# BEYOND COORDINATES: SEMANTIC REWARD FOR GUI AGENT POST-TRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graphical User Interface (GUI) agents have made substantial progress in automating tasks with vision-language models. Reinforcement learning has further improved GUI grounding by optimizing policies with coordinate-based rewards, which provide direct supervision for spatial localization. However, GUI interaction data also contains rich semantic information about the intended operation, target element, and expected outcome, which cannot be fully captured by geometric objectives alone. To exploit this complementary information, we introduce semantic rewards for GUI agent post-training. During reinforcement learning, candidate actions are represented as natural-language descriptions of the intended GUI operations, and a VLM evaluates their semantic consistency with the target interaction. We further propose Self-Reward RFT, where the semantic evaluator is initialized from the policy and periodically updated with the latest policy weights while remaining frozen between updates. This design avoids maintaining a separately trained reward model while providing an adaptive semantic training signal. Extensive experiments across multiple GUI grounding benchmarks demonstrate the effectiveness of semantic supervision across different policy backbones. Using only 6k training samples, our framework with Self-Reward RFT improves MAI-UI-2B from 56.7% to 59.0% on the challenging ScreenSpot-Pro benchmark, achieving strong performance among lightweight GUI models while remaining competitive with several larger models. Further analysis shows that semantic and coordinate-based supervision capture complementary aspects of GUI action correctness, highlighting the value of incorporating semantic feedback into GUI agent post-training.

## 1 INTRODUCTION

GUI agents (Wu et al., 2025; Lu et al., 2025a; Dai et al., 2025) follow natural-language instructions and execute actions directly on smartphone interfaces. Recent vision-language models (VLMs) (Bai et al., 2025a; Wang et al., 2024) have improved GUI agents by replacing handcrafted rules with learned policies that use visual and language inputs. Despite this progress, reliable fine-grained grounding remains difficult in visually dense interfaces, especially when nearby elements have similar appearances or functions.

A central limitation of current RL-based GUI post-training is that reward design is largely based on spatial correctness. Recent work applies reinforcement learning (RL) after supervised fine-tuning (SFT) (Li et al., 2024a; Rawles et al., 2023; Zhou et al., 2025b; Chu et al., 2025; Wang et al., 2025a), where coordinate-based rewards provide a simple and effective signal for learning accurate action locations. However, a GUI interaction contains more than a target coordinate. It also specifies which interface element should be selected, what action should be performed, and what result the interaction is expected to achieve. Much of this semantic information is already available in existing interaction data, but it is not captured by coordinate matching alone. How to effectively use such information during reinforcement learning remains underexplored.

To address this limitation, we introduce semantic reward for GUI agent post-training. During training, candidate actions are represented as natural-language descriptions of the target UI element, action type, and expected effect. A VLM evaluates the semantic consistency between the predicted action and the target interaction and converts this judgment into a reward. This semantic reward complements



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

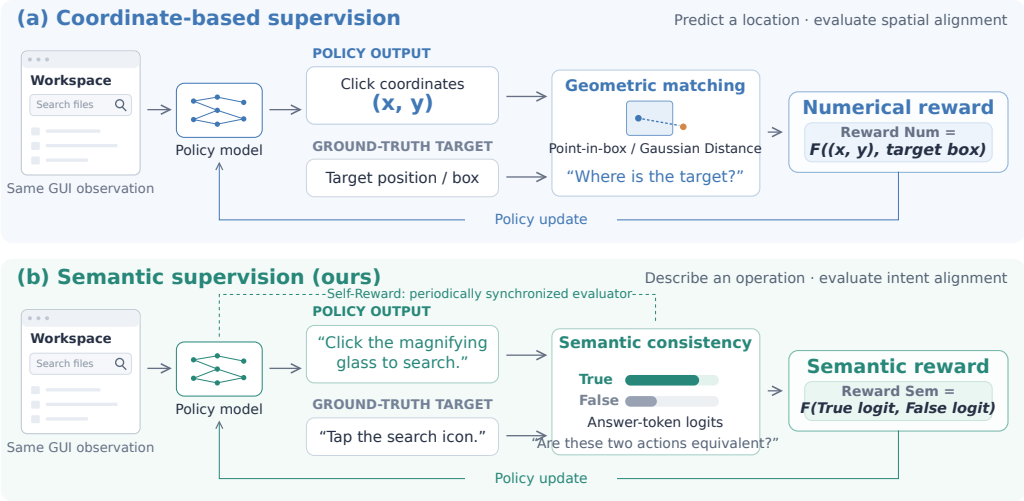


Figure 1: **Comparison of coordinate-based and semantic supervision.** Coordinate-based supervision predicts spatial targets and derives numerical rewards from geometric alignment, whereas our semantic formulation predicts natural-language low-level actions and evaluates their functional consistency with target actions. In Self-Reward RFT, the semantic evaluator is periodically synchronized with the policy while remaining frozen between synchronization points.

coordinate-based supervision by evaluating what action is being performed in addition to where the action occurs, as illustrated in Figure 1. We use this reward for policy optimization with reinforcement learning. To avoid training a separate reward model, we further introduce Self-Reward RFT, where the semantic evaluator is initialized from the policy and periodically updated with the latest policy weights. The evaluator remains frozen between updates, allowing it to follow the evolving policy while keeping the reward function fixed within each optimization interval.

We evaluate the method on multiple GUI grounding benchmarks using two policy backbones, MAI-UI-2B (Zhou et al., 2025a) and Qwen3-VL-2B (Bai et al., 2025a), with only 6k training samples. The method improves both backbones across all evaluated benchmarks. On ScreenSpot-Pro, MAI-UI-2B improves from 56.7% to 59.0%, while Qwen3-VL-2B improves from 43.4% to 45.7%. Ablation studies show that semantic reward provides supervision beyond coordinate-based objectives, while Self-Reward RFT provides further gains without requiring a separately trained reward model. We also find cases where coordinate-based scores and semantic correctness disagree, suggesting that the two signals capture different aspects of GUI action quality.

Our key contributions are summarized as follows:

1. We introduce semantic reward for GUI agent post-training, which evaluates the semantic consistency between predicted and target actions and complements coordinate-based supervision.
2. We propose Self-Reward RFT, where the semantic evaluator is initialized from the policy and periodically updated with the latest policy weights while remaining frozen between updates, avoiding the need for a separately trained reward model.
3. We conduct extensive experiments across 4 GUI grounding benchmarks and two policy backbones. Using only 6k training samples, our method consistently improves grounding performance, while further analysis shows that semantic and coordinate-based supervision capture complementary aspects of GUI action correctness.

## 2 Related Work

### 2.1 GUI Agents

GUI agents represent a distinctive form of embodied intelligence, where an AI system must perceive, interpret, and act within graphical user interfaces in the same manner as a human user. Unlike

API-driven automation pipelines that operate on structured backend signals, GUI agents interact purely through rendered pixels and executable actions such as clicks, scrolls, and text inputs. This setting transforms computer-use into a perception–decision problem grounded entirely in visual context. The emergence of large foundation models, including LLMs and vision-language models, has significantly accelerated progress in this domain (Yang et al., 2024a; Bai et al., 2025b; Guo et al., 2025). Their strong cross-modal reasoning capabilities allow agents to align textual instructions with visual interface elements, enabling more flexible and generalized interaction across applications and platforms. Early systems relied on structured representations such as HTML trees or accessibility APIs (Wen et al., 2023), but recent work demonstrates that directly modeling pixel-based GUIs with high-resolution visual encoders yields superior grounding performance (Gou et al., 2024; Xu et al., 2024; Wu et al., 2024). Representative systems such as AppAgent (Zhang et al., 2023; Li et al., 2024b), UI-TARS (Qin et al., 2025), and AgentS2 (Agashe et al., 2025) further improve interaction fidelity through modular planning, hierarchical reasoning, and generalist-specialist designs.

### 2.2 Reinforcement Learning

RL has become an effective framework for GUI agent post-training, enabling policies to improve beyond supervised demonstrations through reward feedback. Recent work extends RFT from language models (Guo et al., 2025) to multimodal reasoning (Shen et al., 2025; Huang et al., 2025), while GUI-specific methods such as UI-R1 (Lu et al., 2025b) and GUI-R1 (Xia & Luo, 2025) further demonstrate gains in action prediction and grounding. However, existing GUI RL methods still rely primarily on spatially defined rewards. Binary rewards provide sparse success signals, whereas continuous geometric rewards mainly measure coordinate alignment. Although effective for localization, these objectives do not explicitly capture whether the predicted action corresponds to the intended GUI operation or target function. This limitation motivates richer forms of reward modeling that go beyond purely geometric correctness. More broadly, recent self-rewarding methods explore how a model's own judgments can provide supervision for iterative policy improvement. Prior approaches typically use model-derived quality estimates to construct preference data, such as language-model-generated scores or visually calibrated scores, and then optimize the policy through iterative preference learning (Yuan et al., 2024; Zhou et al., 2024). Our setting is different in both the supervision target and the optimization objective. Rather than converting model judgments into preference pairs, we directly evaluate the functional equivalence between predicted and target GUI actions under the current interface state, and use the resulting semantic signal as a reward for GRPO. The semantic evaluator is initialized from the policy and periodically synchronized during training, allowing the model to reuse its own GUI understanding for reward estimation without maintaining a separately trained reward model.

## 3 Methodology

As shown in Figure 2, our method is a Self-Reward RFT framework for semantically grounded optimization of VLM-based GUI agents. We first introduce GRPO, which is used to optimize the policy model. We then describe the semantic reward, where a VLM evaluates the semantic consistency between predicted and target actions rather than relying solely on coordinate-based matching. Finally, we present our Self-Reward RFT framework, in which the semantic reward model is initialized from the policy and periodically synchronized with the latest policy parameters every 50 optimization steps. Between synchronization points, the reward model remains frozen, allowing the semantic evaluator to adapt to the evolving policy while maintaining a stable reward function within each interval.

### 3.1 Preliminaries

GRPO (Shao et al., 2024) offers a computationally efficient alternative to Proximal Policy Optimization (PPO) (Schulman et al., 2017), a widely used actor–critic method. Instead of maintaining a separate critic network for value estimation, GRPO computes normalized group-wise advantages $A_i$ directly from reward scores, thereby removing the value function update and reducing computational cost. The GRPO objective in our framework is defined as follows:

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_i \left(\min(\rho_i A_i,\ \mathrm{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i) - \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}})\right)\right]. \quad (1)$$

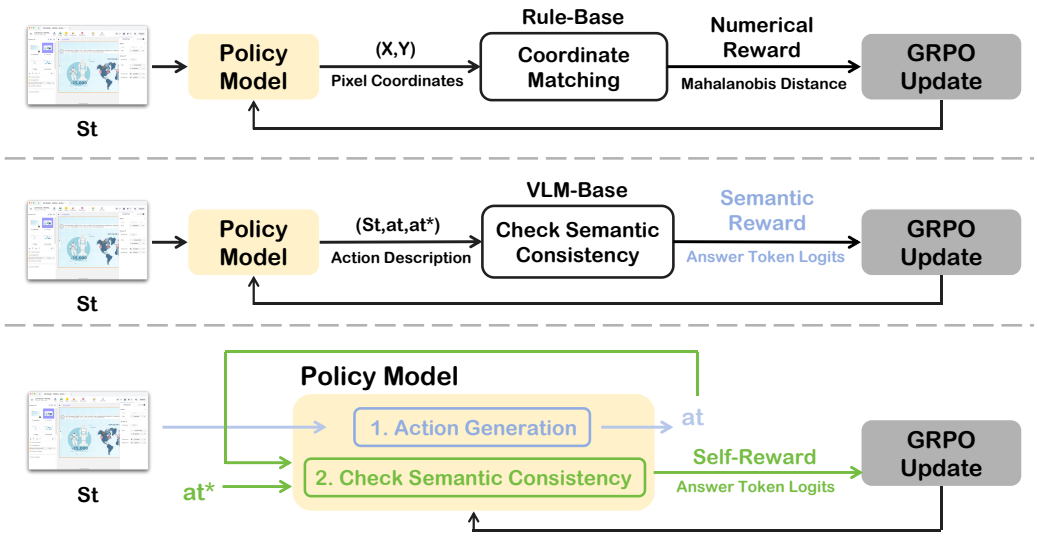


Figure 2: **Self-Reward Reinforcement Fine-Tuning with Semantic Rewards.** We illustrate three reward formulations for GUI agents. **Top:** coordinate-based numerical reward evaluates spatial alignment between predicted and target locations. **Middle:** semantic reward uses a frozen VLM evaluator to assess the functional consistency between predicted and target action descriptions. **Bottom:** our Self-Reward RFT initializes the semantic evaluator from the policy and periodically synchronizes it with the latest policy parameters. The evaluator remains frozen between synchronization points, providing semantic rewards for GRPO updates while avoiding direct reward-function changes after every policy update.

where the importance ratio and relative advantage are computed as:

$$\rho_i = \frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, \qquad A_i = \frac{r_i - \text{mean}(\{r_j\}_{j=1}^G)}{\text{std}(\{r_j\}_{j=1}^G)}. \tag{2}$$

For each query $q$, we sample a group of outputs $\{o_1, o_2, \ldots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$ using high-temperature decoding and compute the relative advantage $A_i$ within the group. A clipped surrogate objective, together with a KL-divergence regularizer toward the reference policy $\pi_{\text{ref}}$, is then used to update the policy parameters.

In our GUI setting, the query $q$ consists of the high-level task instruction, the current GUI observation $S_t$, and the previous interaction history $H_t$. During semantic reinforcement fine-tuning, each sampled output $o_i$ corresponds to a low-level semantic action description $a_t$ expressed in natural language. Given the target semantic action $a_t^*$, the semantic evaluator assesses the functional consistency between $a_t$ and $a_t^*$ under the same GUI observation $S_t$, producing the reward used by GRPO.

The semantic action representation defines the prediction target used during reinforcement fine-tuning and reward computation. Specifically, the policy is optimized to generate a concise natural-language description of the intended next GUI operation rather than directly predicting structured coordinates or tool calls. At evaluation time, the same trained policy is directly prompted to produce benchmark-native coordinate outputs, as detailed in Appendix F.

### 3.2 Semantic Reward Design

We construct a semantic reward using a VLM-based evaluator without training an additional discriminator. Let $S_t$ denote the current GUI observation, $a_t$ the semantic action generated by the policy, and $a_t^*$ the target semantic action. As shown in Figure 2, the evaluator receives the triplet $(S_t, a_t, a_t^*)$ and determines whether $a_t$ and $a_t^*$ describe functionally equivalent operations under the current interface state. The judgment considers the action type, the functional role of the target element, and the input content when applicable, providing semantic supervision complementary to coordinate-based localization.

We define the semantic reward using the margin between the `True` and `False` answer-token logits. Let $z_{\texttt{True}}$ and $z_{\texttt{False}}$ denote the logits of the two candidate answers at the same response position, conditioned on $(S_t, a_t, a_t^*)$. The reward is defined as

$$R(S_t, a_t, a_t^*) = \tanh\left(\frac{z_{\texttt{True}} - z_{\texttt{False}}}{\tau}\right), \tag{3}$$

where $\tau > 0$ is a fixed temperature controlling the scale of the logit margin. We set $\tau = 1.0$ in all experiments and keep it fixed across policy backbones and benchmarks.

The logit margin reflects the evaluator's relative preference for semantic equivalence over non-equivalence and is invariant to a common additive shift of the two logits. The $\tanh$ transformation bounds the reward to $(-1, 1)$ while preserving the sign of this preference. Positive rewards indicate that the evaluator favors functional equivalence, negative rewards indicate non-equivalence, and values near zero correspond to ambiguous judgments. This bounded continuous signal is used for policy optimization.

### 3.3 Self-Reward Reinforcement Fine-Tuning

Let $\pi_\theta$ denote the policy model responsible for generating semantic action descriptions. In a standard semantic-reward setting, the reward model can be implemented as an independent off-the-shelf VLM with fixed parameters, such that reward computation is fully decoupled from policy optimization. In our Self-Reward formulation, the semantic evaluator is instead initialized from the policy model and periodically synchronized with the latest policy parameters. This allows the reward model to inherit the GUI understanding and action semantics learned by the policy without requiring a separately trained reward network.

As shown in Figure 2, given the high-level instruction and current GUI state $S_t$, the policy first generates a semantic action $a_t$. The reward model then performs the *Check Semantic Consistency* task described above by evaluating the triplet $(S_t, a_t, a_t^*)$, where $a_t^*$ denotes the target semantic action. The resulting True/False answer-token logits are used to compute the semantic reward $R(S_t, a_t, a_t^*)$. During training, the reward model is synchronized with the policy every 50 optimization steps and remains frozen between two synchronization points. Therefore, policy updates within each interval do not directly modify the reward function used to evaluate the generated actions. At the next synchronization point, the evaluator is refreshed using the latest policy parameters. This periodically synchronized design refreshes the semantic evaluator using the latest policy parameters while maintaining a fixed reward function within each optimization interval.

Compared with using a separate external reward model, Self-Reward RFT avoids the need to maintain and optimize an additional reward network. It directly reuses the multimodal reasoning capability of the policy model for semantic evaluation, while the periodic synchronization mechanism decouples reward estimation from individual policy updates.

## 4 Experiments

For RFT, we use MAI-UI-2B (Zhou et al., 2025a) and Qwen3-VL-2B (Bai et al., 2025a) as policy backbones and train each model for three epochs. We optimize the policy using AdamW with a learning rate of $1 \times 10^{-6}$ and a global batch size of 96. During GRPO rollout, we sample five candidate actions for each query with a decoding temperature of 1.0 to maintain sufficient exploration. All experiments are conducted on 8$\times$ NVIDIA H100-80G GPUs. Additional details on the training configuration, policy input and output formats, and evaluation protocols are provided in the Appendix.

### 4.1 Main Results

**ScreenSpot-Pro.** Table 1 presents the results on ScreenSpot-Pro. MAI-UI-2B achieves an average grounding accuracy of 56.7 before reinforcement fine-tuning. Our semantic reinforcement framework with Self-Reward RFT improves its performance to 59.0, yielding a gain of 2.3 points. Similarly, Qwen3-VL-2B improves from 43.4 to 45.7. The gains are distributed across multiple task domains and are observed on both text- and icon-based subsets. These results show that semantic reinforcement can improve both a GUI-specialized policy and a general-purpose VLM backbone.

Table 1: Performance comparison on the ScreenSpot-Pro benchmark across different tasks.

| Model | CAD | | Dev | | Creative | | Scientific | | Office | | OS | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Text | Icon | Text | Icon | Text | Icon | Text | Icon | Text | Icon | Text | Icon | Text | Icon | All |
| GPT-4o | 2.0 | 0.0 | 1.3 | 0.0 | 1.0 | 0.0 | 2.1 | 0.0 | 1.1 | 0.0 | 0.0 | 0.0 | 1.3 | 0.0 | 0.8 |
| Claude-CUA | 14.5 | 3.7 | 22.0 | 3.9 | 25.9 | 3.4 | 33.9 | 15.8 | 30.1 | 16.3 | 11.0 | 4.5 | 23.4 | 7.1 | 17.1 |
| Aria-UI | 7.6 | 1.6 | 16.2 | 0.0 | 23.7 | 2.1 | 27.1 | 6.4 | 20.3 | 1.9 | 4.7 | 0.0 | 17.1 | 2.0 | 11.3 |
| ShowUI-2B | 2.5 | 0.0 | 16.9 | 1.4 | 9.1 | 0.0 | 13.2 | 7.3 | 15.3 | 7.5 | 10.3 | 2.2 | 10.8 | 2.6 | 7.7 |
| UI-R1-3B | 11.2 | 6.3 | 22.7 | 4.1 | 27.3 | 3.5 | 42.4 | 11.8 | 32.2 | 11.3 | 13.1 | 4.5 | 24.9 | 6.4 | 17.8 |
| GUI-R1-3B | 26.4 | 7.8 | 33.8 | 4.8 | 40.9 | 5.6 | 61.8 | 17.3 | 53.6 | 17.0 | 28.1 | 5.6 | – | – | – |
| UI-TARS-2B | 17.8 | 4.7 | 47.4 | 4.1 | 42.9 | 6.3 | 56.9 | 17.3 | 50.3 | 17.0 | 21.5 | 5.6 | 39.6 | 8.4 | 27.7 |
| SE-GUI-3B | 38.1 | 12.5 | 55.8 | 7.6 | 47.0 | 4.9 | 61.8 | 16.4 | 59.9 | 24.5 | 40.2 | 12.4 | 50.4 | 11.8 | 35.9 |
| OS-Atlas-7B | 12.2 | 4.7 | 33.1 | 1.4 | 28.8 | 2.8 | 37.5 | 7.3 | 33.9 | 5.7 | 27.1 | 4.5 | 28.1 | 4.0 | 18.9 |
| UGround-7B | 14.2 | 1.6 | 26.6 | 2.1 | 27.3 | 2.8 | 31.9 | 2.7 | 31.6 | 11.3 | 17.8 | 0.0 | 25.0 | 2.8 | 16.5 |
| UGround-V1-7B | 15.8 | 1.2 | 51.9 | 2.8 | 47.5 | 9.7 | 57.6 | 14.5 | 60.5 | 13.2 | 38.3 | 7.9 | 45.2 | 8.1 | 31.1 |
| GUI-R1-7B | 23.9 | 6.3 | 49.4 | 4.8 | 38.9 | 8.4 | 55.6 | 11.8 | 58.7 | 26.4 | 42.1 | 16.9 | – | – | – |
| UI-TARS-7B | 20.8 | 9.4 | 58.4 | 12.4 | 50.0 | 9.1 | 63.9 | 31.8 | 63.3 | 20.8 | 30.8 | 16.9 | 47.8 | 16.2 | 35.7 |
| SE-GUI-7B | 51.3 | 42.2 | 68.2 | 19.3 | 57.6 | 9.1 | 75.0 | 28.2 | 78.5 | 43.4 | 49.5 | 25.8 | 63.5 | 21.0 | 47.3 |
| UI-Venus-1.5-2B | 54.8 | 34.4 | 70.1 | 44.8 | 63.6 | 28.7 | 76.4 | 38.2 | 81.4 | 47.2 | 72.9 | 51.7 | 69.0 | 39.9 | 57.7 |
| MAI-UI-2B | 61.4 | 20.3 | 78.6 | 31.0 | 68.7 | 22.4 | 79.9 | 31.8 | 85.9 | 39.6 | 66.4 | 40.4 | 73.4 | 30.5 | 56.7 |
| + Self-Reward RFT | 61.9 | 23.4 | 76.0 | 37.9 | 71.7 | 27.3 | 79.9 | 35.5 | 85.9 | 43.4 | 72.9 | 40.4 | 74.3 | 34.3 | 59.0 |
| Qwen3-VL-2B | 32.0 | 17.2 | 66.2 | 13.1 | 57.6 | 17.5 | 63.9 | 23.6 | 72.3 | 39.6 | 57.9 | 25.8 | 57.4 | 20.7 | 43.4 |
| + Self-Reward RFT | 37.6 | 21.9 | 64.9 | 15.2 | 59.1 | 19.6 | 64.6 | 25.5 | 76.3 | 37.7 | 62.6 | 28.1 | 60.0 | 22.7 | 45.7 |
| Qwen3-VL-8B | 47.7 | 10.9 | 70.1 | 22.8 | 65.2 | 15.4 | 72.2 | 27.3 | 85.3 | 30.2 | 69.2 | 24.7 | 67.6 | 21.5 | 49.1 |
| + Self-Reward RFT | 54.3 | 12.5 | 74.0 | 20.0 | 67.7 | 15.4 | 71.5 | 30.0 | 78.0 | 30.2 | 69.2 | 25.8 | 68.6 | 21.7 | 50.7 |

Table 2: Performance comparison on various grounding benchmarks.

| Models | Grounding Benchmarks | | | |
|---|---|---|---|---|
| | ScreenSpot-Pro | MMBench | OSWorld-G-R | OSWorld-G |
| Seed1.8 | 64.3 | - | - | - |
| OpenCUA-7B | 50.0 | - | - | 55.3 |
| GTA1-7B | 50.1 | - | 67.7 | 60.1 |
| GTA1-32B | 63.6 | - | 72.2 | 65.2 |
| GUI-Owl-7B | 54.9 | 80.5 | - | 55.9 |
| UI-TARS-1.5-7B | 35.7 | 64.3 | 64.2 | 52.8 |
| UI-Venus-1.0-7B | 50.8 | 79.9 | 61.7 | 54.6 |
| Holo2-8B | 58.9 | 84.5 | 70.1 | 63.5 |
| Step-GUI-4B | 60.0 | 84.0 | 66.9 | 60.5 |
| Step-GUI-8B | 62.6 | 85.6 | 70.0 | - |
| MAI-UI-8B | 65.8 | 88.8 | 68.6 | 60.1 |
| UI-Venus-1.5-8B | 68.4 | 88.1 | 74.1 | 69.7 |
| MAI-UI-2B | 56.7 | 82.6 | 62.5 | 52.0 |
| + Self-Reward RFT | 59.0$_{\uparrow 2.3}$ | 82.8$_{\uparrow 0.2}$ | 62.8$_{\uparrow 0.3}$ | 53.5$_{\uparrow 1.5}$ |
| Qwen3-VL-2B | 43.4 | 61.6 | 44.9 | 34.8 |
| + Self-Reward RFT | 45.7$_{\uparrow 2.3}$ | 64.7$_{\uparrow 3.1}$ | 47.0$_{\uparrow 2.1}$ | 38.8$_{\uparrow 4.0}$ |
| Qwen3-VL-8B | 49.1 | 81.4 | 67.0 | 57.2 |
| + Self-Reward RFT | 50.7$_{\uparrow 1.6}$ | 82.1$_{\uparrow 0.7}$ | 68.3$_{\uparrow 1.3}$ | 58.8$_{\uparrow 1.6}$ |

**Cross-Benchmark Evaluation.** Table 2 evaluates the trained policies across four GUI grounding benchmarks. For MAI-UI-2B, our framework improves ScreenSpot-Pro from 56.7 to 59.0 and also yields gains on MMBench, OSWorld-G-R, and OSWorld-G. A similar pattern is observed for Qwen3-VL-2B, which improves across all four benchmarks, with larger gains on MMBench, OSWorld-G-R, and OSWorld-G. Although the magnitude of improvement varies across benchmarks and policy backbones, no evaluated benchmark shows a performance decrease after reinforcement fine-tuning. These results indicate that semantic reinforcement is not limited to a single benchmark and transfers across different GUI grounding settings.

### 4.2 Ablation Study

To better understand the contribution of each component in our framework, we conduct a series of controlled ablation studies on ScreenSpotPro. Unless otherwise specified, the ablation studies use 3k AndroidControl samples, while the main results use the full 6k training mixture.

**Reward Formulation.** We first study how numerical and semantic rewards affect policies at different levels of GUI specialization (Table 3). For Qwen3-VL-2B, numerical rewards provide clear improvements over the base model, increasing the average score from 43.4 to 45.0 with binary reward and to 45.2 with Gaussian reward. In contrast, for MAI-UI-2B, which has already undergone extensive coordinate-based GUI training, the same numerical rewards yield only marginal gains, from 56.7 to 56.9 and 56.8, respectively. This suggests that coordinate-based reinforcement remains important for a less specialized policy, but provides diminishing returns once strong localization capability has already been established.

Semantic reward exhibits the opposite trend. When applied alone, it provides little benefit to Qwen3-VL-2B, whereas MAI-UI-2B improves substantially from 56.7 to 58.1 with Signed Logits and to 58.4 with TFMargin. These results indicate that semantic supervision is most effective when built on top of a policy with sufficiently strong grounding ability. Rather than replacing geometric supervision, semantic reward complements it by exploiting interaction information that is not captured by coordinate accuracy alone. Combining numerical and semantic rewards further improves Qwen3-

Table 3: Ablation study of different reward signals on ScreenSpot-Pro.

| Num-RL | Sem-RL | Signal | Self-Reward | SSPro | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | CAD | Dev | Creative | Scientific | Office | OS | Avg-Text | Avg-Icon | Avg. |
| **Qwen3-VL-2B as the Policy Model** | | | | | | | | | | | | |
| – | – | – | – | 28.4 | 40.5 | 40.8 | 46.5 | 64.8 | 43.4 | 57.4 | 20.7 | 43.4 |
| 2k | – | Binary | ✗ | 30.3 | 40.8 | 43.1 | 48.8 | 61.7 | 46.4 | 59.4 | 21.7 | 45.0 |
| 2k | – | Gaussian | ✗ | 30.7 | 41.1 | 42.2 | 48.8 | 67.8 | 44.4 | 59.7 | 21.7 | 45.2 |
| – | 3k | Signed Logits | ✗ | 29.1 | 39.5 | 40.8 | 46.5 | 65.7 | 43.4 | 57.5 | 20.7 | 43.5 |
| – | 3k | TFMargin | ✗ | 29.1 | 39.8 | 40.5 | 46.1 | 64.3 | 44.4 | 57.5 | 20.4 | 43.3 |
| 2k | 3k | TFMargin | ✗ | 33.0 | 40.8 | 42.2 | 48.4 | 66.5 | 45.9 | 59.8 | 22.2 | 45.4 |
| 2k | 3k | TFMargin | ✓ | 31.4 | 42.5 | 43.4 | 49.2 | 65.2 | 47.4 | 60.0 | 22.7 | 45.7 |
| **MAI-UI-2B as the Policy Model** | | | | | | | | | | | | |
| – | – | – | – | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 |
| 2k | – | Binary | ✗ | 51.4 | 56.2 | 49.3 | 59.3 | 75.3 | 54.6 | 73.4 | 30.7 | 56.9 |
| 2k | – | Gaussian | ✗ | 51.3 | 59.5 | 48.4 | 58.7 | 71.7 | 54.6 | 72.6 | 31.3 | 56.8 |
| – | 3k | Signed Logits | ✗ | 51.3 | 57.5 | 51.3 | 59.4 | 74.8 | 58.7 | 73.8 | 32.8 | 58.1 |
| – | 3k | TFMargin | ✗ | 50.6 | 59.9 | 50.4 | 59.4 | 74.8 | 59.7 | 73.6 | 33.8 | 58.4 |
| 2k | 3k | TFMargin | ✗ | 52.1 | 56.5 | 50.4 | 60.2 | 76.1 | 60.2 | 74.0 | 33.1 | 58.4 |
| 2k | 3k | TFMargin | ✓ | 51.0 | 59.2 | 50.7 | 59.4 | 75.7 | 59.7 | 74.4 | 32.8 | 58.5 |

Table 4: Ablation study on reward model choice and training cost on ScreenSpot-Pro.

| Reward Model | ScreenSpot-Pro | | | | | | | | | Train Cost |
|---|---|---|---|---|---|---|---|---|---|---|
| | CAD | Dev | Creative | Scientific | Office | OS | Avg-Text | Avg-Icon | Avg. | |
| *MAI-UI-2B as the Policy Model* | | | | | | | | | | |
| ✗ | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 | 0 |
| Binary-Rule | 51.4 | 56.2 | 49.3 | 59.3 | 75.3 | 54.6 | 73.4 | 30.7 | 56.9 | 33 |
| Qwen3-VL-2B | 53.3 | 57.5 | 48.7 | 60.2 | 76.1 | 55.1 | 73.6 | 32.1 | 57.5 | 57 |
| Qwen3-VL-4B | 53.1 | 57.4 | 48.3 | 60.7 | 76.0 | 55.1 | 73.3 | 32.0 | 57.3 | 83 |
| Qwen3-VL-8B | 53.3 | 57.5 | 48.9 | 60.2 | 76.2 | 55.3 | 73.6 | 32.1 | 57.6 | 132 |
| MAI-UI-2B | 52.1 | 56.5 | 50.4 | 60.2 | 76.1 | 60.2 | 74.0 | 33.1 | 58.4 | 48 |
| UI-Venus-1.5-2B | 51.7 | 57.5 | 51.3 | 60.2 | 76.1 | 60.2 | 74.7 | 32.8 | 58.7 | 43 |

VL-2B to 45.4, showing that the two signals can also be complementary when geometric learning has not yet saturated.

**Self-Reward.** Finally, we examine whether the policy model can also serve as the semantic evaluator through periodic synchronization. Self-Reward provides modest improvements for both policy backbones, increasing Qwen3-VL-2B from 45.4 to 45.7 and MAI-UI-2B from 58.4 to 58.5. Although the gains are small, they are obtained without requiring a separately trained reward model. These results support the feasibility of reusing the policy's multimodal representations for semantic evaluation, while periodic synchronization enables a lightweight self-reward mechanism that preserves or slightly improves grounding performance.

### 4.3 Effect of Reward Model Specialization

We investigate how the choice of reward model affects semantic reinforcement learning, considering both performance and computational cost. As shown in Table 4, general-purpose Qwen3-VL models consistently improve MAI-UI-2B over the base score of 56.7, reaching approximately 57.3–57.6. However, scaling the reward model from 2B to 8B does not yield consistent gains, while substantially increasing training cost. In contrast, GUI-specialized reward models are both more effective and more efficient. MAI-UI-2B and UI-Venus-1.5-2B achieve average scores of 58.4 and 58.7, respectively, outperforming larger general-purpose VLMs despite requiring only 48 and 43 GPU hours. By comparison, the 8B general-purpose reward model requires up to 132 GPU hours while providing only limited additional benefit. These results suggest that the effectiveness of semantic reward depends more on domain-aligned GUI representations than on simply scaling generic model capacity, leading to a better trade-off between reward quality and computational efficiency.

## 5 Further Analysis

We conduct further analysis to better understand when and why semantic reward is effective, as well as its practical applicability and remaining limitations. We first examine whether semantic supervision expands the range of GUI data that can be used for reinforcement fine-tuning and whether such supervision can be obtained at low cost from existing datasets. We then analyze model failures and the relationship between numerical and semantic reward signals to characterize where the two

Table 5: Ablation of supervision types and semantic credit assignment on ScreenSpot-Pro.

| Reward | Grounding | Action Prediction | Credit Assign | ScreenSpot-Pro | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | CAD | Dev | Creative | Scientific | Office | OS | Avg-Text | Avg-Icon | Avg. |
| *MAI-UI-2B as the Policy Model* | | | | | | | | | | | | |
| - | - | - | - | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 |
| Binary | 2k | - | - | 51.4 | 56.2 | 49.3 | 59.3 | 75.3 | 54.6 | 73.4 | 30.7 | 56.9 |
| Gauss | 2k | - | - | 51.3 | 59.5 | 48.4 | 58.7 | 71.7 | 54.6 | 72.6 | 31.3 | 56.8 |
| TFMargin | 2k | - | - | 51.7 | 57.9 | 49.9 | 58.7 | 76.1 | 58.2 | 74.1 | 31.8 | 57.9 |
| TFMargin | 2k | 1k | - | 50.6 | 59.9 | 50.4 | 59.4 | 74.8 | 59.7 | 73.6 | 33.8 | 58.4 |
| TFMargin | 2k | 1k | ✓ | 50.6 | 56.9 | 49.6 | 60.6 | 75.7 | 55.1 | 73.1 | 32.0 | 57.4 |

Table 6: Effect of different training data sources on ScreenSpot-Pro.

| Data Source | Data Size | ScreenSpot-Pro | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | CAD | Dev | Creative | Scientific | Office | OS | Avg-Text | Avg-Icon | Avg. |
| *MAI-UI-2B as the Policy Model* | | | | | | | | | | |
| - | 0 | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 |
| AndroidControl | 3k | 51.0 | 59.2 | 50.7 | 59.4 | 75.7 | 59.7 | 74.4 | 32.8 | 58.5 |
| GroundCUA | 3k | 51.7 | 58.2 | 50.4 | 59.4 | 74.8 | 61.2 | 74.0 | 33.3 | 58.4 |
| Mixture | 6k | 52.5 | 57.5 | 53.1 | 60.6 | 76.1 | 58.2 | 74.3 | 34.3 | 59.0 |

forms of supervision agree or diverge. All controlled experiments use MAI-UI-2B as the policy model and follow the same training protocol.

**Extending Usable GUI Training Data.** Coordinate-based GUI grounding typically relies on explicit localization targets, which limits direct reinforcement fine-tuning to interaction steps with coordinate annotations. In contrast, our semantic formulation represents supervision as natural-language actions, thereby relaxing this dependence on explicit localization targets and allowing both grounding and action-prediction data to contribute to training. To evaluate this advantage under a realistic data composition, we follow the natural distribution of AndroidControl and use 2k grounding samples together with 1k action-prediction samples. Using only the 2k grounding samples, TFMargin improves the average ScreenSpot-Pro score from 56.7 to 57.9. Adding the 1k action-prediction samples further improves performance to 58.4. These results show that semantic reward can make use of a broader range of GUI interaction data, including samples that are not directly compatible with coordinate-based grounding objectives.

Inspired by recent work on dense credit assignment for reinforcement learning (Zhao et al., 2026; Yang et al., 2026), we further apply an OPD-style credit signal to the predicted semantic actions. Specifically, we treat each generated semantic action as a short trajectory and assign denser supervision over its generation. However, this does not improve performance: the average ScreenSpot-Pro score decreases from 58.4 to 57.4. One possible reason is that semantic action descriptions are much shorter than typical reasoning or interaction trajectories, leaving limited room for fine-grained credit assignment to provide additional benefit. These results suggest that, in our setting, action-level semantic reward is already sufficient, while further decomposing the reward within a short semantic action brings little advantage.

**Accessibility of Semantic Supervision.** We further examine whether semantic action supervision requires additional manual annotation. In practice, such supervision can often be obtained from existing GUI interaction data. AndroidControl naturally provides aligned high-level task instructions and low-level action descriptions, which can be directly used to construct semantic action training examples. We further consider GroundCUA, where each interaction provides a low-level instruction together with its target coordinate. We construct the corresponding high-level task instruction from the existing interaction information, allowing the resulting high-level–low-level instruction pairs to be used in the same semantic training format. Details of this construction are provided in Appendix D. Using 3k AndroidControl samples achieves an average score of 58.5, while 3k GroundCUA samples obtain a comparable 58.4. Combining the two sources further improves performance to 59.0. These results show that semantic supervision can be constructed from different forms of existing GUI interaction data, making the proposed training formulation applicable to a broader range of datasets.

**Evolution of the Semantic Evaluator.** We further examine how the semantic evaluator changes during Self-Reward RFT. To evaluate its stability across synchronization, we use a fixed functional-equivalence validation set and compare the evaluator at initialization and after synchronization at steps 50, 100, and 150. The detailed validation protocol and data construction are provided in Appendix B.

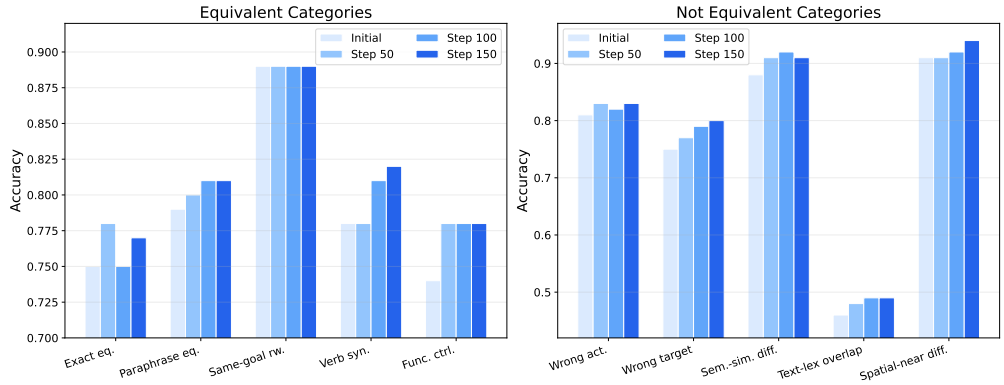


Figure 3: Category-level verifier accuracy across synchronization checkpoints. The left panel reports equivalent action categories, and the right panel reports not-equivalent action categories. Bars progress from light to dark blue from the initial checkpoint to step 150. Abbreviations on the x-axis are expanded as follows: *Exact eq.* = exact equivalent, *Paraphrase eq.* = paraphrase equivalent, *Same-goal rw.* = same-goal rewrite, *Verb syn.* = verb synonym, *Func. ctrl.* = functional control, *Wrong act.* = wrong action type, *Wrong target* = wrong target element, *Sem.-sim. diff.* = semantically similar but different, *Text-lex overlap* = text-dense lexical overlap, and *Spatial-near diff.* = spatially nearby but different. Most categories remain stable or improve slightly across synchronization, while text-dense lexical-overlap cases remain the most challenging.

As shown in Figure 3, the evaluator remains generally stable and improves in several semantic categories over training. In particular, paraphrase equivalence, verb-synonym, wrong-target, and spatially nearby but functionally different cases show clear or modest improvements, while most other categories remain stable. Text-dense lexical overlap remains the most challenging case, but does not degrade over training.

Overall, these results suggest that periodic synchronization allows the semantic evaluator to follow the evolving policy without causing systematic degradation. Appendix B further reports reward separation, score distributions, and verifier stability across synchronization checkpoints, while Appendix C examines policy-side failures, difficult GUI elements, representative verification errors, and cases where semantic and numerical rewards disagree. Together, these analyses provide a more complete view of the reliability and limitations of semantic supervision beyond aggregate benchmark performance.

## 6 Conclusion

In this work, we investigate semantic supervision as a complementary signal for reinforcement learning of GUI agents. While coordinate-based rewards provide effective supervision for spatial localization, they do not fully capture the semantic information available in GUI interaction data. We therefore formulate GUI action learning at the semantic level, where a VLM evaluates the functional consistency between predicted and target action descriptions and converts this judgment into a reward signal. Building on this formulation, we introduce Self-Reward RFT, where the semantic evaluator is initialized from the policy and periodically updated with the latest policy parameters while remaining frozen between updates. This design reuses the policy's multimodal representations for semantic evaluation without requiring a separately trained reward model. Experiments across four GUI grounding benchmarks and two policy backbones show consistent improvements across different grounding settings. Using only 6k training samples, our method improves MAI-UI-2B from 56.7% to 59.0% on ScreenSpot-Pro, achieving strong performance among lightweight 2B GUI models while remaining competitive with several larger models. Further analysis shows that semantic and coordinate-based rewards capture different aspects of GUI action correctness and can provide complementary supervision. These results demonstrate the value of incorporating semantic feedback into GUI agent post-training.

## AI USAGE STATEMENT

We use generative AI to assist with language editing, manuscript organization, code development and debugging, figure presentation, and checks of mathematical consistency. Generative AI is also used in two parts of the experimental pipeline. First, for the GroundCUA training data described in Appendix D, we use GPT-5.5 to infer high-level task instructions from the existing interaction annotations, while retaining the original low-level instructions as the target semantic actions. Second, for the reward-quality validation set described in Appendix B, generative AI is used to synthesize candidate semantic-action pairs; the final functional-equivalence labels are independently determined through human review and majority voting. The authors formulate the research questions, develop the methodological and theoretical arguments, design the experiments and evaluation protocols, and independently assess AI-assisted suggestions concerning equations and implementation. We verify AI-assisted code against the intended computations and check all reported measurements against the recorded experimental outputs. Generative AI does not determine the final experimental labels, reported results, or scientific conclusions. The authors review and revise all AI-assisted material and retain full responsibility for the final text, mathematics, analyses, results, and released artifacts.

## REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents, 2025. URL https://arxiv.org/abs/2504.00906.

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

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

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuanchun Li, Yuqing Yang, Rui Tan, Mo Li, and Lili Qiu. Advancing mobile gui agents: A verifier-driven approach to practical deployment. *arXiv preprint arXiv:2503.15937*, 2025.

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

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

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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*, pp. 8778–8786, 2025.

Wei Li, William E. Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on UI control agents. In *NeurIPS*, 2024a.

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

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for generalist gui agent. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices, 2025a. URL https://arxiv.org/abs/2406.08451.

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

Run Luo, Lu Wang, Wanwei He, 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. doi: 10.48550/arXiv.2504.10458. URL https://arxiv.org/abs/2504.10458.

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

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

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

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

Haozhan Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. Vlm-r1: A stable and generalizable r1-style large vision-language model. https://github.com/om-ai-lab/VLM-R1, 2025. Accessed: 2025-02-15.

Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, Jun Xiao, and Yueting Zhuang. GUI-G$^2$: Gaussian reward modeling for GUI grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33214–33222, 2026. doi: 10.1609/aaai.v40i39.40606. URL https://ojs.aaai.org/index.php/AAAI/article/view/40606.

Baode Wang, Biao Wu, Weizhen Li, Meng Fang, Zuming Huang, Jun Huang, Haozhe Wang, Yanjie Liang, Ling Chen, Wei Chu, et al. Infinity parser: Layout aware reinforcement learning for scanned document parsing. *arXiv preprint arXiv:2506.03197*, 2025a.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception, 2024. URL https://arxiv.org/abs/2401.16158.

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*, 2025b.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. *arXiv preprint arXiv:2308.15272*, 2023.

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.

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. *arXiv preprint arXiv:2410.23218*, 2024.

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

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, Yiheng Xu, Junli Wang, Doyen Sahoo, Tao Yu, and Caiming Xiong. Scaling computer-use grounding via user interface decomposition and synthesis, 2025. URL https://arxiv.org/abs/2505.13227.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a.

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.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. *arXiv preprint arXiv:2412.16256*, 2024b.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

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.

Chi Zhang, Zhao Yang, Jiaxuan Liu, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu. Appagent: Multimodal agents as smartphone users. *arXiv preprint arXiv:2312.13771*, 2023.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, et al. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*, 2025a.

Yiyang Zhou, Zhiyuan Fan, Dongjie Cheng, Sihan Yang, Zhaorun Chen, Chenhang Cui, Xiyao Wang, Yun Li, Linjun Zhang, and Huaxiu Yao. Calibrated self-rewarding vision language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 51503–51531. Curran Associates, Inc., 2024. doi: 10.52202/079017-1630. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5c20c00504e0c049ec2370d0cceaf3c4-Abstract-Conference.html.

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*, 2025b.

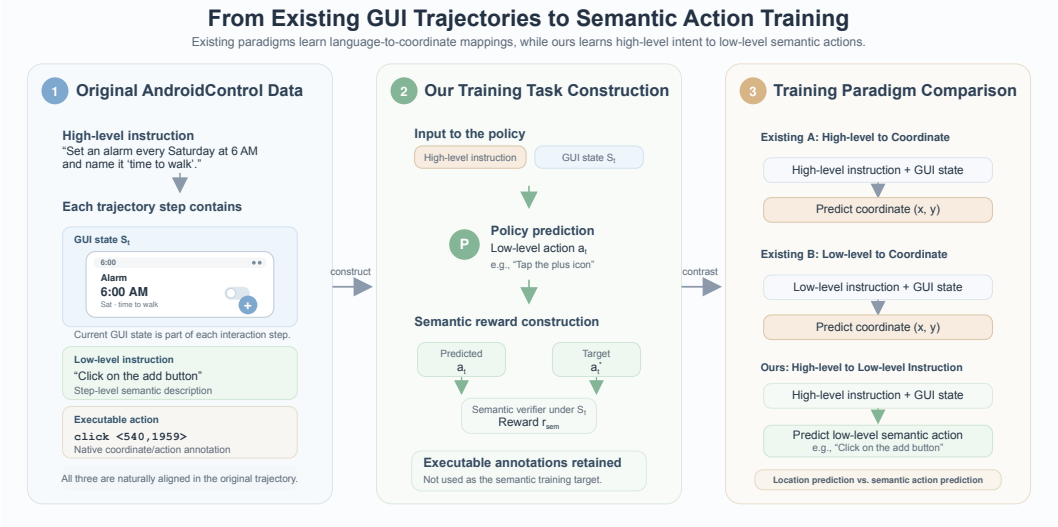


Figure 4: **Construction of semantic action supervision from existing GUI trajectories.** AndroidControl provides aligned high-level instructions, GUI states, low-level instructions, and executable actions. We reuse the low-level instruction as the semantic target, shifting training from language-to-coordinate prediction to high-level-to-low-level semantic action prediction.

# APPENDIX

## A CONSTRUCTING SEMANTIC ACTION SUPERVISION

As illustrated in Fig. 4, AndroidControl provides naturally aligned supervision at multiple levels of GUI interaction. Each trajectory is associated with a high-level task instruction, while every interaction step contains the current GUI observation $S_t$, the previous interaction history $H_t$, a low-level instruction describing the intended operation, and the corresponding executable action annotation. We directly reuse the low-level instruction as the target semantic action $a_t^*$, without introducing additional semantic annotations.

During semantic reinforcement fine-tuning, the policy input consists of the current GUI screenshot, the high-level task instruction, and the previous interaction history. The policy is trained to generate a low-level semantic action $a_t$ in natural language. Concretely, the model is instructed to place its reasoning process inside `<think></think>` tags and its final concise action description inside `<answer></answer>` tags. Typical outputs include actions such as `Click the Close button`, `Press Back`, or `Type the username in the input field`. The interaction history is represented as a serialized sequence of preceding low-level actions and is directly included in the policy prompt.

This formulation changes the prediction target relative to conventional coordinate-based GUI grounding. Rather than directly optimizing a mapping from language instructions to spatial coordinates, the policy predicts the intended low-level operation expressed in natural language. Given the generated semantic action $a_t$ and the target action $a_t^*$, the semantic evaluator assesses their functional equivalence under the same GUI observation $S_t$. The resulting semantic consistency score is then used as the reward for reinforcement fine-tuning.

Importantly, the semantic action representation is used as the prediction target during semantic reinforcement fine-tuning, but it does not alter the underlying action-generation capability of the policy model. During semantic RFT, the policy is prompted to generate natural-language low-level actions for semantic reward computation. For downstream grounding evaluation, we directly switch to the benchmark-specific coordinate-prediction prompt and ask the same policy model to produce spatial actions in the required format. Therefore, no additional parser, grounding module, or semantic-to-executable conversion component is introduced. The proposed semantic formulation changes the training supervision and reward signal, while the output format can be adapted at inference time through prompting to match the requirements of different GUI benchmarks.

| Type | Representative Action Pair |
|---|---|
| **Equivalent** | |
| Paraphrase equivalent | **Target:** Click on the typewriter icon<br>**Candidate:** Tap the typewriter icon |
| Verb synonym | **Target:** Click on the Japanese sushi tab<br>**Candidate:** Select the Japanese sushi tab |
| Same-goal rewrite | **Target:** Click on the CONTINUE button at the bottom<br>**Candidate:** Perform the same intended UI operation by clicking the CONTINUE button at the bottom |
| **Not Equivalent** | |
| Wrong action type | **Target:** Click on the cross icon of the search bar at the top of the screen<br>**Candidate:** Type the text "cross icon of the search bar at the top of the screen" into the current input field |
| Wrong target element | **Target:** Click on the star icon at the bottom-left corner of the screen<br>**Candidate:** Click on the CONTINUE button at the bottom |
| Text-dense lexical overlap | **Target:** Click on the first search result<br>**Candidate:** Click on the first settings result |
| Spatial-nearby but different | **Target:** Click on the OK button to save the date<br>**Candidate:** Click the nearby control next to the OK button rather than the OK button itself |

Table 7: Representative semantic-action pairs used for functional-equivalence validation, including equivalent reformulations and challenging non-equivalent actions.

**Training protocol for numerical and semantic supervision.** For the configurations using both numerical and semantic supervision in Table 3, we adopt separate training stages, with each stage running for one epoch. During the numerical stage, the policy is prompted to predict coordinates and is optimized using the rule-based coordinate reward. During the semantic stage, the policy is prompted to generate a natural-language action description and is optimized using the semantic reward described in Section 3.2. Thus, the two reward signals are applied in separate stages rather than combined on the same generated output. Self-Reward is used only in the semantic stage: the semantic evaluator is initialized from the policy at the start of this stage and synchronized with the policy every 50 optimization steps, remaining frozen between synchronization points. The numerical stage requires no VLM-based reward evaluator. At evaluation time, the trained policy uses the dedicated grounding prompt in Figure 11 to directly predict coordinates.

## B Direct Validation of Semantic Reward Quality

A central question in Self-Reward RFT is whether the semantic evaluator provides an informative and stable training signal as it is periodically synchronized with the evolving policy. Downstream grounding improvements alone do not directly establish reward quality, since policy performance may improve even when the underlying reward signal is noisy or only weakly aligned with functional equivalence. Periodically refreshing the evaluator from the evolving policy also raises the question of whether its reward function remains stable throughout training.

To directly examine these issues, we construct a fixed 1,000-pair functional-equivalence validation set as our primary reward-quality evaluation. The candidate pairs are first generated automatically and then manually reviewed and filtered to ensure that the retained examples have reliable functional-equivalence labels. The resulting set is approximately balanced between equivalent and non-equivalent semantic-action pairs and covers ten action categories, including paraphrases, wrong targets, semantically similar but operationally different actions, text-dense lexical overlap, and spatially nearby but functionally distinct controls. These cases are designed to test whether the verifier captures functional equivalence rather than relying only on superficial textual or spatial similarity.

We evaluate the semantic verifier at initialization and after synchronization at steps 50, 100, and 150, using exactly the same validation examples at every checkpoint. We report accuracy, F1, AUROC, and the reward separation between equivalent and non-equivalent action pairs. These measurements allow us to evaluate both the discriminative quality of the semantic reward and its stability under periodic synchronization.

We further perform a category-level hard-case analysis to characterize where the verifier succeeds and where errors remain concentrated. Finally, we compare the base and step-150 policies on ScreenSpot-Pro as a downstream sanity check. We treat this downstream comparison as supporting evidence,

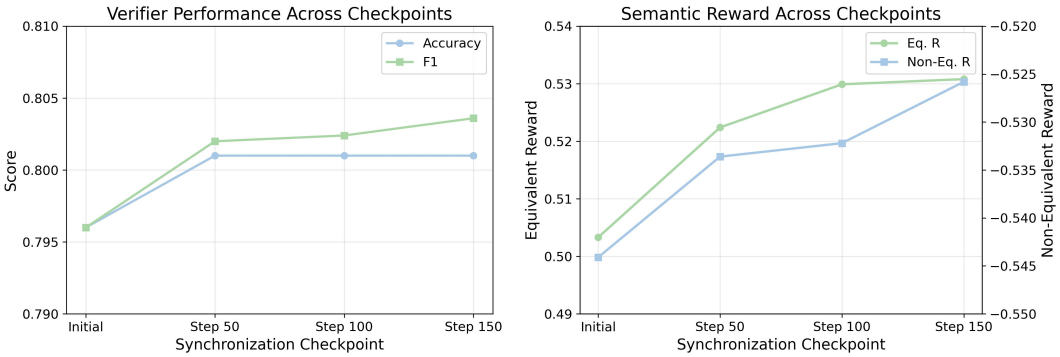


Figure 5: Semantic reward validation across synchronization checkpoints. Left: verifier accuracy and F1 remain stable with a slight upward trend from initialization to step 150.Right: equivalent action pairs consistently receive positive semantic rewards, while non-equivalent pairs receive negative rewards across all checkpoints. Together, these results indicate that periodic synchronization preserves functional-equivalence discrimination without observable reward collapse.

while the direct functional-equivalence evaluation provides the primary evidence for the quality and stability of the semantic reward.

### B.1 Validation Protocol

As shown in Table 7, We construct a fixed validation set of 1,000 semantic-action pairs that is held out from policy optimization. Candidate pairs are first synthesized using Codex GPT-5.6 and are then manually reviewed and filtered to retain examples with clear and reliable functional-equivalence judgments. Each retained example contains a GUI observation $S_t$, a target semantic action $a_t^*$, a candidate semantic action $a_t$, and a binary label indicating whether the two actions correspond to the same intended GUI operation under the current interface state. The final validation set contains 500 equivalent and 500 non-equivalent pairs and is organized into ten categories with 100 examples each. Positive pairs cover exact equivalence, paraphrases, same-goal rewrites, verb-level synonyms, and functionally interchangeable controls, while negative pairs include incorrect action types, incorrect target elements, semantically similar but operationally different actions, text-dense lexical overlap, and spatially nearby but functionally distinct controls. All evaluator checkpoints are tested on exactly the same validation examples, and we evaluate the initial evaluator together with the evaluators obtained after synchronization at steps 50, 100, and 150.

**Full-Balanced 1000-Pair Composition.** The validation set is intentionally balanced between equivalent and non-equivalent action pairs, with 500 examples in each group. Beyond simple exact matches, the positive set includes paraphrases and functionally equivalent actions, while the negative set contains several hard cases in which actions are lexically, semantically, or spatially similar but operationally different. This design allows us to evaluate whether the semantic verifier captures functional equivalence rather than relying on superficial textual or spatial similarity.

### B.2 Human Review and Quality Control

To ensure the reliability of the functional-equivalence labels, we apply a human review and majority-voting procedure to the automatically synthesized action pairs. We first use Codex GPT-5.6 to generate 2,000 candidate semantic-action pairs covering both equivalent and non-equivalent cases. Each candidate is independently reviewed by three human annotators, who judge whether the target and candidate actions correspond to the same intended GUI operation under the given interface state. We retain an example only when at least two of the three annotators agree on its functional-equivalence label. After this majority-voting procedure, 1,468 of the 2,000 synthesized candidates remain as valid examples. From this human-reviewed pool, we perform controlled sampling to construct the final 1,000-pair validation set, balancing equivalent and non-equivalent actions and covering the ten semantic-action categories described above. This procedure separates candidate generation from label validation: Codex GPT-5.6 is used to generate diverse candidate pairs, while the final functional-equivalence judgments used for evaluation are determined through human majority

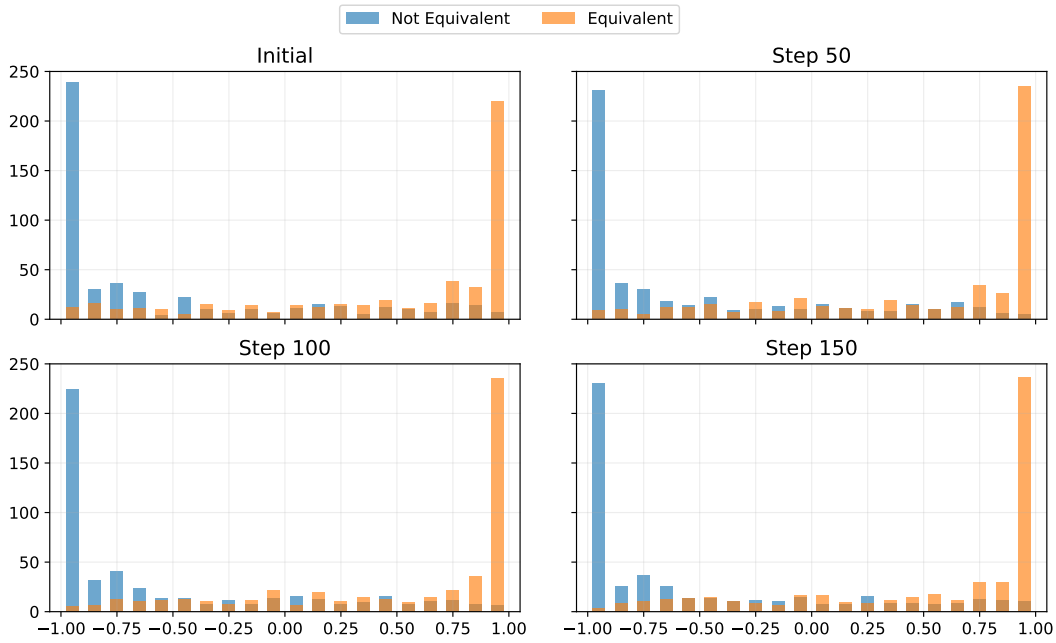


Figure 6: Semantic reward distributions for equivalent and non-equivalent action pairs on the 1,000-pair full-balanced validation set.

voting. The resulting validation set is fixed and is used unchanged to evaluate all semantic-verifier model.

### B.3 Main Reward-Quality Results

We evaluate the semantic verifier on the 1,000-pair full-balanced validation set, which contains equal numbers of equivalent and non-equivalent action pairs and covers all ten validation categories. The semantic verifier maintains consistent discrimination across synchronization checkpoints. AUROC remains between 0.865 and 0.869, while accuracy stays close to 0.80. More importantly, equivalent action pairs consistently receive positive rewards, whereas non-equivalent pairs receive negative rewards, resulting in a stable reward gap of approximately 1.05. These results show that the semantic reward provides a persistent separation between functionally equivalent and non-equivalent actions. Moreover, periodic synchronization with the evolving policy does not produce observable reward collapse or substantial degradation in functional-equivalence discrimination over the evaluated training interval. The results therefore support reward stability under synchronization, rather than a claim of monotonic improvement in reward quality.

### B.4 Reward Distribution

As shown in Figure 6, to further examine whether the semantic reward separates functionally equivalent and non-equivalent actions, we compare the reward distributions of the two groups on the 1,000-pair full-balanced validation set. Equivalent action pairs consistently receive positive rewards, whereas non-equivalent pairs are concentrated on the negative side of the reward distribution. Across synchronization checkpoints, the mean reward for equivalent pairs increases from 0.503 to 0.531, while the mean reward for non-equivalent pairs remains around $-0.54$ to $-0.53$. The resulting separation remains close to 1.05 throughout training. This indicates that the semantic reward preserves a clear distinction between functionally equivalent and non-equivalent actions after periodic synchronization.

### B.5 Semantic Supervision under SFT and RL

To separate the effect of semantic supervision from the optimization objective, we compare SFT and RL using the same 3k semantic-action samples. Without numerical pre-training, semantic SFT

Table 8: Comparison of semantic SFT and semantic RL with or without a preceding numerical RL stage on ScreenSpot-Pro.

| Pre-Num. RL | Sem. Data | Semantic Training | ScreenSpot-Pro | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | CAD | Dev | Creative | Scientific | Office | OS | Avg-T | Avg-I | Avg. |
| *MAI-UI-2B as the Policy Model* | | | | | | | | | | | |
| – | – | Base | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 |
| ✗ | 3k | SFT | 38.7 | 47.2 | 44.3 | 52.8 | 72.6 | 49.0 | 67.6 | 21.5 | 50.0 |
| ✗ | 3k | RL | 50.6 | 59.9 | 50.4 | 59.4 | 74.8 | 59.7 | 73.6 | 33.8 | 58.4 |
| ✓ | 3k | SFT | 44.1 | 47.8 | 45.7 | 53.5 | 67.0 | 49.5 | 68.6 | 21.7 | 50.7 |
| ✓ | 3k | RL | 51.0 | 59.2 | 50.7 | 59.4 | 75.7 | 59.7 | 74.4 | 32.8 | 58.5 |

reduces the average score from 56.7 to 50.0, while semantic RL improves it to 58.4. With a preceding 2k-sample numerical RL stage, the corresponding results are 50.7 and 58.5. This consistent gap shows that the improvement cannot be attributed to additional semantic-action supervision alone; in our setting, semantic feedback is substantially more effective when used as a reinforcement-learning reward than as a direct imitation target. The numerical pre-RL stage contributes only marginally to the final result.

## C Failure Mode Analysis

**Policy-Side Failure Patterns.** To better understand the failure patterns of the policy, we analyze the difficulty of different GUI elements in Figure 7. The most challenging cases include operations involving elements such as the "search icon tab at the bottom of the screen," "tools option," and "play icon at the bottom right of the screen." These elements are often small, low-saliency, or function-oriented controls that provide weaker visual or textual cues than prominent text-based buttons. Since the policy is trained to generate natural-language descriptions of the intended operation, these cases indicate that correctly identifying and expressing the semantics of such controls remains challenging. In particular, the policy appears more reliable on visually salient or text-explicit elements than on less conspicuous functional entry points. This observation suggests that semantic action prediction remains difficult for long-tail GUI elements whose function is not immediately evident from local appearance alone.

**Semantic–Numerical Reward Misalignment.** As shown in Figure 8, we further examine the relationship between semantic and numerical reward signals by randomly sampling 3k interaction instances and merging semantically or structurally similar entries, resulting in 218 evaluated elements. Across these instances, we observe cases in which the two reward signals assign substantially different scores to the same prediction. Some actions receive relatively high semantic scores but low numerical scores, indicating that the predicted operation can be semantically close to the target while remaining spatially distant from the annotated location. Conversely, other predictions receive high numerical scores but low semantic scores, suggesting that spatial proximity alone does not always imply functional equivalence between the predicted and target actions. These observations indicate that semantic and coordinate-based rewards capture different aspects of GUI action correctness. Rather than replacing geometric supervision, semantic reward therefore provides a complementary signal that explicitly evaluates semantic consistency.

**Failure Modes of Semantic Verification.** Semantic rewards remain less reliable in text-dense interfaces, such as the CAD, Dev, and Office Text subsets, where multiple GUI elements often exhibit highly similar lexical and semantic patterns. In these cases, the verifier may overestimate the functional equivalence between semantically related but operationally distinct actions. For example, when the target is 'Clear Command Window," the policy may instead select another generic 'clear" button on the same interface. Although these actions produce different outcomes, their strong lexical overlap can lead the semantic verifier to assign an undesirably high reward. Such false-positive evaluations result in inaccurate credit assignment and make it difficult to distinguish among nearby elements that require fine-grained local reasoning. This limitation helps explain the performance degradation observed on several text-heavy subsets and suggests that semantic rewards are less effective when action correctness depends on subtle distinctions between semantically similar interface elements.

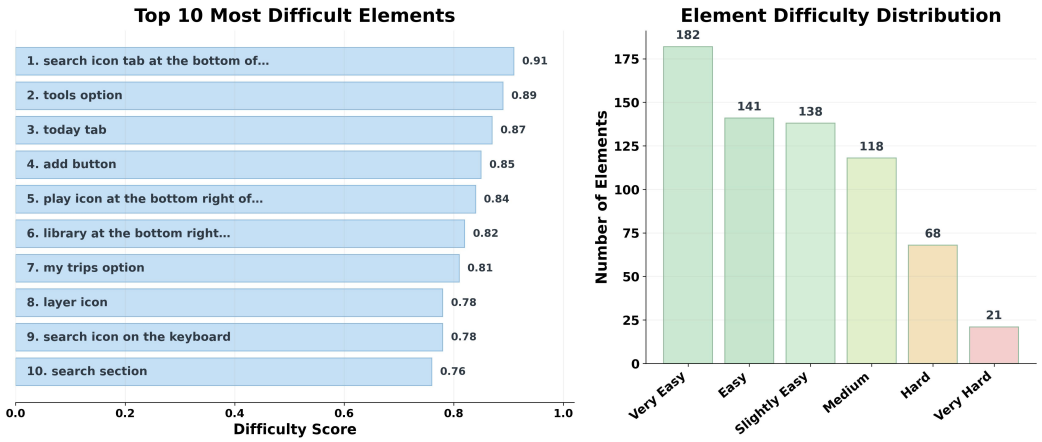


Figure 7: GUI element difficulty analysis. Left: top 10 most difficult UI elements. Right: distribution of element difficulty across six difficulty levels.

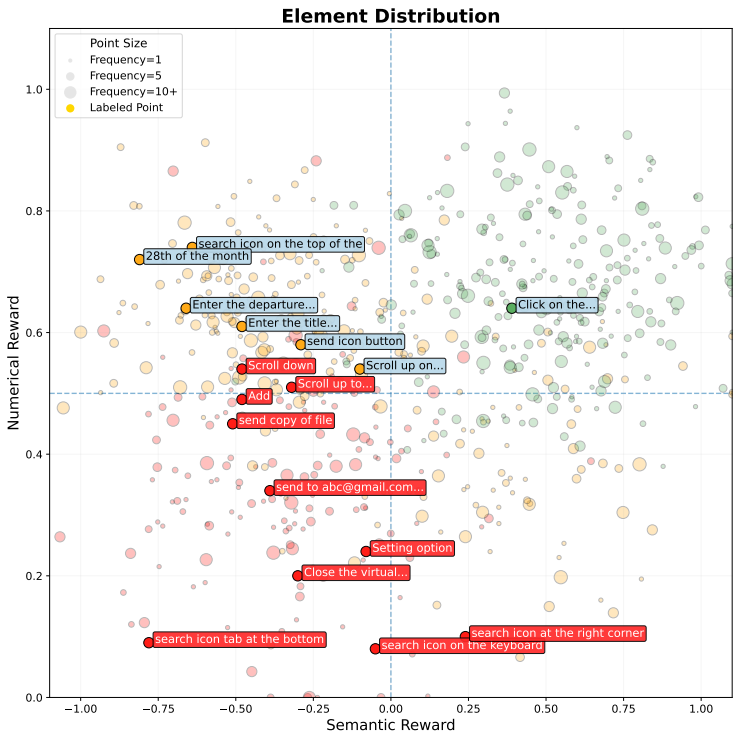


Figure 8: Semantic–numerical reward relationship across sampled GUI interactions. The two reward signals agree for many predictions but also exhibit clear disagreement, showing that semantic and coordinate-based rewards capture different aspects of GUI action correctness.

## D Experimental Details

**Baselines.** We compare our method with a broad set of representative models on ScreenSpot-Pro. These include proprietary general-purpose models, such as GPT-4o (Hurst et al., 2024) and Claude-CUA (Anthropic, 2024), as well as GUI-specialized models, including Aria-UI (Yang et al., 2024b), OS-Atlas (Wu et al., 2024), ShowUI (Lin et al., 2024), UGround and UGround-V1 (Gou et al., 2024), UI-R1 (Lu et al., 2025b), GUI-R1 (Xia & Luo, 2025), UI-TARS (Qin et al., 2025), and SE-GUI (Yuan et al., 2025). We further evaluate Self-Reward RFT on two policy backbones, MAI-UI-2B (Zhou

Table 9: Robustness of semantic reward to different verification prompts on ScreenSpot-Pro. We report results for four prompt variants and multiple random seeds using the original prompt.

| Prompt | Random Seed | ScreenSpot-Pro | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | CAD | Dev | Creative | Scientific | Office | OS | Avg-Text | Avg-Icon | Avg. |
| *MAI-UI-2B as the Policy Model* | | | | | | | | | | |
| - | - | 51.3 | 56.5 | 49.3 | 59.1 | 75.2 | 54.6 | 73.4 | 30.5 | 56.7 |
| Original Prompt | Seed 1 | 52.5 | 57.5 | 53.1 | 60.6 | 76.1 | 58.2 | 74.3 | 34.3 | 59.0 |
| Prompt V2 | Seed 1 | 52.5 | 58.2 | 50.1 | 60.2 | 75.7 | 61.2 | 74.8 | 32.8 | 58.8 |
| Prompt V3 | Seed 1 | 51.7 | 58.9 | 50.4 | 61.0 | 75.7 | 60.2 | 74.9 | 32.8 | 58.8 |
| Prompt V4 | Seed 1 | 50.6 | 58.2 | 51.3 | 59.8 | 76.1 | 60.2 | 74.4 | 32.9 | 58.6 |
| Prompt V5 | Seed 1 | 51.7 | 56.5 | 51.3 | 59.1 | 75.2 | 58.2 | 73.7 | 32.5 | 57.9 |
| Original Prompt | Seed 2 | 52.3 | 57.1 | 52.8 | 60.4 | 75.8 | 58.0 | 74.0 | 34.1 | 58.8 |
| Original Prompt | Seed 3 | 51.9 | 57.6 | 53.4 | 60.8 | 76.2 | 58.4 | 74.5 | 33.8 | 59.2 |
| Original Prompt | Seed 4 | 52.6 | 56.9 | 52.9 | 60.2 | 75.7 | 57.8 | 73.8 | 34.6 | 58.6 |
| Original Prompt | Seed 5 | 52.1 | 57.3 | 53.2 | 60.7 | 76.0 | 58.1 | 74.2 | 34.0 | 59.0 |

et al., 2025a) and Qwen3-VL-2B (Bai et al., 2025a), to examine whether the proposed reward design generalizes across different base models.

For the ablation studies, we implement the Binary coordinate reward following the standard point-in-box formulation used in prior GUI grounding work (Luo et al., 2025), and the Gaussian reward following GUI-G$^2$ (Tang et al., 2026). The Binary reward assigns a score of 1 when the predicted point falls inside the target bounding box and 0 otherwise, while the Gaussian reward provides a continuous spatial signal based on the distance between the predicted and target locations.

**Dataset.** We construct a 6k-sample training set by combining 3k interaction steps from AndroidControl with 3k samples from GroundCUA. AndroidControl naturally provides aligned high-level task instructions and low-level instructions for each interaction step. We therefore directly use the high-level instruction as the policy input and the corresponding low-level instruction as the target semantic action. GroundCUA provides a different annotation structure: each sample contains a low-level instruction together with the coordinate of the corresponding target element, but does not provide the high-level task instruction required by our training formulation. To construct compatible training examples, we use GPT-5.5 to infer a plausible high-level task instruction from the available low-level instruction and interaction context. The original low-level instruction is retained as the target semantic action. This produces a high-level–low-level instruction pair with the same training structure used for AndroidControl. After this conversion, both datasets can be represented using a unified training format, where the policy receives a high-level task instruction and the current GUI state and predicts the corresponding low-level semantic action.

**Benchmarks.** We evaluate our method on four GUI grounding benchmarks: ScreenSpot-Pro (Li et al., 2025), MMBench-GUI L2 (Wang et al., 2025b), OSWorld-G, and OSWorld-G-R (Xie et al., 2025). ScreenSpot-Pro focuses on high-resolution, fine-grained professional interfaces, MMBench-GUI L2 evaluates hierarchical instruction following and compositional reasoning, while OSWorld-G and OSWorld-G-R assess layout understanding, widget matching, and fine-grained interaction.

**Eval Metrics.** Following UGround (Gou et al., 2024), we adopt click-point prediction accuracy, denoted as Grounding, as the evaluation metric. A prediction is considered correct when the predicted interaction point falls within the ground-truth target region. We report the average Grounding accuracy over all evaluation samples.

## E Discussion on Reward Hacking Risk

Although the policy and verifier are instantiated from the same underlying model, they operate with asymmetric information. During action generation, the policy receives the high-level instruction, current interface state $S_t$, and interaction history $H_t$, and predicts an action $a_t$ without access to the target action $a_t^*$. During reward computation, the verifier additionally receives $a_t^*$ and evaluates whether the predicted action is functionally consistent with the target under the current interface state.

This information asymmetry means that the reward depends on instance-specific target information that is unavailable to the policy at generation time, reducing the opportunity to exploit generic phrases

or fixed high-reward output patterns. In addition, the target actions are task-specific, which further limits simple textual shortcut strategies.

Nevertheless, asymmetric information does not eliminate reward hacking entirely. In particular, the verifier may still assign inappropriate rewards when it relies too heavily on superficial lexical similarity rather than functional equivalence. The failure cases discussed above illustrate this form of reward misalignment.

# F Policy Input and Output Formats

**Policy Prompt for Training.** During semantic reinforcement fine-tuning, the policy receives the current GUI screenshot, the high-level task instruction, and the previous interaction history, and generates a natural-language description of the next GUI action. The complete policy prompt used during training is shown in Figure 9.

**Semantic Reward Prompt.** The generated semantic action is evaluated using the semantic consistency prompt shown in Figure 10. The evaluator compares the predicted action with the target semantic action under the current GUI state and produces the semantic reward used for policy optimization.

**GUI Grounding Evaluation Prompt.** At test time, we evaluate the same trained policy on GUI grounding benchmarks using the coordinate-prediction prompt shown in Figure 11. The transition from semantic-action generation during training to coordinate prediction during evaluation is implemented solely by changing the prompt. No additional fine-tuning, parameter modification, parser, grounding module, or semantic-to-coordinate conversion component is introduced.

**Robustness to Semantic Verification Prompts.** We further examine the sensitivity of semantic reward to the prompt used for semantic verification. In addition to the original functional-equivalence prompt, we construct several variants that emphasize different aspects of action consistency, including action type, screenshot-grounded reasoning, conservative matching, and paraphrase robustness. As shown in Table 9, the original prompt achieves the best average score of 59.0, while most alternative formulations remain close at 58.6–58.8. This relatively small variation suggests that the effectiveness of semantic reward does not depend on a particular prompt wording, but generalizes across different formulations of intent-level verification. Detailed prompt designs are provided in Figures 12–15.

At the same time, more substantial changes to the verification criterion can reduce performance, with the lowest variant reaching 57.9. This indicates that the semantic evaluator is more sensitive to how functional equivalence is defined than to superficial prompt phrasing. Overall, the results demonstrate reasonable robustness to prompt design while highlighting the importance of preserving the intended notion of semantic action equivalence.

**Policy Prompt - Train**

Role
You are a GUI agent that reasons about the next action to perform.
Inputs
- Current UI screenshot: `<image>`
- Instruction: `{instruction}`
- Previous action history: `{history}`

Task
Based on the screenshot, instruction, and action history, determine the specific action that should be performed next. Describe the action in short natural language, for example: `Click the Close button`, `Scroll down`, `Type the username`, or `Press Back`.
Output format
Provide a brief rationale inside `<think></think>` tags and the final concise action description inside `<answer></answer>` tags.
Example 1
`<think>` The instruction indicates that the review process is complete, so the next action is to close the review panel. `</think>`
`<answer>` Click the Close button `</answer>`
Example 2
`<think>` The user needs to enter a location name before continuing the search. `</think>`

`<answer>` Type the location name in the search box `</answer>`
Requirements
1. The final `<answer>` must be a short natural-language description of the next GUI action.
2. Do not output coordinates, executable code, or low-level action syntax.
3. Keep the final action concise, specific, and actionable, e.g., `Click Confirm`, `Press Back`, or `Select Settings`.

Required response:
`<think>...</think> <answer>...</answer>`

Figure 9: Prompt used for semantic action generation during reinforcement fine-tuning.

**Semantic Reward Prompt - Train**

You are judging whether a predicted UI action matches the ground-truth action at the intent level.
Focus on FUNCTIONAL EQUIVALENCE – whether they achieve the same user goal.
Consider:
- Action type (click, type, swipe, etc.)
- Target element's functional purpose (e.g., search initiation, navigation, submission)
- Input text meaning (if any)

Key principle: Two different UI elements can be functionally equivalent if they serve the same purpose.
Examples of functional equivalence:
- Clicking a search box vs. clicking a search icon $\rightarrow$ both initiate search $\rightarrow$ MATCH
- Clicking a "Send" button vs. pressing Enter $\rightarrow$ both submit the message $\rightarrow$ MATCH
- Typing in different input fields for the same information $\rightarrow$ MATCH

Guidelines:
1. If actions achieve the same user goal through functionally equivalent means, answer True.
2. If the user goals are clearly different, answer False.
3. Be lenient – focus on what the user wants to accomplish, not how precisely they do it.

Only output: True or False.

Figure 10: Prompt used to evaluate semantic consistency between predicted and ground-truth GUI action description.

**Policy Prompt - Evaluation**

You are a GUI grounding agent.
**Task**
Given a screenshot and the user's grounding instruction, accurately locate the target UI element specified by the instruction.
Carefully examine the screenshot and interpret the user's instruction. First, provide a concise reasoning process for identifying the target element, and then output its final coordinate.
**Output Format**
Return a JSON object containing the predicted coordinate in `[x,y]` format. The reasoning process should be enclosed within `<grounding_think></grounding_think>` tags, and the final coordinate should be enclosed within `<answer></answer>` tags.
<grounding_think>...</grounding_think>
<answer>
{"coordinate": [x,y]}
</answer>
**Input Instruction**
{instruction}

Figure 11: Prompt used for coordinate-based GUI grounding evaluation.

**Prompt V2**

Decide whether Sentence 2 describes a UI action that is semantically equivalent to Sentence 1.
Judge mainly by whether the action type is the same or practically interchangeable, whether the intended target or operation is the same, and whether the input text, if present, has the same meaning.
Do not require exact wording, exact coordinates, or exact formatting.
However, answer False if the predicted action changes the user intent, uses a different important input value, or operates on a clearly different UI target.
**Only output: True or False.**

Figure 12: Prompt V2 for Semantic Reward Prompt .

**Prompt V3**

You are given a screenshot and two UI action descriptions.
Determine whether the predicted action in Sentence 2 would accomplish the same task as the ground-truth action in Sentence 1 on this screen.
Use the screenshot to understand the functional role of UI elements, such as search, submit, back, close, menu, input field, or navigation.
Treat visually different controls as equivalent only when they clearly serve the same function on the screen.
Ignore small wording differences, coordinate differences, and harmless formatting differences.
Answer False if the action would interact with a different functional element or lead to a different user outcome.
**Only output: True or False.**

Figure 13: Prompt V3 for Semantic Reward Prompt .

**Prompt V4**

Check whether the predicted UI action is a correct match for the ground-truth action.
Answer True only when the two actions clearly share the same user intent, compatible action type, same target function, and same input text meaning when applicable.
**Be careful with near matches:**

- Similar-looking buttons are not necessarily equivalent.
- Different fields are not equivalent unless they request the same information.
- Different navigation destinations are not equivalent.
- Missing or changed input text should be False.

Ignore only formatting differences and minor wording differences that do not change the action.
**Only output: True or False.**

Figure 14: Prompt V4 for Semantic Reward Prompt .

**Prompt V5**

Compare the two UI actions as natural-language descriptions of user behavior.
Sentence 2 should be marked True if it is a paraphrase or functionally equivalent version of Sentence 1. Equivalent descriptions may use different words, omit unimportant UI details, or describe the same operation through a different but valid control.
**Focus on:**

- Same intended operation.
- Same functional target.
- Same typed content or same text meaning.
- Same final user-visible effect.

Ignore coordinates, bounding boxes, JSON/string formatting, capitalization, and minor phrasing differences.
Answer False when the predicted action would produce a different result or target a different purpose.
**Only output: True or False.**

Figure 15: Prompt V5 for Semantic Reward Prompt .