

# AN EFFICIENT TRAINING PIPELINE FOR REASONING GRAPHICAL USER INTERFACE AGENTS

**Georgios Pantazopoulos\***  
University of Edinburgh  
Heriot-Watt University  
gmp2000@hw.ac.uk

**Eda B. Özyiğit**  
The Alan Turing Institute  
eozyigit@turing.ac.uk

## ABSTRACT

Existing methods for developing Graphical User Interface agents rely on massive, noisy synthetic datasets obtained by extracting elements in the interface and then generating instructions with an oracle component. While effective, this pipeline often results in misaligned, low quality or repetitive instructions. This work introduces an efficient training pipeline that combines model-based data filtering with parameter-efficient fine-tuning. From 4.8M synthetic examples, 12K clean and diverse instances are curated by first identifying challenging cases, removing misaligned and then selecting a diverse set of multimodal instances. On this data, a 3B-parameter Vision-Language Model is trained under three regimes: supervised fine-tuning, chain-of-thought-augmented fine-tuning, and reinforcement learning via Group Relative Policy Optimization. Models trained with the filtered data and lightweight training strategies match or surpass larger baseline models trained with orders of magnitude more data. These results demonstrate that principled data curation and robust adaptation can rival large-scale training, enabling compact yet capable multimodal reasoning agents.<sup>1</sup>

## 1 INTRODUCTION

Recent advances in Large Language Models (LLMs) and Vision-Language Models (VLMs) enable Graphical User Interface (GUI) agents to execute natural-language instructions by controlling desktop, mobile, and web applications (Zhang et al., 2025a; Wang et al., 2024; Zheng et al., 2024a). The core requirements are: multimodal perception to fuse visual and textual signals; and action execution to perform precise interface operations with keyboard and mouse, commonly accompanied by planning and state tracking Wu et al. (2025). As such, reliable control begins with accurate grounding of instructions to specific GUI elements (e.g. buttons, icons) (Cheng et al., 2024; Li et al., 2025a; Nayak et al., 2025; Pantazopoulos & Özyiğit, 2025; Peng et al., 2024; Li et al., 2025b).

Most agents are fine-tuned VLMs variants on a large-scale synthetic corpora (Gou et al., 2025; Yang et al., 2024; Lin et al., 2025; Qin et al., 2025), where an off-the-shelf GUI element detector first produces candidate regions and then an oracle component that generates instructions tied to each region. This paradigm naturally leads to agents that perceive the environment exclusively through visual observations that match or surpass the performance of agents relying solely on text-based representations (Gou et al., 2025; Zhou et al., 2024; Gur et al., 2024; Deng et al., 2023). However, naively fine-tuning requires large volumes of diverse data often exhibiting weak generalisation in complex and high-resolution professional settings. Furthermore, synthetically creating training instances often result in noisy examples stemming from misaligned instruction-region pairs, low quality or repetitive instructions, and ambiguous regions often as a product of the detection system. With the recent success of reinforcement learning (RL) as a post-training stage, concurrent efforts (Luo et al., 2025; Lu et al., 2025; Yuan et al., 2025) focus on designing reward functions that promote more precise grounding and reasoning capabilities.

These observations suggest that high-volume synthetic data may not be necessary for developing capable GUI agents. In this work, we investigate whether a carefully curated training set can achieve

<sup>1</sup>Available here

comparable or superior performance. Figure 1 provides an overview of the stages of our filtering approach. Starting from 4.8M synthetic examples, we extract 12k challenging, and diverse demonstrations across desktop, web, and mobile applications. We augment these with chain-of-thought traces and explore supervised fine-tuning and Group Relative Policy Optimization (GRPO) (Guo et al., 2025) with sparse rewards. Our best model matches or exceeds the performance of similar and larger agents in both grounding and agentic evaluations.

## 2 RELATED WORK

**GUI agents** Early approaches for developing GUI agents, relied on structured representations such as HTML or accessibility (a11y) trees (Zhou et al., 2024; Gur et al., 2024; Deng et al., 2023) for grounding with labelled bounding boxes (Zhang et al., 2024; He et al., 2024). However, text-based representations of these environments are often noisy and/or incomplete (Gou et al., 2025) while at the same time, encoding the structure of the interface raises significant computational overhead as opposed to its corresponding visual observation (Zheng et al., 2024b). As a result, more recent work has shifted toward pixel-level, visually grounded agents that directly interpret screen-level observations, leveraging advances in visual encoders and multimodal reasoning to achieve robust performance in complex GUI environments (Cheng et al., 2024; Gou et al., 2025; Yang et al., 2024; Lin et al., 2025; Hong et al., 2024; Shaw et al., 2023; Wu et al., 2024). Building upon these developments, frameworks such as UI-TARS (Qin et al., 2025) and AgentS2 (Agashe et al., 2025) adopt modular or generalist-specialist architectures to improve reasoning, planning, and cross-platform generalisation (Bonatti et al., 2024; Xie et al., 2024). Nevertheless, the predominant reliance on supervised fine-tuning (SFT) Dong et al. (2024) across these systems presents enduring challenges, including substantial data requirements, limited adaptability to unseen interfaces, and suboptimal scalability in dynamic GUI contexts (Lu et al., 2025; Yuan et al., 2025). These limitations have motivated increasing interest in learning paradigms that can leverage interaction-based or self-supervised signals to achieve more flexible and data-efficient GUI understanding.

**Reinforcement Learning & GUI agents** Reinforcement Learning (RL) has recently emerged as a promising alternative to SFT for training GUI agents, offering a mechanism to optimise model behaviour through interaction-based feedback rather than static supervision Zhang et al. (2025b). Rule-based reinforcement fine-tuning (RFT) paradigms have demonstrated impressive generalisation across reasoning, code generation, and multimodal perception tasks by employing verifiable reward signals to guide model adaptation (Kim et al., 2025; Wang et al., 2025; Fan et al., 2025). Extending this paradigm to GUI environments, concurrent works (Lu et al., 2025; Luo et al., 2025) have pioneered the integration of RFT for high-level interaction reasoning and low-level grounding, achieving improved action prediction accuracy even under limited data regimes. These approaches illustrate the potential of RL-based methods to enhance both the contextual understanding and decision-making efficiency of GUI agents. Our work is aligned with aforementioned approaches, underscoring the promise of reinforcement-driven learning as a scalable and generalisable foundation for the next generation of intelligent GUI agents, capable of bridging perception, reasoning, and control within unified multimodal frameworks.

## 3 METHOD

### 3.1 DATASET & FILTERING PIPELINE

This section describes our data filtering pipeline, illustrated in Figure 1. We start from a collection of existing open-source GUI grounding datasets. ShowUI-Desktop (Lin et al., 2025), and AriaUI (Yang et al., 2024) resulting in 4.8M grounding instances covering desktop, web, and mobile interfaces. We then employ a filtering scheme to select challenging, diverse, and clean examples as follows:

**Selecting Challenging Instances** We utilise Qwen-2.5-VL-3B (Bai et al., 2025) in a zero-shot setting and generate bounding box predictions for each training example. If the prediction for an example is correct, we consider this example as easy otherwise the example is considered hard and include it in our dataset. The easy instances are used to train a ranking model that identifies misaligned instruction-region pairs as described below.

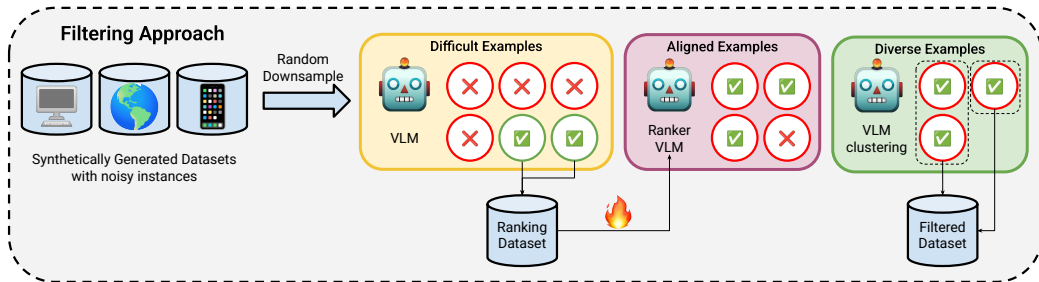


Figure 1: Overview of our filtering approach. We begin with a pool of noisy, GUI examples drawn from desktop, web, and mobile interfaces. A base VLM scores candidates, partitioning the pool into easy and challenging cases. We train a ranking model on the easy subset to decide whether an instruction aligns with a candidate region in the interface. The ranker is then applied to the challenging subset to retain only aligned instances. We then cluster the embeddings from a single forward pass of the VLM and select a diverse set of challenging examples.

**Determining Misaligned Instructions** Many training resources are created using an automatic setup, either by (i) labelling GUI elements via set-of-marks (SoM) (Yang et al., 2023) and then using a VLM to create references on each element; or (ii) using text-based representations by parsing the raw HTML or accessibility (ally) trees (Zhou et al., 2024; Deng et al., 2023). However, due to noise in the collection process, the hard examples obtained from the previous step may not in fact have aligned instruction-bounding box pairs. To detect false-positive examples, we train a ranking model based on Qwen-2.5-VL-3B that determines whether an instruction-bounding box pair is valid. This ranking model is trained using the easy examples from the previous step as we hypothesize that they contain aligned annotations given correct zero-shot predictions (see Section 4.1).

**Identifying Diverse Training Examples** An additional shortcoming of the existing resources is that the automatic data creation process results in repetitive examples. For instance, given a GUI representing a spreadsheet, many approaches relying in a combination SoM+VLM to create annotations, yields multiple examples that point to cell entities in the GUI often with limited language variation (e.g. “Click on Cell F42”). To select diverse and representative training examples, we employ one forward pass and obtain the last hidden state. We then cluster the hidden states using PCA and k-means then obtain the examples closer to each centroid.

**Post-Processing** Finally, we use GPT-4o to further select examples where the instruction clearly points to the correct GUI element without any notion of ambiguity. At this stage, we also manually verified the correctness of our filtered examples. We apply the above steps sequentially on the initial set of 4.8M examples. With this process, we curate a clean version with 12K examples which corresponds to roughly 0.25% of the original data.

**Generating Chain-of-thought-traces** We augment the filtered examples with chain-of-thought traces using GPT-4o. Given the grounding instance, we highlight the correct region and prompt the model to provide an explanation for the instruction. The goal of this step is to provide traces in the form explanations that help the agent identify the correct receptacle within the GUI.

We provide additional information regarding our filtering steps in Section A.

### 3.2 MODEL

**Model Architecture** In all of our experiments, we employ the instruction-tuned variant Qwen-2.5-VL-3B as our base model as it performs exceptionally well on grounding multimodal tasks and supports high-resolution image input both of which are necessary prerequisites for GUI agents. With regards to the image resolution, typical GUI screenshots are much larger than natural images leading to significant computational overhead. As a result, we resize images by preserving their aspect ratio to  $256 \times 28 \times 28$  minimum and  $1080 \times 28 \times 28$  maximum pixels, respectively.

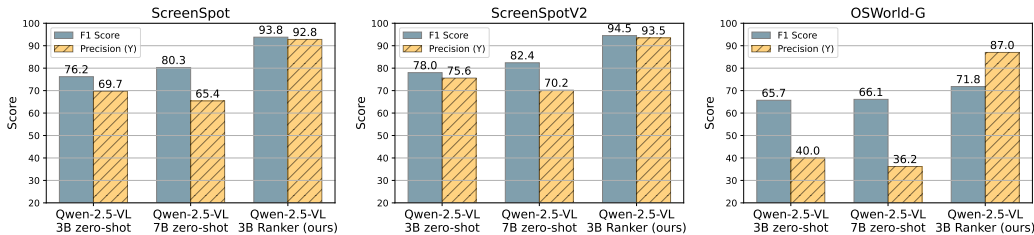


Figure 2: Ranking performance on ScreenSpot, ScreenSpotv2 and OSworld-G. We compare our ranking model against the Qwen-2.5-VL 3B/7B variants on identifying misaligned image-instruction-bbox triplets. Our model consistently outperforms both variants across all three benchmarks in terms of F1-score and Precision shown here in blue and yellow striped bars respectively.

Our goal is to explore various training regimes for efficiently developing GUI agents. For this purpose, we resort to parameter-efficient methods for training, with low-rank adaptation (LoRA) (Hu et al., 2022) being the most prominent approach. Within VLMs, the design choice of where to apply LoRA adapters depends on various factors. Applying adapters on the vision encoder and/or the visual connector can be particularly effective especially when the base model has not been exposed to the target visual domain. Conversely, language adapters can be effective when the base model is not familiar with the output text distribution. Finally, applying adapters to all components can be effective when the base model has not been significantly exposed to the target domain. Following prior approaches (Laurençon et al., 2024), we explore the impact of LoRA adapters on the visual and the language backbone of a GUI-specific model. More specifically, we apply adapters on all linear modules and explore three variants: 1) adapters on the visual backbone and the connector; 2) adapters on the language backbone; and 3) adapters on all components of the VLM.

**Training Regime** With regards to the training objectives, we compare three standard methods: 1) SFT, where the model directly predicts the candidate element in the interface; 2) Supervised fine-tuning with chain-of-thought traces (SFT+CoT) where the model first derives a plausible explanation and then provides the candidate element; and 3) RL with verifiable rewards using GRPO (Shao et al., 2024). Throughout our experiments, we apply eight different LoRA configurations for the vision, language, and the joint model resulting in 24 different runs for each training regime. The best candidate for each training regime is determined by the grounding performance on GUI interfaces (Section 4.2), which is then used for end-to-end agentic evaluation (Section 4.3). For GRPO, we formulate our reward signal based on three binary values: 1) *Format-based*, where the model receives +1 if the output follows the desired format; 2) *Solution-based*, where the model receives +1 if the center of the predicted bounding box falls within the ground-truth region; and 3) *Length-based* as from preliminary experiments we observed that the model collapsed resulting in gibberish generations. For this purpose, we provide +1 credit if the length of the output does not exceed 100 tokens. The final reward is the sum of the above values without any additional scaling.

## 4 EXPERIMENTS

We closely follow the SeeAct-V framework (Gou et al., 2025), where an agent uses only screenshots as environmental observation. SeeAct-V has two stages: planning and grounding, both handled by a VLM. We first evaluate our ranker’s quality (Section 4.1), demonstrating its effectiveness at retrieving aligned instruction-bounding box instances. In Section 4.2, we assess grounding capabilities under both human-generated and planner-generated instructions, testing end-to-end effectiveness within SeeAct-V. Finally, we integrate our agent into the SeeAct-V framework for offline evaluation (Section 4.3), where it accepts instructions across web, mobile, and desktop applications and predicts low-level actions to satisfy user instructions.

Model	ScreenSpot								ScreenSpotv2							
	Mobile		Desktop		Web		Average		Mobile		Desktop		Web		Average	
	Text	Image	Text	Image	Text	Image	Mic	Mac	Text	Image	Text	Image	Text	Image	Micro	Macro
Fuyu 8B (2023)	41.0	1.3	33.0	3.6	33.9	4.4	-	19.5	-	-	-	-	-	-	-	-
CogAgent 18B (2024)	67.0	24.0	74.2	20.0	70.4	28.6	49.6	47.4	-	-	-	-	-	-	-	-
SeeClick 9.6B (2024)	78.0	52.0	72.2	30.0	55.7	32.5	55.8	53.4	78.4	50.7	70.1	29.3	55.2	32.5	55.1	52.7
OmniParser (2024)	93.9	57.0	91.3	63.6	81.3	51.0	-	73.0	-	-	-	-	-	-	-	-
ShowUI-G 2B (2025)	92.3	75.5	76.3	61.1	81.7	63.6	-	75.1	-	-	-	-	-	-	-	-
UGround-V1 2B(2025)	89.4	72.0	88.7	65.7	81.3	68.9	-	77.7	-	-	-	-	-	-	-	-
UGround-V1 7B (2025)	93.0	79.9	93.8	76.4	90.9	84.0	-	86.3	-	-	-	-	-	-	-	-
AriaUI 3.9B (2024)	92.3	73.8	93.3	64.3	86.5	76.2	82.4	81.0	-	-	-	-	-	-	-	-
OS-Atlas-Base 4B (2024)	85.7	58.5	72.2	45.7	82.6	63.1	70.1	68.0	87.2	59.7	72.7	46.4	85.9	63.0	69.2	71.9
OS-Atlas-Base 7B (2024)	93.0	72.9	91.7	62.9	90.9	74.3	82.4	80.1	95.2	75.8	90.7	63.6	90.6	77.3	84.1	82.0
<i>SFT:</i>																
GUI-Qwen (V)	91.9	67.2	89.7	57.1	83.9	62.1	77.0	75.3	96.9	73.8	96.1	73.8	85.4	68.4	83.2	82.4
GUI-Qwen (L)	93.4	71.2	95.4	62.1	86.1	70.4	81.2	79.8	89.1	66.5	88.9	59.5	84.6	60.8	76.5	74.9
GUI-Qwen (VL)	95.6	69.9	93.3	67.9	85.7	68.4	81.4	80.1	96.9	79.1	95.6	69.0	85.4	71.7	84.0	82.9
<i>SFT+CoT:</i>																
GUI-Qwen (V)	86.4	63.8	86.6	57.9	83.9	59.2	74.4	73.0	89.1	66.5	88.9	59.5	84.6	60.8	76.5	74.9
GUI-Qwen (L)	94.5	75.5	93.3	62.9	86.5	67.5	81.6	80.0	96.9	79.1	95.6	69.0	85.4	71.7	84.0	82.9
GUI-Qwen (VL)	94.5	71.2	93.8	69.3	87.0	65.5	81.4	80.2	97.3	73.8	96.1	74.6	85.7	68.4	83.4	82.6
<i>GRPO</i>																
GUI-Qwen (VL)	<b>97.4</b>	<b>78.2</b>	<b>93.8</b>	<b>65.0</b>	<b>89.1</b>	<b>69.9</b>	<b>83.9</b>	<b>82.2</b>	<b>99.2</b>	<b>84.3</b>	<b>95.0</b>	<b>69.0</b>	<b>91.4</b>	<b>71.7</b>	<b>86.6</b>	<b>85.1</b>

Table 1: Standard Setting: Grounding accuracy on ScreenSpot and ScreenSpotv2. (V/L/VL) denotes LoRA parameters on visual, language, or both backbones.

#### 4.1 RANKER MODEL

The ranking model is trained on instances where the base model correctly identifies the target element in the interface. More specifically, given an image-instruction-bbox triplet the ranker predicts whether the instruction matches the element in the image indicated by the bounding box.

**Creating negative annotations** We derive negative image-instruction-bbox triplets by grouping instances from the same image rather than random sampling, which would yield overly easy negatives and bias the ranking model toward positives. During training, we create positive triplets by keeping original annotations with 50% probability, while negative examples swap the bounding box with a different candidate from the same image.

**Results** We provide intermediate evaluations that substantiate the effectiveness of our ranking model by converting ScreenSpot (Cheng et al., 2024), Screenspotv2 (Wu et al., 2024) and Osworld-G (Xie et al., 2024), three popular GUI grounding benchmarks, that we converted into a binary classification, consistent with our training protocol. If an image contains  $n$  examples, for each example  $i$ , we create  $n - 1$  negative examples by using the  $i$ -th instruction and all  $m$  ( $m \neq i$ ) bounding boxes from all remaining examples.

We note that in this type of evaluation 55.7% and 71.5% of the examples correspond to negative labels for ScreenSpot and Osworld-G, respectively. For this reason, we primarily focus on the overall F1-score and the precision of the positive class as we are interested in measures that quantify the correctness of the filtering approach for true-positive instances. As shown in Figure 2, our ranking model outperforms the 3B/7B zero-shot variants, demonstrating its effectiveness within our data filtering pipeline. We also report overall performance metrics in Section B.1.

#### 4.2 GUI VISUAL GROUNDING

We evaluate the grounding performance of our model on ScreenSpot (Cheng et al., 2024) and ScreenSpotv2 (Wu et al., 2024), which are specifically designed for visual grounding on GUIs. Both benchmarks consist of 1,272 single-step instructions and the corresponding bounding boxes of the target elements across mobile (e.g. iOS and Android), desktop (e.g. macOS and Windows), and web environments. The target elements encompass diverse GUI component types, including text-based elements, icons (e.g. trash can icons), and widgets (e.g. to-do lists).

Model	Mobile		Desktop		Web		Average	
	Text	Icon	Text	Icon	Text	Icon	Micro	Macro
SeeClick 9.6B (Cheng et al., 2024)	81.0	59.8	69.6	33.6	43.9	26.2	-	52.3
UGround-V1 2B (Gou et al., 2025)	94.1	77.7	92.8	63.6	90.0	70.9	-	81.5
UGround-V1 7B (Gou et al., 2025)	94.1	79.9	93.3	73.6	89.6	73.3	-	84.0
OS-Atlas-Base 4B (Wu et al., 2024)	94.1	73.8	77.8	47.1	86.5	65.5	76.8	74.1
OS-Atlas-Base 7B (Wu et al., 2024)	93.8	79.9	90.2	66.4	92.6	79.1	85.1	83.7
<i>SFT:</i>								
GUI-Qwen (V)	92.3	66.4	90.2	57.9	83.5	61.7	77.0	75.3
GUI-Qwen (L)	94.1	71.2	94.9	63.6	87.0	67.5	81.1	79.7
GUI-Qwen (VL)	96.0	68.6	93.8	66.4	86.1	67.0	81.0	79.6
<i>SFT+CoT:</i>								
GUI-Qwen (V)	87.6	65.5	86.6	58.6	85.2	61.7	75.6	74.2
GUI-Qwen (L)	95.6	76.0	93.8	62.9	86.5	65.5	81.7	80.1
GUI-Qwen (VL)	92.7	70.7	93.3	59.3	83.5	66.5	79.3	77.7
<i>GRPO</i>								
GUI-Qwen (VL)	<b>97.1</b>	<b>77.7</b>	<b>93.8</b>	<b>63.6</b>	<b>88.7</b>	<b>70.4</b>	<b>83.6</b>	<b>81.9</b>

Table 2: Agent Setting: Grounding accuracy on ScreenSpot with planner-generated referring expressions. (V/L/VL) denotes LoRA parameters on visual, language, or both backbones.

Furthermore, following prior work (Gou et al., 2025), we focus on two settings: 1) Standard Setting, where the instructions are written by human annotations targeting a functional description of the element (e.g. simply “close” to refer to the “X” button that closes a window); and 2) Agent Setting, where the grounding model accepts instructions from a planning model that focus not only functional descriptions but also visual and positional information. To isolate the effects of instruction quality from grounding capability, we employ the instructions generated by Uground (Gou et al., 2025).

**Results** We report the grounding accuracy in the Standard and Agent setting in Table 1 and Table 2, respectively. Among the SFT variants, GUI-Qwen (VL) with LoRA parameters on both visual and language backbones performs on par or even surpass significantly larger models. Surprisingly, the chain-of-thought reasoning shows mixed results depending on the backbone configuration. The combination of CoT+LoRA adapters on the vision backbone results in performance degradation compared to the SFT variant. However, GUI-Qwen (L) with SFT+CoT improves over the base SFT version, while GUI-Qwen (VL) with CoT maintains comparable performance to its SFT counterpart. We hypothesise that this partially stems from discrepancy between the format of the trace and the outputs of the base model. Most importantly, our GRPO-optimised GUI-Qwen (VL) model achieves state-of-the-art results across both benchmarks. Similar trends can be observed in the case of the Agent setting, with planner-based instructions.

### 4.3 OFFLINE AGENT EVALUATION

In this section, we provide experiments regarding offline agent evaluation on MultimodalMind2Web, Android Control, and OmniAct encompassing web, mobile, and desktop applications following their standard evaluation protocol (Zheng et al., 2024b; Li et al., 2024; Kapoor et al., 2024) described below. Importantly, we adopt the planner-generated instructions from Uground (Gou et al., 2025) in each benchmark to decouple the effect of the quality of the planner from the downstream performance. As such, our comparison is focused mostly with the Uground family of models.

**Web: MultimodalMind2Web** We use Multimodal-Mind2Web (Zheng et al., 2024b), which extends the original Mind2Web (Deng et al., 2023) benchmark using multimodal information for web tasks. These tasks are crowdsourced ensuring they represent genuine, practical needs that real users would encounter on these platforms. The test split consists of 1,013 tasks spanning over 100 different websites. Each task is defined by a high-level task instruction and a sequence of actions, with a screenshot of the webpage before each action, as the golden trajectory. We report the element accuracy, i.e. the percentage of correctly selected elements, and omit operation scores because they are orthogonal to grounding comparisons.

Model	Cross-Task	Cross-Website	Cross-Domain	Macro Average
SeeClick 9.6B (2024)	32.1	33.1	33.5	32.9
UGround-V1 2B (2025)	48.6	47.6	47.7	48.0
UGround-V1 7B (2025)	50.7	48.1	48.5	49.1
<i>SFT:</i>				
GUI-Qwen (V)	48.2	46.5	45.5	46.7
GUI-Qwen (L)	49.3	47.1	46.3	47.6
GUI-Qwen (VL)	49.1	46.4	46.2	47.2
<i>SFT+CoT:</i>				
GUI-Qwen (V)	47.8	43.2	45.1	45.4
GUI-Qwen (L)	48.3	46.7	46.1	47.0
GUI-Qwen (VL)	47.7	46.2	46.5	46.8
<i>GRPO</i>				
GUI-Qwen (VL)	<b>48.9</b>	<b>47.2</b>	<b>48.2</b>	<b>48.1</b>

Table 3: Element accuracy on Multimodal-Mind2Web. (V/L/VL) denotes LoRA parameters on visual, language, or both backbones.

Model	High	Low
SeeClick 9.6B (2024)	41.8	52.8
UGround-V1 <sup>†</sup> 2B (2025)	50.0	65.0
UGround-V1 <sup>†</sup> 7B (2025)	49.8	66.2
<i>SFT:</i>		
GUI-Qwen (V)	34.2	60.0
GUI-Qwen (L)	34.8	62.6
GUI-Qwen (VL)	36.2	63.8
<i>SFT+CoT:</i>		
GUI-Qwen (V)	39.0	61.6
GUI-Qwen (L)	36.0	63.2
GUI-Qwen (VL)	37.4	62.0
<i>GRPO</i>		
GUI-Qwen (VL)	<b>44.0</b>	<b>66.0</b>

Table 4: Step accuracy (SA) on AndroidControl over 500 random actions from the test split. Models<sup>†</sup> are fine-tuned for this task.

Model	Action Scores
SeeClick 9.6B (2024)	29.6
UGround-V1 2B (2025)	32.9
UGround-V1 7B (2025)	34.0
<i>SFT:</i>	
GUI-Qwen (V)	33.8
GUI-Qwen (L)	34.2
GUI-Qwen (VL)	34.0
<i>SFT+CoT:</i>	
GUI-Qwen (V)	33.2
GUI-Qwen (L)	<b>34.3</b>
GUI-Qwen (VL)	34.1
<i>GRPO</i>	
GUI-Qwen (VL)	33.8

Table 5: Action scores (AS) on OmniACT.

**Mobile: Android Control** For mobile applications, we use AndroidControl (Li et al., 2024), a dataset with 15k tasks covering 833 apps. Each task is also densely annotated with low-level step by step instructions that correspond to an action. Following prior work (Gou et al., 2025; Li et al., 2024), we use a subset of 500 random steps from the original set and adopt two task settings: 1) high-level control where only the high-level intent is provided; and 2) low-level control, where both the high-level intent and the corresponding low-level instruction for each timestep are available. In both settings, we report the standard step-wise accuracy where a step is considered successful only if all the predicted actions, elements, and arguments (if applicable) are correct.

**Desktop: OmniAct** Finally, we evaluate our model on OmniACT (Kapoor et al., 2024), a collection of 9,802 tasks covering 38 desktop applications and 27 websites across different platforms. Each of these tasks requires the agent to provide a PyAutoGUI script – a sequence of actions to complete the task on a single screenshot. We report the action score which measures how well a code snippet containing the correct action sequence can perform the task.

**Results** Tables 3, 4 and 5, show the results for Multimodal-Mind2Web, Android Control, and Omniact, where our model achieves competitive performance with larger models across web, mobile, and desktop domains. On the Multimodal-Mind2Web benchmark, our best-performing model achieves a macro average element accuracy of 48.1% using GRPO that is competitive with larger

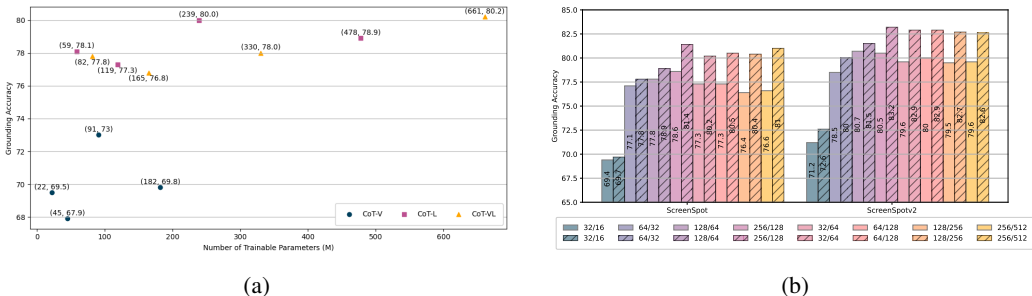


Figure 3: **(a)**: Performance of chain-of-thought models under different LoRA configurations. **(b)**: Data ablation results on. We re-train all LoRA configurations ( $r/\alpha$ ) for GUI-Qwen (VL) using the original data and the data from our filtering method. Bars with stripes (//) indicate training runs using our filtered data. Our filtering approach yields consistent improvements across all configurations.

models. Across training strategies, we observe that SFT yields strong baseline performance, with the language-only LoRA variant achieving 47.6% macro average—the best among SFT approaches. Interestingly, adding chain-of-thought reasoning (SFT+CoT) does not consistently improve performance, with most variants showing slight degradation compared to their SFT counterparts. On AndroidControl, CoT shows mixed results: SFT+CoT with visual-only LoRA achieves 39.0% on high-level control versus 36.2% for best SFT, but benefits vary by configuration. GRPO-trained GUI-Qwen (VL) achieves 44.0% on high-level and 66.0% on low-level tasks, substantially outperforming SFT baselines. Finally, in OmniACT, all of our models cluster around 33-34% action scores, with the SFT+CoT language-only variant achieving the highest score at 34.26% with GRPO providing minimal improvements, which could be explained by the quality of the planner (Gou et al., 2025), as planning errors often bottleneck the overall performance.

#### 4.4 PERFORMANCE ANALYSIS

**Which parameters of a non-thinking VLM can facilitate reasoning capabilities?** Recall that from the previous results (e.g. Table 1), applying LoRA adapters exclusively on the visual components of a VLM can yield performance degradation. Here, we investigate this further by comparing the performance of our model under CoT training and four different LoRA adapters on the visual, language, or both backbones. Figure 3a shows the performance of these runs as a function of the number of training parameters. We observe that by controlling the total number of training parameters, injecting LoRA adapters on the language backbone yields substantial improvement compared to the visual backbone which could be attributed to differences in the expected output text distribution. This is in line with our previous findings as well as prior research in the field (Laurençon et al., 2024). Finally, we observe moderate improvements in the case of vision-and-language over language-only adapters particularly for high-capacity configurations.

**What type of generations are preferred by GUI reasoning models?** We initially assumed reasoning traces would guide bounding box prediction by providing descriptions of the interface and/or the target element. While models with such traces showed improved grounding (especially with LoRA on the language backbone), GRPO-trained models achieved substantial improvements without imposing strict trace structure. This motivated us to identify what trace types these models prefer. For this purpose, we manually reviewed 100 sampled predictions from ScreenSpot using the SFT+CoT and GRPO models. We observed that both models favoured short generations, which was expected given the short CoT training traces and GRPO’s reward penalizing long outputs. However, a notable distinction is that the model trained with GRPO try to repeat the entire or rephrase parts of the instruction and then explicitly attempt to refer to the target element (see examples in Table 9). This contrasts with our original hypothesis where the traces should provide a top-down summary before explicitly pointing to elements in the interface. While still interpretable, this reveals shallower reasoning focused on instruction alignment rather than the deeper visual understanding.

**Data Ablation** To validate the effectiveness of our approach, we conduct an ablation study with GUI-Qwen (VL) using the data before and after our filtering under supervised fine-tuning. We facilitate a fair comparison under controlled conditions by applying the same training hyper-parameters and LoRA configurations. Figure 3b illustrates the performance of our model on ScreenSpot and ScreenSpotv2, when using the original data and the data produced by our filtering approach. Across all settings, we observe consistent improvements when the model is trained using our own smaller but cleaner version of the original training dataset, demonstrating the value of our data curation.

## 5 CONCLUSION & LIMITATIONS

We developed an efficient training pipeline for GUI agents using curated data and lightweight fine-tuning. Our model-based filtering extracted high-quality, diverse examples from large synthetic datasets, reducing volume while improving accuracy. Our analysis highlights the combination of parameter-efficient adaptation, particularly when applied to the language backbone. Finally, by examination of reasoning traces we showed that current models use shallow instruction-alignment patterns rather than true visual abstraction, suggesting directions for improvement.

## 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. *arXiv preprint arXiv:2504.00906*, 2025. URL <https://arxiv.org/abs/2504.00906>.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. URL <https://arxiv.org/abs/2502.13923>.
- Rogério Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Buckner, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024. URL <https://arxiv.org/abs/2409.08264>.
- Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=xYGN086OWDH>.
- 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)*, pp. 9313–9332, 2024. URL <https://arxiv.org/abs/2401.10935>.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets\\_and\\_Benchmarks.html](https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html).
- Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition, 2024. URL <https://arxiv.org/abs/2310.05492>.
- Yue Fan, Xuehai He, Diji Yang, Kaizhi Zheng, Ching-Chen Kuo, Yuting Zheng, Sravana Jyothi Narayanaraju, Xinze Guan, and Xin Eric Wang. Grit: Teaching mllms to think with images. *arXiv preprint arXiv:2505.15879*, 2025. URL <https://arxiv.org/abs/2505.15879>.
- 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=kxnoqaisCT>.

- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=9JQtrumvg8>.
- Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6864–6890, 2024. URL <https://arxiv.org/abs/2401.13919>.
- Wenyi Hong, Weihang Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14281–14290, 2024. URL [https://openaccess.thecvf.com/content/CVPR2024/html/Hong\\_CogAgent\\_A\\_Visual\\_Language\\_Model\\_for\\_GUI\\_Agents\\_CVPR\\_2024\\_paper.html](https://openaccess.thecvf.com/content/CVPR2024/html/Hong_CogAgent_A_Visual_Language_Model_for_GUI_Agents_CVPR_2024_paper.html).
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. Omniaact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pp. 161–178. Springer, 2024. URL [https://link.springer.com/chapter/10.1007/978-3-031-73113-6\\_10](https://link.springer.com/chapter/10.1007/978-3-031-73113-6_10).
- Dongyoung Kim, Sumin Park, Huiwon Jang, Jinwoo Shin, Jaehyung Kim, and Younggyo Seo. Robot-rl: Reinforcement learning for enhanced embodied reasoning in robotics. *arXiv preprint arXiv:2506.00070*, 2025. URL <https://arxiv.org/abs/2506.00070>.
- Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*, 2024. URL <https://arxiv.org/abs/2405.02246>.
- Kaixin Li, Meng ziyang, 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 *Workshop on Reasoning and Planning for Large Language Models*, 2025a. URL <https://openreview.net/forum?id=XaKNDIAHas>.
- Ming Li, Ruiyi Zhang, Jian Chen, Chenguang Wang, Jiuxiang Gu, Yufan Zhou, Franck Dernoncourt, Wanrong Zhu, Tianyi Zhou, and Tong Sun. Towards visual text grounding of multimodal large language model. *arXiv preprint arXiv:2504.04974*, 2025b. URL <https://arxiv.org/abs/2504.04974>.
- Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37:92130–92154, 2024. URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/hash/a79f3ef3b445fd4659f44648f7ea8ffd-Abstract-Datasets\\_and\\_Benchmarks\\_Track.html](https://proceedings.neurips.cc/paper_files/paper/2024/hash/a79f3ef3b445fd4659f44648f7ea8ffd-Abstract-Datasets_and_Benchmarks_Track.html).
- Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for

- gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19498–19508, 2025. URL [https://openaccess.thecvf.com/content/CVPR2025/html/Lin\\_ShowUI\\_One\\_Vision-Language-Action\\_Model\\_for\\_GUI\\_Visual\\_Agent\\_CVPR\\_2025\\_paper.html](https://openaccess.thecvf.com/content/CVPR2025/html/Lin_ShowUI_One_Vision-Language-Action_Model_for_GUI_Visual_Agent_CVPR_2025_paper.html).
- Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024. URL <https://arxiv.org/abs/2408.00203>.
- Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanqing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025. URL <https://arxiv.org/abs/2503.21620>.
- Run Luo, Lu Wang, Wanwei He, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. URL <https://arxiv.org/abs/2504.10458>.
- Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A. Rodriguez, Montek Kalsi, Nicolas Chapados, M. Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, Perouz Taslakian, Spandana Gella, and Sai Rajeswar. UI-vision: A desktop-centric GUI benchmark for visual perception and interaction. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=5Rtj4mYH1C>.
- Georgios Pantazopoulos and Eda B. Özyiğit. Towards understanding visual grounding in visual language models. *arXiv preprint arXiv:2509.10345*, 2025. URL <https://arxiv.org/abs/2509.10345>.
- Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, Qixiang Ye, and Furu Wei. Grounding multimodal large language models to the world. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=1LmqxkfSIw>.
- 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. URL <https://arxiv.org/abs/2501.12326>.
- Bavishi Rohan, Elsen Erich, Hawthorne Curtis, Nye Maxwell, Odena Augustus, Somani Arushi, and Taşlılar Sağnak. Fuyu-8b: A multimodal architecture for ai agents, October 2023. URL <https://www.adept.ai/blog/fuyu-8b>.
- 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. URL <https://arxiv.org/abs/2402.03300>.
- Peter Shaw, Mandar Joshi, James Cohan, Jonathan Berant, Panupong Pasupat, Hexiang Hu, Urvashi Khandelwal, Kenton Lee, and Kristina N Toutanova. From pixels to ui actions: Learning to follow instructions via graphical user interfaces. *Advances in Neural Information Processing Systems*, 36:34354–34370, 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/hash/6c52a8a4fad9c9129c6e1d1745f2dfd0f-Abstract-Conference.html](https://proceedings.neurips.cc/paper_files/paper/2023/hash/6c52a8a4fad9c9129c6e1d1745f2dfd0f-Abstract-Conference.html).
- Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024. URL <https://arxiv.org/abs/2401.16158>.
- Song Wang, Gongfan Fang, Lingdong Kong, Xiangtai Li, Jianyun Xu, Sheng Yang, Qiang Li, Jianke Zhu, and Xinchao Wang. Pixelthink: Towards efficient chain-of-pixel reasoning. *arXiv preprint arXiv:2505.23727*, 2025. URL <https://arxiv.org/abs/2505.23727>.

- Qianhui Wu, Kanzhi Cheng, Rui Yang, Chaoyun Zhang, Jianwei Yang, Huiqiang Jiang, Jian Mu, Baolin Peng, Bo Qiao, Reuben Tan, Si Qin, Lars Liden, Qingwei Lin, Huan Zhang, Tong Zhang, Jianbing Zhang, Dongmei Zhang, and Jianfeng Gao. Gui-actor: Coordinate-free visual grounding for gui agents, 2025. URL <https://arxiv.org/abs/2506.03143>.
- 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. URL <https://arxiv.org/abs/2410.23218>.
- Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024. URL [https://proceedings.neurips.cc/paper\\_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets\\_and\\_Benchmarks\\_Track.html](https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html).
- Jianwei Yang, Hao Zhang, Feng Li, Xuayan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023. URL <https://arxiv.org/abs/2310.11441>.
- 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*, 2024. URL <https://arxiv.org/pdf/2410.05993>.
- 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. URL <https://www.arxiv.org/pdf/2505.12370>.
- Chaoyun Zhang, Liqun Li, Shilin He, Xu Zhang, Bo Qiao, Si Qin, Minghua Ma, Yu Kang, Qingwei Lin, Saravan Rajmohan, et al. Ufo: A ui-focused agent for windows os interaction. *arXiv preprint arXiv:2402.07939*, 2024. URL <https://arxiv.org/abs/2402.07939>.
- Chaoyun Zhang, He Huang, Chiming Ni, Jian Mu, Si Qin, Shilin He, Lu Wang, Fangkai Yang, Pu Zhao, Chao Du, Liqun Li, Yu Kang, Zhao Jiang, Suzhen Zheng, Rujia Wang, Jiaxu Qian, Minghua Ma, Jian-Guang Lou, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Ufo2: The desktop agentos, 2025a. URL <https://arxiv.org/abs/2504.14603>.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Michael Littman, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for llms: A survey, 2025b. URL <https://arxiv.org/abs/2509.02547>.
- Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024a. URL <https://arxiv.org/abs/2401.01614>.
- Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *arXiv preprint arXiv: 2401.01614*, 2024b. URL <https://openreview.net/forum?id=piecKJ2D1B>.
- Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=oKn9c6ytLx>.

**Prompt for Generating Chain-of-Thought Traces**

**Objective:**  
 Your task is to provide guidance on a visual grounding task, where given an image of a graphical user interface, and an instruction, the objective is to provide the correct bounding box that matches the instruction.  
 In the image that you are given there is a red ground truth bounding box and the instruction: `<instruction>`.

Below are some examples showing expected responses:

1. **Example 1:**
  - Instruction: ‘Close the current window’.
  - Response: The image features an open Firefox browser window. To close the current window, click on ‘X’ icon on the top-right corner.
2. **Example 2:** For each translated question-answer pair, check for errors. For example you can identify:
  - Instruction: ‘Use the “Insert” button to add code’.
  - Response: The image shows a LibreOffice document, the ‘Insert’ button is featured on the top menu next to the ‘View’ and ‘Format’ options.
3. **Example 3:**
  - Instruction: ‘Select Lohit Assamese font’.
  - Response: The image displays a character formatting dialog in LibreOffice Writer. The ‘Lohit Assamese’ font is located under the family options.

You need to guide the user by an explanation that allows the user to identify the red bounding box.

**Criteria for explanation:**

- Succinct using 2 sentences at most.
- Avoid explicitly mentioning or referring to the highlighted red bounding box as the user needs to infer it directly from your explanation.
- Overall short summary of the ui.
- If the bounding box covers an element then provide a description tailored to the input instruction of the red groundtruth bounding box. If the bounding box covers plain text in the interface, refer to that text. **DO NOT MENTION THAT THE BOUNDING BOX IS ALREADY HIGHLIGHTED.**
- Be direct but not in the second person.

**Response format:**  
 You must return your response in a json format: `{"response": <response>}`, where `<response>` is your explanation.

Figure 4: Guidelines generating chain-of-thought traces.

## A DATA

### A.1 FILTERING PIPELINE

**Task Difficulty** To identify challenging instances for model training, we adopt the evaluation guidelines established in the compute use and mobile agent benchmarks from Qwen-2.5-VL<sup>2</sup>. Our approach involves generating a single zero-shot prediction for each training example from the original AriaUI and ShowUI datasets. An example is classified as challenging if the predicted bounding box fails to accurately localize the target element—specifically, when the center point of the predicted box falls outside the ground truth target region. This difficulty criterion allows us to prioritize training on instances where the model initially struggles, thereby focusing learning on the most informative examples that can drive performance improvements.

<sup>2</sup><https://github.com/QwenLM/Qwen3-VL/tree/main>

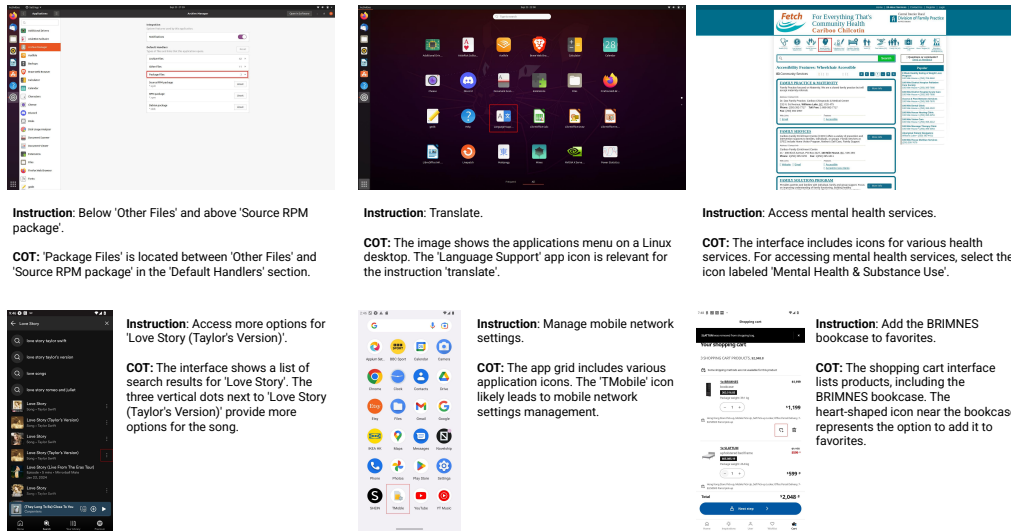


Figure 5: Examples of extracted chain-of-thought traces from desktop, web, and mobile interfaces. For clarity the correct bounding box is already drawn in the interface. Note that the instruction can target an element by describing its position, or point to an icon with underlying semantics. The goal of the trace here is to provide guidance by describing the target element or the semantics of that icon.

**Diverse Training Examples** To ensure diversity in our selected training examples, we employ a representation-based clustering strategy that considers both visual and textual modalities. Given a GUI screenshot and its corresponding instruction, we perform a forward pass through the model and extract the final hidden state from the last layer. This latent representation serves as a comprehensive encoding of the multimodal sequence, capturing the semantic relationship between the visual interface and the textual instruction. Following established practices in embedding analysis (Cai et al., 2021), we first apply Principal Component Analysis (PCA) to reduce the dimensionality of each example’s representation to 768 dimensions, which helps mitigate the curse of dimensionality and emphasizes the most significant variance in the data. Subsequently, we employ k-means clustering with the number of clusters set to 10% of the initial dataset size. From each cluster, we select the single example whose representation lies closest to the centroid, ensuring that our final training set spans the full diversity of the input space. It is worth noting that we applied this clustering procedure exclusively to the AriaUI benchmark, as it contains a substantially larger number of examples compared to ShowUI, making it more suitable for meaningful cluster-based sampling while ShowUI’s smaller scale allows for more direct selection strategies.

## A.2 CHAIN-OF-THOUGHT TRACES

We extract reasoning traces in the form of explanations for each filtered instance using GPT-4o. Figure 4 shows the full prompt for extracting these traces. The objective of this step is to provide guidance for the fine-tuning model when deriving the solution. During extraction, we use SoM to label the corresponding region and provide three in-context examples that we manually selected. We iterated over the generation prompt as during preliminary extraction we observed that the model used the prolonged generations and often referred to the SoM directly essentially providing the solution on the image. Figure 5 illustrates examples of extracted traces across desktop, web, and mobile interfaces.

## A.3 RANKER DATASET

The ranker model is trained to predict whether the instruction/bounding box match the element in the image indicated by the bounding box. As already mentioned, we used the images from the easy set of examples (in terms of difficulty of predicting the correct region). We include images with at

Hyperparameter	Ranker	SFT	SFT+CoT	GRPO
global batch size	64	{16, 32}	{16, 32}	{16, 32}
num epochs	2	{1, 2, 4}	{1, 2, 4}	{1, 2, 4}
lr	{1e-5, 5e-5}	{1e-6, 5e-6, 1e-5}	{1e-6, 5e-6, 1e-5}	{1e-6, 5e-6, 1e-5}
lr schedule	cosine decay	cosine decay	cosine decay	cosine decay
lr warmup	0.1	0.1	0.1	0.1
number of epochs	2	{1, 2, 4}	{1, 2, 4}	{1, 2, 4}
optimizer	AdamW	AdamW	AdamW	AdamW
adam betas	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
adam epsilon	1e-8	1e-8	1e-8	1e-8
weight decay	0.1	0.1	0.1	0.1
DeepSpeed Stage	3	3	3	3
LoRA modules	-	all linear layers V/L/VL	all linear layers V/L/VL	all linear layers VL
LoRA rank/alpha	-	{r=(32, 64, 128, 256), alpha=(0.5r, 2r)}		
min pixels	3136	3136	3136	3136
max pixels	{4028160, 12845056}	846720	846720	846720
num generations	-	-	-	16
temperature	-	-	-	1
top_p	-	-	-	1.0
top_k	-	-	-	-
KL beta	-	-	-	0
reward scaling	-	-	-	1

Table 6: Hyperparameters used to train all models. For LoRA adapters V: applied on only the vision encoder and the connector, L: applied on only the language backbone, VL: applied on the entire model including the vision encoder, connector, and the language backbone.

least  $M$  instruction/bounding box pairs Given that the AriaUI contains significantly more instances than ShowUI, we set  $M = 5$  for AriaUI and  $M = 1$  for ShowUI desktop, respectively. With this process, we create 15k instances corresponding to 130k positive image-instruction-bbox triplets.

## B EXPERIMENTS

**Implementation Details** Table 6 shows the training hyperparameters for all of our models. Note that for a fair comparison, we use the same set of values for each training regime. To determine the best LoRA configuration in each experiment, we selected the model with the highest performance on ScreenSpot. Across all settings, we observed that the lower learning rate yielded the best performance in our case. Furthermore, we note that in the case of SFT with chain-of-thought, the length of the traces can vary significantly. All experiments were conducted using  $2 \times$  A100 80GB.

### B.1 RANKER

Table 7, illustrate the performance of our ranking model on ScreenSpot, ScreenSpotv2 and Osworld-G relative to 3B, and 7B variants. Figure 6, the performance in terms of F1-score of the same models across different domains within ScreenSpot and ScreenSpotv2. Additionally, Table 8 shows the detailed performance across categories within Osworld-G.

### B.2 SENSITIVITY OF RL

Despite outperforming both SFT and SFT/CoT regimes, GRPO training exhibits significant sensitivity to hyperparameters. Figure 7 illustrates the reward curves for four different LoRA configurations throughout training. While all models quickly learn to satisfy format and length constraints from the earliest training stages, we observe catastrophic model collapse in the configuration with the highest number of trainable parameters. Specifically, this configuration initially demonstrates superior performance compared to models with fewer trainable parameters during the first 250 training steps. However, after approximately 500 steps, the model collapses abruptly, yielding extremely low solution rewards. From this point until 1,000 updates, the model attempts recovery but fails to restore performance, ultimately producing zero solution rewards for the remainder of training. We hypoth-

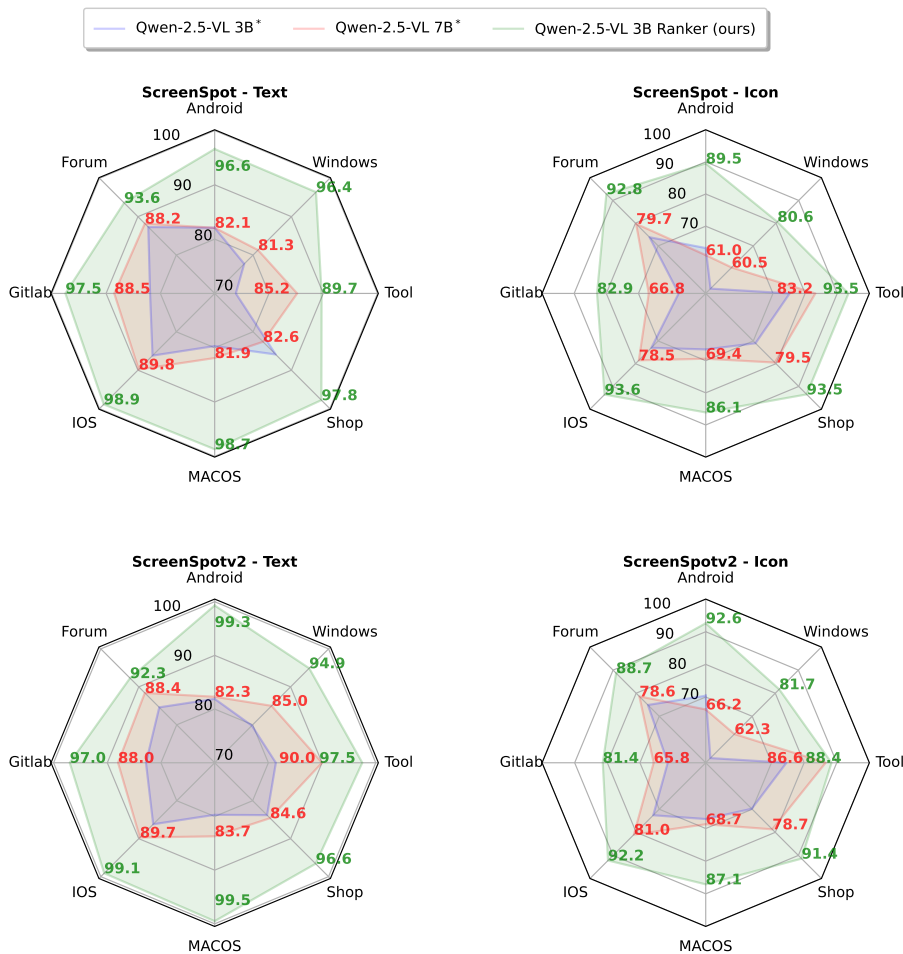


Figure 6: Ranking performance across all domains within ScreenSpot and ScreenSpotv2. \* denotes zero-shot evaluation.



Figure 7: Reward curves for four different LoRA configurations.

esise that this collapse correlates with the interplay between the number of trainable parameters and the reward function’s design. Since format and length rewards saturate early in training, the additional model capacity may be exploited to game the solution component in pursuit of total reward maximisation. This behaviour could potentially be mitigated by increasing the Kullback-Leibler (KL) penalty coefficient to constrain divergence from the base model, or rescaling the solution reward disproportionately to rewards related to the structure of the output.

Model	F1	Acc	Prec	Rec	F1 (Y)	F1 (N)	Prec (Y)	Prec (N)	Rec (Y)	Rec (N)
ScreenSpot										
Qwen-2.5-VL 3B*	76.2	77.6	76.1	76.3	70.5	81.9	69.7	82.5	71.3	81.4
Qwen-2.5-VL 7B*	80.3	82.3	79.1	83.1	73.9	86.6	65.4	92.8	85.0	81.2
Qwen-2.5-VL 3B Ranker	<b>93.8</b>	<b>94.2</b>	<b>93.9</b>	<b>93.8</b>	<b>92.4</b>	<b>95.3</b>	<b>92.8</b>	<b>95.0</b>	<b>92.1</b>	<b>95.5</b>
ScreenSpotv2										
Qwen-2.5-VL 3B*	78.0	78.3	78.0	78.0	75.5	80.5	75.6	80.4	75.4	80.6
Qwen-2.5-VL 7B*	82.4	83.2	81.8	84.6	78.7	86.1	70.2	93.4	89.4	79.8
Qwen-2.5-VL 3B Ranker	<b>94.5</b>	<b>94.6</b>	<b>94.5</b>	<b>94.5</b>	<b>93.8</b>	<b>95.1</b>	<b>93.5</b>	<b>95.4</b>	<b>94.2</b>	<b>94.9</b>
Osworld-G										
Qwen-2.5-VL 3B*	65.7	74.5	64.5	69.3	48.2	83.1	40.0	89.1	60.9	77.8
Qwen-2.5-VL 7B*	66.1	76.3	64.7	73.5	47.6	84.6	36.2	93.2	69.5	77.5
Qwen-2.5-VL 3B Ranker	<b>71.8</b>	<b>73.1</b>	<b>77.1</b>	<b>72.7</b>	<b>65.8</b>	<b>77.9</b>	<b>87.0</b>	<b>67.3</b>	<b>52.9</b>	<b>92.4</b>

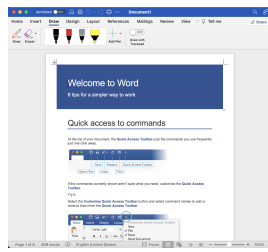
Table 7: Performance of our ranking model on ScreenSpot, Screenspotv2 and Osworld-G benchmarks. Positive and negative classes are denoted as (Y), (N) respectively. \* denotes zero-shot evaluation.

Model	Text Matching	Element Recognition	Layout Understanding	Finegrained Manipulation
Qwen-2.5-VL 3B*	70.5	63.1	66.5	60.5
Qwen-2.5-VL 7B*	69.9	64.6	67.8	58.0
Qwen-2.5-VL 3B Ranker	85.2	63.9	66.8	79.0

Table 8: Ranking performance on Osworld-G. \* denotes zero-shot evaluation.

### B.3 DATA ABLATIONS

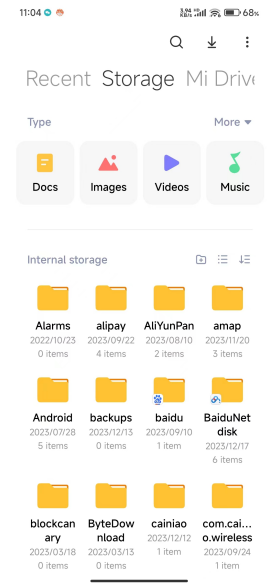
Table 10, and Table 11 illustrates detailed results regarding our data ablations. Similar to the results in the main paper, we report the grounding accuracy on ScreenSpot and Screenspotv2 with and without data filtering across all LoRA configurations. Across all runs, we observe that our model consistently achieves greater grounding performance with our small and curated training seed.



**Instruction:** Choose the red pen

**COT:** {"think": "The image shows a Microsoft Word interface with a toolbar at the top. The red pen is located among the drawing tools, identifiable by its red color.", "bbox": [205, 76, 234, 135]}

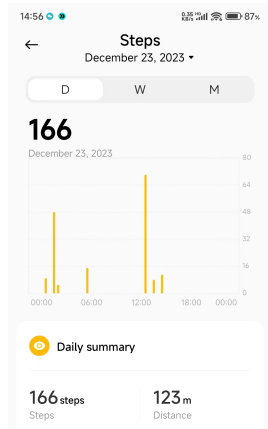
**GRPO:** <think> To choose the red pen, I need to click on the red pen icon. </think><answer>[209,83,231,131]</answer>



**Instruction:** Create a new folder

**COT** {"think": "The interface displays a file manager with various folders listed under Internal storage. To create a new folder, use the icon resembling a plus sign located next to the list of folders.", "bbox": [355, 395, 409, 540]}

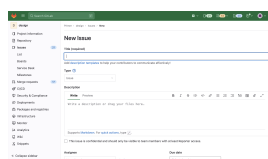
**GRPO:** <think> To create a new folder, I should click on the plus icon. </think><answer>[443,586,460,602]</answer>



**Instruction:** Show week steps recordings

**COT:** {"think": "The interface displays step recordings for December 23, 2023. The option labeled W under the date selection area is used to view week steps recordings.", "bbox": [594, 634, 648, 797]}

**GRPO:** <think>I should click on the W button to view the week steps recordings.</think> <answer>[294,187,320,209]</answer>



**Instruction:** Write a description

**COT:** {"think": "The interface displays a new issue form with fields for title, type, and description. The section labeled Write a description or drag your files here... contains the text input area for writing a description.", "bbox": [762, 819, 833, 864]}

**GRPO:**<think>To write a description, I need to click on the description area. </think><answer>[274,409,623,425]</answer>

Table 9: Samples of generated output from models trained via COT and GRPO.

Params (r/a)	Filtering	Windows		MACOS		IOS		Android		Gitlab		Shop		Forum		Tool		Micro Avg
		Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
32/16	✗	64.1	86.7	56.6	86.5	62.6	93.2	68.9	91.2	54.1	86.5	70.0	84.8	72.1	71.7	60.6	79.5	76.6
32/16	✓	70.3	91.8	61.8	94.8	70.1	98.7	68.0	92.8	62.2	92.3	76.7	88.1	67.4	82.6	65.2	79.5	81.0
64/32	✗	57.8	90.8	55.3	87.5	62.6	93.9	72.1	88.8	51.4	86.5	68.3	84.8	72.1	71.7	63.6	74.0	76.4
64/32	✓	73.4	92.9	63.2	91.7	72.0	98.0	69.7	91.2	59.5	92.3	70.0	86.4	65.1	84.8	63.6	75.3	80.4
128/64	✗	53.1	87.8	52.6	92.7	71.0	92.6	68.9	88.0	56.8	90.4	66.7	86.4	74.4	76.1	71.2	74.0	77.3
128/64	✓	68.8	91.8	63.2	92.7	70.1	98.7	68.9	95.2	56.8	92.3	76.7	88.1	69.8	80.4	60.6	75.3	80.5
256/128	✗	39.1	85.7	52.6	87.5	71.0	94.6	68.0	94.4	54.1	90.4	76.7	84.8	76.7	80.4	71.2	72.6	77.3
256/128	✓	75.0	94.9	60.5	94.8	69.2	98.7	65.6	93.6	54.1	90.4	76.7	86.4	62.8	84.8	63.6	72.6	80.2
32/64	✗	57.8	87.8	52.6	90.6	69.2	93.2	70.5	92.0	56.8	88.5	78.3	88.1	81.4	78.3	66.7	76.7	78.6
32/64	✓	71.9	93.9	64.5	92.7	73.8	98.0	66.4	92.8	67.6	94.2	78.3	89.8	69.8	82.6	59.1	78.1	81.4
64/128	✗	45.3	86.7	50.0	89.6	73.8	91.2	68.9	92.0	62.2	84.6	80.0	86.4	79.1	76.1	69.7	78.1	77.8
64/128	✓	70.3	90.8	55.3	92.7	68.2	96.0	67.2	91.2	54.1	88.5	85.0	91.5	67.4	82.6	56.1	72.6	78.9
128/256	✗	50.0	81.6	50.0	87.5	64.5	92.6	70.5	88.8	54.1	90.4	75.0	88.1	83.7	84.8	72.7	78.1	77.1
128/256	✓	62.5	89.8	51.3	94.8	63.6	93.9	68.9	92.0	46.0	94.2	78.3	89.8	60.5	82.6	63.6	74.0	77.8
256/512	✗	53.1	76.5	34.2	81.3	53.3	91.9	59.8	81.6	46.0	84.6	66.7	83.1	67.4	78.3	54.6	69.9	69.4
256/512	✓	40.6	85.7	36.8	88.5	53.3	91.9	59.8	86.4	40.5	80.8	65.0	84.8	48.8	82.6	53.0	68.5	69.7

Table 10: Grounding accuracy on ScreenSpot with and without data filtering. LoRA adapters are introduced within all linear layers of the model. Across all configurations, the data filtering approach yields greater performance (VL).

Params (r/a)	Filtering	Windows		MACOS		IOS		Android		Gitlab		Shop		Forum		Tool		Micro Avg
		Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	Text	Icon	
32/16	✗	66.7	87.5	68.3	91.7	70.0	93.5	73.6	92.5	58.6	90.9	63.5	88.5	71.8	77.5	59.3	80.3	79.6
32/16	✓	74.6	91.7	68.3	98.8	78.0	97.8	71.4	94.2	58.6	93.2	67.0	86.2	69.2	80.0	68.5	84.9	82.6
64/32	✗	58.7	92.7	66.7	94.1	69.0	92.8	74.7	92.5	58.6	84.1	62.6	87.7	69.2	72.5	66.7	84.9	79.5
64/32	✓	76.2	94.8	69.8	96.4	73.0	97.1	79.1	92.5	62.1	93.2	68.7	86.9	66.7	80.0	64.8	81.8	82.7
128/64	✗	58.7	89.6	57.1	97.6	76.0	94.2	75.8	92.5	69.0	88.6	60.0	88.5	74.4	77.5	66.7	78.8	80.0
128/64	✓	69.8	93.8	69.8	96.4	76.0	98.6	78.0	95.0	55.2	90.9	71.3	88.5	71.8	77.5	61.1	80.3	82.9
256/128	✗	41.3	84.4	57.1	92.9	76.0	94.9	78.0	95.8	58.6	88.6	71.3	88.5	74.4	77.5	64.8	77.3	79.6
256/128	✓	76.2	95.8	63.5	98.8	74.0	97.1	74.7	94.2	62.1	90.9	73.9	88.5	69.2	80.0	64.8	77.3	82.9
32/64	✗	58.7	89.6	58.7	95.2	71.0	93.5	78.0	93.3	69.0	88.6	68.7	87.7	79.5	77.5	59.3	83.3	80.5
32/64	✓	73.0	94.8	74.6	97.6	78.0	98.6	69.2	95.0	65.5	93.2	73.0	88.5	71.8	77.5	57.4	78.8	83.2
64/128	✗	47.6	89.6	54.0	95.2	77.0	92.0	78.0	95.0	62.1	86.4	71.3	91.5	76.9	75.0	70.4	78.8	80.7
64/128	✓	71.4	93.8	58.7	96.4	73.0	96.4	78.0	92.5	55.2	88.6	71.3	90.8	69.2	80.0	61.1	72.7	81.5
128/256	✗	49.2	82.3	55.6	92.9	67.0	92.0	74.7	91.7	62.1	86.4	67.0	86.9	79.5	85.0	74.1	78.8	78.5
128/256	✓	66.7	91.7	55.6	97.6	70.0	94.2	72.5	95.8	48.3	93.2	64.4	90.0	64.1	80.0	57.4	83.3	80.0
256/512	✗	54.0	78.1	36.5	86.9	58.0	92.0	63.7	84.2	51.7	84.1	56.5	80.8	66.7	80.0	46.3	77.3	71.2
256/512	✓	41.3	86.5	41.3	91.7	60.0	92.0	64.8	90.0	41.4	81.8	59.1	87.7	51.3	75.0	50.0	75.8	72.6

Table 11: Grounding accuracy on ScreenSpotv2 with and without data filtering. LoRA adapters are introduced within all linear layers of the model. Across all configurations, the data filtering approach yields greater performance (VL).