Tencent Hunyuan

PhoneBuddy: Training Open Models for Agentic Phone Use

Zhengyang Tang[1,2,*] **Xin Lai**[1,*] **Pengyuan Lyu**[1,*] **Xinyuan Wang**[1,*] **Tianyi Bai**[1,*] **Chenxin Li**[1,*] **Yiduo Guo**[1,*] **Huawen Shen**[1,*] **Yuxuan Liu**[1,3,*] **Junyi Li**[1] **Zhengyao Fang**[1] **Yang Ding**[1] **Yi Zhang**[1] **Weinong Wang**[1] **Xingran Zhou**[1] **Liang Wu**[1] **Fei Tang**[1] **Sunqi Fan**[1] **Shangpin Peng**[1] **Zheng Ruan**[1] **Anran Zhang**[1] **Benyou Wang**[2] **Ji-Rong Wen**[3] **Rui Yan**[4] **Chengquan Zhang**[1,†] **Han Hu**[1]

[1]Tencent Hunyuan [2]The Chinese University of Hong Kong, Shenzhen
[3]Gaoling School of Artificial Intelligence, Renmin University of China [4]Wuhan University

[*]Equal contribution [†]Project Lead Correspondence to: zhytang@tencent.com

Project page: https://phonebuddyai.github.io/

Abstract. Phones are becoming an important execution surface for general-purpose agents, but training open models for reliable phone use remains difficult because the environment that matters at deployment, real devices running real apps, is slow, stateful, side-effectful, and hard to reset or verify, while scalable mock environments only approximate real behavior. We present *PhoneBuddy*, a training recipe and open-model line for agentic phone use that combines a *real-app environment* with a *mock-app environment*, *PhoneWorld*, which reconstructs runnable mock apps from real GUI usage structure. PhoneBuddy first builds a shared supervised fine-tuning stage from trajectories collected in both environments, then compares real-app RL against mixed RL across both environments. Across a 150-task human evaluation on real phones spanning apps, mini-apps, and cross-app workflows, task success rate improves from 36.67% after supervised fine-tuning to 40.67% after real-app RL and 45.33% after mixed RL. On AndroidWorld, the same progression rises from 60.3% to 77.2% to 83.2%. These results show that mock-app training is not a replacement for real-app RL, but a complementary source of scalable, resettable, and automatically checked interaction. The gains are strongest on app and mini-app tasks, while cross-app workflows remain an important open challenge.

Date: June 11, 2026

1 Introduction

Large language models are increasingly expected not only to answer questions, but also to act through software interfaces. Recent work has pushed this direction across web agents, desktop operating-system agents, tool-using agents, and mobile GUI agents (Deng et al., 2023; Zhou et al.,

2023; Koh et al., 2024; Xie et al., 2024; Bonatti et al., 2024; Yang et al., 2025; Zhang et al., 2023; Wang et al., 2024b; Rawles et al., 2024). Phones are a particularly important execution surface because they are the primary interface for messaging, payments, local services, mini-app ecosystems, personal data, and everyday cross-application workflows. A phone agent is therefore not useful merely because it can recognize widgets or describe a screen; it must reliably complete user tasks under real device state, real application behavior, and real user-facing side effects. This makes phone use a harder target than static GUI grounding: success depends on reading the current screen, deciding which action is safe and useful, maintaining progress over many steps, and verifying that the intended outcome actually happened.

The difficulty is amplified by the structure of real phone tasks. Mobile tasks are stateful, permission-rich, and side-effectful; they depend on login status, app-specific business logic, notification state, device settings, prior user data, and sometimes opaque server-side behavior. They also appear in several interaction regimes: single native apps, mini-apps embedded in host platforms, and cross-app workflows that require transferring information between interfaces. Existing datasets, benchmarks, and agents have made progress on mobile screen understanding, action prediction, real-device evaluation, and long-horizon interaction (Rawles et al., 2023; Deng et al., 2024; Wang et al., 2024c; Xu et al., 2025a,c; Kong et al., 2025; Chai et al., 2025; Liu et al., 2025). These advances are necessary, but they leave a narrower training question unresolved: how should an open phone-use model be trained so that it improves task completion on real phones, rather than only improving local action imitation or benchmark-specific interaction?

Our starting point is the mismatch between realism and scalability. A real-app environment, where agents operate authentic apps on real devices, is the setting that ultimately matters and exposes account-dependent behavior, real side effects, app instability, permission flows, and the gap between apparent progress and completed tasks. However, it is expensive to scale, hard to reset, and difficult to verify automatically. A mock-app environment can be reset, repeated, instrumented, and checked at much lower cost, but it risks training agents on simplified behavior that does not transfer to real phones. This tension appears broadly in recent work on synthetic environments, verifiable software worlds, GUI environment generation, and online RL for computer-use agents (Zala et al., 2024; Cao et al., 2026; Dong et al., 2026; Zhang et al., 2026; Wu et al., 2026; Aggarwal et al., 2026; Wang et al., 2026; Wei et al., 2026; Lai et al., 2025; Zhu et al., 2026). We argue that the practical training recipe should not choose one side of this tradeoff. Real-app training and mock-app training solve different parts of the same problem.

This paper introduces *PhoneBuddy*, a training recipe and open-model line built around this complementarity. The real-app environment supplies realism and late-stage optimization on actual phone execution. The mock-app environment, *PhoneWorld*, supplies scalable, resettable, and automatically verifiable interaction reconstructed from real GUI usage structure (Tang et al., 2026b). The central claim is not that PhoneWorld replaces real apps, or that real apps make mock apps unnecessary. Instead, the claim is that real-app RL and mock-app RL should be combined: real-app RL anchors the model to real device behavior and real side effects, while mock-app training adds broader and cheaper interaction signal from tasks that can be repeated and checked reliably. This framing also separates the training problem studied here from adjacent questions about runtime orchestration, privacy, and safety, which remain essential for deployable phone agents (Jason et al., 2026; Tang et al., 2026a,c; Debenedetti et al., 2024; Tur et al., 2025).

Concretely, we study a compact open 4B model line under three stages: supervised fine-tuning, real-app RL, and mixed RL in both the real-app and mock-app environments. All compared checkpoints share the same Qwen3.5-4B backbone, action interface, and evaluation protocol; they differ only

in the final training branch. On a 150-task human evaluation on real phones spanning apps, mini-apps, and cross-app workflows, task success rate improves from 36.67% after supervised fine-tuning to 40.67% after real-app RL and 45.33% after adding mock-app training. On AndroidWorld, the same model line improves from 60.3% to 77.2% to 83.2%. The gains are strongest on single-app and mini-app tasks, where workflow structure is stable and outcomes are easier to check, while cross-app workflows remain a major limitation. We view this boundary as part of the result: better training environments help substantially, but reliable phone agents still need stronger long-horizon state tracking, information handoff across apps, and runtime verification.

Contributions. This paper makes the following contributions:

- We frame real-world agentic phone use as a training problem for open models, rather than only a GUI grounding problem.
- We present *PhoneBuddy*, a training recipe that combines a real-app environment with *PhoneWorld*, our mock-app environment built from real GUI usage structure.
- We show that the combination of real-app RL and PhoneWorld produces stronger results than either supervised fine-tuning or real-app RL alone, improving task success rate from 36.67% to 45.33% on a 150-task real-phone human evaluation and from 60.3% to 83.2% on AndroidWorld.
- We clarify the current capability boundary of the approach: PhoneWorld-driven gains are strongest on app and mini-app tasks, while cross-app workflows remain a major open challenge for future training and system design.

2 Background

2.1 Mobile and GUI Agents

Recent GUI-agent research has moved from static screen understanding toward agents that can operate real software through visual observations, structured action spaces, and multi-step interaction. Web and desktop environments such as WebArena, VisualWebArena, OSWorld, Windows Agent Arena, and macOSWorld established that open-ended software tasks require grounding, planning, tool use, and robust execution rather than isolated perception (Zhou et al., 2023; Koh et al., 2024; Xie et al., 2024; Bonatti et al., 2024; Yang et al., 2025). Tool-use and workflow benchmarks such as API-Bank, ToolLLM, tau-bench, WorkArena, Toolathlon, OSWorld-MCP, and CocoaBench further shifted evaluation toward executable tasks and outcome-based scoring (Li et al., 2023; Qin et al., 2023; Yao et al., 2024; Drouin et al., 2024a; Li et al., 2025; Jia et al., 2025; CocoaBench Team et al., 2026). Mobile agents extend this challenge to smartphones, where touch actions, app navigation, permissions, account state, personal data, and embedded mini-app ecosystems become part of the task environment. Representative systems and datasets such as AppAgent, Mobile-Agent, Android in the Wild, AndroidWorld, MobileBench, AndroidLab, and MobileWorld have improved mobile action prediction, real-device evaluation, and long-horizon interaction (Zhang et al., 2023; Wang et al., 2024b; Rawles et al., 2023, 2024; Deng et al., 2024; Xu et al., 2025c; Kong et al., 2025). These works motivate PhoneBuddy's focus on training models that can complete real phone tasks, not only predict plausible next actions.

2.2 Environment Scaling and Online Optimization

The central training bottleneck is environment scale. Real applications provide high-fidelity behavior, but collecting trajectories, resetting state, and verifying outcomes are expensive. Synthetic or reconstructed environments provide cheaper interaction and stronger supervision, but they must preserve enough structure to transfer to real software. Recent work on EnvGen, GUI-Genesis, Agent-World, InfiniteWeb, AutoWebWorld, Gym-Anything, CUA-Gym, OpenComputer, ComputerRL, and Workflow-GYM explores this broader direction of generated, verifiable, or online-trainable environments for agents (Zala et al., 2024; Cao et al., 2026; Dong et al., 2026; Zhang et al., 2026; Wu et al., 2026; Aggarwal et al., 2026; Wang et al., 2026; Wei et al., 2026; Lai et al., 2025; Zhu et al., 2026). PhoneWorld follows the same scaling logic in the phone domain: it reconstructs runnable mock apps from real GUI usage structure so that tasks can be reset, repeated, and checked automatically. PhoneBuddy asks how such mock-app training should be combined with real-app RL, rather than treating either environment as sufficient by itself.

2.3 Agent Harness and Safety Protection

Training improves the model policy, but a deployable agent also needs a harness that turns model predictions into controlled interaction with real software. Such a harness defines the observation stream, action schema, parser, execution backend, step budget, logging format, and task-level completion checks; it also decides when to use GUI actions, tool calls, CLI commands, or other execution channels. Recent work on tool and workflow agents shows that this runtime layer is part of the agent capability itself, especially when workflows evolve over time or require multiple interaction modes (Li et al., 2025; Jia et al., 2025; Yang et al., 2026b; Li et al., 2026). PhoneHarness follows this direction for phone agents by coordinating mixed GUI, CLI, and MCP-style actions around a shared phone-task interface (Jason et al., 2026). This paper focuses on the training recipe, but we treat the harness as a necessary deployment layer: it mediates between model outputs and real side effects, provides execution traces for debugging and learning, and supplies the structure needed for future online adaptation (Huang et al., 2026). Phone agents also operate close to sensitive user data, so the harness must act as a safety boundary rather than only an action executor. Prior work on sandboxed risk evaluation, web-agent safety, phone privacy, and phone safety shows that capable agents still require guardrails, explicit runtime boundaries, permission checks, and careful evaluation of harmful or privacy-sensitive behavior (Ruan et al., 2023; Debenedetti et al., 2024; Zhang et al., 2024; Tur et al., 2025; Tang et al., 2026a,c).

3 Method

3.1 Problem Setting

PhoneBuddy targets the final stage of training a phone-use model. To solve user instruction, at each step, the agent observes the current screen together with the instruction and interaction history, and predicts a single action. An episode ends when the agent declares the task finished or exhausts its step budget.

The central difficulty stems from a mismatch between the requirements of training and deployment. An effective training environment should be easy to reset and to verify automatically, so that the policy can be optimized against reliable, repeatable outcome signals. The deployment target,

however, is a real phone running authentic apps, whose persistent state and irreversible side effects are precisely what make resetting and automatic verification costly. The two demands therefore stand in tension, and neither can be satisfied by a single environment alone. PhoneBuddy is designed to bridge this gap by training across both: a real-app environment for fidelity and a mock-app environment for scalable, verifiable interaction.

3.2 Overview of PhoneBuddy

PhoneBuddy is a training recipe that turns a single base model into a phone-use agent through a shared supervised fine-tuning (SFT) stage followed by reinforcement learning across two complementary environments, as illustrated in Figure 1. All checkpoints start from the same Qwen3.5-4B backbone and share the same SFT initialization, action interface, and evaluation protocol; they differ only in the final RL branch. This design isolates our object of study: how different training strategies—particularly the choice of reinforcement-learning environment—affect the agent's ability to complete real phone tasks. With architecture, initialization, and data held fixed, any difference in task success is attributable to the final training branch alone.

Throughout the paper we distinguish two training environments:

- a **real-app environment**, in which the agent operates authentic apps on real devices;
- a **mock-app environment**, in which the agent operates runnable mock apps that can be reset and verified automatically. The environment used is **PhoneWorld** (Tang et al., 2026b).

Starting from the shared SFT checkpoint, we compare three variants: the SFT baseline itself (*PhoneBuddy-4B-SFT*), a model further trained with RL only in the real-app environment (*PhoneBuddy-4B-Real*), and a model further trained with mixed RL in both real-app and mock-app environments (*PhoneBuddy-4B-Real+Mock*).

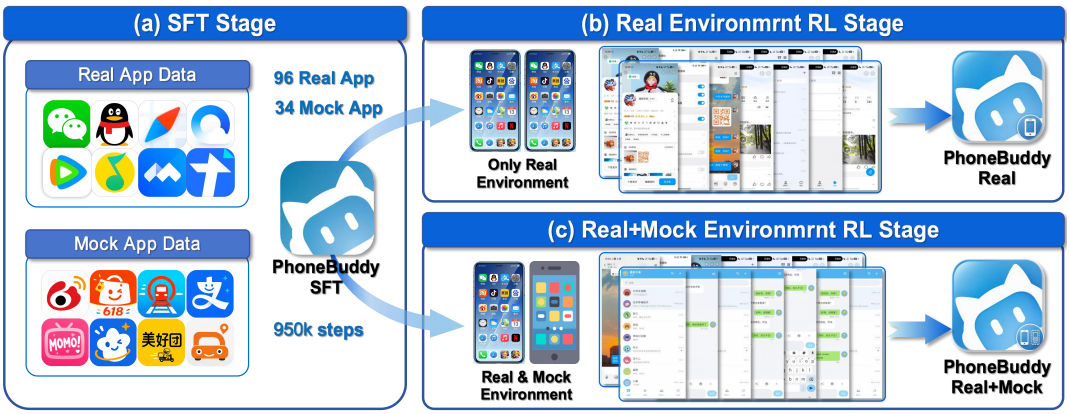

Figure 1: Overview of PhoneBuddy. A shared SFT stage uses trajectories from both the real-app and mock-app environments, after which the same PhoneBuddy-4B-SFT model is branched into a real-app RL checkpoint and a mixed real+mock RL checkpoint.

3.3 Real-App Environment

The real-app environment runs authentic apps on physical devices. It is indispensable because it is the environment the model must ultimately operate in: it faithfully exposes the real app behavior, device state, timing variation, and user-facing side effects that govern actual phone use.

Crucially, it surfaces failure modes that mock apps cannot fully reproduce, such as account-dependent behavior, app-specific instability, permission flows, and the gap between apparent progress and genuine task completion. It also enables *real-app RL*, which we treat as the primary late-stage step for improving task completion on real phones.

Its drawback is cost: rollouts are slower, state is harder to reset, automatic verification is more fragile than in a mock-app environment, and exploration carries real, sometimes irreversible side effects that demand additional risk controls. PhoneBuddy therefore uses the real-app environment selectively, to keep training aligned with deployment while avoiding the cost of relying on it alone.

3.4 Mock-App Environment (PhoneWorld)

PhoneWorld (Tang et al., 2026b) is our mock-app environment. Here "mock app" does not mean a toy demo or a static prototype. It means a runnable Android app reconstructed from real GUI traces, with state that can change and with rules for checking whether a task is finished.

PhoneWorld employs a pipeline to build mock apps from real GUI trajectories and screenshots. From them, it recovers which screens matter, how screens connect, which actions need to be supported, and which state changes need to be saved. It then builds runnable mock Android apps with both read-only content and writable state. From the same apps, it derives tasks and rule-based verifiers so that success can be checked automatically rather than by manual inspection.

In its current version, PhoneWorld spans dozens of consumer-style mobile environments and supplies a large pool of executable tasks and trajectories. For the purposes of this paper, what matters is not the implementation details of any individual generated app, but the role PhoneWorld plays in training: it provides scale, repeatability, and automatic verification precisely in the setting where real-app training is most constrained.

3.5 Training Recipe

Our main empirical study isolates the effect of the final training recipe. All compared checkpoints share the same backbone, action interface, and evaluation protocol.

All three checkpoints share the same supervised fine-tuning stage. In the current training stack, we first collect phone-use trajectories from both the real-app environment and the mock-app environment, and use them to build a shared SFT dataset. Starting from this shared SFT model, we then branch into two RL settings: RL only in the real-app environment, and mixed RL in both the real-app and mock-app environments.

This shared SFT stage matters for the comparison. It puts both environments into the same training format: the model sees the task instruction and current phone screen, and predicts the next phone action. As a result, the later comparison between *PhoneBuddy-4B-Real* and *PhoneBuddy-4B-Real+Mock* isolates the value of the RL branch rather than differences in the basic action interface.

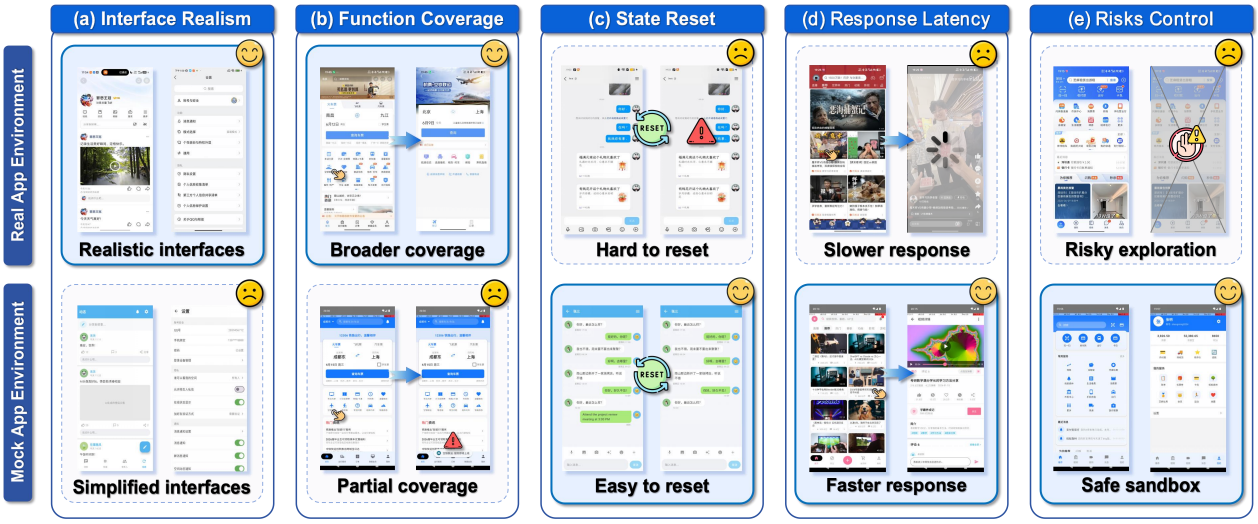

Figure 2: Complementary roles of the real-app and mock-app environments. The real-app environment provides authentic device behavior, app logic, and user-facing side effects, while PhoneWorld provides resettable mock apps, automatic verification, and scalable rollout collection. PhoneBuddy uses both environments rather than treating either one as a complete substitute for the other.

The shared SFT stage starts from Qwen3.5-4B and uses a combined dataset of 950,758 action steps collected from the real-app and mock-app environments. We perform full-parameter fine-tuning for 1,115 optimizer steps with batch size 512. Training uses packed 8,192-token sequences, where shorter examples are concatenated with attention masking between segments to improve utilization. The learning rate decays from 1×10^{-5} to 1×10^{-6}. Because multiple short examples can be packed into one training sequence, the product of batch size and optimizer steps should not be read as a direct count of raw action steps.

For both RL branches, we run 50 online RL steps after the shared SFT stage. The reward is computed with rubric-based verification rather than a single holistic judgment on the whole trajectory. Concretely, for each instruction we first use Gemini-3.1-Pro-Preview to generate a set of task-specific rubrics, then use Qwen/Qwen3.5-122B-A10B to score the agent trajectory against each rubric item. A trajectory is counted as correct only when every rubric item is judged as passed. We use this rubric-by-rubric reward because it is more accurate and more stable than directly scoring the entire trajectory with one overall decision, which in turn makes online RL training more stable.

PhoneBuddy-4B-SFT. This is the supervised fine-tuning baseline. It is trained on phone-use trajectories to establish a common task-completion starting point before online optimization. In the main result table, this checkpoint serves as the reference for measuring the gains from RL and PhoneWorld augmentation.

PhoneBuddy-4B-Real. This checkpoint continues training with reinforcement learning in the real-app environment. The goal of this stage is simple: improve performance on real phone execution, including real app behavior, real device state transitions, and real user-facing side effects. We run 50 online RL steps in the real-app environment only. The reward uses the rubric-based verification described above and is tied to whether the intended phone task is actually completed under real execution.

Checkpoint	Shared SFT Data	RL Environment	Training Objective	Purpose
PhoneBuddy-4B-SFT	Real-app + mock-app trajectories	–	Supervised fine-tuning	Common starting point before RL
PhoneBuddy-4B-Real	Real-app + mock-app trajectories	Real-app only	Reinforcement learning on real phone execution	Improve real-phone task completion
PhoneBuddy-4B-Real+Mock	Real-app + mock-app trajectories	Real-app + mock-app	Mixed reinforcement learning in both environments	Combine real execution with scalable verified interaction

Table 1: Training recipe used in the main comparison. All three checkpoints share the same SFT stage and differ only in the final training branch.

PhoneBuddy-4B-Real+Mock. This checkpoint uses mixed RL in both the real-app environment and the mock-app environment. The key idea is not to replace real-app training, but to supplement it with broader and easier-to-verify phone-use interaction. PhoneWorld contributes task environment that can be reset, repeated, and checked automatically, while real-app RL keeps the model tied to real execution. We also run 50 online RL steps in this branch, with a 50%/50% real/mock rollout mixture. The reward is the same rubric-based signal used for the real-only RL.

At a high level, both RL branches optimize for task completion. The difference is where the outcome signal comes from. In the real-app environment, the signal comes from whether the intended phone task is completed under real execution. In the mock-app environment, the signal comes from the built-in task verifiers provided by PhoneWorld. This keeps the optimization target aligned across environments even though the environments themselves are different.

Why compare these three checkpoints?

The objective of this comparison is not to establish the value of PhoneWorld or real-app RL in isolation, but **to determine whether combining the real-app environment and mock-app environment constitutes a more effective training recipe for phone use agents**. Accordingly, these three checkpoints differ only in their final RL stage, which allows any difference in task success to be attributed directly to the choice of RL environment.

4 Experimental Setup

4.1 Benchmarks and Metrics

We evaluate PhoneBuddy on four task settings. The first three come from our real-phone human evaluation suite: **Single-App Tasks**, **Cross-App Tasks**, and **WeChat Mini-App Tasks**, with 50 tasks in each category for a total of 150 tasks. The fourth setting is **AndroidWorld**. For all four settings, we report *task success rate*. In our real-phone suite, a task is counted as successful only when it is fully completed.

For the main table, we report one number for each of these four settings and an overall **Avg.** computed as the unweighted mean of the four columns. This presentation gives a cleaner view of

where the model is strong and where it still fails. In particular, it prevents improvements on one subset from hiding weaknesses on another subset.

4.2 Evaluation Protocol

All compared checkpoints are evaluated under the same inference and execution setup. For the real-phone human evaluation, a task is counted as successful if human annotators judge that the requested task has been fully completed on the device. We report task success rate only.

We keep the action space, prompt template, step budget, and evaluation harness fixed across compared checkpoints, and change only the training recipe. This is important because the paper aims to isolate the value of the three training branches rather than differences in prompting or runtime setup.

The action space is a shared phone-control API with normalized coordinates in the $[0, 1000]$ range. The model predicts one action at each step from the following set: click, double click, long press, type, scroll, drag, button press with back/home/menu/enter, open app, close app, and wait. During training, the same prompt format also includes task-level communication actions for asking the user for clarification, outputting information, and marking the task as finished, but the core execution interface used for phone control remains fixed across compared checkpoints.

The prompt template is also held fixed. Each inference step is framed as a multimodal action prediction problem with the current screenshot and a structured textual context. The textual prompt contains the user instruction, a serialized history of prior thought-action pairs, and an intermediate-state field carried over from the previous step. The system-side prompt defines the full tool schema and instructs the model to output exactly one structured tool call enclosed by dedicated tags. The response may optionally include a reasoning block and an updated intermediate state, but execution uses only the parsed structured action. At inference time, we extract the tagged tool call, repair minor JSON formatting errors when needed, and map the result into a shared internal action representation for execution. This parsing layer is kept fixed for all compared models.

We use a maximum step budget of 30 during training and 50 during evaluation. The larger test-time budget reduces truncation on long-horizon tasks while preserving the same action interface, prompt contract, and execution stack across all compared checkpoints.

4.3 Model Variants

Our main internal comparison uses three checkpoints from the same 4B line:

- **PhoneBuddy-4B-SFT**: the supervised fine-tuning baseline.
- **PhoneBuddy-4B-Real**: the model after real-app RL.
- **PhoneBuddy-4B-Real+Mock**: the model after mixed RL in both the real-app and mock-app environments.

We compare these models against representative strong closed-source systems, including Gemini 3.1 Pro, GPT-5.4, Claude Opus 4.7, and Seed 2.0.

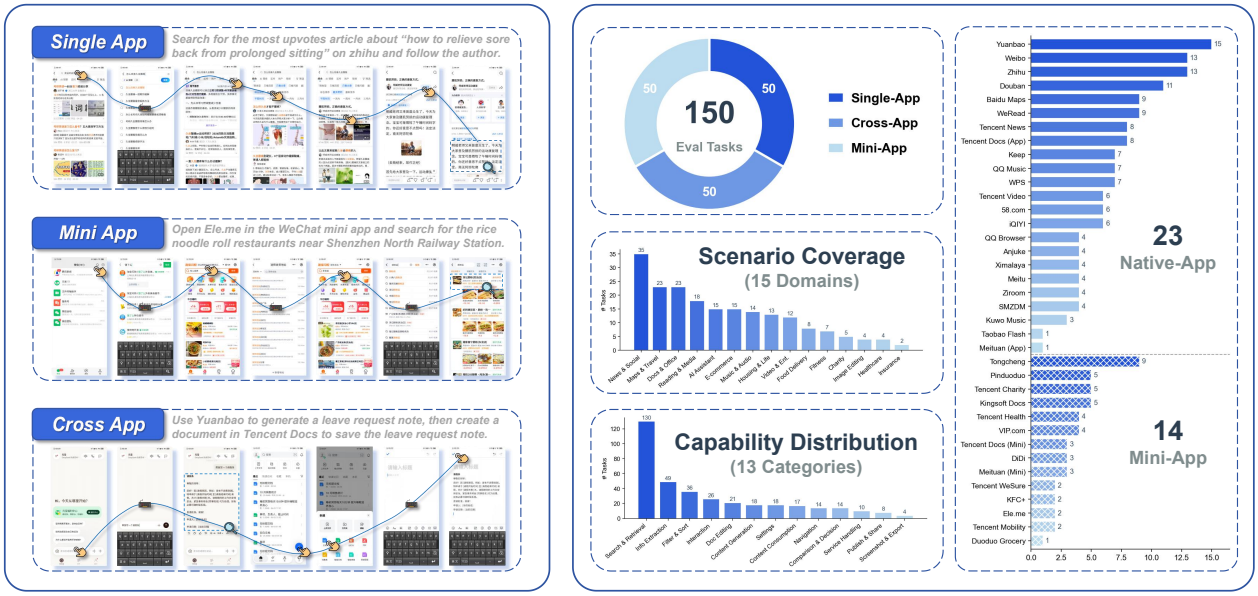

Figure 3: Evaluation settings used in this paper. The real-phone human evaluation covers Single-App, Cross-App, and WeChat Mini-App tasks, each with 50 tasks, while AndroidWorld provides an external dynamic mobile benchmark. All settings are evaluated with task success rate.

Model	Single-App	Cross-App	WeChat Mini-App	AndroidWorld	Avg.
Gemini 3.1 Pro	50.0	**48.0**	58.0	80.2	**59.1**
GPT-5.4	50.0	32.0	40.0	70.7	48.2
Claude Opus 4.7	38.0	16.0	28.0	56.0	34.5
Seed 2.0	44.0	30.0	**60.0**	71.5	51.4
PhoneBuddy-4B-SFT (ours)	34.0	22.0	54.0	60.3	42.6
PhoneBuddy-4B-Real (ours)	54.0	20.0	48.0	77.2	49.8
PhoneBuddy-4B-Real+Mock (ours)	**62.0**	18.0	56.0	**83.2**	54.8

Table 2: Main results across four task settings. The first three columns come from the 150-task real-phone human evaluation, with 50 tasks each for Single-App, Cross-App, and WeChat Mini-App. All columns report task success rate. For the real-phone human evaluation, task success is defined strictly: a task counts as successful only when it is fully completed. Avg. is the unweighted mean of the four columns.

5 Main Results

Table 2 summarizes the main result. The table is intentionally organized around four task settings rather than only a single overall number. This makes the main point easier to see: PhoneWorld helps on top of real-app RL, but the gain is not the same on every task type.

Avg. The best overall internal model is *PhoneBuddy-4B-Real+Mock*. It reaches 54.8 average task success rate across the four settings, improving over *PhoneBuddy-4B-SFT* by 12.2 points and over *PhoneBuddy-4B-Real* by 5.0 points. It also outperforms GPT-5.4 and Seed 2.0 on this average, while remaining below Gemini 3.1 Pro overall.

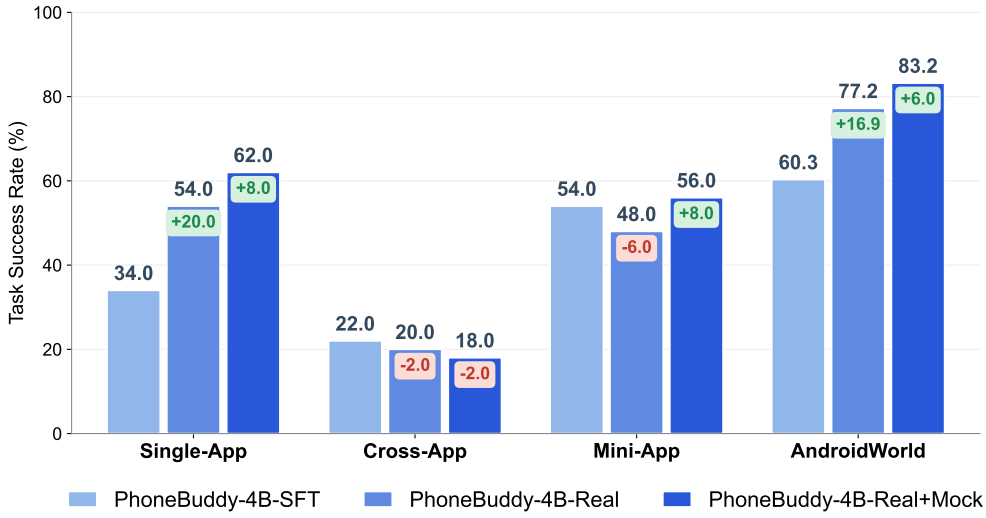

Figure 4: Incremental gains from the two RL branches. The first delta measures the effect of real-app RL over the shared SFT checkpoint, and the second delta measures the additional effect of adding mock-app RL on top of real-app RL. The plot highlights that PhoneWorld improves Single-App, WeChat Mini-App, and AndroidWorld performance, while Cross-App tasks remain difficult.

Single-App Tasks. Single-App Tasks show the clearest gain from the full recipe. Performance rises from 34.0% for *PhoneBuddy-4B-SFT* to 54.0% for *PhoneBuddy-4B-Real*, and then to 62.0% for *PhoneBuddy-4B-Real+Mock*. This is the strongest score in the column, ahead of all compared closed models. The result suggests that real-app RL teaches the model to execute real phone actions more reliably, while mixed RL adds extra coverage on structured app interactions that benefit from repeatable training.

Cross-App Tasks. Cross-App Tasks remain the main gap. The current recipe does not improve this setting: the scores are 22.0%, 20.0%, and 18.0% for *PhoneBuddy-4B-SFT*, *PhoneBuddy-4B-Real*, and *PhoneBuddy-4B-Real+Mock*, respectively. We interpret this result as a real boundary rather than a presentation issue. Cross-app phone use likely requires stronger handling of information transfer across apps, better long-horizon state tracking, and stronger intermediate checking than the current training recipe provides.

WeChat Mini-App Tasks. WeChat Mini-App Tasks show a different pattern. Real-app RL alone does not help this subset, dropping from 54.0% to 48.0%, but mixed RL lifts the score to 56.0%. This is modestly above the SFT baseline and suggests that PhoneWorld is especially helpful when the workflow is multi-step but structurally stable, with state changes that are easier to verify and repeat during training.

AndroidWorld. AndroidWorld shows the cleanest monotonic trend: 60.3% for *PhoneBuddy-4B-SFT*, 77.2% for *PhoneBuddy-4B-Real*, and 83.2% for *PhoneBuddy-4B-Real+Mock*. The final model is also the best overall system in this column. This matters because AndroidWorld is outside the real-phone human evaluation suite used for the first three columns. The gain therefore supports the transfer value of the training recipe rather than only fitting to one internal benchmark.

6 Qualitative Examples

Figure 5 shows two representative trajectories that reveal how Real+Mock training improves execution beyond the aggregate success rates.

• In **Mini-App** case, the agent must search for budget-friendly hotels near Shanghai Disneyland in the WeChat mini-app Tongcheng Travel. PhoneBuddy-SFT reaches a plausible hotel-search page but does not apply the budget constraint, while PhoneBuddy-Real+Mock continues to the filtering interface and reduces the hotel budget to 150 yuan.

• In **Cross-App** case, the agent must generate a leave request note with Yuanbao and save it in Tencent Docs. PhoneBuddy-SFT fails to copy the note generated by Yuanbao and instead inserts stale clipboard content when creating the document. By contrast, PhoneBuddy-Real+Mock correctly copies the generated leave request note and pastes it into the newly created document.

These cases suggest that the benefit of mock-app training is not limited to more exploration. Its resettable and repeatable workflows provide useful supervision for constraint following and content transfer, while real-app training preserves realistic interface and service behavior.

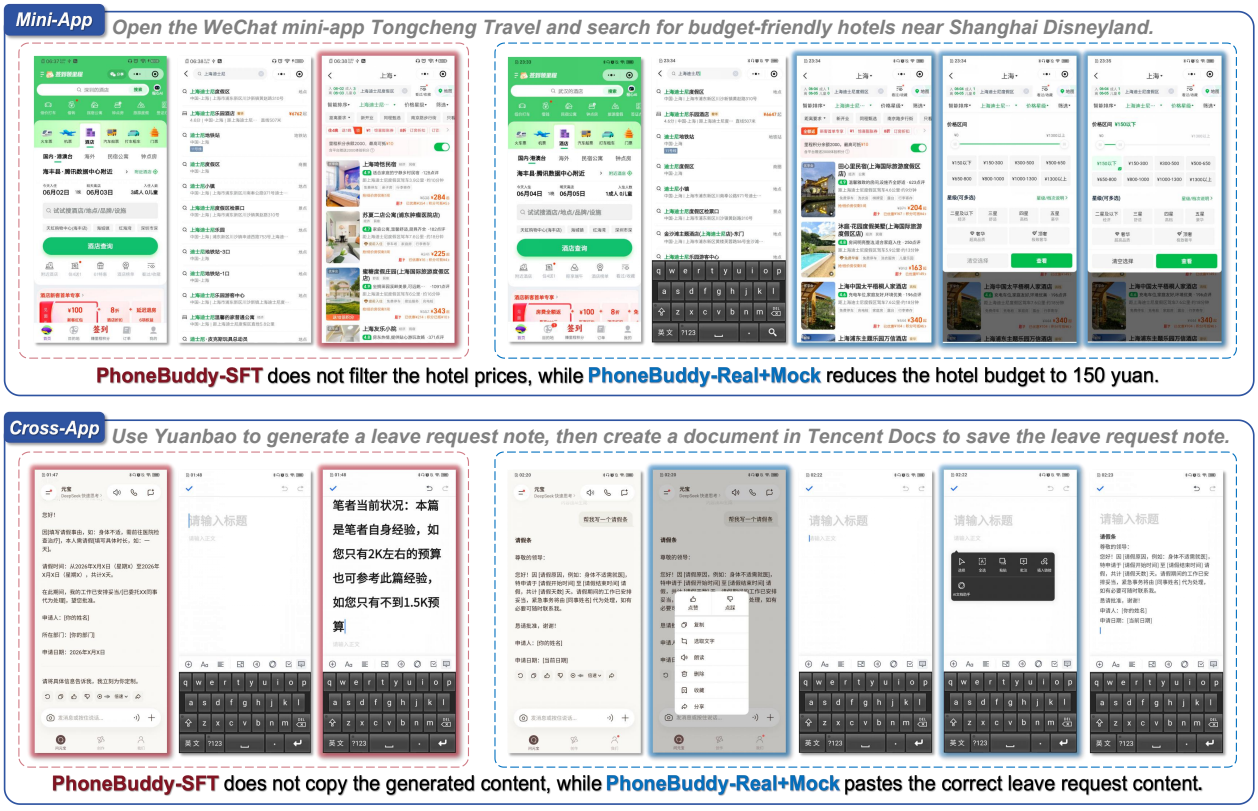

Figure 5: Representative successful trajectories. PhoneBuddy-Real+Mock better preserves task constraints and information transfer in both mini-app and cross-app tasks.

7 Discussion and Limitations

Why Real+Mock Works. The current results support a fairly specific conclusion. Real-app RL and PhoneWorld are complementary. Real-app RL ties the model to real device behavior, real app logic, and real side effects. PhoneWorld then adds scale, easier reset, and automatic verification. This combination is especially effective on tasks where the workflow is stable and the end state is easy to check.

Why Cross-App Still Lags. Cross-app execution remains a clear weakness. We do not think this is only a data-volume problem. Cross-app tasks often require carrying information across interfaces, remembering which artifact was created earlier, re-entering the right app at the right state, and checking whether an intermediate side effect actually happened. Better training helps, but the current results suggest that these workflows also stress runtime coordination and verification.

What This Paper Does Not Solve. This paper is intentionally about training. A deployable phone agent also needs a strong runtime system and clear deployment boundaries around privacy and safety. Those pieces matter for real use, but they are deliberately not the empirical center of this report.

More broadly, PhoneBuddy is the training layer in a larger phone-agent matrix from our research line. PhoneWorld builds the mock-app environments used for scalable training and evaluation (Tang et al., 2026b). PhoneBuddy studies how to train the model itself. PhoneHarness studies runtime execution (Jason et al., 2026), and PhonePrivacy / PhoneSafety study deployment boundaries (Tang et al., 2026a,c). This paper focuses only on the training layer, but it fits into that larger stack rather than standing alone.

8 Conclusion

This paper studies how to train open models for real-world agentic phone use. The main lesson is simple: real-app training alone is not enough, and mock-app training alone is not enough. Real-app RL provides realism; PhoneWorld provides scale, reset, and verification. In the current study, the strongest recipe is a shared SFT stage built from both environments followed by mixed RL across both environments. This recipe improves task success on both our real-phone human evaluation and AndroidWorld, supporting the view that mock-app interaction can transfer when it is grounded in realistic GUI structure. At the same time, the weak cross-app results show that environment scaling does not by itself solve long-horizon state tracking, information handoff, or runtime coordination. Future work should therefore combine better training environments with stronger execution harnesses, intermediate verification, and safety-aware deployment boundaries for real phone agents.

References

Pranjal Aggarwal, Graham Neubig, and Sean Welleck. Gym-anything: Turn any software into an agent environment. *arXiv preprint arXiv:2604.06126*, 2026. URL https://arxiv.org/abs/2604.06126.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024. URL https://arxiv.org/abs/2409.08264.

Yuan Cao, Dezhi Ran, Mengzhou Wu, Yuzhe Guo, Xin Chen, Ang Li, Gang Cao, Gong Zhi, Hao Yu, Linyi Li, et al. Gui-genesis: Automated synthesis of efficient environments with verifiable rewards for gui agent post-training. *arXiv preprint arXiv:2602.14093*, 2026. URL https://arxiv.org/abs/2602.14093.

Yuxiang Chai, Shunye Tang, Han Xiao, Weifeng Lin, Liang Liu, Hanhao Li, Jiayu Zhang, Pengxiang Zhao, Guangyi Liu, Rongduo Han, et al. A3: Android agent arena for mobile gui agents. *arXiv preprint*, 2025.

CocoaBench Team, Shibo Hao, Zhining Zhang, Zhiqi Liang, Tianyang Liu, Yuheng Zha, Qiyue Gao, Jixuan Chen, et al. Cocoabench: Evaluating unified digital agents in the wild. *arXiv preprint arXiv:2604.11201*, 2026. URL https://arxiv.org/abs/2604.11201.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *arXiv preprint arXiv:2406.13352*, 2024. URL https://arxiv.org/abs/2406.13352.

Shihan Deng, Weikai Xu, Hongda Sun, Wei Liu, Tao Tan, Liujianfeng Liujianfeng, Ang Li, Jian Luan, Bin Wang, Rui Yan, et al. Mobile-bench: An evaluation benchmark for llm-based mobile agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 8813–8831, 2024.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023. URL https://arxiv.org/abs/2306.06070.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026. URL https://arxiv.org/abs/2604.18292.

Alexandre Drouin et al. Workarena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024a. URL https://arxiv.org/abs/2403.07718.

Alexandre Drouin et al. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *arXiv preprint arXiv:2407.05291*, 2024b. URL https://arxiv.org/abs/2407.05291.

Junan Hu, Jian Liu, Jingxiang Lai, Jiarui Hu, Yiwei Sheng, Shuang Chen, Jian Li, Dazhao Du, et al. Gui agents with reinforcement learning: Toward digital inhabitants. *arXiv preprint arXiv:2604.27955*, 2026. URL https://arxiv.org/abs/2604.27955.

Kun Huang, Weikai Xu, Yuxuan Liu, Quandong Wang, Pengzhi Gao, Wei Liu, Jian Luan, Bin Wang, and Bo An. Mobileipl: Enhancing mobile agents thinking process via iterative preference learning. *arXiv preprint arXiv:2505.12299*, 2025. URL https://arxiv.org/abs/2505.12299.

Shijue Huang, Hangyu Guo, Chenxin Li, Junting Lu, Xinyu Geng, Zhaochen Su, Zhenyu Li, Shuang Chen, Hongru Wang, and Yi R. Fung. Towards on-policy data evolution for visual-native multimodal deep search agents. *arXiv preprint arXiv:2605.10832*, 2026. URL https://arxiv.org/abs/2605.10832.

Jason, Zhengyao Fang, Zhengyang Tang, Pengyuan Lyu, Xingran Zhou, Xin Lai, Fei Tang, Liang Wu, Yiduo Guo, Weinong Wang, Junyi Li, Yi Zhang, Yang Ding, Huawen Shen, Sunqi Fan, Shangpin Peng, Zheng Ruan, Anran Zhang, Benyou Wang, Chengquan Zhang, and Han Hu. Phoneharness: A mixed-action orchestration harness and benchmark for phone agents across cli, gui, and mcp tools. https://github.com/PhoneHarness/PhoneHarness, 2026. GitHub repository.

Hongrui Jia, Jitong Liao, Xi Zhang, Haiyang Xu, Tianbao Xie, Chaoya Jiang, Ming Yan, Si Liu, Wei Ye, and Fei Huang. Osworld-mcp: Benchmarking mcp tool invocation in computer-use agents. *arXiv preprint arXiv:2510.24563*, 2025. URL https://arxiv.org/abs/2510.24563.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *arXiv preprint arXiv:2401.13649*, 2024. URL https://arxiv.org/abs/2401.13649.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*, 2025. URL https://arxiv.org/abs/2512.19432.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, et al. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents. *arXiv preprint arXiv:2508.14040*, 2025. URL https://arxiv.org/abs/2508.14040.

Chenxin Li, Zhengyang Tang, Mingxin Huang, Yunlong Lin, Shijue Huang, Shengyuan Liu, Bowen Ye, Rang Li, Lei Li, Benyou Wang, and Yixuan Yuan. Claw-eval-live: A live agent benchmark for evolving real-world workflows. *arXiv preprint arXiv:2604.28139*, 2026. URL https://arxiv.org/abs/2604.28139.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. *arXiv preprint arXiv:2510.25726*, 2025. URL https://arxiv.org/abs/2510.25726.

Minghao Li et al. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023. URL https://arxiv.org/abs/2304.08244.

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

Yuxuan Liu, Hongda Sun, Wei Liu, Jian Luan, Bo Du, and Rui Yan. Mobilesteward: Integrating multiple app-oriented agents with self-evolution to automate cross-app instructions. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 883–893, 2025.

Dunjie Lu, Yiheng Xu, Junli Wang, Haoyuan Wu, Xinyuan Wang, Zekun Wang, Junlin Yang, Hongjin Su, Jixuan Chen, Junda Chen, et al. Videoagenttrek: Computer use pretraining from unlabeled videos. *arXiv preprint arXiv:2510.19488*, 2025. URL https://arxiv.org/abs/2510.19488.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025. URL https://arxiv.org/abs/2504.10458.

Shishir G. Patil et al. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023. URL https://arxiv.org/abs/2305.15334.

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

Yujia Qin et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023. URL https://arxiv.org/abs/2307.16789.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Android in the wild: A large-scale dataset for android device control. *arXiv preprint arXiv:2307.10088*, 2023. URL https://arxiv.org/abs/2307.10088.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*, 2024. URL https://arxiv.org/abs/2405.14573.

Yangjun Ruan et al. Identifying the risks of lm agents with an lm-emulated sandbox. *arXiv preprint arXiv:2309.15817*, 2023. URL https://arxiv.org/abs/2309.15817.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025. URL https://arxiv.org/abs/2507.05720.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, et al. Agents' last exam. *arXiv preprint arXiv:2606.05405*, 2026. URL https://arxiv.org/abs/2606.05405.

Zhengyang Tang, Ke Ji, Xidong Wang, Zihan Ye, Xinyuan Wang, Yiduo Guo, Ziniu Li, Chenxin Li, Jingyuan Hu, Shunian Chen, Tongxu Luo, Jiaxi Bi, Zeyu Qin, Shaobo Wang, Xin Lai, Pengyuan Lyu, Junyi Li, Can Xu, Chengquan Zhang, Han Hu, Ming Yan, and Benyou Wang. Do phone-use agents respect your privacy?, 2026a.

Zhengyang Tang, Yuxuan Liu, Xin Lai, Junyi Li, Pengyuan Lyu, Jason, Yiduo Guo, Zhengyao Fang, Yang Ding, Yi Zhang, Weinong Wang, Huawen Shen, Xingran Zhou, Liang Wu, Fei Tang, Sunqi Fan, Shangpin Peng, Zheng Ruan, Anran Zhang, Benyou Wang, Rui Yan, Ji-Rong Wen, Chengquan Zhang, and Han Hu. Phoneworld: Scaling phone-use agent environments, 2026b.

Zhengyang Tang, Yi Zhang, Chenxin Li, Xin Lai, Pengyuan Lyu, Yiduo Guo, Weinong Wang, Junyi Li, Yang Ding, Huawen Shen, Zhengyao Fang, Xingran Zhou, Liang Wu, Fei Tang, Sunqi Fan, Shangpin Peng, Zheng Ruan, Anran Zhang, Benyou Wang, Chengquan Zhang, and Han Hu. Safe, or simply incapable? rethinking safety evaluation for phone-use agents, 2026c. arXiv preprint.

Hung Tran, Langston Nashold, Rayan Krishnan, Antoine Bigeard, and Alex Gu. Vibe code bench: Evaluating ai models on end-to-end web application development. *arXiv preprint arXiv:2603.04601*, 2026. URL https://arxiv.org/abs/2603.04601.

Ada Defne Tur, Nicholas Meade, Xing Han Lù, Alejandra Zambrano, Arkil Patel, Esin Durmus, Spandana Gella, Karolina St-Pierre, and Siva Reddy. Safearena: Evaluating the safety of autonomous web agents. *arXiv preprint arXiv:2503.04957*, 2025. URL https://arxiv.org/abs/2503.04957.

Bowen Wang, Dunjie Lu, Junli Wang, Tianyi Bai, Shixuan Liu, Zhipeng Zhang, Haiquan Wang, Hao Hu, et al. Cua-gym: Scaling verifiable training environments and tasks for computer-use agents. *arXiv preprint arXiv:2605.25624*, 2026. URL https://arxiv.org/abs/2605.25624.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025a. URL https://arxiv.org/abs/2509.02544.

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

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

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

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025b. URL https://arxiv.org/abs/2508.09123.

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

Jinbiao Wei, Qianran Ma, Yilun Zhao, Xiao Zhou, Kangqi Ni, Guo Gan, and Arman Cohan. Opencomputer: Verifiable software worlds for computer-use agents. *arXiv preprint arXiv:2605.19769*, 2026. URL https://arxiv.org/abs/2605.19769.

Yifan Wu, Yiran Peng, Yiyu Chen, Jianhao Ruan, Zijie Zhuang, Cheng Yang, Jiayi Zhang, Man Chen, Yenchi Tseng, Zhaoyang Yu, et al. Autowebworld: Synthesizing infinite verifiable web environments via finite state machines. *arXiv preprint arXiv:2602.14296*, 2026. URL https://arxiv.org/abs/2602.14296.

Ziqiao Xi, Shuang Liang, Qi Liu, Jiaqing Zhang, Letian Peng, Fang Nan, Meshal Nayim, Tianhui Zhang, Rishika Mundada, Lianhui Qin, et al. Toolgym: An open-world tool-using environment for scalable agent testing and data curation. *arXiv preprint arXiv:2601.06328*, 2026. URL https://arxiv.org/abs/2601.06328.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *arXiv preprint arXiv:2404.07972*, 2024. URL https://arxiv.org/abs/2404.07972.

Frank F. Xu et al. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024. URL https://arxiv.org/abs/2412.14161.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, et al. Mobile-agent-v3.5: Multi-platform fundamental gui agents. *arXiv preprint arXiv:2602.16855*, 2026. URL https://arxiv.org/abs/2602.16855.

Weikai Xu, Zhizheng Jiang, Yuxuan Liu, Pengzhi Gao, Wei Liu, Jian Luan, Yuanchun Li, Yunxin Liu, Bin Wang, and Bo An. Mobile-bench-v2: A more realistic and comprehensive benchmark for vlm-based mobile agents. *arXiv preprint arXiv:2505.11891*, 2025a. URL https://arxiv.org/abs/2505.11891.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*, 2025b. URL https://arxiv.org/abs/2509.18119.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. Androidlab: Training and systematic benchmarking of android autonomous agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 2144–2166, 2025c.

John Yang, Kilian Lieret, Jeffrey Ma, Parth Thakkar, Dmitrii Pedchenko, Sten Sootla, Emily McMilin, Pengcheng Yin, et al. Programbench: Can language models rebuild programs from scratch? *arXiv preprint arXiv:2605.03546*, 2026a. URL https://arxiv.org/abs/2605.03546.

Pei Yang, Hai Ci, and Mike Zheng Shou. macosworld: A multilingual interactive benchmark for gui agents. *arXiv preprint arXiv:2506.04135*, 2025. URL https://arxiv.org/abs/2506.04135.

Yuhao Yang, Tianyu Fan, and Chao Huang. Cli-anything: Towards agent-native computer use. *arXiv preprint arXiv:2606.03854*, 2026b. URL https://arxiv.org/abs/2606.03854.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*, 2022. URL https://arxiv.org/abs/2207.01206.

Shunyu Yao et al. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. URL https://arxiv.org/abs/2406.12045.

Abhay Zala, Jaemin Cho, Han Lin, Jaehong Yoon, and Mohit Bansal. Envgen: Generating and adapting environments via llms for training embodied agents. *arXiv preprint arXiv:2403.12014*, 2024. URL https://arxiv.org/abs/2403.12014.

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

Zhexin Zhang et al. Agent-safetybench: Evaluating the safety of llm agents. *arXiv preprint arXiv:2412.14470*, 2024. URL https://arxiv.org/abs/2412.14470.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. Infiniteweb: Scalable web environment synthesis for gui agent training. *arXiv preprint arXiv:2601.04126*, 2026. URL https://arxiv.org/abs/2601.04126.

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

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

Liya Zhu, Jingzhe Ding, Jian Zhang, Jianbo Xue, Shihao Liang, Ge Zhang, Xiang Gao, Qingshui Gu, et al. Workflow-gym: Towards long-horizon evaluation of computer-use agentic tasks in real-world professional fields. *arXiv preprint arXiv:2606.11042*, 2026. URL https://arxiv.org/abs/2606.11042.