.18	0.00	10.90
Gemini-2.5-Pro Thinking	51.97	63.25	46.75	0.00	0.00	68.99	21.18	33.33	43.74
UI-TARS-1.5	84.43	73.49	48.05	25.48	0.00	70.19	24.71	**66.67**	67.69
Step-GUI-4B	87.11	86.01	44.16	**99.43**	52.01	93.39	74.12	**66.67**	87.02
Step-GUI-8B	**88.37**	**88.25**	**71.43**	**99.43**	**86.04**	**94.41**	**95.29**	**66.67**	**89.91**

Table 6 | Performance on AndroidDaily (End-to-End) Benchmark.

Model	Task Type			Complexity			Ambiguity			Total
	Filter	Query	Analyze	Atomic	Comp.	Cond.	Low	Mid	High	
UI-TARS-1.5	57.64	65.97	36.71	61.41	13.64	60.38	57.05	54.90	57.89	56.64
Step-GUI-4B	44.77	64.29	33.72	54.03	19.61	42.86	51.21	38.32	59.52	49.06
Step-GUI-8B	52.50	63.82	32.95	59.09	14.00	42.86	54.08	44.55	61.54	52.50

7B (34.9). Furthermore, compared to our baseline models Qwen3-VL-4B-Instruct (26.2) and Qwen3-VL-8B-Instruct (33.9), Step-GUI-8B achieves remarkable improvements of +22.3 and +14.6 points respectively, validating the effectiveness of our training pipeline and model design. The smaller Step-GUI-4B variant also demonstrates competitive performance with a score of 40.4, outperforming models with significantly larger parameters such as UI-TARS-1.5-7B (27.4).

AndroidWorld. Rawles et al. (2024) Given the significant stability challenges in the AndroidWorld testing environment (including unstable ADB communication, frequent Android emulator crashes, and system response delays), we employ the Pass@3 metric to exclude failures caused by infrastructure faults rather than model capability limitations. On the AndroidWorld benchmark, our Step-GUI-8B achieves state-of-the-art performance with a score of 80.2, tying with MobileRL-9B and establishing new performance standards for mobile GUI agents. This represents a substantial improvement over both closed-source and open-source methods. Specifically, our model outperforms UI-TARS-2 (73.3), Mobile-Agent-v3 (73.3), and the recent Gemini-2.5-Computer Use (69.7) by margins of +6.9, +6.9, and +10.5 points respectively. Compared to open-source GUI understanding models, Step-GUI-8B surpasses GUI-Owl-7B (66.4) by +13.8 points and SeedVL-1.5 (62.1) by +18.1 points. The performance gain over our baseline models is even more pronounced, with improvements of +34.9 points over Qwen3-VL-4B-Instruct (45.3) and +32.6 points over Qwen3-VL-8B-Instruct (47.6). Our smaller Step-GUI-4B variant achieves 75.8, securing the second-best performance among all tested models and outperforming all other open-source alternatives.

These results collectively demonstrate that our approach achieves exceptional performance across diverse GUI interaction scenarios, establishing new benchmarks for both desktop and mobile environments through our effective training pipeline and data construction methodology.

Performance on AndroidDaily. We report evaluation results on both static action prediction and end-to-end task completion protocols.

Static Action Prediction. As shown in Table 5, Step-GUI-4B and Step-GUI-8B achieve state-of-the-art performance with average accuracies of 87.02% and 89.91% respectively, substantially outperforming all baseline models. UI-TARS-1.5 demonstrates competitive performance at 67.69%, while general-purpose foundation models including GPT-4o (17.73%), Claude-4.5-

Table 7 | Performance on Mainstream Multimodal Benchmarks. Best results are in **bold**, second best are underlined.

Benchmark	GLM-4.6V Flash	Gemini-2.5 flash-lite	Qwen3-VL 8B-Instruct	Qwen3-VL-30B A3B-Instruct	GPT5-nano minimal	Step-GUI-4B	Step-GUI-8B
V*	-	-	86.4	86.4	66.5	79.1	**89.0**
PhyX	-	-	28.3	31.1	24.0	33.0	**41.9**
OmniOCR	-	-	**78.7**	74.4	13.5	73.6	77.1
OCRBench	84.2	81.3	89.7	**90.8**	70.1	84.6	88.0
LogicVista	-	-	52.3	**58.2**	40.3	52.1	53.7
MMBench_en	84.7	82.4	89.9	**90.8**	51.6	87.2	89.1
MMBench_cn	85.8		88.0	**89.5**	82.2	87.0	88.0
MMStar	**72.9**	71.3	69.5	69.4	41.3	64.7	68.0
MathVista_mini	80.0	70.3	74.6	**80.1**	40.9	72.3	74.4
CharXiv_DQ	-	73.5	83.0	85.5	64.4	84.1	**86.3**
CharXiv_RQ	-	44.6	46.4	**48.9**	31.7	39.2	44.6
SimpleVQA	-	52.2	50.2	**52.7**	39.0	48.1	50.8

sonnet (10.90%), and Gemini-2.5-Pro Thinking (43.74%) lag significantly behind. This substantial performance gap indicates that Step-GUI and UI-TARS-1.5 possess superior domain-specific knowledge for Chinese mobile operations.

Across different action types, Step-GUI models consistently dominate. For basic interactions, both variants exceed 85% accuracy on CLICK and TYPE actions. Notably, Step-GUI-8B achieves 71.43% on SLIDE actions, significantly outperforming UI-TARS-1.5 (48.05%), demonstrating better spatial reasoning for swipe gestures. For sophisticated operations including AWAKE, INFO, COMPLETE, WAIT, and LONG_PRESS, Step-GUI models maintain strong performance with Step-GUI-8B achieving particularly high accuracy on INFO (86.04%) and WAIT (95.29%) actions.

End-to-End Task Completion. Table 6 presents results where models interact with real applications. UI-TARS-1.5 achieves the highest success rate of 56.64%, while Step-GUI-4B and Step-GUI-8B achieve comparable performance at 49.06% and 52.50% respectively. Despite UI-TARS-1.5's advantage in this setting, Step-GUI models demonstrate competitive end-to-end task completion capabilities, with Step-GUI-8B trailing by only 4.14 percentage points while significantly outperforming on static action prediction (89.91% vs 67.69%).

Analyzing by task type, all models perform best on Query tasks (requiring information retrieval) with success rates around 64-66%, while Analyze tasks (demanding complex reasoning) prove most challenging with rates dropping to 33-37%. For complexity dimensions, Atomic tasks show strong performance (54-61%), while Composite tasks remain challenging for all models (14-20%), indicating the difficulty of multi-step operation planning. Interestingly, examining instruction ambiguity reveals that Step-GUI-8B excels on high-ambiguity cases (61.54%), surpassing UI-TARS-1.5 (57.89%), demonstrating enhanced robustness in interpreting underspecified instructions, a critical capability for real-world deployment.

Performance on General Multi-modal Benchmarks. To demonstrate that our model maintains strong general-purpose capabilities rather than being narrowly specialized for GUI tasks, we conduct comprehensive evaluations on mainstream multi-modal benchmarks. As shown in Table 7, **Step-GUI-8B** achieves superior or competitive performance across diverse evaluation scenarios compared to the base model Qwen3-VL-8B-Instruct and the larger Qwen3-30BA3B-Instruct.

Specifically, Step-GUI-8B achieves **89.0** on V*, surpassing both Qwen3-VL-8B-Instruct (86.4)

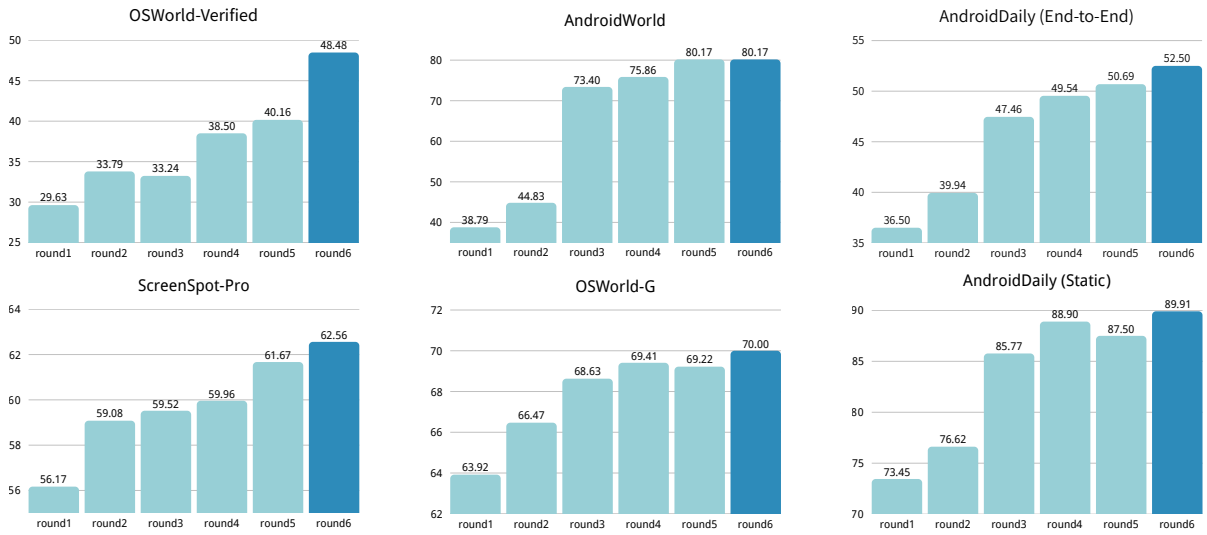

Figure 7 | Performance Evolution of Step-GUI-8B Across Six Self-Evolving Training Rounds.

and Qwen3-30BA3B-Instruct (86.4) by a significant margin. On OCRBench, our model scores **88.0**, outperforming Qwen3-VL-8B-Instruct (89.7) while maintaining comparable performance to Qwen3-30BA3B-Instruct (90.8). For mathematical reasoning evaluated on MathVista$_{mini}$, Step-GUI-8B achieves **74.4**, demonstrating strong reasoning capabilities that exceed the base model (74.6) while approaching the performance of the larger 30B model (80.1). Notably, our model demonstrates consistent improvements across diverse benchmarks. Step-GUI-8B achieves **87.2** on MMBench_en and **88.0** on MMBench_cn for general multi-modal understanding, **68.0** on MMStar for visual reasoning, and strong performance on document understanding tasks (CharXiv$_{DQ}$: **86.3**, OmniOCR: **77.1**), indicating that GUI-oriented training does not compromise general multi-modal understanding capabilities. Furthermore, Step-GUI-4B, despite having only half the parameters, maintains competitive performance across most benchmarks, achieving **84.6** on OCRBench, which validates the effectiveness of our training approach.

These results demonstrate that our training methodology successfully enhances GUI-specific capabilities while *preserving and even improving* general-purpose multi-modal understanding abilities. This characteristic is crucial for practical deployment, as it enables the model to handle diverse real-world tasks beyond GUI interaction without requiring separate specialized models.

5.3. Detailed Analysis

5.3.1. Self-Evolving Training Dynamics

Based on the experimental data spanning Rounds 1 through 6 (as illustrated in the figure7), we validate the effectiveness of the self-evolving training framework proposed in Step-GUI.

"Phase Transition" on AndroidWorld. On the AndroidWorld benchmark, we observe a remarkable performance jump from Round 2 (44.83%) to Round 3 (73.40%). We attribute this surge to the Date Flow 1 (Generation) mechanism within the CSRS. During the initial phase (Round 1), the model addresses fundamental knowledge gaps via the Cold-Start stage. As the model's capabilities reach a critical threshold in Round 2, the CSRS begins to effectively capture and verify successful long-horizon trajectories generated during exploration. These self-discovered, high-quality trajectories enriched with Chain-of-Thought serve as high-value signals that flow back into the training pipeline, catalyzing an explosive growth in performance.

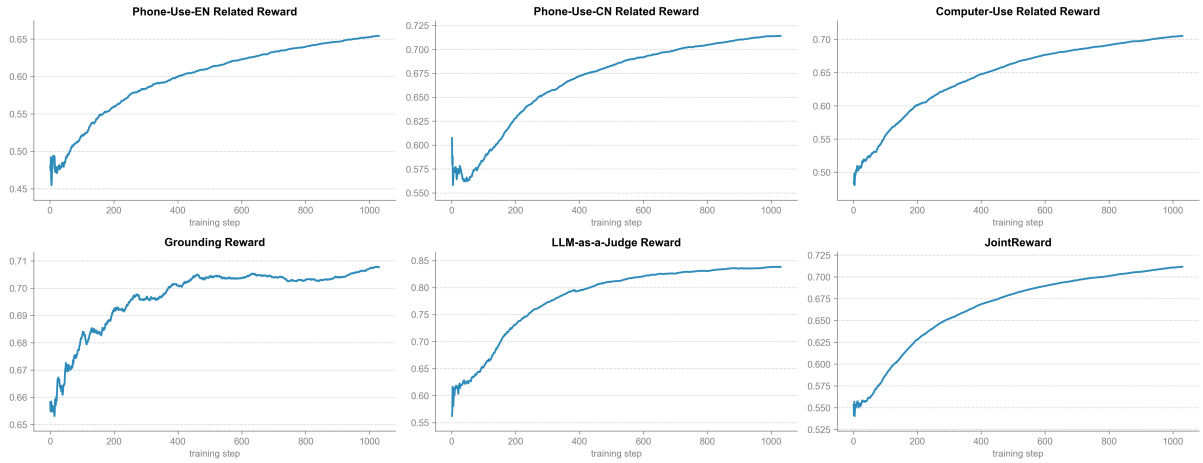

Figure 8 | Reward Dynamics during RLVR Training. The smooth, monotonic ascent across task-specific sub-rewards (Top) and the aggregate JointReward (Bottom Right) demonstrates stable convergence without oscillatory collapse. The strong correlation between LLM-as-a-Judge Reward and objective metrics confirms the policy improves genuine capabilities while avoiding reward hacking.

Steady Advancement on OSWorld. Compared to the mobile domain, desktop tasks on OSWorld present greater complexity. Our data indicates a steady, quasi-linear improvement on OSWorld performance, rising from 29.63% to 46.26%. This trajectory highlights the efficacy of the Date Flow 1 (Refinement). By employing "Rejection Sampling" and "Self-Distillation" on challenging samples, the system continuously identifies model weaknesses near complex decision boundaries. It subsequently converts failed trajectories into Knowledge Data for targeted reinforcement, thereby enabling sustained breakthroughs in long-horizon, complex tasks.

Furthermore, as illustrated by the AndroidDaily (End-to-End) results in figure 7, the performance curve exhibits a smooth upward trend (49.06% → 65.06%). This stability stems from the CSRS design, which discards traditional step-level rewards in favor of rewards assigned via objective verification of final task success. This sparse yet high-confidence signal ensures the purity of the data fed back into the system, thereby guaranteeing robustness across multiple training iterations. In conclusion, the synchronous capability improvements observed across diverse domains indicate that Step-GUI's self-evolving system successfully establishes a robust, general-purpose multimodal foundation through closed-loop training iterations. The system effectively transmutes computational resources into data quality, which subsequently evolves into model intelligence. This process marks a paradigm shift, transitioning the model from a reliance on human priors to a mechanism driven by self-exploration and reflection.

5.3.2. RLVR Training: Stability and Convergence

To rigorously assess the stability and convergence properties of the Step-GUI during RLVR stage, we monitor the optimization dynamics through a comprehensive suite of reward signals and off-policy correction metrics. These metrics quantify the distributional shift between the rollout policy (π_{rollout}) and the training policy (π_{train}), validating the robustness of our training.

Reward Consistency and Alignment. As illustrated in Figure 8, the training process exhibits a remarkably smooth and monotonic ascent across all signal sources. The JointReward rises steadily without the oscillatory behavior or collapse often associated with RL training. Notably,

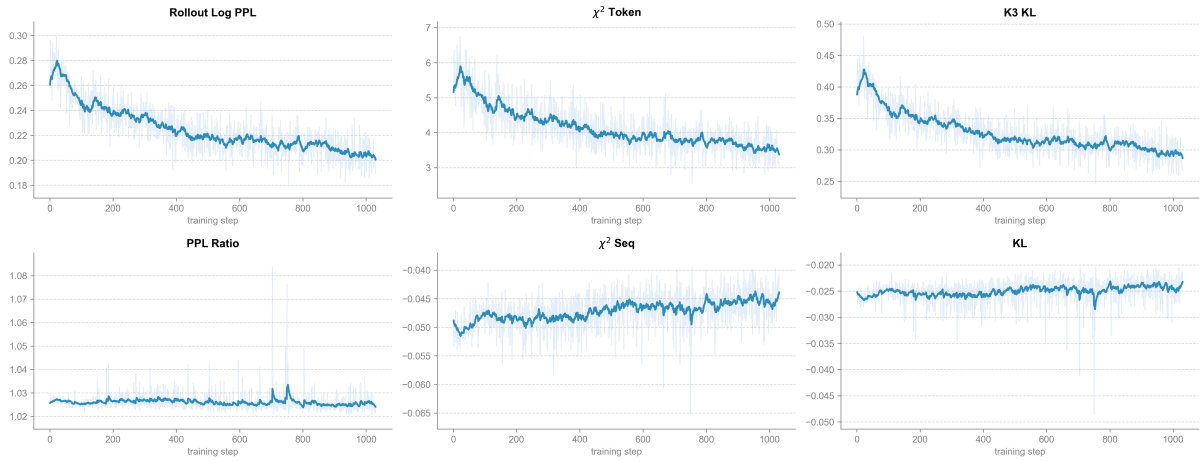

Figure 9 | Off-Policy Correction and Stability Diagnostics. The downward trends in Rollout Log PPL and χ^2 (Token/Seq) variances indicate increasing model confidence and reduced gradient estimation noise. Complementing this, the decreasing K3-KL divergence demonstrates effective suppression of extreme policy deviations, while the low and stable PPL Ratio and KL confirm high synchronization between training and rollout policies, ensuring updates remain within a safe trust region.

the sub-rewards for specific capabilities, including phone-use-en, phone-use-cn, computer-use, and grounding, show high correlation with the LLM-as-a-Judge reward curve, which is demonstrated to align well with human annotations. This synchronization confirms that our composite reward function effectively anchors the model's optimization trajectory to human-aligned reasoning, preventing reward hacking.

Policy Synchronization. We scrutinize the off-policy divergence through the diagnostic metrics presented in Figure 9. First, the Rollout Log PPL displays a consistent decline (from ~ 0.28 to ~ 0.20), indicating that the rollout policy progressively gains confidence and reduces perplexity in its generated trajectories. Concurrently, the PPL Ratio, quantifying the deviation between π_{train} and π_{rollout}, remains consistently low and stable, fluctuating strictly between 1.02 and 1.03. This dual observation confirms that even as the model becomes more deterministic, the training policy maintains high synchronization with the rollout policy, ensuring that gradient updates remain valid and effectively "on-policy" despite the inherent lag in data collection.

Tail Risk Suppression. To measure distributional divergence, we monitor both standard KL divergence and the K3-KL estimator ($E[e^{\Delta} - \Delta - 1]$), which is highly sensitive to extreme deviations in the policy's tail distribution. While standard KL remains stable, the K3-KL metric in Figure 9 exhibits a distinct downward trend (from ~ 0.45 to ~ 0.30). This demonstrates that the model progressively minimizes "surprise" or extreme deviations from the exploration trajectory, keeping updates within a safe trust region.

Variance Reduction. We further employ Importance Sampling (IS) with a sequence-truncate strategy to stabilize gradients. The effectiveness of this mechanism is evidenced by both the token-level and sequence-level χ^2 metrics, which approximate the variance of importance weights $Var(\rho)$. As shown in Figure 9, the χ^2 Token metric exhibits a consistent decline from ~ 5.5 to ~ 3.5, indicating that local weight fluctuations are progressively smoothed out. Complementing this, the χ^2 Sequence maintains a stable, low-magnitude profile throughout training. This joint behavior implies that the importance weights are becoming more uniform at both the fine-grained token level and the holistic sequence level. Consequently, the reduction in gradient

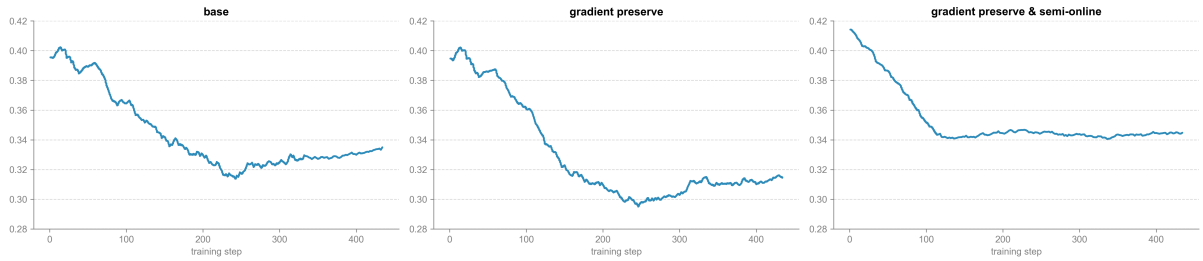

Figure 10 | Evolution of Policy Entropy under Different Training Strategies. Baseline GRPO (Left) shows rapid entropy decay followed by stabilization. Gradient Preservation (Middle) further reduces entropy, indicating tighter exploitation but risking mode collapse. Semi-Online strategy (Right) significantly revitalizes entropy by injecting ground-truth hints into failed rollouts, effectively disrupting narrow distributions and sustaining healthy exploration as a counter-force to premature convergence.

estimation noise directly substantiates the smooth convergence observed in the reward curves.

Training Entropy. To evaluate exploration-exploitation dynamics, we analyze policy entropy evolution in Figure 10. The baseline GRPO (Left) exhibits rapid decay followed by stabilization. Integrating Gradient Preservation (Middle) leads to a further reduction in entropy, indicating tighter exploitation of clipped samples but a heightened risk of mode collapse. In contrast, the Semi-Online strategy (Right) significantly revitalizes entropy levels. By injecting ground-truth hints into failed rollouts, this mechanism effectively disrupts narrow distributions and sustains healthy exploration, serving as a crucial counter-force to premature convergence.

6. Related Work

6.1. Reasoning with Language Models

Chain-of-thought (CoT) prompting Wei et al. (2022) demonstrates that eliciting intermediate reasoning steps significantly enhances large language models' performance on complex reasoning tasks. Building on this foundation, synthetic prompting Shao et al. (2023) and STaR Zelikman et al. (2022) explore automated generation of reasoning demonstrations: the former employs backward-forward construction to create high-quality exemplars, while the latter bootstraps rationale datasets from minimal seed examples and introduces rationalization, training models to justify correct answers post-hoc. Tree of Thoughts Yao et al. (2023) extends CoT to breadth-first search over alternative reasoning branches, enabling backtracking and global solution evaluation. Quiet-STaR Zelikman et al. (2024) further generalizes this paradigm to learn reasoning from unstructured text across all token positions through parallel sampling.

To enhance reasoning reliability, recent work introduces verification mechanisms. Self-consistency Wang et al. (2022) samples multiple reasoning paths and selects the most frequent answer, while self-verification Weng et al. (2023) and Self-Refine Madaan et al. (2023) implement iterative refinement through self-critique. Addressing hallucinations in knowledge-intensive scenarios, CoT-RAG Li et al. (2025a) proposes knowledge graph-driven CoT generation with expert-provided decision trees that encapsulate domain reasoning logic, while self-refinement-enhanced knowledge retrieval Niu et al. (2024) identifies high-hallucination tokens via attribution analysis and corrects inaccuracies through targeted retrieval. DDCoT Zheng et al. (2023) extends this to multimodal reasoning by partitioning responsibilities between vision models and language models through duty-distinct prompting. For self-improvement through data

generation, impossible distillation Jung et al. (2023) filters low-quality samples generated from suboptimal models, while ReST Gulcehre et al. (2023) employs offline reinforcement learning to iteratively improve policies from self-generated data. Despite these advances, efficiently obtaining high-quality trajectory and reasoning data for target domains remains challenging; we address this through a CSRS-centered data flywheel that ensures annotation quality while maintaining scalability.

6.2. GUI Agents

Recent work, including UI-TARS Qin et al. (2025), UI-TARS-2 Wang et al. (2025a), OpenCUA Wang et al. (2025c), GUI-Owl Ye et al. (2025), and UITron Zeng et al. (2025), systematically investigates data curation, cleaning pipelines, and training methodologies spanning pretraining and post-training phases. Concurrently, mainstream foundation models such as Claude series Anthropic (2025c), Seed series Guo et al. (2025a), and Qwen series **?** begin natively integrating GUI interaction capabilities, underscoring the strategic importance of this research direction. Most GUI agents follow the ReAct paradigm Yao et al. (2022), observe, reason via CoT, then act, with recent systems extending this through multi-agent collaboration and hierarchical architectures. Notable examples include Mobile-Agent-v3 Ye et al. (2025), which introduces specialized agents (Manager, Worker, Reflector, Notetaker) for knowledge evolution and reflective reasoning, and other systems Agashe et al. (2025); Liu et al. (2025a); Song et al. (2025); Zhang et al. (2025) that explore hierarchical planning, code-visual hybrid control, and compositional agent coordination. We propose the Step-GUI series models, achieving state-of-the-art performance among models of similar size, while Step-GUI-4B can run fully locally on consumer-grade hardware, enabling deployment in privacy-sensitive scenarios.

To evaluate these advances, the community develops both static benchmarks including ScreenSpot Cheng et al. (2024), ScreenSpot-v2 Wu et al. (2024), ScreenSpot-Pro Li et al. (2025c), FuncPred Li et al. (2025b), MoTIF Burns et al. (2022), RefExp Rastogi et al. (2021), Llama-Touch Zhang et al. (2024), VWB-AG Liu et al. (2024a), and VWB-EG Liu et al. (2024a), that primarily assess element grounding and task planning accuracy through prediction-annotation matching, and interactive benchmarks such as AndroidWorld Rawles et al. (2024) and OSWorld Xie et al. (2024) that enable agents to execute tasks within realistic or virtualized operating system environments with multi-dimensional task completion evaluation. In contrast to these efforts, we focus on scalable reasoning trajectory synthesis and efficient agent architecture design for real-world GUI automation scenarios.

7. Conclusion

In this work, we present a holistic framework advancing practical GUI agents across three dimensions: data, deployment, and evaluation. We introduce the Calibrated Step Reward System (CSRS), a self-evolving pipeline that drastically reduces annotation costs while enabling the training of SOTA Step-GUI models (4B/8B). To standardize deployment, we propose GUI-MCP, a hierarchical privacy-centric protocol that balances execution efficiency with on-device data security. Finally, we contribute AndroidDaily, a benchmark grounded in authentic usage patterns to rigorously evaluate real-world utility. Together, these innovations bridge the critical gap between research capabilities and reliable daily assistance.

Contributors

Our contributor list is primarily sorted alphabetically by first name, with the project leader placed at the very end. ⋆ denoting GELab team, and † indicates the project leader.

Core Contributors

Algorithm: Haolong Yan⋆, Jia Wang⋆, Xin Huang⋆, Yeqing Shen⋆, Ziyang Meng⋆, Zhimin Fan, Kaijun Tan⋆†

Infra: Jin Gao, Lieyu Shi, Mi Yang, Shiliang Yang, Zhirui Wang, Brian Li†, Kang An†

Application: Chenyang Li, Lei Lei, Mengmeng Duan, Danxun Liang†

Data: Guodong Liu, Hang Cheng, Hao Wu, Jie Dong, Junhao Huang, Mei Chen, Renjie Yu, Shunshan Li, Xu Zhou, Yiting Dai, Yineng Deng, Yingdan Liang, Zelin Chen, Sun Wen†

Contributors

Chengxu Yan, Chunqin Xu, Dong Li, Fengqiong Xiao, Guanghao Fan, Guopeng Li, Guozhen Peng, Hongbing Li, Hang Li, Hongming Chen, Jingjing Xie, Jianyong Li, Jingyang Zhang, Jiaju Ren, Jiayu Yuan, Jianpeng Yin, Kai Cao, Liang Zhao, Liguo Tan, Liying Shi, Mengqiang Ren, Min Xu, Manjiao Liu, Mao Luo, Mingxin Wan, Na Wang, Nan Wu, Ning Wang, Peiyao Ma, Qingzhou Zhang, Qiao Wang, Qinlin Zeng, Qiong Gao, Qiongyao Li, Shangwu Zhong, Shuli Gao, Shaofan Liu, Shisi Gao, Shuang Luo, Xingbin Liu, Xiaojia Liu, Xiaojie Hou, Xin Liu, Xuanti Feng, Xuedan Cai, Xuan Wen, Xianwei Zhu, Xin Liang, Xin Liu, Xin Zhou, Yingxiu Zhao, Yukang Shi, Yunfang Xu, Yuqing Zeng, Yixun Zhang, Zejia Weng, Zhonghao Yan, Zhiguo Huang, Zhuoyu Wang

Supervisor: Zheng Ge, Jing Li, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Daxin Jiang

References

S. Agashe, K. Wong, V. Tu, J. Yang, A. Li, and X. E. Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. arXiv preprint arXiv:2504.00906, 2025.

Anthropic. Model context protocol. https://modelcontextprotocol.io/, 2023.

Anthropic. Claude sonnet 4.5. `https://www.anthropic.com/claude/sonnet`, 2025a.

Anthropic. Claude 4. `https://www.anthropic.com/news/claude-4`, 2025b.

Anthropic. Introducing claude opus 4.5. `https://www.anthropic.com/news/claude-opus-4-5`, 2025c. Accessed: .

S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, Z. Cheng, L. Deng, W. Ding, C. Gao, C. Ge, W. Ge, Z. Guo, Q. Huang, J. Huang, F. Huang, B. Hui, S. Jiang, Z. Li, M. Li, M. Li, K. Li, Z. Lin, J. Lin, X. Liu, J. Liu, C. Liu, Y. Liu, D. Liu, S. Liu, D. Lu, R. Luo, C. Lv, R. Men, L. Meng, X. Ren, X. Ren, S. Song, Y. Sun, J. Tang, J. Tu, J. Wan, P. Wang, P. Wang, Q. Wang, Y. Wang, T. Xie, Y. Xu, H. Xu, J. Xu, Z. Yang, M. Yang, J. Yang, A. Yang, B. Yu, F. Zhang, H. Zhang, X. Zhang, B. Zheng, H. Zhong, J. Zhou, F. Zhou, J. Zhou, Y. Zhu, and K. Zhu. Qwen3-vl technical report. arXiv preprint arXiv:2511.21631, 2025a.

S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al. Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923, 2025b.

A. Burns, D. Arsan, S. Agrawal, R. Kumar, K. Saenko, and B. A. Plummer. A dataset for interactive vision-language navigation with unknown command feasibility. In European Conference on Computer Vision, pages 312–328. Springer, 2022.

L. Chen, J. Li, X. Dong, P. Zhang, Y. Zang, Z. Chen, H. Duan, J. Wang, Y. Qiao, D. Lin, et al. Are we on the right way for evaluating large vision-language models? Advances in Neural Information Processing Systems, 37:27056–27087, 2024.

K. Cheng, Q. Sun, Y. Chu, F. Xu, Y. Li, J. Zhang, and Z. Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. arXiv preprint arXiv:2401.10935, 2024.

X. Cheng, W. Zhang, S. Zhang, J. Yang, X. Guan, X. Wu, X. Li, G. Zhang, J. Liu, Y. Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4637–4646, 2025.

G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. 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.

T. Gan and Q. Sun. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation. arXiv preprint arXiv:2505.03275, 2025.

L. Gao, L. Zhang, S. Wang, S. Wang, Y. Li, and M. Xu. Mobileviews: A large-scale mobile gui dataset. arXiv preprint arXiv:2409.14337, 2024.

Google. Introducing the gemini 2.5 computer use model. `https://blog.google/technology/google-deepmind/gemini-computer-use-model`, 2025.

C. Gulcehre, T. L. Paine, S. Srinivasan, K. Konyushkova, L. Weerts, A. Sharma, A. Siddhant, A. Ahern, M. Wang, C. Gu, et al. Reinforced self-training (rest) for language modeling. arXiv preprint arXiv:2308.08998, 2023.

D. Guo, F. Wu, F. Zhu, F. Leng, G. Shi, H. Chen, H. Fan, J. Wang, J. Jiang, J. Wang, et al. Seed1. 5-vl technical report. arXiv preprint arXiv:2505.07062, 2025a.

D. Guo, F. Wu, F. Zhu, F. Leng, G. Shi, H. Chen, H. Fan, J. Wang, J. Jiang, J. Wang, et al. Seed1. 5-vl technical report. arXiv preprint arXiv:2505.07062, 2025b.

J. Jung, P. West, L. Jiang, F. Brahman, X. Lu, J. Fisher, T. Sorensen, and Y. Choi. Impossible distillation: from low-quality model to high-quality dataset & model for summarization and paraphrasing. arXiv preprint arXiv:2305.16635, 2023.

F. Li, P. Fang, Z. Shi, A. Khan, F. Wang, D. Feng, W. Wang, X. Zhang, and Y. Cui. Cot-rag: Integrating chain of thought and retrieval-augmented generation to enhance reasoning in large language models. arXiv preprint arXiv:2504.13534, 2025a.

H. Li, J. Chen, J. Su, Y. Chen, Q. Li, and Z. Zhang. Autogui: Scaling gui grounding with automatic functionality annotations from llms. arXiv preprint arXiv:2502.01977, 2025b.

K. Li, Z. Meng, H. Lin, Z. Luo, Y. Tian, J. Ma, Z. Huang, and T.-S. Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In Proceedings of the 33rd ACM International Conference on Multimedia, pages 8778–8786, 2025c.

H. Liu, X. Zhang, H. Xu, Y. Wanyan, J. Wang, M. Yan, J. Zhang, C. Yuan, C. Xu, W. Hu, et al. Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc. arXiv preprint arXiv:2502.14282, 2025a.

J. Liu, Y. Song, B. Y. Lin, W. Lam, G. Neubig, Y. Li, and X. Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? arXiv preprint arXiv:2404.05955, 2024a.

Y. Liu, H. Duan, Y. Zhang, B. Li, S. Zhang, W. Zhao, Y. Yuan, J. Wang, C. He, Z. Liu, et al. Mmbench: Is your multi-modal model an all-around player? In European conference on computer vision, pages 216–233. Springer, 2024b.

Y. Liu, Z. Li, M. Huang, B. Yang, W. Yu, C. Li, X.-C. Yin, C.-L. Liu, L. Jin, and X. Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. Science China Information Sciences, 67(12):220102, 2024c.

Z. Liu, J. Xie, Z. Ding, Z. Li, B. Yang, Z. Wu, X. Wang, Q. Sun, S. Liu, W. Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. arXiv preprint arXiv:2509.15221, 2025b.

P. Lu, H. Bansal, T. Xia, J. Liu, C. Li, H. Hajishirzi, H. Cheng, K.-W. Chang, M. Galley, and J. Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. arXiv preprint arXiv:2310.02255, 2023.

A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, et al. Self-refine: Iterative refinement with self-feedback. Advances in Neural Information Processing Systems, 36:46534–46594, 2023.

M. Niu, H. Li, J. Shi, H. Haddadi, and F. Mo. Mitigating hallucinations in large language models via self-refinement-enhanced knowledge retrieval. arXiv preprint arXiv:2405.06545, 2024.

OmniAI. Omniai ocr benchmark. `https://getomni.ai/blog/ocr-benchmark`, 2025.

OpenAI. Gpt-5 system card. 2025a.

OpenAI. Openai o3 operator. `https://openai.com/index/o3-o4-mini-system-card-addendum-operator-o3/`, 2025b.

C. Ouyang, L. Yue, S. Di, L. Zheng, L. Yue, S. Pan, J. Yin, and M.-L. Zhang. Code2mcp: Transforming code repositories into mcp services. arXiv preprint arXiv:2509.05941, 2025.

Y. Qin, Y. Ye, J. Fang, H. Wang, S. Liang, S. Tian, J. Zhang, J. Li, Y. Li, S. Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. arXiv preprint arXiv:2501.12326, 2025.

C. Qu, S. Dai, X. Wei, H. Cai, S. Wang, D. Yin, J. Xu, and J.-R. Wen. Tool learning with large language models: A survey. Frontiers of Computer Science, 19(8):198343, 2025.

A. K. Rastogi, B. A. y Arcas, C. Bai, J. J. Chen, S. K. Sunkara, X. Zang, and Y. Xu. Uibert: Learning generic multimodal representations for ui understanding. 2021.

C. Rawles, A. Li, D. Rodriguez, O. Riva, and T. Lillicrap. Androidinthewild: A large-scale dataset for android device control. In Advances in Neural Information Processing Systems, 36:59708–59728, 2023.

C. Rawles, S. Clinckemaillie, Y. Chang, J. Waltz, G. Lau, M. Fair, A. Li, W. Bishop, W. Li, F. Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. arXiv preprint arXiv:2405.14573, 2024.

Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In International conference on machine learning, pages 30706–30775. PMLR, 2023.

H. Shen, T. Wu, Q. Han, Y. Hsieh, J. Wang, Y. Zhang, Y. Cheng, Z. Hao, Y. Ni, X. Wang, et al. Phyx: Does your model have the" wits" for physical reasoning? arXiv preprint arXiv:2505.15929, 2025.

L. Song, Y. Dai, V. Prabhu, J. Zhang, T. Shi, L. Li, J. Li, S. Savarese, Z. Chen, J. Zhao, et al. Coact-1: Computer-using agents with coding as actions. arXiv preprint arXiv:2508.03923, 2025.

Z. Su, L. Pan, M. Lv, Y. Li, W. Hu, F. Zhang, K. Gai, and G. Zhou. Ce-gppo: Coordinating entropy via gradient-preserving clipping policy optimization in reinforcement learning. arXiv preprint arXiv:2509.20712, 2025.

H. Wang, H. Zou, H. Song, J. Feng, J. Fang, J. Lu, L. Liu, Q. Luo, S. Liang, S. Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. arXiv preprint arXiv:2509.02544, 2025a.

W. Wang, Z. Gao, L. Gu, H. Pu, L. Cui, X. Wei, Z. Liu, L. Jing, S. Ye, J. Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. arXiv preprint arXiv:2508.18265, 2025b.

X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171, 2022.

X. Wang, B. Wang, D. Lu, J. Yang, T. Xie, J. Wang, J. Deng, X. Guo, Y. Xu, C. H. Wu, et al. Opencua: Open foundations for computer-use agents. arXiv preprint arXiv:2508.09123, 2025c.

X. Wang, Z. Wu, J. Xie, Z. Ding, B. Yang, Z. Li, Z. Liu, Q. Li, X. Dong, Z. Chen, et al. Mmbench-gui: Hierarchical multi-platform evaluation framework for gui agents. arXiv preprint arXiv:2507.19478, 2025d.

Z. Wang, M. Xia, L. He, H. Chen, Y. Liu, R. Zhu, K. Liang, X. Wu, H. Liu, S. Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. Advances in Neural Information Processing Systems, 37:113569–113697, 2024.

J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35:24824–24837, 2022.

Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao. Large language models are better reasoners with self-verification. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 2550–2575, 2023.

P. Wu and S. Xie. V?: Guided visual search as a core mechanism in multimodal llms. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 13084–13094, 2024.

Z. Wu, Z. Wu, F. Xu, Y. Wang, Q. Sun, C. Jia, K. Cheng, Z. Ding, L. Chen, P. P. Liang, et al. Os-atlas: A foundation action model for generalist gui agents. arXiv preprint arXiv:2410.23218, 2024.

Y. Xiao, E. Sun, T. Liu, and W. Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts. arXiv preprint arXiv:2407.04973, 2024.

T. Xie, D. Zhang, J. Chen, X. Li, S. Zhao, R. Cao, T. J. Hua, Z. Cheng, D. Shin, F. Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Advances in Neural Information Processing Systems, 37:52040–52094, 2024.

Y. Xu, X. Liu, X. Liu, J. Fu, H. Zhang, B. Jing, S. Zhang, Y. Wang, W. Zhao, and Y. Dong. Mobilerl: Online agentic reinforcement learning for mobile gui agents. arXiv preprint arXiv:2509.18119, 2025.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations, 2022.

S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems, 36:11809–11822, 2023.

J. Ye, X. Zhang, H. Xu, H. Liu, J. Wang, Z. Zhu, Z. Zheng, F. Gao, J. Cao, Z. Lu, et al. Mobile-agent-v3: Fundamental agents for gui automation. arXiv preprint arXiv:2508.15144, 2025.

Z.ai. Autoglm goes open source. `https://autoglm.z.ai/blog`, 2025a.

Z.ai. Introducing the gemini 2.5 computer use model. `https://z.ai/blog/glm-4.6v`, 2025b.

E. Zelikman, Y. Wu, J. Mu, and N. Goodman. Star: Bootstrapping reasoning with reasoning. Advances in Neural Information Processing Systems, 35:15476–15488, 2022.

E. Zelikman, G. Harik, Y. Shao, V. Jayasiri, N. Haber, and N. D. Goodman. Quiet-star: Language models can teach themselves to think before speaking. arXiv preprint arXiv:2403.09629, 2024.

Z. Zeng, J. Huang, L. Zheng, W. Han, Y. Zhong, L. Chen, L. Yang, Y. Chu, Y. He, and L. Ma. Uitron: Foundational gui agent with advanced perception and planning. arXiv preprint arXiv:2508.21767, 2025.

C. Zhang, H. Huang, C. Ni, J. Mu, S. Qin, S. He, L. Wang, F. Yang, P. Zhao, C. Du, et al. Ufo2: The desktop agentos. arXiv preprint arXiv:2504.14603, 2025.

L. Zhang, S. Wang, X. Jia, Z. Zheng, Y. Yan, L. Gao, Y. Li, and M. Xu. Llamatouch: A faithful and scalable testbed for mobile ui task automation. In Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology, pages 1–13, 2024.

G. Zheng, B. Yang, J. Tang, H.-Y. Zhou, and S. Yang. Ddcot: Duty-distinct chain-of-thought prompting for multimodal reasoning in language models. Advances in Neural Information Processing Systems, 36:5168–5191, 2023.