Learn Where to Click from Yourself: On-Policy Self-Distillation for GUI Grounding

Yan Zhang[1,4,*] **Daiqing Wu**[3,*] **Huawen Shen**[1,4] **Can Ma**[1,4,†] **Yu Zhou**[2,†]
[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]VCIP & TMCC & DISSec, College of Computer Science, Nankai University
[3]Department of Psychological and Cognitive Sciences, Tsinghua University
[4]School of Cyber Security, University of Chinese Academy of Sciences
zhangyan2022@iie.ac.cn; yzhou@nankai.edu.cn

Abstract

Graphical User Interface (GUI) grounding maps natural language instructions to the visual coordinates of target elements and serves as a core capability for autonomous GUI agents. Recent reinforcement learning methods (e.g., GRPO) have achieved strong performance, but they rely on expensive multiple rollouts and suffer from sparse signals on hard samples. These limitations make on-policy self-distillation (OPSD), which provides dense token-level supervision from a single rollout, a promising alternative. However, its applicability to GUI grounding remains unexplored. In this paper, we present GUI-SD, the first OPSD framework tailored for GUI grounding. First, it constructs a visually enriched privileged context for the teacher using a target bounding box and a Gaussian soft mask, providing informative guidance without leaking exact coordinates. Second, it employs entropy-guided distillation, which adaptively weights tokens based on digit significance and teacher confidence, concentrating optimization on the most impactful and reliable positions. Extensive experiments on six representative GUI grounding benchmarks show that GUI-SD consistently outperforms GRPO-based methods and naive OPSD in both accuracy and training efficiency. Code and training data are available at https://zhangyan-ucas.github.io/GUI-SD/.

1 Introduction

Autonomous GUI agents have emerged as a promising direction for human-computer interaction, where GUI grounding serves as the fundamental capability of mapping natural language instructions to visual coordinates of target elements [4, 7]. To this end, a growing body of work [2, 6, 41, 53] has adopted reinforcement learning for GUI grounding, among which GRPO-based methods [21, 60] have become the dominant paradigm as shown in Figure 1(a). Specifically, given a user instruction, GRPO [8, 29] encourages the policy model to explore diverse solutions by sampling multiple rollouts, and evaluates each with a designed verifiable reward, such as binary [21], distance-constrained [48], or gaussian-based feedback [34]. The advantage of each rollout is then computed relative to the group reward distribution, steering the policy to reinforce successful explorations while discouraging unsuccessful ones [50].

Despite the advances, GRPO-based training for GUI grounding still depends on expensive multiple rollouts to estimate advantages and suffers from sparse signals on hard samples where all rollouts receive zero reward. These limitations call for a paradigm that can deliver dense supervision from

*Equal contribution
†Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

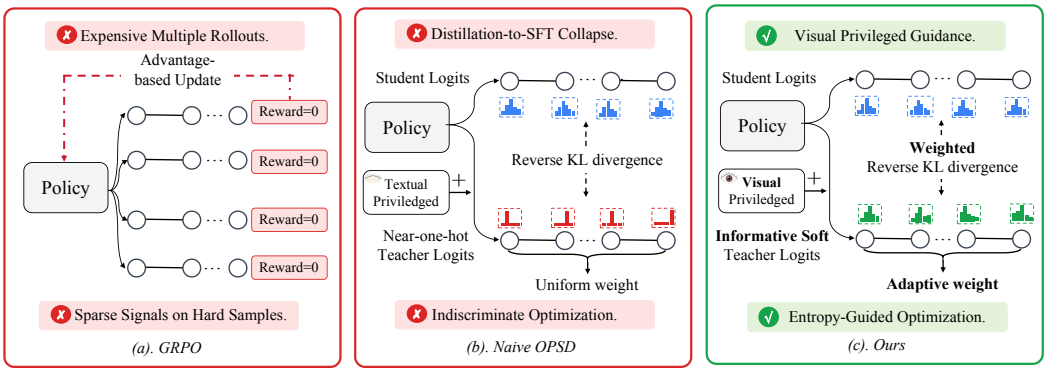


Figure 1: (a) **GRPO** requires expensive multiple rollouts and produces zero reward on hard samples. (b) **Naive OPSD** forwards the policy twice and distills via reverse KL between student and teacher logits with uniform per-token weight $w = 1.0$, yet suffers from distillation-to-SFT collapse and indiscriminate optimization. (c) **Ours** addresses both issues via visual privileged guidance and entropy-guided optimization.

fewer interactions. Recently emerging on-policy self-distillation (OPSD) [12, 28, 30, 32, 46] offers such a possibility, providing token-level supervision from a single rollout by deploying the same model as both teacher and student under asymmetric contexts. Specifically, the asymmetry lies in the privileged information that is accessible to the teacher but hidden from the student, such as reference solutions [30], verifier signals [28], and environment feedback [12]. Guided by this privileged context, the teacher acts as a stronger model, yielding a more reliable output distribution whose per-token log-probabilities form a reverse Kullback-Leibler (KL) divergence loss that continuously refines the student. By replacing sparse outcome-level rewards with dense token-level guidance, OPSD provides an appealing alternative for improving both training efficiency and supervision quality.

Motivated by these advantages, we explore for the first time the application of OPSD to GUI grounding. However, as illustrated in Figure 1(b), directly adapting OPSD to this setting encounters two critical bottlenecks: 1) **Distillation-to-SFT Collapse.** Naive OPSD paradigm directly appends the target coordinate as text to the teacher's input, causing the teacher's supervisory distribution to collapse into near-one-hot targets with near-zero entropy. In this regime, minimizing the KL divergence between teacher and student becomes equivalent to minimizing cross-entropy against hard labels, effectively reducing distillation to supervised fine-tuning (SFT) and erasing the dark knowledge [10] that makes soft-label supervision beneficial. 2) **Indiscriminate Optimization.** Naive OPSD applies reverse KL to distill all tokens uniformly, yet higher-order coordinate digits steer the optimization direction far more effectively than lower-order digits. Furthermore, the teacher's confidence varies across tokens, and treating all tokens equally propagates unreliable signals from low-confidence positions, leading to sub-optimal gradients.

To address these issues, we propose GUI-SD (**GUI** Grounding via **S**elf-**D**istillation), an OPSD framework tailored for GUI grounding, which combines visually enriched privileged context with an entropy-guided loss to deliver rich token-level supervision for precise coordinate generation. Specifically, GUI-SD builds the teacher's privileged context by highlighting the ground-truth region with a bounding box and applying a Gaussian soft mask that gradually fades the surrounding areas. Paired with an instructional hint, this visual prompt delivers informative yet constrained prior knowledge, guiding the teacher to the target without leaking the exact coordinates. Furthermore, GUI-SD introduces entropy-guided distillation, an adaptive objective that replaces uniform token weighting with targeted supervision. It prioritizes higher-order coordinate digits that dominate grounding accuracy while amplifying supervision from confident teacher predictions.

Extensive experiments across six representative grounding benchmarks (ScreenSpot-v2 [42], ScreenSpot-Pro [18], UI-Vision [26], MMBench GUI L2 [39], OSWorld-G[43], and OSWorld-G-Refine [43]) demonstrate that GUI-SD substantially outperforms GRPO-based methods [21, 34, 48] and naive OPSD [30] in both accuracy and training efficiency. Detailed ablation studies further validate that visually enriched privileged context provides effective teacher guidance, while entropy-guided distillation concentrates optimization on the most impactful coordinate tokens.

Our main contributions are summarized as follows:

- To the best of our knowledge, we present the first exploration of the OPSD framework in the GUI grounding domain, offering an appealing alternative to GRPO-based methods that suffer from expensive multiple rollouts and sparse signals on hard samples.
- We propose GUI-SD, which integrates visually grounded teacher guidance with entropy-aware distillation, enabling rich and reliable supervision that concentrates optimization on the most impactful coordinate tokens.
- Extensive experiments verify the effectiveness of GUI-SD over naive OPSD and GRPO-based methods across six representative GUI grounding benchmarks, demonstrating significant improvements in both accuracy and training efficiency, establishing OPSD as a promising paradigm for future GUI grounding research.

2 Preliminary

OPD. While GRPO-type reinforcement learning has driven significant progress in GUI grounding, its sparse sequence-level rewards provide no dense token-level guidance, offer little or even zero feedback on difficult samples, and require heavy online sampling [3]. On-Policy Distillation (OPD) [31] offers an alternative paradigm, where a separate, typically larger, teacher model $\pi_{\hat{\theta}}$ provides token-level supervision along the student's sampled trajectories. By distilling the teacher's output distribution at each decoding step, OPD delivers continuous learning signals that enable more sample-efficient training and meaningful gradient updates even for samples that would otherwise receive no reward.

OPSD. To remove the dependence on a separate teacher, On-Policy Self-Distillation (OPSD) [55] deploys the same model π_θ as both teacher and student, with the two roles operating under asymmetric contexts. Specifically, the teacher is granted access to privileged information r (e.g., ground-truth answers [30] or verified reasoning traces [28]) that is unavailable to the student, yielding more informative token-level distributions along the student's sampled trajectories. Formally, given the sample (x, r), where x denotes the input query and r the privileged context, the student generates an on-policy trajectory under x. Meanwhile, the teacher, conditioned on both (x, r), produces step-wise target distributions along the same trajectory. Training then minimizes the per-token divergence between the student and teacher distributions at each decoding step:

$$\begin{aligned} \text{Student:} \quad & P_S(y_t) \triangleq \pi_\theta(y_t \mid x,\, y_{<t}), \\ \text{Teacher:} \quad & P_T(y_t) \triangleq \pi_\theta(y_t \mid x,\, r,\, y_{<t}), \\ \mathcal{L}(\theta) = \mathbb{E}_{y\sim P_S} & \left[\frac{1}{|y|} \sum_{t=1}^{|y|} D_{\mathrm{KL}}\big(P_S(y_t) \,\big\|\, P_T(y_t)\big) \right], \end{aligned} \tag{1}$$

where D_{KL} measures the KL divergence, $y_{<t}$ denotes the generated trajectory up to step t, y_t denotes the token generated at step t, and $|y|$ is the total length of the trajectory.

3 Empirical Analysis of OPSD for GUI Grounding

To understand the failure of naive OPSD in GUI grounding, we analyze its teacher supervision signal through two complementary entropy-based views. Following prior distillation studies [10, 13, 16, 33, 37], we consider: (1) sample-level entropy, which assesses whether the overall teacher distribution remains informative or collapses toward near-one-hot targets [10, 16, 37]; and (2) token-level entropy, which evaluates the reliability of supervision across coordinate digits [13, 33].

3.1 Distillation-to-SFT Collapse at Sample-level

Prior OPSD methods provide the teacher with privileged information in textual form, such as reference solutions, verifier signals, or environment feedback. Following this design, a naive adaptation to GUI grounding feeds the ground-truth coordinate directly into the teacher input as text. We evaluate the

Table 1: Comparison between naive OPSD teacher supervisory signal and SFT signal. SSP Acc denotes model accuracy on ScreenSpot-Pro.

	Avg Entropy	Avg Top-1 Prob	SSP Acc
SFT Signal	0.00	1.00	54.1
Teacher Signal (Naive OPSD)	0.17	0.82	55.6
Teacher Signal (GUI-SD)	0.50	0.59	60.7

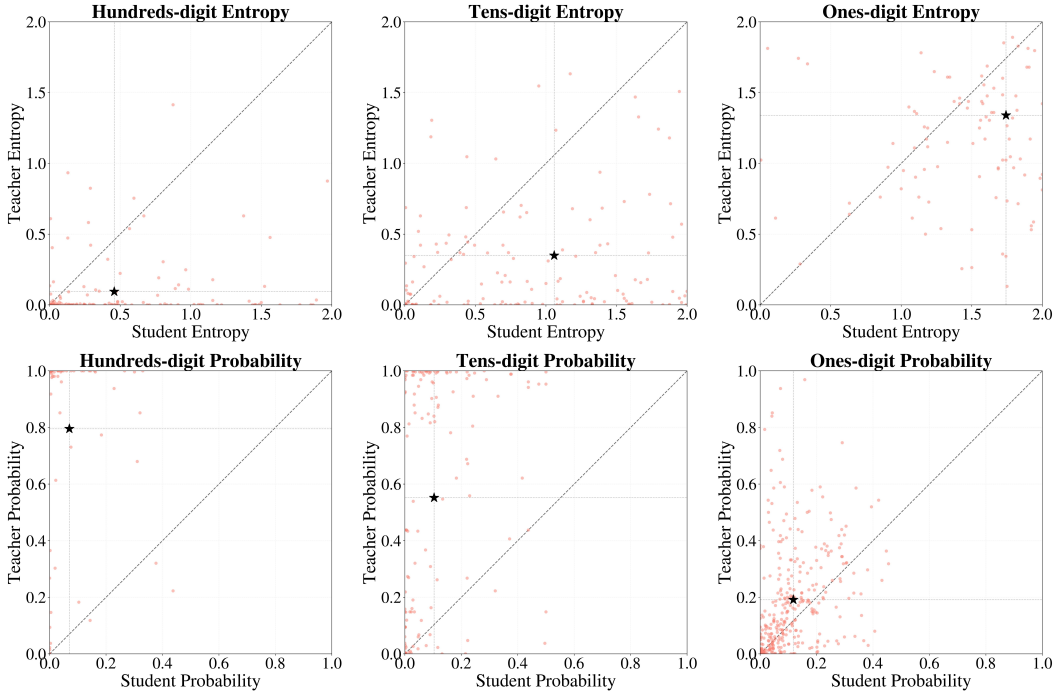


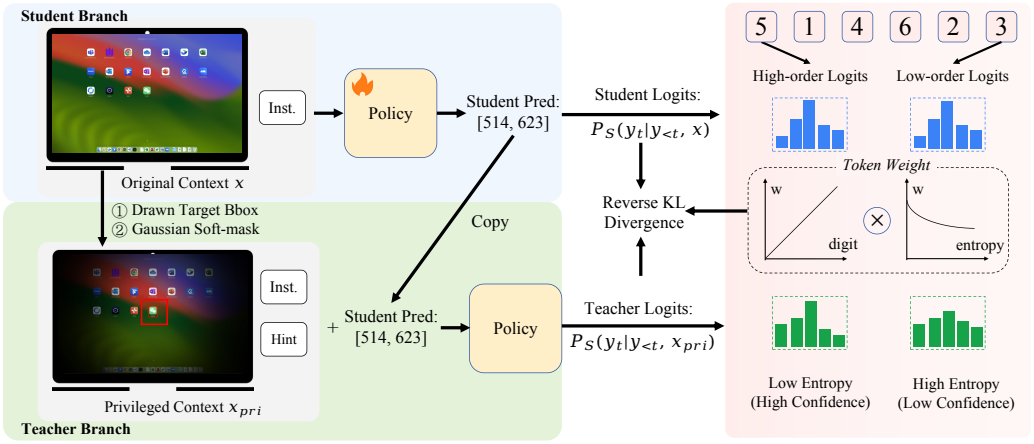
Figure 2: Per-token analysis of teacher and student predictions on incorrectly predicted tokens across digit positions (hundreds, tens, units). Top: teacher vs. student prediction entropy. Bottom: teacher vs. student ground-truth probability. Stars denote the mean teacher and student values for each digit position.

resulting teacher signal at the sample level using two statistics over coordinate digits: average entropy and average top-1 probability. As shown in Table 1, the naive OPSD teacher produces a nearly deterministic distribution, with an average entropy of only 0.17 and an average top-1 probability of 0.82. Prior distillation studies [10, 16] suggest that such low-entropy targets behave similarly to hard labels, causing divergence minimization to degenerate toward standard cross-entropy training. This interpretation is consistent with downstream performance: naive OPSD improves over SFT by only 1.5% on ScreenSpot-Pro (55.6 vs. 54.1), indicating that textual privilege yields little benefit beyond hard-label supervision. In contrast, the optimized privileged context in GUI-SD produces a more informative teacher signal, reaching 60.7% on ScreenSpot-Pro, a 6.6% improvement over SFT, demonstrating that a well-designed privilege context can unlock substantial gains beyond hard-label supervision.

Finding 1: In GUI grounding, textual privileged information drives the teacher distribution toward near-zero entropy, largely collapsing distillation into near-SFT behavior. We term this failure mode Distillation-to-SFT Collapse.

3.2 Indiscriminate Optimization at Token-level

A second question concerns the token level: under a well-designed privileged context, is the teacher's supervision equally reliable across coordinate digits? To investigate this, we analyze incorrectly predicted tokens at each digit position, where teacher guidance is most needed.



(a). Teacher Privileged Guidance *(b). Entropy-Guided Optimization*

Figure 3: Overview of the GUI-SD framework. (a) The teacher branch receives a privileged context x_{pri}, which augments the student's original input x with a target bounding box, a Gaussian soft mask, and a hint prompt, while the student branch operates on the original context only. (b) GUI-SD trains the student with a weighted reverse-KL objective between teacher and student token distributions, where the token weight $w(t)$ prioritizes high-order tokens via positional credit assignment and filters unreliable supervision via entropy-gated supervision.

As shown in Figure 2 (Top), the teacher generally exhibits lower entropy than the student at each digit position, indicating that its token-level preferences are sharper and thus more likely to be amplified during distillation, even on positions where the student predicts incorrectly. Notably, entropy increases from hundreds to tens to units for both teacher and student, suggesting that supervision becomes progressively less certain on lower-order digits. This pattern is echoed in Figure 2 (Bottom), where the teacher assigns higher probability to the ground-truth token than the student at every digit position, with the largest gap observed at the hundreds digit.

Finding 2: Teacher supervision in GUI grounding is inherently position-dependent: higher-order digits carry stronger and more reliable signals than lower-order digits. A uniform reverse-KL objective ignores this heterogeneity and therefore not only misallocates optimization budgets, but also risks amplifying erroneous token preferences on more uncertain digit positions. We term this failure mode Indiscriminate Optimization.

4 Method

Building on the above analysis, we propose GUI-SD as a targeted solution to the two failure modes of naive OPSD in GUI grounding. As illustrated in Figure 3, GUI-SD comprises two complementary components. (a) **Visual Privileged Guidance** replaces textual privilege with a visually enriched privileged context, enabling the teacher to maintain informative soft distributions rather than collapsing toward near-one-hot targets. (b) **Entropy-Guided Optimization** replaces uniform reverse KL with adaptive token weighting, so that optimization emphasizes high-value coordinate digits while downweighting uncertain supervision.

4.1 Visual Privileged Guidance

Finding 1 shows that textual privileged information collapses the teacher distribution toward near-one-hot targets, reducing distillation to near-SFT behavior. To address this issue, GUI-SD replaces textual coordinate privilege with visually grounded privileged context, allowing the teacher to receive target-aware guidance without direct access to the exact coordinate.

Specifically, as shown in Figure 3(a), the teacher input is augmented with a red bounding box over the ground-truth region and a lightweight hint prompt, e.g., "Hint: The answer is located within the red rectangle." This combination supplies informative yet constrained prior knowledge, guiding the teacher toward the target while preserving non-degenerate soft supervision. In addition, to improve

localization in high-resolution GUI scenes with dense layouts, we apply a Gaussian soft mask that progressively attenuates image regions farther from the annotated target, effectively creating a zoom-in effect around the privileged region. The modulation factor for each pixel is defined as:

$$\alpha(x, y) = \exp\left(-\frac{d^2}{2\sigma^2}\right), \tag{2}$$

where d denotes the distance from pixel (x, y) to the nearest edge of the ground-truth bounding box (with $d = 0$ inside the box), and σ is scaled according to the target size with a minimum floor to prevent over-masking small objects. This design preserves full visibility of the target region while smoothly suppressing irrelevant surrounding content.

4.2 Entropy-Guided Optimization

In GUI-SD, the privileged information r in the general OPSD formulation (eq. (1)) is instantiated as the visually enriched context x_{pri}. Conditioned on x_{pri}, the teacher produces step-wise target distributions to provide supervisory signals along the student trajectory.

Finding 2 shows that uniform per-token distillation is suboptimal for GUI grounding: coordinate digits differ substantially in both positional importance and supervision reliability. GUI-SD therefore replaces the uniform objective with an entropy-guided weighted reverse-KL loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{y\sim P_S}\left[\frac{1}{|y|}\sum_{t=1}^{|y|} w(t)\cdot D_{\mathrm{KL}}\big(P_S(y_t)\,\big\|\,P_T(y_t)\big)\right], \tag{3}$$

where

$$w(t) = w^{\text{pos}}(t)\cdot w^{\text{ent}}(t). \tag{4}$$

Positional Credit Assignment. The first factor, $w^{\text{pos}}(t)$, captures the positional significance of coordinate digits. In decimal coordinate prediction, an error in a higher-order digit induces a much larger spatial deviation than an error in a lower-order digit. For example, an incorrect hundreds digit may introduce roughly 100 pixels of error, whereas an incorrect units digit affects only about 1 pixel. To reflect this positional asymmetry, we assign decaying weights from the most significant digit exponentially to the least significant digit:

$$w^{\text{pos}}(t) = \alpha \cdot k_t, \tag{5}$$

where t is the token index in the generated sequence, $k_t \in \{1, 2, 3, 4\}$ denotes the digit position of token t (1 for units, 2 for tens, 3 for hundreds, 4 for thousands), and $\alpha > 0$ is a scaling factor. For non-numeric tokens, we set $w^{\text{pos}}(t) = 1$.

Entropy-Gated Supervision. The second factor, $w^{\text{ent}}(t)$, captures the reliability of teacher supervision. Since the teacher's confidence varies across tokens, uniformly distilling all positions may over-amplify uncertain teacher preferences and propagate unreliable gradients. We therefore use the entropy of the teacher distribution to modulate supervision strength:

$$w^{\text{ent}}(t) = \exp\left(-\frac{H(p_T(\cdot \mid x_{pri}, y_{<t}))}{\tau}\right), \tag{6}$$

where $H(\cdot)$ denotes the per-token entropy of the teacher distribution and τ is a scaling factor. Tokens with low entropy receive stronger supervision, whereas uncertain predictions are automatically down-weighted.

5 Experiment

5.1 Experimental Setup

We conduct experiments on top of Qwen3-VL-Instruct-8B [1], using approximately 7K samples drawn from the ScaleCUA GUI datasets [22] for training. We compare GUI-SD against the following baselines. **GRPO-Binary** is a standard RLVR method [21] that uses a sparse reward, yielding 1 when

Table 2: GUI grounding accuracy on six benchmarks including ScreenSpot-V2 (SS2), ScreenSpot-Pro (SSP), MMBenchGUI (MMG), UI-Vision (UIV), OSWorld-G (OSW-G), and OSWorld-G_R (OSW-GR). Bold indicates the best results. Training time is measured in hours per epoch on 8×A800-80G. The detailed experimental results on each benchmark are in the Section C.1.

Method	Time/epoch	SSP	SS2	UIV	OSW-G	OSW-GR	MMG	Avg.
UI-TARS-7B [27]	-	35.7	91.6	17.6	-	-	-	-
GTA1-7B [47]	-	50.1	92.4	-	55.1	67.7	-	-
GUI-Actor-7B [41]	-	44.6	89.5	-	-	-	-	-
TongUI-7B [52]	-	24.7	88.7	18.0	-	-	-	-
JEDI-7B [43]	-	39.5	91.7	25.2	-	-	-	-
InfiGUI-R1-7B [21]	-	35.7	-	-	-	-	-	-
SE-GUI-7B [48]	-	47.3	90.3	-	-	-	-	-
LPO-7B [35]	-	-	90.5	-	-	-	-	-
GUI-G^2-7B [34]	-	47.5	93.3	-	-	-	-	-
HyperClick-7B [53]	-	48.2	93.7	25.7	-	-	79.6	-
UI-Ins-7B [3]	-	52.2	94.0	-	-	-	83.1	-
ZoomUI-7B [23]	-	52.8	-	27.1	54.2	-	72.8	-
ZwZ-8B [40]	-	56.8	-	-	60.0	69.0	-	-
MolmoWeb-Ground-8B [9]	-	-	91.8	-	-	-	-	-
Propose-then-Critic-8B [38]	-	58.7	91.3	28.9	59.6	-	78.4	-
Qwen3-VL-Instruct [1]	-	53.6	93.2	25.2	58.7	67.4	83.0	63.5
+ *GRPO-Binary* [21]	16.9	56.8	94.6	27.6	61.2	68.6	84.3	$65.5_{+2.0}$
+ *GRPO-Distance* [48]	16.7	56.6	93.8	27.5	62.1	69.9	83.3	$65.5_{+2.0}$
+ *GRPO-Gaussian* [34]	16.8	57.4	94.0	28.2	61.9	70.0	83.7	$65.9_{+2.4}$
+ *Ours*	**4.2**	**60.7**	**95.1**	**33.3**	**64.0**	**70.9**	**86.7**	$\mathbf{68.4}_{+4.9}$

the prediction falls inside the target bounding box and 0 otherwise. **GRPO-Distance** computes a dense reward based on the normalized distance between the click point and the center point of the target bounding box, following SE-GUI [49]. **GRPO-Gaussian** models GUI elements as continuous Gaussian distributions to provide dense reward signals, following GUI-G^2 [34]. Details of the training setting of GUI-SD are provided in Appendix B.

We comprehensively evaluate GUI-SD on six representative GUI grounding benchmarks: ScreenSpot-v2 [42], ScreenSpot-Pro [18], UI-Vision [26], MMBench GUI L2 [39], OSWorld-G [43], and OSWorld-G-Refine [43]. Together, these benchmarks cover diverse application scenarios, hierarchical instruction following, and cross-platform generalization across different operating systems. More details on each benchmark are provided in Appendix A.

5.2 Main Results

Comparisons with Baselines. Table 2 presents the evaluation results across six representative GUI grounding benchmarks. GUI-SD achieves the highest average accuracy while demonstrating superior training efficiency compared to existing GRPO-based methods. We attribute this superiority to dense token-level credit assignment and requiring only a single rollout of the policy model instead of multiple. The former provides fine-grained supervision at every decoding step, particularly beneficial on ScreenSpot-Pro and UI-Vision, where small targets on high-resolution screenshots and diverse desktop layouts across 83 applications both demand precise spatial reasoning, achieving gains of +3.3% and +5.1% over GRPO-Gaussian, respectively. The latter substantially reduces training cost, achieving approximately 4× faster training per epoch compared to GRPO-based methods.

Comparisons with SOTA Methods. GUI-SD surpasses existing GUI grounding methods in average accuracy across six representative benchmarks. On ScreenSpot-Pro, GUI-SD achieves 60.7%, outperforming Propose-then-Critic [38], which relies on test-time scaling with substantial inference overhead. On OSWorld-G-Refine, GUI-SD reaches 70.9%, surpassing ZwZ [40], which depends on large teacher models (e.g., Qwen3-VL-235B) for distillation, requiring significantly more computational resources. Notably, GUI-SD achieves these improvements through self-distillation

Table 3: Ablation studies on teacher privileged context. We evaluate student performance on ScreenSpot-Pro (SSP) under varying teacher guidance settings, along with teacher supervised signal quality: sample accuracy (Acc), sample-averaged entropy (Ent.), and sample-averaged top-1 probability (Top-1). Δ OPSD denotes the performance difference relative to Naive OPSD (Row 2). "Orig." denotes Original, "Inst." is Instruction, and "Gauss." is Gaussian.

Teacher's Guidance Setting		**Student**		**Teacher Signal**		
Visual Context	**Text Context**	**SSP**	**Δ OPSD**	**Acc**	**Ent.**	**Top-1**
Orig. Image	Inst.	53.0	-2.6	52.0	0.59	0.53
Orig. Image	Inst. + Text BBox	55.6	0	93.0	0.17	0.82
Orig. Image + Drawn BBox	Inst. + Drawn Hint	59.9	+4.3	89.8	0.53	0.61
Gauss. Zoom + Drawn BBox	Inst. + Drawn Hint	**60.7**	**+5.1**	**99.6**	**0.50**	**0.59**

Table 4: Ablation study of entropy-guided distillation components. We report ScreenSpot-Pro overall accuracy, hundreds-digit accuracy, and performance on the hard subset, which consists of samples that the base model fails to predict correctly across all 8 rollouts. Entropy-gated and Positional Credit denote entropy-gated supervision and positional credit assignment, respectively.

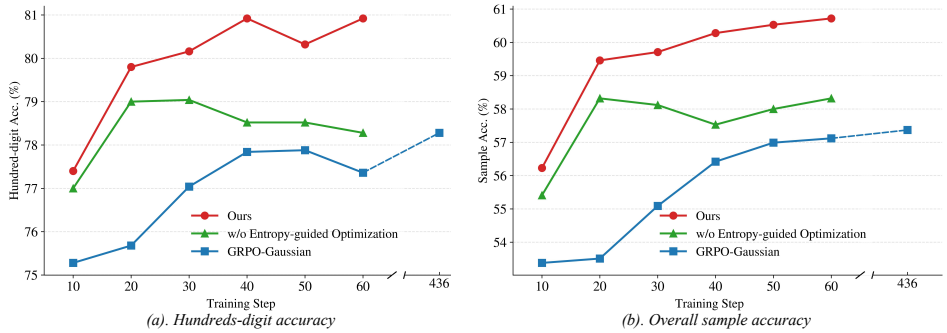
Entropy-gated	**Positional Credit**	**Screenspot-Pro**	**Hundreds-digit Accuracy**	**Hard Subset**
✗	✗	58.3	77.0	17.5
✗	✓	59.6	78.7	19.2
✓	✗	59.2	78.0	19.9
✓	✓	**60.7**	**79.7**	**21.1**

without relying on test-time scaling or external large-scale models, demonstrating that dense token-level supervision from a well-designed privileged context can be more effective.

5.3 Ablation Studies

Effectiveness of Teacher Visual Context. As shown in Table 3, we evaluate student performance on ScreenSpot-Pro under varying teacher guidance settings, along with teacher signal quality (sample accuracy, entropy, and top-1 probability). In Row 1, the teacher and student receive identical inputs without any privilege, yielding the weakest student score (53.0%), confirming that asymmetric context is critical for effective self-distillation. Row 2 (Naive OPSD) appends the ground-truth coordinate as text to the teacher's input, reaching 55.6%. However, although teacher accuracy reaches 93.0%, the entropy drops to merely 0.17, indicating a near-collapsed distribution that reduces soft-label distillation to hard-label SFT. To address this, Row 3 delivers ground-truth information through the visual channel — drawing a bounding box on the teacher's input image with an instructional hint. This forces the teacher to reason over the image rather than copying the answer, raising entropy to 0.53 and producing a substantially softer distribution that enables richer gradient signals, lifting the student to 59.9% (+4.3%). Row 4 further introduces a Gaussian soft-mask zoom-in that suppresses surrounding content, pushing teacher accuracy to 99.6% while maintaining healthy entropy (0.50), achieving the best student performance of 60.7% (+5.1% over Naive OPSD).

Effectiveness of Entropy-guided Optimization. As shown in Table 4, we ablate the individual components of entropy-guided distillation and report performance on ScreenSpot-Pro, its corresponding hard subset, and hundreds-digit accuracy. The first row serves as the baseline optimized solely with the standard reverse KL loss, yielding the lowest performance across all metrics. Introducing the digit-position weighting (Row 2) directly improves the hundreds-digit accuracy from 77.0% to 78.7%, which consequently lifts the overall ScreenSpot-Pro score to 59.6%, confirming that the targeted optimization of leading digits is highly effective for GUI grounding. Incorporating only the entropy-guided weighting (Row 3) steadily improves the overall score to 59.2% and yields substantial gains on the hard subset (from 17.5% to 19.9%). Finally, combining both components (Row 4) yields complementary gains, achieving the best results across the board: 60.7% on ScreenSpot-Pro, 79.7% on hundreds-digit accuracy, and a substantial jump to 21.1% on the hard subset.



(a). Hundreds-digit accuracy *(b). Overall sample accuracy*

Figure 4: Training dynamics of GUI-SD, Standard Reverse KL, and GRPO-Gaussian over optimization steps within a single epoch. (a) Hundreds-digit accuracy. (b) Overall sample accuracy.

5.4 Training Dynamics

Figure 4 illustrates the training dynamics over the optimization steps within a single epoch, comparing GUI-SD against two baselines: GRPO-Gaussian and an ablated variant without entropy-guided optimization. As shown in Figure 4(a), GUI-SD achieves significantly higher hundreds-digit accuracy than both GRPO-Gaussian and the ablated variant, which we attribute to positional credit assignment that concentrates optimization on leading digits and entropy-gated supervision that amplifies high-confidence teacher signals. As shown in Figure 4(b), this improved precision on leading digits translates directly into superior overall sample accuracy. Notably, GUI-SD reaches higher performance in substantially fewer training steps than GRPO-Gaussian, validating the efficiency of dense token-level supervision over sequence-level rewards for GUI grounding.

6 Related Work

6.1 On-Policy Self Distillation

OPD has recently emerged as an effective on-policy training paradigm for delivering rich, token-level feedback [15, 19, 31]. Specifically, the student first generates a rollout, which is subsequently fed into a stronger teacher model to provide step-by-step guidance. OPSD eliminates the need for an external teacher by having the same model serve both roles, conditioned solely on different input contexts, such as reference solutions, verifier signals, and environment feedback. A representative work, SDPO [12], exemplifies this by casting the feedback-conditioned model as a self-teacher, distilling its enriched next-token predictions back into the student policy. RLVR [46] extends this by leveraging self-distillation to obtain token-level supervision for fine-grained update magnitudes, concurrently deriving reliable update directions from environmental feedback. However, previous explorations of OPSD are predominantly confined to the natural language domain. When directly applied to visual tasks, especially GUI grounding, OPSD encounters critical issues such as distillation-to-SFT collapse, as detailed in Section 3.

6.2 GUI Grounding via Verifiable Reinforcement Learning

Reinforcement learning with verifiable rewards (RLVR) methods, such as GRPO, have become an effective paradigm for post-training reasoning models by enabling them to autonomously explore solution trajectories under verifiable feedback [8, 17, 45, 56, 57, 58]. Recent efforts have extended this paradigm to GUI grounding [5, 11, 14, 20, 51, 59]. GUI-R1 [25] and UI-R1 [24] adopt a binary reward based on whether the predicted coordinate falls inside the target bounding box. To mitigate binary feedback limitations, GUI-G1 [60] introduces dense rewards with size-based difficulty coefficients, and GUI-G^2 [34] further models GUI elements as continuous Gaussian distributions for precise spatial alignment. Despite these improvements, GRPO-based training remains hindered by expensive multiple rollouts, sparse signals on hard samples, and heavy reliance on manually designed rewards.

7 Conclusion and Limitations

In this paper, we present GUI-SD, the first exploration of on-policy self-distillation for GUI grounding, which integrates visually grounded teacher guidance with entropy-guided optimization to deliver targeted token-level supervision for precise coordinate generation. Extensive evaluations across six benchmarks demonstrate that GUI-SD substantially outperforms both GRPO-based methods [21, 34, 48] and naive OPSD [30, 44, 54] in accuracy while achieving approximately $4\times$ faster training. One limitation is that we have not yet explored scaling to larger models or other model families beyond Qwen3-VL. A promising future direction is extending GUI-SD to long-horizon GUI agent tasks, where multi-step planning and sequential interactions introduce additional challenges.

8 Acknowledgement

Supported by the National Natural Science Foundation of China (Grant NO 62376266 and 62406318), CAAI-Tencent Rhino-Bird Open Research Fund, and CAAI-Ant Group Research Fund.

References

[1] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[2] Chen Chen, Jiawei Shao, Dakuan Lu, Haoyi Hu, Xiangcheng Liu, Hantao Yao, and Wu Liu. Gui-eyes: Tool-augmented perception for visual grounding in gui agents. *arXiv preprint arXiv:2601.09770*, 2026.

[3] Liangyu Chen, Hanzhang Zhou, Chenglin Cai, Jianan Zhang, Panrong Tong, Quyu Kong, Xu Zhang, Chen Liu, Yuqi Liu, Wenxuan Wang, et al. Ui-ins: Enhancing gui grounding with multi-perspective instruction-as-reasoning. *arXiv preprint arXiv:2510.20286*, 2025.

[4] Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9313–9332, 2024.

[5] Sicheng Fan, Qingyun Shi, Shengze Xu, Shengbo Cai, Tieyong Zeng, Li Ling, Yanyi Shang, and Dehan Kong. Webfactory: Automated compression of foundational language intelligence into grounded web agents. *arXiv preprint arXiv:2603.05044*, 2026.

[6] Yue Fan, Handong Zhao, Ruiyi Zhang, Yu Shen, Xin Eric Wang, and Gang Wu. Gui-bee: Align gui action grounding to novel environments via autonomous exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33249–33266, 2025.

[7] 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.

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

[9] Tanmay Gupta, Piper Wolters, Zixian Ma, Peter Sushko, Rock Yuren Pang, Diego Llanes, Yue Yang, Taira Anderson, Boyuan Zheng, Zhongzheng Ren, et al. Molmoweb: Open visual web agent and open data for the open web. *arXiv preprint arXiv:2604.08516*, 2026.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[11] Kun Huang, Weikai Xu, Yuxuan Liu, Quandong Wang, Pengzhi Gao, Wei Liu, Jian Luan, Bin Wang, and Bo An. Mobileipl: Enhancing mobile agents thinking process via iterative preference learning. *arXiv preprint arXiv:2505.12299*, 2025.

[12] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

[13] Seongryong Jung, Suwan Yoon, DongGeon Kim, and Hwanhee Lee. Todi: Token-wise distillation via fine-grained divergence control. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8089–8102, 2025.

[14] Weitai Kang, Bin Lei, Gaowen Liu, Caiwen Ding, and Yan Yan. Guirlvg: Incentivize gui visual grounding via empirical exploration on reinforcement learning. *arXiv preprint arXiv:2508.04389*, 2025.

[15] Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dohyung Kim, Jiwon Jeon, Dongsheng Li, and Yuqing Yang. Why does self-distillation (sometimes) degrade the reasoning capability of llms? *arXiv preprint arXiv:2603.24472*, 2026.

[16] Jeonghyun Kim, SooKyung Kim, Richeng Xuan, and Hyunsoo Cho. Trust the uncertain teacher: distilling dark knowledge via calibrated uncertainty. *arXiv preprint arXiv:2602.12687*, 2026.

[17] Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025.

[18] Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8778–8786, 2025.

[19] Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huan-ang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026.

[20] Yichao Liu, Huawen Shen, Liu Yu, Shiyu Liu, Zeyu Chen, and Yu Zhou. Drs-gui: Dynamic region search for training-free gui grounding. *arXiv preprint arXiv:2605.15542*, 2026.

[21] 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*, 2025.

[22] Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*, 2025.

[23] Ziwei Liu, Tao Feng, Borui Kang, Yanbing Yang, and Jun Luo. Zoom to essence: Trainless gui grounding by inferring upon interface elements. *arXiv preprint arXiv:2603.14448*, 2026.

[24] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 17608–17616, 2026.

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

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

[27] 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.

[28] Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

[29] 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.

[30] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

[31] Mingyang Song and Mao Zheng. A survey of on-policy distillation for large language models. *arXiv preprint arXiv:2604.00626*, 2026.

[32] Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

[33] Chi-Ping Su, Ching-Hsun Tseng, Bin Pu, Lei Zhao, Jiewen Yang, Zhuangzhuang Chen, and Shin-Jye Lee. Ea-kd: Entropy-based adaptive knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 731–740, 2025.

[34] Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g^2: Gaussian reward modeling for gui grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 33214–33222, 2026.

[35] Jiaqi Tang, Yu Xia, Yi-Feng Wu, Yuwei Hu, Yuhui Chen, Qing-Guo Chen, Xiaogang Xu, Xiangyu Wu, Hao Lu, Yanqing Ma, et al. Lpo: Towards accurate gui agent interaction via location preference optimization. *arXiv preprint arXiv:2506.09373*, 2025.

[36] Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, et al. Ui-venus-1.5 technical report. *arXiv preprint arXiv:2602.09082*, 2026.

[37] Hao Wang, Hao Gu, Hongming Piao, Kaixiong Gong, Yuxiao Ye, Xiangyu Yue, Sirui Han, Yike Guo, and Dapeng Wu. Learning while staying curious: Entropy-preserving supervised fine-tuning via adaptive self-distillation for large reasoning models. *arXiv preprint arXiv:2602.02244*, 2026.

[38] Wenkai Wang, Xiyun Li, Hongcan Guo, Wenhao Yu, Tianqing Fang, Haitao Mi, Dong Yu, and Shengyu Zhang. Measure twice, click once: Co-evolving proposer and visual critic via reinforcement learning for gui grounding. *arXiv preprint arXiv:2604.21268*, 2026.

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

[40] Lai Wei, Liangbo He, Jun Lan, Lingzhong Dong, Yutong Cai, Siyuan Li, Huijia Zhu, Weiqiang Wang, Linghe Kong, Yue Wang, et al. Zooming without zooming: Region-to-image distillation for fine-grained multimodal perception. *arXiv preprint arXiv:2602.11858*, 2026.

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

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

[43] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

[44] Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. *arXiv preprint arXiv:2505.13227*, 2025.

[45] Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*, 2025.

[46] Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled rlvr. *arXiv preprint arXiv:2604.03128*, 2026.

[47] Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, et al. Gta1: Gui test-time scaling agent. *arXiv preprint arXiv:2507.05791*, 2025.

[48] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

[49] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

[50] Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

[51] Xiangjian Zeng, Wenjing Li, Qingqiang Wu, and Liang Zhang. Fdc-ground: Improving grpo for gui grounding via exponential rewards and fact-aligned pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 28122–28130, 2026.

[52] Bofei Zhang, Zirui Shang, Zhi Gao, Wang Zhang, Rui Xie, Xiaojian Ma, Tao Yuan, Xinxiao Wu, Song-Chun Zhu, and Qing Li. Tongui: Building generalized gui agents by learning from multimodal web tutorials. *arXiv e-prints*, pages arXiv–2504, 2025.

[53] Shaojie Zhang, Pei Fu, Ruoceng Zhang, Jiahui Yang, Anan Du, Xiuwen Xi, Shaokang Wang, Ying Huang, Bin Qin, Zhenbo Luo, et al. Hyperclick: Advancing reliable gui grounding via uncertainty calibration. *arXiv preprint arXiv:2510.27266*, 2025.

[54] Shaojie Zhang, Ruoceng Zhang, Pei Fu, Shaokang Wang, Jiahui Yang, Xin Du, Shiqi Cui, Bin Qin, Ying Huang, Zhenbo Luo, et al. Btl-ui: Blink-think-link reasoning model for gui agent. *arXiv preprint arXiv:2509.15566*, 2025.

[55] Xinsen Zhang, Zhenkai Ding, Tianjun Pan, Run Yang, Chun Kang, Xue Xiong, and Jingnan Gu. Opsdl: On-policy self-distillation for long-context language models. *arXiv preprint arXiv:2604.17535*, 2026.

[56] Yan Zhang, Gangyan Zeng, Huawen Shen, Daiqing Wu, Yu Zhou, and Can Ma. Track the answer: Extending textvqa from image to video with spatio-temporal clues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 10275–10283, 2025.

[57] Yan Zhang, Gangyan Zeng, Daiqing Wu, Huawen Shen, Binbin Li, Yu Zhou, Can Ma, and Xiaojun Bi. Gather and trace: Rethinking video textvqa from an instance-oriented perspective. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 876–885, 2025.

[58] Yan Zhang, Pei Fu, Daiqing Wu, Huawen Shen, Ruoceng Zhang, Shaojie Zhang, Jiahui Yang, Yu Zhou, Can Ma, Zhenbo Luo, et al. Towards omni-dimensional gui agent navigation with masked trajectory prediction. *arXiv preprint arXiv:2609.25769*, 2026.

[59] Yuan Zhao, Hualei Zhu, Tingyu Jiang, Shen Li, Xiaohang Xu, and Hao Henry Wang. Co-epg: A framework for co-evolution of planning and grounding in autonomous gui agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 36582–36590, 2026.

[60] Yuqi Zhou, Sunhao Dai, Shuai Wang, Kaiwen Zhou, Qinglin Jia, and Jun Xu. Gui-g1: Understanding r1-zero-like training for visual grounding in gui agents. *arXiv preprint arXiv:2505.15810*, 2025.

Appendix

The appendix includes the following aspects:

- Appendix A: Evaluation Benchmarks.
- Appendix B: Training Details.
- Appendix C: Additional Experiments and Ablations.
- Appendix D: Declaration of LLM usage.

A Evaluation Benchmarks

ScreenSpot-v2 [42]. ScreenSpot-v2 is a refined version of the original ScreenSpot benchmark [4], designed to address annotation ambiguities in earlier versions. It covers mobile, desktop, and web platforms, with each sample consisting of a GUI screenshot paired with a natural language instruction and a ground-truth bounding box. ScreenSpot-v2 is widely adopted as a standard benchmark for general-purpose GUI grounding evaluation.

ScreenSpot-Pro [18]. ScreenSpot-Pro focuses on the under-explored challenge of grounding in professional, high-resolution software environments. It comprises 1,581 instructions captured from 23 real-world applications spanning five professional industries, including development tools (e.g., VSCode, PyCharm), creative applications (e.g., Photoshop, Premiere), CAD/engineering software (e.g., AutoCAD, SolidWorks), scientific tools (e.g., MATLAB, Stata), and office software (e.g., Word, Excel), across three operating systems (Windows, macOS, Linux). The central challenge of ScreenSpot-Pro is the extremely small target size: UI elements occupy on average only 0.07% of the high-resolution screenshot area. Dense and complex interface layouts further increase the difficulty of precise localization, making ScreenSpot-Pro one of the most demanding grounding benchmarks available.

UI-Vision [26]. UI-Vision is the largest desktop-centric GUI benchmark to date, spanning 83 open-source desktop applications across six domains: Productivity, Development, Creativity, Education, Browsers, and Entertainment. We evaluate on its Element Grounding benchmark, which contains over 8,200 query-label pairs with high-quality human-annotated bounding boxes. A distinguishing aspect of UI-Vision is its cross-application diversity, requiring GUI agents to generalize across highly varied software interfaces and interaction patterns, exposing limitations in spatial reasoning and professional software understanding.

OSWorld-G and OSWorld-G-Refine [43]. OSWorld-G is a comprehensive GUI grounding benchmark set in the Linux environment, comprising 564 finely annotated samples that cover 32 distinct UI element types. OSWorld-G captures diverse real-world computer-use interactions, requiring software knowledge, layout understanding, and fine-grained operations. The benchmark organizes tasks into four categories: text matching, element recognition, layout understanding, and precise operation. OSWorld-G-Refine is a refined version that rewrites instructions to remove domain-specific knowledge, isolating the model's pure spatial grounding ability from its software understanding.

MMBench GUI L2 [39]. MMBench-GUI is a hierarchical, cross-platform benchmark for evaluating GUI automation agents across six platforms: Windows, macOS, Linux, iOS, Android, and Web. The benchmark is organized into four ascending capability levels: L1-Content Understanding, L2-Element Grounding, L3-Task Automation, and L4-Task Collaboration. In our evaluation, we adopt the L2-Element Grounding level, which assesses the model's ability to localize target GUI elements from natural language instructions. L2 includes both basic and advanced difficulty tiers with diverse instruction styles (e.g., action descriptions, target element descriptions, and refusal cases). Its unique contribution lies in enabling consistent cross-platform comparison under a unified evaluation protocol, revealing how models handle the varying interface designs and visual layouts across different operating systems.

Table 5: Hyperparameter settings for training.

Hyperparameter	Value
Learning Rate	from 2.5e-6 to 0
Per Device Train Batch Size	1
Gradient Accumulation Steps	16
Number of Training Epochs	1
Warmup Ratio	0.05
Maximum Sequence Length	20000
Maximum Completion Length	128
EMA Decay Coefficient	0.95
DeepSpeed (Student)	ZeRO-2
DeepSpeed (Teacher)	ZeRO-3
σ Scale Factor	1.5
Minimum σ Floor	$\sqrt{0.1} \cdot \min(W, H)$

Table 6: Performance comparison on the ScreenSpot-Pro benchmark [18].

Methods	CAD		Development		Creative		Scientific		Office		OS		Avg.
	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
Qwen3-VL-Instruct [1]	58.38	14.06	79.22	24.83	69.70	17.48	76.39	27.27	80.79	35.85	71.03	26.97	53.57
+ *GRPO-Binary* [21]	63.45	20.31	80.52	31.72	70.71	18.88	79.17	31.82	84.18	37.74	72.90	30.34	56.80
+ *GRPO-Distance* [48]	62.44	18.75	82.47	29.66	70.71	19.58	78.47	33.64	83.62	39.62	71.03	30.34	56.61
+ *GRPO-Gaussian* [34]	62.44	21.88	84.42	30.34	71.21	20.28	79.86	34.55	84.18	39.62	71.96	29.21	57.37
+ Ours	**67.01**	**31.25**	**83.77**	**35.86**	**73.74**	**22.38**	**82.64**	**37.27**	**84.75**	**52.83**	**72.90**	**37.08**	**60.72**

B Training Details.

Training Data. Our training data is sourced entirely from the grounding training subset of Scale-CUA [22]. Since the original annotations contain labeling errors and lack instruction diversity, we apply a two-stage data curation pipeline. First, we leverage UI-Venus1.5-8B [36] to filter the dataset, retaining only samples where its prediction agrees with the original annotation, removing noisy labels. Second, we employ Qwen3-VL-8B [1] to rewrite the original instructions into diverse paraphrases, enriching instruction variety. After filtering and rewriting, approximately 7K samples remain for training.

Hyperparameters. The hyper-parameter details for GUI-SD are provided in Table 5

C Additional Experiments and Ablations

C.1 Detailed Benchmark Results

We provide per-category results for each evaluation benchmark. Table 6 reports ScreenSpot-Pro results across six professional domains (CAD, Development, Creative, Scientific, Office, OS), split by Text and Icon targets. Table 7 reports ScreenSpot-v2 results across Mobile, Desktop, and Web platforms. Table 8 reports UI-Vision results across Basic, Functional, and Spatial grounding tasks. Table 9 reports MMBench GUI L2 results across six operating systems (Windows, macOS, Linux, iOS, Android, Web), split by Basic and Advanced difficulty. Table 10 and Table 11 report OSWorld-G and OSWorld-G-Refine results across four task types (Text Matching, Element Recognition, Layout Understanding, Fine-Grained Manipulation). GUI-SD consistently outperforms all GRPO baselines across the majority of sub-categories.

C.2 Ablation on Visual Privilege Design

Beyond the main ablation in Table 3, we further compare different visual privilege designs in Table 12. The first row is the Naive OPSD baseline that appends the target coordinate as text. The second row replaces textual privilege with a standard zoom that masks all regions outside a fixed-size area

Table 7: Performance comparison on the Screenspotv2 benchmark [42].

Methods	Mobile		Desktop		Web		Avg.
	Text	Icon	Text	Icon	Text	Icon	
Qwen3-VL-Instruct [1]	97.93	88.63	96.91	89.29	95.30	87.68	93.16
+ *GRPO-Binary* [21]	99.66	89.57	98.97	89.29	96.15	90.15	94.58
+ *GRPO-Distance* [48]	99.66	88.63	97.94	89.29	96.15	86.70	93.71
+ *GRPO-Gaussian* [34]	99.66	89.10	97.42	89.29	95.73	88.67	93.95
+ Ours	**99.66**	**89.57**	**97.42**	**92.86**	**97.01**	**91.13**	**95.05**

Table 8: Performance comparison on the UI-Vision benchmark [26].

Methods	Basic	Functional	Spatial	Avg.
Qwen3-VL-Instruct [1]	30.53	31.88	14.16	25.19
+ *GRPO-Binary* [21]	34.03	34.59	15.35	27.61
+ *GRPO-Distance* [48]	34.09	33.97	15.45	27.47
+ *GRPO-Gaussian* [34]	35.16	34.76	15.76	28.18
+ Ours	**41.87**	**39.39**	**19.79**	**33.27**

Table 9: Performance comparison on the MMBench-GUI L2 benchmark [39].

Methods	Windows		MacOS		Linux		iOS		Android		Web		Avg.
	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	Bas.	Adv.	
Qwen3-VL-Instruct [1]	89.30	65.07	85.51	70.81	76.96	58.16	95.86	84.24	96.35	85.63	95.48	77.60	82.99
+ *GRPO-Binary* [21]	90.04	67.65	86.67	71.97	76.44	61.73	96.18	86.36	96.35	87.89	95.48	80.52	84.32
+ *GRPO-Distance* [48]	91.88	69.12	83.19	69.36	75.39	58.16	96.18	84.85	96.07	87.61	94.52	78.25	83.27
+ *GRPO-Gaussian* [34]	91.88	68.38	83.19	69.65	75.39	58.67	96.50	86.06	96.07	88.17	95.16	80.52	83.71
+ Ours	**91.14**	**72.06**	**90.72**	**76.30**	**78.01**	**66.33**	**97.13**	**86.67**	**96.91**	**91.27**	**95.81**	**83.44**	**86.65**

Table 10: Performance comparison on the OSWorld-G benchmark [43].

Methods	Text Matching	Element Recognition	Layout Understanding	Fine-Grained Manipulation	Avg.
Qwen3-VL-Instruct [1]	47.37	67.16	66.67	62.12	58.69
+ *GRPO-Binary* [21]	47.37	70.90	70.22	62.88	61.17
+ *GRPO-Distance* [48]	42.11	74.63	70.67	62.88	62.06
+ *GRPO-Gaussian* [34]	42.11	73.13	70.67	63.64	61.88
+ Ours	**52.63**	**75.37**	**73.78**	**63.64**	**64.01**

Table 11: Performance comparison on the OSWorld-G-Refine benchmark [43].

Methods	Text Matching	Element Recognition	Layout Understanding	Fine-Grained Manipulation	Avg.
Qwen3-VL-Instruct [1]	52.63	78.36	79.56	65.15	67.38
+ *GRPO-Binary* [21]	47.37	79.85	79.11	70.45	68.62
+ *GRPO-Distance* [48]	47.37	82.09	80.00	71.97	69.86
+ *GRPO-Gaussian* [34]	52.63	82.09	80.00	71.97	70.04
+ Ours	**52.63**	**82.09**	**83.56**	**69.70**	**70.92**

Table 12: Ablation on visual privilege design. SSP denotes ScreenSpot-Pro accuracy. Δ OPSD denotes the performance difference relative to Naive OPSD.

Teacher's Guidance Setting		Student	
Visual Context	**Text Context**	**SSP**	**Δ OPSD**
Orig. Image	Inst. + Text BBox	55.6	0
Standard Zoom + Drawn BBox	Inst. + Drawn Hint	60.1	+4.5
Adaptive Zoom + Drawn BBox	Inst. + Drawn Hint	60.3	+4.7

Table 13: Ablation on teacher update strategy. q_θ denotes using the current policy as the teacher (updated every step), $q_{\theta_{\text{ref}}}$ denotes using a frozen copy of the initial policy as the teacher, and EMA denotes the exponential moving average teacher with the specified decay coefficient.

Teacher	ScreenSpot-Pro
q_θ	59.4
$q_{\theta_{\text{ref}}}$	59.6
EMA = 0.90	59.8
EMA = 0.95	**60.7**

Table 14: Performance comparison across six representative grounding benchmarks for Qwen3-VL from 2B to 8B. ScreenSpot-Pro (SSP), ScreenSpot-v2 (SS2), UI-Vision (UIV), OSWorld-G (OSW-G), OSWorld-G-Refine (OSW-GR), and MMBench GUI L2 (MMG).

Method	**SSP**	**SS2**	**UIV**	**OSW-G**	**OSW-GR**	**MMG**
Qwen3-VL-Instruct-2B [1]	39.4	86.4	13.3	46.6	62.2	70.3
+ *Ours*	**43.1**	**90.7**	**19.4**	**49.8**	**62.6**	**75.2**
Qwen3-VL-Instruct-4B [1]	56.1	92.7	23.8	62.8	70.9	83.1
+ *Ours*	**60.7**	**94.0**	**31.5**	**63.3**	**71.1**	**85.2**
Qwen3-VL-Instruct-8B [1]	53.6	93.2	25.2	58.7	67.4	83.0
+ *Ours*	**60.7**	**95.1**	**33.3**	**64.0**	**70.9**	**86.7**

centered on the target while drawing a bounding box on it. The third row uses our adaptive zoom with a hard mask that adjusts the visible region proportionally to the ground-truth bounding box size. Both visual privilege variants substantially outperform Naive OPSD (+4.5 and +4.7), confirming that delivering ground-truth information through the visual channel is critical. The adaptive zoom achieves a further gain over standard zoom, as its flexible masking better adapts to varying target sizes across different GUI layouts.

C.3 The Self-teacher Improves during Training

A key design choice in GUI-SD is how the teacher model is maintained during training. Unlike off-policy distillation where the teacher is typically frozen, on policy self-distillation allows the teacher to evolve alongside the student. As shown in Table 13, using the current policy q_θ directly as the teacher yields 59.4%. In this setting, the teacher updates simultaneously with the student at every step, resulting in minimal divergence between their distributions and weakening the distillation signal. Freezing the teacher at initialization ($q_{\theta_{\text{ref}}}$) performs similarly at 59.6%, as the fixed teacher quickly becomes outdated once the student improves beyond its initial capacity. Exponential moving average (EMA) with a decay of 0.95 achieves the best result (60.7%), balancing stability and adaptability by allowing the teacher to gradually absorb the student's improving policy while maintaining a smoother, more reliable distribution. A lower decay of 0.90 updates the teacher more slowly, causing it to lag behind the student's progress, reducing performance to 59.8%.

C.4 Performance Across Model Sizes

As shown in Table 14, GUI-SD consistently improves over the base model across all three scales of Qwen3-VL-Instruct (2B, 4B, and 8B). On ScreenSpot-Pro, GUI-SD achieves gains of +3.7, +4.6, and +7.1 at the 2B, 4B, and 8B scales respectively. Similar improvements are observed across all other benchmarks, confirming that the proposed method is effective regardless of model capacity.

D Declaration of LLM Usage

This paper does not use large language models (LLMs) as any important, original, or non-standard component of the core methodology. During the preparation of this manuscript, LLM-based assistants (e.g., ChatGPT) were used solely for assisting with language polishing and editing of the text.

These tools did not contribute to the research design, experimental implementation, data analysis, or interpretation of results. The authors take full responsibility for the content of this work.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 contributions: the first OPSD framework for GUI grounding, visually enriched privileged context, and entropy-guided distillation. All claims are supported by experiments in Section 5.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: We discuss limitations in Section 7 (Conclusion), noting that we have not yet explored scaling to larger model sizes beyond 8B.

Guidelines:

- The answer [N/A] 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: [N/A]

   Justification: This paper focuses on method design and experimental validation, and does not include formal theoretical results or proofs.

   Guidelines:

   - The answer [N/A] 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: We provide model architecture, training data source, hyperparameters (Appendix), and evaluation benchmarks in Section 5.1. Code and data will be publicly released.

Guidelines:

- The answer [N/A] 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: The code and training data will be publicly released upon publication. The code repository is provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details including base model, dataset, hyperparameters, and hardware are provided in Section 5.1 and the Appendix.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: We do not report error bars as GUI grounding evaluation on fixed benchmarks with deterministic inference produces stable results with minimal variance. Additionally, the high computational cost of repeated training runs on 8×A800-80G GPUs makes multiple runs impractical.

   Guidelines:

   - The answer [N/A] 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 training time per epoch and GPU configuration (8×A800-80G) in Table 2 and the Appendix.

Guidelines:

- The answer [N/A] 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 conforms to it in every respect.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: This work focuses on improving GUI grounding accuracy for autonomous agents. It is foundational research without a direct path to negative societal applications beyond those inherent to GUI automation in general.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released model is a GUI grounding model fine-tuned from a publicly available base model, which does not pose significant risks for misuse.

Guidelines:

- The answer [N/A] 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: All datasets and models used are properly cited. The training data is sourced from ScaleCUA, and all evaluation benchmarks are publicly available with their respective licenses.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: No new assets are released with this submission.

Guidelines:

- The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as any important, original, or non-standard component in our core method. LLM-based tools were used only for language polishing during manuscript preparation. A detailed declaration is provided in the Appendix.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.