TransBench: Breaking Barriers for Transferable Graphical User Interface Agents in Dynamic Digital Environments

Yuheng Lu[1*], Qian Yu[1*], Hongru Wang[2*], Zeming Liu[1†], Wei Su[3], Yanping Liu[3], Yuhang Guo[3], Maocheng Liang[1], Yunhong Wang[1], Haifeng Wang[4]

[1]Beihang University, Beijing, China [2]The Chinese University of Hong Kong, China
[3]Beijing Institute of Technology, Beijing, China [4]Baidu Inc., Beijing, China
`{20375061, qiantraceyyu, zmliu}@buaa.edu.cn` `hrwang@se.cuhk.edu.hk`

Abstract

Graphical User Interface (GUI) agents, which autonomously operate on digital interfaces through natural language instructions, hold transformative potential for accessibility, automation, and user experience. A critical aspect of their functionality is *grounding* — the ability to map linguistic intents to visual and structural interface elements. However, existing GUI agents often struggle to adapt to the dynamic and interconnected nature of real-world digital environments, where tasks frequently span multiple platforms and applications while also being impacted by version updates. To address this, we introduce `TransBench`, the first benchmark designed to systematically evaluate and enhance the transferability of GUI agents across three key dimensions: *cross-version transferability* (adapting to version updates), *cross-platform transferability* (generalizing across platforms like iOS, Android, and Web), and *cross-application transferability* (handling tasks spanning functionally distinct apps). `TransBench` includes 15 app categories with diverse functionalities, capturing essential pages across versions and platforms to enable robust evaluation. Our experiments demonstrate significant improvements in grounding accuracy, showcasing the practical utility of GUI agents in dynamic, real-world environments. Our code and data will be publicly available at `TransBench`.

1 Introduction

GUI (Graphical User Interface) agents (Zhang et al., 2024), which are autonomous agents acting in the digital world via operating on GUIs, enables users to accomplish complex tasks through natural language instructions (Chen et al., 2024a; Ma et al., 2024; Baechler et al., 2024; Hong et al., 2024; Wu et al., 2024b; Wang et al., 2024a). These agents locate and manipulate multimodal GUI elements (i.e., buttons, icons, and menus) (Kapoor et al., 2025) and autonomously execute corresponding operations (e.g., clicking, scrolling) (Gao et al., 2024a) across diverse interfaces given the user instructions (Lu et al., 2024b; Mukhtar, 2025). By translating natural language instructions into precise actions, GUI agents democratize access to digital systems, offering transformative potential for accessibility, automation, and user experience. A core aspect of their functionality is *grounding*: the ability to map linguistic intents from instruction to visual and structural interface components. It is crucial to ensure effective grounding, as failures in accurately interpreting and localizing GUI elements propagate to downstream execution errors, rendering even sophisticated action planning futile.

While prior work has advanced GUI agents' grounding capabilities, existing approaches focus exclusively on platform-specific settings, such as mobile apps (e.g., GUI-Odyssey (Lu et al., 2024a), AUITestAgent (Hu et al., 2024)), desktop interface (e.g., AssistGUI (Gao et al., 2024a)), and web environments (e.g., Mind2web (Deng et al., 2024)). However, real-world applications operate dynamically: they span multiple platforms (e.g., iOS, Android, Web) and evolve continuously, with version updates frequently altering GUI layouts and functionalities. Meanwhile, user instructions often span applications with relevant or distinct functionalities, such as requesting "comparing products on *Amazon* and *Alibaba* with its review videos on *Youtube*" and implicitly assume cross-version or cross-platform consistency. This challenge highlights a crucial aspect of grounding for GUI agents — transferability. Without it, rigid version-, platform-, or app-specific grounding fails to generalize, making agents brittle in practice.

To address this issue, we first formally identify

* Equal contribution
† Corresponding author: Zeming Liu.

Findings of the Association for Computational Linguistics: ACL 2025, pages 12464–12478

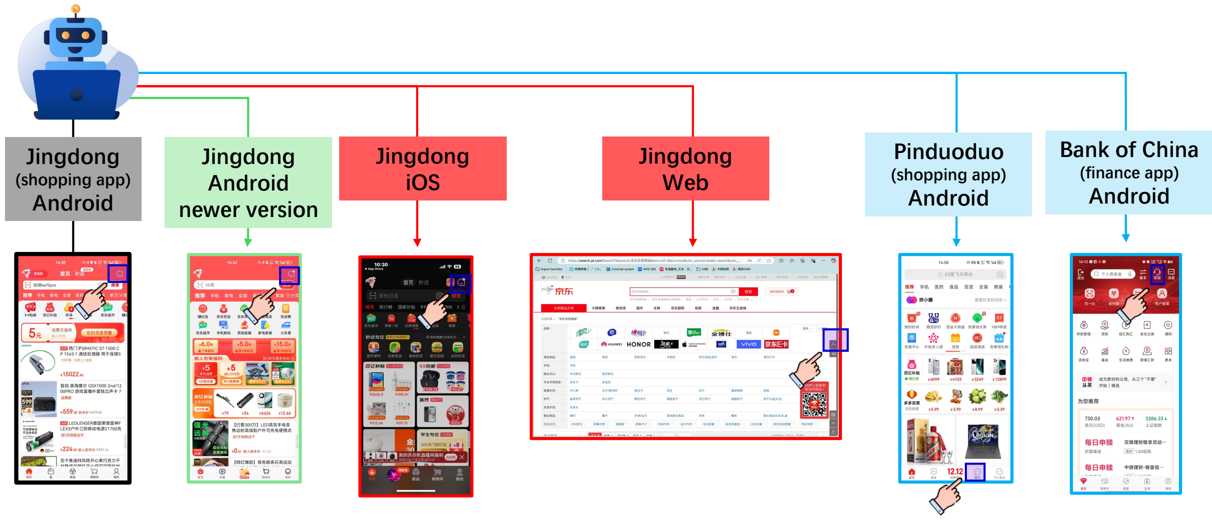


Figure 1: Interpretation of Transferability's three aspects. Green means *cross-version transferability*: transferring the knowledge learned from the homepage of Jingdong (a Chinese shopping app) from Android version 12.0.0 to a newer Android version, 13.6.8. Red means *cross-platform transferability*: transferring from the Android version of Jingdong to its iOS version 13.8.1 and Web version. Blue means *cross-application transferability*: transferring from Jingdong to other apps with the same functionality (e.g., shopping: Pinduoduo) or with different functionality (e.g., Finance: Bank of China)

three levels of transferability in terms of grounding capability of GUI agents, as shown in Figure 1: 1) *cross-version transferability*: localizing GUI elements despite interface changes from version updates; 2) *cross-platform transferability*: transferring grounding knowledge between platforms with divergent interaction patterns; and 3) *cross-application transferability*: generalizing from interchangeable features to partially overlapping or functionally distinct ones. Therefore, these dimensions collectively determine whether the GUI agents can generalize beyond narrow, static settings to cross-version, cross-platform, and cross-application workflows.

To systematically evaluate and enhance the transferability at these three levels, we introduce `TransBench`, the first benchmark that addresses version and platform discrepancies with abundant real-world applications. Specifically, we carefully design a data collection pipeline in three consecutive steps: 1) screenshot collections, ranging from Android (including multiple versions), iOS [1], and web platforms; and 2) bounding boxes annotations; and 3) user instruction generation. We conduct rigorous quality control and human verification to ensure the quality and diversity of our benchmarks. We hope our dataset not only provides a robust foundation but also sets a new standard for future research in transferability, enabling the development of more adaptive and generalizable GUI agents across dynamic digital environments. Overall, our contributions can be summarized as follows:

- We are the first to identify and highlight the challenge of transferability in grounding tasks, focusing on enabling GUI agents to execute multi-app tasks while adapting to version updates and platform differences.

- To support this, we design `TransBench`, a benchmark featuring datasets with screenshots of each applications essential pages across versions and platforms, covering 15 categories with diverse functionalities.

- Using `TransBench`, our experiments showcase significant progress in grounding accuracy, highlighting increased robustness and practicality for transferability in dynamic digital environments.

2 Related work

2.1 GUI Agents Datasets

GUI agent datasets play a crucial role in evaluating model performance and enhancing agents' abilities to understand GUI elements and execute tasks across diverse applications (Liu et al., 2023; Chen et al., 2024b; Liu et al., 2024b). Most of the existing benchmarks tend to specialize in specific

[1] Old versions for iOS applications are unavailable.

platforms, such as web (Deng et al., 2024), mobile (Hu et al., 2024; Lu et al., 2024a), and desktop (Gao et al., 2024a). For example, Mind2Web (Deng et al., 2024) is designed for web-based environments, while Android-specific datasets include PixelHelp (Li et al., 2020), MoTIF (Burns et al., 2022), GUI Odyssey (Lu et al., 2024a), and MobileViews (Gao et al., 2024b). Only in recent works, such as VisualAgentBench (Liu et al., 2024c) and WebHybrid (Gou et al., 2024), has the importance of cross-platform evaluation been widely recognized. However, they primarily concentrate on structural differences between different platforms, overlooking other levels of transferability for GUI agents, such as cross-version transferability and cross-application transferability. To the best of our knowledge, no existing dataset simultaneously tackles all these challenges. It is believed that our proposed `TransBench` can fulfill this blank by providing a benchmark that comprehensively and systematically evaluates GUI agents across these crucial dimensions.

2.2 GUI Agents Models

GUI agents play a crucial role in enabling intelligent automation, assisting users in navigating digital environments, and improving human-computer interaction (Lu et al., 2024b; Mukhtar, 2025). Recent advancements in GUI agent models have significantly improved their ability to understand complex interface layouts and user interactions (Wang et al., 2024f). For example, UI-BERT (Bai et al., 2021) enhances agents' comprehension of user intent and interface structures by leveraging contextual representations, and Auto-GLM (Liu et al., 2024a) builds on this by integrating both textual and visual data, improving adaptability but at the cost of increased computational demands However, they still struggle with generalizing to unseen or frequently changing layouts. Besides that, MobileVLM (Chu et al., 2023) enhances task execution efficiency in mobile applications, while MobileAgent (Wang et al., 2024b) leverages multimodal data to handle multi-step commands. Furthermore, AutoMobileGPT (Yang et al., 2023) advances natural language task execution by enabling seamless interaction across diverse applications. As the need for GUI agents that can seamlessly adapt across different app versions, platforms, and functionalities continues to grow, advancing their transferability remains a crucial research challenge. Furthermore, addressing

Name	Transferability			Lan
	Version	*Platform*	*Application*	
Mind2Web (Deng et al., 2024)	✗	✗	✗	en
PixelHelp (Li et al., 2020)	✗	✗	✗	en
MoTIF (Burns et al., 2022)	✗	✗	✗	en
GUI Odyssey (Lu et al., 2024a)	✗	✗	✓	en
E-ANT (Wang et al., 2024c)	✗	✗	✗	en
Mobile3M (Wu et al., 2024a)	✗	✗	✗	ch
MobileViews (Gao et al., 2024b)	✗	✗	✗	ch
VisualAgentBench (Liu et al., 2024c)	✗	✓	✗	en
WebHybrid (Gou et al., 2024)	✗	✓	✗	en
TRANSBENCH (Ours)	✓	✓	✓	ch

Table 1: Comparison between TransBench to other GUI agent datasets from transferabilities' three aspects, including cross-version, cross-platform, and cross-application. "Lan" stands for "Language" and "ch" means targeting Chinese apps).

the complexities of varying user interfaces and interaction patterns will be key to ensuring robust performance in real-world applications.

3 `TransBench` Construction

It is difficult to directly leverage existing benchmarks as our seed dataset, since there are two significant challenges. First of all, current datasets are typically constructed as task sets and lack critical metadata, such as app names, page titles, version numbers, and platform details. However, these details are crucial for the study of transferability, as it depends on accurately identifying and comparing interface variations across different versions, platforms, and apps. Secondly, a comprehensive evaluation of transferability requires establishing correspondence relationships across multiple dimensions, such as mapping an app's current version to its previous versions and linking different platform-specific versions of the same app with the same pages. Existing datasets typically confine tasks within a single version or platform, making it infeasible to be reused as an evaluation of transferability. Therefore, we provide a detailed data collection pipeline for `TransBench` in this section (as shown in Figure 2), alongside the formal task definition. Table 1 shows the detailed comparison between `TransBench` with other popular benchmarks.

3.1 Task Definition

The input is a user instruction u and a screenshot s_i^j where i stands for i_{th} APP and j means j_{th} screenshot of this APP. Screenshot s_i^j is combined with a set of instructions $\{u_1, u_2, ..., u_m\}$, which of each is associated with a ground truth bounding

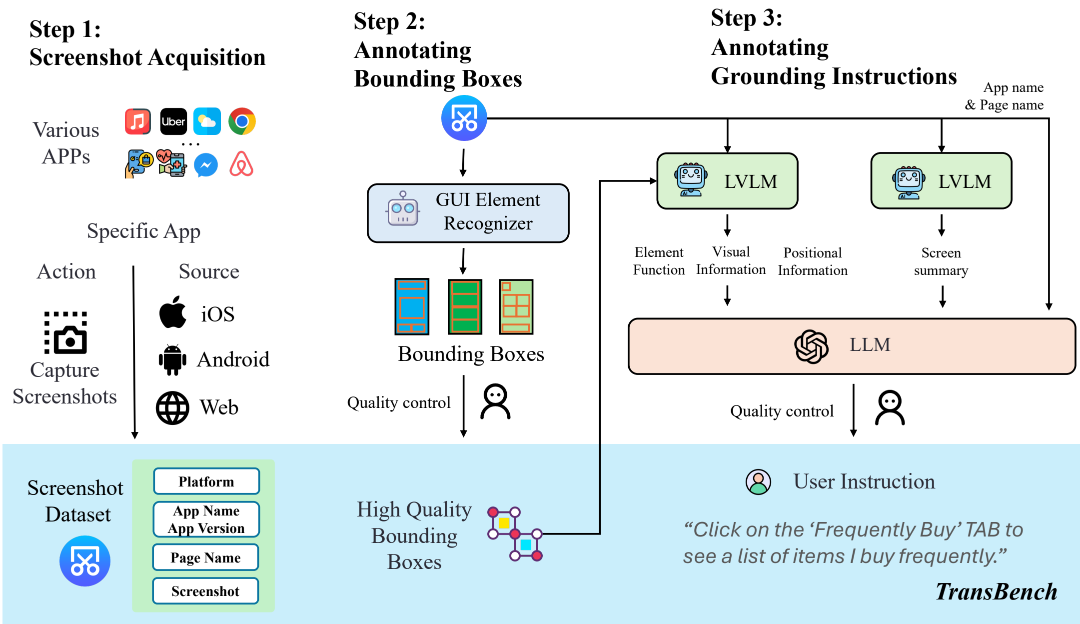


Figure 2: Interpretation of data collection process. The blue box represents our proposed benchmark -TransBench, which consists of three parts: ScreenShot Acquisition, Annotating Bounding Boxes, and Annotating Grounding Instructions. Platform means iOS, Android, and Web. Page names are manually divided into page names according to human semantics, such as "Shopping cart," "My page," "Home," "Comments," and so on, which usually have similar functions.

box $b_k = (x_{\min}, y_{\min}, x_{\max}, y_{\max})$. The agent's goal is to output a coordinate (x, y). The prediction is correct if (x, y) falls within the corresponding bounding box b_k. The objective is to improve the accuracy of GUI grounding across varying versions, platforms, and applications.

3.2 Step 1: Screenshot Acquisition

To cover diverse user instructions and applications, we identify 81 commonly used multi-platform applications in practice drawn from both previous studies and everyday usage, including shopping, video streaming, social networking, travel, lifestyle, maps, music, communication, finance, email, reading, education, camera, fitness, and utility tools. A complete list of applications is provided in the Appendix 11. Furthermore, considering the different levels of transferabilities, we define two types of screenshots: 1) fundamental screens that are common across most applications (i.e., homepage, message page, user profile page), allowing agents to establish a basic understanding of frequently encountered GUI elements; and 2) domain-specific screenshots which capture the unique functionalities of each application, enabling agents to adapt to specialized tasks. Consequently, we successfully collect the seed dataset, which includes a total of 1,459 screenshots: 825 from Android (covering both new and old versions), 429 from iOS, and 205 from web platforms. Tables listing the names of the applications and the titles of their corresponding pages are provided in the Appendix A.3.

3.3 Step 2: Annotating Bounding Boxes

Bounding boxes are essential for identifying and localizing GUI interface components such as buttons or input fields (Gou et al., 2024). In this way, we first utilize an automated annotation tool to generate preliminary bounding boxes, providing a foundational layer for the following work. After that, manual verification is conducted to address any discrepancies identified in the automated process.

Automatic Annotations. We use OmniParser (Lu et al., 2024b) to automatically identify the bounding boxes due to its strong GUI element recognition capabilities. This automated process enables efficient identification of key GUI elements, reducing the manual workload in the initial stages. Additionally, an automated filtering process is applied to exclude non-essential elements, such as status bars or advertisements. This filtering step ensures that the dataset remains focused on GUI elements relevant for evaluating interaction capabilities.

Manual Verification. Following automated annotation, manual verification is performed by four well-educated human annotators. To streamline this process, we develop a specialized annotation tool, **GUILabeller**[2], designed to facilitate flexible and efficient manual adjustments. Each human annotator is required to review each bounding box to identify whether it correctly encapsulates the intended GUI element. If discrepancies are found, the annotators manually adjust or redraw the bounding box to ensure precision. Additionally, particular attention is given to verifying the semantic equivalence of GUI components (Gou et al., 2024), which refers to cases where multiple GUI elements might belong to a larger GUI component and trigger identical outcomes. To address this issue, an additional larger bounding box is added during verification to encapsulate semantically equivalent elements, forming a hierarchical GUI element structure.

3.4 Step 3: Annotating Grounding Instructions

Following the completion of bounding box annotation, resulting in over 65,000 bounding boxes, the subsequent step is to generate high-quality grounding instructions while minimizing human intervention to ensure diversity and accuracy. This process is structured into three key steps, including extracting bounding box attributes, generating screen summaries, and constructing grounding instructions. Each step's prompt details can be found in Appendix A.1[3]. Manual verification is performed at the end to ensure correctness. Data examples can be found in Appendix A.2.

Bounding Box Attributes Acquisition. Screenshots and their corresponding bounding boxes are processed to Qwen2VL (Wang et al., 2024e) to obtain three-dimensional attributes (inspired by ARIA-UI (Yang et al., 2024)), including visual features, positional relationships, and functional characteristics.

Screen Summaries Generation. Each screenshot, alongside its relevant metadata such as the application name and page title, is incorporated into prompts designed for Qwen2VL to generate screen summaries. These summaries synthesize both visual and contextual information, providing a holistic understanding of the interface's layout and functionality, which serves as the foundation for generating grounding instructions.

[2]Details of our tool can be found in Appendix A.4

[3]All prompts can be found in the Appendix if not stated.

Statistics	Android old	Android new	iOS	Web
# Apps	77	80	81	47
# Screenshots	393	432	429	205
# Bounding Boxes	17,455	19,384	14,477	14,341
# Checked Instructions	5,696	6,305	6,046	4,191
# Fundamental Pages	300	300	300	150
# Domain-Specific Pages	93	132	129	55
Avg. # Screenshots	5.1	5.4	5.3	4.4
Avg. # Bounding Boxes	226.7	242.3	178.7	305.1
Avg. # Instructions	74.0	78.8	74.6	89.2
Avg. #Fundamental Pages	3.9	3.8	3.7	3.2
Avg. # Domain-Specific Pages	1.2	1.7	1.6	1.2

Table 2: The data statistics of our proposed `TransBench`. Avg. meas average on single App. Bounding boxes include boxes with unchecked instructions and boxes with checked Instructions.

User Instruction Construction. Using the bounding box attributes and screen summaries obtained from previous steps, we prompt Qwen-plus (for its strong reasoning, imagination, and instruction-following abilities) to construct the required user instruction. The generation process leverages multidimensional visual information provided by the visual model, as well as commonsense information associated with application names and page titles as prior knowledge.

Quality Control. Finally, to ensure data quality, manual verification is performed. Four human annotators together address inconsistencies and refine instructions while necessary. As a result, more than 22,000 high-quality grounding instructions are refined, and only these refined instructions are used in the following experiments. Furthermore, following Liu et al. (2020), Shi et al. (2023) and Wang et al. (2024d), we employ human evaluations to assess data accuracy. An instruction is considered correct only if it precisely corresponds to the single corresponding bounding box, while pointing to more than one bounding box is considered incorrect. The final evaluation yields an average score of *95.5%*, confirming the high quality of the grounding instructions in the dataset.

3.5 Data Statistics

Table 2 illustrates the statistics of `TransBench`. Specifically, it includes up to 81 apps across Android (old and new versions), iOS, and Web platforms, with a total of 1,459 screenshots and over 65,000 bounding boxes. On average, each app contains a maximum of 5.4 screenshots, 305.1 bounding boxes, and 89.2 instructions. The dataset balances fundamental pages (common across apps) and domain-specific pages (unique to each app), ensuring broad coverage of GUI elements and tasks.

To assess meaningful differences between versions, we manually analyzed screenshots to quantify interface changes. Among all screenshots, 74.6% showed significant differences. We further examined 20 pairs of old and new version screenshots, classifying elements into six categories: layout or icon/text changes, both layout and icon/text changes, additions, deletions, and no change. Results are summarized in the table 3.

Element Type	Percentage
Layout Change	10.0%
Text/Icon Change	2.0%
Layout and Text/Icon Change	13.6%
Addition	30.8%
Deletion	14.0%
No Change	29.6%

Table 3: Differences between Android old version and Android new version.

4 Experiment

4.1 Setup

Models. We select several top-performing LLMs on ScreenSpot. Specifically, we include Cogagent (Hong et al., 2024) (glm-4v (Zeng et al., 2024)), Seeclick (Cheng et al., 2024) (Qwen-VL (Bai et al., 2023)), Aria-UI (Yang et al., 2024) (Aria (Li et al., 2024)), OS-Atlas (Wu et al., 2024c) (Qwen2-VL (Wang et al., 2024e)), UGround (Gou et al., 2024) (Qwen2-VL), and Qwen2.5VL (Team, 2025). Among them, UGround and CogAgent have been updated compared to the versions in their paper, and Qwen2.5VL is the latest released model.

Implementation Details. Following (Huang et al., 2024; Zhuang et al., 2023; Cheng et al., 2024), we conduct evaluations under two configurations: 1) the Standard Set, where all baselines are tested and evaluated on our complete dataset. Evaluation details can be found in Appendix B.1. 2) the Finetuning Set, which utilizes the rich metadata of our dataset to construct different partitions, allowing models to be fine-tuned on the training sets of each partition and tested on the corresponding test sets. Details of the training set division and finetuning are available in Appendix B.2 and B.3.

4.2 Evaluation Metrics

In line with previous practices, we evaluate *grounding accuracy* on `TransBench` by determining a prediction to be correct if the predicted location is contained within the ground truth bounding box. Moreover, to compare the precision of click positions at a finer scale, we introduce *an average distance evaluation metric* D, which refers to the Euclidean distance between the predicted click position (x_i, y_i) and the center of the Ground Truth bounding box $(\hat{x_i}, \hat{y_i})$. To accommodate various interface sizes and ensure readability, we uniformly scale x, y with screen width and height to a range of 0-100 by $x' = \frac{x}{W} \times 100$ and $y' = \frac{y}{H} \times 100$, thereby enabling the comparison of precision across screens with varying width and height.

$$D = \frac{1}{N} \sum_{i=1}^{N} \left\| (\hat{x}_i', \hat{y}_i'), (x_i', y_i') \right\| \quad (1)$$

4.3 Results

Table 4 presents the `TransBench` evaluation results of different LLMs across different platforms and app versions. Several key observations can be made:

Overall, Qwen2.5VL achieves the highest accuracy, while UGround minimizes distance. Qwen2.5VL consistently outperforms all other models in terms of accuracy across all settings, achieving the highest overall accuracy (89.62%). Meanwhile, UGround exhibits the lowest distance metric in most cases, except on the Web, where Qwen2.5VL attains the smallest distance (7.35). Other models show a notable performance gap compared to these two, with SeeClick and Cogagent demonstrating particularly weak performance, as reflected in both accuracy and distance scores.

Models released at different times exhibit varying performance across different versions of Android, with newer models generally performing better on the Android new version. In detail, older models such as CogAgent (76.04% old vs 75.70% new) and SeeClick (46.86% old vs 46.42% new) demonstrate better performance on android old version. Conversely, newer models, including Aria-UI, OS-Atlas, UGround, and the top-performing Qwen2.5VL (88.87% old vs 90.29% new) achieve higher results on new versions. These results suggest substantial differences between app versions, which potentially

Models	General		Android Overall		Android Old		Android New		iOS		Web	
	acc↑	dis↓	acc↑	dis↓	acc↑	dis↓	acc↑	dis↓	acc↑	dis↓	acc↑	dis↓
Cogagent	72.16	14.99	75.86	14.13	76.04	13.99	75.70	14.25	68.61	16.28	66.69	15.58
SeeClick	39.90	22.72	46.63	19.27	46.86	19.07	46.42	19.45	43.57	19.71	15.37	36.96
Aria-UI	77.51	9.26	81.18	8.99	80.97	9.10	81.38	8.89	77.61	9.61	66.86	9.55
OS-Atlas	81.37	8.36	84.56	8.24	84.52	8.10	84.60	8.36	79.64	8.89	74.76	7.97
UGround	<u>84.18</u>	**7.23**	<u>87.34</u>	**6.89**	<u>86.94</u>	**6.89**	<u>87.71</u>	**6.89**	<u>82.43</u>	**7.42**	<u>77.62</u>	<u>7.94</u>
Qwen2.5VL	**86.43**	<u>7.72</u>	**89.62**	<u>7.68</u>	**88.87**	<u>7.82</u>	**90.29**	<u>7.55</u>	**84.72**	<u>8.04</u>	**79.79**	**7.35**

Table 4: Accuracy rate (%) of different LLMs on `TransBench`.

come from evolving GUI components and interaction philosophies.

GUI agents tend to perform grounding best on Android, followed by iOS, with the worst performance on the web. As reported in Table 4, we can find that the performance on Android is always higher than iOS, and the performance of both Android and iOS is substantially better than Web interfaces, no matter which method is chosen. For example, Qwen2.5VL achieves 89.6%, 84.72%, and 79.79% accuracy on Android, iOS, and Web, respectively. The substantially weaker Web results indicate that GUI differences between mobile and web platforms pose a significant challenge, and the observed performance variations between Android and iOS platforms also underscore the inherent heterogeneity within mobile ecosystems.

Our proposed distance serves as a strong complementary evaluation metric to accuracy, offering a finer-grained assessment of precision in grounding tasks. Unlike accuracy, which simply checks whether a click falls within the bounding box, distance considers the exact position of the click relative to the box's center. It is observed that Cogagent exhibits a large discrepancy between accuracy and distance. Despite only a 7% accuracy gap compared to Aria-UI, its distance is 62% higher. Further inspection reveals that Cogagent interprets some tasks as already completed, although we precisely prompt it to perform a click action. Qwen2.5VL achieves the highest accuracy, but it underperforms UGround in terms of the distance metric. With more careful inspection, we speculate that this may be due to its absolute coordinate output (align with screenshot resolution), whereas UGround normalizes coordinates to a 0-1000 scale, making it more robust to varying screen resolutions.

5 Analysis

In this section, we fine-tune Aria-UI[4]. We use ARIA-UI as an experimental subject to address three key research questions. **RQ1:** How does transferability across versions impact the performance of GUI agents, and can fine-tuning on older versions improve adaptability to newer ones? (Sec 5.1) **RQ2:** To what extent can models generalize across platforms (i.e., from Android, iOS, to Web)? (Sec 5.2) **RQ3:** How do models perform when transferring knowledge across applications with varying functionalities, and what are the limitations in cross-application generalization? (Sec 5.3)

5.1 *Cross-Version Transferability* Evaluation

To investigate the cross-version transferability, we split our dataset into a training set (containing 5,000 samples of low-version Android data) and a test set (composed of high-version Android data, iOS data, and Web data). We then fine-tune the Aria-UI model using the training dataset, resulting in the Aria-UI-Android-old model. We provide the performance of it on the test set in Table 5. Training focused on comparing old and new versions demonstrates significant performance improvements across all platforms. Specifically, accuracy on the Android new version increases from 81.38% to 88.36%, on iOS from 77.61% to 82.57%, and the Web from 66.86% to 73.61%. Notably, the performance on Android new and iOS surpasses that of the second-best model, UGround, and approaches the performance of the top-performing Qwen2.5VL. This indicates strong transferability from old to newer versions, suggesting that finetuning on older versions can yield robust performance even after application updates. Furthermore, it highlights the potential of

[4]All the code and scripts will be open-sourced. Please see Appendix B for more details

Models	Android New		Android Old		iOS		Web	
	acc	dis	acc	dis	acc	dis	acc	dis
Base Model								
Aria-UI-Base	81.38	8.89	80.97	9.10	77.61	9.61	66.86	9.55
Fine-tuned Model								
Aria-UI-Android-old	**88.36**	**5.80**	**89.37**	**5.92**	<u>82.57</u>	7.74	73.61	<u>8.43</u>
Aria-UI-iOS	87.06	6.98	86.83	6.92	82.03	**7.09**	<u>73.66</u>	8.74
Aria-UI-General	<u>88.15</u>	<u>6.17</u>	<u>87.20</u>	<u>6.08</u>	**83.15**	<u>7.21</u>	**76.54**	**7.58**

Table 5: Accuracy rate (%) of Aria-UI after fine-tuning on the Different split of `TransBench`.

leveraging historical data to improve model adaptability across version changes and even different platforms.

In addition, we evaluated whether fine-tuning data from the old version improves a models performance on newly added UI elements in the new version. Using 271 randomly sampled new elements from the updated version, we compared the accuracy of the model before and after fine-tuning on the old Android version.

Model	New Element Accuracy (%) ↑
Aria-UI-Base	80.15
Aria-UI-Android-Low	87.50

Table 6: Aria-UI Accuracy Change for Newly Added Elements in New Version

The results in Table 6 indicate that the GUI elements newly added in the new version can benefit from fine-tuning based on the old version.

5.2 *Cross-Platform Transferability* Evaluation

To assess cross-platform transferability, we further create two additional training sets: 1) iOS split using 5000 samples of iOS data as the training set (Aria-UI-iOS); and 2) general split, which mixes all available data and randomly select 5000 samples as the training set to maintain consistency (Aria-UI-General). Furthermore, to evaluate transferability from the Web to other platforms, we create a Web split using 4,000 samples of Web data due to the smaller data scale as the training set (Aria-UI-Web).

General, iOS, and Android comparison. Table 5 shows the results of Aria-UI-iOS, Aria-UI-Android-old and Aria-UI-General. *It is observed that finetuning on Android data provides the most substantial performance gains across platforms.* For instance, iOS accuracy improves by 4.96% when fine-tuned on Android data, compared to 4.42% when fine-tuned on iOS data. This suggests that Android data, being more diverse and representative than iOS, offers better transferability to other platforms, and finetuning on it is highly effective. In addition, *we can found it is not easy to directly transfer from Android or iOS to Web without the web data* since the performance of Aria-UI-iOS and Aria-UI-Android-old is significantly worse than Aria-UI-General.

Models	Android New		Android Old		iOS		Web	
	acc	dis	acc	dis	acc	dis	acc	dis
Base Model								
Aria-UI-Base	81.38	8.89	80.97	9.10	77.61	9.61	66.86	9.55
Fine-tuned Model								
Aria-UI-Web	84.87	7.70	84.08	7.70	80.62	8.39	66.49	9.60

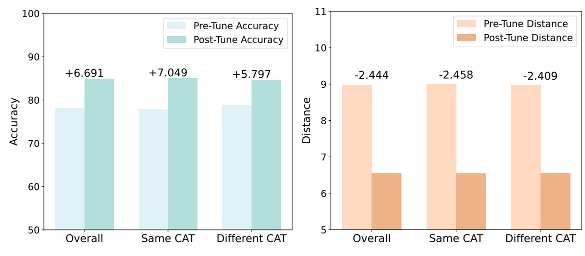
Table 7: Accuracy rate (%) of Aria-UI after fine-tuning on the Web split of `TransBench`.

Web results. Besides that, the result in Table 7 on the one hand shows that fine-tuning on the Web with 4,000 samples can not significantly enhance performance on the Web test set, but on the other hand, can improve performance on Android and iOS. Combined with our previous findings in Table 5, the Aria-UI-General model, fine-tuned on the general split, achieves the most significant improvement on Web (76.54%) and iOS (83.15%). This further confirms that diverse, multi-platform data is crucial for enhancing not only Web performance, but for achieving robust cross-platform transferability.

5.3 *Cross-Application Transferability* Evaluation

To evaluate cross-application transferability, we split app categories into two parts[5]: 1) the training set contains 18 Apps from the first seven categories; and 2) the test set, which is composed of 35 Apps from the same seven categories (i.e., Same CAT), and 28 Apps from other categories (i.e, Different CAT). Then we fine-tune the Aria-UI model on the training set and inferences at these two different test sets. Figure 3 shows the significant improvements in accuracy and reductions in the dis-

[5]The training set has 6247 samples and the test set has 15991 samples. We randomly selected 5000 samples from the training set for finetuning.



(a) Accuracy↑ (b) Distance↓

Figure 3: Sub-figure (a), (b) shows the variation of average accuracy and average distance after finetuning Aria-ui on App Split. "CAT" means category.

tance across both apps in *same category* and *different category*. However, the performance gains are slightly more pronounced for apps in same category, benefiting from the task similarity. Despite this, the difference between the two category types is not substantial, indicating that app similarity has less impact on transferability compared to platform and version differences. This suggests that while app-specific finetuning can yield improvements, the overall transferability of GUI agents is more influenced by platform and version adaptability.

6 Conclusion

In this paper, we explore three major fine-grained aspects of transferability (i.e., *cross-version, cross-platform, and cross-application*) of grounding capabilities for GUI agents to better accommodate diverse user instructions and complex real-world scenarios. To this end, we build the first comprehensive benchmark – `TransBench`, covering various applications spanning different versions and platforms. Our experimental results on a more fine-grained evaluation showcase that there is still a big gap between different levels of transferability, and we hope our benchmarks and new metrics can pave the way for more effective and adaptable GUI agents in practical applications.

Limitation

One notable limitation of our approach is the high computational requirements for training and finetuning the models. The extensive dataset, combined with the need for multi-dimensional partitioning and rigorous evaluation of transferability across versions, platforms, and applications, demands significant computational resources. This can pose challenges for researchers or organizations with limited access to high-performance computing infrastructure, potentially restricting the reproducibility and scalability of our methods.

Ethical Statement

This study adheres to ethical guidelines by ensuring the security and privacy of all data used during research. Screenshots collected from real applications are anonymized, and no sensitive user information is included. Additionally, manual annotations are performed under strict supervision to avoid human bias in data labeling. AI assistants, including DeepSeek and Qwen, are used to assist in understanding code and translating. Open-source components are used responsibly, with proper attribution given to their developers. The benchmark dataset will be shared following data-sharing agreements, promoting transparency while safeguarding any proprietary information, and ensuring responsible dissemination and compliance with ethical standards in academic research.

Acknowledgments

Thanks for the insightful comments and feedback from the reviewers. This work was supported by the National Natural Science Foundation of China (No. 62406015), and CCF-Baidu Open Fund (No. CCF-BAIDU202411).

References

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Carbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. 2024. Screenai: A vision-language model for ui and infographics understanding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 3058–3068. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. 2021. Uibert: Learning generic multimodal representations for ui understanding. In *International Joint Conference on Artificial Intelligence*.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond.

Andrea Burns, Deniz Arsan, Sanjna Agrawal, Ranjitha Kumar, Kate Saenko, and Bryan A. Plummer. 2022. A dataset for interactive vision language navigation with unknown command feasibility. In *European Conference on Computer Vision (ECCV)*.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2024a. Gui-world: A dataset for gui-oriented multimodal llm-based agents. *ArXiv*, abs/2406.10819.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. 2024b. Guicourse: From general vision language models to versatile gui agents. *arXiv preprint arXiv:2406.11317*.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. 2024. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935*.

Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. 2023. Mobilevlm: A fast, strong and open vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Difei Gao, Lei Ji, Zechen Bai, Mingyu Ouyang, Peiran Li, Dongxing Mao, Qinchen Wu, Weichen Zhang, Peiyi Wang, Xiangwu Guo, et al. 2024a. Assistgui: Task-oriented pc graphical user interface automation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13289–13298.

Longxi Gao, Li Zhang, Shihe Wang, Shangguang Wang, Yuanchun Li, and Mengwei Xu. 2024b. Mobileviews: A large-scale mobile gui dataset. *arXiv preprint arXiv:2409.14337*.

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

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. 2024. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14281–14290.

Yongxiang Hu, Xuan Wang, Yingchuan Wang, Yu Zhang, Shiyu Guo, Chaoyi Chen, Xin Wang, and Yangfan Zhou. 2024. Auitestagent: Automatic requirements oriented gui function testing. *ArXiv*, abs/2407.09018.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, et al. 2024. Planning, creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. *arXiv preprint arXiv:2401.17167*.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2025. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer.

Dongxu Li, Yudong Liu, Haoning Wu, Yue Wang, Zhiqi Shen, Bowen Qu, Xinyao Niu, Guoyin Wang, Bei Chen, and Junnan Li. 2024. Aria: An open multimodal native mixture-of-experts model. *ArXiv*, abs/2410.05993.

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

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. 2024a. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, Jiadai Sun, Xinyue Yang, Yu Yang, Zehan Qi, Shuntian Yao, Xueqiao Sun, Siyi Cheng, Qi Zheng, Hao Yu, Hanchen Zhang, Wenyi Hong, Ming Ding, Lihang Pan, Xiaotao Gu, Aohan Zeng, Zhengxiao Du, Chan Hee Song, Yu Su, Yuxiao Dong, and Jie Tang. 2024b. Visualagentbench: Towards large multimodal models as visual foundation agents. *ArXiv*, abs/2408.06327.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. 2024c. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*.

Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Towards conversational recommendation over multi-type dialogs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1036–1049, Online. Association for Computational Linguistics.

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

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. 2024b. Omniparser for pure vision based gui agent. *ArXiv*, abs/2408.00203.

Xinbei Ma, Zhuosheng Zhang, and Hai Zhao. 2024. CoCo-agent: A comprehensive cognitive MLLM agent for smartphone GUI automation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9097–9110, Bangkok, Thailand. Association for Computational Linguistics.

Hamid Mukhtar. 2025. Artificial intelligence techniques for human-machine interaction. In *Artificial Intelligence and Multimodal Signal Processing in Human-Machine Interaction*, pages 19–42. Elsevier.

Xiaoming Shi, Zeming Liu, Chuan Wang, Haitao Leng, Kui Xue, Xiaofan Zhang, and Shaoting Zhang. 2023. MidMed: Towards mixed-type dialogues for medical consultation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8145–8157, Toronto, Canada. Association for Computational Linguistics.

Qwen Team. 2025. Qwen2.5-vl.

Hongru Wang, Yujia Qin, Yankai Lin, Jeff Z. Pan, and Kam-Fai Wong. 2024a. Empowering large language models: Tool learning for real-world interaction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 29832986, New York, NY, USA. Association for Computing Machinery.

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

Ke Wang, Tianyu Xia, Zhangxuan Gu, Yi Zhao, Shuheng Shen, Changhua Meng, Weiqiang Wang, and Ke Xu. 2024c. E-ant: A large-scale dataset for efficient automatic gui navigation. *arXiv preprint arXiv:2406.14250*.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and Yunhong Wang. 2024d. A survey on data synthesis and augmentation for large language models. *ArXiv*, abs/2410.12896.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024e. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, abs/2409.12191.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Weinan Gan, Xingshan Zeng, Shuai Yu, Xinlong Hao, Kun Shao, Yasheng Wang, and Ruiming Tang. 2024f. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Jianfeng Liu, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024a. Mobilevlm: A vision-language model for better intra-and inter-ui understanding. *arXiv preprint arXiv:2409.14818*.

Qinzhuo Wu, Weikai Xu, Wei Liu, Tao Tan, Liujian Liujianfeng, Ang Li, Jian Luan, Bin Wang, and Shuo Shang. 2024b. MobileVLM: A vision-language model for better intra- and inter-UI understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10231–10251, Miami, Florida, USA. Association for Computational Linguistics.

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

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

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

Team Glm Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Ming yue Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiaoyu Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yi An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhenyi Yang, Zhengxiao Du, Zhen-Ping Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *ArXiv*, abs/2406.12793.

Chaoyun Zhang, Shilin He, Jiaxu Qian, Bowen Li, Liqun Li, Si Qin, Yu Kang, Ming-Jie Ma, Qingwei Lin, S. Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Large language model-brained gui agents: A survey. *ArXiv*, abs/2411.18279.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

A Data Collection

A.1 Prompt Details

The prompts we used are shown in Tabel 8, 9 and 10.

You are a powerful GUI recognizer, your mission is to accurately recognize GUI element on screenshot and output information.

You have two input images, first image is a crop of your target GUI element, second is a target GUI element zoomed-in view of full screenshot.

Output three types of information, including: Visual information such as "The vertical three-dot button, with text more at the bottom", Positional information such as "Next to the entry "28 YEARS LATER - Official Trailer", and the Functional information such as "Access more options for the video entry". The output should be in chinese and included in json dict: {"Visual":"visual information", "Positional": "positional information", "Functional": "functional information"}.

Table 8: The prompt used to generate the bounding box attributes, including visual, positional, and functional information.

You are a GUI annotator. The input screenshot is the *{page_title}* page of the *{app_name}* application. Please fully describe the page in Chinese.

Table 9: The prompts to generate the screen summaries using information of app names with page titles.

Your mission is to generate instructions that correspond to potential interactions when user want to act with the specified GUI element on current screen, such as instruction "watch video about happy dog." corresponding to a video about a happy dog in video list on the screen. Remember that your output should like a normal user instruction.

Your output based on three part of input information, first is a screen summary, second is the description of target GUI element, third is the current app name and page name. Input information may conflict or including some errors, neglect these conflicts or errors, ensure that the instructions you generate correspond uniquely to the GUI on the screen.

Input1: Screen summary is { "screen_summary": "*{screen_summary}*"}.
Input2: The target GUI element is *{caption_data}* .
Input3: The current app is *{app_name}* app, current page is *{page_name}* page.

The output should be in Chinese and included in json dict{"Instruction": Instructions that the user would say.}

Table 10: The prompts to generate the grounding instructions using information of screen summary, description of target GUI element, and app names with page titles. Caption data is a json dict like: {"Visual": "xxx", "Positional": "xxx", "Functional": "xxx"}

A.2 Example Description

Here are data examples (Figure 4, 5, and 6) for three platforms: Android new, iOS, and Web. In each example, there are several bounding boxes and the corresponding instructions. It is worth noting that the different colors are only for convenience in viewing more details and do not have different semantics.

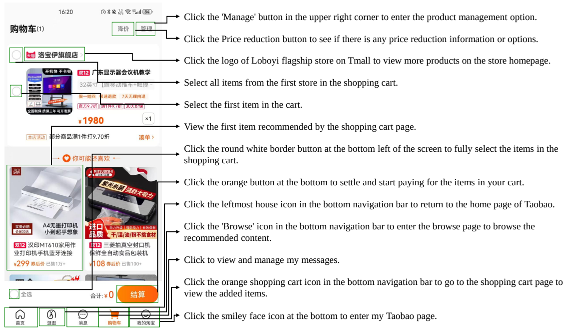


Figure 4: An example of Android version.

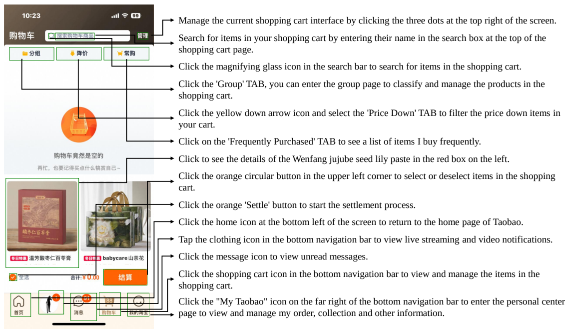


Figure 5: An example of iOS version.

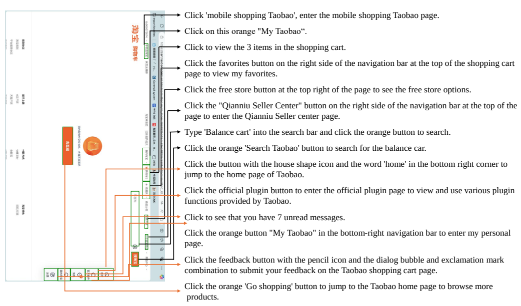


Figure 6: An example of the Web version.

A.3 Application Names and Page Titles

Table 11 and 12 respectively list each category's application names as well as page titles (comprised of both fundamental pages and domain-specific pages).

App Categories	APP Names
Shopping	*Taobao*, *Jingdong*, *Weipinhui*, *TMall*, *Pinduoduo*, *Dewu*, *Yitao*, *Alibaba1688*
Video Streaming	*Aiqiyi*, *Tencent Video,*, *Youku Video*, *bilibili*, *Mangou*, *Xigua*, *Tudou*, *Souhu*
Social Networking	*Douyin*, *Xiaohongshu*, *Kuaishou*, *Douyinjisu*, *Douyinhuoshan*, *Weibo*, *Jinritoutiao*
Travel	*Qunaer*, *Xiecheng*, *12306*, *Tongcheng*, *Feizhu*, *Zhixing*, *Tuniu*
Lifestyle	*eleme*, *meituan*, *dingdong*, *Hema*, *Didi*, *Dazhong*
Maps	*Gaode Maps*, *Baidu Maps*, *Tencent Maps*, *Beidou*
Music	*Netease*, *QQ Music*, *Kugou Music*, *Qishui Music*, *Migu*, *Kuwo*, *Bodian*, *Quanmin*
Communication	*QQ*, *Feishu*, *DingTalk*
Finance	*Bank of China*, *Bank of Construction*, *Alipay*
Email	*QQ Mail*, *Netease Mail*, *189 Mail*
Reading	*Kindle*, *Wechat Reading*, *Fanqie*, *Douban*, *Qimao*, *Netease Reading*
Education	*Wanciwang*, *Xindongfang*, *Momo*, *Hujiang*, *Baicizhan*
Camera	*Huangyou*, *Qingyan*, *Meitu*, *B612*, *Xingtu*
Fitness	*Keep*, *MeiriYoga*, *Yinghan*
Utility Tools	*Fanqie*, *Ticktick*

Table 11: List of all Apps and their corresponding names in `TransBench`

App Categories	Page Titles	
	Fundamental pages	Domain-specific pages
Shopping	*Home*, *Me*, *Message*	*Cart*, *Orders*
Video Streaming	*Home*, *Me*, *Search*	*Video*, *Full Screen*, *History*, *Advertisement*
Social Networking	*Home*, *Me*, *Search*	*Full Screen*, *Comments*
Travel	*Home*, *Me*	*Orders*, *Booking*, *Search*, *Flights*
Lifestyle	*Home*, *Me*	*Cart*, *Recommendations*, *Search*, *Details*, *Orders*
Maps	*Home*	*Details*
Music	*Home*, *Me*, *Search*	*Video*, *Comments*, *Favorites*
Communication	*Home*, *Me*	*Profile*, *Settings*, *Contacts*, *Moments*, *More*
Finance	*Home*, *Me*, *Search*	*Customer Service*
Email	*Home*	*Inbox*, *Emails*, *Compose*
Reading	*Home*, *Me*, *Search*	*Details*
Education	*Home*, *Search*	*Details*
Camera	*Home*	*Photo*, *Edit*
Fitness	*Home*, *Me*, *Search*	*Start Exercise*
Utility Tools	*Home*, *Me*	*Data Statistics*, *Add to-do Items*

Table 12: List of all App categories and their corresponding page titles in `TransBench`

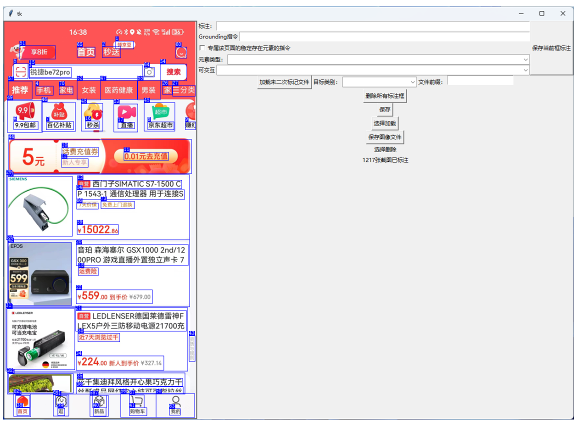


Figure 7: Enter Caption

A.4 Other details

Tools details. To quickly inspect the generated data, we develop a manual inspection tool in Figure 7 called GUILabeller, which uses Python and can run across different platforms. We will open-source this tool on GitHub.

Data collection pipeline details. To accelerate the generation speed, as well as due to resource constraints, we utilized a combination of online Qwen series models (including qwen-VL-plus, qwen-VL-max, and qwen-plus) and locally run Qwen2VL-72b model and Qwen2-72b model. Specifically, the locally run models are running on an NVIDIA A800 server cluster using int4 format quantization.

B Experiments

B.1 Evaluation Details

To ensure optimal performance for each model, all hyperparameters (e.g., temperature) are set consistently with their publicly released versions. We adapt our evaluation framework to each model and provide the evaluation scripts in our repository. Since different models utilize distinct prompt structures during training, we strictly follow the prescribed prompt formats for each model to achieve their best performance. Specifically, CogAgent requires the platform type as input, and we provide the correct platform type accordingly. We will open-source our testing scripts in the GitHub repository, which facilitates the easy addition of new agents for testing. The detailed prompts are in Table 13.

B.2 Training Set Division

Our dataset is annotated with five key dimensions: app names, app categories, page titles, app versions, and platform types, enabling multi-dimensional partitioning. As illustrated in the figure below, models fine-tuned on specific partitions are exclusively evaluated on their corresponding test sets to prevent data leakage.

Android-Low Partition: From the dataset, 5,696 low-version Android samples are selected as candidates. We randomly chose 5,000 for training, with the remaining 696 and an additional 16,542 samples used for testing.

iOS Partition: A total of 6,046 iOS samples are filtered as candidates. We randomly select 5,000 for training, leaving 1,046 and an additional 16,192 samples for testing.

Web Partition: From 4,191 Web platform samples, 4,000 are randomly chosen for training, with the remaining 191 and an additional 18,047 samples used for testing.

Normal Partition: We randomly select 5,000 samples from the entire dataset for training, using the remaining data for testing.

App Partition: To evaluate cross-app transferability, we first select the top 7 app categories with the most data. From these, 40% of the apps are reserved for testing, resulting in 6,247 candidate

ARIA-UI:

Given a GUI image, what are the relative (0-1000) pixel point coordinates for the element corresponding to the following instruction or description: *{instruction}*

CogAgent:

Task: Click on the element most relevant to the instruction *{instruction}*
History steps:
(platform: *{platform}*)
(Answer in Status-Action-Operation-Sensitive format.)

OS-Atlas:

In this UI screenshot, what is the position of the element corresponding to the command "*{instruction}* " (with box)?

Qwen2.5VL:

The user query: Please click the most suitable*{instruction}* element:

SeeClick:

In this UI screenshot, what is the position of the element corresponding to the command "*{instruction}* "(with point)?

UGround:

Your task is to help the user identify the precise coordinates (x, y) of a specific area/element/object on the screen based on a description.
- Your response should aim to point to the center or a representative point within the described area/element/object as accurately as possible.
- If the description is unclear or ambiguous, infer the most relevant area or element based on its likely context or purpose.
- Your answer should be a single string (x, y) corresponding to the point of the interest.
Description: *{instruction}*
Answer:

Table 13: The model evaluation prompts used on LLMs. Qwen2.5VL has long system message, which follows the message in Qwen2.5VL repository.

samples. We then randomly select 5,000 for training, with the remaining 1,247 and an additional 15,991 samples used for testing.

B.3 Finetuning

We follow the Lora fine-tuning parameters officially provided by Aria and perform Lora fine-tuning. Specifically, we trained Lora with r=8, alpha=32, dropout=0.05, and target_modules as "fc1", "fc2", "q_proj", "k_proj", "v_proj", "linear", "o_proj", "up_proj", "down_proj", "out_proj", "gate_proj", "lm_head". The learning rate was set to 5e-5, the batch size was set to 16, and a total of 2 epochs were trained.

To validate the reasonableness of selecting two epochs, we documented the accuracy and distance variations in the validation set during fine-tuning Aria-ui on the normal partition, as illustrated in Figure 8. The results indicate that two epochs essentially achieve the model's optimal performance, with minimal gains from further training, which could potentially lead to overfitting.

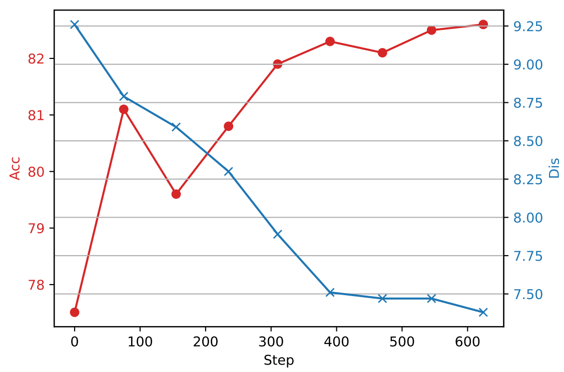


Figure 8: Variation of Average Accuracy and Average Distance on the Test Partition with Respect to Fine-Tuning Steps on the Normal Training Partition.

All fine-tuning and experiments were conducted for 200 GPU hours on the NVIDIA A800 cluster. Aria-ui has 25B parameters. We set seed as 42 for shuffle, using default config of transformers Trainer. We also plan to open-source our training scripts.

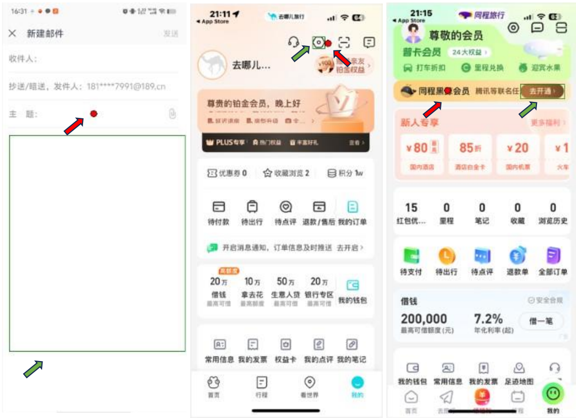
B.4 Error Analysis

The failure cases can be summarized into three categories, illustrated in Figure 9:

Incorrect GUI prediction. Models do not understand grounding instructions and predict wrong click positions.

Incorrect location with correct prediction. Despite the ability to understand grounding instructions, models generate predicted positions that fall near the bounding boxes.

Incorrect prediction affected by nearby elements. While models predict the target as a whole, instructions focus on only a specific part of it.



Type1. Incorrect GUI prediction. (Left)
Grounding Instruction: 点击输入邮件内容。
English version: Click to enter the email content.
Ground truth bounding box: [0.045, 0.324, 0.991, 0.848]
predicted location: [0.499, 0.272]

Type2. Incorrect location with correct prediction. (Middle)
Grounding Instruction: 点击屏幕顶部工具栏的六边形图标，可以访问设置。
English version: Tap the hexagonal icon in the toolbar at the top of the screen to access Settings.
Ground truth bounding box: [0.639, 0.080, 0.703, 0.108]
predicted location: [0.720, 0.095]

Type3. Incorrect prediction affected by nearby elements. (Right)
Grounding Instruction: 点击这个棕色按钮，去开通同程黑鲸会员服务。
English version: Click the brown button to open the service named 同程黑鲸会员.
Ground truth bounding box: [0.700, 0.196, 0.926, 0.231]
predicted location: [0.317, 0.211]

Figure 9: Three examples of error cases.