CRAFT-GUI: CURRICULUM-REINFORCED AGENT FOR GUI TASKS

Songqin Nong[1], Xiaoxuan Tang[1], Jingxuan Xu[1], Sheng Zhou[2], Jianfeng Chen[1], Tao Jiang[1], Wenhao Xu[1,*]
[1]Ant Group, [2]Zhejiang University
{nongsongqin.nsq, tangxiaoxuan.txx, xujingxuan.xjx, jianfengchen.cjf, tara.jt, hao.xuwh}@antgroup.com
zhousheng@zju.edu.cn

ABSTRACT

As autonomous agents become adept at understanding and interacting with graphical user interface (GUI) environments, a new era of automated task execution is emerging. Recent studies have demonstrated that Reinforcement Learning (RL) can effectively enhance agents' performance in dynamic interactive GUI environments.

However, these methods face two key limitations: (1) they overlook the significant variation in difficulty across different GUI tasks by treating the entire training data as a uniform set, which hampers the agent's ability to adapt its learning process; and (2) most approaches collapse task-specific nuances into a single, coarse reward, leaving the agent with a uniform signal that yields inefficient policy updates. To address these limitations, we propose **CRAFT-GUI**, a curriculum learning framework based on Group Relative Policy Optimization (GRPO) that explicitly accounts for the varying difficulty across trajectories. To enable more fine-grained policy optimization, we design a reward function that combines simple rule-based signals with model-judged evaluation, providing richer and more nuanced feedback during training. Experimental results demonstrate that our method achieves significant improvements over previous state-of-the-art approaches, outperforming them by 7.1% on public benchmarks AndroidWorld and 10.3% on our private constructed dataset, respectively. These findings empirically validate the effectiveness of integrating reinforcement learning with curriculum learning in GUI interaction tasks.

1 INTRODUCTION

Graphical User Interface (GUI) agents offer a promising approach for automating software interactions by autonomously perceiving and manipulating visual elements. Unlike traditional automation tools Rawles et al. (2023); Wang et al. (2025) that depend on structured inputs, modern GUI agents Qin et al. (2025); Gou et al. (2024); Lin et al. (2025) leverage multimodal large language models to directly interpret GUI screenshots and perform user-specified tasks, enabling robust automation even without API access.

Although recent studies have made progress, current GUI agents Cheng et al. (2024); Gou et al. (2024); Lu et al. (2025b); Luo et al. (2025); Nong et al. (2024) typically treat all training instances uniformly. However, real-world GUI tasks exhibit substantial variation in difficulty, as shown in Figure 1. Ignoring these differences leads to two critical drawbacks: *(i) optimization instability during training and (ii) constrained model capability growth*. Uniform treatment of samples impedes the agent's ability to adaptively adjust its learning based on task complexity—analogous to expecting a child to simultaneously master elementary and university curricula.

Furthermore, when applying reinforcement learning with verified rewards, existing reward functions rely on coarse, rule-based heuristics that fail to differentiate task difficulty. Rather than fundamentally addressing how task difficulty impacts learning, current methods Lu et al. (2025a) resort to intricate

*Corresponding Author

Figure 1: Contrast between complex and simple GUI tasks.

task selection and experience replay strategies, adding unnecessary complexity while neglecting nuanced, task-aware feedback for efficient policy optimization.

Inspired by how humans naturally progress from simple to complex GUI interactions, we propose a curriculum-aware reinforcement learning framework tailored to GUI agent training. While previous works Team et al. (2025); Xie et al. (2025); Xiaomi (2025) have shown benefits of staged learning in language domains, GUI automation presents unique challenges: heterogeneity in semantic complexity, grounding requirements, and interface dependencies render simple curriculum staging insufficient. This motivates our research question: *Can a difficulty-aware reinforcement learning strategy boost GUI task performance?*

Our method organizes training samples based on trajectory complexity, progressing from shorter, simpler tasks to longer, more complex ones. We incorporate fine-grained reward mechanisms providing informative feedback for tool usage, spatial precision, and semantic correctness. By integrating both operation and understanding tasks, the agent jointly develops low-level action competence and high-level task comprehension. Experimental results demonstrate significant improvements: 7.1% gain on AndroidWorld and 10.3% on our private benchmark over prior state-of-the-art approaches.

Contributions. (1) A curriculum RL strategy that systematically progresses from simple to complex tasks based on trajectory characteristics. (2) Fine-grained hybrid reward mechanisms integrating rule-based and model-predicted evaluation for stable convergence. (3) Comprehensive evaluation demonstrating strong improvements over SFT and RL baselines on both operation and understanding tasks.

2 Related Work

2.1 GUI Agent

Current GUI agent implementations follow two main paradigms. The first relies on closed-source multimodal modelsAnthropic (2024); OpenAI (2024) with prompt engineering, using visual input and system APIs for perception and decision-making. Representative works include AppAgent Zhang et al. (2025); Li et al. (2024b); Jiang et al. (2025) and Mobile-Agent series Wang et al. (2024b;a).

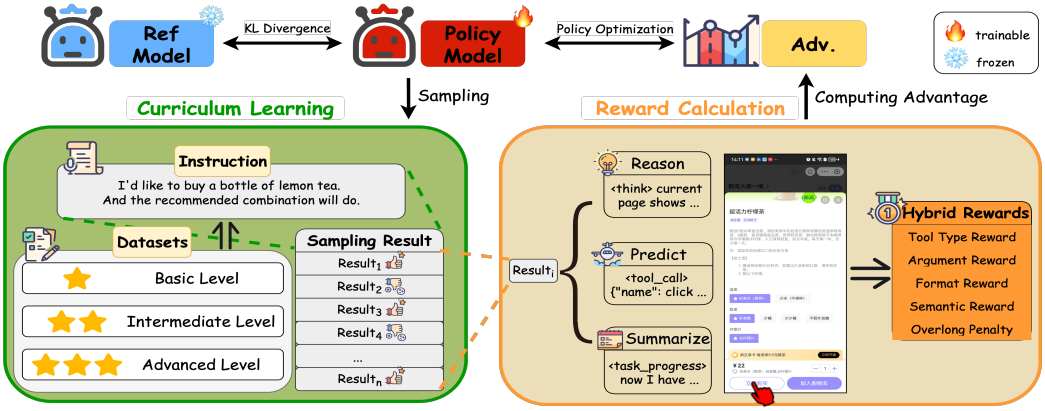

Figure 2: Overview of the Training Framework. The policy model samples for a given task prompt from specific datasets. Then updating with fine-grained reward mechanism and GRPO objective. Iteration starts from basic level tasks to advanced level tasks.

However, these approaches face key limitations: reliance on manual prompt tuning, performance bottlenecks from general-purpose models, and limited adaptability to diverse scenarios.

The second approach trains specialized models for GUI tasks, emphasizing task-specific optimization. Notable implementations include CogAgent Hong et al. (2024), AutoGLM Liu et al. (2024), Ferret-UI series You et al. (2024); Li et al. (2024c), UI-Venus Gu et al. (2025) and UI-TARS Qin et al. (2025). While these methods demonstrate improved performance on specific benchmarks, they remain constrained by dependence on large-scale pretraining data and performance degradation in out-of-distribution scenarios.

2.2 Reinforcement Learning for GUI Agents

The RL paradigm with verifiable rewards has emerged as a promising approach, inspired by reasoning models like OpenAI o1 Jaech et al. (2024) and DeepSeek-R1 Guo et al. (2025). In GUI domains where verifiable reward signals can be derived through predefined rules, RL-based training strategies are gaining increasing adoption. UI-R1 Lu et al. (2025b), GUI-G2 Tang et al. (2025) and GUI-R1 Luo et al. (2025) apply rule-based GRPO training for operation tasks; InfiGUI-R1 Liu et al. (2025a) implements SFT cold-start prior to GRPO; UIShift Gao et al. (2025) restricts GRPO training to the LLM component; ARPO Lu et al. (2025a) proposes experience replay-optimized algorithms.

However, current implementations exhibit three fundamental limitations: (1) *Data distribution bias*: none address the detrimental impact of imbalanced difficulty distributions through systematic curriculum approaches; (2) *Task scope restriction*: excessive focus on operation tasks with inadequate support for understanding tasks; (3) *Reward granularity*: overly simplistic reward formulations lacking multi-dimensional feedback. Our work addresses these gaps through curriculum-based RL with fine-grained, task-aware reward mechanisms.

3 Method

In this section, we present the key components of our method. We first revisit Group Relative Policy Optimization (GRPO) Shao et al. (2024) and describe how it is combined with curriculum learning to improve training stability and sample efficiency in complex GUI environments. We then introduce our task-specific reward design, followed by a detailed formulation of the training procedure.

3.1 Curriculum-Guided GRPO Training

We adopt Group Relative Policy Optimization (GRPO) as the foundational reinforcement learning algorithm in our framework. Unlike traditional methods such as PPOSchulman et al. (2017), GRPO

Algorithm 1: Curriculum GRPO for GUI Reasoning Model

Input: initial policy model $\pi_{\theta_{\text{init}}}$; reward functions r_φ; task prompts $\mathcal{D}$
Output: π_θ

policy model $\pi_\theta \leftarrow \pi_{\theta_{\text{init}}}$;
for *each dataset* $\mathcal{D}$ *in curriculum order* **do**
  reference model $\pi_{\text{ref}} \leftarrow \pi_\theta$;
  for *step* $= 1, \ldots, M$ **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$;
    Update the old policy model $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$;
    Sample G outputs $\{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$ for each question $q \in \mathcal{D}_b$;
    Compute rewards $\{r_i\}_{i=1}^{G}$ for each sampled output o_i by running r_φ;
    Compute $\hat{A}_i$ for o_i with ground truth gt_i through group relative advantage estimation;
    Update the policy model π_θ by maximizing the GRPO objective;
  end
end

eliminates the need for a separate value function by leveraging group-wise advantage estimation, substantially reducing memory usage and computational overhead. This makes it particularly suitable for GUI-based training under limited hardware resources.

Our GRPO-based curriculum learning framework is illustrated in Figure 2. Within this framework, the policy model undergoes gradient updates during training while the reference model remains frozen, with KL divergence regularization ensuring training stability. Training tasks are first partitioned according to operational complexity, where difficulty is defined by the number of interaction steps (i.e., trajectory length) required to complete an instruction. Tasks involving more steps are considered more challenging. This stratification yields three difficulty tiers:

1) **Stage 1: Basic level** - tasks requiring ≤ 3 interaction steps.
2) **Stage 2: Intermediate level** - tasks involving 4 to 8 steps, representing typical multi-step operational workflows.
3) **Stage 3: Advanced level** - tasks requiring > 8 steps or involving visual understanding components such as VQA, information extraction, and element localization.

Visual understanding tasks are also included in the Hard level due to their demand for fine-grained visual-semantic reasoning and contextual interpretation, which are essential capabilities for building general-purpose UI agents. Except for single-step tasks (path length = 1, typically representing low-level atomic actions), all levels conform to the high-level operation definitions established in industry standards. This progressive curriculum, advancing from basic UI manipulations to complex cross-modal cognition, facilitates human-like skill acquisition throughout training.

Each training batch follows a structured four-phase GRPO process: first, the policy model samples a group of $\{G\}$ candidate outputs $\{o_1, o_2, \ldots, o_G\}$ for a given task prompt; second, each output is evaluated using verifiable reward functions to produce scalar rewards $\{r_1, r_2, \ldots, r_G\}$; third, group-relative advantage values $\{\hat{A}_i\}$ are computed by normalizing the reward distribution within the group; finally, the policy π_θ is updated using the GRPO objective with the estimated advantages. The full training procedure is detailed in Algorithm 1.

Notably, we follow the GRPO's original implement of advantage estimation. As shown in 1, the normalized advantage $\hat{A}_i$ is computed by the mean and standard deviation of the rewards respectively. This simplified approach eliminates intricate advantage computation procedures, significantly enhancing training efficiency while maintaining theoretical soundness.

$$\hat{A}_i = \frac{r_i - \text{Mean}(\{r_1, r_2, \ldots, r_G\})}{\text{Std}(\{r_1, r_2, \ldots, r_G\})} \tag{1}$$

3.2 FINE-GRAINED VERIFIED REWARD DESIGN

To enable a more generalizable and capable UI agent, we design task-specific reward functions for both mobile operation and visual understanding tasks. Inspired by the rule-based reward paradigm introduced in DeepSeek-R1Guo et al. (2025)—known for its simplicity and effectiveness—we adopt verified reward mechanisms that combine format validation and outcome assessment. This approach has recently expanded from LLMs to VLMs, as demonstrated in works such as VLM-R1 Shen et al. (2025), and Visual-RFT Liu et al. (2025b). In our framework, most rewards are defined through verifiable rules, providing lightweight yet reliable signals to guide curriculum-based reinforcement learning.

3.2.1 MOBILE OPERATION TASKS

For mobile operation tasks, the designed reward function is shown in Equation 2. We formulate CRAFT-GUI's action prediction as tool call for mobile operations, where distinct actions correspond to specific tools (e.g., click actions activate the click tool, swipe actions trigger the swipe tool). Accordingly, the operational rewards primarily derive from two aspects: 1) Correctness reward of tool selection R_{tool}, and 2) Accuracy reward of parameter specification in tool invocation R_{args}. A format reward R_{format} is additionally incorporated to ensure structured output generation. w represents the respective weight assigned to each reward.

$$\begin{aligned} R_{operation} = w_{tool} * R_{tool} + w_{args} * R_{args} \\ + w_f * R_{format} + w_l * P_{length} \end{aligned} \tag{2}$$

Each reward component is described in detail below to illustrate its role in guiding agent behavior.

Tool Type Reward. In mobile operation tasks, we follow Qwen2.5VL Bai et al. (2025)'s agent paradigm by formulating action spaces through agent tool call: *click tool*, *swipe tool*, *type tool*, *long press tool*, *open app tool*, *system button tool*, *wait tool*, and *terminate tool*. This toolkit achieves near-comprehensive coverage of terminal operation requirements. If the predicted tool o_{tool} matches the ground truth tool gt_{tool}, it is assigned with a reward of 1, or 0 otherwise. As shown in Equation 3:

$$R_{tool} = \begin{cases} 1 & \text{if } o_{tool} \text{ matches } gt_{tool}, \\ 0 & \text{else.} \end{cases} \tag{3}$$

Tool Arguments Reward. Different tool invocations are associated with distinct parameter types:

1) For *click tool* and *long press tool*, the parameter represents a point on screen with coordinate (x, y), where x denotes pixels from the left edge and y indicates pixels from the top edge. As the ground truth is also a coordinate point, we adopt a specific verified rule. A prediction is rewarded if it falls within the same widget with the ground truth.
2) For *swipe tool*, parameters consist of two points: the start point (x_1, y_1) and end point (x_2, y_2). A prediction is considered valid if the angle between the predicted swipe vector and the ground-truth vector is within 30°.
3) The *type tool* and *open app tool* require textual parameters. In these cases, we employ semantic similarity assessment combined with character-level matching to evaluate alignment with the ground truth text.
4) *System button* tool takes one of several enumerated commands (e.g., Back, Home, Menu, Enter). Validation is performed via strict string matching.
5) *Wait* and *terminate* tools require no arguments; thus, no reward is computed for these cases.

All these rewards can be formulated as 4:

$$R_{paras} = \begin{cases} 1 & \text{if } o_{paras} \text{ matches } gt_{paras}, \\ 0 & \text{else.} \end{cases} \tag{4}$$

Format reward. We design task-specific response templates to guide the model in producing structured reasoning processes prior to final outputs, following paradigms such as ReAct and chain-of-thought. To this end, we introduce thinking tokens, which enhance the model's logical reasoning

capabilities and improve both stability and performance on complex tasks. All responses—across both mobile operation and visual understanding tasks—are formatted using HTML-style tags for consistency and interoperability.

For operation tasks, the response structure comprises three key components:

1) The $< think >$ tag explicitly documents the model's reasoning trajectory, enhancing output interpretability through transparent cognitive process visualization.
2) The $< tool\_call >$ tag standardizes action execution with JSON-formatted tool invocations, ensuring structured parameter specification and operational consistency.
3) The $< task\_progress >$ tag facilitates task state summarization and historical context preservation, functioning as an essential memory mechanism for multi-turn interaction.

For a more detailed description of the format rewards for both operation and visual understanding tasks, please refer to the supplementary materials. We formulate the format reward signal in Equation 5 as follows:

$$R_{format} = \begin{cases} 1 & \text{if } o_i \text{ matches response pattern,} \\ 0 & \text{else.} \end{cases} \tag{5}$$

Overlong Response Penalty During GRPO training we observe a progressive increase in response length, primarily driven by expanding thinking token sequences. If left unregulated, this phenomenon risks triggering explosive thinking token growth that may ultimately lead to model performance collapse. Inspired by DAPO Yu et al. (2025)'s optimization principles, we implement an adaptive length constraint mechanism. This approach applies graduated penalties to responses exceeding predetermined length thresholds, effectively suppressing overgeneration tendencies while preserving essential reasoning components. As shown in Equation 6, $L_{\max}$ is the max output length, and L_{cache} is the buffer length for soft penalty.

$$P_{\text{length}} = \begin{cases} 0, & |o_i| \le L_{\max} - L_{\text{cache}} \\ \frac{(L_{\max} - L_{\text{cache}}) - |o_i|}{L_{\text{cache}}}, & L_{\max} - L_{\text{cache}} < |o_i| \le L_{\max} \\ -1, & L_{\max} < |o_i| \end{cases} \tag{6}$$

3.2.2 Visual Understanding Tasks

For visual understanding tasks covering visual question answering, information extraction, and element localization, we design a dedicated reward function as shown in Equation 7. These tasks enhance the agent's ability to perceive and semantically interpret screen content. The semantic reward term R_{sem} evaluates the alignment between model responses and annotated ground truths. Due to the complexity of natural language, R_{sem} is based on a LLM-as-judged approach. A detailed implementation is provided in the supplementary material.

$$R_{understanding} = w_s * R_{sem} + w_f * R_{format} + w_l * P_{length}, \tag{7}$$

Specifically, we employ a format reward R_{format} to enforce consistent output structures and incorporate an adaptive length penalty P_{length} that progressively penalizes responses exceeding predefined length limits. Each reward component is weighted by its respective coefficient w.

4 Experiments

4.1 Implement Details

Training and inference. During our multi-stage curriculum reinforcement learning experiments, we evaluated performance scaling across varying model sizes by employing Qwen2.5VL with 7B, 32B parameters as base models. All experiments are conducted using computational resources equivalent to 96 NVIDIA A100 GPUs (80GB). In terms of the consistency of the evaluation protocol, we maintain identical mobile operation and visual understanding prompts between the training and inference phases. Model capabilities are assessed through: 1) zero-shot generalization on unseen task distributions; 2) pass@1 success rate under strict single-attempt execution constraints

Evaluation benchmarks and metrics. We established an online evaluation benchmark to comprehensively assess model capabilities in real-world operational environments, focusing on three critical dimensions: page comprehension accuracy, contextual reasoning effectiveness, and integrated decision-making proficiency. To validate the methodological efficacy proposed in this work, parallel evaluations were conducted on open-source benchmark suites including Android ControlLi et al. (2024a) and AndroidWorldRawles et al. (2024). The primary evaluation metric employed is task success rate (SR), calculated as Equation 8:

$$SR = \frac{N_{\text{success}}}{N_{\text{total}}} \times 100\% \tag{8}$$

4.2 Experimental Results

To validate the effectiveness of our proposed methodology – comprising multi-stage curriculum reinforcement learning and joint training of operation-understanding tasks – we conducted systematic evaluations across both in-house and open-source datasets.

Table 1: Comparison of Agent Models on AndroidControl-Low, AndroidControl-High and AndroidWorld; * Claude refers Claude-computer-use.

Agent Models	AndroidControl-Low			AndroidControl-High			AndroidWorld
	Type	Grounding	SR	Type	Grounding	SR	SR
Claude*	74.3%	0.0%	19.4%	63.7%	0.0%	12.5%	27.9%
GPT-4o	74.3%	0.0%	19.4%	66.3%	0.0%	20.8%	34.5%
SeeClick	93.0%	73.4%	75.0%	82.9%	62.9%	59.1%	–
InternVL-2-4B	90.9%	84.1%	80.1%	84.1%	72.7%	66.7%	–
Qwen2-VL-7B	91.9%	86.5%	82.6%	83.8%	77.7%	69.7%	–
Aria-UI	–	87.7%	67.3%	–	43.2%	10.2%	–
OS-Atlas-7B	93.6%	88.0%	85.2%	85.2%	78.5%	71.2%	–
Aguvis-72B	–	–	84.4%	–	–	66.4%	26.1%
UI-TARS-72B	98.1%	89.9%	91.3%	85.2%	81.5%	74.7%	46.6%
Qwen2.5-VL-32B	–	–	93.3%	–	–	69.6%	22.0%
Qwen2.5-VL-72B	–	–	93.7%	–	–	67.4%	35.0%
CRAFT-GUI-32B-stage1	98.0%	92.4%	91.3%	88.7%	81.9%	75.6%	41.4%
CRAFT-GUI-32B-stage2	98.8%	93.1%	92.2%	90.8%	84.4%	78.7%	44.8%
CRAFT-GUI-32B-stage3	**98.9%**	**93.6%**	92.7%	**91.0%**	**85.8%**	**80.3%**	**51.7%**

4.2.1 Open-source Datasets Training and Evaluation

To validate the feasibility of our proposed methodology, we conducted training on public datasets and performed corresponding evaluations on designated benchmarks. Specifically, we selected the offline evaluation platforms - Android Control - along with the online evaluation platform AndroidWorld for comprehensive testing.

For the Android Control training set, we implemented curriculum reinforcement learning training using the Qwen2.5-VL-32B as our base model. The training configuration strictly followed our proposed methodology. Task trajectories are stratified into basic, intermediate, and advanced difficulty levels based on task path complexity metrics. And the training progress is executed with our hybrid reward mechanism. During evaluation, we employ the complete Android Control test set as our benchmark, maintaining strict separation between training and evaluation data splits. For Android Control, we report two settings (low and high). Meanwhile we use the same weights to evaluate on AndroidWorld. As shown in Table 1, the performance of CRAFT-GUI-32B demonstrates the effectiveness of our proposed method.

4.2.2 In-house Datasets Training and Evaluation

We developed a proprietary training dataset comprising 80K samples across six major categories of mobile applications: food delivery, in-store dining, medical services, financial services, insurance services, and gaming apps. All data are collected from multiple device platforms (iPhone, Android,

iPad) to ensure cross-platform generalization. This multi-domain corpus is organized into three difficulty levels to support phased curriculum learning: Basic level (21K operation samples, $\leq$3 steps), Intermediate level (28K operation samples, 4-8 steps), and Advanced level (8K operation samples and 23K understanding samples, $>$8 steps or involving visual reasoning tasks).

Our method significantly outperforms industry baselines including Claude-3.7-Sonnet Anthropic (2024) and GPT-4.1 OpenAI (2025) across all application domains. CRAFT-GUI-32B achieves 75.7% average success rate, representing 10.3% improvement over the best baseline (detailed results in Appendix Table 4). Progressive curriculum learning yields consistent gains: our 32B model improves from 62.6% (Stage 1) to 75.7% (Stage 3), while the 7B variant progresses from 43.9% to 64.1% (see Appendix Table 5 for stage-wise breakdown). Additional understanding task results are provided in Appendix Table 7.

4.3 Ablation Study

Table 2: SR of Different Training Method

Training Method	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
SFT	52.9%	48.6%	62.7%	58.8%	48.5%	91.5%	60.8%
Vanilla GRPO	60.8%	69.2%	70.6%	75.5%	**62.4%**	92.6%	71.9%
Curriculum GRPO	**76.5%**	**73.8%**	**73.5%**	**77.5%**	60.4%	**92.7%**	**75.7%**

Table 3: SR of Different Data Mixture

Training Data	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
Operation Only	67.6%	72.0%	73.3%	74.3%	59.4%	91.5%	73.2%
Operation + Understanding	**76.5%**	**73.8%**	**73.5%**	**77.5%**	**60.4%**	**92.7%**	**75.7%**

We conduct ablation experiments on the 32B-scale model to systematically demonstrate the effects of different ablated components. The ablated models are trained on our proprietary internal dataset and thoroughly evaluated on the online benchmark comprising six categories of mobile applications. The evaluation results are presented in tabular format for clear visualization. The ablation study outcomes conclusively validate the effectiveness of our proposed methodology.

The Effectiveness of Curricumlum RL Strategy In the experiments of training strategies, we utilized identical raw datasets. For the three strategy, each sample's system prompt, user prompt, and assistant prompt remain strictly consistent across training configurations. Moreover, the training datasets all incorporate operation-type and understanding-type data. As shown in the Table 2, the curriculum reinforcement learning algorithm achieves the highest average operation success rate of 75.7% across six categories of mobile applications. The traditional reinforcement learning implementation demonstrates an 11.1% improvement over the supervised fine-tuning (SFT) baseline, while the curriculum reinforcement learning approach yield a 14.9% enhancement compared to SFT. Furthermore, applying curriculum learning strategies to the reinforcement learning framework provides an additional 3.8% performance gain over the standard reinforcement learning implementation.

The Effectiveness of Training Data Mixture Based on the curriculum reinforcement learning training strategy, we investigate the necessity of mixing operation tasks and understanding tasks. As shown in Table 3, we conduct two sets of experiments under identical curriculum reward training strategy settings: one group containing only operation-type data, while the other incorporates complex and challenging understanding-type data in the third phase. The results demonstrate that with the inclusion of understanding-type data, the model exhibits enhanced comprehension and mastery of operation-type tasks, achieving a 2.5% improvement in execution success rate compared to using single-type data.

5 CONCLUSION

In this work, we present CRAFT-GUI, a GUI reasoning model developed through a novel multi-stage reinforcement learning training framework. Owing to the joint training of operation tasks and understanding tasks, CRAFT-GUI thereby advances towards a more versatile GUI-Agent system that autonomously reasons and interacts with mobile devices to complete user instructions. The carefully engineered reward and penalty mechanisms ensure stable model training while maintaining robust performance metrics in evaluations. Additional training details and case studies can be found in the Appendix section. In future work, we plan to extend CRAFT-GUI to computer tasks and introduce trial-and-error with rollback, enabling more general-purpose intelligent agents.

REFERENCES

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

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

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

Longxi Gao, Li Zhang, and Mengwei Xu. Uishift: Enhancing vlm-based gui agents through self-supervised reinforcement learning. *arXiv preprint arXiv:2505.12493*, 2025.

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

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

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

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. Appagentx: Evolving gui agents as proficient smartphone users. *arXiv preprint arXiv:2503.02268*, 2025.

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

Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*, 2024b.

Zhangheng Li, Keen You, Haotian Zhang, Di Feng, Harsh Agrawal, Xiujun Li, Mohana Prasad Sathya Moorthy, Jeff Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*, 2024c.

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

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

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

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025b.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025a.

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

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

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

OpenAI. Introducing gpt-4o. Available at: https://openai.com/index/hello-gpt-4o, 2024.

OpenAI. Introducing gpt-4.1. Available at: https://openai.com/index/gpt-4-1/, 2025.

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

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

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710, 2024a.

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

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*, 2025.

LLM-Core-Team Xiaomi. Mimo-vl technical report, 2025. URL https://arxiv.org/abs/2506.03569.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Keen You, Haotian Zhang, Eldon Schoop, Floris Weers, Amanda Swearngin, Jeffrey Nichols, Yinfei Yang, and Zhe Gan. Ferret-ui: Grounded mobile ui understanding with multimodal llms. In *European Conference on Computer Vision*, pp. 240–255. Springer, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–20, 2025.

A Additional Training details

Understanding tasks adopt a dual-phase structure, as shown in Figure4:

1) The $< think >$ tag enables multi-step semantic reasoning, capturing hierarchical understanding of visual-semantic content.
2) The $< answer >$ tag delivers final responses with precise information encapsulation, strictly following predefined knowledge representation formats.

The Mobile Operation Template

<think>
reasoning process here.
</think>
<tool_call>
{"name": <tool-name>, "arguments": <args-json>}
</tool_call>
<task_progress>
progress summary here.
</task_progress>

Figure 3: The Mobile Operation Template.

The Visual Understanding Template

<think>
reasoning process here.
</think>
<answer>
final answer here.
</answer>

Figure 4: The Visual Understanding Template.

Figure 5 illustrates the workflow used to construct and run an evaluation episode. For every task, we (i) reconstruct the corresponding UI flow on an emulator, (ii) tag critical intent checkpoints such as item selection, quantity confirmation, or payment submission, and (iii) introduce stochastic branches (e.g., pop-ups, alternative menus) together with latency or layout noise. During execution the evaluator records success at each checkpoint, enabling fine-grained and realistic performance measurement.

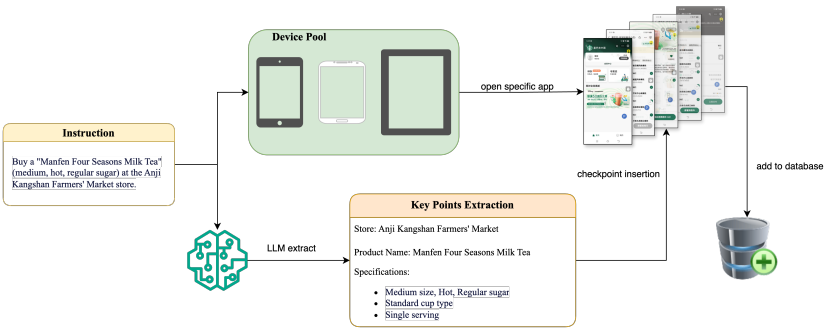

Figure 5: Illustration of Our Online Benchmark building pipeline.

Semantic Consistency Reward In operation tasks involving textual parameters and visual understanding tasks, semantic consistency reward is typically involved. This reward quantifies the textual similarity between the model's output o_{sem} and the ground truth gt_{sem}, where higher similarity corresponds to greater reward values. Traditional lexical-based metrics (e.g., TF-IDF, Rouge-L) prove insufficient to capture nuanced semantic equivalence due to linguistic diversity, particularly in handling synonymy, paraphrasing, and contextual awareness.

To address this limitation, we remotely deploy small language reward model for semantic evaluation. This model implements a comprehensive multi-dimensional assessment framework, considering:

1) Entity consistency - verification of critical information alignment
2) Content alignment - semantic equivalence of descriptive elements
3) Contextual emphasis - preservation of key focus points
4) Paraphrase robustness - tolerance to syntactical variations

This design effectively bridges the gap between rigid pattern matching and flexible semantic understanding in reward assignment. The scoring process can be formulated as Equation 9, where $\mathcal{S}$ means the semantic similarity between output o_{sem} and the ground truth gt_{sem}.

$$R_{sem} \in \{0,\ 0.2,\ 0.4,\ \ldots,\ 1\} \quad \text{depends on } \mathcal{S}(o_{sem}, gt_{sem}) \tag{9}$$

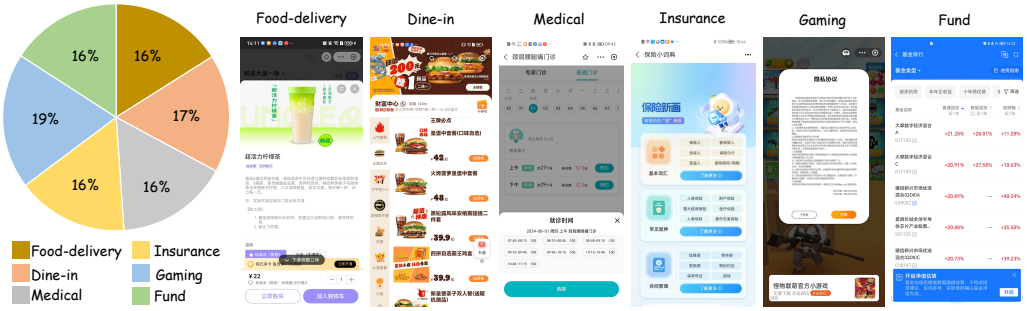

Figure 6: Domain-specific Distribution of Our Online Benchmark.

Table 4: SR (Success Rate, Pass@1) on Internal Online Operation Benchmark

Model	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
Claude-3.7-Sonnet	31.7%	28.3%	44.4%	24.2%	25.6%	61.7%	35.9%
GPT-4.1	19.5%	18.8%	32.4%	21.3%	18.0%	56.2%	27.7%
UI-TARS-72B	65.4%	57.2%	67.5%	61.8%	55.1%	85.3%	65.4%
Qwen2.5-VL-72B	65.7%	52.3%	67.6%	57.4%	56.9%	88.6%	64.8%
CRAFT-GUI-7B	43.9%	48.4%	74.5%	67.4%	54.2%	96.4%	64.1%
CRAFT-GUI-32B	**76.5%**	**73.8%**	**73.5%**	**77.5%**	**60.4%**	92.7%	**75.7%**

B Online Benchmark for Operation Task

Interactive applications deployed in the real world—such as food delivery, medical appointment scheduling, navigation, and ticketing—exhibit highly variable user flows. To be useful, an agent must adapt to individual habits and dynamically changing requirements.

We constructed an online benchmark encompassing six categories of daily-use mobile applications: food delivery, in-store dining, medical services, financial services, insurance services, and gaming applications. Each category incorporates a diverse selection of mobile apps reflecting real-world operational complexity. The benchmark composition follows domain-specific distribution as Figure 6:

The benchmark's tasks are statistically derived from real-human interaction logs, preserving authentic operational frequency across application domains. Each category contains 120-200 unique test cases with configurable difficulty levels, supporting both functional verification and cognitive capability assessment.

C More Samples

CRAFT-GUI-7B Table 5 presents the evaluation outcomes of the 7B model across distinct curriculum reinforcement learning stages on the online benchmark. The results demonstrate progressive success rate enhancements across all mobile application categories as training progresses through sequential phases.

CRAFT-GUI-32B Table 6 details the evaluation results of the 32B model under identical training framework configurations on the online benchmark. The expanded parameter space of the base model yielded substantial performance gains, with the 32B variant achieving 11.6% higher average success rate compared to the 7B model during final curriculum phase evaluation.

Progressive performance escalation is observed throughout the three-stage curriculum learning process: stage 1 (62.6% SR), stage 2 (69.9% SR), and final stage (75.7% SR), demonstrating large relative improvement from initial to final training completion.

Table 5: CRAFT-GUI-7B Performance (SR) Across Mobile Application Categories by RL Period

RL Period	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
Stage 1	24.3%	17.2%	38.4%	50.2%	40.8%	92.7%	43.9%
Stage 2	26.3%	29.2%	58.5%	57.8%	47.8%	95.0%	52.4%
Stage 3	**43.9%**	**48.4%**	**74.5%**	**67.4%**	**54.2%**	**96.4%**	**64.1%**

Table 6: CRAFT-GUI-32B Performance (SR) Across Mobile Application Categories by RL Period

RL Period	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
Stage 1	51.0%	55.1%	63.7%	61.8%	51.5%	92.7%	62.6%
Stage 2	59.8%	72.9%	66.7%	68.6%	58.4%	**93.5%**	69.9%
Stage 3	**76.5%**	**73.8%**	**73.5%**	**77.5%**	**60.4%**	92.7%	**75.7%**

Understanding metrics For visual understanding scenarios such as visual question answering, information extraction and element localization, we construct respective benchmarks covering six kinds of mobile applications (Food delivery, et al.). The evaluation metric employed is accuracy (Acc), calculated as Equation 10:

$$Acc = \frac{N_{\text{correct}}}{N_{\text{total}}} \times 100\% \tag{10}$$

Table 7: Acc on Visual Understanding Benchmark

Model	Average
	VQA & Info Extraction & Localization
Claude-3.7-Sonnet	90.4%
GPT-4.1	91.8%
CRAFT-GUI-7B	80.5%
CRAFT-GUI-32B	**94.0%**

The Effectiveness of Overlong Response Penalty Since the training process incorporates understanding-related tasks, the think tokens exhibit uncontrollable growth during training progression. To address this, we conduct ablation experiments to investigate the impact of applying an overlong response penalty on model performance. In this experimental group, while maintaining identical configurations of training data, three-phase curriculum reinforcement learning strategy, and multi-task data mixing protocol with the baseline, we solely introduce the length penalty constraint. The ablation study reveals that without length penalty regulation, the model progressively generates excessive nonsensical phrases and enters infinite reasoning loops until exceeding the maximum sequence length threshold, ultimately leading to catastrophic performance collapse. In contrast, with the application of response length penalty, the training process remains stable and achieves the anticipated performance benchmarks. Details on the online benchmark can be found in Table 8.

Table 8: SR of Different Penalty Strategy

Training Data	Mobile Application Categories						Average
	Food-delivery	Dine-in	Medical	Finance	Insurance	Gaming	
w/o Overlong Penalty	59.6%	61.9%	64.5%	64.3%	48.4%	82.6%	63.5%
w/ Overlong Penalty	**76.5%**	**73.8%**	**73.5%**	**77.5%**	**60.4%**	**92.7%**	**75.7%**

As shown in the Table 9, the curriculum RL also achieves the highest average accuracy of 94.0% on our visual understanding benchmark across three types of task. The vanilla RL method get 9.2%

Table 9: Acc of Different Training Method

Model	Average
	VQA & Info Extraction & Localization
SFT	82.2%
Vanilla GRPO	91.4%
Curriculum GRPO	**94.0%**

improvement over the SFT baseline. And curriculum RL get another 2.6% enhancement over the standard RL.