GUI-G1: Understanding R1-Zero-Like Training for Visual Grounding in GUI Agents

Yuqi Zhou[1], Sunhao Dai[1], Shuai Wang[2*], Kaiwen Zhou[2], Qinglin Jia[2], Jun Xu[1*]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Huawei Noah's Ark Lab
{yuqizhou,sunhaodai}@ruc.edu.cn

Abstract

Recent Graphical User Interface (GUI) agents replicate the R1-Zero paradigm, coupling online Reinforcement Learning (RL) with explicit chain-of-thought reasoning prior to object grounding and thereby achieving substantial performance gains. In this paper, we first conduct extensive analysis experiments of three key components of that training pipeline: input design, output evaluation, and policy update—each revealing distinct challenges arising from blindly applying general-purpose RL without adapting to GUI grounding tasks. Input design: Current templates encourage the model to generate chain-of-thought reasoning, but longer chains unexpectedly lead to worse grounding performance. Output evaluation: Reward functions based on hit signals or box area allow models to exploit box size, leading to reward hacking and poor localization quality. Policy update: Online RL tends to overfit easy examples due to biases in length and sample difficulty, leading to under-optimization on harder cases. To address these issues, we propose three targeted solutions. First, we adopt a **Fast Thinking Template** that encourages direct answer generation, reducing excessive reasoning during training. Second, we incorporate a box size constraint into the reward function to mitigate reward hacking. Third, we revise the RL objective by adjusting length normalization and adding a difficulty-aware scaling factor, enabling better optimization on hard samples. Our **GUI-G1-3B**, trained on 17K public samples with Qwen2.5-VL-3B-Instruct, achieves **90.3%** accuracy on ScreenSpot and **37.1%** on ScreenSpot-Pro. This surpasses all prior models of similar size and even outperforms the larger UI-TARS-7B, establishing a new state-of-the-art in GUI agent grounding. The project repository is available at https://github.com/Yuqi-Zhou/GUI-G1.

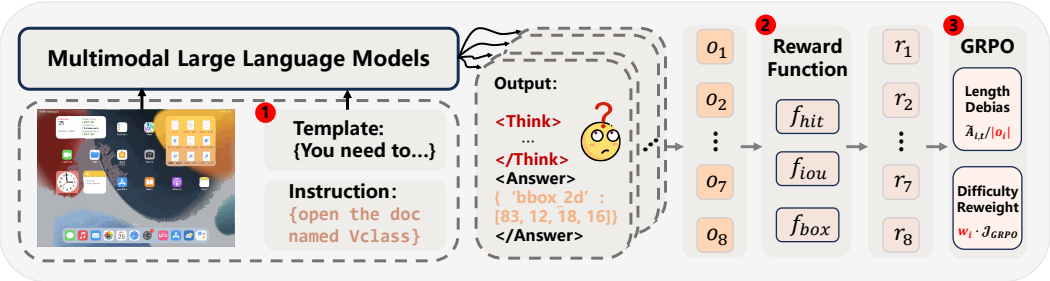

Figure 1: This framework employs the GRPO algorithm for optimization, emphasizing three critical components: input design, output evaluation, and policy update.

*Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

1 Introduction

DeepSeek-R1-Zero [12] revolutionizes the post-training pipeline of Large Language Models (LLMs) by introducing the R1-Zero paradigm, which applies RL directly to base LLMs without relying on supervised fine-tuning (SFT) as an intermediate step. Motivated by this approach, recent work in the domain of GUI agents [25, 28, 40] has increasingly adopted RL, particularly the GRPO algorithm [34], in order to address two key limitations: (1) SFT requires large-scale, high-quality labeled datasets, resulting in significant computational costs; (2) existing open-source GUI agents trained with SFT often exhibit poor generalization to out-of-domain (OOD) scenarios [5, 27].

While RL has emerged as a popular choice for training GUI agents in recent work, attributing performance gains solely to the algorithm itself remains nontrivial. These R1-style models often differ in multiple dimensions—including backbone architectures, data sources, and training protocols—making it difficult to isolate the specific contribution of online RL. To better isolate the role of RL, we focus exclusively on the grounding task [11, 23], which we consider the core capability for effective GUI interaction [24]. Building on these observations, this work rethinks the role of RL in R1-style GUI agents training by (1) disentangling its algorithmic contributions from other system-level factors, and (2) focusing exclusively on grounding as the reinforcement objective.

For this, we decompose the R1-Zero-like training pipeline into three core components: input design, output evaluation, and policy update. Each reveals distinct challenges arising from blindly applying general-purpose RL without adapting to grounding tasks. **First**, we observe that the grounding performance of the state-of-the-art R1-style model, InfiGUI-R1 [25], drops as reasoning increases in Sec. 3.1, suggesting that reasoning templates may not benefit grounding in GUI agents. **Second**, we find that commonly used reward functions based on hit signals or box area lead to opposite forms of reward hacking in Sec. 3.2: the former encourages smaller boxes with higher accuracy, while the latter favors larger boxes to increase Intersection over Union (IoU). **Finally**, we identify two biases in the GRPO objective: length bias [26] and difficulty bias in Sec. 3.3. Length bias encourages longer but incorrect responses, which, as previously observed, further degrade grounding performance. Difficulty bias treats all samples equally, hindering the model's ability to learn from more challenging examples. Together, these biases make it harder for the model to learn from difficult samples.

To address the above issues, we implement the following improvements. First, we introduce the **Fast Thinking Template**, which encourages the policy to generate answers directly during training. Second, to counteract the hacking in common reward functions that prefer boxes of different sizes during policy optimization, we propose a box-size-based reward function as a constraint. Finally, we remove the length normalization term from the original GRPO objective, following [26]. We also introduce a difficulty coefficient for each sample's loss, allowing the model to receive greater gradients for more challenging samples. The difficulty coefficient is calculated from the relative box size, which serves as a proxy difficulty indicator in the grounding task [18].

Building on the above solutions, we train our model, **GUI-G1-3B**, using Qwen2.5-VL-3B-Instruct and a small (about 17K) set of grounding samples, showing strong performance with limited supervision from public datasets such as UI-BERT [3] and OS-Atlas [39]. Our model achieves new state-of-the-art performance on GUI grounding benchmarks, with **90.3%** accuracy on ScreenSpot [8] and **37.1%** on ScreenSpot-Pro [18]. It surpasses the previous best R1-style GUI agent, InfiGUI-R1 [25], while requiring significantly less data, fewer output tokens, and fewer training stages.

In summary, the contributions of this paper are as follows: (1) We identify three distinct challenges in the R1-Zero-like training pipeline of R1-style GUI agents: grounding is harmed by longer reasoning due to grounding's reliance on image tokens; common reward functions induce size-sensitive reward hacking; and GRPO biases agents toward simpler examples due to its objective. (2) We further analyze and propose three solutions: a **Fast Thinking Template** for policy training, a box size–based reward to regularize box size, and a modified GRPO with difficulty weighting and no length normalization. (3) Trained on only 17K fully open-source grounding samples, our **GUI-G1-3B** achieves state-of-the-art performance while using fewer tokens when testing.

2 R1-Zero-Like Training Paradigm for GUI Grounding

We begin by describing how to train Multimodal Large Language Models (MLLMs) for grounding tasks. Given a screenshot s and a textual description d, the MLLM is trained to predict the target

location B, typically represented as a bounding box or a point. Following prior work [8], we formulate grounding as a language generation task, where the MLLM produces a response o that includes the predicted location as well as additional components such as the reasoning process or objective descriptions in Figure 1. In our implementation, the predicted location is expressed as a bounding box $B_{\text{pred}} = (\hat{x}_1, \hat{y}_1, \hat{x}_2, \hat{y}_2)$, where x and y denote the horizontal and vertical coordinates, respectively. This prediction is evaluated against the ground-truth box $B_{\text{gt}} = (x_1, y_1, x_2, y_2)$.

When RL is applied via the algorithm like GRPO [34], a template is first used to guide the response format, and the model generates N candidate responses $O = \{o_1, o_2, \ldots, o_N\}$. Each response is then evaluated using a set of rule-based reward functions, yielding a reward set $\{r_1, r_2, \ldots, r_N\}$. The relative advantage A_i of each response is computed as:

$$A_i = \frac{r_i - \texttt{mean}(r_1, r_2, \ldots, r_N)}{\text{std}(r_1, r_2, \ldots, r_N)}, \tag{1}$$

where `mean` and `std` denote the mean and standard deviation of the rewards, respectively. Finally, the policy model is optimized using the GRPO objective with KL-divergence regularization.

3 How R1-Zero-Like Training Affects Grounding for GUI Agents?

We first aim to understand R1-Zero-like training paradigm for grounding task in GUI agents by examining three essential components: the input design (template) (Sec. 3.1), the output evaluation (reward function) (Sec. 3.2), and policy update (RL objective) (Sec. 3.3). Finally, we present our model, **GUI-G1**, in Sec. 3.4, where we also summarize and compare our approach with existing R1-style agents to demonstrate its advantages in grounding tasks.

3.1 Analysis on Template

Recent R1-style GUI agents have increasingly incorporated explicit reasoning by prompting the model to "think before action" [25, 28, 40], as illustrated in Figure 1. For example, InfiGUI-R1 [25] uses a **Slow Thinking Template**. **While such reasoning-augmented agents achieve strong performance, it remains unclear whether the gains truly arise from the reasoning process itself.** In fact, we find that reasoning is often unnecessary for the grounding task in GUI agents. Before conducting the analysis, we formalize the model's input and output to ensure consistency across experiments. The input consists of an image segment s and an instruction component $\mathbf{t}_{\text{ins}}$, which includes the user query and a guiding template. The output o comprises the reasoning trace $\mathbf{t}_{\text{think}}$ and the final response $\mathbf{t}_{\text{ans}}$, where $\mathbf{t}_{\text{ans}}$ contains the predicted location B_{pred}. We define the number of reasoning tokens n_{think} as **output tokens**, and the tokens derived from the image n_{img} as **image tokens**. The text ratio is given by $\frac{n_{\text{ins}}+n_{\text{think}}}{n_{\text{img}}+n_{\text{ins}}+n_{\text{think}}}$, where n_{ins} is the instruction tokens number.

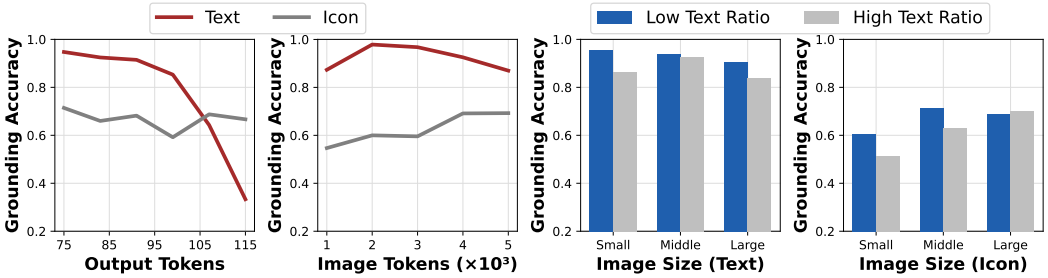

Figure 2: **(Left)** shows the grounding accuracy under varying numbers of output tokens and image tokens. "Text" refers to cases where the target is a textual element, while "Icon" refers to image targets. **(Right)** presents the grounding accuracy on the Text and Icon subsets across different image sizes. Within each group, samples are evenly divided based on their text ratio.

Longer thinking leads to worse grounding performance. While prior work, such as OpenAI-o1 [30] and DeepSeek-R1 [12] demonstrates that longer reasoning chains can enhance performance on System-2 tasks like mathematics and programming, more recent studies [19, 44] have found that introducing intermediate reasoning steps may impair performance in tasks such as image classification and GUI grounding. Building on these observations, we further find that longer reasoning chains

consistently degrade grounding accuracy on the ScreenSpot dataset [8], as shown in Figure 2(Left). This suggests that longer chains are not only unnecessary but can be increasingly detrimental in GUI grounding, especially when the target item to be grounded is text.

Grounding benefits from appropriately scaled image tokens rather than from scaled text thinking. In Figure 2 (Left Middle), we observe that the grounding performance of InfiGUI-R1-3B improves as the number of image tokens increases. This observation raises a central question: **Is grounding more reliant on image tokens or text tokens?** To investigate this, we first partition the test samples based on the number of image tokens, ensuring each subset has a comparable level of visual input. Within each subset, we further divide samples into two categories according to their text ratio and evaluate grounding accuracy for each. As shown in Figure 2 (Right), a higher text ratio consistently correlates with lower grounding performance, indicating that enriching visual content is more effective than injecting additional textual reasoning.

As shown in Figure 2, explicit reasoning does not always improve grounding accuracy. Inspired by *Thinking, Fast and Slow* [15], we propose a Fast Thinking Template that guides the model to generate grounding outputs directly without reasoning steps. As detailed in Appendix B, models trained with this template outperform baselines on ScreenSpot and ScreenSpot-Pro.

Template 1 (*Fast Thinking Template*) *You are a helpful assistant. \nUser: Grounding instruction is: {question} Please help to locate and output the bbox coordinates in JSON format.\nAssistant:*

Template 2 (*Slow Thinking Template*) *You FIRST think about the reasoning process as an internal monologue and then provide the final answer.\nThe reasoning process MUST BE enclosed within <think> </think> tags. \n User: The screen's resolution is {width}x{height}. \nPoint to the UI element most relevant to "{question}", output its coordinates using JSON format:\n "'json\n[\n{{"point_2d": [x, y], "label": "object name/description"}}\n]"' \nAssistant:*

3.2 Analysis on Reward Function

The rule-based reward function introduced in DeepSeek-R1 [12] exemplifies a simple yet effective approach based on exact match. In grounding tasks, current reward functions for R1-style GUI agents are mainly categorized into **Hit-based rewards** [25, 28, 40] and **IoU-based rewards** [25] in Table 1. Here, (x_p, y_p) is the center of the predicted box, computed as $x_p = (\hat{x}_1 + \hat{x}_2)/2$, $y_p = (\hat{y}_1 + \hat{y}_2)/2$. The Hit-based reward checks whether predicted box center hits within B_{gt}, while the IoU-based reward measures the IoU between B_{pred} and B_{gt}. While prior work has employed R_{Hit} and R_{IoU} as reward signals for grounding-based RL, it remains unclear how these objectives jointly influence training dynamics. To answer this, we implement both types of reward functions for a comparative analysis. The detailed experimental settings and evaluation metrics can be found in Appendix C. Unless otherwise specified, all subsequent analyses follow the same setup.

Table 1: Comparison of rule-based reward functions and their effects on training dynamics. "–" indicates failure to optimize (e.g., R_{Box} alone).

Reward	Formula	Driven By	Box Size(↑↓)	Accuracy(↑↓)	IoU(↑↓)
R_{Hit}	$\mathbf{1}((x_p, y_p) \in B_{\text{gt}})$	Point Accuracy	↓	↑	↓
R_{IoU}	$\text{IoU}(B_{\text{pred}}, B_{\text{gt}})$	IoU	↑	↓	↑
R_{Box}	$R_{\text{Box}} = \frac{4}{x_{p_1}+x_{p_2}+y_{p_1}+y_{p_2}}$	Box Size	-	-	-

Individually optimizing R_{Hit} and R_{IoU} leads to conflicting reward hacking behaviors. As shown in Figure 3 (Left), optimizing R_{Hit} improves accuracy but causes IoU to drop in later training. Conversely, optimizing R_{IoU} enhances overlap quality but reduces accuracy. This illustrates reward hacking in GUI grounding, where models overfit to one objective at the cost of others. These metrics capture complementary yet competing aspects: R_{Hit} focuses on correctly identifying the target box, while R_{IoU} measures overlap with ground truth. Their conflict when optimized separately highlights the challenge of designing balanced rewards.

GRPO's sample selection bias toward different box sizes leads to reward hacking. To investigate the cause of reward hacking, we visualize two cases with predicted bounding boxes in Figure 4 (Left). Models trained with R_{Hit} tend to produce boxes smaller than the ground truth, while R_{IoU} leads to

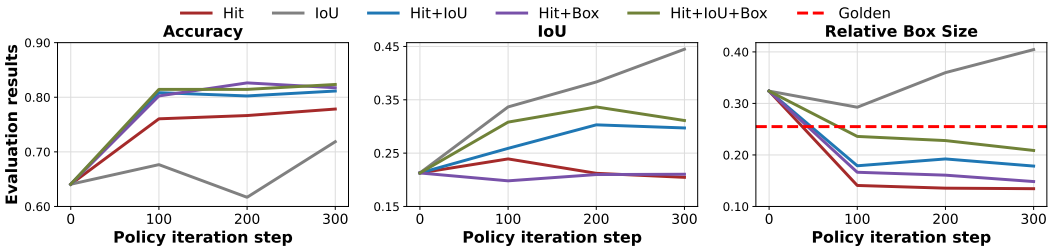

Figure 3: Changes in accuracy (Left), IoU (Middle), and relative box size (Right) across policy iterations during model training on the ScreenSpot dataset.

significantly larger boxes. This pattern is quantitatively confirmed in Figure 3 (Right), where the relative size of predicted boxes increases over training under R_{IoU}, but decreases under R_{Hit}. Further, as illustrated in Figure 4 (Right), the cause of these opposite trends lies in how GRPO's sample selection interacts with the reward functions: optimizing R_{Hit} encourages the model to pick smaller boxes that better capture the core target region, improving accuracy, whereas optimizing R_{IoU} favors larger boxes that yield higher overlap with ground truth, thus boosting IoU.

R_{Box} helps mitigate reward hacking by regularizing box size. To address reward hacking, a straightforward solution is to jointly optimize both R_{Hit} and R_{IoU}. However, as shown in Figure 3, training may still be dominated by one of the two, resulting in suboptimal balance. To alleviate this, we introduce a new reward function R_{Box} in Table 1. Here, $x_{p_1} = \frac{1}{1-|\hat{x}_1 - x_1|/\text{image width}}$, with similar definitions for the other terms. This reward encourages the predicted bounding box to match the ground truth in terms of size. As shown in Figure 3, incorporating R_{Box} leads to further improvements in both accuracy and IoU, with predicted box sizes becoming more aligned with the ground truth. We also experiment with using R_{Box} alone, but the model fails to produce outputs in the correct format. We hypothesize this is because R_{Box} assigns non-zero rewards even to poorly grounded predictions, encouraging optimization on uninformative samples. Therefore, R_{Box} should be used in conjunction with R_{Hit} and R_{IoU}, which directly reflect the evaluation metrics and serve as auxiliary constraints.

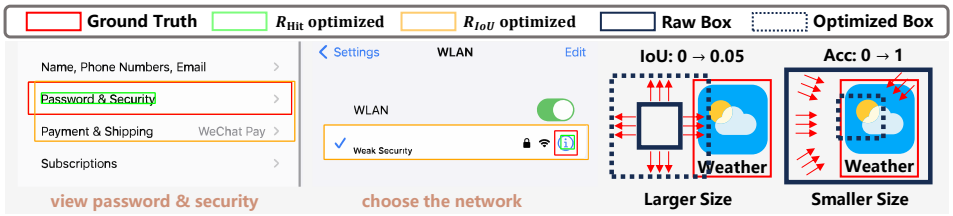

Figure 4: **(Left)** Two cases with predicted bounding boxes and golden-truth boxes. **(Right)** Two examples illustrating why R_{IoU} favors larger boxes, while R_{Hit} prefers smaller ones.

3.3 Analysis on GRPO Objective

Recent approaches to improving GUI agents [25, 28, 40] have adopted RL techniques, such as the GRPO algorithm proposed by DeepSeekMath [34]. GRPO optimizes the policy π_θ by sampling a set of candidate responses $\{\mathbf{o}_i\}_{i=1}^{N}$ from the old policy $\pi_{\theta_{\text{old}}}$ for each input query $\mathbf{q}$, where each response $\mathbf{o}_i$ has length $|\mathbf{o}_i|$. The policy is updated based on a normalized advantage $\hat{A}_{i,t}$ computed for each token, forming the objective $\mathcal{J}_{\text{GRPO}}(\pi_\theta)$:

$$\mathcal{J}_{\text{GRPO}}(\pi_\theta) = \mathbb{E}_{\mathbf{q}\sim p_{\mathcal{Q}},\{\mathbf{o}_i\}_{i=1}^{N}\sim\pi_{\theta_{\text{old}}}(\cdot|\mathbf{q})}$$
$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathbf{o}_i|}\sum_{t=1}^{|\mathbf{o}_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|\mathbf{q},\mathbf{o}_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right]\right\}, \tag{2}$$

where t is the index of the token in the response, ϵ is a hyperparameter that controls the maximum allowed deviation from the old policy, and $\text{clip}(\cdot, 1-\epsilon, 1+\epsilon)$ applies clipping to stabilize training.

In the setting of GUI grounding tasks, Eq. 2 introduces two biases (see also in Figure 5):

Figure 5: Illustration of the response-level length biases and query-level difficulty biases in GRPO.

- **Response-level length bias [26]**: It has been observed [26] that GRPO introduces a length bias: longer responses are preferred among incorrect ones, while shorter responses are favored among correct ones. This arises from dividing the objective $\mathcal{J}_{\text{GRPO}}(\pi_\theta)$ by $|\mathbf{o}_i|$, which amplifies the per-token gradient for shorter responses when the advantage is positive ($\hat{A}_{i,t} > 0$), pushing the policy toward simpler correct outputs. Conversely, it encourages unnecessarily long incorrect answers. As shown in Figure 6 (Left), training gradually results in longer incorrect and shorter correct responses. This trend further harms performance, as longer outputs are shown to degrade accuracy in Section 3.1. Therefore, length bias in grounding tasks is especially problematic: it not only increases token count but also reduces overall quality.
- **Question-level difficulty bias**: It has been noted in [26] that dividing the centered outcome rewards by std$(r_1, r_2, \ldots, r_N)$ can lead the model to focus disproportionately on either harder or easier samples. However, we argue that assigning higher weights to harder samples during policy updates is desirable. In grounding tasks, the relative box size of the target can serve as a proxy for task difficulty [18]. Based on this intuition, we modify the original objective to $w_q \cdot \mathcal{J}_{\text{GRPO}}(\pi_\theta)$, where w_q reflects the difficulty of query $\mathbf{q}$. The weight w_q is computed based on the relative box size, where a larger relative size indicates an easier grounding instance. Detailed computation is provided in Appendix C.3. Multiplying the objective by w_q assigns greater gradients to harder samples, thus encouraging the model to focus on more challenging instances. **In fact, length bias can also be viewed as a form of difficulty bias, as it guides the model toward generating longer incorrect responses, which exacerbates the difficulty of learning from such examples and indirectly shifts the focus toward easier samples.**

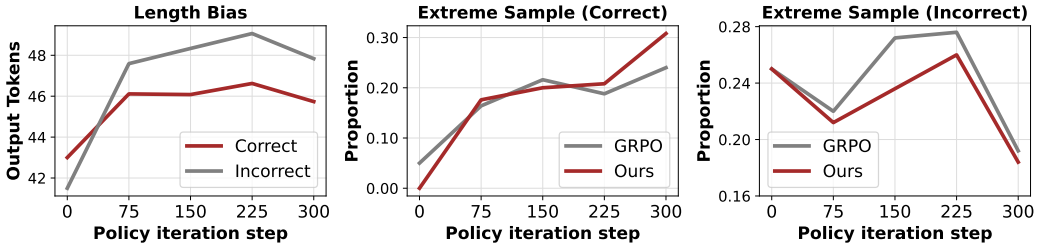

Figure 6: Changes in output length and proportion of extreme samples during policy training.

Experimental Results. We implement both improvements, with the results reported in Table 2 and in Table 9 (Appendix D.2). Mitigating length and difficulty biases consistently enhances model performance. Figure 6 (middle and right) further tracks the ratio of extreme samples, where all sampled responses are either correct or incorrect, throughout training. In the middle plot, our method initially lags on easy samples due to their lower weights, but gradually outperforms the original GRPO as these examples are eventually learned. In the right plot, our method maintains a lower proportion of extreme cases on hard samples, indicating that difficulty re-weighting facilitates better learning from challenging instances.

Table 2: Evaluation results on ScreenSpot after mitigating length and difficulty biases.

Training Objective	Mobile			Desktop			Web			Avg.
	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.	
Standard GRPO [34]	96.5	82.4	90.6	87.6	60.7	76.3	85.0	68.4	77.3	82.3
$\lvert\mathbf{o}_i\rvert \rightarrow$ `Max_Tokens` [26]	96.5	81.9	90.4	86.6	65.7	77.8	85.4	71.9	79.1	83.2
$\mathcal{J}_{\text{GRPO}}(\pi_\theta) \rightarrow w_p \cdot \mathcal{J}_{\text{GRPO}}(\pi_\theta)$	97.2	79.6	89.0	85.5	62.8	76.0	88.0	73.8	81.4	83.3

3.4 GUI-G1: A Tailored RL Visual Grounding Model

Based on the above analysis, we identify key limitations in existing training paradigms for grounding tasks. We now summarize our proposed improvements and present a comparison with prior methods.

Our method, GUI-G1, addresses the identified issues through:

- **Thinking leads to poorer grounding performance in Sec. 3.1**: We adopt a template without intermediate reasoning to prevent the policy from generating long thinking during training.
- **R_{Hit} and R_{IoU} cause opposing types of reward hacking in Sec. 3.2**: We combine R_{Hit} and R_{IoU} as the reward signal and introduce an additional R_{Box} term to regularize predicted box sizes, mitigating reward hacking caused by box size mismatch.
- **Original GRPO introduces length and difficulty biases in Sec. 3.3**: We remove these biases by replacing $|o_i|$ with a constant `Max_Tokens` [26] and by weighting the GRPO objective $\mathcal{J}_{\text{GRPO}}(\pi_\theta)$ with a difficulty coefficient w_p.

To make the distinctions clearer, Table 3 provides a structured comparison between GUI-G1 and existing R1-style GUI agents in grounding tasks.

Table 3: Comparison of R1-style GUI agents for grounding tasks, focusing on RL, template, reward, and support for length control and difficulty awareness. α and β are tunable hyperparameters.

Method	RL	Template Design	Reward Design	Length Control	Difficulty Aware
U1-R1 [28]	GRPO	Thinking Template	R_{Hit}	✗	✗
GUI-R1 [40]	GRPO	Thinking Template	R_{Hit}	✗	✗
InfiGUI-R1 [25]	RLOO [1, 16]	Thinking Template	$R_{\text{Hit}} + R_{\text{IoU}}$	✗	✗
GUI-G1 (Ours)	GRPO	No-Thinking Template	$R_{\text{Hit}} + \alpha R_{\text{IoU}} + \beta R_{\text{Box}}$	✓	✓

4 Experiments

In this section, we introduce the experimental setup for training and evaluating our proposed **GUI-G1-3B** agent. We outline the implementation details, describe the training dataset and evaluation benchmarks, and provide a detailed comparison with state-of-the-art methods.

Implementation Details. Our model is built upon the Qwen2.5-VL-3B-Instruct and trained using the VLM-R1 framework [35]. The reward function follows the form $R_{\text{Hit}} + \alpha R_{\text{IoU}} + \beta R_{\text{Box}}$, where α is set to 0.25 and β to 0.125. We conduct training on 4 NVIDIA H800 GPUs over 3 days, with a global batch size of 32 and a learning rate of 1×10^{-6}. No KL divergence regularization is applied. Only one training epoch is required.

Training Dataset and Evaluation Benchmarks. We construct a 17K-sample grounding dataset spanning three domains: **Mobile** (from UI-BERT [3]), **Web** (from OS-Atlas [39]), and **Desktop** (from OS-Atlas, covering Windows, Linux, and MacOS). More details of the training dataset are provided in the Appendix D.1. To ensure data quality, each sample is prompted eight times using Qwen2.5-VL-3B-Instruct, and those with consistently correct or incorrect responses are discarded [6]. For evaluation, we adopt ScreenSpot [8] and **ScreenSpot-Pro** [18]. While ScreenSpot assesses grounding performance across diverse platforms, including Mobile, Web, and Desktop, ScreenSpot-Pro emphasizes more challenging desktop scenarios, featuring high-resolution screens.

Performance Comparison on ScreenSpot. We compare **GUI-G1-3B** with a range of state-of-the-art open-source and proprietary GUI agents, using results reported in their original papers. Table 4 summarizes performance on the ScreenSpot benchmark. **GUI-G1-3B** achieves state-of-the-art results, outperforming proprietary systems like Gemini 2.0 [9], general-purpose models such as the Qwen2.5 series [4], GUI-specific SFT models like OS-Atlas [39] and UGround [11], as well as R1-style models including UI-R1 [28], GUI-R1 [40], and InfiGUI-R1 [25]. It also surpasses larger models like OS-Atlas-7B [39]. Despite its strong performance, our model is trained on only 17K samples and requires no intermediate reasoning steps. Moreover, it achieves higher inference efficiency by generating significantly fewer tokens than other methods (see Appendix D.3, Table 10).

Performance Comparison on ScreenSpot-Pro. As shown in Table 5, **GUI-G1-3B** achieves competitive performance on the challenging ScreenSpot-Pro benchmark, with an overall average score of

Table 4: Performance on ScreenSpot across Mobile, Desktop, and Web. **Bold** highlights the best results, underlined the second-best. "-" indicates missing values due to unavailable results in the original paper, unreleased model checkpoints, and unreleased inference code.

Model	#Training Samples	Accuracy (%)						Avg.
		Mobile		Desktop		Web		
		Text	Icon	Text	Icon	Text	Icon	
Proprietary Models								
GPT-4o [31]	-	30.5	23.2	20.6	19.4	11.1	7.8	18.8
Claude Computer Use [2]	-	-	-	-	-	-	-	83.0
Gemini 2.0 (Project Mariner) [9]	-	-	-	-	-	-	-	84.0
General Open-source Models								
Qwen2-VL-7B [37]	-	61.3	39.3	52.0	45.0	33.0	21.8	42.9
Qwen2.5-VL-3B [4]	-	-	-	-	-	-	-	55.5
Qwen2.5-VL-7B [4]	-	-	-	-	-	-	-	84.7
GUI-specific Models (SFT)								
CogAgent-18B [13]	222M	67.0	24.0	74.2	20.0	70.4	28.6	47.4
SeeClick-9.6B [8]	1M	78.0	52.0	72.2	30.0	55.7	32.5	53.4
UGround-7B [11]	10M	82.8	60.3	82.5	63.6	80.4	70.4	73.3
UGround-v1-7B [11]	-	93.0	79.9	93.8	76.4	90.9	**84.0**	86.3
OS-Atlas-7B [39]	13M	93.0	72.9	91.8	62.9	90.9	74.3	82.5
ShowUI-2B [23]	256K	92.3	75.5	76.3	61.1	81.7	63.6	75.1
Aguvis-7B [41]	1M	95.6	77.7	93.8	67.1	88.3	75.2	84.4
UI-TARS-2B [33]	-	93.0	75.5	90.7	68.6	84.3	74.8	82.3
UI-TARS-7B [33]	-	94.5	85.2	95.9	**85.7**	90.0	83.5	87.7
GUI-specific Models (RL)								
UI-R1-3B [28]	136	-	-	90.2	59.3	85.2	73.3	-
GUI-R1-3B [40]	3K	-	-	93.8	64.8	89.6	72.1	-
GUI-R1-7B [40]	3K	-	-	91.8	73.6	91.3	75.7	-
InfiGUI-R1-3B [25]	32K	97.1	81.2	94.3	77.1	**91.7**	77.6	87.5
Ours								
GUI-G1-3B	17K	**98.6**	**85.8**	**96.4**	80.7	91.4	82.3	**90.3**

37.1%. It outperforms the larger UI-TARS-7B model (35.7%) and significantly surpasses the best-performing R1-based model, InfiGUI-R1-3B (35.7%). Although both **GUI-G1-3B** and OS-Atlas-7B use the same training dataset, our model performs worse on the OS subset (16.1% vs. OS-Atlas-7B's 16.8%), suggesting that its gains mainly result from post-training that activates pretrained knowledge rather than from task-specific data. This demonstrates the robustness and generalization ability of our approach in real-world scenarios. Additionally, we apply our improved training approach to the GUI-G1 model across various model sizes and families, and reproduce the UI-R1 setting in the Appendix D.4. The results confirm that GUI-G1 maintains strong performance, highlighting the robustness and effectiveness of our method.

5 Related Work

Grounding for GUI Agents. Grounding is central to GUI agents research [38, 47], driving advances in data collection and model design. Early works such as VUT [21] and Spotlight [17] aligned task structures and modalities (e.g., screenshots, instructions) using BERT-based [10] representations, while RUIG [45] employed reinforcement learning to map instructions to UI coordinates. With MLLMs, the focus shifted to fine-tuning pretrained models for cross-platform interaction and GUI adaptation. ShowUI [23] improved efficiency by reducing redundant visual tokens, and Ferret-UI 2 [22] enhanced understanding via high-resolution encoding and cross-platform generalization. Aria-UI [43] introduced multi-turn grounding with sequential reasoning for dynamic multi-step interactions. More recently, OS-Atlas [39] and UGround [11] advanced the field by releasing large open-source datasets and training models robust to out-of-distribution tasks. Unlike these data-intensive, supervised approaches, our work explores how minimal data and an R1-Zero-like method can unlock MLLM grounding capability for GUI tasks.

Table 5: Comparison of agent models on ScreenSpot-Pro across Text, Icon, and Average task metrics. Best results are shown in **bold**, with second-best results underlined.

Model	CAD			Development			Creative			Scientific			Office			OS			Avg.		
	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.	Text	Icon	Avg.
Proprietary Models																					
GPT-4o [31]	2.0	0.0	1.5	1.3	0.0	0.7	1.0	0.0	0.6	2.1	0.0	1.2	1.1	0.0	0.9	0.0	0.0	0.0	1.3	0.0	0.8
Claude Computer Use [2]	14.5	3.7	11.9	22.0	3.9	12.6	25.9	3.4	16.8	33.9	15.8	25.8	30.1	16.3	26.9	11.0	4.5	8.1	23.4	7.1	17.1
General Open-source Models																					
Qwen2-VL-7B [37]	0.5	0.0	0.4	2.6	0.0	1.3	1.5	0.0	0.9	6.3	0.0	3.5	3.4	1.9	3.0	0.9	0.0	0.5	2.5	0.2	1.6
Qwen2.5-VL-3B [4]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	23.9
Qwen2.5-VL-7B [4]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	29.0
Kimi-VL [36]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	34.5
GUI-specific Models (SFT)																					
SeeClick [8]	2.5	0.0	1.9	0.6	0.0	0.3	1.0	0.0	0.6	3.5	0.0	2.0	1.1	0.0	0.9	2.8	0.0	1.5	1.8	0.0	1.1
CogAgent-18B [13]	7.1	3.1	6.1	14.9	0.7	8.0	9.6	0.0	5.6	22.2	1.8	13.4	13.0	0.0	10.0	5.6	0.0	3.1	12.0	0.8	7.7
Aria-UI [43]	7.6	1.6	6.1	16.2	0.0	8.4	23.7	2.1	14.7	27.1	6.4	18.1	20.3	1.9	16.1	4.7	0.0	2.6	17.1	2.0	11.3
OS-Atlas-4B [39]	2.0	0.0	1.5	7.1	0.0	3.7	3.0	1.4	2.3	9.0	5.5	7.5	5.1	3.8	4.8	5.6	0.0	3.1	5.0	1.7	3.7
OS-Atlas-7B [39]	12.2	4.7	10.3	33.1	1.4	17.7	28.8	2.8	17.9	37.5	7.3	24.4	33.9	5.7	27.4	27.1	4.5	16.8	28.1	4.0	18.9
ShowUI-2B [23]	2.5	0.0	1.9	16.9	1.4	9.4	9.1	0.0	5.3	13.2	7.3	10.6	15.3	7.5	13.5	10.3	2.2	6.6	10.8	2.6	7.7
UGround-7B [11]	14.2	1.6	11.1	26.6	2.1	14.7	27.3	2.8	17.0	31.9	2.7	19.3	31.6	11.3	27.0	17.8	0.0	9.7	25.0	2.8	16.5
UGround-V1-7B [11]	-	-	13.5	-	-	35.5	-	-	27.8	-	-	38.8	-	-	48.8	-	-	26.1	-	-	31.1
UI-TARS-2B [33]	17.8	4.7	14.6	47.4	4.1	26.4	42.9	6.3	27.6	56.9	17.3	39.8	50.3	17.0	42.6	21.5	5.6	14.3	39.6	8.4	27.7
UI-TARS-7B [33]	20.8	9.4	18.0	**58.4**	**12.4**	**36.1**	**50.0**	9.1	**32.8**	**63.9**	**31.8**	**50.0**	63.3	20.8	53.5	30.8	**16.9**	24.5	47.8	16.2	35.7
GUI-specific Models (RL)																					
InfiGUI-R1-3B [25]	33.0	**14.1**	28.4	51.3	**12.4**	32.4	44.9	7.0	29.0	58.3	20.0	41.7	65.5	28.3	57.0	**43.9**	12.4	**29.6**	49.1	14.1	35.7
UI-R1-3B [28]	11.2	6.3	-	22.7	4.1	-	27.3	3.5	-	42.4	11.8	-	32.2	11.3	-	13.1	4.5	-	-	-	17.8
GUI-R1-3B [40]	26.4	7.8	-	33.8	4.8	-	40.9	5.6	-	61.8	17.3	-	53.6	17.0	-	28.1	5.6	-	-	-	-
GUI-R1-7B [40]	23.9	6.3	-	49.4	4.8	-	38.9	8.4	-	55.6	11.8	-	58.7	26.4	-	42.1	**16.9**	-	-	-	-
Ours																					
GUI-G1-3B	**39.6**	9.4	**32.2**	50.7	10.3	31.1	36.6	**11.9**	26.6	61.8	30.0	48.0	**67.2**	**32.1**	**59.1**	23.5	10.6	16.1	**49.5**	**16.8**	**37.1**

R1-Zero-like Training for MLLMs. DeepSeek-R1-Zero [12] introduced a GRPO-based post-training framework that enhances reasoning via structured outputs. This approach was extended to multimodal settings by Vision-R1 [14], MM-EUREKA [29], and VisualThinker-R1-Zero [46], all improving vision-language reasoning. LMM-R1 [32] achieved strong results with a two-stage RL scheme and low cost. However, recent work [19] found that reasoning-averse models can outperform reasoning-based ones in multimodal classification, suggesting reasoning is not always beneficial. In GUI agents, studies such as UI-R1 [28], GUI-R1 [40], and InfiGUI-R1 [25] verified the effectiveness of R1-Zero-like training in action prediction and grounding, showing consistent gains on ScreenSpot [8], ScreenSpot-Pro [18], and AndroidControl [20]. This work focuses on the grounding task and revisits whether the standard R1-Zero-like settings are suitable for GUI scenarios.

6 Conclusion

In this work, we revisit the R1-Zero-style training setup for GUI grounding agents from three aspects: input design, reward evaluation, and policy update. We show that lengthy reasoning impairs grounding performance and propose a Fast Thinking Template to address it. We then find and analyze opposing types of reward hacking issues in existing reward designs and introduce a box-size constraint to mitigate them, leading to improved performance. Finally, we examine GRPO's length and difficulty biases in grounding tasks and address them by removing length normalization and incorporating difficulty-based weighting. With only 17K samples, our **GUI-G1-3B** surpasses larger R1-style models on ScreenSpot and ScreenSpot-Pro.

7 Acknowledgments

This work was funded by the National Key R&D Program of China (2023YFA1008704), the National Natural Science Foundation of China (62472426,62376275). Supported by fund for building world-class universities (disciplines) of Renmin University of China. Work partially done at Beijing Key Laboratory of Research on Large Models and Intelligent Governance, and Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE. We thank Yuchong Sun for helpful discussions and insights during the development of this work.

References

[1] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

[2] Anthropic. Developing a computer use model. `https://www.anthropic.com/news/developing-computer-use`, 2024. Accessed: 2025-04-12.

[3] C. Bai, X. Zang, Y. Xu, S. Sunkara, A. Rastogi, J. Chen, et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint arXiv:2107.13731*, 2021.

[4] 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*, 2025.

[5] Y. Chai, S. Huang, Y. Niu, H. Xiao, L. Liu, D. Zhang, P. Gao, S. Ren, and H. Li. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490*, 2024.

[6] Z. Chen, Y. Min, B. Zhang, J. Chen, J. Jiang, D. Cheng, W. X. Zhao, Z. Liu, X. Miao, Y. Lu, et al. An empirical study on eliciting and improving r1-like reasoning models. *arXiv preprint arXiv:2503.04548*, 2025.

[7] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024.

[8] K. Cheng, Q. Sun, Y. Chu, F. Xu, L. YanTao, J. Zhang, and Z. Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, 2024.

[9] G. DeepMind. Gemini-2.0 (project mariner). `https://deepmind.google/technologies/project-mariner`, 2024. Accessed: 2025-04-12.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[11] B. Gou, R. Wang, B. Zheng, Y. Xie, C. Chang, Y. Shu, H. Sun, and Y. Su. Navigating the digital world as humans do: Universal visual grounding for gui agents. In *13th International Conference on Learning Representations, ICLR 2025*, 2025.

[12] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[13] W. Hong, W. Wang, Q. Lv, J. Xu, W. Yu, J. Ji, Y. Wang, Z. Wang, Y. Dong, M. Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290, 2024.

[14] W. Huang, B. Jia, Z. Zhai, S. Cao, Z. Ye, F. Zhao, Z. Xu, Y. Hu, and S. Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

[15] D. Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, October 2011.

[16] W. Kool, H. van Hoof, and M. Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

[17] G. Li and Y. Li. Spotlight: Mobile ui understanding using vision-language models with a focus. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[18] 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. *arXiv preprint arXiv:2504.07981*, 2025.

[19] M. Li, J. Zhong, S. Zhao, Y. Lai, and K. Zhang. Think or not think: A study of explicit thinking in rule-based visual reinforcement fine-tuning. *arXiv e-prints*, pages arXiv–2503, 2025.

[20] W. Li, W. Bishop, A. Li, C. Rawles, F. Campbell-Ajala, D. Tyamagundlu, and O. Riva. On the effects of data scale on computer control agents. *arXiv e-prints*, pages arXiv–2406, 2024.

[21] Y. Li, G. Li, X. Zhou, M. Dehghani, and A. Gritsenko. Vut: Versatile ui transformer for multi-modal multi-task user interface modeling. *arXiv preprint arXiv:2112.05692*, 2021.

[22] Z. Li, K. You, H. Zhang, D. Feng, H. Agrawal, X. Li, M. P. S. Moorthy, J. Nichols, Y. Yang, and Z. Gan. Ferret-ui 2: Mastering universal user interface understanding across platforms. *arXiv preprint arXiv:2410.18967*, 2024.

[23] K. Q. Lin, L. Li, D. Gao, Z. Yang, S. Wu, Z. Bai, W. Lei, L. Wang, and M. Z. Shou. Showui: One vision-language-action model for gui visual agent. *arXiv preprint arXiv:2411.17465*, 2024.

[24] X. Liu, B. Qin, D. Liang, G. Dong, H. Lai, H. Zhang, H. Zhao, I. L. Iong, J. Sun, J. Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.

[25] Y. Liu, P. Li, C. Xie, X. Hu, X. Han, S. Zhang, H. Yang, and F. Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025.

[26] Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. S. Lee, and M. Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[27] Y. Lu, J. Yang, Y. Shen, and A. Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024.

[28] Z. Lu, Y. Chai, Y. Guo, X. Yin, L. Liu, H. Wang, G. Xiong, and H. Li. Ui-r1: Enhancing action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025.

[29] F. Meng, L. Du, Z. Liu, Z. Zhou, Q. Lu, D. Fu, T. Han, B. Shi, W. Wang, J. He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

[30] OpenAI. Learning to reason with llms. urlhttps://openai.com/index/learning-to-reason-with-llms/. Accessed: 15 March 2025.

[31] OpenAI. Gpt-4o, 2024. Accessed: 2025-01-03.

[32] Y. Peng, G. Zhang, M. Zhang, Z. You, J. Liu, Q. Zhu, K. Yang, X. Xu, X. Geng, and X. Yang. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. *arXiv preprint arXiv:2503.07536*, 2025.

[33] 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.

[34] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[35] H. Shen, P. Liu, J. Li, C. Fang, Y. Ma, J. Liao, Q. Shen, Z. Zhang, K. Zhao, Q. Zhang, R. Xu, and T. Zhao. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

[36] K. Team, A. Du, B. Yin, B. Xing, B. Qu, B. Wang, C. Chen, C. Zhang, C. Du, C. Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025.

[37] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[38] S. Wang, W. Liu, J. Chen, Y. Zhou, W. Gan, X. Zeng, Y. Che, S. Yu, X. Hao, K. Shao, et al. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2024.

[39] 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. In *13th International Conference on Learning Representations, ICLR 2025*, 2025.

[40] X. Xia and R. Luo. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025.

[41] Y. Xu, Z. Wang, J. Wang, D. Lu, T. Xie, A. Saha, D. Sahoo, T. Yu, and C. Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024.

[42] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[43] Y. Yang, Y. Wang, D. Li, Z. Luo, B. Chen, C. Huang, and J. Li. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024.

[44] L. Zhang, L. Gao, and M. Xu. Does chain-of-thought reasoning help mobile gui agent? an empirical study. *arXiv preprint arXiv:2503.16788*, 2025.

[45] Z. Zhang, W. Xie, X. Zhang, and Y. Lu. Reinforced ui instruction grounding: Towards a generic ui task automation api. *arXiv preprint arXiv:2310.04716*, 2023.

[46] H. Zhou, X. Li, R. Wang, M. Cheng, T. Zhou, and C.-J. Hsieh. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*, 2025.

[47] Y. Zhou, S. Wang, S. Dai, Q. Jia, Z. Du, Z. Dong, and J. Xu. Chop: Mobile operating assistant with constrained high-frequency optimized subtask planning. *arXiv preprint arXiv:2503.03743*, 2025.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our key findings and contributions, including critical analysis of R1-Zero components and the proposal of a new method, which are supported by experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper includes a dedicated "Limitations" section in the Appendix A that clearly discusses the constraints of our approach. These include dataset coverage, key assumptions made during modeling, and the sensitivity of the method to specific configurations. The section also reflects on the generalizability of the results and the computational considerations, offering a transparent view of the work's scope.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer:[NA]

   Justification: The paper does not include any theoretical results or formal proofs.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides all necessary information to reproduce the main experimental results. We include detailed descriptions of the training objectives in Sec 3.3. The training settings, dataset preprocessing procedures, computing resources, evaluation protocols, and model architecture are specified in Sec 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include code, data, and reproduction instructions (including environment and commands) via an anonymous link in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe the experimental settings and necessary details in the appendix and supplemental material, with a summary provided in Sec. 4 of the main paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: Due to the computational cost of large-scale experiments, we did not perform repeated trials or statistical significance testing. All reported results are from single runs under fixed settings, ensuring consistency across comparisons.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report compute details in Sec. 4. All training was conducted on 4 NVIDIA H800 GPUs and took approximately 3 days.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our research complies with all relevant ethical guidelines. Our work does not involve human subjects, sensitive data, or foreseeable negative societal impact.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper includes a dedicated "Broader Impacts" section in the Appendix E, where we discuss potential positive outcomes such as improving accessibility and automation via GUI agents, as well as possible negative consequences, including reward hacking and model bias. We also highlight the importance of fairness and robustness in real-world deployment.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper does not release any new pretrained models or datasets that pose a risk of misuse. The study is based on public datasets and models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We used the publicly released Qwen2.5-VL-3B-Instruct model, which is licensed under the "Qwen RESEARCH LICENSE AGREEMENT," and cited the official release. We also used the ScreenSpot and ScreenSpot-Pro datasets, which are publicly available and cited accordingly in the paper. All assets were used under their respective licenses and terms of use.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We do not release a new model checkpoint, but we provide the dataset and code as supplementary material. Comprehensive documentation, including usage instructions, dataset details, and limitations, is included alongside the assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This work does not involve crowdsourcing or research with human subjects and thus no IRB approval was needed.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The research employs multimodal large language models (MLLMs) as a key component in the proposed method, specifically using Qwen2.5-VL-3B-Instruct for training the GUI agent, which is central to the approach and results.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Limitations

While our method demonstrates strong performance in GUI grounding, there remain several limitations that offer directions for future work: (1) **Focus on grounding.** The current work focuses on grounding, which is essential for GUI agents, but does not cover tasks like action prediction or long-horizon planning. Future research can extend this approach to support full decision-making in GUI interaction. (2) **Scope of RL analysis.** The study mainly examines online reinforcement learning, especially GRPO. Other factors such as dataset composition, model design, and hyperparameter tuning are not fully explored and deserve further analysis. (3) **Limited training data.** Our model is trained on a relatively small set of public datasets, which constrains its performance ceiling. In future work, we plan to scale up training using larger and more diverse datasets, such as those adopted in GUI-R1 [40], to further improve generalization and robustness.

B Fast Thinking Template versus Slow Thinking Template

To examine the role of reasoning in grounding tasks, we formally define two instruction templates, the Fast Thinking Template and the Slow Thinking Template. We then train models under both settings to compare their effects on grounding performance. The detailed training configurations and implementation settings are provided in Section B.2.

B.1 Formal Definition of Fast and Slow Thinking

Following Thinking, Fast and Slow [15], Fast Thinking corresponds to System 1, characterized by intuitive, automatic, and effortless responses. Slow Thinking corresponds to System 2, characterized by deliberate, analytical, and effortful reasoning. In the context of large language models, Fast Thinking denotes direct answer generation without intermediate reasoning, as exemplified by the non-thinking mode in Qwen3 [42]. Slow Thinking refers to explicit multi-step reasoning processes, such as those elicited through chain-of-thought prompting. In our work, Fast Thinking represents direct grounding prediction from visual inputs, whereas Slow Thinking involves reasoning over the observed scene before producing the final grounding output.

B.2 Experimental Setting

We train our Qwen2.5-VL-3B-Instruct model on the UI-R1-3B-Train dataset[2], which consists of 101 samples with grounding annotations. During training, the vision encoder is kept frozen. We set the learning rate to 1×10^{-6} and the sampling temperature to 0.9, generating eight responses per prompt. Training is performed on four L20 (48G) GPUs with a batch size of four samples per GPU, $\beta = 0$, and a gradient accumulation step of one.

B.3 Experimental Results

The results in Table 6 and Table 7 show that encouraging direct answers with Fast Thinking Template consistently leads to better performance, supporting our hypothesis that explicit reasoning can hinder grounding effectiveness.

Table 6: Results on ScreenSpot after optimization with Fast and Slow Thinking Templates.

Template	Mobile	Desktop	Web	Avg.
Slow Thinking Template	94.6	84.7	84.9	88.7
Fast Thinking Template	96.2	87.4	84.3	**89.8**

[2] https://huggingface.co/datasets/LZXzju/UI-R1-3B-Train

Table 7: Results on ScreenSpot-Pro after optimization with Fast and Slow Thinking Templates.

Template	CAD	Development	Creative	Scientific	Office	OS	Avg.
Slow Thinking Template	18.0	19.1	20.5	32.3	40.0	19.4	24.4
Fast Thinking Template	20.3	19.1	21.1	33.1	43.5	17.3	**25.3**

C Analysis Experiments Settings

C.1 Training Details

We fine-tune all parameters of the Qwen2.5-VL-3B-Instruct model using samples evenly drawn from three domains: mobile (UIBERT [3]), web, and desktop (OS-Atlas [39]). Due to computational constraints, we randomly sample 300 grounding examples from each domain. Despite the relatively small dataset, the model achieves strong performance after fine-tuning, ensuring the reliability of our subsequent analysis. Training follows the default setup of the VLM-R1 repository [35], using 8 rollouts per example and no KL-divergence regularization by default.

C.2 Evaluation Metrics

Specifically, we use the **relative box size** to measure the size of the predicted bounding box. It is calculated as:

$$\lambda = \frac{\hat{y}_2 + \hat{x}_2 - \hat{y}_1 - \hat{x}_1}{\texttt{IMAGE\_WIDTH} + \texttt{IMAGE\_HEIGHT}},$$

where `IMAGE_WIDTH` and `IMAGE_HEIGHT` denote the pixel width and height of the input image, respectively.

C.3 Difficulty-Aware Weighting Strategy

To compute the difficulty weight w_q for each sample based on its relative box size $\lambda_q \in (0, 1]$, we first take the inverse of the size to reflect the intuition that smaller boxes are harder: $\lambda'_q = \frac{1}{\lambda_i}$. We then normalize the inverted values to the range $[0, 1]$ by computing $\tilde{\lambda}_q = \frac{\lambda'_q - \min_i \lambda'_i}{\max_i \lambda'_i - \min_i \lambda'_i}$. Finally, we linearly rescale the normalized scores to the interval $(0.5, 1.5]$ to obtain the final difficulty weights: $w_q = 0.5 + \tilde{\lambda}_q$. Putting everything together, the final formula is

$$w_q = 0.5 + \frac{\frac{1}{\lambda_q} - \min_i \left(\frac{1}{\lambda_i}\right)}{\max_i \left(\frac{1}{\lambda_i}\right) - \min_i \left(\frac{1}{\lambda_i}\right)}.$$

This ensures that harder samples (with smaller boxes) receive higher weights, while keeping the values in a stable and bounded range.

D More Experiments Details

D.1 Training Data Composition

To provide a comprehensive grounding resource across diverse platforms, we construct a dataset containing 17K samples distributed across three representative domains: Mobile, Web, and Desktop.

- The Mobile domain is derived from the UI-BERT dataset [3], which consists of user interface data collected from Android applications.
- The Web domain is sourced from OS-Atlas [39], including interactive web elements and browser-based environments.
- The Desktop domain is also based on OS-Atlas, but focuses on native applications and interfaces from major desktop operating systems, including Windows, Linux, and MacOS.

Each domain contains a balanced set of grounding instances that pair natural language commands with corresponding UI elements. Table 8 summarizes the number of samples collected in each domain.

Table 8: Statistics and sources of the grounding dataset across five platforms.

	Mobile	Web	Windows	Linux	MacOS
Source	UI-BERT [3]	OS-Atlas [39]	OS-Atlas [39]	OS-Atlas [39]	OS-Atlas [39]
Size	575	7,832	5,576	1,667	1,835

D.2 Debias Results on ScreenSpot-Pro

The results in Table 9 indicate that both length bias correction and difficulty reweighting enhance performance on ScreenSpot-Pro. The limited effect of length bias removal may stem from the predominance of high-resolution images, where text tokens contribute little relative to image tokens. In contrast, difficulty reweighting produces notable improvements, validating its benefit for harder samples.

Table 9: Evaluation results on ScreenSpot-Pro after mitigating length and difficulty biases.

Model	CAD	Development	Creative	Scientific	Office	OS	Avg.
Standard GRPO [34]	17.2	18.4	20.8	30.7	40.4	14.8	23.5
$\lvert o_i \rvert \rightarrow$ `Max_Tokens` [26]	16.1	19.7	21.4	32.3	40.4	14.8	23.9
$\mathcal{J}_{\text{GRPO}}(\pi_\theta) \rightarrow w_p \cdot \mathcal{J}_{\text{GRPO}}(\pi_\theta)$	21.5	20.1	22.0	31.1	45.2	15.8	**25.6**

D.3 Output Token Efficiency Analysis

To assess the efficiency of different models during inference, we compare the average number of output tokens generated per example across Mobile, Desktop, and Web domains on ScreenSpot [8]. As shown in Table 10, **GUI-G1-3B** generates substantially fewer tokens than **InfiGUI-R1-3B** [25] in all domains—approximately one-third as many on average—while maintaining or even improving task accuracy. This compact output not only reduces computational cost but also reflects the model's ability to produce precise and concise responses without relying on verbose intermediate reasoning.

Table 10: Average number of output tokens generated per example on ScreenSpot during inference.

Model	Mobile	Desktop	Web
InfiGUI-R1-3B [25]	107	107	114
GUI-G1-3B	37	39	39

D.4 Experiments Across Various Model Sizes and Families

To evaluate the generality of our method, we conduct experiments across different model families and sizes. We apply our approach to three models: InternVL3-2B-Instruct (a smaller model from a different family), Qwen2.5-VL-3B-Instruct, and Qwen2.5-VL-7B-Instruct (a larger variant). Following the training protocol of UI-R1 [28], we retrain the 7B and 2B models on the same dataset, while Qwen2.5-VL-3B-Instruct is retrained using our proposed method for direct comparison. Due to resource constraints, LoRA fine-tuning is applied to the 7B model, whereas the others are fully fine-tuned. All models use identical training data and hyperparameter settings for a fair comparison. Additional training details are provided in Appendix B.2.

The results in Table 11 and Table 12 show that our method consistently improve performance across both Qwen and InternVL families, although InternVL generally shows lower baselines. This trend aligns with recent observations that most leading GUI agents [28, 40, 25] adopt Qwen as their backbone. In terms of model scale, increasing from 3B to 7B unexpectedly results in lower grounding

accuracy. We attribute this to two factors: (a) the 7B model relies on LoRA instead of full fine-tuning, and (b) larger models tend to benefit less from task-specific grounding fine-tuning, as similarly observed in Table 1 of GUI-R1 [40]. For example, UI-TARS-72B performs worse on ScreenSpot (88.4) than the smaller UI-TARS-7B (89.5). Our results on the Qwen series further confirm that grounding performance does not necessarily scale with model size.

Table 11: Performance on ScreenSpot across model sizes and families under our method and the UI-R1 [28] setting.

Model	Mobile	Desktop	Web	Avg.
Qwen2.5-VL-3B				
Base Model	85.6	70.4	65.9	74.8
UI-R1 [28]	-	-	-	85.4
Our Method	96.2	87.4	84.3	**89.8**
Qwen2.5-VL-7B				
Base Model	88.2	80.5	80.1	83.4
UI-R1 [28]	80.6	78.7	84.2	81.4
Our Method	88.8	82.3	80.8	**84.4**
InternVL3-2B-Instruct [7]				
Base Model	0.2	13.2	0.2	3.6
UI-R1 [28]	0.4	14.4	0.5	4.1
Our Method	0.8	16.8	0.2	**4.8**

Table 12: Performance on ScreenSpot-Pro across model sizes and families under our method and the UI-R1 [28] setting.

Model	CAD	Development	Creative	Scientific	Office	OS	Avg.
Qwen2.5-VL-3B							
Base Model	13.0	10.7	14.4	24.4	25.7	10.7	16.3
UI-R1 [28]	-	-	-	-	-	-	17.8
Our Method	20.3	19.1	21.1	33.1	43.5	17.3	**25.3**
Qwen2.5-VL-7B							
Base Model	9.6	13.4	14.7	25.2	35.7	14.3	18.3
UI-R1 [28]	10.3	13.4	16.1	25.2	39.6	18.9	19.9
Our Method	9.6	14.1	15.5	26.7	35.2	17.4	**19.2**
InternVL3-2B-Instruct [7]							
Base Model	1.5	0.0	0.6	1.2	0.4	0.0	0.6
UI-R1 [28]	3.1	0.7	0.9	3.5	1.3	0.0	1.6
Our Method	3.5	0.0	0.9	3.9	1.3	0.5	**1.6**

E Broader Impacts

Our work contributes to the development of more robust and accurate GUI agents by addressing key training challenges in reinforcement learning for visual grounding. This could improve the reliability of accessibility tools and human-computer interaction systems. However, care should be taken when deploying such agents in real-world systems, as reward design choices may cause unintended behavior such as reward hacking or bias toward easy cases. We encourage future research to further study fairness, robustness, and privacy considerations in GUI agent training and deployment.