MacArena: Benchmarking Computer Use Agents on an Online macOS Environment

Anonymous Authors[1]

Abstract

Computer-use agents (CUAs) operate graphical user interfaces (GUIs) through vision and control primitives, and their capabilities have advanced rapidly, driven in part by standardized online evaluation benchmarks such as OSWorld, which serve both as evaluation tools and as training environments for reinforcement learning. However, macOS remains underserved in this landscape: the only existing benchmark, macOSWorld, covers a narrow slice of first-party applications with simpler tasks, and runs on x86 virtual machines incompatible with Apple Silicon. We introduce MacArena, a benchmark of 421 manually verified tasks spanning 50 applications that combines a curated port of OSWorld tasks, content sourced from macOSWorld, and 49 new macOS-native tasks, all running on Apple's native Virtualization framework on Apple Silicon. We argue that macOS presents distinct GUI challenges beyond what Linux-based benchmarks capture, and our evaluation supports this claim: strong model performance on existing benchmarks can reflect familiarity with task distributions rather than genuine cross-platform GUI competence. Notably, model rankings invert between ported and macOS-native tasks, with a leading model trailing by over 26% on the MacArena subset, suggesting that macOS poses a genuinely harder environment for current GUI agents[1].

1. Introduction

Computer-use agents (CUAs) are systems capable of interacting with graphical user interfaces (GUIs) through direct manipulation — clicking, dragging, and typing on visible on-screen elements (Sager et al., 2026; Nguyen et al., 2025). CUAs operate directly on the pixel-level representations that human users see, enabling them to navigate applications, complete multi-step tasks, and respond to dynamic interface states (Zheng et al., 2024). Their ability to generalize across diverse software environments without requiring programmatic access to underlying systems positions them as a promising direction toward general-purpose digital assistants capable of executing real-world computer tasks on behalf of users.

Benchmarking CUA capabilities has been an active research area, with several interactive evaluation environments proposed in recent years. OSWorld (Xie et al., 2024) established the leading cross-platform benchmark, spanning Linux and Windows, using real applications and executable tasks. Yet, macOS remains underserved as an evaluation target. The only existing macOS benchmark, macOSWorld (Yang et al., 2025), covers a narrow slice of the platform's task space: UI navigation sequences are simpler, and task specifications are less ambiguous than those found in cross-platform benchmarks. Coverage is also limited almost exclusively to built-in applications, leaving a wide range of commonly used third-party software unevaluated — a significant gap given how central such software is to real-world macOS usage. Furthermore, macOSWorld relies on x86-based virtual machines, making it incompatible with the entire modern Apple Silicon lineup and unable to reflect the performance characteristics of current hardware. This raises a broader question: whether macOS presents distinct challenges for GUI agents that go beyond what existing Linux-based benchmarks capture.

To address these gaps, we introduce **MacArena**, a benchmark for evaluating computer-use agents on macOS (Figure 1). MacArena is built from three sources: a curated port of tasks from OSWorld to macOS, a set of tasks sourced from macOSWorld, and 49 novel macOS-specific tasks that increase task complexity and broaden coverage to non-standard applications.

Our main contributions are:

- **A large-scale macOS CUA benchmark** of 421 high-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]Code available online https://anonymous.4open.science/r/macarena-0448

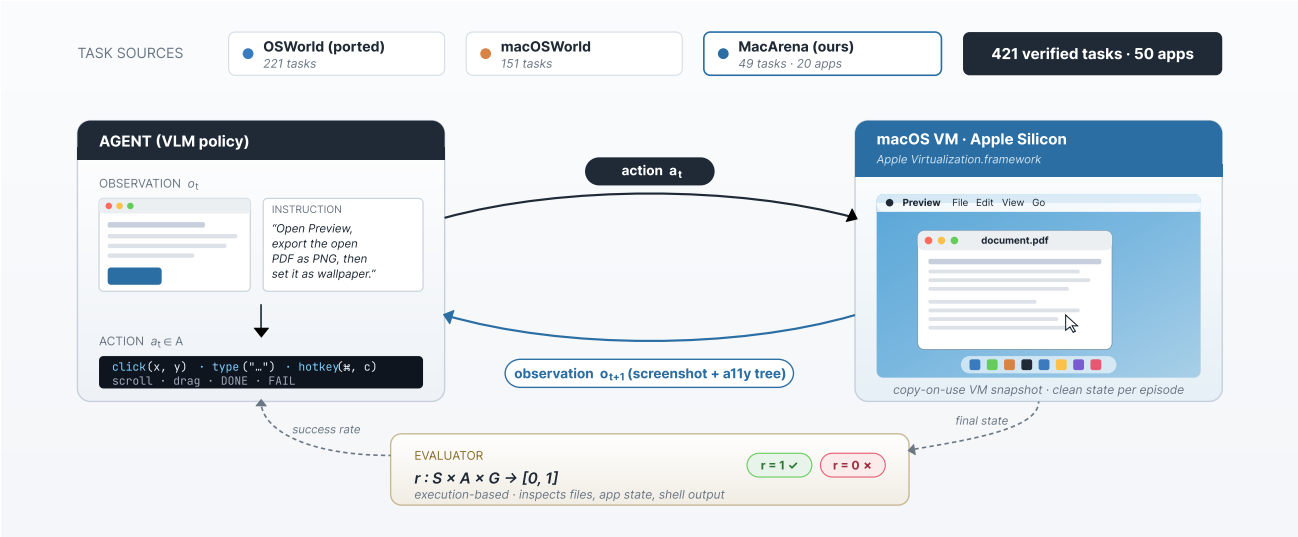

Figure 1. Overview of MacArena. Tasks are drawn from three sources: OSWorld (ported to macOS), macOSWorld, and 49 newly collected macOS-native tasks, totaling 421 human-verified tasks across 50 applications. At each timestep, the agent receives a screenshot and an accessibility tree as observations and produces an action executed within an Apple Silicon VM running via Apple's Virtualization framework. An execution-based evaluator inspects the final environment state to assign a score $r \in [0, 1]$

quality tasks, combining manually ported OSWorld tasks with verified macOSWorld tasks and 49 new macOS-specific tasks into a unified evaluation suite.

- **Human verification of all tasks**, ensuring each task is executable, unambiguous, and correctly specified, providing a higher-quality signal than automated task generation or partial review.
- **Full reproducibility**, with all code publicly released to enable community extension of the benchmark.
- **Evaluation of models**, establishing baseline results and surfacing strengths and limitations of existing CUAs on macOS.

Our evaluation establishes baseline results for current CUAs on macOS and reveals a consistent pattern: performance degrades across all evaluated models relative to Linux, suggesting macOS poses a genuinely harder environment for current GUI agents.

2. Related Work

2.1. GUI Agents and Computer-Use Systems

Early GUI agents were built around prompt-based pipelines that combined frontier vision-language models with modular planning and memory components. Systems such as UFO (Zhang et al., 2025) and SeeAct (Zheng et al., 2024) demonstrated that GPT-4V could complete desktop and web tasks by reasoning over screenshots, while multi-agent frameworks decomposed tasks into subtasks handled by specialized subagents. Though effective in constrained settings, these systems were limited by the capabilities of the underlying models and the overhead of multi-step orchestration.

A second line of work focused on improving visual grounding, the ability to precisely localize UI elements from natural language descriptions. CogAgent (Hong et al., 2024) introduced a dual-encoder architecture trained specifically on GUI layouts, and subsequent models developed dedicated grounding modules that allowed agents to click and interact with greater spatial precision. This grounding capability became a prerequisite for reliable performance on complex desktop tasks.

More recently, end-to-end trained agents have largely superseded prompt-based pipelines. Models such as UI-TARS (Qin et al., 2025) and Aguvis (Xu et al., 2024) are trained natively on large collections of GUI interaction trajectories across desktop, web, and mobile platforms, achieving strong generalization without relying on external orchestration. These single-model agents are simpler to deploy and have become the dominant paradigm for GUI agent development.

Reinforcement learning has emerged as a further lever for improving agent performance. DigiRL (Bai et al., 2024) demonstrated offline-to-online RL fine-tuning on Android tasks, and ComputerRL (Lai et al., 2025) scaled online RL to desktop environments using thousands of parallel virtual machines. UI-TARS-2 (Wang et al., 2025) extended

Table 1. Comparison of CUA benchmarks. ✓ indicates the property is present, ✗ that it is absent, and — that it is not applicable or not reported.

Benchmark	Platform	# Tasks	# Apps	3rd-party Apps	Manual Verif.
Offline benchmarks					
Mind2Web (Deng et al., 2023)	Web	2,350	137	—	✗
AITW (Rawles et al., 2023)	Android	30k	159	—	✗
ScreenSpot (Cheng et al., 2024)	Multi	1,200	—	✓	✓
ScreenSpot-V2 (Wu et al., 2024)	Multi	1,272	—	✓	✓
ScreenSpot-Pro (Li et al., 2025)	Multi	1,581	23	✓	✓
GUIrilla-Gold (Garkot et al., 2026)	macOS	1,283	219	✓	✓
Online benchmarks					
OSWorld (Xie et al., 2024)	Linux, Win	369	9	✓	✓
WAA (Bonatti et al., 2024)	Windows	154	15	✓	✗
macOSWorld (Yang et al., 2025)	macOS	202	30	✗	✗
WebArena (Zhou et al., 2024)	Web	812	—	—	✗
VisualWebArena (Koh et al., 2024)	Web	910	—	—	✗
WorkArena (Drouin et al., 2024)	Web	33	1	—	✗
AndroidWorld (Rawles et al., 2024)	Android	116	20	✓	✗
B-MoCA (Lee et al., 2025)	Android	131	10	✓	✗
MacArena (ours)	**macOS**	**421**	50	✓	✓

this with a multi-turn RL framework that generates training trajectories at scale. Across all of this work, OSWorld and AndroidWorld have served as the primary training and evaluation environments, underscoring the absence of a comparable macOS benchmark for desktop GUI research.

2.2. Computer Use Benchmarks

Before interactive benchmarks, researchers developed offline datasets to evaluate agents on static screenshots. Mind2Web (Deng et al., 2023) and AITW (Rawles et al., 2023) collected large sets of human demonstrations for web and mobile navigation, enabling supervised training of GUI policies. ScreenSpot (Cheng et al., 2024), ScreenSpot-V2 (Wu et al., 2024), and ScreenSpot-Pro (Li et al., 2025) established benchmarks focused specifically on element localization, the ability to map a natural language instruction to the correct location on a screen. GUIrilla (Garkot et al., 2026) extended this to macOS, providing localization annotations across a wide range of third-party applications. While these offline benchmarks remain useful for model development, they do not capture the sequential decision-making, error recovery, and dynamic environment feedback that define real-world agent behavior, and therefore do not measure whether an agent can complete tasks in a live environment.

OSWorld (Xie et al., 2024) is the most comprehensive existing interactive benchmark, covering Linux and Windows with real applications, multi-step tasks, and automated grading based on functional outcomes. Tasks are initialized from virtual machine snapshots. OSWorld has become the standard environment for both training and evaluating desktop GUI agents. For macOS specifically, macOSWorld (Yang et al., 2025) introduced a benchmark targeting Apple's built-in applications, such as Finder, Safari, and Calendar. Yet its tasks tend to be simpler, and more narrowly defined than those in OSWorld, with coverage limited almost entirely to first-party software.

Hardware compatibility compounds this limitation. Following Apple's 2020 transition away from Intel processors, x86-based macOS environments are increasingly misaligned with real-world usage: Apple Silicon now powers the entire Mac lineup, and Intel-based machines are no longer manufactured. Yet macOSWorld was designed around x86 virtual machines with no native Apple Silicon support. While cloud-based evaluation on Apple Silicon hardware (e.g., via EC2 Mac instances) is technically possible, it introduces significant cost overhead that makes large-scale benchmarking and RL training pipelines impractical.

MacArena addresses these gaps directly. macOS presents distinct challenges for GUI agents: from its application conventions and complex window management to the widespread use of third-party software, that existing benchmarks leave largely unexamined. As shown in Table 1, MacArena is the only macOS online benchmark combining third-party application coverage with full manual verification.

3. MacArena Environment

3.1. Problem Formulation

We formalize an autonomous agent interaction in MacArena as a Partially Observable Markov Decision Process (POMDP) (Xie et al., 2024; Bonatti et al., 2024; Yang et al., 2025), defined by the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \Omega, r, \gamma, \mu_0, \mathcal{G}, p_g, \varphi)$, where $\mathcal{S}$ is the full state space of the macOS environment (including hidden system states such as background processes and file system contents), $\mathcal{O}$ is the observation space accessible to the agent (e.g., screenshots and accessibility trees), $\mathcal{A}$ is the action space of mouse and keyboard interactions (full action space is listed in Table 2). $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition function, Ω is the observation function mapping states to observations, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is the reward function, γ is the discount factor, μ_0 is the initial state distribution, $\mathcal{G}$ is the space of task goals (expressed as natural language instructions), p_g is the distribution over goals, and $\varphi : \mathcal{O} \to \mathcal{G}$ is a mapping from observations to goals.

At each timestep t, the agent receives an observation $o_t \in \mathcal{O}$ consisting of a screenshot of the current macOS desktop, optionally with the accessibility tree. Accessibility (a11y) is a structured, hierarchical representation of UI elements (buttons, text fields, menus) exposed by macOS via the Accessibility API, providing element labels, roles, and bounding boxes without requiring visual parsing. Based on this observation, the agent produces an executable action $a_t \in \mathcal{A}$, e.g., click. The action is executed within the virtual machine, transitioning the environment to a new state $s_{t+1} \in \mathcal{S}$ and yielding a new observation $o_{t+1} \in \mathcal{O}$. This loop continues until the agent emits a terminal action (`DONE` or `FAIL`) or the maximum number of steps is reached.

MacArena implements an execution-based reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to [0, 1]$. At the final step, a custom evaluation script compares the resulting environment state to the task objective and assigns a score, with higher values indicating greater task completion.

3.2. Benchmark Structure

MacArena consists of three core components: the environment, the tasks, and the evaluation framework.

Environment. MacArena runs inside virtual machines (VMs) managed by UTM[2], which is built on Apple's native Virtualization framework (see Appendix A for details). We maintain two distinct VMs. The first is dedicated to tasks sourced from OSWorld and macOSWorld; it was configured manually to satisfy the setup assumptions of those benchmarks, including installed applications, system permissions, and verified evaluation conditions. The second VM is purpose-built for MacArena's own tasks and is provisioned entirely via an automated build script. This approach eliminates manual configuration, simplifies migration across macOS versions, and makes it straightforward to add new applications in the future.

Since UTM does not natively support VM snapshot revert, we adopt a copy-on-use strategy: before each task episode, the original VM image is copied to a temporary instance used for the episode and discarded upon completion. This guarantees a clean, reproducible initial state for every evaluation run.

The environment exposes two types of observations to the agent: pixel-level screenshots and, optionally, the macOS accessibility tree taken using `macapptree` (Garkot et al., 2026).

Tasks. MacArena contains 421 tasks organized into two dimensions: application and task type, including cross-application workflows that require coordinating multiple macOS apps. Tasks are sourced from two existing benchmarks: OSWorld and macOSWorld, and supplemented with our own newly collected tasks, all reviewed by humans to ensure each task is executable, unambiguous, and correctly specified.

Each task is defined by three required components. The `instruction` field provides a natural language description of what the agent must accomplish. The `pre_command/config` field specifies an initialization procedure that prepares the VM for the task, such as downloading required files, launching applications, or opening documents. The `evaluator` field defines a deterministic function that programmatically verifies whether the task was completed successfully by inspecting the final environment state.

MacArena supports two task formats inherited from its source benchmarks. The OSWorld format uses a set of predefined evaluation functions composed via a structured configuration file. The macOSWorld format uses shell scripts for both initialization and evaluation, offering greater flexibility for tasks that require more complex or platform-specific logic. Our own tasks follow both formats.

Evaluation Framework. Task success in MacArena is determined by execution-based evaluation: after the agent emits a terminal action, the corresponding evaluator script is executed against the final VM state. Each evaluator returns a value in $[0, 1]$ indicating how well the task was completed; the higher, the better. Evaluation criteria vary by task and may inspect file contents, application state, system properties, or the output of shell commands, depending on what the task requires.

[2] https://github.com/utmapp/UTM

Table 2. Supported actions and their parameters.

Category	Action	Params	Description
Mouse	MOVE_TO	x, y	Move cursor to position
	CLICK	x, y, button (l/r/m)	Click at position
	RIGHT_CLICK	x, y	Right-click at position
	DOUBLE_CLICK	x, y	Double-click at position
	DRAG_TO	x, y	Drag to target position
	SCROLL	dx, dy	Scroll by delta
	MOUSE_DOWN	button	Press and hold mouse button
	MOUSE_UP	button	Release mouse button
Keyboard	TYPING	text	Type a sequence of characters
	PRESS	key	Press a single key
	KEY_DOWN	key	Hold a key down
	KEY_UP	key	Release a held key
	HOTKEY	[keys]	Press a key combination
Terminal	WAIT	—	Sleep until next action
	FAIL	—	Signal task cannot be completed
	DONE	—	Signal task successfully completed

Each of the 49 tasks in the MacArena subset is associated with a unique, hand-crafted evaluation script, yielding 49 distinct evaluation functions in total. This one-to-one correspondence between tasks and evaluators reflects the diversity of verification requirements across macOS applications and task types.

3.3. Benchmark Statistics

MacArena comprises 421 tasks in total, drawn from three sources: 221 tasks adapted from OSWorld, 151 tasks from macOSWorld, and 49 newly collected tasks across 5 categories for the MacArena-specific subset. The MacArena-specific subset spans 20 macOS applications. OSWorld and macOSWorld tasks retain their original application coverage, which includes both macOS-exclusive apps and cross-platform productivity tools.

Tasks are organized into 20 categories reflecting the structure of their source benchmarks. The OSWorld-derived tasks cover seven categories: `chrome`, `gimp`, `libreoffice_calc`, `libreoffice_writer`, `thunderbird`, `vs_code`, and `multi_apps`. The macOSWorld-derived tasks span seven categories: `sys_and_interface`, `sys apps`, `file management`, `productivity`, `media`, `multi apps`, and `advanced`. The MacArena-specific tasks are organized into 5 categories, inheriting macOSWorld names: `file management`, `system and interface`, `advanced apps`, `built-in apps`, and `productivity`, covering tasks that require interaction with a single macOS application at a time.

Figure 2 shows the distribution of tasks across all 20 categories, illustrating the diversity of macOS use cases covered by MacArena.

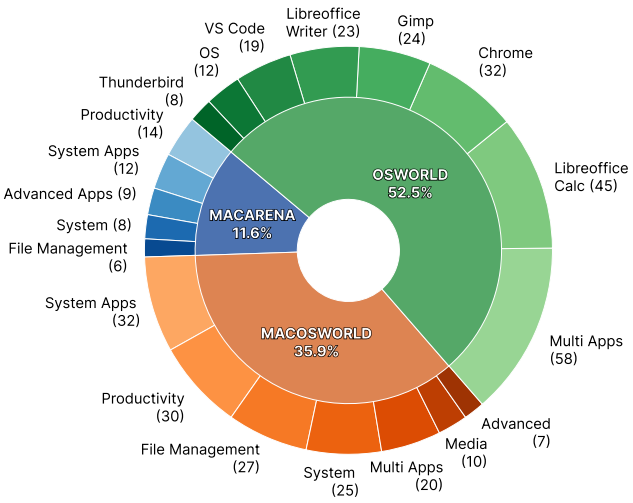

Figure 2. Distribution of tasks per category across the full MacArena benchmark.

Table 3. Success rates (%) of baseline agents on MacArena. Best result in each row is **bold**.

Category	UI-TARS-1.5 7B	Qwen3-VL 2B	Qwen3-VL 4B	OpenAI CUA
OSWorld subset				
Overall	**21.27**	9.95	16.36	16.74
Chrome	28.12	15.62	**37.50**	**37.50**
GIMP	**41.67**	8.33	12.50	4.17
LibreOffice Calc	**6.67**	2.22	4.44	4.44
LibreOffice Writer	17.39	13.04	17.39	**21.74**
Multi-App	0.00	1.72	**3.45**	1.72
OS	**50.00**	16.67	36.36	33.33
Thunderbird	25.00	25.00	25.00	**37.50**
VS Code	**68.42**	31.58	36.84	47.37
macOSWorld subset				
Overall	24.50	15.89	39.74	**52.32**
Advanced	0.00	14.29	14.29	**42.86**
File Management	25.93	7.41	37.04	**40.74**
Media	**40.00**	20.00	30.00	30.00
Multi-App	5.00	10.00	**15.00**	**15.00**
Productivity	26.67	20.00	63.33	**70.00**
Sys & Interface	32.00	32.00	44.00	**60.00**
System Apps	28.12	9.38	40.62	**71.88**
MacArena subset				
Overall	10.20	4.08	12.24	**36.73**
Productivity	7.14	7.14	14.29	**28.57**
Sys & Interface	25.00	0.00	12.50	**37.50**
File Management	0.00	0.00	0.00	0.00
Advanced	11.11	11.11	11.11	**55.56**
System Apps	8.33	0.00	16.67	**50.00**
Overall	21.14	11.40	24.23	**31.83**

4. Experiments

We evaluate four baseline agents on MacArena: UI-TARS-1.5 7B (Qin et al., 2025), Qwen3-VL 2B, Qwen3-VL 4B (Team, 2025), and OpenAI Computer Use Preview (OpenAI, 2025). All agents interact with the macOS virtual machine through raw mouse and keyboard actions. Each agent receives a screenshot of the current desktop at every step. Each task is limited to 15 steps, and each model has 2 runs. Task success is determined by execution-based evaluation scripts as described in Section 3.2. We report Success Rate (SR) as the primary metric, defined as the percentage of tasks for which the evaluation script returns a positive result.

Table 3 reports the success rates of all four agents across the three subsets and 16 task categories. OpenAI Computer Use Preview achieves the highest overall success rate of 31.83%, followed by Qwen3-VL 4B (24.23%), UI-TARS-1.5 7B (21.14%), and Qwen3-VL 2B (11.40%).

4.1. macOS vs. Linux Performance Gap

A key motivation for MacArena is the hypothesis that macOS presents distinct challenges for GUI agents beyond what existing Linux-based benchmarks capture. To investigate this, we compare model performance on the OSWorld subset of MacArena against officially reported scores on the original OSWorld benchmark, which runs on Ubuntu Linux with a 15-step budget. Table 4 presents this comparison for models where official scores are publicly available.

Table 4. Success rates (%) on the original OSWorld benchmark (Ubuntu, 15 steps) vs. the OSWorld subset of MacArena (macOS, 15 steps).

Model	Ubuntu	macOS	Δ
UI-TARS-1.5 7B	24.5	21.27	−3.23
OpenAI CUA	26.0	16.74	−9.26
Qwen3-VL 2B	17.0	9.95	−7.05
Qwen3-VL 4B	26.2	16.36	−9.84

Both models with available reference scores show a meaningful drop when evaluated on macOS, despite the task set being identical. This gap is due to macOS introducing platform-specific differences in application appearance, keyboard shortcuts, window management, and system behavior that models trained primarily on Linux and Windows trajectories are not adapted to. The consistent degradation across both models suggests that macOS poses a genuinely harder environment for current GUI agents.

4.2. Analysis

Multi-app tasks. The multi-app tasks category involves interaction between 2 or more applications. This category remains the hardest category across all models and both the OSWorld and macOSWorld subsets, with most agents scoring at or near 0%. This is consistent with prior work (Xie et al., 2024; Yang et al., 2025) and indicates that coordinating across applications remains an open challenge even for state-of-the-art models.

Category-level strengths and weaknesses. UI-TARS-1.5 7B achieves the highest scores on VS Code (68.42%) and OS tasks (50.00%) within the OSWorld subset, suggesting stronger adaptation to terminal and code editing workflows. In contrast, it performs poorly on macOS-native categories such as Advanced (0.00%) and MacArena (10.2%), indicating limited exposure to macOS-specific application patterns during training. OpenAI CUA dominates the macOSWorld subset, particularly System Apps (71.88%) and Productivity (70.00%), reflecting stronger adaptation to native macOS applications.

Divergence between OSWorld and MacArena subsets. An interesting reversal emerges when comparing performance on the OSWorld subset against the MacArena-specific subset. UI-TARS-1.5 7B outperforms OpenAI CUA on the OSWorld subset (21.27% vs. 16.74%), yet this advantage completely inverts on the MacArena-specific tasks, where OpenAI CUA scores 36.73% against UI-TARS-1.5 7B's 10.2%, a gap of over 26.5 percentage points in the opposite direction. This divergence suggests that strong performance on tasks originally designed for Linux does not transfer to novel macOS-native tasks. UI-TARS-1.5 7B likely benefits from having seen OSWorld-style tasks or similar Linux-based GUI trajectories during training, which gives it an advantage on the OSWorld subset despite the platform shift. However, when faced with genuinely new macOS applications and interaction patterns in the MacArena subset, this advantage disappears entirely. OpenAI CUA, by contrast, appears to have broader macOS-specific knowledge, possibly from training on diverse real-world computer use data that includes macOS. This result highlights an important limitation of evaluating GUI agents exclusively on existing benchmarks: a model can appear competitive by pattern-matching previously seen task structures while failing to generalize to new environments. MacArena's own task subset is specifically designed to surface this gap.

Task difficulty via step consumption. To further characterize the relative difficulty of each subset, we analyze the average number of steps consumed per task by OpenAI CUA, the best-performing model in our evaluation. We report both the average steps across all tasks and the average steps on completed tasks only, as the latter reflects the true complexity of tasks the model was capable of solving. Table 5 summarizes these statistics aggregated by subset.

Table 5. Average steps consumed by OpenAI CUA per subset, across all tasks, and only on completed tasks.

Subset	Avg. Steps (All)	Avg. Steps (Done)
macOSWorld	10.92	8.05
OSWorld	13.88	11.08
MacArena	13.96	12.69

The macOSWorld subset has the lowest average step consumption across all tasks (10.92) and on completed tasks (8.05), indicating that its tasks are, on average, shorter and less complex than those in the other two subsets. This provides a quantitative explanation for why models achieve higher success rates on macOSWorld: beyond any potential platform familiarity, the tasks themselves require fewer interaction steps to complete. The OSWorld subset requires more steps on average (13.88 overall, 11.08 for completed tasks). The MacArena-specific subset exhibits the highest step consumption of all three, both overall (13.96) and on completed tasks (12.69).

5. Limitations and Future Work

Automatic task generation. All tasks in MacArena were manually created by human annotators, which is time-consuming and limits scalability. A promising direction for future work is to automate task generation by leveraging LLMs to synthesize diverse, plausible task instructions, potentially guided by application-specific schemas or interaction logs. However, automatically generated tasks introduce quality concerns: instructions may be ambiguous, infeasible, or trivially easy. Ensuring that generated tasks are valid, non-redundant, and appropriately challenging would therefore require either human validation or automated feasibility checks, for example, by verifying that a reference agent can complete the task within a bounded number of steps. Addressing this pipeline would substantially reduce annotation cost and enable MacArena to scale to a broader range of applications and task types.

Human performance baseline. While tasks in MacArena are 100% human-verified and each task is possible to do, benchmark does not currently include a human performance study. Establishing human baselines would provide an important reference point for interpreting model results and understanding the remaining headroom for future improvement, as done in OSWorld (Xie et al., 2024) and macOSWorld (Yang et al., 2025).

Use of LLMs. Parts of this work were refined using LLMs for better readability and structure.

6. Conclusion

We presented MacArena, a benchmark for evaluating GUI agents on macOS, comprising 421 tasks drawn from OSWorld and macOSWorld, as well as a newly collected set of 49 macOS-native tasks spanning 20 applications. MacArena provides a unified evaluation framework that runs all tasks within a reproducible virtual machine environment using execution-based evaluation scripts, enabling direct comparison of agent behavior across task origins on a single platform.

Our evaluation of agents reveals that macOS remains a challenging and underexplored environment for current GUI agents. All models with available reference scores perform worse on the same task set when evaluated on macOS compared to the original Linux-based OSWorld environment, confirming that the platform gap is real. Furthermore, relative model rankings shift substantially depending on which subset of MacArena is examined: a model that leads on OSWorld-derived tasks can fall far behind on novel macOS-native tasks. This suggests that strong performance on existing benchmarks may reflect familiarity with specific task distributions rather than genuine cross-platform GUI understanding, and that macOS-native tasks are necessary to expose this limitation.

MacArena is designed to serve as a foundation for GUI agents that generalize reliably across operating systems, and to establish macOS as a first-class evaluation target alongside Linux and Windows.

Impact Statement

This paper introduces a benchmark for evaluating computer-use agents on macOS. The primary goal is to advance the scientific study of GUI agents by providing a more comprehensive and reproducible evaluation environment. We do not anticipate direct negative societal consequences specific to this work. Improvements in agent capabilities enabled by better evaluation tools may contribute to automating knowledge work, with socioeconomic implications widely discussed in the AI community. We do not believe these consequences require specific highlighting here beyond what is already widely discussed in the context of autonomous agent research.

References

Bai, H., Zhou, Y., Cemri, M., Pan, J., Suhr, A., Levine, S., and Kumar, A. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning, 2024. URL https://arxiv.org/abs/2406.11896.

Bonatti, R., Zhao, D., Bonacci, F., Dupont, D., Abdali, S., Li, Y., Wagle, J., Koishida, K., Bucker, A., Jang, L., and Hui, Z. Windows agent arena: Evaluating multi-modal os agents at scale, September 2024. URL https://arxiv.org/abs/2409.08264.

Cheng, K., Sun, Q., Chu, Y., Xu, F., Li, Y., Zhang, J., and Wu, Z. Seeclick: Harnessing gui grounding for advanced visual gui agents, 2024. URL https://arxiv.org/abs/2401.10935.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2web: towards a generalist agent for the web. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Drouin, A., Gasse, M., Caccia, M., Laradji, I. H., Del Verme, M., Marty, T., Vazquez, D., Chapados, N., and Lacoste, A. WorkArena: How capable are web agents at solving common knowledge work tasks? In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Garkot, S., Shamrai, M., Synytsia, I., and Hirna, M. Guirilla: A scalable framework for automated desktop ui exploration, 2026. URL https://arxiv.org/abs/2510.16051.

Hong, W., Wang, W., Lv, Q., Xu, J., Yu, W., Ji, J., Wang, Y., Wang, Z., Dong, Y., Ding, M., and Tang, J. Cogagent: A visual language model for gui agents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14281–14290, 2024. doi: 10.1109/CVPR52733.2024.01354.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. VisualWebArena: Evaluating multimodal agents on

realistic visual web tasks. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Lai, H., Liu, X., Zhao, Y., Xu, H., Zhang, H., Jing, B., Ren, Y., Yao, S., Dong, Y., and Tang, J. Computerrl: Scaling end-to-end online reinforcement learning for computer use agents, 2025. URL https://arxiv.org/abs/2508.14040.

Lee, J., Min, T., An, M., Hahm, D., Lee, H., Kim, C., and Lee, K. Benchmarking mobile device control agents across diverse configurations, 2025. URL https://arxiv.org/abs/2404.16660.

Li, K., Ziyang, M., Lin, H., Luo, Z., Tian, Y., Ma, J., Huang, Z., and Chua, T.-S. Screenspot-pro: GUI grounding for professional high-resolution computer use. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=XaKNDIAHas.

Nguyen, D., Chen, J., Wang, Y., Wu, G., Park, N., Hu, Z., Lyu, H., Wu, J., Aponte, R., Xia, Y., Li, X., Shi, J., Chen, H., Lai, V. D., Xie, Z., Kim, S., Zhang, R., Yu, T., Tanjim, M., Ahmed, N. K., Mathur, P., Yoon, S., Yao, L., Kveton, B., Kil, J., Nguyen, T. H., Bui, T., Zhou, T., Rossi, R. A., and Dernoncourt, F. Gui agents: A survey, 2025. URL https://arxiv.org/abs/2412.13501.

OpenAI. Computer-using agent: Introducing a universal interface for ai to interact with the digital world, 2025. URL https://openai.com/index/computer-using-agent.

Qin, Y., Ye, Y., Fang, J., Wang, H., Liang, S., Tian, S., Zhang, J., Li, J., Li, Y., Huang, S., et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. URL https://arxiv.org/abs/2501.12326.

Rawles, C., Li, A., Rodriguez, D., Riva, O., and Lillicrap, T. Android in the wild: A large-scale dataset for android device control, 2023. URL https://arxiv.org/abs/2307.10088.

Rawles, C., Clinckemaillie, S., Chang, Y., Waltz, J., Lau, G., Fair, M., Li, A., Bishop, W., Li, W., Campbell-Ajala, F., Toyama, D., Berry, R., Tyamagundlu, D., Lillicrap, T., and Riva, O. Androidworld: A dynamic benchmarking environment for autonomous agents, 2024. URL https://arxiv.org/abs/2405.14573.

Sager, P. J., Meyer, B., Yan, P., Von Wartburg-Kottler, R., Etaiwi, L., Enayati, A., Nobel, G., Abdulkadir, A., Grewe, B. F., and Stadelmann, T. A comprehensive survey of agents for computer use: Foundations, challenges, and future directions. *Journal of Artificial Intelligence Research*, 85, March 2026. ISSN 1076-9757. doi: 10.1613/jair.1.19490. URL http://dx.doi.org/10.1613/jair.1.19490.

Team, Q. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wang, H., Zou, H., Song, H., Feng, J., Fang, J., Lu, J., Liu, L., Luo, Q., Liang, S., Huang, S., Zhong, W., Ye, Y., Qin, Y., Xiong, Y., Song, Y., Wu, Z., Li, A., Li, B., Dun, C., Liu, C., Zan, D., Leng, F., Wang, H., Yu, H., Chen, H., Guo, H., Su, J., Huang, J., Shen, K., Shi, K., Yan, L., Zhao, P., Liu, P., Ye, Q., Zheng, R., Xin, S., Zhao, W. X., Heng, W., Huang, W., Wang, W., Qin, X., Lin, Y., Wu, Y., Chen, Z., Wang, Z., Zhong, B., Zhang, X., Li, X., Li, Y., Zhao, Z., Jiang, C., Wu, F., Zhou, H., Pang, J., Han, L., Liu, Q., Ma, Q., Liu, S., Cai, S., Fu, W., Liu, X., Wang, Y., Zhang, Z., Zhou, B., Li, G., Shi, J., Yang, J., Tang, J., Li, L., Han, Q., Lu, T., Lin, W., Tong, X., Li, X., Zhang, Y., Miao, Y., Jiang, Z., Li, Z., Zhao, Z., Li, C., Ma, D., Lin, F., Zhang, G., Yang, H., Guo, H., Zhu, H., Liu, J., Du, J., Cai, K., Li, K., Yuan, L., Han, M., Wang, M., Guo, S., Cheng, T., Ma, X., Xiao, X., Huang, X., Chen, X., Du, Y., Chen, Y., Wang, Y., Li, Z., Yang, Z., Zeng, Z., Jin, C., Li, C., Chen, H., Chen, H., Chen, J., Zhao, Q., and Shi, G. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.02544.

Wu, Z., Wu, Z., Xu, F., Wang, Y., Sun, Q., Jia, C., Cheng, K., Ding, Z., Chen, L., Liang, P. P., and Qiao, Y. Os-atlas: A foundation action model for generalist gui agents, 2024. URL https://arxiv.org/abs/2410.23218.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. Osworld: benchmarking multimodal agents for open-ended tasks in real computer environments. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Xu, Y., Wang, Z., Wang, J., Lu, D., Xie, T., Saha, A., Sahoo, D., Yu, T., and Xiong, C. Aguvis: Unified pure vision agents for autonomous gui interaction, 2024. URL https://arxiv.org/abs/2412.04454.

Yang, P., Ci, H., and Shou, M. Z. macosworld: A multilingual interactive benchmark for gui agents, 2025. URL https://arxiv.org/abs/2506.04135.

Zhang, C., Li, L., He, S., Zhang, X., Qiao, B., Qin, S., Ma, M., Kang, Y., Lin, Q., Rajmohan, S., Zhang, D., and Zhang, Q. UFO: A UI-focused agent for windows OS interaction. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 597–622, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.26. URL https://aclanthology.org/2025.naacl-long.26/.

Zheng, B., Gou, B., Kil, J., Sun, H., and Su, Y. Gpt-4v(ision) is a generalist web agent, if grounded. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., and Neubig, G. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

A. Apple Virtualization Framework

MacArena is built on Apple's Virtualization framework[3], a native hypervisor API introduced in macOS 11 (Big Sur) that enables the creation and management of virtual machines directly on Apple Silicon hardware. Unlike traditional virtualization solutions such as QEMU, which rely on software emulation, the framework leverages the hardware virtualization extensions of M-series chips to run guest operating systems at near-native performance.

A key practical constraint of the framework is that Apple Silicon hosts can run at most two macOS guest VMs simultaneously — a hard limit enforced at the system level. This restricts the degree of parallelism available per machine and is an important consideration for benchmark throughput. In MacArena, we account for this by running up to two evaluation tasks in parallel on a single host. To scale evaluation further, multiple host machines can be employed independently, as the per-host VM limit does not preclude distributed execution across several Apple Silicon machines.

[3] https://developer.apple.com/documentation/virtualization