CUA-LITE: AN OPEN PLATFORM FOR COMPUTER USE AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Open resources for computer use agents (CUAs) — environments, datasets, and models — remain scattered: without a standard, these resources cannot be integrated and scaled; additionally, evaluation, RL, and SFT cannot share one infrastructure. We present **CUA-Lite**, an open platform for developing CUAs, built around three standardized abstractions: **(1) Unified environment interface**, under which we integrate 12 benchmarks across desktop, browser, and mobile, plus lightweight VM-free sandboxes that replicate OSWorld's desktop at much lower cost, providing over 30,000 verifiable tasks. **(2) Unified supervised data format**, into which we convert the major public CUA datasets, from GUI grounding and understanding to full rollouts, and record our fresh rollouts from frontier CUAs on these verifiable environments. **(3) Model harnesses shared across evaluation, RL, and SFT**, one per model and implemented for 14 model families. Each harness runs its model in these computer environments, producing the rollouts that evaluation and RL consume, or renders stored supervised data into the model's own training format for SFT. We release reference scores across the integrated benchmarks, and show that RL and SFT through the same harness improve open-weight models. We release our dataset, code, and models, and the platform is open to contributions.

1 INTRODUCTION

Computer use agents (CUAs) operate desktop, browser, and mobile applications in computer environments. Developing CUAs requires environments with verifiable tasks, supervised data, and models, each with a harness that runs the model in an environment. Open resources exist for all three: environments on desktop (Xie et al., 2024; Yuan et al., 2026; Bonatti et al., 2025; Li et al., 2025; Xie et al., 2025; Wang et al., 2026; Lv et al., 2026; Aggarwal et al., 2026), browser (Zhou et al., 2024; Koh et al., 2024; Liu et al., 2018; He et al., 2024; Bai et al., 2026), and mobile (Rawles et al., 2025; Xu et al., 2025a; Kong et al., 2026; Wu et al., 2026); supervised datasets, from GUI grounding labels to full rollouts (Deng et al., 2023; Xu et al., 2025b; Wang et al., 2025c; Mu et al., 2025); and models, from specialist CUAs (Qin et al., 2025; Wang et al., 2025c; Liu et al., 2026a; Xue et al., 2026) and vision–language models (VLMs) (Bai et al., 2025b;a) to API models with their own native computer use tools.

However, these resources remain scattered: each environment has its own interface and runtime (e.g., a virtual machine (VM), a browser, a mobile emulator); each supervised dataset its own format; each model its own action space, prompt format, and rollout code. Without a standard, connecting models to environments means rewriting the same code: an action mapping, context management, and a rollout loop. Therefore, open resources cannot be integrated and scaled; additionally, evaluation, reinforcement learning (RL), and supervised finetuning (SFT) cannot share one infrastructure.

To address this gap, we present CUA-Lite, an open platform for developing CUAs built around three standardized abstractions (Figure 1): CUA-Lite unifies environments under one interface, *Lite.Gym*; supervised data under one format, *Lite.Sample*; and adapts any model to the environment interface and the data format through the model's own harness.

(1) Unified environment interface. Lite.Gym (§2.1) exposes an environment as a Gym-style `reset`, `step`, and `close` loop (Towers et al., 2025). The loop standardizes one *observation format*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

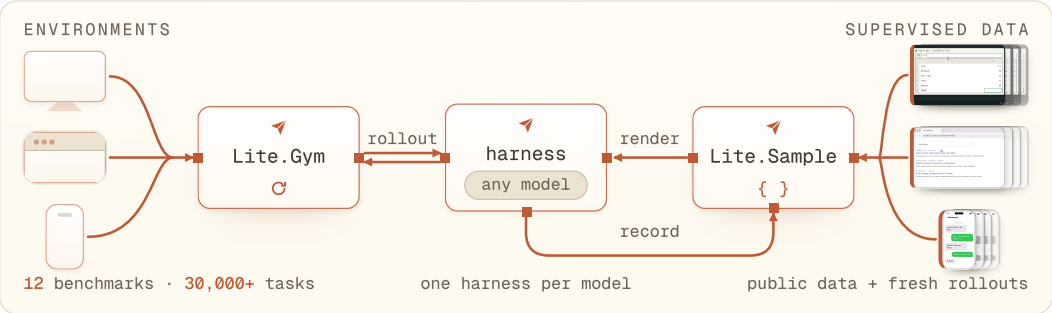


Figure 1: **CUA-Lite overview.** One environment interface, Lite.Gym, serves every integrated environment, regardless of its runtime. One supervised data format, Lite.Sample, stores all supervised data, converted from public datasets or recorded through rollouts. One harness per model connects it to both, so evaluation, RL, and SFT share the same code (§2.3).

across environments: a screenshot, text, or both. It also standardizes one *GUI action space* per platform, issued as tool calls (e.g., `click` and `type` on desktop and browser, or `tap` and `swipe` on mobile). Each environment can add *extra tools* of its own (e.g., `bash` in our desktop sandboxes). The loop wraps the environment's own runtime: a VM, a browser, or a mobile emulator. We integrate 12 benchmarks across desktop, browser, and mobile, and add VM-free desktop sandboxes of our own. These Docker containers replicate OSWorld's desktop at much lower cost and provide over 30,000 verifiable tasks for training (§2.1.2).

(2) Unified supervised data format. Lite.Sample (§2.2) stores any supervised data, from GUI grounding and understanding to full rollouts, in one standardized format: multi-turn messages that interleave the model's tool calls with the environment's observations. We convert 9 major public CUA datasets (Deng et al., 2023; Xu et al., 2025b; Wang et al., 2025c; Mu et al., 2025) and record fresh rollouts from frontier CUAs.

(3) Model harnesses shared across evaluation, RL, and SFT. Each model has one harness (§2.3), the code that adapts the model to the environment interface and the data format. Through its harness, a model runs in every integrated environment, and its rollouts serve both evaluation and RL. The same harness renders any stored Lite.Sample, including other models' rollouts, into the model's own training format for SFT. Harnesses exist for 14 model families, including open-weight models (UI-TARS, Qwen, and others) and API models (GPT, Claude, and Gemini).

We release reference scores for 15 models on the 12 integrated benchmarks (§3). We further demonstrate how the rollout data and the verifiable tasks of CUA-Lite improve an open-weight computer use agent, by SFTing Qwen3.5-4B on GPT-5.5 rollouts and then training it with GRPO on our verifiable tasks (§4). We release our dataset, code, and models, and welcome contributions.

2 The CUA-Lite Platform

In this section we introduce the three standardized abstractions: the environment interface Lite.Gym (§2.1), under which we integrate existing benchmarks and our VM-free desktop sandboxes with verifiable tasks; the supervised data format Lite.Sample (§2.2), into which we convert public supervised datasets and record fresh rollouts; and the per-model harness (§2.3) that runs a model against both. Appendix A reviews related work.

2.1 Environment Interface: Lite.Gym

Lite.Gym exposes every integrated environment through one interface (Figure 2), a Gym-style loop (Towers et al., 2025) that standardizes one observation format across environments and one GUI action space per platform, plus optional extra tools per environment (Table 1).

GUI actions as tool calls. CUA-Lite standardizes the GUI actions of each platform and exposes them through one tool per platform: `click`, `type`, and `key` through `computer` tool on desktop and

```python
env = gym.make("osworld@chrome_030", max_steps=30)   # any integrated environment
obs = await env.reset()                              # obs.image, obs.text

# one step: three GUI actions in one call, one paired result
step = await env.step([make_tool_call("computer", {"actions": [
    {"action": "click", "coordinate": [412, 87]},
    {"action": "type",  "text": "example.com"},
    {"action": "key",   "keys": ["enter"]}]}, call_id="call_0")])
# step.results (one per call id), step.reward, step.terminated, step.truncated

# end to end: the whole rollout, driven by the model's harness
agent  = agents.make("gpt-5.5", env=env)             # resolves the harness
sample = await agent.sample(env)                     # the rollout, as a Lite.Sample
```

Figure 2: **One step and one rollout through the environment interface.** Three GUI actions in one `computer` call return one result per `call_id`, so action batching costs one round trip.

Table 1: **GUI actions and extra tools per platform.** See Appendix B for details.

Platform	Tool and actions
GUI actions	
Desktop, Browser	`computer`: `click`, `mouse_move`, `mouse_down`, `mouse_up`, `drag`, `scroll`, `type`, `key`, `key_down`, `key_up`, `hold_key`, `wait`, `screenshot`, `cursor_position`
Mobile	`mobile`: `tap`, `long_press`, `swipe`, `drag`, `pinch`, `type`, `system_button`
Grounding	`point`, `bbox`: return a coordinate or a box
Extra tools	
All platforms	`response`, `terminate`: finish the task
Desktop	`bash`: run a shell command
Browser	`goto`, `back`, `forward`: navigate
	`new_tab`, `switch_tab`, `close_tab`: manage tabs
Mobile	`open_app`: launch an application
	`ask_user`: ask the user a question

browser; `tap`, `swipe`, and `system_button` through `mobile` tool on mobile (Table 1). The model's harness maps its raw actions into that space (§2.3), so a model built for one action vocabulary runs on any integrated environment. Each tool call may carry a sequence of actions, so a model can plan several steps in one turn with *action batching* (Figure 2). A GUI action's coordinates are normalized to $[0, 1000]$ on every platform.

Extra tools per environment. Beyond the GUI actions, an environment optionally exposes tools of its own: `bash` on Linux desktops, `goto` and `back` on browser, and `open_app` on mobile (Table 1). A model issues these tools alongside the platform's GUI actions.

Observations as tool results. A GUI action returns a screenshot, plus any text the environment emits. An extra tool returns its own result, such as a shell command's output.

2.1.1 Benchmarks and Training Environments

We integrate 12 benchmarks and 6 training environments with verifiable tasks across desktop, browser, and mobile (Table 2). Most environments run unchanged on their own runtimes, with two kinds of exception. Three benchmarks run on third-party implementations: WebVoyager on WebHarbor's offline mirrors of its 15 websites (WebHarbor Team and Contributors, 2026), and WebArena and MiniWoB++ on BrowserGym (Le Sellier de Chezelles et al., 2025) (Appendix C). The four `Lite.*` environments run on our VM-free desktop sandboxes, described in §2.1.2.

2.1.2 VM-Free Desktop Sandboxes (`Lite.*`)

Most existing desktop environments, such as OSWorld, are distributed as virtual machines. Each instance needs KVM, which shared and cloud hosts often do not expose, and carries a full guest

Table 2: **The benchmarks and training environments in CUA-Lite.** Each `Lite.*` environment runs the tasks of its source (OSWorld (Xie et al., 2024), SCALECUA (Lv et al., 2026), CUA-Gym (Wang et al., 2026), or Gym-Anything (Aggarwal et al., 2026)) on our VM-free desktop sandboxes; every other environment runs on its own runtime.

Benchmark	#Eval tasks	Benchmark	#Eval tasks
Grounding		**Browser**	
ScreenSpot-Pro (2025)	1,581	WebVoyager (2024)	643
OSWorld-G (2025)	564	WebArena (2024)	812
Desktop		MiniWoB++ (2018)	125
OSWorld (2024)	369	**Mobile**	
OSWorld 2.1 (2026)	108	AndroidWorld (2025)	116
WindowsAgentArena (2025)	154	AndroidLab (2025a)	138
		MobileWorld (2026)	201
		MobileGym (2026)	256
Training environment	**#Train tasks**	**Training environment**	**#Train tasks**
Desktop		**Browser**	
Lite.OSWorld	2,429	WebGym (2026)	292,092
Lite.ScaleCUA (2026)	20,289	**Mobile**	
Lite.CUAGym (2026)	10,910	MobileGym (2026)	160
Lite.CUAWorld (2026)	2,479		

	OSWorld	Lite.OSWorld
Runtime	QEMU/KVM VM	Docker container
Host	`/dev/kvm`, nested virt.	any Docker host
Disk	31 GB	6.8 GB
Memory	3.8 GB	0.6 GB
Cold start	29.3 s	21.3 s
CPU when idle	5.7 %	1.4 %

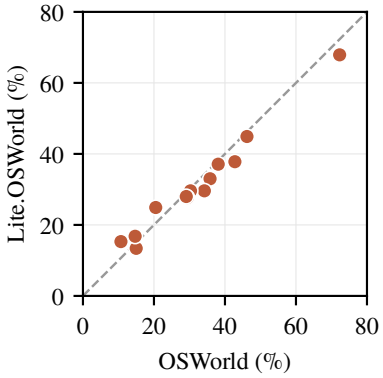


Figure 3: **Lite.OSWorld against OSWorld.** Left: cost of one instance, median over 20 cold starts with both runtimes allocated 4 GB and 4 cores; memory (resident usage within that allocation) and CPU are cgroup counters over a 300 s window with no action sent (Appendix D). Right: evaluation scores (%) of 13 models. The dashed line marks equal scores. See Table D.1 for per-model scores.

OS, which makes it heavy to run and slow to boot. **We propose VM-free desktop sandboxes (`Lite.*`), containerized Linux desktops that replicate OSWorld's desktop at much lower cost.** The official OSWorld VM ships as a disk image with no build recipe, so we reverse-engineer it. We recover the applications installed in the VM and their configuration, and rebuild the desktop from them as `Dockerfile`s. Our rebuilt OSWorld sandbox, Lite.OSWorld, keeps OSWorld's 369 evaluation tasks, setups, and verifiers unchanged.

Cost. A VM-free sandbox carries no guest OS and shares the host kernel, so it runs on a sixth of OSWorld's memory and boots in three quarters of the time (Figure 3, left).

Fidelity. Lite.OSWorld yields evaluation scores close to OSWorld's. We run 13 models, 12 open-weight and GPT-5.5, on OSWorld (Xie et al., 2024) and on Lite.OSWorld, over the same 332 tasks (OSWorld's 369 minus 29 infeasible tasks and 8 that need a Google account) and with identical setup, verifiers, harnesses, and seed (Figure 3, right). Scores differ by 2.7 points on average and by 5.0 at worst, on EvoCUA-8B, and the two rankings agree with a Spearman correlation of 0.98 across all 13 models (Table D.1).

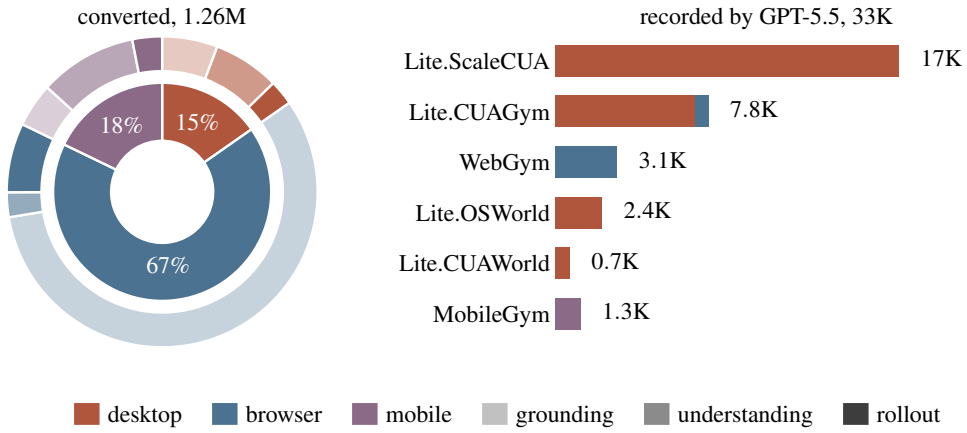


Figure 4: **The stored Lite.Samples.** Left: the 9 converted datasets, by platform (inner ring) and by task type within each platform (outer ring). Right: GPT-5.5's rollouts with verified reward on our training environments, by platform; Appendix E.3 adds those of Qwen3.5-27B and Qwen3.8-27B.

Training tasks. Beyond the evaluation tasks they inherit, our sandboxes provide over 30,000 training tasks with verifiers (Table 2):

- **(1)** Lite.OSWorld adds two training splits to OSWorld's 369 evaluation tasks: 1,722 tasks synthesized from templates and 707 perturbed from 256 of them (Appendix D).
- **(2)** Lite.ScaleCUA runs the 20,289 desktop tasks SCALECUA (Lv et al., 2026) synthesized on that same OSWorld desktop.
- **(3)** Lite.CUAGym runs the 10,910 desktop and browser tasks of CUA-Gym (Wang et al., 2026), across mock sites and real applications.
- **(4)** Lite.CUAWorld runs 2,479 tasks on the 41 professional applications of Gym-Anything (Aggarwal et al., 2026).

2.2 Supervised Data Format: Lite.Sample

A Lite.Sample stores any supervised data (a UI grounding label, UI question understanding, or a full rollout) in one standardized format: multi-turn messages that interleave the model's tool calls with the environment's observations. See Appendix E.1 for the schema.

Converted datasets. We convert 9 major public CUA datasets, covering GUI grounding, understanding, and full rollouts, into 1.26M Lite.Samples (Deng et al., 2023; Lu et al., 2025; Xu et al., 2025b; Wang et al., 2025c; Liu et al., 2026a; Mu et al., 2025) (Figure 4, left). See Appendix E.2 for the datasets and the conversion rules.

Fresh rollouts. We record rollouts of GPT-5.5, Qwen3.5-27B, and Qwen3.8-27B on our training environments and keep those with verified reward, 82K Lite.Samples in all (Figure 4, right). Each recorded step stores the model's `reasoning_content`, an `action_description` of the step in one line of natural language, and the `tool_call` it issues, carrying either a batch of GUI actions or one of the environment's extra tools (§2.1). Appendix E.3 details how we collect them.

2.3 Model Harness Shared Across Evaluation, RL, and SFT

A harness is the scaffolding that makes a model an agent. It turns each observation and the history into the model's input, parses the output back into actions, and drives the loop. CUA-Lite has one harness per model, built around an `adapter` that converts between a Lite.Sample and the model's own input and output. For SFT, the adapter renders a stored Lite.Sample into the model's own format for training (§2.2). During a rollout for evaluation or RL, it renders the Lite.Sample as it grows and converts the model's reply back into Lite.Gym actions. Inside the adapter, `action_space` maps the

Table 3: **Reference evaluation scores (%) on all 12 benchmarks.** Bold: best per benchmark; underline: best open-weight model. WAA: WindowsAgentArena. –: not applicable. Qwen models run with reasoning off and API models at medium effort. See Appendix F for details.

Model	ScreenSpot-Pro	OSWorld-G	OSWorld	OSWorld 2.1	WAA	WebVoyager	WebArena	MiniWoB++	AndroidWorld	AndroidLab	MobileWorld	MobileGym
	Grounding		Desktop			Browser			Mobile			
Specialist CUAs												
UI-TARS-1.5-7B	49.8	60.6	27.5	0.5	16.7	46.2	9.1	18.4	19.8	36.2	11.1	13.3
EvoCUA-8B	43.5	57.3	35.6	0.7	16.6	5.8	10.8	56.0	–	–	–	–
MAI-UI-8B	62.2	70.0	–	–	–	–	–	–	61.2	49.3	29.9	25.8
UI-Venus-2-9B	71.9	78.2	50.5	1.4	41.3	72.9	30.7	59.2	62.1	47.8	59.0	34.4
General VLMs												
Qwen3-VL-4B	58.3	65.5	23.6	0.9	9.5	20.6	8.7	47.2	43.1	36.2	6.8	13.7
Qwen3-VL-8B	55.2	64.3	28.1	0.3	14.5	29.3	7.1	53.6	51.7	39.9	8.5	16.0
Qwen3-VL-32B	57.7	72.0	34.3	1.2	24.0	42.8	14.6	60.8	62.1	36.2	17.9	23.4
Qwen3.5-4B	57.7	62.9	27.0	0.7	11.6	12.2	11.3	28.0	60.3	34.8	11.1	17.6
Qwen3.5-9B	58.1	68.2	38.6	0.4	13.1	35.0	15.4	50.4	62.1	36.2	10.3	18.0
Qwen3.5-27B	69.5	74.3	48.9	2.8	25.4	51.7	21.9	68.8	65.1	40.6	15.4	26.6
Qwen3.8-27B	75.4	79.6	59.2	2.4	37.0	55.4	24.5	64.8	64.7	51.4	32.5	40.2
API models												
Claude Opus 5	85.3	89.2	**78.7**	**63.1**	**72.4**	**90.4**	51.7	**91.2**	81.0	**62.3**	78.6	70.2
GPT-5.5	74.5	90.4	71.1	38.6	66.6	78.4	47.1	84.8	78.4	60.9	80.3	53.9
GPT-5.6 Sol	82.5	91.2	73.3	43.9	70.0	79.8	**53.3**	82.4	75.9	60.9	**86.3**	64.1
GPT-6 Astra	**93.9**	**92.5**	75.1	63.0	63.4	84.3	**53.3**	87.2	**84.5**	60.9	85.5	**80.9**

model's raw actions to and from the standardized GUI actions (§2.1), and `protocol` manages the model's context across turns. Modules are shared and inherited across models (Qwen3.5, Qwen3.8, and EvoCUA all build on Qwen3-VL's), so a new harness implements only what differs. Harnesses currently exist for 14 model families.

3 EVALUATION

In this section we run each model on the benchmarks it supports and release the resulting reference scores (Table 3). *Most previously reported scores come with no runnable evaluation script, so we do not claim to reproduce them; instead, we offer these scores as a reference that can be rerun from the CUA-Lite configuration we release*.

Models. We run 15 models: specialist CUAs (UI-TARS-1.5, EvoCUA, MAI-UI, UI-Venus-2), general VLMs (Qwen3-VL at 4B–32B, Qwen3.5 at 4B–27B, Qwen3.8-27B), and API models (Claude Opus 5, GPT-5.5, GPT-5.6 Sol, GPT-6 Astra). Each runs through its own harness (§2.3), with the settings of Table F.1. By default, open-weight models run with reasoning off and API models at medium reasoning effort. The one exception is UI-Venus-2, which by its official default runs with reasoning on for every task other than grounding.

Benchmarks. We run the 12 benchmarks of Table 2 across grounding, desktop, browser, and mobile. Most scores come from one rollout per task under one fixed seed, with greedy decoding and each benchmark's own step budget. See Appendix F for the tasks we evaluate.

Results. At the same default reasoning effort (medium), GPT-6 Astra and Claude Opus 5 score highest, Astra on grounding and Opus 5 on most desktop and browser benchmarks (Table 3). Among open-weight models, Qwen3.8-27B and UI-Venus-2-9B score highest, and each Qwen family improves with model size on average.

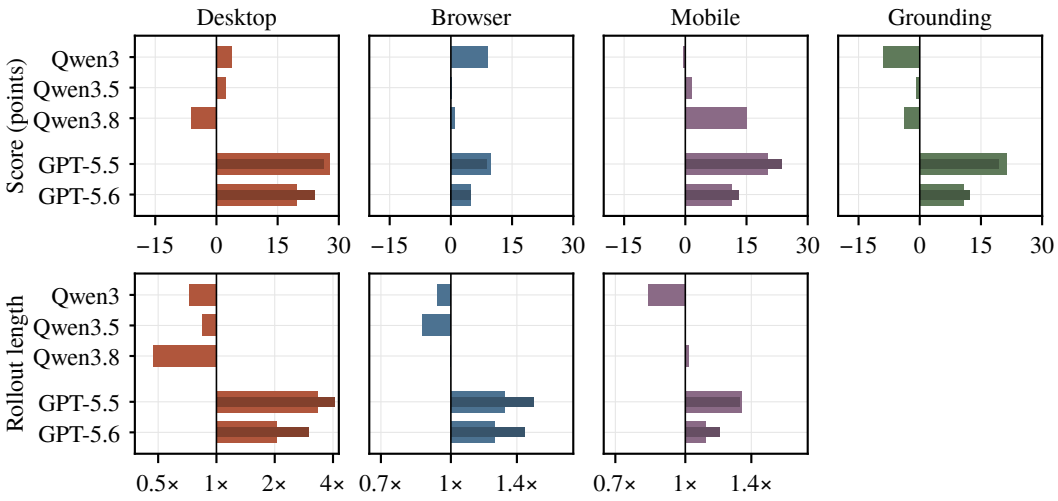


Figure 5: **Effect of reasoning on score and rollout length.** Top: the change in score. Bottom: the ratio of rollout lengths. For Qwen models, each bar compares reasoning on with reasoning off; for GPT models, `medium` (wide bar) or `xhigh` (narrow, darker bar) reasoning effort with `none`. Each platform averages its benchmarks. The Qwen3 and Qwen3.5 rows average Qwen3-VL (4B–32B) and Qwen3.5 (4B–27B), shown per model in Figure F.1.

Reasoning ablation. We rerun Qwen3-VL, Qwen3.5, and Qwen3.8 with reasoning on and off (default), and GPT-5.5 and GPT-5.6 Sol at `none`, `medium` (default), and `xhigh` reasoning effort (Figure 5, Table F.2). **Reasoning helps the GPT models far more than the Qwen models.** The GPT models gain on every platform from `none` to `medium`, and little more at `xhigh`, and their rollouts grow longer as they verify their work before terminating. For the Qwen models, reasoning lowers every model's grounding average and mostly shortens their rollouts, but its effect on their scores on the other platforms varies. For example, Qwen3.8-27B gains 15 points on mobile but loses 6.3 on desktop, where its rollouts are 53% shorter than without reasoning.

4 Training

In this section, we demonstrate how the rollout data and the verifiable tasks of CUA-Lite improve an open-weight computer use agent through SFT and RL. Specifically, we first SFT Qwen3.5-4B on the GPT-5.5 rollouts of §2.2 (§4.1), and then further train the SFTed model with GRPO (Shao et al., 2024) on the verifiable tasks of §2.1.1 (§4.2). Every SFT and RL run uses a single node of 8 A100 GPUs, 96 vCPU, 1.8 TB RAM.

4.1 SFT

Setup. We SFT Qwen3.5-4B on GPT-5.5 rollouts collected from the verifiable training tasks (§2.2) that pass the verifier, and evaluate on held-out tasks. For desktop, we train on 5,000 Lite.Samples from Lite.ScaleCUA and evaluate on Lite.OSWorld's 332 evaluation tasks (OSWorld's 369, excluding the 29 infeasible tasks and the 8 that need a Google account). For browser, we train on about 2,000 Lite.Samples from WebGym and evaluate on 128 held-out WebVoyager tasks. For mobile, we train on about 1,000 Lite.Samples from MobileGym, whose 160 training tasks are templates that each generate many instances, and evaluate on its own 256 held-out tasks. We train Qwen3.5-4B with its official computer use harness (§2.3), with an image history of one screenshot (1920×1080 on desktop, 1280×720 on browser, 720×1600 on mobile) and the full text history. We evaluate every model three times at temperature 0 and report the mean success rate (Appendix G).

Results. SFT on the GPT-5.5 rollouts raises Qwen3.5-4B from 22.6 to 44.8 on desktop, from 4.4 to 60.0 on browser, and from 14.6 to 26.8 on mobile (Table 4). As an ablation, we run SFT on these

Table 4: **Evaluation scores (%) of Qwen3.5-4B before and after SFT on GPT-5.5 rollouts.** The rows without SFT differ by whether Qwen3.5-4B's own reasoning is on, the SFT rows by whether the GPT-5.5 rollouts used for SFT contain reasoning. The rows without SFT use the configuration of SFT (one screenshot) for fair comparison rather than the official configuration of Table 3 (up to four screenshots), so their scores differ from Table 3.

	Desktop Lite.OSWorld	**Browser** WebVoyager	**Mobile** MobileGym
Without SFT			
w/o reasoning	22.6	4.4	14.6
w/ reasoning	27.1	14.8	13.9
SFT on GPT-5.5 rollouts			
w/o reasoning	37.2	48.7	21.4
w/ reasoning	**44.8**	**60.0**	**26.8**

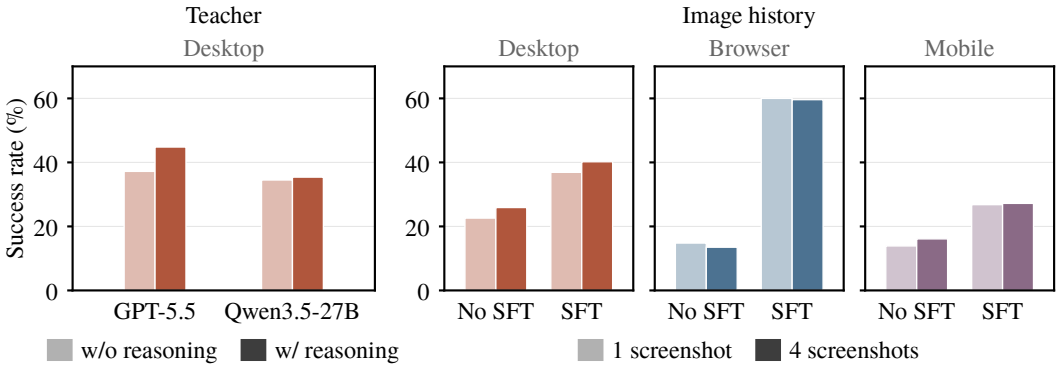


Figure 6: **SFT ablations of Qwen3.5-4B.** Left: desktop evaluation scores (%) after SFT on each teacher's rollouts, with and without reasoning. Right: evaluation scores (%) with the one or four most recent screenshots in context, without SFT and after SFT on GPT-5.5 rollouts, both with reasoning. Desktop screenshots are 1920×1080 on the left and 1280×720 on the right, as in Table G.1.

rollouts without their reasoning (§2.2), which costs 7.6, 11.3, and 5.4 points respectively, while Qwen3.5-4B's own reasoning does not always help before SFT.

Ablations. We also ablate two choices of the SFT setup: the teacher that provides the rollouts and the image history (Table G.1 gives the exact scores).

(1) Teacher (Figure 6, left). We ablate the model whose rollouts Qwen3.5-4B is SFTed on, replacing GPT-5.5 with Qwen3.5-27B on desktop. **GPT-5.5 is the stronger teacher, and most of its lead comes from reasoning**. Without reasoning, Qwen3.5-27B's rollouts train a model close to GPT-5.5's, and adding its reasoning helps little.

(2) Image history (Figure 6, right). We ablate how many of the most recent screenshots Qwen3.5-4B keeps in its context, one or four. **The latest screenshot alone does about as well as four**, both before and after SFT.

4.2 RL

Setup. We train the SFTed Qwen3.5-4B with GRPO on verifiable tasks (§2.1.1) and evaluate on held-out tasks. For desktop, we train on the 814 LibreOffice tasks (Impress, Calc, Writer) that Lite.OSWorld synthesizes, and evaluate on the 117 LibreOffice tasks of the official OSWorld evaluation set. For browser, we use the 622 WebVoyager tasks that do not modify their site, on WebHarbor's offline mirrors, and train on 494 of them and evaluate on the other 128, the held-out tasks of §4.1. For mobile, we drop the hardest level of MobileGym's evaluation split and group the rest into 37 task families. We train on one task from each family and evaluate on the 52 remaining tasks

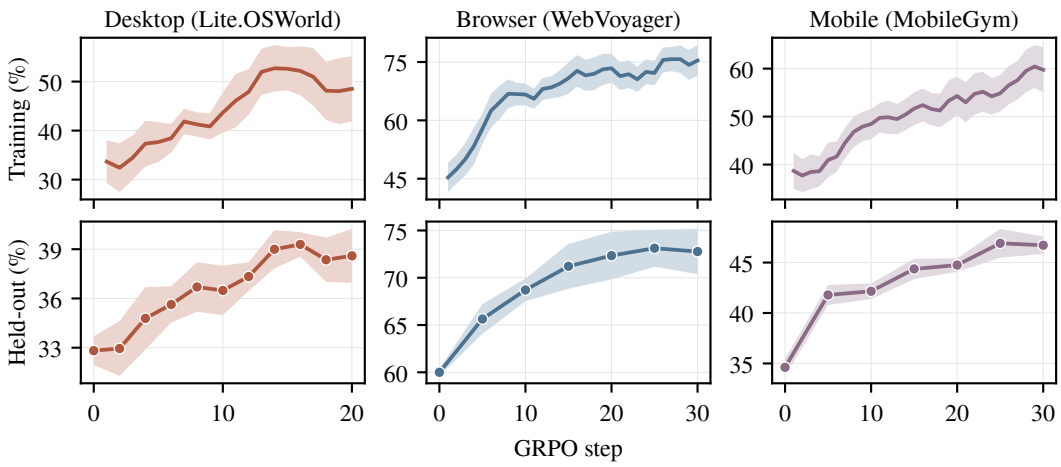


Figure 7: **GRPO from the SFTed Qwen3.5-4B.** Top: training reward per GRPO step, smoothed over five GRPO steps. Bottom: held-out evaluation score. Both rows show the mean and standard error over five seeds. Mobile evaluates on a subset of Table 4's held-out tasks, and desktop on OSWorld's LibreOffice tasks.

of those families, with a shaped reward that gives partial credit for passed goal checks. We run five seeds per platform (Appendix G).

Results. GRPO further improves the SFTed Qwen3.5-4B on every platform (Figure 7). On held-out tasks, the evaluation score rises from 32.8 to 38.6 on desktop, the success rate from 60.0 to 72.8 on browser, and the shaped reward from 34.6 to 46.7 on mobile.

Choosing training tasks. The obvious training tasks do not always suit RL. On desktop, before the runs in Figure 7, we tried GRPO on the official Lite.ScaleCUA RL split but found it did not raise the held-out score. We hypothesize that SFT on GPT-5.5's Lite.ScaleCUA rollouts had already saturated the model on those tasks. The desktop runs in Figure 7 therefore use Lite.OSWorld's synthesized LibreOffice tasks, one application suite so that a run fits on one node. On browser, training on live websites is prone to IP blocking, so we instead train on WebHarbor's offline mirrors, using the WebVoyager tasks outside our held-out set. On mobile, MobileGym's official training split is much easier than its evaluation split, so we draw our training and held-out tasks from it.

5 Limitations and Conclusion

Limitations. CUA-Lite standardizes how environments, data, and models connect, but it also inherits their limitations. Open mobile environments with verifiable training tasks are scarce, so MobileGym is our only mobile training environment. WebVoyager runs on offline mirrors of its websites, so its scores do not measure the live sites. We do not fully audit the training verifiers, and a model judge scores some tasks in Lite.CUAGym and Lite.CUAWorld, so SFT data and RL rewards can include rollouts that did not solve their task. Our VM-free desktop sandboxes cover only Linux desktops, so Windows environments such as WindowsAgentArena still need a virtual machine, and mobile environments such as AndroidWorld still need an emulator. In §4, the desktop and mobile models are evaluated on the applications they train on, so their gains do not show transfer to unseen applications. Most previously reported scores come with no runnable evaluation script, so we do not claim to reproduce them and offer ours only as reference scores. Our harnesses implement only each model's own GUI loop, not richer harnesses such as Codex and Claude Code.

Conclusion. CUA-Lite gives computer use agent development one standard: an environment interface, a supervised data format, and per-model harnesses shared across evaluation, RL, and SFT. On that standard we integrate 12 benchmarks and VM-free desktop sandboxes with over 30,000 verifiable tasks, convert the major public CUA datasets, and record fresh rollouts. CUA-Lite stays open to contributions, and every number in this paper ships with the command that produces it.

Reproducibility Statement

Appendix C describes the benchmark implementations, Appendix D the desktop sandboxes and their training tasks, Appendix E the data format and datasets, Appendix F the evaluation settings of every model and benchmark, and Appendix G the training hyperparameters. The code, configurations, data, and models are already open source; we omit the link to preserve anonymity.

Ethics Statement

Autonomous CUAs can take consequential actions; all rollouts in this work run inside isolated container sandboxes with no access to real accounts. Converted datasets are redistributed under their original licenses, recorded in the dataset card we publish for each (Appendix E.2), and the applications inside our desktop sandboxes are open-source or licensed for redistribution. We use API models under their providers' terms of service.

AI Use Statement

We used generative AI tools to assist with writing, coding, and preparing figures, and the authors reviewed and verified all AI-assisted content. We did not use generative AI tools for research ideation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

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.

Ahmed Awadallah, Yash Lara, Raghav Magazine, Hussein Mozannar, Akshay Nambi, Yash Pandya, Aravind Rajeswaran, Corby Rosset, Alexey Taymanov, Vibhav Vineet, et al. Fara-7B: An efficient agentic model for computer use. *arXiv preprint arXiv:2511.19663*, 2025. URL https://arxiv.org/abs/2511.19663.

Hao Bai, Alexey Taymanov, Tong Zhang, Aviral Kumar, and Spencer Whitehead. WebGym: Scaling training environments for long-horizon visual web agents with realistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows Agent Arena: Evaluating multi-modal OS agents at scale. In *International Conference on Machine Learning (ICML)*, 2025.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. GUICourse: From general vision language model to versatile GUI agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. doi: 10.18653/v1/2025.acl-long.1065.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. doi: 10.52202/075280-1220.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.371.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.50.

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. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026. doi: 10.18653/v1/2026.acl-long.278.

Hanyu Lai, Xiao Liu, Yanxiao Zhao, Han Xu, Hanchen Zhang, Bohao Jing, Yanyu Ren, Shuntian Yao, Yuxiao Dong, and Jie Tang. ComputerRL: Scaling end-to-end online reinforcement learning for computer use agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, et al. The BrowserGym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=5298fKGmv3.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM)*, 2025. doi: 10.1145/3746027.3755688.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, et al. ScaleCUA: Scaling open-source computer use agents with cross-platform data. In *International Conference on Learning Representations (ICLR)*, 2026a.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, et al. GEM: A gym for generalist LLMs. In *International Conference on Learning Representations (ICLR)*, 2026b.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. GUIOdyssey: A comprehensive dataset for cross-app GUI navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. doi: 10.1109/ICCV51701.2025.02080.

Bowen Lv, Xiao Liu, Yanyu Ren, Hanyu Lai, Bohao Jing, Hanchen Zhang, Yanxiao Zhao, Shuntian Yao, Jie Tang, and Yuxiao Dong. SCALECUA: Scaling computer use agents with verifiable task synthesis and efficient online RL. *arXiv preprint arXiv:2607.11185*, 2026. URL https://arxiv.org/abs/2607.11185.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024. doi: 10.52202/079017-2365.

Meta PyTorch and Hugging Face. OpenEnv: Agentic execution environments, 2025. URL https://github.com/huggingface/OpenEnv. Release v0.5.0, accessed 2026-09-24.

Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, et al. GUI-360°: A comprehensive dataset and benchmark for computer-using agents. *arXiv preprint arXiv:2511.04307*, 2025. URL https://arxiv.org/abs/2511.04307.

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.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Fei Tang, Zhiqiong Lu, Boxuan Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. ClawGUI: A unified framework for training, evaluating, and deploying GUI agents. *arXiv preprint arXiv:2604.11784*, 2026. URL https://arxiv.org/abs/2604.11784.

Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. doi: 10.52202/085713-4916.

Daniel Toyama, Philippe Hamel, Anita Gergely, Gheorghe Comanici, Amelia Glaese, Zafarali Ahmed, Tyler Jackson, Shibl Mourad, and Doina Precup. AndroidEnv: A reinforcement learning platform for Android. *arXiv preprint arXiv:2105.13231*, 2021. URL https://arxiv.org/abs/2105.13231.

Venus Team, Zhuohan Cai, Haoxing Chen, Jiaxuan Chen, Weizhi Chen, Changlong Gao, Zhangxuan Gu, Yuan Guo, Yusong Hu, Jianrong Jiang, et al. UI-Venus-2 technical report. *arXiv preprint arXiv:2609.00028*, 2026. URL https://arxiv.org/abs/2609.00028.

Bowen Wang, Dunjie Lu, Junli Wang, Tianyi Bai, Shixuan Liu, Zhipeng Zhang, Haiquan Wang, Hao Hu, Tianbao Xie, Shuai Bai, 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.

Shuai Wang, Weiwen Liu, Jingxuan Chen, Yuqi Zhou, Weinan Gan, Xingshan Zeng, Yuhan Che, Shuai Yu, Xinlong Hao, Kun Shao, et al. GUI agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890*, 2025a. URL https://arxiv.org/abs/2411.04890.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations (ICLR)*, 2025b.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025c. doi: 10.52202/085713-4669.

WebHarbor Team and Contributors. WebHarbor: Docking real websites for evolving GUI agent environments, 2026. URL https://aiming-lab.github.io/webharbor.github.io. Project website.

Dingbang Wu, Rui Hao, Haiyang Wang, Shuzhe Wu, Han Xiao, Zhenghong Li, Bojiang Zhou, Zheng Ju, Zichen Liu, Lue Fan, et al. MobileGym: A verifiable and highly parallel simulation platform for mobile GUI agent research. *arXiv preprint arXiv:2605.26114*, 2026. URL https://arxiv.org/abs/2605.26114.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, et al. AgentGym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *International Conference on Learning Representations (ICLR)*, 2026.

Han Xiao, Guozhi Wang, Yuxiang Chai, Zimu Lu, Weifeng Lin, Hao He, Lue Fan, Liuyang Bian, Rui Hu, Liang Liu, et al. UI-Genie: A self-improving approach for iteratively boosting MLLM-based mobile GUI agents. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. doi: 10.52202/085713-5028.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. doi: 10.52202/079017-1650.

Tianbao Xie, Jiaqi Deng, Xiaochuan Li, Junlin Yang, Haoyuan Wu, Jixuan Chen, Wenjing Hu, Xinyuan Wang, Yuhui Xu, Zekun Wang, et al. Scaling computer-use grounding via user interface decomposition and synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. doi: 10.52202/085713-0817.

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 (Volume 1: Long Papers)*, 2025a. doi: 10.18653/v1/2025.acl-long.107.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous GUI interaction. In *International Conference on Machine Learning (ICML)*, 2025b.

Taofeng Xue, Chong Peng, Mianqiu Huang, Linsen Guo, Tiancheng Han, Haozhe Wang, Jianing Wang, Xiaocheng Zhang, Xin Yang, Dengchang Zhao, et al. EvoCUA: Evolving computer use agents via learning from scalable synthetic experience. *arXiv preprint arXiv:2601.15876*, 2026. URL https://arxiv.org/abs/2601.15876.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, et al. OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026. URL https://arxiv.org/abs/2606.29537.

Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, et al. AgentCPM-GUI: Building mobile-use agents with reinforcement fine-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2025. doi: 10.18653/v1/2025.emnlp-demos.12.

Longtao Zheng, Zhiyuan Huang, Zhenghai Xue, Xinrun Wang, Bo An, and Shuicheng Yan. AgentStudio: A toolkit for building general virtual agents. In *International Conference on Learning Representations (ICLR)*, 2025.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

A Related Work

Environments for CUAs span desktop (Xie et al., 2024; Yuan et al., 2026; Bonatti et al., 2025; Li et al., 2025; Xie et al., 2025; Wang et al., 2026; Lv et al., 2026; Aggarwal et al., 2026), browser (Liu et al., 2018; Zhou et al., 2024; Koh et al., 2024; He et al., 2024; Bai et al., 2026), and mobile (Toyama et al., 2021; Rawles et al., 2025; Xu et al., 2025a; Kong et al., 2026; Wu et al., 2026); each defines its own action space and runtime, and most desktop benchmarks are distributed as virtual machines. **Datasets** range from GUI grounding and understanding labels to full rollouts, each in its own format (Deng et al., 2023; Lu et al., 2025; Xu et al., 2025b; Wang et al., 2025c; Liu et al., 2026a; Mu et al., 2025). **Models** range from specialist CUAs (Qin et al., 2025; Wang et al., 2025c; Liu et al., 2026a; Zhou et al., 2025; Xue et al., 2026; Awadallah et al., 2025; Venus Team et al., 2026) to VLMs (Bai et al., 2025b;a) and API models with native computer use tools, each with its own action space and prompt format (Wang et al., 2025a). **Systems and infrastructure.** OpenHands targets software engineering (Wang et al., 2025b), BrowserGym and AgentLab the evaluation of web agents (Le Sellier de Chezelles et al., 2025), and ClawGUI the training and deployment of mobile agents (Tang et al., 2026); AgentStudio is a toolkit of environments, annotation tools, and benchmarks for virtual agents (Zheng et al., 2025), and AgentBoard focuses on analytical evaluation (Ma et al., 2024). GEM and AgentGym-RL provide gym-style interfaces for text agents (Liu et al., 2026b; Xi et al., 2026); ComputerRL, SCALECUA, CUA-Gym, MobileGym, and WebGym scale environments or verified tasks for one platform (Lai et al., 2026; Lv et al., 2026; Wang et al., 2026; Wu et al., 2026; Bai et al., 2026); OpenEnv standardizes environment packaging (Meta PyTorch & Hugging Face, 2025). Each of these systems covers one platform, or one of the three abstractions. CUA-Lite provides all three across desktop, browser, and mobile, and one harness per model serves evaluation, RL, and SFT alike.

B Environment Interface (Lite.Gym)

Table B.1 lists every GUI action with its arguments, and Table B.2 the extra tools, their arguments, and the environments that expose them.

Table B.1: **GUI actions and their arguments.** Coordinates are normalized to $[0, 1000]$ (§2.1).

Action	Arguments
computer (desktop, browser)	
`click`, `mouse_down`, `mouse_up`	`coordinate`, `button`, `clicks`$\in \{1, 2, 3\}$
`mouse_move`	`coordinate`
`drag`	`start_coordinate`, `coordinate`, `button`
`scroll`	`direction`, `amount`, `coordinate`
`type`	`text`, `press_enter`
`key`, `key_down`, `key_up`, `hold_key`	`keys`, `duration`
`wait`	`duration`
`screenshot`, `cursor_position`	none
mobile (mobile)	
`tap`, `long_press`	`coordinate`, `clicks`, `duration`
`swipe`, `drag`, `pinch`	`start_coordinate`, `coordinate`, `direction`
`type`	`text`
`system_button`	`button`
`wait`	`duration`
`screenshot`	none
Grounding	
`point`, `bbox`	`coordinate`, `box`

C Benchmark Implementations

We wrap each benchmark's official implementation, with its tasks, runtime, and evaluator, except for three browser benchmarks that run on a third-party implementation.

Table B.2: **Extra tools per platform.**

Tool	Arguments	Environments
All platforms		
`response`	`text`	every non-grounding environment
`terminate`	`status`, `reason`	every non-grounding environment
Desktop		
`bash`	`command`	Lite.OSWorld, Lite.ScaleCUA, Lite.CUAGym, Lite.CUAWorld
Browser		
`goto`	`url`	WebGym, WebArena, MiniWoB++
`back`	none	WebGym, WebArena, MiniWoB++, WebVoyager
`forward`, `new_tab`, `close_tab`	none	WebArena, MiniWoB++
`switch_tab`	`index`	WebArena, MiniWoB++
Mobile		
`open_app`	`app_name`	AndroidWorld, AndroidLab, MobileWorld, MobileGym
`ask_user`	`question`	MobileWorld

Table D.1: **Evaluation scores (%) on OSWorld and Lite.OSWorld.** Both runtimes run in one parity campaign with the same settings, separate from the reference run of Table 3.

Model	OSWorld	Lite.OSWorld	Δ
GPT-5.5	72.3	67.9	−4.4
Qwen3.5-27B	46.2	44.9	−1.3
EvoCUA-8B	42.8	37.8	−5.0
Qwen3.5-9B	38.1	37.1	−1.0
Qwen3-VL-32B	35.8	33.0	−2.8
Qwen3-VL-8B	34.2	29.6	−4.6
Qwen3.5-4B	30.3	29.6	−0.7
UI-TARS-1.5-7B	29.4	27.7	−1.7
Qwen3-VL-4B	29.1	28.0	−1.1
Qwen3-VL-2B	20.5	24.9	+4.4
ScaleCUA-7B	15.0	13.4	−1.6
UI-TARS-7B-DPO	14.7	16.8	+2.1
Qwen3.5-2B	10.7	15.3	+4.6

WebVoyager. WebVoyager's tasks run on WebHarbor's offline mirrors of its 15 websites (WebHarbor Team and Contributors, 2026), served from one container whose databases reset when it starts, so its scores measure the mirrors rather than the live sites. A GPT-4.1 judge scores the last three screenshots and the final answer of each rollout.

WebArena and MiniWoB++. Both run on their BrowserGym implementations (Le Sellier de Chezelles et al., 2025), WebArena against its self-hosted website containers and MiniWoB++ against locally served pages. BrowserGym's fuzzy-match judge for WebArena runs on GPT-4.1 instead of the retired GPT-4 Turbo model it hard-codes.

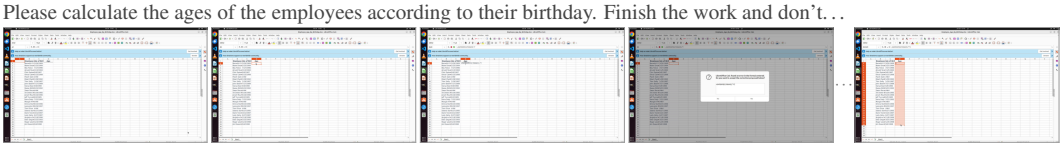
D VM-Free Desktop Sandboxes

Fidelity. Table D.1 gives the per-model scores behind Figure 3 (right), comparing OSWorld and Lite.OSWorld on the same 332 tasks (OSWorld's 369, excluding the 29 infeasible tasks and the 8 that need a Google account; Appendix F) under the same setup, verifiers, and harness. Figure D.1 shows four of those tasks (in LibreOffice Calc, Chrome, VLC, and LibreOffice Writer) step by step on both runtimes.

Cost. Figure 3 (left) measures one instance of each runtime under a matched allocation. OSWorld distributes its VM with 4 GB of guest memory and 4 cores, so we cap the Lite.OSWorld container at the same 4 GB and 4 cores for measurement.

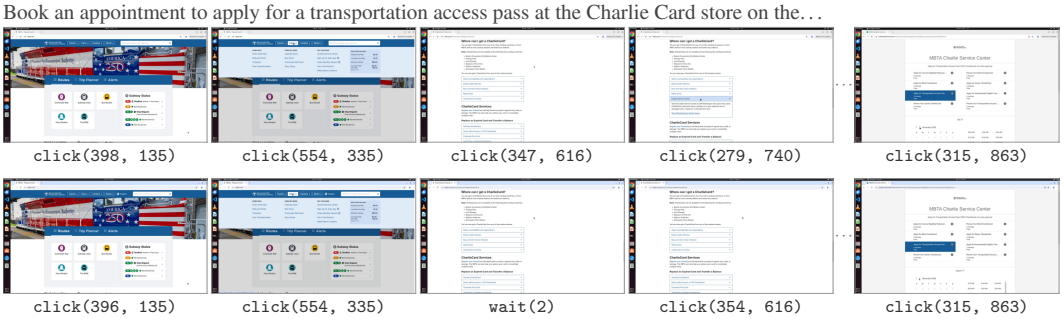
LibreOffice Calc

Please calculate the ages of the employees according to their birthday. Finish the work and don't...



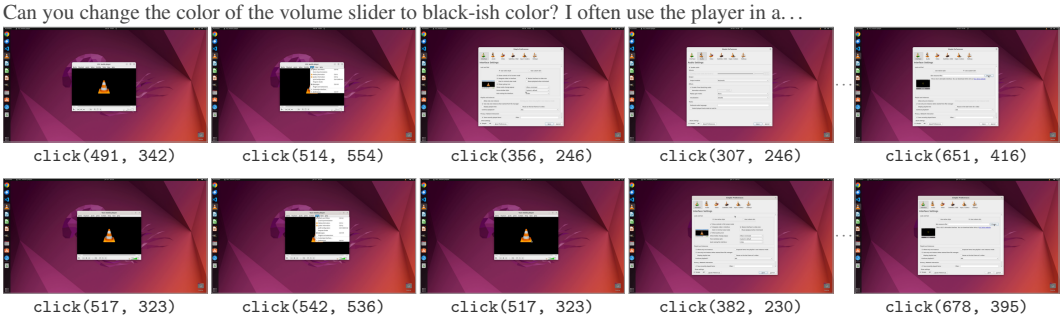
Chrome

Book an appointment to apply for a transportation access pass at the Charlie Card store on the...



VLC

Can you change the color of the volume slider to black-ish color? I often use the player in a...



LibreOffice Writer

Please help me make the first letter of each word to uppercase.

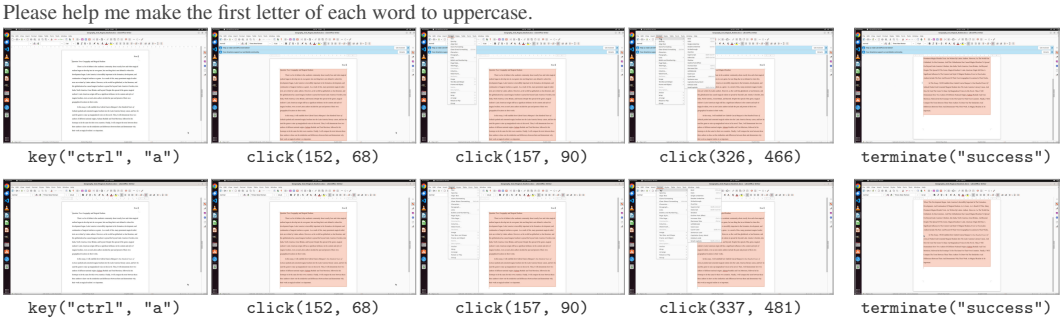


Figure D.1: **Four tasks, step by step, on both runtimes.** Each block is one task, the OSWorld VM on the upper row and the Lite.OSWorld container on the lower. A row gives a rollout's first four steps and its last, with $\cdots$ for the steps between and, under each panel, the action GPT-5.5 took from that screen.

- **Cold start**: the wall time from `gym.make` to the first observation `reset` returns, which covers the boot, the task's setup, and the first screenshot. Setup differs from task to task, so we alternate two `os`-domain tasks rather than repeat one, and both runtimes run that same pair, which leaves setup equal on the two sides. We report the median of 20 cold starts per runtime, interleaved so that host drift reaches both.
- **Memory**: resident memory, read from the instance's cgroup as `memory.current` minus `inactive_file`, at the end of a 300 s idle window that opens 60 s after `reset` returns, with no action sent. Both numbers are usage rather than the cap. A container takes what it touches, and a VM holds the guest RAM its host must provide.
- **Idle CPU**: the cgroup's cumulative CPU time (`cpu.stat`'s `usage_usec`) differenced across that same window and expressed as a share of one core, rather than a sampled rate, because one desktop repaint moves a sampled percentage by two orders of magnitude.
- **Disk**: the image size, plus OSWorld's `qcow2`.

Training tasks. Lite.ScaleCUA, Lite.CUAGym, and Lite.CUAWorld inherit the task sets of their upstream releases, with the bugs we found fixed. Lite.OSWorld adds two training splits to OSWorld's 369 evaluation tasks (Table D.2):

- **Synthesized** (1,722 tasks): we generate each task from a template, and the templates of each application cover the operations of its OSWorld evaluation tasks. The tasks do not use OSWorld's files. Most generate their own in the sandbox or copy in public ones (e.g., Wikimedia Commons photos and Project Gutenberg books), and the rest start from the application alone. Only the Thunderbird tasks reuse OSWorld's mail profile.
- **Perturbed** (707 tasks): we perturb 256 of OSWorld's evaluation tasks, keeping each task's files and setup and changing the request.

Both splits mostly reuse OSWorld's evaluator functions as verifiers, and we check that each task's oracle solution scores 1. We train on the synthesized tasks (§4.2). The perturbed tasks start from the same state as evaluation tasks, so they are useful for debugging and analysis; we do not train on them and do not recommend it, as it would overfit to OSWorld.

Table D.2: **Lite.OSWorld tasks per application.**

	Training		
Application	Synthesized	Perturbed	**Evaluation**
Chrome	132	91	46
GIMP	73	42	26
LibreOffice Calc	278	106	47
LibreOffice Impress	287	108	47
LibreOffice Writer	249	43	23
Multiple apps	379	138	101
OS	147	34	24
Thunderbird	48	32	15
VLC	37	44	17
VS Code	92	69	23
Total	1,722	707	369

E DATASETS (LITE.SAMPLE)

E.1 FORMAT

A Lite.Sample stores supervised data (a UI grounding label, UI question understanding, or a full rollout) in three fields: the images it refers to, the conversation, and the metadata that labels it (Figure E.1). Each message is tagged with its role (user, assistant, or tool), each content part with its kind (text, image, or action description), and the reasoning has its own `reasoning_content` field. Because every part is tagged, a harness can find a part by its type instead of parsing text. For example, it can drop the reasoning, keep only the most recent screenshots, or replace past steps with their action descriptions. Figure E.2 shows one Lite.Sample of each kind.

```python
@dataclass
class LiteSample:                    # one label, answer, or rollout
    images: list[Image]              # what the messages refer to, deduplicated
    messages: list[LiteMessage]      # user, assistant and tool turns
    metadata: LiteCUAMetadata

class LiteCUAMetadata:
    dims: tuple[Platform, TaskType]  # what it holds, and where it runs
    others: dict                     # provenance: the model, the rollout's reward

Platform = "desktop" | "browser" | "mobile"
TaskType = "grounding.action" | "grounding.point" | "grounding.bbox"
           | "understanding" | "use"

class LiteUserMessage(TypedDict):    # the instruction, and the observation
    content: list[TextContent | ImageContent]

class LiteAssistantMessage(TypedDict):
    tool_calls: list[LiteToolCall]   # Lite.Gym actions, coordinates in [0, 1000]
    reasoning_content: str           # a native reasoning channel, if any
    content: list[TextContent                # a question, an answer
                  | ActionDescriptionContent  # the action in one line
                  | InlineReasoningContent    # reasoning in the visible output
                  | HistorySummaryContent]    # a running summary of the rollout

class LiteToolMessage(TypedDict):    # what those actions returned
    tool_call_id: str
    content: list[TextContent | ImageContent]
```

Figure E.1: **The Lite.Sample types.**

E.2 CONVERTED DATASETS

We convert 9 public datasets into Lite.Samples, across desktop, browser, and mobile (Table E.1). One converter per dataset reads the upstream release and emits Lite.Samples. Coordinates are rescaled to $[0, 1000]$, actions are rebuilt in the Lite.Gym action set so that key names share one vocabulary, and images are deduplicated by content. We drop, rather than repair, any Lite.Sample whose image is missing or whose coordinates fall outside that range; for a `use` rollout we drop every step of the rollout. Each dataset's upstream release and license are recorded in the dataset card we publish.

E.3 FRESH ROLLOUTS

We run each model through its own harness (§2.3) on our training environments and keep the rollouts with verified reward (Table E.2). A post-processing pass then runs over the raw logs. It strips the no-op actions `screenshot` and `wait`, and records in `metadata.others.exclude_reason` why a rollout might be excluded, for example because it never finished or the agent installed software. It

Table E.1: **The 9 converted datasets, in Lite.Samples by platform.** Grounding sums the `action`, `point`, and `bbox` sub-types.

Dataset	Platform	Grounding	Understanding	Rollout	Total
Aguvis (2025b)	Desktop	260	0	0	260
	Browser	634,314	0	3,729	638,043
	Mobile	57,498	0	17,820	75,318
ScaleCUA (2026a)	Browser	69,246	32,445	80,800	182,491
GUI-360 (2025)	Desktop	72,415	86,730	10,482	169,627
GUIOdyssey (2025)	Mobile	0	125,197	8,330	133,527
GUIAct (2025)	Browser	14,379	0	5,497	19,876
	Mobile	0	0	7,438	7,438
OpenCUA (2025c)	Desktop	0	0	22,336	22,336
UI-Genie-Agent (2025)	Mobile	0	0	4,705	4,705
CAGUI (2025)	Mobile	0	2,715	600	3,315
Multimodal-Mind2Web (2023)	Browser	0	0	597	597
Total		848,112	247,087	162,334	1,257,533

Table E.2: **Rollouts with verified reward per teacher, in Lite.Samples.** –: not recorded.

Training environment	Platform	GPT-5.5	Qwen3.5-27B	Qwen3.8-27B	Total
Lite.ScaleCUA	Desktop	17,493	17,539	17,807	52,839
Lite.CUAGym	Desktop, browser	7,793	–	10,052	17,845
Lite.OSWorld	Desktop	2,374	–	2,410	4,784
WebGym	Browser	3,143	967	–	4,110
MobileGym	Mobile	1,279	–	–	1,279
Lite.CUAWorld	Desktop	728	–	–	728
Total		32,810	18,506	30,269	81,585

drops a step outright when its coordinates fall outside $[0, 1000]$, its action is not in the Lite.Gym action set, or its call names a tool the environment did not expose.

Every recorded step carries the model's tool calls and an `action_description`. Whether it also carries reasoning, and where that reasoning comes from, differs by model:

- **Qwen3.5-27B**: reasoning on, so it writes `reasoning_content` natively.
- **Qwen3.8-27B**: reasoning off, so it carries no reasoning.
- **GPT-5.5**: its reasoning is prompted, not native. We prompt it for one `Thought:` line per step and store that line as `reasoning_content`.

Every published rollout with reasoning therefore carries it in `reasoning_content`.

Grounding: a question about a screenshot; the label is the tool call that answers it.

```
{"images": ["images/9c/9c4e...7b.png"],
 "metadata": {"dims": ["desktop", "grounding.point"]},
 "messages": [
   {"role": "user",
    "content": [
      {"type": "image", "index": 0},
      {"type": "text", "text": "Where is the save button?"}]},
   {"role": "assistant",
    "tool_calls": [{"id": "call_0", "type": "function",
      "function": {"name": "point", "arguments": {"coordinate": [850, 120]}}}]}]}
```

Understanding: a question about a screenshot, answered in text, with no tool call.

```
{"images": ["images/1b/1b77...3d.png"],
 "metadata": {"dims": ["mobile", "understanding"]},
 "messages": [
   {"role": "user",
    "content": [
      {"type": "image", "index": 0},
      {"type": "text", "text": "What does this screen show?"}]},
   {"role": "assistant",
    "content": [{"type": "text", "text": "The music player, paused."}]}]}
```

Rollout (`use`): assistant and tool turns alternate until the model answers.

```
{"images": ["images/3f/3fa9...c1.png", "images/b2/b20d...4e.png", "..."],
 "metadata": {"dims": ["desktop", "use"], "others": {"model_id": "gpt-5.5"}},
 "messages": [
   # the instruction, and the screen it starts on
   {"role": "user",
    "content": [
      {"type": "text", "text": "Set the title font to Arial."},
      {"type": "image", "index": 0}]},

   # the model reasons, describes what it will do, and acts
   {"role": "assistant",
    "reasoning_content": "The title is selected, so the font box applies to it.",
    "content": [{"type": "action_description", "text": "Open the font menu."}],
    "tool_calls": [{"id": "call_0", "type": "function",
      "function": {"name": "computer", "arguments": {"actions": [
        {"action": "click", "coordinate": [412, 87]}]}}}]},

   # the screen that action produced
   {"role": "tool", "tool_call_id": "call_0",
    "content": [{"type": "image", "index": 1}]},

   # the next step
   {"role": "assistant",
    "reasoning_content": "The menu is open and Arial is visible in the list.",
    "content": [{"type": "action_description", "text": "Choose Arial."}],
    "tool_calls": [{"id": "call_1", "type": "function",
      "function": {"name": "computer", "arguments": {"actions": [
        {"action": "click", "coordinate": [388, 214]}]}}}]},
   {"role": "tool", "tool_call_id": "call_1",
    "content": [{"type": "image", "index": 2}]},

   # many more assistant and tool turns, omitted here
   "...",

   # the answer that ends the rollout
   {"role": "assistant",
    "content": [{"type": "text", "text": "Done."}]}]}
```

Figure E.2: **One Lite.Sample per task type.**

F Evaluation Details

Tasks evaluated. We evaluate every task of ScreenSpot-Pro, WebVoyager, MiniWoB++, AndroidWorld, AndroidLab, and MobileGym, and a subset of the other six benchmarks (Table 2 gives the full counts):

- **OSWorld-G:** 510 of 564, without the 54 infeasible tasks, whose target element is absent.
- **OSWorld:** 332 of 369, without the 29 infeasible tasks and the 8 that need a Google account.
- **OSWorld 2.1:** 99 of 108, without the 7 tasks that need a simulated user, the one other task that needs a GitLab server, and the multi-phase task.
- **WindowsAgentArena:** 138 of the 154 tasks with task context, without the 13 infeasible tasks and the 3 whose evaluator already passes on the setup state.
- **WebArena:** 241 of 812, the first task of each of its 241 intent templates.
- **MobileWorld:** 117 of 201, without the 40 tasks that need MCP tools and the 44 that need a simulated user to answer the agent.

Harness settings. Each model runs through the harness of its own release (§2.3). The harness sets how many screenshots and how much text history the model keeps, and the environment sets the screenshot resolution (Table F.1). Reasoning is off for every open-weight model except UI-Venus-2, whose official harness turns it on for every task but grounding. In §4, Qwen3.5-4B instead keeps only the latest screenshot and runs mobile at a downsampled 720×1600, so its scores before SFT differ from Table 3.

Table F.1: **Evaluation settings of Table 3.** Top: per model, the most recent screenshots and the text history kept in context, the same on every platform. An action summary (e.g., 'Previous Actions: ') keeps earlier action descriptions in the context. When a fifth screenshot arrives, Qwen3.5 and Qwen3.8 drop the four they keep and start again from it, so they keep one to four. Bottom: per benchmark, the screenshot resolution.

Model	Screenshots	Text history
Specialist CUAs		
UI-TARS-1.5	5	full
EvoCUA	4	action summary
MAI-UI	3	full
UI-Venus-2	3	full
General VLMs		
Qwen3-VL	4	action summary
Qwen3.5, Qwen3.8	1–4	full
API models		
Claude Opus 5	all	full
GPT models	all	full

Platform	Benchmark	Resolution
Grounding	ScreenSpot-Pro, OSWorld-G	each image's own
Desktop	OSWorld, OSWorld 2.1	1920×1080
	WindowsAgentArena	1280×800
Browser	WebVoyager, WebArena	1280×720
	MiniWoB++	500×320
Mobile	AndroidWorld, MobileWorld	1080×2400
	MobileGym	1080×2400
	AndroidLab	740×1600

Reasoning ablation. Figure F.1 is Figure 5 with every model on its own row, and Table F.2 gives the score of each model at each reasoning setting we evaluate.

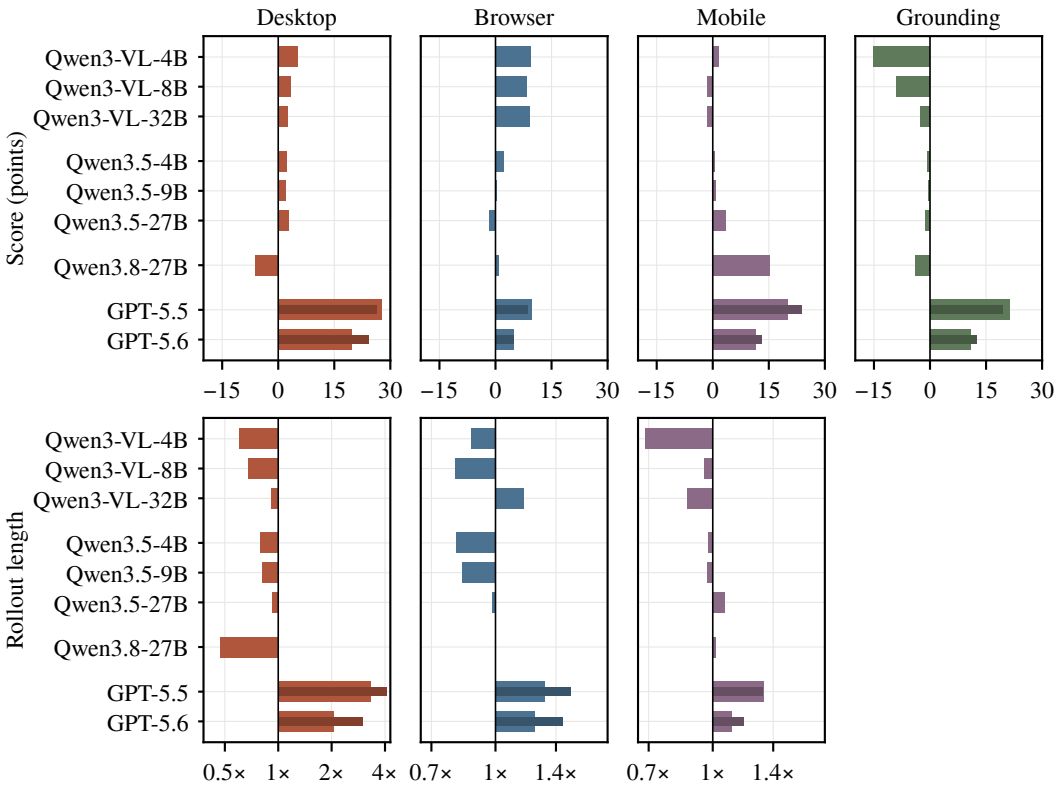


Figure F.1: **Effect of reasoning on score and rollout length, per model.** Figure 5 with Qwen3 (Qwen3-VL) and Qwen3.5 split into their sizes.

Table F.2: **Reference evaluation scores (%) of the 15 models at each reasoning setting we evaluate.** A row marked ↪ reruns the model above it at another reasoning setting: reasoning on for an open-weight model, and the named reasoning effort for an API model. Qwen3-VL's reasoning rows run Qwen3-VL-Thinking, a separate checkpoint, instead of Qwen3-VL-Instruct. Qwen3.5 and Qwen3.8 switch reasoning within one checkpoint. WAA: WindowsAgentArena. –: not applicable. ○: not run, because WebArena rollouts are slow. Rows marked (default), which Table 3 reports, run open-weight models with reasoning off and API models at medium effort.

Model	ScreenSpot-Pro	OSWorld-G	OSWorld	OSWorld 2.1	WAA	WebVoyager	WebArena	MiniWoB++	AndroidWorld	AndroidLab	MobileWorld	MobileGym
	Grounding		Desktop			Browser			Mobile			
Specialist CUAs												
UI-TARS-1.5-7B	49.8	60.6	27.5	0.5	16.7	46.2	9.1	18.4	19.8	36.2	11.1	13.3
EvoCUA-8B	43.5	57.3	35.6	0.7	16.6	5.8	10.8	56.0	–	–	–	–
MAI-UI-8B	62.2	70.0	–	–	–	–	–	–	61.2	49.3	29.9	25.8
UI-Venus-2-9B	71.9	78.2	50.5	1.4	41.3	72.9	30.7	59.2	62.1	47.8	59.0	34.4
General VLMs												
Qwen3-VL-4B	58.3	65.5	23.6	0.9	9.5	20.6	8.7	47.2	43.1	36.2	6.8	13.7
↪ reasoning	40.5	53.1	29.8	1.2	18.8	37.2	○	49.6	40.1	36.2	12.8	17.2
Qwen3-VL-8B	55.2	64.3	28.1	0.3	14.5	29.3	7.1	53.6	51.7	39.9	8.5	16.0
↪ reasoning	44.3	57.1	32.4	0.4	20.2	40.3	○	59.2	46.6	39.1	7.7	17.2
Qwen3-VL-32B	57.7	72.0	34.3	1.2	24.0	42.8	14.6	60.8	62.1	36.2	17.9	23.4
↪ reasoning	56.4	68.2	41.5	0.8	24.6	50.7	○	71.2	59.5	42.8	13.7	17.2
Qwen3.5-4B	57.7	62.9	27.0	0.7	11.6	12.2	11.3	28.0	60.3	34.8	11.1	17.6
↪ reasoning	57.4	61.8	29.6	0.8	16.0	16.6	○	28.0	57.8	35.5	16.2	16.8
Qwen3.5-9B	58.1	68.2	38.6	0.4	13.1	35.0	15.4	50.4	62.1	36.2	10.3	18.0
↪ reasoning	55.3	70.2	36.6	0.5	20.9	36.7	○	49.6	59.1	37.7	16.2	17.2
Qwen3.5-27B	69.5	74.3	48.9	2.8	25.4	51.7	21.9	68.8	65.1	40.6	15.4	26.6
↪ reasoning	65.9	75.5	48.2	3.2	34.0	48.1	○	68.8	66.4	44.9	21.4	28.5
Qwen3.8-27B	75.4	79.6	59.2	2.4	37.0	55.4	24.5	64.8	64.7	51.4	32.5	40.2
↪ reasoning (xhigh)	73.1	74.1	44.5	0.5	34.8	62.2	○	60.0	73.3	54.3	63.2	58.6
API models												
Claude Opus 5	85.3	89.2	78.7	63.1	72.4	90.4	51.7	91.2	81.0	62.3	78.6	70.2
GPT-5.5 (none)	45.2	77.3	48.1	5.8	39.2	70.9	○	72.8	62.5	52.9	42.7	34.8
↪ medium (default)	74.5	90.4	71.1	38.6	66.6	78.4	47.1	84.8	78.4	60.9	80.3	53.9
↪ xhigh	71.1	90.2	74.8	33.2	64.3	76.5	○	84.8	77.6	63.0	84.6	62.5
GPT-5.6 Sol (none)	72.4	79.8	60.3	17.0	50.7	75.7	○	76.8	78.4	56.5	60.7	46.1
↪ medium (default)	82.5	91.2	73.3	43.9	70.0	79.8	53.3	82.4	75.9	60.9	86.3	64.1
↪ xhigh	84.2	92.7	74.8	54.0	71.6	79.2	○	83.2	78.4	63.8	82.9	68.8
GPT-6 Astra	93.9	92.5	75.1	63.0	63.4	84.3	53.3	87.2	84.5	60.9	85.5	80.9

G TRAINING DETAILS

SFT. Every SFT run in §4.1 uses the same hyperparameters on every platform: two epochs over the training set, a global batch of 32 rollouts, and the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1) with a learning rate of 5×10^{-6}, linear warmup over the first 10% of steps, and cosine decay to 10^{-6}. Unless an ablation varies them, runs keep one screenshot (1920×1080 on desktop, 1280×720 on browser, 720×1600 on mobile) and the full text history. The loss is averaged over the model's output tokens. We evaluate the checkpoint after the second epoch three times at temperature 0 on the held-out tasks, and report the mean evaluation score.

Table G.1: **Evaluation scores (%) of Qwen3.5-4B in every setting of §4.1.** The table extends Table 4 with the data of Figure 6. Screenshots: the number kept in context. –: not run.

		Desktop Lite.OSWorld		**Browser** WebVoyager	**Mobile** MobileGym
	Screenshots	1920×1080	1280×720	1280×720	720×1600
Without SFT					
w/o reasoning	1	22.6	14.9	4.4	14.6
	4	–	25.0	8.3	15.2
w/ reasoning	1	27.1	22.6	14.8	13.9
	4	–	25.9	13.5	16.1
SFT on GPT-5.5 rollouts					
w/o reasoning	1	37.2	–	48.7	21.4
w/ reasoning	1	44.8	36.9	60.0	26.8
	4	–	40.2	59.6	27.2
SFT on Qwen3.5-27B rollouts					
w/o reasoning	1	34.5	–	–	–
w/ reasoning	1	35.4	–	–	–

GRPO. Every GRPO run in §4.2 starts from the checkpoint SFTed on GPT-5.5 rollouts with reasoning and one screenshot, and uses the same hyperparameters on every platform: 16 training tasks per GRPO step, 8 rollouts per task at temperature 1, 8 gradient updates per GRPO step, clip ratios of 0.2 and 0.28, no KL penalty, and the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.98$, weight decay 0.1) with a constant learning rate of 10^{-6}. The platforms differ in training length, reward, and evaluation:

- **Desktop**: 20 GRPO steps with 24 concurrent environments. The reward is the Lite.OSWorld verifier's score, which gives partial credit on some tasks. Every two GRPO steps, we evaluate each held-out task four times at temperature 1.
- **Browser**: 30 GRPO steps with 24 concurrent environments. The reward is 1 if WebVoyager's GPT-4.1 judge accepts the rollout and 0 otherwise. Every five GRPO steps, we evaluate each held-out task once at temperature 0.
- **Mobile**: 30 GRPO steps with 32 concurrent environments. The reward is 1 if the task succeeds, and otherwise half the fraction of its goal checks that pass. Every five GRPO steps, we evaluate each held-out task four times at temperature 1. A task family is the tasks of one app whose names share a prefix, at levels L1–L3; we drop L4, the hardest level, which takes 39% of the step budget but succeeds on only 4% of its tasks.