

000 001 002 003 004 005 XBOUND: EXPLORING CAPABILITY BOUNDARIES OF 006 DEVICE-CONTROL AGENTS AT THE STATE LEVEL 007 008 009

010 **Anonymous authors**
011 Paper under double-blind review
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029

030 ABSTRACT 031

032 Recent advancements in vision-language models have increased interest in Device-
033 Control Agents (DC agents) for managing graphical user interfaces (GUIs). With
034 the growing complexity and integration of such agents into various applications,
035 effective evaluation methods have become crucial. The current evaluation method
036 for DC agents primarily focuses on the instruction level, providing the current
037 state (e.g., screenshots) and past execution history to determine actions for target
038 instructions, helping identify potential execution failures. However, in GUI envi-
039 ronments, a single state may contain multiple interactive widgets, each linked to
040 different instructions, presenting an opportunity for diverse actions based on vari-
041 ous instruction targets. Evaluating the agent’s performance solely at the instruction
042 level may overlook the broader context of these interactions. To capture a more
043 comprehensive view of agent performance, we propose a new evaluation method,
044 XBOUND, to evaluate the accuracy of instruction completion on a per-state basis.
045 XBOUND provides a state-level evaluation framework, serving as a tool to assess
046 agents’ capabilities within environmental states. Our evaluation yields several key
047 insights: UI-TARS stands out as the strongest 7B model, current agents display a
048 bimodal performance pattern in instruction unification, and sub-7B models remain
049 limited in state mastery. We further identify GPT-based planning as a critical
050 bottleneck, and show that grounding data mainly benefits action matching, while
051 trajectory data is more effective for instruction unification.
052
053

054 1 INTRODUCTION 055

056 The recent advancement in vision-language models (VLMs) has spurred increased interest in Device-
057 Control Agents (DC agents), such as utilizing in-the-wild device control to manage graphical user
058 interfaces (GUIs) (Achiam et al., 2023; Anil et al., 2023; Zhang & Zhang, 2023; Hong et al., 2024;
059 Yang et al., 2023). There has been an increasing number of DC agents, making evaluating their
060 capability important.
061

062 With the growing complexity and integration of such agents into various applications, effective
063 evaluation methods have become crucial. The current evaluation method for DC agents primarily
064 focuses on the instruction level, providing the current state (e.g., screenshots) and past execution
065 history to determine actions for target instructions (Chen et al., 2024; Deng et al., 2024; Xie et al.,
066 2024; Deng et al., 2023). This method intuitively reveals which instructions DC agents might
067 fail to execute successfully (Rawles et al., 2023; Li et al., 2020; Burns et al., 2022). However, in
068 GUI environments, a single state may contain multiple interactive widgets, each linked to different
069 instructions, presenting an opportunity for diverse actions based on various instruction targets.
070 Evaluating the agent’s performance solely at the instruction level may overlook the broader context
071 of these interactions.
072

073 To capture a more comprehensive view of agent performance, we propose a new evaluation method
074 based on state-specific instruction accuracy. For each state, we assess the accuracy rate of completing
075 the instructions associated with it, and use the mean accuracy as an indicator of DC agent performance
076 for each state. In a given state, strong agent performance indicates effective learning, whereas poor
077 performance reveals areas where the agent lacks capability and requires improvement. By mapping
078 these strengths and weaknesses across different states, we can precisely delineate the capability
079 boundaries of DC agents. Specifically, our evaluation focuses on the following two research questions:
080
081

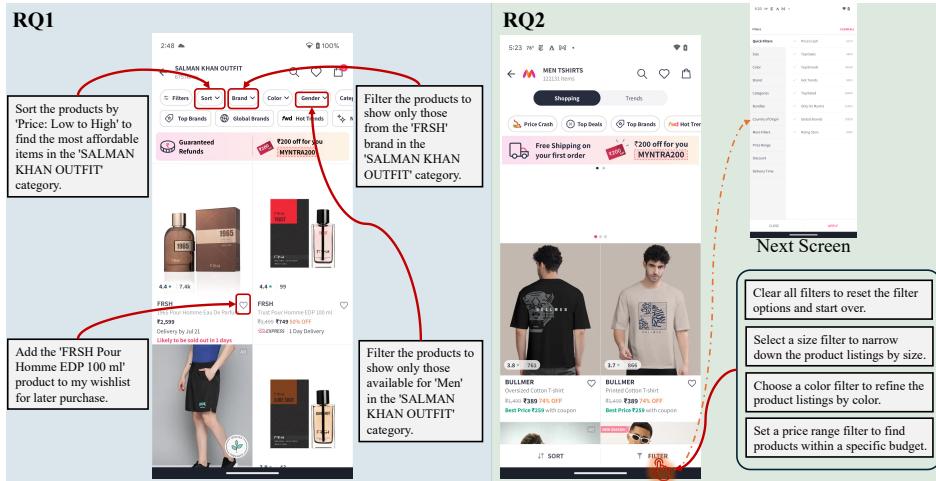


Figure 1: The two examples correspond to the two RQs. The left figure represents RQ1, and the right figure represents RQ2. In RQ1, we explore the interactive widgets in the current state along with their corresponding instructions and actions. In RQ2, we explore the set of instructions that may require the same action.

RQ1: Can the agent correctly discriminate and execute multiple distinct instructions mapped to different UI widgets under the same state?

RQ2: Can the agent unify semantically diverse instructions into the same action when they should converge on a single widget?

We use the two examples in Figure 1 to illustrate these two questions. To address these questions, we introduce an innovative evaluation method, EXploring the Capabilities **BOUNDaries** of Device-Control Agent Capabilities (XBOUND). Additionally, we propose two scenarios, Multi-Widget Action Matching and Uni-Widget Instruction Unification, to facilitate state-level analysis. Compared to previous evaluation methods, XBOUND employs a novel Exploration Metric that quantifies the extent to which DC agents master various states. We compute the average action accuracy of instructions associated with each state in these two scenarios. The comparison between XBOUND and existing evaluation methods is shown in Figure 4.

In this work, we evaluate 11 DC agents using the XBOUND method in the Mobile Use domain, aiming to assess their capabilities and limitations across two scenarios systematically. Additionally, we experimentally validate the role of grounding and trajectory data in enhancing agent capabilities across two scenarios. Finally, we summarize four challenging states. This evaluation yields noteworthy insights summarized below:

- Among models below the 7B parameter scale, UI-TARS stands out as the most competitive open-source model, achieving superior performance in both Multi-Widget Action Matching and Uni-Widget Instruction Unification.
- In Uni-Widget Instruction Unification, most current models exhibit a bimodal performance distribution, where DC agents either demonstrate exceptional proficiency or perform poorly.
- Models with fewer than 7B parameters, such as ShowUI and OS-Atlas-4B, demonstrate moderate performance and limited state mastery, indicating that terminal deployment of DC agents remains challenging.

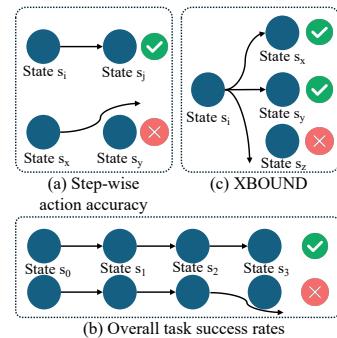


Figure 2: Comparison between XBOUND and existing evaluation methods.

- 108 • In Uni-Widget Instruction Unification, UGround performs particularly poorly. Through
109 observations across the three tasks, we find that the overall performance is mainly dragged
110 down by poor planning results from GPT in certain tasks.
- 111 • Comparative analysis indicates that grounding data primarily enhances Multi-Widget Action
112 Matching, whereas trajectory data is more effective for improving Uni-Widget Instruction.
- 113 • A comprehensive evaluation of DC intelligence requires not only measuring task completion
114 rates but also analyzing fine-grained challenging states such as widget disambiguation,
115 action topology reasoning, and dynamic state understanding.

117 2 RELATED WORK

118 2.1 LLM AS DEVICE-CONTROL AGENTS

119 Recently, there has been considerable exploration in the field of Device-Control Agents, ranging from
120 box prediction based on HTML and OCR parsing to coordinate prediction based on images (Li & Li,
121 2022; Li et al., 2024b; Wang et al., 2024a; Zhang et al., 2024b). For example, Yan et al. (2023) utilized
122 the MM-Navigator method to enhance the localization capabilities of GPT-4V. Zheng et al. (2024)
123 proposed a novel prompt method, SeeAct, which combines the reasoning abilities of LLMs with more
124 accurate HTML and OCR parsing to improve GPT-4V’s action prediction. Ma et al. (2024) trained a
125 segmented reasoning CoCo-Agent to boost action prediction accuracy. Wu et al. (2024) employed
126 significant engineering effort to collect multi-platform data and train a more powerful Device-Control
127 Grounding Agent OS-Atlas. Qin et al. (2025) trained UI-TARS on large-scale GUI screenshot data,
128 enabling context-aware understanding of UI widgets and precise captioning of interfaces. Gou et al.
129 (2024) introduces a human-like embodiment for DC agents that perceive the environment entirely
130 visually and directly perform pixel-level operations on the GUI.

131 2.2 EVALUATION FOR DEVICE-CONTROL AGENTS

132 To advance the development of DC Agents, researchers have constructed numerous datasets to
133 evaluate DC agents (Zhou et al., 2023; Xie et al., 2024; Rawles et al., 2024; Lu et al., 2024). Bai et al.
134 (2021); Deka et al. (2017); Cheng et al. (2024) created datasets focused on understanding UI Icons,
135 where models are required to identify the location of relevant UI Icons based on queries. As the
136 development of DC agents progresses, the demands for GUI datasets have shifted, necessitating agents
137 to perform a series of actions in response to user instructions. For example, Rawles et al. (2023); Sun
138 et al. (2022) constructed datasets containing episodes in the form of a sequence of screen-action pairs.
139 Zhang et al. (2024a) supplemented the AITW dataset by adding thought processes. Li et al. (2024a)
140 constructed a fine-grained AndroidControl dataset by including low-level instructions during episodes.
141 Wang et al. (2025) introduced a hierarchical benchmark for evaluating GUI automation agents across
142 six platforms. Zhao et al. (2025) assessed an agent’s capability to autonomously generate shortcuts.
143 Zheng et al. (2025) introduced a novel benchmark engineered on the principle of Causal Pathways.

144 However, current evaluation methods primarily focus on the instruction level and may neglect the
145 broader context of interactions within a given state. To address this limitation, we propose the
146 XBOUND evaluation method, a state-level framework designed to assess agents’ capabilities within
147 environmental states.

148 3 NEW METRIC: EXPLORATION METRIC

149 In this section, we first define the capability of DC agents to clarify how their performance manifests
150 across different scenarios. Alongside this definition, we present the Exploration Metric, outlining its
151 calculation in various scenarios and discussing its significance for evaluating agent capabilities.

152 3.1 CAPABILITY OF DC AGENTS

153 From the perspective of state evaluation, the capabilities of DC agents are intrinsically linked to
154 action matching and instruction unification. Action matching requires agents to correctly map
155 diverse instructions in the same state to their corresponding UI widgets and execute the intended

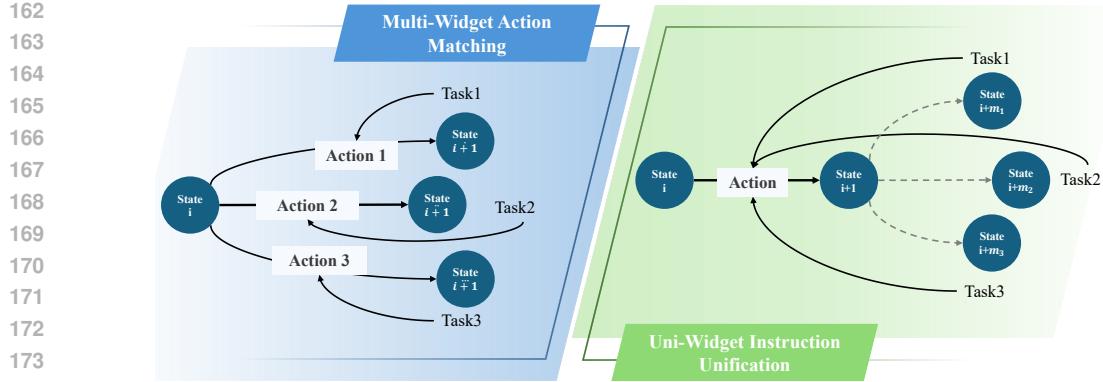


Figure 3: Abstract examples of the two scenarios. Multi-Widget Action Matching: In *State i*, executing the corresponding *Action* under different *Tasks* requirements can transition to different *States i+1*. Uni-Widget Instruction Unification: Executing *Action* in *State i* can transition to *State i+1*, and from *State i+1*, different *Tasks* can lead to different *States i+m*.

actions, reflecting their ability to discriminate between multiple instruction–action pairs. In contrast, instruction unification requires agents to interpret semantically diverse instructions that should converge to the same UI action, reflecting their ability to comprehend instruction-level semantics and generalize across varied command expressions. Consequently, we categorize the capability of DC agents into two scenarios: Multi-Widget Action Matching and Uni-Widget Instruction Unification.

Multi-Widget Action Matching: In this scenario, a set of instructions is collected under the same state, where each instruction corresponds to a distinct UI widget and requires executing its associated action. This setting evaluates whether the agent can accurately perceive the relevant UI widgets and correctly match each instruction to the intended action, thus reflecting its capability of instruction–action discrimination.

Uni-Widget Instruction Unification: In this scenario, a set of diverse instructions is provided under the same state, but all instructions should converge to the same target action on a single widget. This setting evaluates whether the agent can understand the semantic equivalence of varied instruction expressions and unify them into a consistent action, thus reflecting its capability of instruction-level semantic comprehension.

Figure 3 presents examples of these two scenarios. By dividing these two scenarios, we calculate the agents’ performance in different states to identify their capability boundaries.

3.2 EXPLORATION METRIC

To evaluate the capabilities of DC agents within these two scenarios, we introduce the XBOUND evaluation method, which measures agent performance along two dimensions: MWAM and UWIU. The **MWAM** dimension aligns with **Multi-Widget Action Matching**, assessing whether agents can correctly map instructions to their corresponding UI widgets and generalize these behaviors across diverse visual contexts. The **UWIU** dimension corresponds to **Uni-Widget Instruction Unification**, evaluating whether agents can consistently execute the same action when faced with semantically varied instructions, thereby reflecting their robustness in handling diverse task formulations.

XBOUND introduces a novel Exploration Metric for quantifying agent capability within environmental states. We gather the set of executable instructions within the same state and calculate the average accuracy of this instruction set as a measure of the agent’s capability in the given state. By evaluating performance across all states, we derive corresponding values for each state’s agent performance, ultimately illustrating the agent’s capability boundaries. The formulas for the Exploration Metric(EM) are as follows:

$$EM_{state} = \frac{1}{m} \sum_{i=1}^m I(A_i), \quad (1)$$

$$EM_{all} = \frac{1}{s} \sum_{j=1}^s EM_{state_j}, \quad (2)$$

where $I(\cdot)$ is the indicator function, which equals 1 if the action A_i is correct and 0 otherwise. The variable m represents the number of instructions associated with each screenshot, while s denotes the number of screenshots.

Since current benchmarks cannot directly utilize the XBOUND evaluation technique, it is necessary to expand the existing dataset to meet our requirements. Owing to community development in recent years, current benchmarks provide detailed accessibility trees information, which we leverage to improve the Android Control dataset (Li et al., 2024a). The detailed pipeline is provided in the Appendix A.3. Finally, we retain 43,759 instructions in the MWAM dimension, while the UWIU dimension contains 13,460 instructions.

4 EXPERIMENT

The experimental setup is described in Sec. 4.1; overall evaluation results are presented in Sec. 4.2; task-specific performance is detailed in Sec. 4.3; error analysis is provided in Sec. 4.4; the comparison between grounding training and trajectory training is discussed in Sec. 4.5; and the summary of challenging states is included in Sec. 4.6.

4.1 EXPERIMENTAL SETUP

DC agent. We select eleven open-source DC agents with fewer than 7B parameters as evaluation models, including ShowUI (Lin et al., 2024), SeeClick (Cheng et al., 2024), Qwen2-VL-Instruct (Wang et al., 2024b), Uground (Gou et al., 2024), OS-Atlas (Wu et al., 2024), Aguvis (Xu et al., 2024), GUI-Owl (Ye et al., 2025), and UI-TARS (Qin et al., 2025). We adhere to the prompts they utilize while deliberately excluding execution history from the inputs. Our experiments are conducted on an A100 GPU with 80GB of memory.

Evaluation Metrics. In line with the criteria set forth by Zhang & Zhang (2023); Wu et al. (2024), an action is considered correct if its type matches the ground-truth type. Specifically, for CLICK and LONG PRESS actions, correctness in the UWIU dimension is determined if they occur within a 14% screen distance from the reference gestures. In the MWAM dimension, correctness is assessed based on whether the actions fall within the bounding box of the ground truth UI icon. For SCROLL actions, correctness is evaluated by checking if the direction (up, down, left, or right) matches the reference gesture. For TYPE actions, correctness is assessed using the F1 score; the action is deemed correct if the score is below a threshold of 0.5, as set in our experiments.

4.2 COMPREHENSIVE EVALUATION

We calculate the results of the Exploration Metric corresponding to each state. To enhance state analysis, we partition the Exploration Metric into four distinct intervals:

- **Learning Stage** ($EM_{state} < 30\%$)
The current state suggests that DC agents are still in the learning and adaptation phase, indicating unfamiliarity with the environment.
- **Improvement Stage** ($30\% \leq EM_{state} < 60\%$)
The current state indicates that DC agents have started to grasp certain operations and are making progress.
- **Proficient Stage** ($60\% \leq EM_{state} < 90\%$)
The current state signifies that DC agents possess a relatively proficient understanding and can perform most actions.
- **Expert Stage** ($90\% \leq EM_{state} \leq 100\%$)
The current state implies that DC agents have achieved a comprehensive and expert level of understanding.

270
 271 Table 1: The assessment of DC agents’ capabilities spans two dimensions: MWAM and UWIU.
 272 Specifically, the percentage of states is reported for the four stages, i.e., Learning Stage (LS),
 273 Improvement Stage (IS), Proficient Stage (PS), and Expert Stage (ES). The Exploration Metric
 274 (EM) quantifies the overall mastery of states by DC agents, reflecting their accuracy in completing
 275 instructions within each state. In contrast, the Success Rate (SR) measures the step-wise success
 276 rate of DC agents across the dataset, indicating their proficiency in executing individual steps. For
 277 each DC agent, the best-performing Stage results are highlighted in bold, and the second-best are
 278 underlined. For the EM and SR, we bold the best-performing results of the 11 DC agents.
 279

Model	MWAM						UWIU					
	LS	IS	PS	ES	EM	SR	LS	IS	PS	ES	EM	SR
ShowUI-2B	75.09	<u>19.17</u>	4.17	1.57	18.51	19.70	68.44	9.76	5.77	<u>16.03</u>	25.27	25.54
OS-Atlas-4B-Pro	37.29	<u>34.25</u>	22.51	5.95	41.92	42.82	57.16	<u>19.60</u>	7.25	15.99	31.80	30.24
OS-Atlas-7B-Pro	20.41	<u>27.74</u>	35.48	16.37	57.59	59.50	<u>36.83</u>	12.95	12.90	37.32	53.44	53.22
SeeClick	53.16	<u>24.68</u>	15.97	6.19	32.58	33.72	77.05	<u>10.60</u>	3.42	8.93	17.15	14.93
Qwen2-VL-7B-Ins	26.05	35.00	<u>30.98</u>	7.97	49.30	50.88	58.84	14.85	10.21	<u>16.11</u>	31.52	32.19
Aguvis-7B	19.23	<u>29.16</u>	37.25	14.35	57.39	59.23	37.32	15.91	12.95	<u>33.91</u>	51.29	51.47
UGround-7B	14.20	<u>25.53</u>	38.11	22.16	63.00	63.66	75.95	3.39	1.90	<u>18.81</u>	21.97	20.35
UI-TARS-7B-SFT	10.54	<u>24.40</u>	42.52	22.55	66.96	68.20	39.41	13.98	11.97	<u>34.64</u>	50.53	52.40
UI-TARS-7B-DPO	11.45	<u>24.59</u>	42.20	21.76	66.08	67.57	37.55	14.03	13.13	<u>35.29</u>	52.02	53.41
UI-TARS-1.5-7B	13.54	<u>25.33</u>	40.24	20.89	64.25	65.82	<u>16.66</u>	8.25	10.03	65.05	76.44	79.69
GUI-Owl-7B	14.13	<u>25.97</u>	41.24	18.65	62.97	64.87	<u>32.03</u>	14.33	14.59	39.05	56.96	58.41

293
 294 DC agents with stronger capabilities should have a higher proportion in the Proficient Stage and
 295 Expert Stage, and a lower proportion in the Learning Stage and Improvement Stage. We present the
 296 four stages and the Exploration Metric of various DC agents across different dimensions in Table 1.
 297

298 **Performance of models with fewer than 7 billion parameters reveals insufficient proficiency.**
 299 Analysis of ShowUI and OS-Atlas-4B-Pro indicates that agents with parameter scales smaller than 7
 300 billion still exhibit inadequate performance. Across both MWAM and UWIU dimensions, most state
 301 performances remain within the Learning Stage and Improvement Stage. Specifically, ShowUI-2B is
 302 in the Learning Stage as high as 75.09%, highlighting the challenges of achieving effective terminal
 303 deployment with smaller models.

304 **UI-TARS models demonstrate superior performance among the 7 billion parameter models.**
 305 Observations from the table show that the UI-TARS series is the top performer within this parameter
 306 range. Each of the three UI-TARS models achieves at least 64% overall Exploration Metric perfor-
 307 mance in the MWAM dimension, with most state performances in the Proficient Stage and Expert
 308 Stage. In the UWIU dimension, UI-TARS-1.5-7B records the best performance. Given the unknown
 309 specifics of their training data, we speculate that version 1.5 includes more trajectory tasks, enhancing
 310 the model’s comprehension of current action execution concerning future tasks.

311 **UGround exhibits anomalous performance in the UWIU dimension.** Analysis indicates that
 312 UGround’s errors largely stem from incorrect planning by GPT, primarily due to erroneous environ-
 313 mental perception. For example, the model may plan to return to the desktop to open an email app
 314 when attempting to forward content within an app. This highlights the necessity of training the model
 315 with relevant planning data.

316 **A bimodal distribution phenomenon observed in the UWIU dimension.** In the UWIU dimension,
 317 a bimodal distribution emerges where agents exhibit complete absence or presence of action learning.
 318 This suggests that current DC agents are yet to achieve human-like intelligence and still have
 319 significant developmental strides to make.

320 4.3 CAPABILITY EVALUATION BASED ON TASK

321 We select three tasks from the test data and sample 2,000 instructions for each task. Correct
 322 instructions are manually filtered, and highly repetitive instructions are removed. The final statistical

324
 325 Table 2: The proportion of states in the four stages for 6 agents across three tasks. The best-performing
 326 Stage results are highlighted in bold, and the second-best are underlined.

327 328 329 330 331 332 333 334 335	336 337 338 339 340 341	342 343 344 345 346 347	Task	Model	MWAM				UWIU			
					LS	IS	PS	ES	LS	IS	PS	ES
Maps	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	6.90	24.14	37.93	<u>31.03</u>	40.92	13.64	18.18	<u>27.27</u>		
			0	<u>27.59</u>	51.72	20.69	<u>43.75</u>	0	0	56.25		
			0	17.24	<u>37.93</u>	44.83	<u>40.91</u>	0	0	59.09		
			3.45	10.34	<u>37.93</u>	48.28	<u>45.45</u>	0	0	54.55		
			3.45	10.34	<u>34.48</u>	51.72	9.09	9.09	9.09	72.73		
News	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	3.45	6.90	<u>27.59</u>	62.07	<u>27.27</u>	13.64	22.73	36.36		
			4	24	<u>32</u>	40	8	<u>28</u>	4	60		
			4	36	28	<u>32</u>	73.68	0	0	<u>26.32</u>		
			0	16	<u>32</u>	52	<u>56</u>	0	0	<u>44</u>		
			4	12	<u>32</u>	52	<u>56</u>	0	0	<u>44</u>		
Shopping	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	Aguvis-7B UGround-7B UI-TARS-7B-SFT UI-TARS-7B-DPO UI-TARS-1.5-7B GUI-Owl-7B	4.65	11.63	53.49	<u>30.23</u>	21.88	<u>28.12</u>	3.12	46.88		
			11.63	23.26	34.88	<u>30.23</u>	84.62	0	0	<u>15.38</u>		
			2.33	9.3	<u>37.21</u>	51.16	<u>43.75</u>	0	0	56.25		
			2.33	6.98	<u>39.53</u>	51.16	<u>43.75</u>	0	0	56.25		
			2.33	6.98	<u>34.88</u>	55.81	<u>9.38</u>	<u>9.38</u>	0	81.25		

350 results are presented in the Appendix A.3. We evaluate the performance of 11 agents on these three
 351 tasks. The proportion of states in each of the four stages for six agents is shown in Table 2; results
 352 for the remaining five agents are presented in Appendix A.4. The EM capability chart is provided in
 353 Figure 4. Analysis of Table 2 reveals the following:

354 The UI-TARS and GUI-Owl series currently demonstrate
 355 the best performance in the MWAM dimension (Multi-
 356 Widget Action Matching) for the Maps, News, and Shop-
 357 ping tasks.

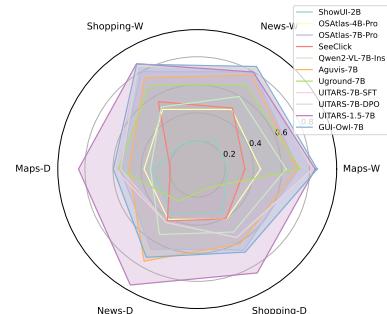
358 UGround performs well on the Maps task but poorly on
 359 News and Shopping. This indicates that employing GPT
 360 as a planning model does not consistently result in poor
 361 performance across all tasks; rather, certain tasks adversely
 362 affect its overall performance.

363 UGround, UI-TARS-7B-SFT, and UI-TARS-7B-DPO ex-
 364 hibit a pronounced bimodal distribution in the UWIU di-
 365 mension for Maps, News, and Shopping tasks, suggesting
 366 that these models fail to train on certain actions and possess
 367 learning blind spots.

368 Figure 4 shows that UI-TARS-1.5-7B is the best-performing
 369 agent overall across the three tasks, while Aguvis, UI-TARS-7B-SFT, UI-TARS-7B-DPO, and
 370 GUI-Owl display comparable performance.

373 4.4 ERROR ANALYSIS

375 By calculating the Exploration Metric, we can quickly identify states with low accuracy. In further
 376 analyzing these poor performance states, we look closely at the actions undertaken by the agent
 377 and compute their similarity. Typically, the more similar the actions, the closer the similarity value
 approaches 1; conversely, the more disparate the actions, the closer the similarity value approaches 0.



372
 373 Figure 4: The EM of 11 agents across
 374 three tasks.

378
 379 Table 3: The comparison between grounding training and trajectory training. The best-performing
 380 Stage results are highlighted in bold, and the second-best are underlined. For the EM, we bold the
 381 best-performing results.

Model	MWAM					UWIU				
	LS	IS	PS	ES	EM	LS	IS	PS	ES	EM
Qwen2-VL-7B-Ins	26.05	35.00	<u>30.98</u>	7.97	49.30	58.84	14.85	10.21	<u>16.11</u>	31.52
OS-Atlas-7B-Base	16.07	<u>27.40</u>	39.26	17.27	60.61	52.14	17.93	9.22	<u>20.71</u>	36.91
OS-Atlas-7B-Pro	20.41	<u>27.74</u>	35.48	16.37	57.59	<u>36.83</u>	12.95	12.90	37.32	53.44

388
 389
 390 If the similarity value of erroneous actions nears 1, this indicates systematic errors within these
 391 states. We explain the cause of mistakes through the two scenarios (Section 3.1). (1) Multi-Widget
 392 Action Matching: In this scenario, the agent may execute incorrect actions (such as press_back
 393 and press_home) due to insufficient understanding of the state, implying that the current state is
 394 unfamiliar to the agent. (2) Uni-Widget Instruction Unification: This may be due to the agent having
 395 learned only the action associated with the current state, without recognizing distinctions between
 396 task instructions, which leads it to execute the same action regardless of the task. This indicates that
 397 the model possesses limited capability to handle diverse instructions in that state.

398 If the similarity value of erroneous actions is closer to 0, it signifies inadequate learning by the agent
 399 in these states. We explain the cause of mistakes through the two scenarios. (1) Multi-Widget Action
 400 Matching: In this scenario, the current state usually contains UI widgets unlearned by the agent,
 401 leading to incorrect actions, evidencing insufficient action matching capability for the current state.
 402 (2) Uni-Widget Instruction Unification: The agent fails to grasp the relationship between the current
 403 action and future state transitions, resulting in errors in action selection. This indicates inadequate
 404 learning of actions by the agent. More examples are presented in the Appendix A.5.

405 4.5 GROUNDING TRAINING AND TRAJECTORY TRAINING COMPARISON

406
 407 In this section, we discuss the impact of grounding training and trajectory training on Mobile Use
 408 Agents. Utilizing the XBOUND evaluation method, we assess three models: Qwen2-7B-VL-Instruct,
 409 OS-Atlas-7B-Base, and OS-Atlas-7B-Pro. OS-Atlas-7B-Base is trained with grounding data based on
 410 Qwen2-7B-VL-Instruct, whereas OS-Atlas-7B-Pro incorporates trajectory data based on OS-Atlas-
 411 7B-Base. The results are presented in Table 3.

412 Comparing the Base model with the Qwen model reveals that grounding data enhanced the Base
 413 model’s performance in the MWAM dimension, improving its understanding of Multi-Widget
 414 Action Matching. However, significant improvement in the UWIU dimension is not observed until
 415 trajectory data is employed for training, which subsequently enhances performance within Uni-
 416 Widget Instruction Unification. This also demonstrates that grounding data is associated with the
 417 agents’ action-matching abilities, whereas trajectory data is linked to their decision-making abilities.

418 4.6 CHALLENGING STATES

419 During the XBOUND evaluation process, we identify several challenging states that require a certain
 420 level of intelligence from DC agents. We categorize these states into the following four demands:

421 (1) **Understanding of UI Icons:** Due to limitations in current grounding and trajectory data, many UI
 422 icons are absent from agents’ training, potentially causing agents to fail in learning tasks associated
 423 with certain UI icons. This scenario heavily assesses the agents’ action-matching abilities regarding
 424 UI icons. In Figure 5(a), the agent fails to recognize the image-search icon and repeatedly selects the
 425 text search box instead.

426 (2) **Distinction Between Similar UI Icons:** Tasks in domains such as shopping and media often
 427 present multiple visually similar icons on the same page. Agents must correctly align instructions
 428 with the intended target, testing their widget-level discrimination ability. In Figure 5(b), the agent
 429 misinterprets a “like” command and clicks on the wrong post.

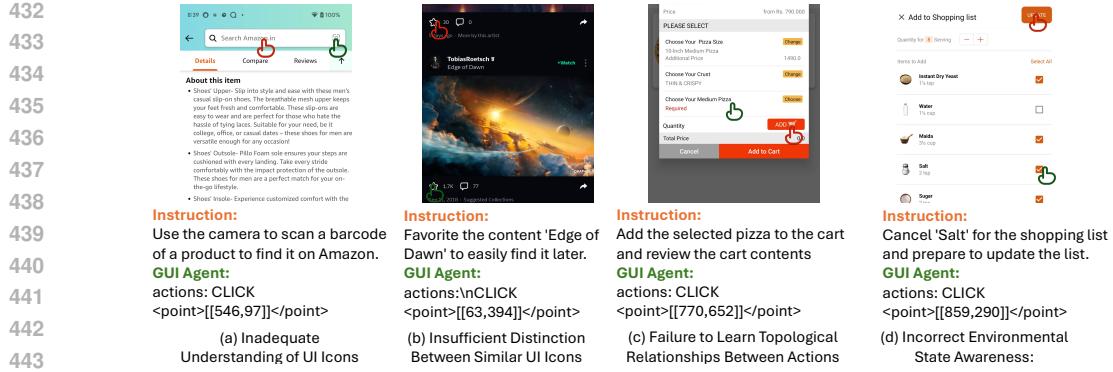


Figure 5: Four types of challenging states are illustrated through representative examples. A green pointer indicates the correct action, while a red pointer denotes the incorrect action.

(3) **Topological Relationships Between Actions:** In specific task scenarios on certain pages, actions may have topological dependencies where action B must precede action A. This scenario requires agents to comprehend the topological order of actions and possess a deeper understanding of their meanings. In Figure 5(c), the instruction is to complete the customization of a pizza directly. However, the “Choose Your Medium Pizza” option isn’t selected, and “Add” should only be executed after completing the “Choose Your Medium Pizza” step.

(4) **Environmental State Awareness:** For some ordering and shopping tasks, product requirements often need alteration. This scenario demands that agents understand or perceive the environmental state necessary for task execution correctly. In Figure 5(d), the goal is to remove “Salt” before the update, but the DC agent fails to notice that “Salt” is already added and proceeds with the update action instead.

5 LIMITATIONS AND FUTURE WORK

This work presents a novel evaluation method, which we extend with the Android Control dataset since it does not directly apply to existing data. While LLM assists in filtering, the resulting data may not fully ensure the accuracy of instructions and actions. Despite this, **the XBOUND method highlights promising directions for future research.**

More Complete and Refined Evaluation: The evaluation data in this work have the following characteristics: offline data and trajectory independence, which make it impossible to build a full trajectory tree. In future development of evaluation data for Mobile Use, we can organize app-level trajectory tree data and evaluate DC agents’ capabilities app-wise using the XBOUND method.

New Ideas for Enhancing DC Agents’ Capabilities: Currently, improving DC agents’ performance primarily focuses on the instruction level, relying on extensive trajectory data collection to enhance agents’ capabilities. In the future, if augmented instructions cease to be efficient in enhancing agents’ capabilities, we can evaluate agents’ performance across different states and focus on improving performance in underperforming states to boost their overall capabilities.

6 CONCLUSION

This study delineates the scenarios of DC agents’ capability within states and introduces a novel evaluation method, XBOUND. XBOUND provides a state-level evaluation framework, serving as a tool to assess agents’ capabilities within environmental states. From the perspective of state evaluation, we define two distinct scenarios, evaluate the performance of 11 open-source agents within the Mobile Use domain, offering new insights at the state level. Our evaluation reveals several key insights. UI-TARS emerges as the most effective model at the 7B scale. Current agents exhibit a bimodal performance pattern in instruction unification, while sub-7B models demonstrate limited state mastery. In addition, we identify GPT-based planning as a critical bottleneck and show that grounding data primarily improves action matching, whereas trajectory data proves more effective for instruction unification.

486 7 ETHICS STATEMENT
487488 We confirm that this work adheres to the ethical guidelines set forth by the ICLR 2026 conference.
489490 8 REPRODUCIBILITY STATEMENT
491492 We hereby declare that our work is fully reproducible. Using our data augmentation pipeline, multiple
493 instructions can be generated. By applying these instructions (or those we provide) to evaluate the 11
494 DC agents, the same observations can be obtained in both the Multi-Widget Action Matching and
495 Uni-Widget Instruction Unification scenarios.
496

497 498 REFERENCES

499 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,
500 Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report.
501 *arXiv preprint arXiv:2303.08774*, 2023.502 Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos,
503 Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv*
504 *preprint arXiv:2305.10403*, 2023.505 Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen,
506 et al. Uibert: Learning generic multimodal representations for ui understanding. *arXiv preprint*
507 *arXiv:2107.13731*, 2021.508 Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A Plummer. A
509 dataset for interactive vision-language navigation with unknown command feasibility. In *European*
510 *Conference on Computer Vision*, pp. 312–328. Springer, 2022.511 Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui
512 Zhou, Weiwen Liu, Shuai Wang, et al. Spa-bench: A comprehensive benchmark for smartphone
513 agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.514 Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu.
515 Seeclink: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint*
516 *arXiv:2401.10935*, 2024.517 Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hirschman, Daniel Afergan, Yang Li, Jeffrey
518 Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design
519 applications. In *Proceedings of the 30th annual ACM symposium on user interface software and*
520 *technology*, pp. 845–854, 2017.521 Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin
522 Wang, Rui Yan, et al. Mobile-bench: An evaluation benchmark for llm-based mobile agents. *arXiv*
523 *preprint arXiv:2407.00993*, 2024.524 Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su.
525 Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing*
526 *Systems*, 36:28091–28114, 2023.527 Boyu Gou, Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and
528 Yu Su. Navigating the digital world as humans do: Universal visual grounding for gui agents.
529 *arXiv preprint arXiv:2410.05243*, 2024.530 Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan
531 Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents.
532 In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.
533 14281–14290, 2024.534 Gang Li and Yang Li. Spotlight: Mobile ui understanding using vision-language models with a focus.
535 *arXiv preprint arXiv:2209.14927*, 2022.

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

543

544 Yanda Li, Chi Zhang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei.
 545 Appagent v2: Advanced agent for flexible mobile interactions. *arXiv preprint arXiv:2408.11824*,
 546 2024b.

547

548 Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. Mapping natural language
 549 instructions to mobile ui action sequences. *arXiv preprint arXiv:2005.03776*, 2020.

550

551 Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Zechen Bai, Weixian Lei, Lijuan Wang,
 552 and Mike Zheng Shou. Showui: One vision-language-action model for generalist gui agent. In
 553 *NeurIPS 2024 Workshop on Open-World Agents*, volume 1, 2024.

554

555 Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang,
 556 Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui
 557 navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024.

558

559 Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. Coco-agent: A comprehensive cognitive mllm agent
 560 for smartphone gui automation. *arXiv preprint arXiv:2402.11941*, 2024.

561

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

565

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

569

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

573

574 Liangtai Sun, Xingyu Chen, Lu Chen, Tianle Dai, Zichen Zhu, and Kai Yu. Meta-gui: Towards
 575 multi-modal conversational agents on mobile gui. *arXiv preprint arXiv:2205.11029*, 2022.

576

577 Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang,
 578 and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via
 579 multi-agent collaboration. *arXiv preprint arXiv:2406.01014*, 2024a.

580

581 Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu,
 582 Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the
 583 world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

584

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

588

589 Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen
 590 Ding, Liheng Chen, Paul Pu Liang, et al. Os-atlas: A foundation action model for generalist gui
 591 agents. *arXiv preprint arXiv:2410.23218*, 2024.

592

593 Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua,
 594 Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents
 595 for open-ended tasks in real computer environments. *Advances in Neural Information Processing*
 596 *Systems*, 37:52040–52094, 2024.

597

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

594 An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu
 595 Zhong, Julian McAuley, Jianfeng Gao, et al. Gpt-4v in wonderland: Large multimodal models for
 596 zero-shot smartphone gui navigation. *arXiv preprint arXiv:2311.07562*, 2023.

597

598 Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and
 599 additional opinions. *arXiv preprint arXiv:2306.02224*, 2023.

600 Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu
 601 Gao, Junjie Cao, Zhengxi Lu, et al. Mobile-agent-v3: Foundamental agents for gui automation.
 602 *arXiv preprint arXiv:2508.15144*, 2025.

603

604 Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu
 605 Tang. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*,
 606 2024a.

607 Shaoqing Zhang, Zhuosheng Zhang, Kehai Chen, Xinbei Ma, Muyun Yang, Tiejun Zhao, and Min
 608 Zhang. Dynamic planning for llm-based graphical user interface automation. *arXiv preprint
 609 arXiv:2410.00467*, 2024b.

610 Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents.
 611 *arXiv preprint arXiv:2309.11436*, 2023.

612

613 Pengxiang Zhao, Guangyi Liu, Yaozhen Liang, Weiqing He, Zhengxi Lu, Yuehao Huang, Yaxuan
 614 Guo, Kexin Zhang, Hao Wang, Liang Liu, et al. Mas-bench: A unified benchmark for shortcut-
 615 augmented hybrid mobile gui agents. *arXiv preprint arXiv:2509.06477*, 2025.

616 Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web
 617 agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

618

619 Zihan Zheng, Tianle Cui, Chuwen Xie, Jiahui Zhang, Jiahui Pan, Lewei He, and Qianglong Chen.
 620 Naturegaia: Pushing the frontiers of gui agents with a challenging benchmark and high-quality
 621 trajectory dataset. *arXiv preprint arXiv:2508.01330*, 2025.

622 Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng,
 623 Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building
 624 autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648 A APPENDIX
649650 A.1 THE USE OF LARGE LANGUAGE MODELS
651652 In this work, LLMs serve two primary purposes: one is for polishing the paper, and the other is for
653 data augmentation.
654655 A.2 TRAJECTORY TREE DATASET
656657 Common GUI trajectory datasets record a series of action sequences that result in screen transi-
658 tions. However, trajectory trees focus on other possible actions and tasks that may branch out
659 from a given state. Formally, given the state S_t at the time step t and various user instructions
660 $I_t^1, I_t^2, \dots, I_t^M$, the DC agent G will take corresponding actions under different instructions, such
661 as $A_t^1 = G(S_t, I_t^1), A_t^2 = G(S_t, I_t^2), \dots, A_t^M = G(S_t, I_t^M)$, where M represents the number of
662 instructions. Each instruction completes a corresponding trajectory sequence $E = \{(S_t, A_t)_{t=1}^T, I\}$,
663 where T represents the total steps. These trajectory sequences are constructed into a trajectory tree
664 dataset $T = \{(S_t^m, (I^m, A^m), S_{t+1}^m)_{t=1}^M\}_{m=1}^M$ based on overlapping states.
665666 A.3 DATA EXPANSION METHOD
667668 Using the accessibility trees provided, we commence by annotating each screenshot with accessibility
669 trees from Android Control to identify clickable and visible UI icons, which are sequentially num-
670 bered. Red boxes highlight these icons on screenshots, and their numbers ensure clear identification.
671 Subsequently, GPT-4omini is employed to generate both high-level and low-level instructions. After
672 acquiring the instructions, we utilize Qwen2.5-VL-72B-Instruct to translate them into corresponding
673 actions (e.g., click, scroll, type, etc.).
674675 In the MWAM dimension, we focus on collecting single screenshots and request GPT-4omini to
676 produce task instructions based on the UI icons present on these screenshots. In the UWIU dimension,
677 we primarily gather screenshot-action-screenshot pairs (S_t, A_t, S_{t+1}) and ask GPT-4omini to expand
678 instructions for the UI icons on the subsequent screenshot S_{t+1} . We then combine the high-level
679 instructions $I_{t+1}^1, I_{t+1}^2, \dots, I_{t+1}^M$ collected from screenshot S_{t+1} with action A_t and screenshot S_t
680 into the structure $\{(S_t, (I^m, A_t), S_{t+1})\}_{m=1}^M$. When DC agents strive to execute the expanded
681 instructions from the subsequent screenshot, they must perform the collected actions on the previous
682 screenshot.
683684 To ensure data quality, GPT-4omini evaluates whether actions and low-level instructions satisfy the
685 high-level instructions, resulting in a dataset of successful interactions. Since high-level instructions
686 are generated per screenshot and do not constitute a complete trajectory, the test dataset is termed a
687 “pseudo” trajectory tree dataset. Figure 6 visually represents the dataset construction process for these
688 dimensions.
689690 Our main experiment dataset comprises 1,536 episodes with 43,759 instructions, where the MWAM
691 dimension includes 43,759 instructions, and the UWIU dimension contains 13,460 instructions. We
692 have tallied the number of instructions associated with each screenshot and the action distribution,
693 with detailed information presented in Figure 7 and Figure 8.
694695 Following Li et al. (2024a), we use Qwen2-vl-7B-Instruct to classify the test data based on the app
696 categories provided. Ultimately, we select the three most prevalent app categories within the test set
697 for further analysis. The statistical results of the collected instructions are presented in Table 4.
698699 Table 4: Data statistics of the three tasks in both dimensions.
700701

Task	Width	UWIU
Maps	135	73
News	115	58
Shopping	205	99

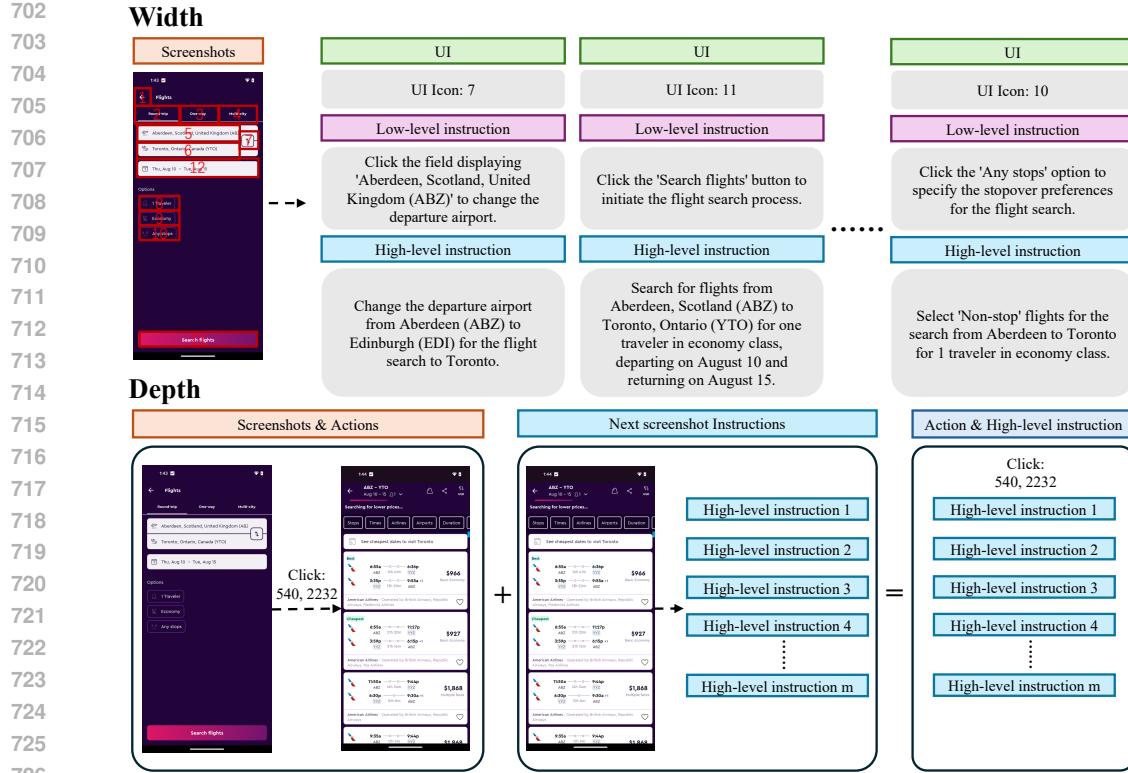


Figure 6: The data collection construction process involves MWAM and UWIU dimensions. **MWAM Dimension:** Screenshots are annotated, and GPT4o-mini is utilized to select UI elements for generating both low-level and high-level instructions. **UWIU Dimension:** High-level instructions corresponding to subsequent screenshots are identified based on transitions between screenshots, alongside the collection of actions and high-level instructions.

A.4 ANOTHER 5 AGENT CAPABILITY PERFORMANCE

The performance of the remaining five agents on three different tasks is reported in Table 5.

We observe that the 2B model still performs poorly, with ShowUI remaining the weakest across all three tasks. In contrast, OS-Atlas-4B-Pro has already outperformed SeeClick and is approaching the performance of Qwen2-VL-7B-Ins.

A.5 SIMILARITY CASE

Figure 9 illustrates an example from the Multi-Widget Action Matching scenario where the similarity of incorrect actions is 0, while Figure 10 presents a case from the same scenario where the similarity of incorrect actions is 1. Figure 11 illustrates an example from the Uni-Widget Instruction Unification scenario where the similarity of incorrect actions is 0, while Figure 12 presents a case from the same scenario where the similarity of incorrect actions is 1.

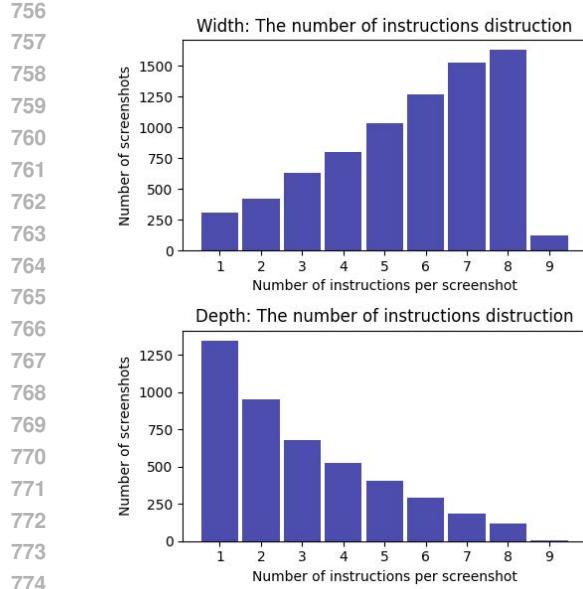


Figure 7: Instructions per screenshot distribution

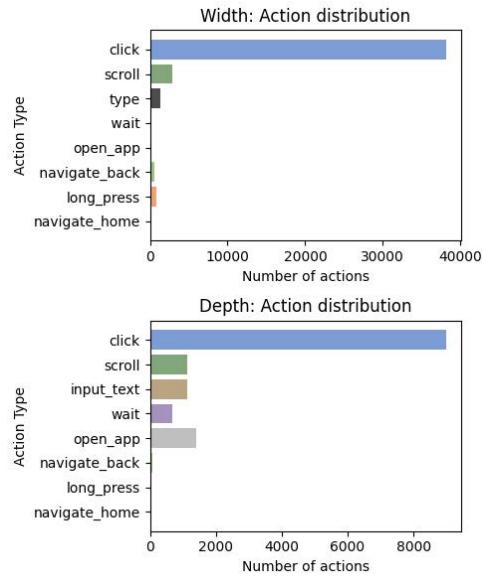


Figure 8: Action distribution

Table 5: The proportion of states in the four stages for 11 agents across three tasks.

Task	Model	Width				UWIU			
		LS	IS	PS	ES	LS	IS	PS	ES
Maps	ShowUI-2B	68	<u>24</u>	8	0	70	0	0	<u>30</u>
	OS-Atlas-4B-Pro	<u>31.03</u>	27.59	37.93	3.45	54.55	<u>18.18</u>	9.09	<u>18.18</u>
	OS-Atlas-7B-Pro	17.24	10.34	<u>24.14</u>	48.28	36.36	13.64	13.64	<u>34.36</u>
	SeeClick	44.83	<u>34.48</u>	17.24	3.45	81.25	0	6.25	<u>12.5</u>
	Qwen2-VL-7B-Ins	4	44	<u>36</u>	16	57.14	14.29	9.52	<u>19.05</u>
News	ShowUI-2B	64	24	8	4	48	<u>24</u>	4	24
	OS-Atlas-4B-Pro	20	20	36	<u>24</u>	48	<u>28</u>	0	24
	OS-Atlas-7B-Pro	4	16	<u>32</u>	48	<u>20</u>	<u>20</u>	12	48
	SeeClick	24	36	<u>28</u>	12	52.63	10.53	0	<u>36.84</u>
	Qwen2-VL-7B-Ins	12	<u>28</u>	48	12	<u>36</u>	20	4	40
Shopping	ShowUI-2B	67.44	<u>30.23</u>	2.33	0	59.38	12.5	0	<u>28.12</u>
	OS-Atlas-4B-Pro	<u>25.58</u>	<u>25.58</u>	44.19	4.65	46.88	15.62	12.5	<u>25</u>
	OS-Atlas-7B-Pro	2.33	13.95	41.86	<u>41.86</u>	<u>21.88</u>	15.62	12.5	50
	SeeClick	28.6	<u>30.23</u>	37.21	13.95	50	19.23	3.85	<u>26.92</u>
	Qwen2-VL-7B-Ins	23.26	<u>32.56</u>	39.53	4.65	<u>34.38</u>	25	3.12	37.5

A.6 PROMPT

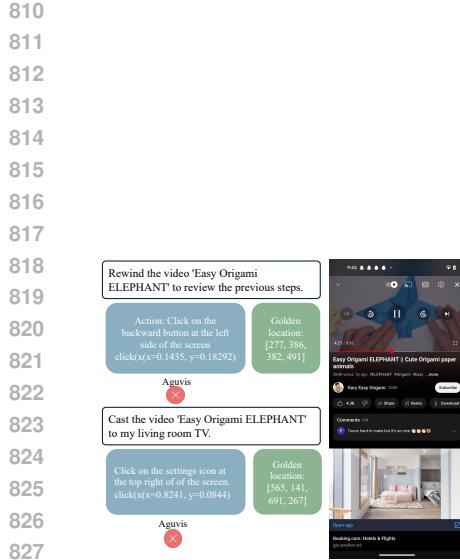


Figure 9: Similarity=0 in the MWAM dimension.

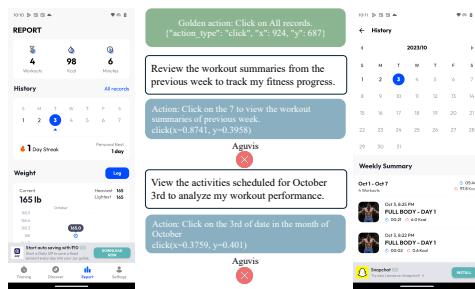


Figure 11: Similarity=0 in the UWIU dimension.



Figure 10: Similarity=1 in the MWAM dimension.

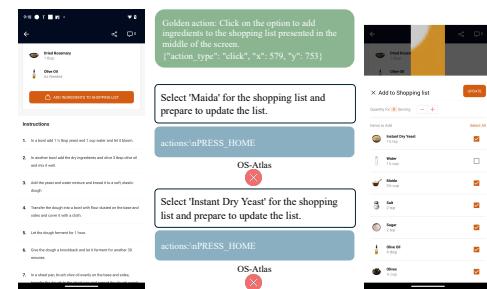


Figure 12: Similarity=1 in the UWIU dimension.

864
865
866
867
868
869
870
871

872 Prompt for constructing the trajectory tree dataset.
873

874 You are a mobile expert who excels at interacting with elements on mobile screens to
875 complete tasks. I have a task for you, and I hope you can use your extensive knowledge
876 to identify interactive elements on mobile screens. I will provide you with the following
877 information:

878 1. The type of action currently being executed, which can be one of five types: CLICK,
879 SCROLL, TYPE, PRESS_BACK, and LONG_PRESS. You need to choose an action
880 that can interact with the current screen.

881 2. Analysis of the mobile screen, which corresponds to the marked boxes in the images.
882 Your task is to identify five interactive elements on the current mobile screen. The
883 output should include four parts:

884 1. **Sub-Instruction:** Identify the interactive elements and generate natural language
885 instructions for interacting with these elements. The instructions should be concise,
886 clear, and executable, and must include critical details such as filenames, times, or
887 other content as they appear on the screen. For example: "Scroll left to open the app
888 drawer, displaying all installed applications on the device", "Click the chat interface,
889 allowing the user to view and participate in conversation", "Type the username 'Agent',
890 preparing for the next step in logging into the account".

891 2. **Analysis:** Analyze possible subsequent operations based on the current interface and
892 action instructions. This analysis should involve step-by-step reasoning, considering
893 potential changes on the screen and actions that can be taken after these changes. For
894 example: "After clicking the plus button, a dropdown menu appears with an option
895 to create a document. I can select this option to create a new document. First, I need
896 to name the document, then enter content into the document, and finally save the
897 document and exit".

898 3. **High-Level Instruction:** Based on the analysis results, envision a high-level task
899 that can be completed within the current interface. There are two types of High-Level
900 Instructions: Task-Oriented: Completing a series of operations to achieve a specific
901 goal. Question-Oriented: Performing a series of operations and deriving an answer to
902 a specific question. For example: Share my favorite Book "the Queen's Gambit" to my
903 Friend Natalie larson over her gmail address -natalie.larson1998@gmail.com from the
904 PocketBook app. Ensure that the High-Level Instruction is executable by including all
905 critical specifics, such as filenames, relevant timings, or required details.

906 4. **UI item:** Based on the current page parsed result and action instructions, identify the
907 corresponding UI item and provide the specific number.

908 You only need to return a dictionary formatted as follows: { "Sub-Instruction": "xxx",
909 "Analysis": "xxx", "High-Level-Instruction": "xxx", "UI item": x }

910 Current screen analysis:

911
912
913
914
915
916
917

Figure 13: Prompt for constructing the trajectory tree dataset.

918
919
920
921
922

Prompt for reasoning the correct golden action.

923 You are a GUI task expert, I will provide you with a low-level instruction, a golden ui
 924 with its corresponding ID.
 925 Low-level instruction:
 926 UI ID:
 927 Please generate the action for the next step.
 928 Candidate Actions:
 929 "action_type": "click", "ui": `ui_idx`,
 930 "action_type": "long_press", "ui": `ui_idx`,
 931 "action_type": "type", "text": `text_input`,
 932 "action_type": "scroll", "direction": `up, down, left, or right`,
 933 "action_type": "navigate_home",
 934 "action_type": "navigate_back",
 935 "action_type": "open_app", "app_name": `app_name`,
 936 "action_type": "wait",
 937 "action_type": "status", "goal_status": `"successful", "infeasible"`,
 938 You need to generate a script in the form: actions: ACTION
 939 Make sure to consider the details in the screenshot and the task requirements to create
 940 an accurate and functional script."

941
942
943
944
945
946
947
948

Figure 14: Prompt for evaluating whether actions correctly execute low-level instructions.

949
950

Prompt for reasoning the correct golden action.

951 You are a mobile expert who excels at interacting with elements on mobile screens to
 952 complete tasks. I have a task for you, and I hope you can use your extensive knowledge
 953 to identify interactive elements on mobile screens. I will provide you with the following
 954 information:
 955 1. A low-level instruction, which we will follow to perform actions on the current
 956 screen.
 957 2. The type of action currently being executed, which can be one of two types: CLICK
 958 or LONG_PRESS. You need to determine whether this action can fulfill the current
 959 low-level instruction.
 960 3. The current screen environment, with the position where the action(click and
 961 long_press) needs to be executed marked by a red box.
 962 I will provide you with a screenshot, along with the low-level instructions and the
 963 action to be executed. Your task is to determine whether the current action brings us
 964 closer to achieving the low-level instruction. If the current action contributes to the
 965 realization of the low-level instruction, answer "Yes"; otherwise, answer "No".
 966 You only need to return a dictionary formatted as follows: { "Analysis": "xxx",
 967 "Correct": Yes/No }

968
969
970
971

Figure 15: Prompt for evaluating whether actions correctly execute low-level instructions.

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989

990 Prompt for evaluating whether low-level instructions solve high-level instructions.

991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008

You are a mobile expert who excels at interacting with elements on mobile screens to complete tasks. I have a task for you, and I hope you can use your extensive knowledge to identify interactive elements on mobile screens. I will provide you with the following information:

1. A high-level instruction, which is our ultimate goal to be executed.
2. A low-level instruction, which we will follow to perform actions on the current screen.
3. The current screen environment, with the position where the action needs to be executed marked by a red dot.

I will provide you with a screenshot, along with the high-level and low-level instructions to be executed. Your task is to determine whether the current low-level instruction brings us closer to achieving the high-level instruction. If the current low-level instruction contributes to the realization of the high-level instruction, answer "Yes"; otherwise, answer "No".

You only need to return a dictionary formatted as follows: { "Analysis": "xxx", "Correct": Yes/No }

1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

Figure 16: Prompt for evaluating whether low-level instructions solve high-level instructions.

1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034 Prompt for classifying trajectories into specific app tasks.
 1035
 1036 You are a GUI agent. I'll give you a total goal, a screenshot, and some categories of
 1037 apps, and I'll ask you to choose the closest to the general goal among those categories.
 1038 ## Output Format
 1039 You only need to return a dictionary formatted as follows: { "Analysis": "xxx",
 1040 "Categories": "xxx" }
 1041 ## APP Categories
 1042 1. Shopping
 1043 2. Productivity & Office
 1044 3. Other
 1045 4. Files
 1046 5. Transportation
 1047 6. Health & Fitness
 1048 7. Recipes
 1049 8. Flights
 1050 9. Clock & Alarms
 1051 10. Reminders
 1052 11. Voice recording
 1053 12. Education
 1054 13. Books
 1055 14. Email
 1056 15. Calendar
 1057 16. Notes & Todos
 1058 17. Maps
 1059 18. Videos
 1060 19. News
 1061 20. Meditation
 1062 21. Weather
 1063 22. Finance
 1064 23. Art & crafts
 1065 24. Gardening
 1066 25. Contacts
 1067 26. Drawing
 1068 27. Music
 1069 28. Real estate
 1070 29. Messaging
 1071 ## Total Goal

Figure 17: Prompt for classifying trajectories into specific app tasks.

1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079