MAS-Bench: A Unified Benchmark for Shortcut-Augmented Hybrid Mobile GUI Agents

Pengxiang Zhao[1*], Guangyi Liu[1*], YaoZhen Liang[1*], Weiqing He[1], Zhengxi Lu[1], WenHao Wang[1], Yuehao Huang[1], Yuxiang Chai[2], Zhaolu Kang[3], Yaxuan Guo[2], Hao Wang[2], Kexin Zhang[1†], Liang Liu[2‡], Yong Liu[1†]

[1]Zhejiang University [2]vivo AI Lab [3]Peking University

Abstract

Shortcuts such as APIs and deep-links have emerged as efficient complements to flexible GUI operations, fostering a promising hybrid paradigm for MLLM-based mobile automation. However, systematic evaluation of GUI–shortcut hybrid agents remains largely underexplored. To bridge this gap, we introduce **MAS-Bench**, a benchmark that pioneers the evaluation of GUI-shortcut hybrid agents with a specific focus on the mobile domain. Beyond merely using predefined shortcuts, MAS-Bench assesses an agent's capability to *autonomously generate* shortcuts by discovering and creating reusable, low-cost workflows. It features 139 complex tasks across 11 real-world applications, a knowledge base of 88 predefined shortcuts (APIs, deep-links, RPA scripts), and 9 evaluation metrics. Experiments demonstrate that hybrid agents achieve up to 68.3% success rate and 39% greater execution efficiency than GUI-only counterparts. Furthermore, our evaluation framework effectively reveals the quality gap between predefined and agent-generated shortcuts, validating its capability to assess shortcut generation methods. MAS-Bench addresses the lack of systematic benchmarks for GUI-shortcut hybrid mobile agents, providing a foundational platform for future advancements in creating more efficient and robust intelligent agents. Project page: https://pengxiang-zhao.github.io/MAS-Bench.

1 Introduction

The rise of Large Language Models (LLMs) (OpenAI, 2024; Huang et al., 2025b; Jiang et al., 2025a; Li et al., 2026) is driving the development of Graphical User Interface (GUI) Agents, enabling them to operate diverse digital platforms such as computers, web browsers, and smartphones (Xu et al., 2024; Liu et al., 2025b,a). Early research on mobile agents primarily focused on replicating human-like flexibility through GUI-only interaction (Wang et al., 2025a; Lu et al., 2026; Xiao et al., 2026; Lu et al., 2025). This approach grants agents the generality to operate on any application, but it often overlooks the significant efficiency advantages offered by more direct methods.

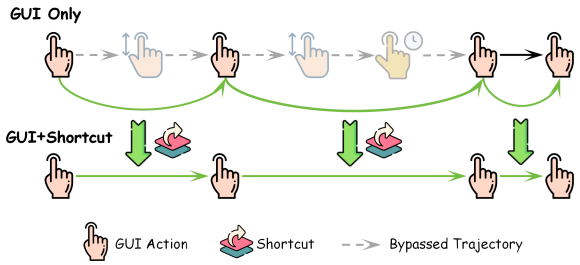

Figure 1: **Workflow of GUI Only vs. GUI-Shortcut Hybrid Agent.** Shortcuts improve agent execution efficiency by bypassing GUI operations.

As GUI agents are increasingly applied to complex tasks, enhancing their efficiency has become a central research focus. In this context, hybrid paradigms have emerged as a promising solution, demonstrating effectiveness across a wide range of platforms (Zhang et al., 2025a; Guo et al., 2026). This approach combines the speed and reliability of "shortcuts" such as Application Programming Interface (API) calls, deep links, and Robotic Process Automation (RPA) scripts (Agostinelli et al., 2022), with the flexibility of GUI operations. As Fig. 1 illustrates, a hybrid agent bypasses multi-step GUI operations by invoking a single shortcut, drastically reducing operational complexity and time.

However, despite these advancements, a framework for systematically evaluating and benchmarking hybrid agents is still lacking (Zhang et al., 2025a; Wang et al., 2025b; Huang et al., 2025a). This gap leaves the full potential of GUI-Shortcut agents far from being thoroughly explored and as-

*Equal contribution.
†Corresponding author.
‡Project Lead.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 6960–6985
July 2-7, 2026 ©2026 Association for Computational Linguistics

sessed. Mobile platforms provide a controlled and reproducible setting, making them a natural starting point for GUI-shortcut evaluation.

Therefore, we present **MAS-Bench**, **the first benchmark explicitly designed to evaluate GUI-shortcut hybrid mobile agents.** The tasks within MAS-Bench are solvable entirely through GUI interactions, but shortcuts can substantially improve execution efficiency. MAS-Bench evaluates whether agents can reliably choose between GUI actions and shortcuts under dynamic mobile environments. Our experiments show that MAS-GLM-4.5V achieves the best success rate of 68.3%. In a matched comparison, MAS-MobileAgent improves over its GUI-only counterpart from 35.2% to 63.3% SR, yielding a 79.8% relative gain and reducing the successful-task step ratio by 38.9%.

Furthermore, **we introduce a novel framework to evaluate agents' capacity to generate new shortcuts from interaction.** This framework integrates agent-generated shortcuts into a standard baseline agent and measures subsequent task performance. Our findings reveal a performance gap: while our predefined shortcuts prove highly reliable (100% success rate), agent-generated shortcuts lag in robustness and efficiency. Dynamic shortcuts demonstrate significant potential. Despite their task completion rate of 38% (baseline 43%), they offer the highest efficiency, highlighting them as a promising direction for future research into robust shortcut generation.

Our contributions are threefold:

- We introduce **MAS-Bench**, the first benchmark for systematically evaluating GUI-shortcut hybrid mobile agents. It comprises 139 complex tasks spanning 11 real-world applications, supported by a knowledge base of 88 predefined shortcuts and 9 distinct evaluation metrics.
- We establish extensive baselines across agentic workflows, general-purpose models, and specialized GUI models, demonstrating that GUI-shortcut hybrid operation substantially improves success rate and efficiency while exposing shortcut misuse.
- We propose the first framework to evaluate an agent's ability to generate shortcuts from interaction trajectories. Using predefined structural shortcuts as a reference upper-bound baseline, our experiments show that current generated shortcuts still lag behind predefined ones in efficiency and robustness, highlighting a key direction for future work.

2 Related Work

Mobile Task Automation. Mobile task automation evolves from methods based on predefined scripts to more intelligent and adaptive agents driven by LLMs (Li et al., 2025; Wang et al., 2025c). Traditional approaches, such as API calls, deep links, and Robotic Process Automation (RPA) scripts, offer direct execution paths but suffer from significant limitations, including invalidation from app updates, and a sensitivity to UI changes that hinders their ability to adapt (Kennedy and Everett, 2011; Xiao et al., 2025).

The rapid development of LLM-based GUI agents significantly overcomes traditional automation challenges by simulating human interaction through visual or multimodal inputs (Cheng et al., 2024). However, complete reliance on LLMs for fine-grained GUI interaction presents new problems: step-by-step action for routine tasks leads to inefficiency and increased costs; in complex operations, LLM hallucinations or misinterpretations can cause cumulative errors, affecting task success rates and reliability (Wen et al., 2024). Zhang et al. (Zhang et al., 2025a) provide a detailed analysis of API and GUI agents; the former gain attention for their stability and programming integration capabilities, while the latter possesses strong adaptability. These considerations regarding the efficiency and robustness of GUI agents lead to an exploration of how to combine LLM with efficient shortcut methods, thereby promoting the rise of research in GUI-Shortcut agents.

GUI-Shortcut Hybrid Operations. Researchers explore GUI-Shortcut methods to enhance agent capabilities and address the persistent challenges in task completion rate, costs, and operational efficiency for LLM-based GUI agents. For instance, UFO2 (Zhang et al., 2025b), a desktop AgentOS, demonstrates robust task execution through its multi-agent architecture and a unified GUI-API action layer that deeply integrates with operating system and application features. Inspired by such work, AppAgentX (Jiang et al., 2025b) improves efficiency by evolving high-level actions into shortcuts based on task execution history. MobileAgent-E (Wang et al., 2025d) uses a self-evolving module to learn and store general-purpose *Tips* and

Benchmark	# Inst.	DS	DL	EC	SG
AndroidArena (Xing et al., 2024)	221	✗	✗	✗	✗
AndroidWorld (Rawles et al., 2024)	116	✗	✓	✗	✗
LlamaTouch (Zhang et al., 2024)	495	✗	✓	✗	✗
AndroidLab (Xu et al., 2024)	138	✗	✗	✓	✗
SPA-Bench (Chen et al., 2024)	340	✗	✓	✓	✗
MAS-Bench (Ours)	139	✓	✓	✓	✓

Table 1: **Comparison of MAS-Bench and other smartphone agent benchmarks with dynamic environments.** Column definitions: # Inst. (number of instructions), # DS (support diverse shortcuts), # DL (difficulty level), # EC (efficiency&cost metrics), # SG (evaluate shortcut generation).

reusable *Shortcuts* from past experiences, continuously improving performance and efficiency over time. While these efforts focused on the mobile domain are promising, they also highlight the urgent need for a systematic method to evaluate the effectiveness of these emerging hybrid operations on mobile devices.

Mobile GUI Agent Benchmark. As agents evolve from GUI-based interaction to hybrid GUI-Shortcut operations, they place new demands on benchmarking. Among existing benchmarks, MobileAgentBench (Wang et al., 2024b) focuses on the foundational GUI navigation and task completion capabilities of LLM-powered mobile agents. AndroidWorld and SPA-Bench (Rawles et al., 2024; Chen et al., 2024) benchmark mobile agents on diverse smartphone tasks, spanning dynamic environments, task difficulty. While these benchmarks advance research in various dimensions, there remains a need for a benchmark that can comprehensively evaluate the ability of agents to intelligently discover, decide upon, and execute diverse shortcuts within GUI interactions and also measure the overall task effectiveness of this GUI-Shortcut hybrid operation (Zhang et al., 2025b).

3 MAS-Bench

3.1 Online Evaluation Environment

MAS-Bench is built on a dynamic Android platform to evaluate mobile agent performance on complex tasks comprehensively. Unlike static benchmarks, which are often limited to fixed datasets, our platform enables the real-time assessment of agents' capacity to select and utilize shortcuts intelligently. To guarantee reproducibility, we employ a snapshot-based reset mechanism that restores the environment to a consistent initial state after each task, and utilize dedicated test accounts to isolate real-world side effects (detailed in Appendix A.2).

3.2 GUI-Shortcut Hybrid Action Space

To complete tasks, agents in MAS-Bench operate within a hybrid action space that combines conventional GUI interactions with diverse programmatic shortcuts. This space comprises two action types:

GUI Action. GUI actions simulate direct human interaction with an application's interface. This action space consists of basic operations such as click, swipe, and type, enabling the agent to navigate and interact with UI elements. Detailed definitions for each action are provided in Appendix A.3. GUI actions are essential for dynamic scenarios with unavailable predefined shortcuts.

Shortcut Action. Shortcut actions can trigger specific functions or navigate to designated pages, bypassing multi-step GUI actions. In MAS-Bench, shortcuts are categorized into two types: **Predefined shortcuts** and **Agent-generated shortcuts**.

As illustrated in Fig. 2, the predefined shortcuts in MAS-Bench primarily consist of APIs, deep-links, and RPA scripts.

- **API**: An intent-based programmatic shortcut that invokes application functionality without GUI interaction, as shown in Fig. 2(a).
- **Deep Link**: A specialized URI that targets a specific page or function within an application. As shown in Fig. 2(b), this enables direct navigation, bypassing multi-step GUI actions.
- **RPA Script**: An Automation script designed to handle a specific, highly repetitive subtask. Unlike a single action, an RPA script encapsulates a complex workflow of GUI actions, API calls, or deep-links, consolidating a multi-step process into a single, efficient shortcut action, as shown in Fig. 2(c).

The agent-generated shortcuts are created dynamically by the agent itself. They are typically formed by identifying and abstracting repetitive subtasks from the agent's execution history into new, executable routines. This capability allows the agent to learn from experience and progressively optimize its performance across future tasks by creating customized shortcuts.

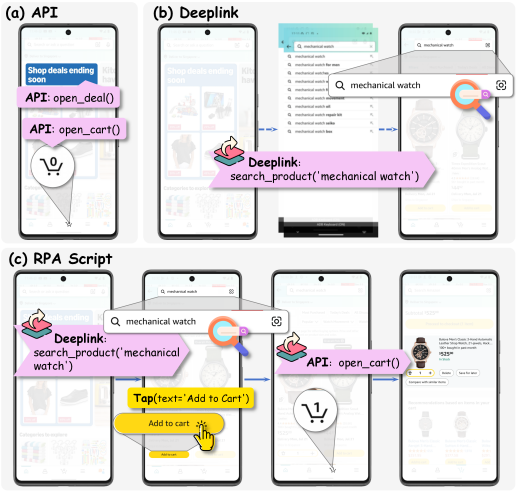

Figure 2: **Functional Comparison of APIs, Deep-Links, and RPA Scripts.** The figure uses the Amazon app as an example: (a) the *open_cart()* API directly opens the shopping cart; (b) the *search_product()* deep-link directly performs a product search; and (c) an RPA script combines APIs, deep links, and GUI operations to automate a complete workflow.

3.3 Shortcut Knowledge Base

Predefined Shortcuts Knowledge Base. To validate the feasibility of hybrid GUI-shortcut operations on mobile devices, we construct a predefined knowledge base of commonly used mobile shortcuts. Based on 11 popular apps, it includes 11 APIs, 70 Deep Links, and 7 custom RPA scripts, for a total of 88 predefined shortcuts. The distribution of shortcuts across applications varies based on app complexity and functionality, ranging from 2 shortcuts for Google Calendar to 21 for Fitbit (see Appendix Table 17). To ensure real-world relevance, APIs were identified from official documentation and static analysis of application packages. These APIs target intent-accessible endpoints and are executed through standard Android IPC/ADB commands without requiring root access, debug signatures, or emulator system modifications. At the same time, Deep Links were identified by analyzing each application's declared URI schemes and entry points. The custom RPA scripts were designed to encapsulate common, high-repetition sub-tasks, testing an agent's ability to leverage complex, pre-built automation routines. More details on the collection and design methodology are available in the Appendix A.4.

Agent-Generated Shortcuts. Besides predefined shortcuts, MAS-Bench supports evaluating agent-generated shortcuts to test an agent's learning and abstraction capabilities. In order to demonstrate the effectiveness of MAS-Bench in evaluating agents' capabilities of generating shortcuts, we have incorporated several contrasting shortcut generation methods for evaluation. We include Macro-level action trajectory replay (abbreviated as action replay) shortcuts that record and replay entire task or sub-task execution trajectories. We also incorporate dynamic shortcuts that require the agent to perform real-time grounding based on execution history, as well as shortcuts generated by the MobileAgent-E (Wang et al., 2025d). This diverse set of generation strategies allows for a comprehensive assessment of an agent's ability to autonomously create efficient workflows.

3.4 Tasks Design for GUI-Shortcut Hybrid Agent

MAS-Bench comprises 139 complex tasks derived from real-world scenarios across 11 Android applications, each featuring one or more invocable predefined shortcuts. These tasks reflect authentic user needs, with an average of 9.27 steps for single-app and 17.66 for cross-app workflows based on human operation. Tasks are categorized into three difficulty levels (detailed in Appendix A.5), with category-level examples summarized in Table 6.

Specifically, tasks in MAS-Bench maintain high real-world relevance. They are designed by considering user needs from daily life, such as searching for products in a shopping app or navigating to a location. The tasks typically involve multiple steps and sub-goals, and usable shortcuts exist within our established knowledge base for each task. However, our shortcut knowledge base for these complex tasks is intentionally incomplete. It offers only basic shortcuts, which are insufficient to complete an entire task independently. This design compels agents to synthesize solutions by combining shortcut invocations with multi-step GUI operations, thus providing a more realistic test of their planning and reasoning abilities.

To test the agent's learning and generation capabilities, we introduce scenarios with repetitive sub-tasks. In these scenarios, the knowledge base intentionally lacks predefined shortcuts for these recurrent operations. This design aims to test whether an agent, after completing a sub-task via GUI operations for the first time, can identify the repetitive pattern and then autonomously generate an efficient shortcut for subsequent use. Notably, some

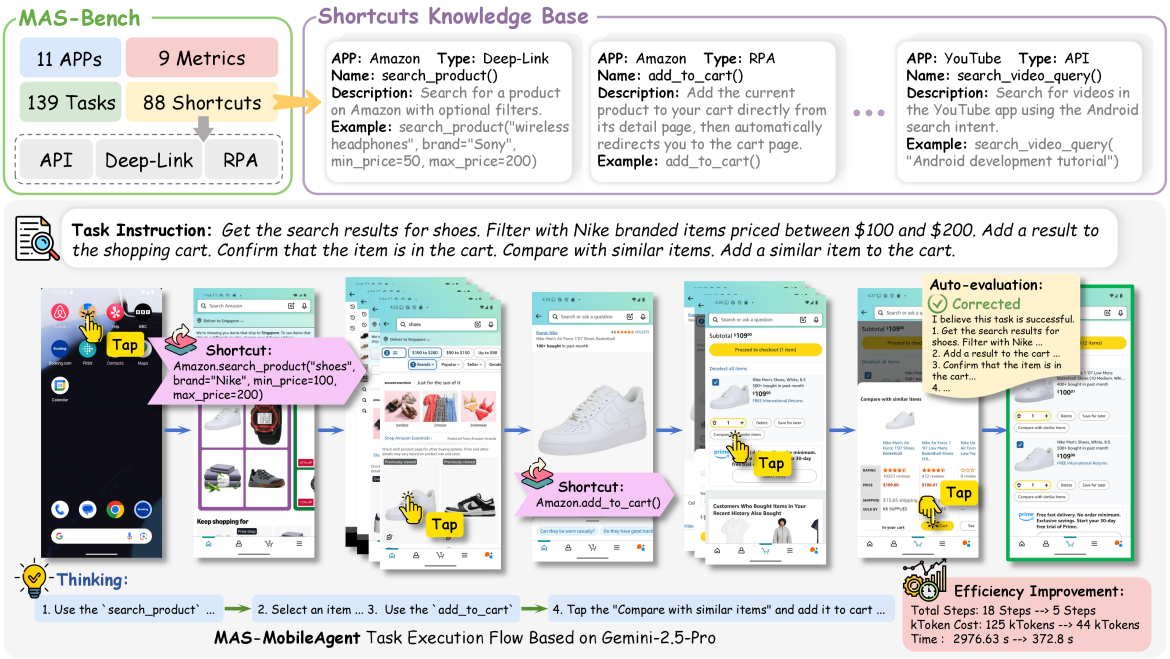

Figure 3: **The pipeline of MAS-Bench**. The GUI-Shortcut agent first filters products using the *search_product* shortcut, selects an item via GUI operations, and then adds it to the cart using the *add_to_cart* shortcut. The entire process is monitored by an automated evaluation module, which outputs metrics such as success rate and efficiency.

tasks in MAS-Bench are designed as incremental variations, where a task builds upon the previous one by adding additional requirements. This incremental design serves two purposes: it creates realistic scenarios where agents must handle evolving user needs, and it provides an ideal testbed for evaluating shortcut generation by presenting repeated sub-task patterns that agents should learn to abstract into reusable shortcuts.

3.5 Systematic Hybrid Agent Evaluation

GUI-Shortcut Hybrid Operation Evaluation. MAS-Bench is designed to evaluate an agent's ability to autonomously discover and utilize shortcuts to improve task success, efficiency, and cost-effectiveness. This evaluation mandates that an agent possess sophisticated recognition and decision-making capabilities. Agents must not only identify available shortcuts based on the current task and environment but also critically assess their applicability and efficiency. Furthermore, the agent must choose between GUI operations and shortcuts to reduce steps and time while avoiding irrelevant shortcut calls.

Shortcut Generation Evaluation. We expect GUI agents not only to rely on predefined shortcuts within the knowledge base but also to possess the capability for shortcut generation. The quality of agent-generated shortcuts serves as a direct indicator of their generative capability. However, a systematic framework to develop and evaluate the effectiveness of these generation methods is currently lacking. Therefore, MAS-Bench encourages and evaluates the agent's ability to generate new shortcuts autonomously. As shown in Fig. 4, we design the following evaluation method. First, a self-generation shortcut agent explores the MAS-Bench environment and creates its shortcut knowledge base. To ensure fairness and eliminate interference from the intrinsic abilities of the agent under test, we use a unified baseline GUI agent as the evaluated agent and import these shortcut knowledge bases into the baseline agent for task execution. The baseline agent's performance on the tasks directly reflects the quality of the imported knowledge base, thus enabling an evaluation of each agent's shortcut generation capabilities.

3.6 Evaluation Metrics

We employ 9 comprehensive metrics to evaluate agent performance across three dimensions: (1) *Success*: Success Rate (SR) measures task completion; (2) *Efficiency*: Mean Steps (MS), Mean Step Ratio (MSR), Mean Step Ratio on Successful tasks (MSRS), and Mean Execution Time (MET) assess operational efficiency; (3) *Cost and Resource Uti-*

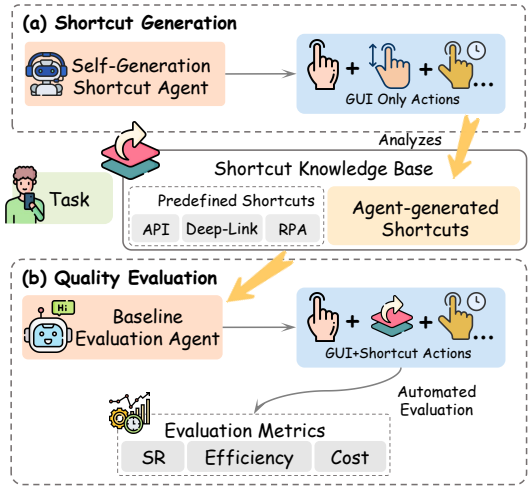

Figure 4: **Evaluation Workflow for Agents' Shortcut Generation Capability.** The process consists of two stages: (a) Shortcut Generation Stage, where the agent creates its shortcut knowledge base; and (b) Quality Evaluation Stage. The agent-generated shortcut knowledge base is imported into a baseline agent for performance testing. Its quality is then measured by comparing this performance against the GUI-only baseline agent and the baseline agent with predefined shortcuts.

lization: Mean Token Cost (MToC), Mean Shortcut Call Count (MSC), Shortcut Success Rate (SSR), and Shortcut-to-GUI Ratio (S2GR) evaluate resource consumption and operational strategy. Table 8 in Appendix A.6 provides detailed definitions, units, and improvement directions for all metrics.

3.7 Evaluation Methodology

We introduce **MAS-Bench-Eval** as a scalable and reproducible evaluation protocol for MAS-Bench. Illustrated in Fig. 7, this automated pipeline adopts a two-stage *Describe-and-Judge* framework:

Stage 1: Action Semantics Extraction. An efficient MLLM processes the agent's trajectory step-by-step, generating textual captions that describe the executed actions and corresponding UI state transitions. This stage transforms the raw visual stream into a structured semantic history.

Stage 2: Success Determination. A highly capable MLLM acts as the judge. It reasons over the task instruction, the semantic history derived from Stage 1, and the terminal screenshot to issue a binary success verdict.

This decoupled architecture effectively optimizes the trade-off between inference efficiency and evaluation capabilities. To validate its reliability, we compare MAS-Bench-Eval against 278 human-annotated trajectories and obtain high agreement with human judgments, with an F1 score of 96.3% and precision of 98.1%. We provide the full analysis in Appendix D.

4 MAS: Shortcut-Augmented Hybrid Mobile Agents

To evaluate the efficacy of the GUI-Shortcut hybrid operation, we introduce MAS-MobileAgent, which serves as a reference for GUI-shortcut Hybrid Agents. Built upon MobileAgent-V2 (Wang et al., 2024a), which relies on visual perception for UI understanding, MAS-MobileAgent augments its decision-making process by retrieving relevant shortcuts from the knowledge base and integrating them with visual inputs.

To further assess the generalization of our shortcut injection mechanism, we introduce MAS-T3A. Derived from T3A (Rawles et al., 2024), this agent operates on a distinct modality, utilizing structured UI trees rather than visual screenshots. By extending the hybrid operation to this text-based framework, we aim to verify its transferability to non-visual perceptual foundations.

Finally, to validate the effectiveness of hybrid operation within the *Agent-as-a-Model* paradigm, we apply the same shortcut injection strategy to multiple model families, including Qwen3-VL (Bai et al., 2025a), GLM-4.5V (Hong et al., 2025), ScaleCUA (Liu et al., 2025c), and MAI-UI (Zhou et al., 2025). Relevant shortcuts are incorporated directly into the system prompt without modifying the model weights. Collectively, these diverse implementations enable a rigorous and comprehensive assessment of our framework's universality across mainstream agent architectures.

5 Experiments

We conduct the experiments in the online, dynamic environment of MAS-Bench and evaluate the performance of GUI-shortcut hybrid mobile GUI agents. We also employ an evaluation framework to measure the efficiency of agent-generated and predefined shortcuts.

5.1 Experiment Setup

We evaluate GUI agents on the 139 tasks in MAS-Bench, including three categories: 1) GUI-only agents, covering Agentic Workflows (e.g., T3A, M3A (Rawles et al., 2024), and MobileAgentV2 (Wang et al., 2024a)) and Agent-as-a-Model methods, including General-Purpose Mod-

Agent	Input		SR↑	Efficiency			Cost		S2GR↑
	SS	VH		MS↓	MSRS↓	MET↓	MToC↓	MSC↑	
Human	✓		-	*12.11*	*1.000*	-	-	-	-
Agentic Workflow (Gemini-2.5-Pro)									
M3A (Rawles et al., 2024)	✓	✓	0.503	16.655	1.131	266.505	200.509	0	0
MobileAgent-E (Wang et al., 2025d)	✓		0.259	**4.808**	0.857	462.636	**87.819**	1.011	0.114
T3A (Rawles et al., 2024)		✓	0.453	15.755	1.067	178.005	440.919	0	0
+ MAS-T3A (Ours)		✓	0.554$_{\uparrow 22.3\%}$	12.768$_{\uparrow 19.0\%}$	0.823$_{\uparrow 22.9\%}$	**148.505**$_{\uparrow 16.6\%}$	341.402$_{\uparrow 23\%}$	1.438	0.140
MobileAgentV2 (Wang et al., 2024a)	✓		0.352	19.252	1.122	1364.979	156.441	0	0
+ MAS-MobileAgent (Ours)	✓		**0.633**$_{\uparrow 79.8\%}$	12.928$_{\uparrow 32.8\%}$	**0.686**$_{\uparrow 38.9\%}$	951.918$_{\uparrow 30.3\%}$	130.980$_{\uparrow 16\%}$	**1.948**	**0.341**
General-Purpose Models									
Qwen2.5-VL-3B (Bai et al., 2025b)	✓		0.036	21.626	1.341	**120.975**	-	0	0
Qwen2.5-VL-7B (Bai et al., 2025b)	✓		0.022	21.835	1.498	168.832	-	0	0
Qwen3-VL-4B (Bai et al., 2025a)	✓		0.228	17.547	1.273	179.208	-	0	0
+ MAS-Qwen3-VL-4B (Ours)	✓		0.237$_{\uparrow 3.9\%}$	20.403$_{-16.3\%}$	0.971$_{\uparrow 23.7\%}$	188.078$_{-4.9\%}$	-	2.461	0.170
Qwen3-VL-8B (Bai et al., 2025a)	✓		0.259	16.576	1.183	179.377	-	0	0
+ MAS-Qwen3-VL-8B (Ours)	✓		0.425$_{\uparrow 64.1\%}$	14.921$_{\uparrow 10.0\%}$	0.868$_{\uparrow 26.6\%}$	155.977$_{\uparrow 13.0\%}$	-	1.309	0.110
Qwen3-VL-32B (Bai et al., 2025a)	✓		0.338	17.834	1.207	394.102	-	0	0
+ MAS-Qwen3-VL-32B (Ours)	✓		0.446$_{\uparrow 32.0\%}$	15.913$_{\uparrow 10.8\%}$	0.848$_{\uparrow 29.7\%}$	358.282$_{\uparrow 9.1\%}$	-	**3.216**	**0.276**
Qwen3-VL-235B (Bai et al., 2025a)	✓		0.417	16.604	1.111	185.201	-	0	0
+ MAS-Qwen3-VL-235B (Ours)	✓		0.525$_{\uparrow 25.9\%}$	14.424$_{\uparrow 13.1\%}$	**0.785**$_{\uparrow 29.3\%}$	152.652$_{\uparrow 17.6\%}$	-	2.784	0.233
GLM-4.5V (Hong et al., 2025)	✓		0.526	17.237	1.194	281.928	-	0	0
+ MAS-GLM-4.5V (Ours)	✓		**0.683**$_{\uparrow 29.8\%}$	**14.050**$_{\uparrow 18.5\%}$	0.900$_{\uparrow 24.6\%}$	238.579$_{\uparrow 15.4\%}$	-	1.051	0.097
Specialized GUI Models									
UI-TARS-1.5-7B (Seed, 2025)	✓		0.287	19.209	1.191	188.143	-	0	0
GUI-Owl-7B (Ye et al., 2025)	✓		0.295	**15.568**	1.239	168.148	-	0	0
ScaleCUA-7B (Liu et al., 2025c)	✓		0.108	18.194	1.202	**123.018**	-	0	0
+ MAS-ScaleCUA-7B (Ours)	✓		0.115$_{\uparrow 6.5\%}$	19.446$_{-6.9\%}$	1.358$_{-13.0\%}$	141.535$_{-15.1\%}$	-	0.007	0.000
ScaleCUA-32B (Liu et al., 2025c)	✓		0.231	19.209	1.286	141.654	-	0	0
+ MAS-ScaleCUA-32B (Ours)	✓		0.216$_{-6.5\%}$	18.935$_{\uparrow 1.4\%}$	1.260$_{\uparrow 2.0\%}$	140.126$_{\uparrow 1.1\%}$	-	0.000	0.000
MAI-UI-8B (Zhou et al., 2025)	✓		0.489	19.237	1.247	180.678	-	0	0
+ MAS-MAI-UI-8B (Ours)	✓		**0.583**$_{\uparrow 19.2\%}$	18.065$_{\uparrow 6.1\%}$	**1.179**$_{\uparrow 5.5\%}$	169.463$_{\uparrow 6.2\%}$	-	**0.273**	**0.018**

Table 2: **Overall performance comparison on MAS-Bench with predefined shortcuts knowledge base (139 tasks).** Results are weighted averages based on task distribution (92 single-app, 47 cross-app tasks). Bold and underlined values denote the best and second-best results within each block, respectively. Methods marked with "**+**" are shortcut-augmented variants of their corresponding baselines. SS: Screenshot; VH: View Hierarchy; MS: Mean Steps; MSRS: Mean Step Ratio on Successful tasks; MET: Mean Execution Time (seconds); MToC: Mean Token Cost (in thousands); MSC: Mean Shortcut Call count; S2GR: Shortcut-to-GUI action Ratio. Detailed results in Appendix B.1.

els (e.g., Qwen3-VL (Bai et al., 2025a) and GLM-4.5V (Hong et al., 2025)) and Specialized GUI Models (e.g., UI-TARS-1.5 (Seed, 2025), GUI-Owl (Ye et al., 2025), ScaleCUA (Liu et al., 2025c), and MAI-UI (Zhou et al., 2025)); 2) self-generated shortcut frameworks (MobileAgent-E (Wang et al., 2025d)); 3) shortcut-augmented Hybrid Agents (GUI-shortcut agents), including MAS-T3A, MAS-MobileAgent, and MAS variants of Qwen3-VL, GLM-4.5V, ScaleCUA, and MAI-UI. For agentic workflows, we use Gemini-2.5-Pro as the base model. To examine the impact of predefined shortcuts across models with different capabilities, we also evaluate Gemini-2.0-Flash. All evaluations follow the metrics defined in Sec. 3.6 and Appendix A.

5.2 Evaluation Protocols

Predefined Shortcut Evaluation. To evaluate GUI-shortcut hybrid operation, we augment each compatible GUI agent with the predefined shortcut knowledge base and compare it with its GUI-only counterpart under the same task set, environment, and evaluation metrics.

Shortcut Invocation Robustness. To evaluate whether agents reliably select shortcuts, we annotate ground-truth shortcuts for each task and compare them with those invoked by agents. We report Mean Task Shortcut Recall (MTSR), Mean Shortcut Selection Precision (MSSP), MF1, and redundancy rate. To simulate long-tail cases where

no suitable shortcut, we construct an interference setting on 23 cross-app tasks by removing ground-truth shortcuts and exposing task-irrelevant distractors. We report the changes in SR and MS, together with Mean Irrelevant Shortcut Calls (MISC). See Appendix B.4 for details.

Shortcut Generation Evaluation. To assess the quality of shortcut knowledge bases generated by different strategies, we conduct a two-stage evaluation on a randomly selected test subset comprising 50% of the tasks in MAS-Bench. In the **Shortcut Generation Stage**, we construct several types of agent-generated shortcuts to form knowledge bases using two methods. One leverages MobileAgent-E to generate shortcuts ($S_{\text{MobileAgent-E}}$). The other, based on execution trajectories of M3A, generates three types of shortcuts: task-level action replays ($S_{\text{Replay-Task}}$), subtask-level action replays ($S_{\text{Replay-Subtask}}$), and dynamic shortcuts (S_{Dynamic}) that require the model to perform grounding. Details of the shortcut knowledge base are provided in Sec. 3.3. In the **Quality Evaluation Stage**, each knowledge base is mapped to a standardized action space and integrated into the T3A baseline agent. This method treats the baseline agent's performance as a reflection of the imported knowledge base's quality, enabling an unbiased, executor-controlled evaluation of the generation strategies.

5.3 Effectiveness of GUI-Shortcut Operation

Based on our experimental results in Table 2, we identify four key findings. Additional single-app and cross-app breakdowns for representative agents are provided in Appendix B.1.

Finding 1: Integrating a predefined shortcut knowledge base significantly enhances agent performance, validating the effectiveness of the hybrid GUI-shortcut operation. Experiments show that using the predefined knowledge base of 88 shortcuts significantly improves task success rate and efficiency across multiple agent families. Compared to MobileAgentV2, MAS-MobileAgent improves the success rate (SR) from 35.2% to 63.3%, an 80% improvement, while reducing the token cost from 156.441k to 130.980k. The gain also holds for Agent-as-a-Model approaches: MAS-GLM-4.5V achieves the best overall SR of 68.3%, and MAS-Qwen3-VL-8B improves SR from 25.9% to 42.5%. Furthermore, the Mean Step Ratio on Successful tasks (MSRS) serves as a robust efficiency metric, particularly for mitigating the impact of premature termination in long-horizon tasks. MAS-MobileAgent achieves an MSRS of 0.686, which is significantly lower than the baseline version's 1.122, indicating that its execution path is closer to the optimal solution. This also highlights the efficiency gap between current GUI-only agents and the human pure-GUI step baseline: strong GUI-only baselines remain above MSRS $= 1.0$, while hybrid agents can reduce MSRS below 1.0 by bypassing GUI paths with shortcuts. The SR and MS results for Different-level Tasks can be found in the Appendix B.5.

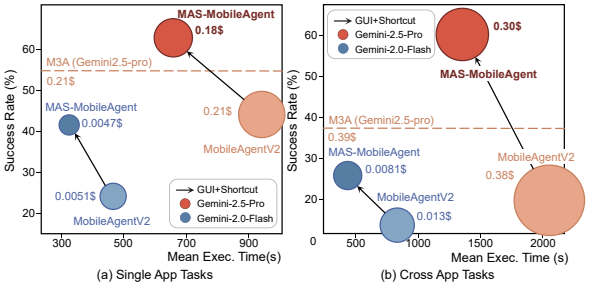

Figure 5: **Performance comparison of MAS-MobileAgent with and without shortcuts.** The base models are Gemini-2.5-Pro and Gemini-2.0-Flash. Data points show the relationship between SR and MET for single-app and cross-app tasks, with circle size representing mean cost. Results demonstrate that shortcuts benefit both models, with more significant improvements for the weaker Gemini-2.0-Flash.

Finding 2: The effectiveness of predefined shortcuts is framework-agnostic, benefiting agents with different input modalities. Experiments show that both agentic workflows based on structured UI trees (MAS-T3A) and raw screenshots (MAS-MobileAgent), as well as Agent-as-a-Model variants (e.g., MAS-Qwen3-VL and MAS-GLM-4.5V), benefit from the same predefined shortcut knowledge base. This independence stems from the fact that shortcut execution is decoupled from the agent's perception modality: shortcuts directly send commands to the operating system or application, reducing the need to parse the UI layout in real time. The results further show that General-Purpose Models adapt better to the hybrid action space, whereas specialized GUI models such as ScaleCUA do not automatically benefit without reliable shortcut selection.

Finding 3: Shortcut gains depend on base model capability and shortcut-use reliability. The Qwen3-VL series shows that shortcut gains are not monotonic with model scale: Qwen3-VL-

8B achieves the largest relative SR gain (+64.1%), while Qwen3-VL-4B suffers an efficiency drop (+16.3% MS). This indicates that hybrid operation requires sufficient reasoning ability to select and invoke shortcuts reliably. At the same time, shortcuts can substantially help weaker but capable agents bypass failure-prone GUI steps; for example, augmenting Gemini-2.0-Flash boosts its cross-app SR from 0% to 23.4% (Appendix B.6).

Finding 4: MAS-Bench exposes and penalizes shortcut misuse. Shortcut selection quality directly affects performance, as detailed in Appendix B.4. GLM-4.5V maintains low redundancy (4.8%) and reduces MS by 18.5%, whereas Qwen3-VL-4B has higher redundancy (33.5%) and increases MS by 16.3% (Table 11). In the no-relevant-shortcut interference setting, Qwen3-VL-4B suffers a 66.7% relative SR drop under irrelevant shortcuts, while GLM-4.5V drops only 7.7% (Table 12). These results show that MAS-Bench can evaluate and quantify agent robustness under noisy shortcut pools, not only performance when relevant shortcuts are available.

5.4 Results of Shortcut Generation

Our experiments include a baseline T3A framework without shortcuts and five distinct shortcut variants: Predefined ($S_{\text{Predefined}}$), Replay-Task ($S_{\text{Replay-Task}}$), Replay-Subtask ($S_{\text{Replay-Subtask}}$), Dynamic ($S_{\text{Dynamic}}$), and MobileAgent-E ($S_{\text{MobileAgent-E}}$). The predefined shortcut knowledge base serves as a reference upper bound for high-quality structural shortcuts, rather than a target that current automatic methods are expected to surpass. This setup allows us to compare shortcut generation paradigms under the same executor and quantify how close generated shortcuts are to reliable human-designed shortcuts.

In our experiments, Predefined shortcuts achieve the best performance, improving the success rate (SR) by 9% over the baseline while maintaining a perfect shortcut success rate (SSR) of 100%. Notably, agents using Predefined shortcuts reduce average execution steps by 25% and decrease total execution time by approximately 16%. Despite having the second-highest shortcut call count per task (1.45), Predefined shortcuts maintain excellent robustness, executing successfully across diverse task configurations. The increased success rate and reduced execution time validate that well-designed shortcuts will enhance GUI task efficiency.

Shortcut	SR↑	SSR↑	MSRS↓	MSC↑	MET↓
Human	-	-	*1.00*	-	-
Baseline	0.43	-	0.96	-	188.93
$S_{\text{Predefined}}$	**0.52**	**1.00**	**0.71**	1.45	**152.15**
$S_{\text{Replay-Task}}$	0.34	0.10	0.91	**3.04**	244.61
$S_{\text{Replay-Subtask}}$	0.43	0.73	1.13	1.22	236.67
S_{Dynamic}	0.38	0.75	0.82	0.91	216.24
$S_{\text{MobileAgent-E}}$	0.49	0.71	1.00	1.01	224.87

Table 3: **The results of different shortcut generation methods.** Column definitions: # SR (success rate), # MSRS (Mean Step Ratio on Successful tasks), # MSC (Mean Shortcut Call Count), # SSR (Shortcut Success Rate), # MET (Mean Execution Time).

In contrast, Replay-Task shortcuts exhibit poor robustness with only a 10% success rate, indicating high susceptibility to environmental variations. This fragility negatively impacts task completion rates and execution efficiency, underscoring that while effective shortcuts boost GUI performance, poorly designed ones are counterproductive.

Results reveal substantial room for improvement in model-generated shortcuts. Across variants, only Predefined and MobileAgent-E shortcuts improve model accuracy, with only Predefined shortcuts simultaneously enhancing execution speed and success rate. These results indicate current limitations in shortcut generation capabilities within our framework. Future work should focus on developing more efficient, robust shortcuts with higher utilization rates that effectively reduce execution steps.

6 Conclusion

In this paper, we introduce MAS-Bench, the first unified and comprehensive benchmark designed to evaluate the effectiveness of GUI-shortcut hybrid mobile agents. Furthermore, MAS-Bench provides a framework to assess the quality and validity of shortcuts that are autonomously generated by the agent. Our experiments demonstrate that the GUI-shortcut hybrid operation significantly enhances both the success rate and efficiency of task execution. Moreover, we also validate that our benchmark effectively measures the quality and robustness of the agent-generated shortcuts. We hope MAS-Bench will support future work on efficient GUI-shortcut hybrid mobile agents, including agent self-improvement through iterative shortcut accumulation, validation, and reuse.

7 Limitations

While our predefined shortcuts demonstrate robustness across application versions (Appendix A.4), agents should maintain the capability to fall back to GUI operations when shortcuts occasionally fail due to environmental variations. Additionally, our evaluation on standardized emulators ensures reproducibility and fair comparison; future work will validate these findings on real-world devices with diverse configurations. Another direction is to integrate dynamic shortcut discovery into MAS-Bench, e.g., by mining AndroidManifest files, static analysis results, official documentation, or web resources to construct structural shortcuts automatically. Finally, although MAS-Bench-Eval shows strong agreement with human judgments, future versions can combine rule-based state checks with LLM-as-a-Judge to further improve evaluation stability. This work highlights the importance of efficiency in mobile GUI agents and calls for further research into GUI-shortcut hybrid approaches.

References

Simone Agostinelli, Marco Lupia, Andrea Marrella, and Massimo Mecella. 2022. Reactive synthesis of software robots in rpa from user interface logs. *Computers in Industry*, 142:103721.

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

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025b. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, and 1 others. 2024. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*.

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*.

Weiyang Guo, Zesheng Shi, Liye Zhao, Jiayuan Ma, Zeen Zhu, Junxian He, Min Zhang, and Jing Li. 2026. E3-tir: Enhanced experience exploitation for tool-integrated reasoning. *Preprint*, arXiv:2604.09455.

Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, and 1 others. 2025. Glm-4.5 v and glm-4.1 v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006*.

Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. 2025a. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios. *arXiv preprint arXiv:2506.13977*.

Yuehao Huang, Liang Liu, Shuangming Lei, Yukai Ma, Hao Su, Jianbiao Mei, Pengxiang Zhao, Yaqing Gu, Yong Liu, and Jiajun Lv. 2025b. Cogddn: A cognitive demand-driven navigation with decision optimization and dual-process thinking. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 5237–5246.

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. 2025a. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*.

Wenjia Jiang, Yangyang Zhuang, Chenxi Song, Xu Yang, Joey Tianyi Zhou, and Chi Zhang. 2025b. Appagentx: Evolving gui agents as proficient smartphone users. *arXiv preprint arXiv:2503.02268*.

Courtney Kennedy and Stephen E Everett. 2011. Use of cognitive shortcuts in landline and cell phone surveys. *Public Opinion Quarterly*, 75(2):336–348.

Xiaoyuan Li, Keqin Bao, Yubo Ma, Moxin Li, Wenjie Wang, Rui Men, Yichang Zhang, Fuli Feng, Dayiheng Liu, and Junyang Lin. 2025. Mtr-bench: A comprehensive benchmark for multi-turn reasoning evaluation. *arXiv preprint arXiv:2505.17123*.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, Xiangwen Zhang, Guanhua Chen, Zheng Pan, Xin Li, and Yong Liu. 2026. No more stale feedback: Co-evolving critics for open-world agent learning. *arXiv preprint arXiv:2601.06794*.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Zhiming Chen, Yuxiang Chai, Shuai Ren, Hao Wang, Shibo He, and Wenchao Meng. 2025a. Learnact: Few-shot mobile gui agent with a unified demonstration benchmark. *arXiv preprint arXiv:2504.13805*.

Guangyi Liu, Pengxiang Zhao, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, and 1 others. 2025b. Llm-powered gui agents in phone automation: Surveying progress and prospects. *arXiv preprint arXiv:2504.19838*.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, and 1 others. 2025c. Scalecua: Scaling open-source computer use agents with cross-platform data. *arXiv preprint arXiv:2509.15221*.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, and 1 others. 2026. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 17608–17616.

Zhengxi Lu, Jiabo Ye, Fei Tang, Yongliang Shen, Haiyang Xu, Ziwei Zheng, Weiming Lu, Ming Yan, Fei Huang, Jun Xiao, and 1 others. 2025. Ui-s1: Advancing gui automation via semi-online reinforcement learning. *arXiv preprint arXiv:2509.11543*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

ByteDance Seed. 2025. Ui-tars-1.5. https://seed-tars.com/1.5.

Chengbing Wang, Yang Zhang, Wenjie Wang, Xiaoyan Zhao, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2025a. Think-while-generating: On-the-fly reasoning for personalized long-form generation. *arXiv preprint arXiv:2512.06690*.

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

Luyuan Wang, Yongyu Deng, Yiwei Zha, Guodong Mao, Qinmin Wang, Tianchen Min, Wei Chen, and Shoufa Chen. 2024b. Mobileagentbench: An efficient and user-friendly benchmark for mobile llm agents. *arXiv preprint arXiv:2406.08184*.

Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, Jian Du, Yaxin Du, Xianghe Pang, Keduan Huang, Yanfeng Wang, Qiang Yan, and Siheng Chen. 2025b. Mcp-flow: Facilitating llm agents to master real-world, diverse and scaling mcp tools. *Preprint*, arXiv:2510.24284.

Wenhao Wang, Zijie Yu, Rui Ye, Jianqing Zhang, Guangyi Liu, Liang Liu, Siheng Chen, and Yanfeng Wang. 2025c. Fedmabench: Benchmarking mobile gui agents on decentralized heterogeneous user data. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26398–26419.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025d. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *arXiv preprint arXiv:2501.11733*.

Hao Wen, Shizuo Tian, Borislav Pavlov, Wenjie Du, Yixuan Li, Ge Chang, Shanhui Zhao, Jiacheng Liu, Yunxin Liu, Ya-Qin Zhang, and 1 others. 2024. Autodroid-v2: Boosting slm-based gui agents via code generation. *arXiv preprint arXiv:2412.18116*.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, and 1 others. 2025. Ui-genie: A self-improving approach for iteratively boosting mllm-based mobile gui agents. *arXiv preprint arXiv:2505.21496*.

Han Xiao, Guozhi Wang, Hao Wang, Shilong Liu, Yuxiang Chai, Yue Pan, Yufeng Zhou, Xiaoxin Chen, Yafei Wen, and Hongsheng Li. 2026. Ui-mem: Self-evolving experience memory for online reinforcement learning in mobile gui agents. *arXiv preprint arXiv:2602.05832*.

Mingzhe Xing, Rongkai Zhang, Hui Xue, Qi Chen, Fan Yang, and Zhen Xiao. 2024. Understanding the weakness of large language model agents within a complex android environment. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6061–6072.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2024. Androidlab: Training and systematic benchmarking of android autonomous agents. *arXiv preprint arXiv:2410.24024*.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, and 1 others. 2025. Mobile-agent-v3: Fundamental agents for gui automation. *arXiv preprint arXiv:2508.15144*.

Chaoyun Zhang, Shilin He, Liqun Li, Si Qin, Yu Kang, Qingwei Lin, and Dongmei Zhang. 2025a. Api agents vs. gui agents: Divergence and convergence. *arXiv preprint arXiv:2503.11069*.

Chaoyun Zhang, He Huang, Chiming Ni, Jian Mu, Si Qin, Shilin He, Lu Wang, Fangkai Yang, Pu Zhao, Chao Du, and 1 others. 2025b. Ufo2: The desktop agentos. *arXiv preprint arXiv:2504.14603*.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. 2024. Llamatouch: A faithful and scalable testbed for mobile ui automation task evaluation. *arXiv preprint arXiv:2404.16054*.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and 1 others. 2025. Mai-ui technical report: Real-world centric foundation gui agents. *arXiv preprint arXiv:2512.22047*.

Appendix

A MAS-Bench Environment

A.1 Observation Space

MAS-Bench evaluates mobile agents in a highly standardized and controllable environment using an Android Virtual Device (AVD). Specifically, the environment is based on a Pixel 7 emulator running Android 15 (API Level 35). Its screen resolution is set to 1080 x 2400 pixels, and the screen density is 420 dpi. The emulator has 24GB of RAM and 24GB of storage. The environment support parallel execution of multi-threaded evaluations. All model deployments and evaluations are conducted on 4 NVIDIA L40S GPUs 48GB.

A.2 Environment Control and Reproducibility

MAS-Bench's dynamic evaluation approach moves beyond simply verifying task outcomes, focusing on the agent's decision-making process in live scenarios. To ensure real-world relevance, the benchmark incorporates a suite of widely used Android applications covering diverse daily-life scenarios. Ensuring consistent and reproducible evaluation environments is critical for fair agent comparison, particularly when working with dynamic online applications. MAS-Bench addresses this challenge through a snapshot-based rapid recovery mechanism combined with carefully controlled test accounts.

Snapshot-Based Rapid Recovery. We leverage Android emulator snapshot functionality to guarantee absolute environment consistency across evaluations. Prior to experimentation, we construct a pre-configured Android Virtual Device image, termed **MAS-Bench-AVD**, which includes all 11 benchmark applications with necessary permissions granted and dedicated test accounts logged in. This clean baseline state is saved as a snapshot. We maintain a single system-level snapshot rather than task- or app-specific snapshots, so this design introduces no additional per-task storage overhead beyond the base AVD image.

Online Content Control. For applications involving dynamic online content (e.g., Amazon, YouTube, Gmail), we employ pre-registered anonymous dedicated test accounts to maintain a relatively stable recommendation baseline. These accounts are isolated from real user data. For tasks involving email communication, we utilize anonymous test accounts configured with automatic reply functionality, ensuring that all email-based tasks interact only within our controlled test environment. This design prevents interference with real-world users while maintaining realistic task scenarios.

Parallel Execution and Scalability. To support large-scale evaluation, MAS-Bench implements ADB port-based isolation for managing multiple emulator instances concurrently. Each emulator instance operates independently with its own snapshot rollback mechanism, enabling parallel task execution without cross-contamination. This isolation strategy is equally effective for shortcut actions, ensuring that they can be executed concurrently without interference.

Real-World Side Effect Mitigation. All benchmark tasks execute within our controlled test environment using dedicated test accounts. Actions such as adding an item to cart, or sending emails affect only these isolated test accounts and never impact real users or production systems. This design ensures ethical evaluation practices while maintaining task authenticity.

A.3 Details of Action Space

Action	Definition
Tap (x, y)	Tap at coordinates (x, y).
Type (text)	Enter text content into an input field.
Swipe (x_1, y_1, x_2, y_2)	Swipe from (x_1, y_1) to (x_2, y_2).
Home	Go to the home screen.
Back	Go back to the previous app screen.
Stop	Complete the current task.

Table 4: Examples of the basic action space in MAS-Bench.

The MAS-Bench environment facilitates agent-device interaction through a low-level interface based on the Android Debug Bridge (ADB). Crucially, the specific action space available to an agent during evaluation ultimately depends on its architecture and capabilities. Table 4 shows examples of the basic action set.

A.4 Details of Predefined Shortcuts Knowledge Base

Details of Predefined Shortcuts. Our shortcut knowledge base was created to provide agents with an authentic yet challenging environment. We collect and design shortcuts for 11 selected applications, ensuring their relevance to real-world usage patterns. Importantly, we focus on collecting shortcuts that represent **commonly used functionalities** of these applications, which tend to be stable across app versions.

- **API Collection**: The APIs in our knowledge base are intent-accessible endpoints rather than private or non-exported APIs. We identify documented endpoints from official Android documentation and use static analysis of application packages to locate additional endpoints exposed through standard Android IPC. We retain only APIs that can be invoked by standard ADB commands, such as `adb shell am start`, with Intent actions, data URIs, and extras. They do not require root access, debug signatures, hidden permissions, or emulator system modifications.
- **Deep-Links Collection**: We collect deep-links by analyzing each application's `AndroidManifest.xml` file. This file typically declares the URL schemes and paths the application can handle, providing direct mappings to specific pages or functions.
- **RPA Script Design**: RPA scripts are designed for common, highly repetitive sub-tasks (e.g., adding an item with specific options to a shopping cart). These scripts are implemented using UI tree-based element identification, which provides better robustness against UI layout changes compared to coordinate-based approaches. The scripts can be executed as a single action, serving as a high-level shortcut. This design enables us to test an agent's ability to utilize complex, pre-built automation routines.

Stability of Shortcuts. The stability of shortcuts across application versions is a critical consideration for real-world deployment. Our collection strategy deliberately focuses on core, commonly used functionalities that applications typically maintain across updates. **API and Deep-Link shortcuts** exhibit high stability, as they interface with core application functions that rarely change across versions. These shortcuts are based on documented or intent-accessible endpoints and declared URI schemes, which developers typically maintain for backward compatibility. **RPA scripts** in our knowledge base are implemented using UI tree-based element identification rather than fixed coordinates. This approach leverages the structured view hierarchy, which remains more stable across application updates compared to pixel-based coordinate systems, resulting in enhanced robustness against UI layout changes.

To validate the robustness of our shortcut knowledge base, we have tested all shortcuts across at least five different versions of each application, spanning updates released over six months. Our validation confirms that all shortcuts in our knowledge base remain functional across these versions without modification, demonstrating their practical viability for real-world agent deployment.

Details of Agent-Generated Shortcuts. We incorporate several methods to create agent-generated shortcuts. The primary purpose for including this variety of shortcut types is to validate the effectiveness of the MAS-Bench framework itself in evaluating an agent's shortcut generation capabilities. By creating test cases with diverse characteristics, we can verify our benchmark's ability to assess different shortcut generation strategies.

- **Macro-level action trajectory replay (action replay) Shortcuts**: This category of shortcuts is constructed by recording and replaying an agent's historical action trajectories. They execute a fixed sequence of low-level actions, such as clicks and swipes at specific coordinates, to automate a previously completed workflow. This method is implemented at two distinct granularities:
  - **Task-Level Replay**: Captures and replays the entire action sequence of a full task, from start to finish. This results in a single, high-level macro that is highly specific to one complete workflow.
  - **Subtask-Level Replay**: Identifies and abstracts recurrent, multi-step operations within a larger task. This approach generates shorter, more modular shortcuts that automate common sub-workflows.
- **Dynamic Shortcuts**: Unlike static macro-replays, these shortcuts are adaptive and ro-

bust to UI changes. Instead of replaying hardcoded coordinates, they utilize real-time grounding to semantically identify and interact with target UI elements based on the current screen context. This allows them to function correctly even if the position or layout of UI elements changes.

Ultimately, this diverse suite of shortcut types validates that MAS-Bench provides a comprehensive framework for evaluating agents' shortcut generation capabilities.

A.5 Details of Tasks in MAS-Bench

We have developed a specialized benchmark suite to validate the feasibility of GUI-shortcut hybrid actions for mobile agents and establish a framework for the comprehensive evaluation of an agent's ability to utilize and generate shortcuts.

This suite comprises 139 tasks strategically designed across 11 popular real-world applications. The tasks are categorized into 92 single-app tasks and 47 cross-app tasks, with their difficulty distribution illustrated in Table 7. We have also curated a corresponding knowledge base of predefined shortcuts to support these tasks. Table 17 presents a breakdown of the selected applications and the number of single-app tasks and shortcuts associated with each.

Table 6 summarizes the task coverage and representative examples. Single-app tasks cover shopping, local services, information browsing, media, productivity, health, and navigation workflows. Approximately 37% of single-app tasks are designed as incremental variations, where later tasks extend earlier ones with additional constraints or sub-goals. Cross-app tasks emphasize inter-app coordination and are mostly constructed by composing foundational single-app operations into information transfer, sharing, planning, and multi-app workflows.

Human Baseline Collection Methodology. To establish optimal step counts for efficiency metrics (MSR and MSRS), we conducted a systematic human expert annotation process. Three experienced mobile device users were recruited to independently complete each of the 139 tasks in MAS-Bench. Each expert was instructed to complete tasks as efficiently as possible using **only pure GUI operations**, specifically standard interactions such as taps, swipes, and text input, without access to any programmatic shortcuts. For each task, we recorded the number of steps taken by each expert and selected the *minimum* step count across the three demonstrations as the human baseline. This methodology ensures that our baseline represents the shortest achievable pure-GUI path under realistic conditions, providing a fair and reproducible reference for agent efficiency evaluation. The resulting human baselines average 9.27 steps for single-app tasks and 17.66 steps for cross-app tasks, reflecting the inherent complexity difference between these task categories.

A.6 Evaluation Metrics

To comprehensively evaluate agent performance on MAS-Bench, we employ a multi-dimensional evaluation framework that assesses success, efficiency, and cost. Table 8 provides a complete summary of all metrics. Our evaluation framework is designed to encourage agents to efficiently complete tasks by intelligently utilizing existing and self-generated shortcuts, thereby achieving lower costs and greater efficiency.

Success Rate (SR). We use SR to measure whether an agent completes a task. Success is typically defined as the agent reaching the final goal state of the task while satisfying all of its requirements.

Efficiency. We use the following metrics to evaluate the impact of shortcut actions on agents' efficiency, measured by the time and operations required to complete a task:

- **Mean Steps (MS)**: The average number of steps an agent takes to complete a task.
- **Mean Step Ratio (MSR)**: The ratio of agent steps to human steps (pure GUI operations). Lower values indicate better efficiency, with values below 1.0 representing fewer steps than the human.
- **Mean Step Ratio on Successful tasks (MSRS)**: The same ratio as MSR, computed only over successfully completed tasks.
- **Mean Execution Time (MET)**: The average time an agent takes to complete a task.

Cost and Resource Utilization. Cost metrics evaluate the computational resources and overhead an agent consumes to complete a task. A high-performing agent should minimize the overall cost

Agent	Base Model	SR↑	Efficiency			Cost		S2GR↑
			MS↓	MSRS↓	MET↓	MToC↓	MSC↑	
Single-app Tasks (92 Tasks)								
Human	-	-	*9.272*	*1.000*	-	-	-	-
MobileAgentV2	Gemini-2.0-Flash	0.250	12.881	1.280	472.426	40.406	0	0
MAS-MobileAgent	Gemini-2.0-Flash	0.402	9.087$_{\uparrow 29\%}$	0.719$_{\uparrow 44\%}$	**335.181**	**39.597**	**3.565**	**0.505**
MobileAgentV2	Gemini-2.5-Pro	0.446	12.424	1.058	1013.386	120.212	0	0
MAS-MobileAgent	Gemini-2.5-Pro	**0.641**	**9.109**$_{\uparrow 27\%}$	**0.613**$_{\uparrow 42\%}$	682.547	99.780	1.348	0.345
Cross-app Tasks (47 Tasks)								
Human	-	-	*17.660*	*1.000*	-	-	-	-
MobileAgentV2	Gemini-2.0-Flash	0	31.553	-	952.660	106.501	0	0
MAS-MobileAgent	Gemini-2.0-Flash	0.234	24.132$_{\uparrow 24\%}$	0.938	**403.798**	**71.350**	**5.362**	**0.709**
MobileAgentV2	Gemini-2.5-Pro	0.170	32.617	1.247	2053.133	227.128	0	0
MAS-MobileAgent	Gemini-2.5-Pro	**0.617**	**20.404**$_{\uparrow 37\%}$	**0.829**$_{\uparrow 34\%}$	1441.586	189.836	3.128	0.320

Table 5: Evaluation results of different base models. MS: Mean Steps; MSRS: Mean Step Ratio on Successful tasks; MET: Mean Execution Time (seconds); MToC: Mean Token Cost (in thousands); MSC: Mean Shortcut Call count; S2GR: Shortcut-to-GUI action Ratio.

Task Type	Category	# Tasks	Representative Apps	Example Task Pattern
Single-app	Shopping and local services	28	Amazon, Booking.com, Yelp	Search, filter, and verify products, hotels, attractions, or restaurants within one app.
Single-app	Information and media browsing	26	BBC News, Chrome, YouTube	Navigate news sections, search the web or videos, and save or play selected content.
Single-app	Productivity, health, and navigation	38	Contacts, Fitbit, Gmail, Calendar, Maps	Create contacts, draft emails, add calendar events, log health records, or obtain map directions.
Cross-app	Information management	16	YouTube, Calendar, Booking.com, Contacts, BBC News	Retrieve information in one app and record, schedule, or reuse it in another app.
Cross-app	Web shopping and travel planning	11	Amazon, Booking.com, Gmail, Calendar, Chrome	Search products, hotels, flights, or rentals and transfer the result to email, calendar, or another app.
Cross-app	Multi-app coordination	11	Maps, Yelp, Gmail, Calendar, BBC News	Coordinate several apps to complete chained goals such as finding places and sharing details.
Cross-app	Social sharing	5	Maps, BBC News, Gmail	Find content such as locations, photos, or articles and share the result through email.
Cross-app	Media and entertainment	4	YouTube, Chrome, Fitbit, Calendar	Search or consume media content and connect it with logging or scheduling actions.

Table 6: **Task categories and representative examples in MAS-Bench.** MAS-Bench contains 92 single-app tasks and 47 cross-app tasks across 11 real-world Android applications.

Difficulty Level	**Proportion (%)**	**Human Steps**
Level 1	27.3	6.2
Level 2	47.5	12.6
Level 3	25.2	17.6

Table 7: Distribution of task difficulty levels in MAS-Bench. The table shows the proportion of tasks at each level and the average number of steps required for a human to complete them.

by intelligently utilizing or autonomously generating shortcuts. We use the following metrics to evaluate the cost of an agent:

- **Mean Token Cost (MToC)**: The average number of tokens (in thousands) consumed by the LLM during task execution.
- **Mean Shortcut Call Count (MSC)**: The average number of shortcuts the agent calls during the task execution.
- **Shortcut Success Rate (SSR)**: The ratio of successfully executed shortcut calls to the total number of shortcut calls made by the agent.
- **Shortcut to GUI Action Ratio (S2GR)**: This metric reflects an agent's operational strategy preference by calculating the ratio of shortcut operations to GUI operations. A higher ratio indicates that the agent relies more heavily

on efficient shortcuts rather than manual GUI actions.

A.7 Task Difficulty Classification

MAS-Bench tasks are categorized into three difficulty levels to enable systematic evaluation across varying complexity. The classification criteria differ between single-app and cross-app tasks, reflecting their distinct operational characteristics.

Single-App Tasks. For the 92 single-app tasks, difficulty is primarily determined by **operational complexity** measured through the number of required steps and the sophistication of sub-goals:

- **Level 1 (Easy)**: Tasks requiring ≤ 7 steps. These involve basic, direct operations with a single primary goal, such as simple searches or navigation (e.g., searching for a product, opening a specific page).
- **Level 2 (Medium)**: Tasks requiring 8-15 steps. These involve multi-step workflows with intermediate complexity, including filtering operations, state verification, or coordination of multiple sub-goals (e.g., searching with filters and adding items to cart).
- **Level 3 (Hard)**: Tasks requiring ≥ 15 steps. These involve complex workflows with multiple interdependent sub-goals, advanced filtering, cross-page navigation, and explicit state verification (e.g., multi-criteria product comparison with cart management and confirmation).

Cross-App Tasks. For the 47 cross-app tasks, difficulty is determined by both the **number of applications involved** and the **complexity of inter-app coordination**:

- **Level 1 (Easy)**: Tasks involving 2 applications with straightforward information transfer, typically requiring ≤ 12 steps. These tasks involve basic data lookup in one app and simple recording or sharing in another.
- **Level 2 (Medium)**: Tasks involving 2-3 applications with moderate coordination complexity, typically requiring 13-20 steps. These tasks involve filtering, processing, or transforming information across applications.
- **Level 3 (Hard)**: Tasks involving 3-4 applications with complex multi-app workflows, typically requiring ≥ 20 steps. These tasks demand sophisticated inter-app dependencies, multi-stage data processing, and coordination across diverse application domains.

B Details of Predefined Shortcut Evaluation

B.1 Detailed Results on Single-app and Cross-app Tasks

This section provides detailed performance for representative agents on single-app and cross-app tasks in MAS-Bench. Table 9 shows the results for 92 single-app tasks, while Table 10 presents the results for 47 cross-app tasks. These detailed results complement the overall weighted average results shown in Table 2 in the main text.

B.1.1 Detailed Results of Single-app and Cross-app Tasks

The performance comparison between single-app and cross-app tasks reveals several critical insights into agent capabilities and the effectiveness of shortcut augmentation. Table 9 and Table 10 show the detailed results for representative agents.

Task Complexity and Performance Degradation. Cross-app tasks demonstrate significantly higher complexity than single-app tasks, as evidenced by the substantial performance degradation across many agents. For instance, the baseline T3A achieves an SR of 51.1% on single-app tasks but drops to 34.0% on cross-app tasks, representing a 33% relative decrease. Similarly, MobileAgentV2 shows an even more pronounced decline from 44.6% to 17.0% (62% relative decrease). This trend is also observed in several Agent-as-a-Model baselines, with GUI-Owl-7B declining from 40.2% to 8.5%. The human baseline steps also increase from 9.272 to 17.660, confirming the inherent complexity difference between task types.

Amplified Benefits of Shortcuts on Cross-app Tasks. Predefined shortcuts demonstrate disproportionately larger benefits on cross-app tasks compared to single-app tasks. MAS-MobileAgent improves MobileAgentV2's SR by 44% on single-app tasks but achieves a remarkable 263% improvement on cross-app tasks (17.0% $\rightarrow$ 61.7%). Similarly, MAS-T3A shows a 13% improvement on single-app tasks versus 50% on cross-app tasks.

Metric	Symbol	Definition	Unit	Dir.
		Success		
Success Rate	SR	Percentage of successfully completed tasks	%	↑
		Efficiency		
Mean Steps	MS	Average number of steps (GUI actions + shortcuts) per task	steps	↓
Mean Step Ratio	MSR	Ratio of agent steps to human baseline steps (pure GUI)	ratio	↓
Mean Step Ratio on Successful	MSRS	MSR calculated only on successfully completed tasks	ratio	↓
Mean Execution Time	MET	Average wall-clock time to complete a task	seconds	↓
		Cost and Resource Utilization		
Mean Token Cost	MToC	Average LLM tokens consumed per task	thousands	↓
Mean Shortcut Call Count	MSC	Average number of shortcut invocations per task	count	↑
Shortcut Success Rate	SSR	Ratio of successful shortcut calls to total calls	%	↑
Shortcut-to-GUI Ratio	S2GR	Ratio of shortcut actions to GUI actions	ratio	↑

Table 8: **Summary of evaluation metrics in MAS-Bench.** Dir.: Direction of improvement (↑ higher is better, ↓ lower is better). MSR and MSRS use human expert demonstrations (pure GUI operations) as the baseline. Values below 1.0 indicate agent performance exceeds human efficiency through shortcut utilization.

Agent	Input		SR↑	Efficiency			Cost		S2GR↑
	SS	VH		MS↓	MSRS↓	MET↓	MToC↓	MSC↑	
Human	✓		-	*9.272*	*1.000*	-	-	-	-
Agentic Workflow (Gemini-2.5-Pro)									
M3A (Rawles et al., 2024)	✓	✓	0.565	11.772	1.064	192.775	155.281	0	0
MobileAgent-E (Wang et al., 2025d)	✓		0.359	**4.080**	0.818	459.574	**88.772**	0.378	0.081
T3A (Rawles et al., 2024)		✓	0.511	11.750	1.056	137.641	346.382	0	0
+ MAS-T3A (Ours)		✓	0.576$_{\uparrow 13\%}$	10.595$_{\uparrow 10\%}$	0.915$_{\uparrow 13\%}$	**129.279**$_{\uparrow 6\%}$	291.391$_{\uparrow 16\%}$	1.043	0.117
MobileAgentV2 (Wang et al., 2024a)	✓		0.446	12.424	1.058	1013.386	120.212	0	0
+ MAS-MobileAgent (Ours)	✓		**0.641**$_{\uparrow 44\%}$	9.109$_{\uparrow 27\%}$	**0.613**$_{\uparrow 42\%}$	682.547$_{\uparrow 33\%}$	99.780$_{\uparrow 17\%}$	**1.348**	**0.345**
General-Purpose Models									
Qwen2.5-VL-3B (Bai et al., 2025b)	✓		0.054	17.380	1.341	**99.564**	-	0	0
Qwen2.5-VL-7B (Bai et al., 2025b)	✓		0.022	17.315	1.625	167.892	-	0	0
Qwen3-VL-4B (Bai et al., 2025a)	✓		0.290	13.739	1.274	143.098	-	0	0
+ MAS-Qwen3-VL-4B (Ours)	✓		0.315$_{\uparrow 8.6\%}$	15.011$_{-9.3\%}$	0.888$_{\uparrow 30.3\%}$	147.230$_{-2.9\%}$	-	**2.620**	0.219
Qwen3-VL-8B (Bai et al., 2025a)	✓		0.326	12.761	1.179	148.021	-	0	0
+ MAS-Qwen3-VL-8B (Ours)	✓		0.522$_{\uparrow 60\%}$	11.065$_{\uparrow 13\%}$	**0.792**$_{\uparrow 33\%}$	124.370$_{\uparrow 16\%}$	-	1.239	0.129
Qwen3-VL-32B (Bai et al., 2025a)	✓		0.402	12.793	1.150	293.941	-	0	0
+ MAS-Qwen3-VL-32B (Ours)	✓		0.543$_{\uparrow 35.1\%}$	11.859$_{\uparrow 7.3\%}$	0.810$_{\uparrow 29.6\%}$	269.291$_{\uparrow 8.4\%}$	-	2.272	**0.271**
Qwen3-VL-235B (Bai et al., 2025a)	✓		0.478	11.989	1.113	166.083	-	0	0
+ MAS-Qwen3-VL-235B (Ours)	✓		0.598$_{\uparrow 25.1\%}$	10.924$_{\uparrow 8.9\%}$	0.849$_{\uparrow 23.7\%}$	134.274$_{\uparrow 19.2\%}$	-	1.989	0.219
GLM-4.5V (Hong et al., 2025)	✓		0.533	13.739	1.209	214.544	-	0	0
+ MAS-GLM-4.5V (Ours)	✓		**0.739**$_{\uparrow 38.6\%}$	**10.337**$_{\uparrow 24.8\%}$	0.823$_{\uparrow 31.9\%}$	192.489$_{\uparrow 10.3\%}$	-	0.957	0.110
Specialized GUI Models									
UI-TARS-1.5-7B (Seed, 2025)	✓		0.380	14.543	1.203	147.966	-	0	0
GUI-Owl-7B (Ye et al., 2025)	✓		0.402	**11.554**	1.279	132.698	-	0	0
ScaleCUA-7B (Liu et al., 2025c)	✓		0.163	13.554	1.202	**98.037**	-	0	0
+ MAS-ScaleCUA-7B (Ours)	✓		0.174$_{\uparrow 6.7\%}$	14.685$_{-8.3\%}$	1.358$_{-13.0\%}$	111.684$_{-13.9\%}$	-	0.011	0.001
ScaleCUA-32B (Liu et al., 2025c)	✓		0.283	14.065	1.273	113.134	-	0	0
+ MAS-ScaleCUA-32B (Ours)	✓		0.272$_{-3.9\%}$	14.230$_{-1.2\%}$	1.245$_{\uparrow 2.2\%}$	115.300$_{-1.9\%}$	-	0	0
MAI-UI-8B (Zhou et al., 2025)	✓		0.565	13.793	1.213	137.677	-	0	0
+ MAS-MAI-UI-8B (Ours)	✓		**0.652**$_{\uparrow 15.4\%}$	13.065$_{\uparrow 5.3\%}$	**1.130**$_{\uparrow 6.8\%}$	131.573$_{\uparrow 4.4\%}$	-	**0.315**	**0.024**

Table 9: **Detailed performance on single-app tasks (92 tasks).** Bold and underlined values denote the best and second-best results within each block, respectively. Methods marked with "**+**" are shortcut-augmented variants of their corresponding baselines. SS: Screenshot; VH: View Hierarchy; MS: Mean Steps; MSRS: Mean Step Ratio on Successful tasks; MET: Mean Execution Time (seconds); MToC: Mean Token Cost (in thousands); MSC: Mean Shortcut Call count; S2GR: Shortcut-to-GUI action Ratio.

Agent	Input		SR↑	Efficiency			Cost		S2GR↑
	SS	VH		MS↓	MSRS↓	MET↓	MToC↓	MSC↑	
Human	✓		-	*17.660*	*1.000*	-	-	-	-
Agentic Workflow (Gemini-2.5-Pro)									
M3A (Rawles et al., 2024)	✓	✓	0.383	26.213	1.262	410.828	289.040	0	0
MobileAgent-E (Wang et al., 2025d)	✓		0.064	**6.234**	0.934	468.630	**85.954**	2.250	0.177
T3A (Rawles et al., 2024)		✓	0.340	23.596	1.087	257.015	625.970	0	0
+ MAS-T3A (Ours)		✓	0.511$_{\uparrow 50\%}$	17.021$_{\uparrow 28\%}$	**0.643**$_{\uparrow 41\%}$	**186.139**$_{\uparrow 28\%}$	439.296$_{\uparrow 30\%}$	2.213	0.185
MobileAgentV2 (Wang et al., 2024a)	✓		0.170	32.617	1.247	2053.204	227.357	0	0
+ MAS-MobileAgent (Ours)	✓		**0.617**$_{\uparrow 263\%}$	20.404$_{\uparrow 37\%}$	0.829$_{\uparrow 34\%}$	1479.197$_{\uparrow 28\%}$	192.052$_{\uparrow 16\%}$	**3.122**	**0.333**
General-Purpose Models									
Qwen2.5-VL-3B (Bai et al., 2025b)	✓		0	29.936	-	**162.886**	-	0	0
Qwen2.5-VL-7B (Bai et al., 2025b)	✓		0.021	30.681	1.250	170.672	-	0	0
Qwen3-VL-4B (Bai et al., 2025a)	✓		0.106	25.000	1.272	249.891	-	0	0
+ MAS-Qwen3-VL-4B (Ours)	✓		0.085$_{-19.7\%}$	30.957$_{-23.8\%}$	1.133$_{\uparrow 10.9\%}$	268.035$_{-7.3\%}$	-	2.149	0.075
Qwen3-VL-8B (Bai et al., 2025a)	✓		0.128	24.043	1.192	240.755	-	0	0
+ MAS-Qwen3-VL-8B (Ours)	✓		0.234$_{\uparrow 83\%}$	22.468$_{\uparrow 7\%}$	1.016$_{\uparrow 15\%}$	217.846$_{\uparrow 9\%}$	-	1.447	0.073
Qwen3-VL-32B (Bai et al., 2025a)	✓		0.213	27.702	1.318	590.161	-	0	0
+ MAS-Qwen3-VL-32B (Ours)	✓		0.255$_{\uparrow 19.7\%}$	23.849$_{\uparrow 13.9\%}$	0.923$_{\uparrow 30.0\%}$	532.476$_{\uparrow 9.8\%}$	-	**5.064**	**0.287**
Qwen3-VL-235B (Bai et al., 2025a)	✓		0.298	25.638	1.106	222.624	-	0	0
+ MAS-Qwen3-VL-235B (Ours)	✓		0.383$_{\uparrow 28.5\%}$	**21.276**$_{\uparrow 17.0\%}$	**0.659**$_{\uparrow 40.4\%}$	188.626$_{\uparrow 15.3\%}$	-	4.340	0.261
GLM-4.5V (Hong et al., 2025)	✓		0.511	24.085	1.165	413.829	-	0	0
+ MAS-GLM-4.5V (Ours)	✓		**0.574**$_{\uparrow 12.3\%}$	21.319$_{\uparrow 11.5\%}$	1.051$_{\uparrow 9.8\%}$	328.798$_{\uparrow 20.5\%}$	-	1.234	0.071
Specialized GUI Models									
UI-TARS-1.5-7B (Seed, 2025)	✓		0.106	28.340	1.169	266.787	-	0	0
GUI-Owl-7B (Ye et al., 2025)	✓		0.085	**23.426**	**1.161**	237.539	-	0	0
ScaleCUA-7B (Liu et al., 2025c)	✓		0	27.277	-	**171.917**	-	0	0
+ MAS-ScaleCUA-7B (Ours)	✓		0	28.766$_{-5.5\%}$	-	199.968$_{-16.3\%}$	-	0	0
ScaleCUA-32B (Liu et al., 2025c)	✓		0.128	29.277	1.310	197.481	-	0	0
+ MAS-ScaleCUA-32B (Ours)	✓		0.106$_{-17.2\%}$	28.145$_{\uparrow 3.9\%}$	1.290$_{\uparrow 1.5\%}$	188.723$_{\uparrow 4.4\%}$	-	0	0
MAI-UI-8B (Zhou et al., 2025)	✓		0.340	29.894	1.313	264.849	-	0	0
+ MAS-MAI-UI-8B (Ours)	✓		**0.447**$_{\uparrow 31.5\%}$	27.851$_{\uparrow 6.8\%}$	1.274$_{\uparrow 3.0\%}$	243.632$_{\uparrow 8.0\%}$	-	**0.191**	**0.007**

Table 10: **Detailed performance on cross-app tasks (47 tasks).** Bold and underlined values denote the best and second-best results within each block, respectively. Methods marked with "**+**" are shortcut-augmented variants of their corresponding baselines. SS: Screenshot; VH: View Hierarchy; MS: Mean Steps; MSRS: Mean Step Ratio on Successful tasks; MET: Mean Execution Time (seconds); MToC: Mean Token Cost (in thousands); MSC: Mean Shortcut Call count; S2GR: Shortcut-to-GUI action Ratio.

This amplified effect stems from shortcuts' ability to bridge cross-application boundaries. Complex inter-app transitions that are particularly challenging for GUI-only agents become single atomic actions with appropriate shortcuts.

Efficiency Gains Across Task Types. The Mean Step Ratio on Successful tasks (MSRS) metric reveals consistent efficiency improvements from shortcut augmentation across both task types. For agentic workflows, MAS-MobileAgent achieves an MSRS of 0.613 on single-app tasks and 0.829 on cross-app tasks, substantially lower than the baseline MobileAgentV2's 1.058 and 1.247, respectively. This indicates that shortcut-augmented agents consistently execute paths closer to optimal solutions. Notably, MobileAgent-E, despite its low MS due to frequent premature termination from errors, maintains a competitive MSRS of 0.818 on single-app tasks, demonstrating that when it does succeed, its self-generated shortcuts enable efficient execution.

Agent-as-a-Model Performance Characteristics. The expanded Agent-as-a-Model results show a clear difference between general-purpose models and specialized GUI models. General-purpose models adapt more effectively to the hybrid action space: MAS-GLM-4.5V achieves the highest SR on both single-app (73.9%) and cross-app tasks (57.4%), while MAS-Qwen3-VL-32B and MAS-Qwen3-VL-235B obtain stable efficiency gains. In contrast, specialized GUI models do not automatically benefit from shortcuts. MAI-UI-8B improves

with MAS, but ScaleCUA variants rarely invoke shortcuts and show marginal or negative changes. These results suggest that shortcut augmentation requires not only GUI perception but also reliable shortcut selection and tool-use reasoning.

Cost and Efficiency. Token cost analysis reveals that shortcut augmentation provides substantial efficiency gains while marginally increasing cost. On single-app tasks, MAS-MobileAgent reduces token cost by 17% compared to MobileAgentV2, while on cross-app tasks, the reduction is 16%. Mean execution time (MET) improvements are particularly pronounced on cross-app tasks: MAS-T3A reduces MET by 28% compared to T3A (257s $\rightarrow$ 186s), demonstrating that shortcuts not only improve success rates but also significantly accelerate task completion through reduced step counts and more efficient execution paths.

B.2 Details of MAS-MobileAgent and MAS-T3A

Compared to MobileAgentV2, MAS-MobileAgent acquires the standard screenshot and UI elements and retrieves relevant and potentially applicable shortcuts from the shortcut knowledge base, based on the current task goal and interface context. This retrieved shortcut information is injected as an additional context into the prompt provided to the base model, along with screen information and the task goal.

MAS-T3A operates on a similar principle, but its core distinction lies in its input modality: it relies on the structured UI tree, whereas MAS-MobileAgent processes visual screenshots. By evaluating both agents, we aim to demonstrate that the effectiveness of predefined shortcuts is framework-agnostic and can benefit agents with fundamentally different perceptual modalities.

B.3 Shortcut Retrieval, Injection, and Action Space Mapping

Retrieval and Injection Mechanism. To enable efficient shortcut utilization, we implement a keyword-based retrieval mechanism. For each task instruction, we extract the mentioned application names and filter the shortcut knowledge base to retrieve only applicable shortcuts for those applications. The retrieved shortcuts are then injected into the agent's prompt as structured text descriptions with function signatures, parameters, and natural language descriptions (e.g., `Amazon.search_product(query: str) - Searches for products matching the query in Amazon`). This application-scoped filtering reduces context length while ensuring all relevant shortcuts remain available. This controlled retrieval setting isolates shortcut selection and execution from open-domain retrieval failures. Future extensions can replace this module with unfiltered shortcut search or RAG-based shortcut retrieval, while our interference study (Appendix B.4) provides an initial stress test under noisy shortcut pools.

Action Space Standardization. To ensure fair evaluation across different shortcut types, we adopt the T3A action space as our standardization target. All shortcuts, whether predefined (APIs, deep-links, RPA scripts) or agent-generated, are mapped to T3A-compatible action sequences during execution. This unified framework enables direct comparison between different shortcut generation strategies.

Coordinate-Free Dynamic Shortcuts vs. Action Replay. A critical design choice distinguishes agent-generated shortcuts. *Dynamic shortcuts* specify UI elements through semantic descriptions (e.g., "the search button") rather than hardcoded coordinates or indices, requiring the agent to perform real-time grounding during execution. This design offloads the *planning* burden to the shortcut itself while requiring only *grounding* at runtime, reducing strategic errors and enhancing robustness to UI changes.

In contrast, *action replay shortcuts* (task-level and subtask-level) record fixed sequences with specific coordinates and element indices. While simpler to generate, they are brittle to UI changes. As shown in Table 3, this design difference significantly impacts Shortcut Success Rate (SSR): predefined shortcuts achieve 100% SSR, dynamic shortcuts achieve 75% SSR, while replay-task shortcuts achieve only 10% SSR due to their sensitivity to UI variations. The subtask-level replay (SSR=73%) performs better than task-level replay by abstracting shorter, more reusable patterns, but still suffers from coordinate brittleness. These results validate that coordinate-free dynamic shortcuts better balance generation complexity with execution robustness, making them more suitable for real-world deployment where application UIs evolve over time.

B.4 Shortcut Invocation Robustness

To evaluate whether agents can use shortcuts reliably, we annotate the ground-truth shortcuts for

each task and compare them with the shortcuts selected by each agent. Table 11 reports Mean Task Shortcut Recall (MTSR), Mean Shortcut Selection Precision (MSSP), MF1, and redundancy rate. The results show that shortcut selection quality is closely tied to efficiency gains. For example, GLM-4.5V maintains high precision with a low redundancy rate (4.8%), leading to an 18.5% reduction in MS. In contrast, Qwen3-VL-4B has lower precision (0.439) and higher redundancy (33.5%), causing MS to increase by 16.3% despite a small SR gain.

We further test robustness in a no-relevant-shortcut setting. Specifically, we select 23 cross-app tasks with reasonable baseline success rates and remove all ground-truth shortcuts for each task while retaining task-irrelevant shortcuts as distractors. As shown in Table 12, incorrect shortcut invocation consistently hurts performance. Qwen3-VL-4B makes the most irrelevant shortcut calls (MISC=2.478) and suffers the largest SR drop (-66.67%), whereas GLM-4.5V makes fewer irrelevant calls (MISC=0.478) and remains close to its GUI-only baseline (-7.69% SR). These results show that MAS-Bench can expose and penalize shortcut misuse, making it suitable for evaluating agents under larger and noisier shortcut pools.

B.5 Success Rate on Different-Level Tasks

As shown in Table 13, an analysis of metrics such as MSRS reveals that the GUI-shortcut hybrid agent, by utilizing the predefined shortcuts knowledge base, demonstrates improvements in both success rate and efficiency across tasks of varying difficulty levels when compared to the GUI-only agent.

Similarly, Table 14 presents the performance comparison of Agent-as-a-Model approaches across different difficulty levels. The results show that MAS-Qwen3-VL-8B consistently outperforms its GUI-only counterpart across all difficulty levels, with particularly notable improvements in MSRS, demonstrating the effectiveness of shortcut augmentation for this category of agents.

B.6 Influence of Base Model Capability

To assess the impact of GUI-shortcut hybrid actions on base models with varying capabilities, we conducted experiments using Gemini-2.0-Flash and Gemini-2.5-Pro. As shown in Table 5, shortcut augmentation improves both models, but the gains appear in different forms. On single-app tasks, Gemini-2.0-Flash improves from 25.0% to 40.2% SR and reduces MS by 29%, while Gemini-2.5-Pro improves from 44.6% to 64.1% SR and reduces MS by 27%.

The role of shortcuts is clearer on cross-app tasks. With Gemini-2.0-Flash, the GUI-only agent fails all cross-app tasks (0% SR), whereas MAS-MobileAgent reaches 23.4% SR and reduces MET by 57.6%. With Gemini-2.5-Pro, MAS-MobileAgent improves SR from 17.0% to 61.7% and reduces MS by 37%. These results indicate that shortcuts can make weak base models solve tasks that are otherwise infeasible, while also improving the reliability and efficiency of stronger base models.

The Qwen3-VL series in Table 2 provides a more fine-grained view of this trend. MAS-Qwen3-VL-8B achieves the largest relative SR gain (+64.1%), suggesting that mid-scale models can benefit substantially once they have sufficient reasoning ability to use shortcuts. In contrast, MAS-Qwen3-VL-4B yields only a small SR gain (+3.9%) and increases MS by 16.3%, indicating that shortcut misuse can offset the efficiency benefit for under-capable models. Larger models, such as Qwen3-VL-32B and Qwen3-VL-235B, achieve more stable improvements (+32.0% and +25.9% SR), but their relative gains are smaller due to stronger GUI-only baselines. Overall, shortcut augmentation is most effective when the base model is capable enough to select shortcuts reliably but still benefits from bypassing long GUI interaction paths.

C Details of Shortcut Generation Evaluation

C.1 Shortcut Generation Method

To validate the effectiveness of the MAS-Bench framework in evaluating an agent's shortcut generation capabilities, we generated a diverse set of agent-generated shortcuts using various methods.

We employ M3A as our baseline agent to generate different types of shortcuts. Without altering its core operational logic, M3A autonomously explores the environment and generates task execution trajectories, utilizing Gemini-2.5-Pro as its base model. Upon completing each task, we construct a prompt containing the task instruction, the whole action history, and the reasoning for each step. This prompt is then provided to Gemini-2.5-Pro, generating the corresponding shortcuts.

Examples of the resulting shortcut types are

Agent	MTSR↑	MSSP↑	MF1↑	Redun.(%)↓	SR Impr.(%)↑	MS Red.(%)↑
MAS-MobileAgent	0.709	**0.721**	**0.715**	13.866	**+79.452**	**+32.847**
MAS-Qwen3-VL-32B	**0.721**	0.540	0.617	44.784	+31.803	+10.771
MAS-GLM-4.5V	0.555	0.681	0.607	**4.848**	+29.996	+18.489
MAS-Qwen3-VL-235B	0.674	0.550	0.605	31.156	+25.930	+13.128
MAS-Qwen3-VL-8B	0.496	0.601	0.539	23.404	+63.914	+9.985
MAS-T3A	0.457	0.632	0.530	22.500	+22.252	+18.963
MAS-Qwen3-VL-4B	0.422	0.439	0.428	33.498	+4.162	-16.277
MAS-MAI-UI-8B	0.218	0.719	0.335	22.807	+19.177	+6.096
MAS-ScaleCUA-32B	0.000	0.000	0.000	0.000	-6.383	+1.424
MAS-ScaleCUA-7B	0.000	0.000	0.000	0.000	+6.748	-6.882

Table 11: **Shortcut selection quality and performance changes on MAS-Bench.** Bold and underlined values denote the best and second-best results, respectively. MTSR: Mean Task Shortcut Recall; MSSP: Mean Shortcut Selection Precision; MF1: mean F1 score; Redun.: redundancy rate of shortcut calls, ranked among agents with nonzero shortcut use. SR Impr. and MS Red. denote relative success-rate improvement and mean-step reduction over the corresponding GUI-only baseline; negative MS Red. indicates increased mean steps.

Agent	BL SR↑	ΔSR↑	BL MS↓	ΔMS↓	MISC↓
GLM-4.5V	**0.565**	**-0.043 (-7.69%)**	21.739	**+0.087**	**0.478**
Qwen3-VL-235B	0.522	-0.043 (-8.33%)	20.870	+2.391	1.913
Qwen3-VL-32B	0.435	-0.043 (-10.00%)	**20.391**	+4.783	1.087
Qwen3-VL-8B	0.348	-0.087 (-25.00%)	21.000	+5.348	1.261
Qwen3-VL-4B	0.391	-0.261 (-66.67%)	22.261	+9.652	2.478

Table 12: **Resistance to task-irrelevant shortcut interference on 23 cross-app tasks.** Bold and underlined values denote the best and second-best results, respectively. BL denotes the GUI-only baseline. In the interference setting, all ground-truth shortcuts are removed and only task-irrelevant shortcuts are exposed. MISC denotes the mean number of irrelevant shortcut calls per task. For ΔSR, larger values indicate smaller degradation, with the relative drop in parentheses used to break ties.

shown in Fig. 6. As detailed in Table 15, this process resulted in 46 Subtask-level Macro-Replay, 45 Dynamic, 39 Task-level Macro-Replay, and 36 MobileAgent-E shortcuts.

C.2 Subset Task

To validate our framework for evaluating an agent's shortcut generation capability, we randomly selected a subset of 50% from the 139 tasks in MAS-Bench. This subset comprises a total of 69 tasks, including 46 single-app and 23 cross-app tasks.

D Reliability Analysis of Automated Evaluation

D.1 Evaluation Pipeline Architecture

Fig. 7 illustrates the two-stage *Describe-and-Judge* framework underlying MAS-Bench-Eval. This decoupled architecture optimizes the trade-off between computational efficiency and evaluation accuracy.

Stage 1: Action Semantics Extraction. An efficient MLLM processes the agent's execution trajectory step-by-step, analyzing each screenshot-action pair to generate concise textual captions. Each caption describes both the executed action (e.g., "invoked search_product shortcut") and the resulting UI state transition (e.g., "navigated to search results page"). This stage transforms potentially lengthy visual trajectories into structured semantic histories, enabling efficient downstream processing while preserving critical behavioral information.

Stage 2: Success Determination. A highly capable MLLM serves as the judge, receiving three inputs: the task instruction, the step-by-step action descriptions from Stage 1, and the final three screenshots. The judge performs compositional reasoning to verify whether all task requirements are satisfied, including sub-goal completion, final state correctness, and workflow integrity. It outputs a binary verdict with justification.

Agent	DL	SR	MS	MSRS
T3A	Level 1	71.05	7.53	0.94
	Level 2	36.36	17.62	1.08
	Level 3	45.71	21.17	1.08
M3A	Level 1	65.79	7.66	1.02
	Level 2	51.52	16.98	1.21
	Level 3	31.43	25.34	1.04
MobileAgentV2	Level 1	52.63	7.87	1.04
	Level 2	34.85	20.05	1.14
	Level 3	17.14	30.11	1.35
MobileAgentE	Level 1	44.74	3.66	0.83
	Level 2	20.00	5.15	0.97
	Level 3	17.14	5.43	0.60
MAS-T3A	Level 1	76.32	6.16	0.76
	Level 2	53.03	13.21	0.82
	Level 3	37.14	20.69	0.99
MAS-MobileAgent	Level 1	84.21	3.97	0.54
	Level 2	65.15	13.14	0.76
	Level 3	37.14	22.26	0.77

Table 13: **Performance comparison of Agentic Workflow agents (Gemini-2.5-Pro) on MAS-Bench across different difficulty levels.** Column definitions: # DL (difficulty level), # SR (success rate), # MS (Mean step), # MSRS (Mean Step Ratio on Successful tasks).

Agent	DL	SR	MS	MSRS
Qwen2.5-VL-3B	Level 1	10.53	11.92	1.40
	Level 2	1.52	22.68	1.09
	Level 3	0.00	30.17	-
Qwen2.5-VL-7B	Level 1	7.89	11.95	1.50
	Level 2	0.00	23.08	-
	Level 3	0.00	30.23	-
UI-TARS-1.5-7B	Level 1	55.26	8.82	1.17
	Level 2	22.73	20.00	1.31
	Level 3	11.43	29.00	1.10
GUI-Owl-7B	Level 1	55.26	9.08	1.38
	Level 2	21.21	15.68	1.02
	Level 3	17.14	22.40	1.07
Qwen3-VL-4B	Level 1	44.74	9.61	1.26
	Level 2	13.64	19.33	1.39
	Level 3	17.14	22.80	0.88
Qwen3-VL-8B	Level 1	50.00	8.00	1.14
	Level 2	16.67	18.11	1.13
	Level 3	17.14	23.00	1.01
MAS-Qwen3-VL-8B	Level 1	63.16	7.16	0.85
	Level 2	37.88	15.98	0.89
	Level 3	28.57	21.34	0.71

Table 14: **Performance comparison of Agent-as-a-Model approaches on MAS-Bench across different difficulty levels.** Column definitions: # DL (difficulty level), # SR (success rate), # MS (Mean step), # MSRS (Mean Step Ratio on Successful tasks).

This decoupled architecture provides three key advantages: (1) *Efficiency*: lightweight models handle repetitive trajectory processing, while capable models focus on critical judgment; (2) *Interpretability*: semantic histories enable transparent evaluation and failure analysis; (3) *Modularity*: individual components can be upgraded independently as stronger models emerge.

D.2 Validation Against Human Judgment

To validate the reliability of **MAS-Bench-Eval**, we conducted a comprehensive study comparing its automated verdicts against manual verification. We sampled 278 execution trajectories generated by MAS-MobileAgent on MAS-Bench, covering a diverse set of 184 single-app and 94 cross-app tasks. All trajectories were manually annotated to establish ground truth labels for task success. In our implementation of MAS-Bench-Eval, we instantiated the efficient VLM in **Stage 1 (Action Semantics Extraction)** with `Gemini-2.5-Flash` to generate step-by-step trajectory captions. For **Stage 2 (Success Determination)**, we employed the highly capable `Gemini-2.5-Pro` as the judge. This model reasons over the task instruction, the extracted semantic history, and the final three screenshots to issue a final success verdict.

Table 16 reports the quantitative alignment between MAS-Bench-Eval and human evaluation. The results demonstrate remarkable consistency, with F1 scores exceeding 95% across both scenarios (96.3% for single-app and 95.1% for cross-app). Specifically, the high precision (up to 98.1%) indicates a minimal false-positive rate, ensuring that the benchmark does not overestimate agent performance. Meanwhile, the strong recall (up to 98.0%) confirms the system's sensitivity in identifying true successes. These metrics verify that MAS-Bench-Eval serves as a reliable and scalable proxy for human assessment, enabling extensive experiments without the need for manual intervention.

E Bad Case Analysis

We further analyze the performance of the GUI-shortcut hybrid agent on failed tasks, with a particular focus on cases where the GUI-only agent succeeds, but the GUI-shortcut hybrid agent fails. Our analysis of these failure cases reveals three

Shortcut Type	Number of Shortcuts
$S_{\text{Replay-Task}}$	39
$S_{\text{Replay-Subtask}}$	46
S_{Dynamic}	45
$S_{\text{MobileAgent-E}}$	36

Table 15: Number of agent-generated shortcuts by different methods.

Task-Level Replay


```
"searchAndClearHistoryOnChrome": {
   "name": "searchAndClearHistoryOnChrome",
   "description": "Searches for a query on Chrome, opens the third result, and then clears the
browsing history.",
   "arguments": ["searchQuery"],
   "actions": [{"action_type": "click", "index": 2}, {"action_type": "input_text",  "index": 2, "text":
"<searchQuery>"}, {"action_type": "scroll",  "direction": "down"}, {"action_type": "click", "index":
37}, {"action_type": "click", "index": 9}, {"action_type": "click", "index": 7}, {"action_type":
"click", "index": 9}, {"action_type": "click", "index": 3}, {"action_type": "click", "index": 3}]
}
```


Subtask-Level Replay


```
"searchOnChrome": {
   "name": "searchOnChrome",
   "description": "Searches for a query on Chrome.",
   "arguments": ["searchQuery"],
   "actions": [{"action_type": "click", "index": 2}, {"action_type": "input_text", "index": 2, "text":
"<searchQuery>"}]
}
"clearChromeHistory": {
   "name": "clearChromeHistory",
   "description": "Deletes the Chrome browsing history for the last 15 minutes.",
   "arguments": [],
   "actions": [{"action_type": "click", "index": 5}, {"action_type": "click", "index": 8},
{"action_type": "click", "index": 1}]
}
```


Dynamic Shortcut


```
"clearChromeHistory": {
    "name": "clearChromeHistory",
    "description": "Navigates to Chrome's history and clears all browsing data.",
    "arguments": ["clearBrowsingDataButton", "clearDataButton", "confirmClearButton",
"historyButton", "moreOptionsButton", "okGotItButton"],
    "actions": [{"action_type": "click", "index": "<moreOptionsButton>"}, {"action_type": "click",
"index": "<historyButton>"}, {"action_type": "click", "index": "<clearBrowsingDataButton>"},
{"action_type": "click", "index": "<clearDataButton>"},{"action_type": "click", "index":
"<confirmClearButton>"}, {"action_type": "click", "index": "<okGotItButton>"}]
}
```


Figure 6: Examples of the resulting shortcut types. Action Replay shortcuts (Task-Level and Subtask-Level) use a sequence of actions with fixed indices, while Dynamic Shortcuts use variable arguments that correspond to UI elements.

primary.

Selection and Planning Errors. This category of error pertains to flaws in the agent's task comprehension and strategic planning. These errors typically occur when the agent fails to accurately map the natural language instruction to the correct shortcut, resulting in the invocation of an irrelevant shortcut that leads to task failure.

For instance, in an experiment with MAS-MobileAgent using Gemini-2.0-Flash (Fig. 8(a)), the agent was given the instruction: "Navigate to the Attractions section. Search for attractions near Berlin for the 6th of next month." After launching the application, the agent made a planning error by incorrectly invoking the `search_hotel()` shortcut, which was irrelevant to the "Attractions" goal, thus causing the task to fail.

Task Type	F1	Precision	Recall
Single App (184 tasks)	96.3	98.1	94.5
Cross App (94 tasks)	95.1	92.5	98.0

Table 16: **Reliability of MAS-Bench-Eval.** We evaluate the automated evaluation system against manual verification on 278 tasks.

Behavioral and Adaptation Errors. This error class occurs when the agent fails to evaluate the outcome of an action and adjust its subsequent action. The deficiency typically stems from an inability to learn from the action histories and modify future behavior accordingly.

As shown in Fig. 8(b), when tasked to "Get the search results for the short videos Stephen Curry. Select the first Short in the results and like it. Leave a comment good shot!", The agent successfully uses shortcuts to search for the video. However, at the step requiring it to select the first video, it fails to switch to a GUI-based action and instead erroneously calls the `open_shorts()` shortcut again. This results in an unproductive loop, demonstrating a failure to adapt its operational mode based on the task's state.

Execution and Formulation Errors. This type of error is specific to the invocation of agent-generated shortcuts. As noted in the main text, the execution success rate for these shortcuts is considerably low. Failures typically result from the shortcut being poorly formulated during the generation phase, rendering it inherently flawed or not robust enough to handle slight variations in the UI during execution.

App name	Tasks	Shortcuts	Shortcut Example
YouTube	11	5	`youtube.search_video_query("cat")`
			`youtube.open_shorts()`
Amazon	10	10	`amazon.open_amazon()`
			`amazon.open_cart()`
Booking.com	10	10	`booking.open_hotelid()`
			`booking.open_my_trips()`
Google Calendar	9	2	`calendar.open_calendar_main()`
			`calendar.create_event(title="Meeting")`
Google Maps	9	8	`google_map.navigate_to_address("Beijing")`
			`google_map.search_address("Square")`
BBC	8	16	`bbc.open_news()`
			`bbc.open_live()`
Fitbit	8	21	`fitbit.open_fitbit_main()`
			`fitbit.open_sleep_log()`
Gmail	8	3	`gmail.open_gmail()`
			`gmail.open_setting()`
Chrome	7	4	`chrome.search_query("Python")`
			`chrome.open_incognito()`
Yelp	8	6	`yelp.open_yelp_main()`
			`yelp.open_events()`
Contacts	4	3	`contacts.open_contacts_main()`
			`contacts.open_contact_starred()`

Table 17: List of MAS-Bench apps and number of single-app tasks and shortcuts for each one.

Figure 7: **Two-Stage Evaluation Pipeline of MAS-Bench-Eval.** Stage 1 (Action Semantics Extraction) employs an efficient MLLM to generate step-by-step textual descriptions of executed actions and UI state transitions. Stage 2 (Success Determination) uses a highly capable MLLM to judge task completion by reasoning over the task instruction, action descriptions from Stage 1, and the final three screenshots.

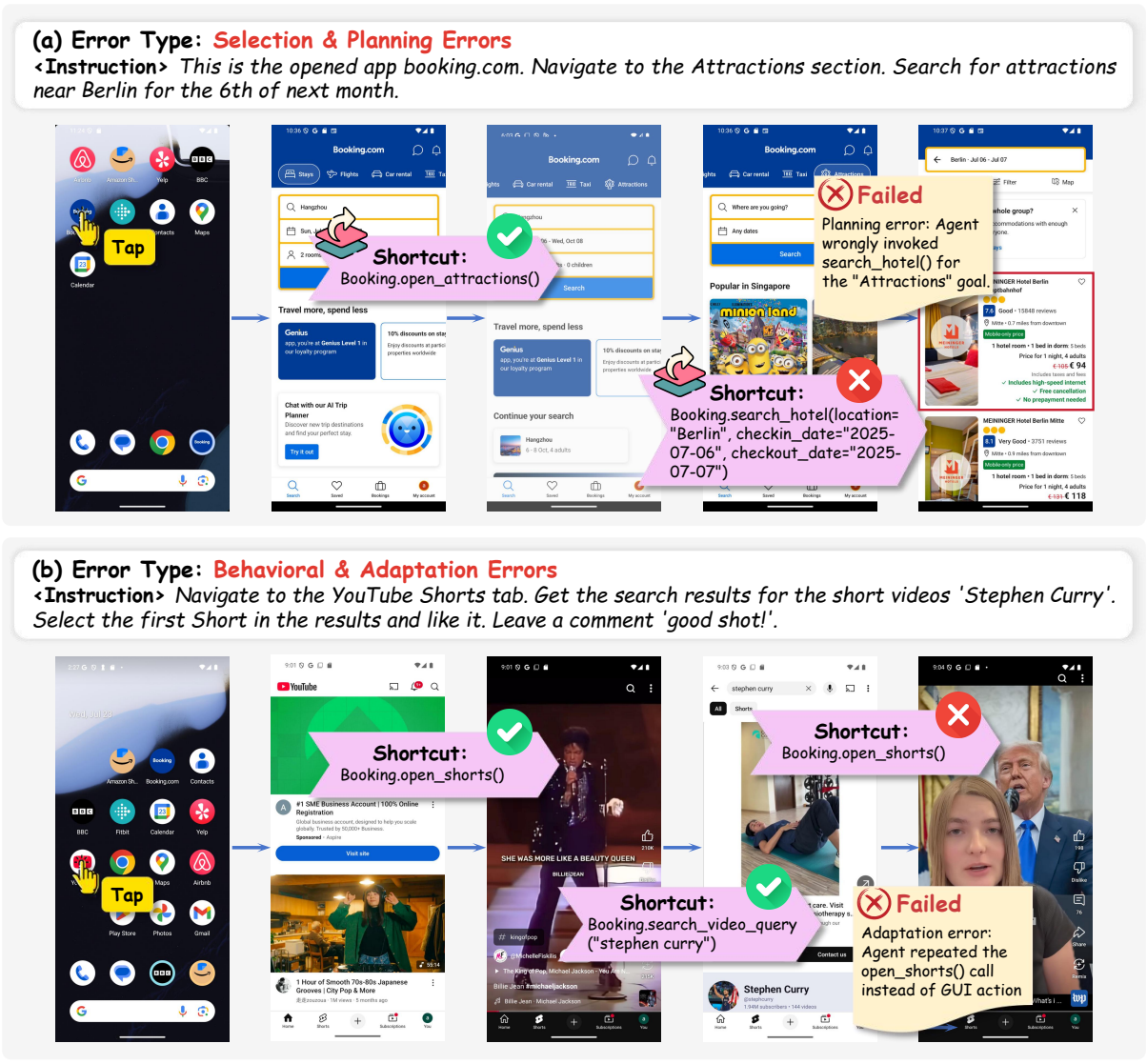

Figure 8: **Examples of GUI-shortcut hybrid agent failure cases.** (a) Selection and Planning Error: The agent incorrectly invokes search_hotel() shortcut instead of searching for attractions, demonstrating failure in mapping natural language instructions to appropriate shortcuts. (b) Behavioral and Adaptation Error: The agent repeatedly calls open_shorts() shortcut instead of switching to a GUI-only action to select the video, showing an inability to adapt operational mode based on task state.