EvoGUI: An Evolution-Aware Benchmark for GUI State-Transition Understanding

Anonymous ACL submission

Abstract

GUI agents must reason about how actions transform interface states, but end-to-end success rates entangle this ability with perception, grounding, planning, and recovery. We introduce EvoGUI, an annotation-free diagnostic framework that converts logged GUI trajectories into three visual question answering probes: temporal ordering, inverse action/value prediction, and one-step reachability discrimination. By deriving supervision from trajectory order and recorded actions, EvoGUI scales to new GUI corpora through data normalization rather than manual labeling. We instantiate EvoGUI-Bench from Mind2Web and WebLINX , and evaluate 28 vision-language model configurations in a zero-shot setting. Results show that EvoGUI-Bench uncovers unsaturated, non-trivial differences in GUI state-evolution understanding, thereby validating its efficacy and highlighting the need for explicit evolution-aware diagnostics.

1 Introduction

Vision-language models (VLMs) are increasingly used as the perception and reasoning backbone of graphical user interface (GUI) agents. Unlike static visual recognition, GUI interaction is inherently evolution-aware: an agent must understand not only what is visible on the current screen, but also how actions such as clicking, typing, or selecting transform the interface state. Such state-evolution understanding is necessary for reliable execution because agents must anticipate how actions change GUI states.

Existing GUI benchmarks have made substantial progress in evaluating agents in realistic web (Zhou et al., 2024; Koh et al., 2024; Drouin et al., 2024), mobile (Rawles et al., 2025), and operating-system environments (Xie et al., 2024; Bonatti et al., 2025). These execution-based benchmarks are essential for measuring deployed task performance, but their success rates conflate perception, OCR, grounding, planning, tool use, and recovery.

Recent dynamics-oriented evaluations use world modeling as a useful lens for diagnostic evaluation. ENACT (Wang et al., 2025) separates forward and inverse world modeling in embodied egocentric interaction, while MobileWorldBench studies semantic world modeling for mobile agents (Li et al., 2025). These efforts suggest that transition-level evaluation can expose failures hidden by aggregate task success. For GUI agents, the missing step is to turn existing trajectories into the same kind of transition-focused diagnostic, so that state-evolution reasoning can be evaluated without live rollout or future-frame generation.

In this work, we present **EvoGUI**, which turns logged GUI trajectories into transition-level diagnostics without manual annotation, and instantiate it as **EvoGUI-Bench**, a benchmark for GUI state-transition understanding. EvoGUI constructs three discriminative VQA probes: temporal ordering, inverse action/value prediction, and one-step reachability discrimination. Because their labels come from logged trajectories, the same pipeline can be scaled to additional GUI corpora.

Our contributions can be summarized as follows:

- **Automated Framework.** We develop **EvoGUI**, a scalable and annotation-free framework that transforms raw GUI trajectories into VQA-style dynamics instances.
- **Benchmark.** We introduce **EvoGUI-Bench**, a 3,000-instance evolution-aware diagnostic benchmark across 120 domains for evaluating GUI state-transition understanding.
- **Systematic Analysis.** We evaluate 28 representative VLM configurations and show that EvoGain remains far from saturation; general VLM strength, GUI specialization, and parameter count are useful but insufficient proxies

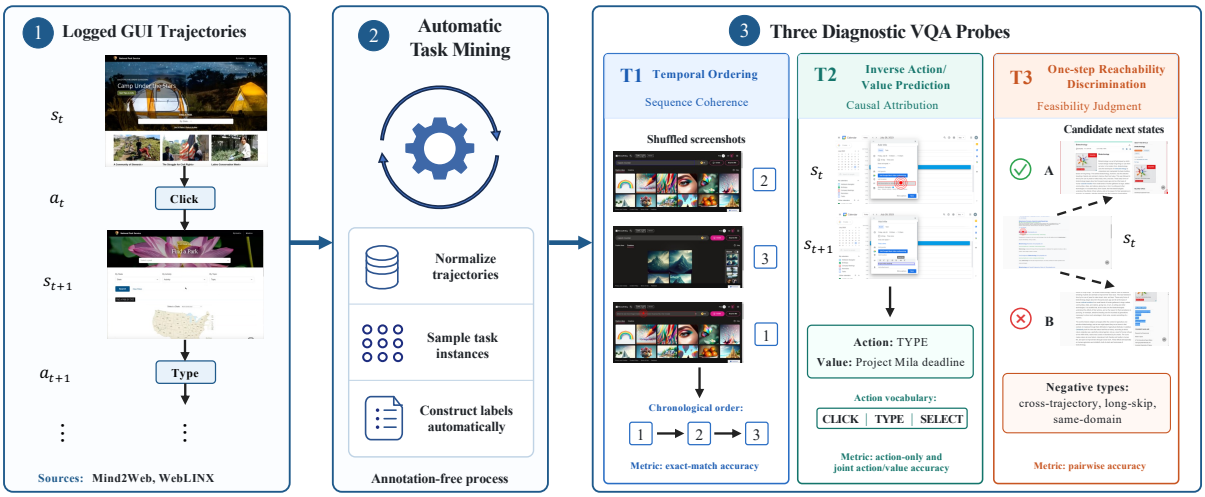

Figure 1: Overview of EvoGUI benchmark construction. Logged GUI trajectories from Mind2Web and WebLINX are normalized into transition records and automatically mined into three diagnostic VQA probes: T1 temporal ordering for sequence coherence, T2 inverse action/value prediction for causal attribution, and T3 one-step reachability discrimination for feasibility judgment. Labels are derived from trajectory order and recorded actions, requiring no additional manual annotation.

for state-evolution ability.

2 Benchmark

Figure 1 summarizes the EvoGUI construction pipeline. The remainder of this section formalizes the normalized trajectory schema, the three diagnostic VQA tasks, the automatic mining procedure, and the resulting EvoGUI-Bench split.

2.1 Trajectory Schema

EvoGUI converts existing GUI interaction logs into diagnostic visual question answering tasks. We assume each source dataset can be normalized into a trajectory

$$\tau = [(s_t, a_t, s_{t+1}, m_t)]_{t=1}^{T},$$

where s_t and s_{t+1} are consecutive screenshots, a_t is the recorded user operation, and m_t stores metadata such as the trajectory id, domain, instruction, and source dataset.

2.2 Diagnostic Tasks

The benchmark contains three tasks that probe complementary aspects of GUI state-transition reasoning. T1 asks whether observed states can be temporally ordered, T2 asks which action explains an adjacent state change, and T3 asks whether a candidate is immediately reachable in one step.

T1, *temporal ordering*, evaluates temporal state structure. Given $K \in \{3, 4, 5\}$ screenshots sampled from the same trajectory and shown in a shuffled order, the model must output their original chronological order.

T2, *inverse action/value explanation*, evaluates whether a model can explain an observed transition. Given a pair (s_t, s_{t+1}), the model predicts an action class from $\{\text{CLICK}, \text{TYPE}, \text{SELECT}\}$. For TYPE and SELECT, the model must also generate the typed text or selected option.

T3, *one-step reachability discrimination*, assesses single-action transition feasibility. Given a source screenshot s_t and two shuffled candidates, the model must identify the one reachable from s_t by a single GUI interaction. The negative candidate is sampled from one of three tiers: *cross-trajectory*, a screen from a different trajectory; *same-domain*, a screen from the same domain or application that is intended to be more plausible than a random cross-trajectory screen; and *long-skip*, a later screen s_{t+n} from the same trajectory with $n \geq 3$. Long-skip negatives are especially diagnostic because they are real future states in the same workflow, but not immediate next states.

2.3 Automatic Task Mining

EvoGUI builds all three tasks automatically from normalized trajectories. The miner uses only temporal order, recorded actions, action values already present in the source logs, and automatically constructed candidate pairs; no new annotator judgment is needed for the benchmark labels.

For T1, the miner first selects trajectories with at

least K valid screenshots, samples temporally ordered index sets for K, shuffles the selected screenshots, and stores the chronological index order as the label.

For T2, the miner iterates over adjacent transitions and keeps steps whose normalized action belongs to the target vocabulary. Actions that carry semantic content are retained only when their value is available; this prevents a model from receiving credit for predicting TYPE or SELECT while ignoring the input value.

For T3, the miner treats every valid adjacent transition as a positive pair and constructs a pairwise question by adding one negative candidate.

2.4 Benchmark Instantiation

We instantiate the benchmark as EvoGUI-Bench, a 3,000-instance split constructed from Mind2Web (Deng et al., 2023) and WebLINX (Lu et al., 2024). We map CLICK/HOVER style operations to CLICK, TYPE/TEXTINPUT/PASTE to TYPE, and SELECT/CHANGE to SELECT. Other intents, such as LOAD, SUBMIT, SCROLL, COPY, and tab-management actions, are excluded from T2 but their screenshots can still contribute to T1 and T3.

The same construction is easy to transfer to other GUI corpora: after source logs are normalized into the trajectory schema above, labels for ordering, inverse prediction, and reachability are derived automatically from temporal order and recorded actions. This high degree of automation lets EvoGUI scale beyond the two sources used in EvoGUI-Bench without requiring new manual annotation.

3 Experiments

3.1 Setup

Models. We evaluate 28 complete vision-language model configurations. We group models by release status. The open-weight group includes Qwen3.5 (Qwen Team, 2026a), Qwen3.6 (Qwen Team, 2026b,c), Qwen3-VL (Bai et al., 2025), Gemma-3-4B-IT (Team et al., 2025), Kimi-K2.5/K2.6 (Team et al., 2026a; Moonshot AI, 2026), GLM-4.5V (Team et al., 2026b), UI-TARS-1.5-7B (Qin et al., 2025), and MiMo-V2.5 (Xiaomi MiMo Team, 2026). The closed-weight group includes Claude-3-Haiku (Anthropic, 2024), OpenAI GPT-5/GPT-5.4 variants (OpenAI, 2025, 2026b,a,c), Doubao Seed-2.0 variants (ByteDance Seed, 2026), GLM-5V-Turbo (Z.AI, 2026), Gemini-3-Flash-Preview (Google DeepMind, 2025), and Grok-4.20 (xAI, 2026). All models are evaluated zero-shot with the same task prompts and JSON output parser.

Table 1: Statistics of EvoGUI-Bench. Each task contains 1,000 instances; T3 is balanced by negative type.

Part	Slice	Count
T1	$K = 3/4/5$	334 / 333 / 333
T2	click / type / select	461 / 461 / 78
T3	cross / same-domain / long-skip	334 / 333 / 333

Data. We evaluate on EVOGUI-BENCH, the 3,000-instance benchmark split summarized in Table 1. Each task contains 1,000 examples, and T2 uses the unified {CLICK, TYPE, SELECT} action vocabulary. The split is balanced over T1 sequence length and T3 negative type, and it is domain-balanced by construction.

Protocol and metrics. All model runs use temperature 0 and no cache reuse. For locally executed Qwen models, we use BF16 SGLang inference with one running request; hosted models use the corresponding provider endpoint with the same prompts and parser. Table 2 standardizes the main comparison at `max_image_side=512`. We report T1 exact ordering accuracy, T2 action-only top-1 and action/value joint top-1, and T3 pairwise transition accuracy. T3 long-skip accuracy is also a meaningful immediate-reachability slice, which is reported in Table 5 in Appendix. The random-choice baseline is the average of four random runs.

To synthesize these structurally decoupled probes with random baselines , we introduce the *EvoGain Index*, a relative improvement score over the random baseline. For each task $i \in \{\mathrm{T1}, \mathrm{T2}, \mathrm{T3}\}$, we compute the normalized gain

$$\mathrm{NormScore}_i = \frac{\mathrm{ModelScore}_i - \mathrm{Baseline}_i}{100 - \mathrm{Baseline}_i} \times 100,$$

where the model scores are T1 exact ordering accuracy, T2 joint accuracy, and T3 pairwise accuracy, respectively. EvoGain is the macro-average of the three normalized gains.

3.2 Main Results

Table 2 shows that EvoGUI-Bench produces clear and interpretable model separation. After removing task-specific chance effects, EvoGain ranks Gemini-3-Flash-Preview highest at 60.4%, followed by Seed-2.0-lite-260428 at 50.7%. In contrast, several models with high raw T3 scores drop

Table 2: Main results on EvoGUI-Bench at `max_image_side=512`. Rows are grouped by release status and ordered within each group from lower to higher EvoGain. All values are percentages. Bold indicates the best result; underline indicates the second-best.

Model	T1 ↑	T2 top-1 ↑	T2 joint ↑	T3 ↑	EvoGain ↑
Baseline					
Random choice	8.0	33.3	15.0	49.9	0.0
Open-weight models					
Qwen3.5-0.8B	8.1	49.6	47.4	57.7	17.9
Gemma-3-4B-IT	8.8	50.3	37.8	67.3	20.8
UI-TARS-1.5-7B	16.2	52.2	49.3	63.4	25.4
Qwen3.5-2B	12.3	52.1	47.1	71.3	28.4
Qwen3.5-4B	27.6	55.8	49.0	69.6	33.5
Qwen3.5-9B	28.9	59.9	51.4	70.3	35.4
GLM-4.5V	30.9	61.9	52.3	71.2	37.1
Qwen3-VL-235B-A22B	30.9	64.1	55.1	70.2	37.5
Qwen3.6-27B	29.2	59.1	55.1	72.6	38.5
Qwen3.5-397B-A17B	34.4	64.7	56.6	70.3	39.5
Qwen3.5-27B	29.7	62.3	54.8	74.1	39.6
Qwen3.5-35B-A3B	31.1	60.9	54.8	74.7	40.5
Kimi-K2.5	33.4	61.9	52.8	76.3	41.6
Qwen3.6-35B-A3B	33.3	61.9	56.6	74.7	42.0
MiMo-V2.5	34.2	59.8	54.5	76.1	42.4
Qwen3.5-122B-A10B	32.8	63.4	56.4	77.8	43.8
Kimi-K2.6	34.4	61.2	52.6	79.4	43.9
Closed-weight models					
Claude-3-Haiku	8.8	51.7	36.2	62.7	17.1
GPT-5-nano	13.3	47.3	41.0	69.4	25.1
GPT-5.4-nano	25.1	55.8	49.0	73.5	35.2
GPT-5-mini	30.3	59.3	52.5	69.2	35.6
Grok-4.20	27.6	60.6	49.9	72.3	35.7
Seed-2.0-mini-260428	37.2	62.3	45.9	72.3	37.6
GLM-5V-Turbo	31.9	60.0	54.0	76.9	41.9
GPT-5.4	42.1	60.4	50.9	75.4	43.4
Seed-2.0-pro-260215	44.4	66.0	55.7	80.9	49.8
Seed-2.0-lite-260428	49.4	62.3	51.7	81.9	50.7
Gemini-3-Flash-Preview	**58.3**	**75.0**	**65.4**	**83.6**	**60.4**

substantially after normalization. Even the best model exceeds random choice by only 50.3 points on T1, 50.4 points on T2 joint accuracy, and 33.7 points on T3, indicating substantial remaining headroom.

The results further show that state-evolution ability cannot be reliably inferred from general capability, GUI specialization, or parameter count. Although strong generalist models lead overall, the GUI-specialized UI-TARS-1.5-7B reaches only 25.4 EvoGain, and larger Qwen models do not consistently outperform smaller variants. For example, Qwen3.5-397B-A17B scores below Qwen3.5-122B-A10B, while Qwen3-VL-235B-A22B scores below Qwen3.5-35B-A3B. This non-monotonic trend, visualized in Appendix Figure 2, suggests that transition-level understanding should be evaluated directly rather than inferred from model scale or general performance.

3.2.1 T1: Temporal Ordering

T1 provides the clearest separation of high-level state-evolution ability. However, even the best model still makes over 40% exact-ordering errors, indicating that recovering multi-screen temporal structure remains challenging and cannot be assumed from general vision-language capability.

3.2.2 T2: Inverse Action/Value Prediction

T2 shows that action recognition is easier than full transition explanation. Across models, action-only accuracy consistently exceeds joint action/value accuracy, indicating that models can often detect salient visual changes but fail to recover value-bearing details such as typed strings or selected options. The per-class results in Appendix Table 4 further identify TYPE and SELECT transitions as the main bottlenecks.

3.2.3 T3: One-Step Reachability Discrimination

T3 shows that aggregate GUI plausibility can overstate transition reasoning. While stronger models achieve high aggregate pairwise accuracy, Appendix Table 5 shows that many still struggle with long-skip negatives, which require distinguishing immediate successors from later states in the same workflow. This gap suggests that current VLMs often capture coarse workflow plausibility without reliably modeling action-conditioned one-step reachability. Appendix Table 6 further shows that higher screenshot resolution improves inverse prediction but does not consistently repair temporal reachability.

4 Conclusion

In this work, we introduced EvoGUI, an annotation-free diagnostic framework that isolates GUI state-transition understanding from end-to-end execution. By automatically mining logged trajectories into ordering, inverse prediction, and reachability probes, EvoGUI provides a scalable way to evaluate whether VLMs understand how interface states evolve. Systematic benchmarking of 28 VLM configurations on EvoGUI-Bench using the baseline-normalized EvoGain Index shows that current models remain far from saturation: strong generalist models lead overall, but general capability, GUI specialization, and parameter count do not reliably determine transition-level performance. These findings suggest that future GUI-agent research should complement execution success rates with explicit diagnostics for action-conditioned interface dynamics.

Limitations

EvoGUI is a transition-level diagnostic rather than a complete measure of deployed agent performance. Its current instantiation contains 3,000 instances mined from Mind2Web and WebLINX, which is sufficient for a controlled first benchmark but should be expanded to larger trajectory pools, more domains, and longer interaction traces. The present task space is also centered on browser-like visual GUIs and a shared {CLICK, TYPE, SELECT} action vocabulary; future versions should cover mobile and desktop interfaces, scrolling-heavy workflows, asynchronous updates, hidden state, and non-visual interaction settings such as command-line interfaces. Finally, the automatically sampled T3 negatives diagnose one-step reachability under observed trajectory structure, but they are not exhaustive counterfactuals over all possible application states. Scaling the same annotation-free construction to broader environments, and pairing it with execution-validated counterfactuals, is an important next step.

Ethical Considerations

EvoGUI is constructed from publicly released Mind2Web and WebLINX trajectories and does not introduce new human-subject data collection or additional manual annotation. The benchmark uses screenshots, recorded actions, and trajectory metadata only to create diagnostic evaluation instances. Because web screenshots may contain user- or site-provided text inherited from the source corpora, any public release of EvoGUI-Bench should follow the source dataset licenses and filter or mask sensitive content when required. The benchmark is intended for measuring GUI state-transition understanding, not for deploying agents to perform real-world actions.

References

Anthropic. 2024. The Claude 3 model family: Opus, Sonnet, Haiku. Model card. Accessed: 2026-05-25.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Keunho Jang, and Zheng Hui. 2025. Windows agent arena: Evaluating multi-modal OS agents at scale. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4874–4910. PMLR.

ByteDance Seed. 2026. Seed2.0 model card: Towards intelligence frontier for real-world agency. Accessed: 2026-05-25.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pages 28091–28114. Curran Associates, Inc.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. 2024. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11642–11662. PMLR.

Google DeepMind. 2025. Gemini 3 Flash model card. Accessed: 2026-05-25.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. 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)*, pages 881–905, Bangkok, Thailand. Association for Computational Linguistics.

Shufan Li, Konstantinos Kallidromitis, Akash Gokul, Yusuke Kato, Kazuki Kozuka, and Aditya Grover. 2025. Mobileworldbench: Towards semantic world modeling for mobile agents. *Preprint*, arXiv:2512.14014.

Xing Han Lu, Zdeněk Kasner, and Siva Reddy. 2024. WebLINX: Real-world website navigation with multi-turn dialogue. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 33007–33056. PMLR.

Moonshot AI. 2026. Kimi K2.6 model card. Model card. Accessed: 2026-05-25.

OpenAI. 2025. GPT-5 model. API documentation. Accessed: 2026-05-25.

OpenAI. 2026a. GPT-5.4 Mini model. API documentation. Accessed: 2026-05-25.

OpenAI. 2026b. GPT-5.4 model. API documentation. Accessed: 2026-05-25.

OpenAI. 2026c. GPT-5.4 Nano model. API documentation. Accessed: 2026-05-25.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, arXiv:2501.12326.

Qwen Team. 2026a. Qwen3.5: Towards native multimodal agents.

Qwen Team. 2026b. Qwen3.6-27B: Flagship-level coding in a 27B dense model.

Qwen Team. 2026c. Qwen3.6-35B-A3B: Agentic coding power, now open to all.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, volume 2025, pages 406–441.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, and 307 others. 2026a. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 74 others. 2026b. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *Preprint*, arXiv:2507.01006.

Qineng Wang, Wenlong Huang, Yu Zhou, Hang Yin, Tianwei Bao, Jianwen Lyu, Weiyu Liu, Ruohan Zhang, Jiajun Wu, Li Fei-Fei, and Manling Li. 2025. Enact: Evaluating embodied cognition with world modeling of egocentric interaction. *Preprint*, arXiv:2511.20937.

xAI. 2026. Grok 4.20. API documentation. Accessed: 2026-05-25.

Xiaomi MiMo Team. 2026. MiMo-V2.5 model card. Model card. Accessed: 2026-05-25.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pages 52040–52094. Curran Associates, Inc.

Z.AI. 2026. GLM-5V-Turbo developer documentation. API documentation. Accessed: 2026-05-25.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

A Reproducibility Details and Additional Results

A.1 Split Construction

This appendix reports construction and evaluation details that are secondary to the main argument but necessary for auditing the benchmark. We construct the reported split from normalized Mind2Web and WebLINX trajectories, then sample 1,000 instances for each diagnostic task. The main paper gives the aggregate split statistics; here we specify the eligibility rules that determine which logged records can become labels.

For T1, screenshots are sampled from valid trajectories with $K \in \{3, 4, 5\}$ and are shuffled before being shown to the model. For T2, an instance is retained only when the normalized action is in the shared {CLICK, TYPE, SELECT} vocabulary; value-bearing actions are retained only when the typed string or selected option is available in the source log. For T3, the positive pair is any valid adjacent screenshot transition, including transitions whose source action is outside the T2 vocabulary, and the negative candidate is drawn from a cross-trajectory, long-skip, or same-domain pool. This separation keeps T2 value-sensitive while allowing richer logged interactions to contribute to temporal and reachability probes.

A.2 Relation Among Diagnostic Tasks

T1, T2, and T3 are related but not redundant because each probe blocks a different shortcut. T1 can reward recognizing workflow progress, but it does not require identifying the user action that produced each transition. T2 fixes the before–after pair

Table 3: Action normalization used when constructing EvoGUI-Bench. Actions outside the shared vocabulary are excluded from T2 but may still provide screenshots for T1 and T3.

Source action	Unified action	T2 use
`click`, `hover`	CLICK	retained
`type`, `textInput`, `paste`	TYPE	retained with value
`select`, `change`	SELECT	retained with value
`load`, `submit`, `scroll`	not mapped	excluded
copy, tab-management actions	not mapped	excluded

and asks for the causal action/value explanation, but it does not test whether the model can order a longer state sequence. T3 fixes the source state and asks for an immediate successor; its long-skip negatives are deliberately plausible future screens, which prevents eventual workflow plausibility from being treated as one-step reachability. Reporting the probes separately therefore avoids interpreting success on one transition cue as evidence for all forms of GUI state-evolution understanding.

A.3 Evaluation Protocol and Parsing

All main-comparison model rows use a single zero-shot prompt per task, temperature 0, no cache reuse, and the same task files at `max_image_side=512`. Each model row contains 1,000 raw predictions per task, for 3,000 predictions per configuration. The main comparison therefore contains 84,000 raw model predictions across 28 configurations. The random-choice baseline in Table 2 is the average of four random runs, and only the average row is used in the paper. Table 6 additionally reports a complete 1024-pixel Qwen sweep for the eight models that also have complete 512-pixel runs.

The parser first extracts a JSON object and validates the task-specific schema. T1 is valid only when the `order` field is a complete permutation of the shown image indices. T2 is valid only when `action` belongs to the task vocabulary; the `value` field is exact-matched for TYPE and SELECT, and is ignored for CLICK. T3 is valid only when `answer` selects one of the candidate labels. If JSON parsing fails, the fallback parser performs only conservative substring matching for the discrete label; it does not recover free-form values, so the T2 joint metric is not inflated by malformed responses. Outputs that remain invalid after fallback are counted as incorrect.

A.4 T2 Per-Class Recall

Table 4 shows why T2 joint accuracy is a stricter inverse metric than aggregate action top-1 accuracy.

Table 4: T2 aggregate action top-1 accuracy and per-class recall on EvoGUI-Bench for the model rows in Table 2. Rows use the same release-status grouping and model order as Table 2. All rows use `max_image_side=512`; values are percentages. Bold indicates the best result; underline indicates the second-best.

Model	T2 top-1 ↑	Click ↑	Type ↑	Select ↑
Open-weight models				
Qwen3.5-0.8B	49.6	**96.5**	10.8	1.3
Gemma-3-4B-IT	50.3	56.4	48.6	24.4
UI-TARS-1.5-7B	52.2	<u>96.3</u>	16.7	1.3
Qwen3.5-2B	52.1	85.5	25.6	11.5
Qwen3.5-4B	55.8	81.8	36.9	14.1
Qwen3.5-9B	59.9	80.0	45.8	24.4
GLM-4.5V	61.9	80.3	49.7	25.6
Qwen3-VL-235B-A22B	64.1	87.6	45.6	34.6
Qwen3.6-27B	59.1	94.8	30.6	16.7
Qwen3.5-397B-A17B	64.7	88.5	49.0	16.7
Qwen3.5-27B	62.3	88.9	43.8	14.1
Qwen3.5-35B-A3B	60.9	91.5	37.1	20.5
Kimi-K2.5	61.9	80.5	48.8	29.5
Qwen3.6-35B-A3B	61.9	95.2	35.4	21.8
MiMo-V2.5	59.8	92.0	33.2	26.9
Qwen3.5-122B-A10B	63.4	91.3	43.6	15.4
Kimi-K2.6	61.2	80.9	46.0	34.6
Closed-weight models				
Claude-3-Haiku	51.7	57.9	46.9	43.6
GPT-5-nano	47.3	76.8	24.3	9.0
GPT-5.4-nano	55.8	83.3	34.3	20.5
GPT-5-mini	59.3	84.8	40.3	20.5
Grok-4.20	60.6	73.1	53.8	26.9
Seed-2.0-mini-260428	62.3	55.3	**73.8**	35.9
GLM-5V-Turbo	60.0	89.4	37.7	17.9
GPT-5.4	60.4	81.3	40.8	<u>52.6</u>
Seed-2.0-pro-260215	<u>66.0</u>	78.1	58.6	38.5
Seed-2.0-lite-260428	62.3	67.7	59.4	47.4
Gemini-3-Flash-Preview	**75.0**	82.0	<u>69.6</u>	**65.4**

Click recall can be high even when type and select recall remain low, which means that action-only accuracy can be driven by visually salient click-like transitions while underrepresenting value-bearing failures.

A.5 T3 Slice Results

Table 5 explains why the main paper reports aggregate T3 in Table 2 and keeps long-skip accuracy as an appendix diagnostic. Cross-trajectory negatives mostly test coarse mismatch rejection, while long-skip negatives are real later states from the same workflow. The latter therefore better tests whether a model distinguishes eventual plausibility from one-step reachability, and it remains substantially harder for most models.

A.6 Resolution Comparison

Table 6 indicates a consistent gain for inverse prediction: all eight Qwen rows improve on T2 top-1

Table 5: T3 accuracy by negative type for the 512-pixel model rows in Table 2. Rows follow the release-status ordering in Table 2. Cross-trajectory negatives test coarse mismatch rejection, same-domain negatives test related-interface distractors, and long-skip negatives test immediate reachability against real later states from the same trajectory. All values are percentages. Bold indicates the best result; underline indicates the second-best.

Model	Cross-traj. ↑	Same-domain ↑	Long-skip ↑
Open-weight models			
Qwen3.5-0.8B	66.8	55.6	50.8
Gemma-3-4B-IT	92.2	65.8	43.8
UI-TARS-1.5-7B	80.8	57.7	51.7
Qwen3.5-2B	95.8	68.2	49.9
Qwen3.5-4B	94.9	66.7	47.2
Qwen3.5-9B	97.6	67.9	45.4
GLM-4.5V	96.4	71.5	45.6
Qwen3-VL-235B-A22B	97.3	71.5	41.7
Qwen3.6-27B	97.0	71.8	49.0
Qwen3.5-397B-A17B	95.8	66.4	48.6
Qwen3.5-27B	97.3	73.9	51.1
Qwen3.5-35B-A3B	98.5	74.5	51.1
Kimi-K2.5	97.3	77.5	54.1
Qwen3.6-35B-A3B	98.5	75.7	49.9
MiMo-V2.5	97.0	76.0	55.3
Qwen3.5-122B-A10B	98.8	75.7	58.9
Kimi-K2.6	99.4	81.7	57.1
Closed-weight models			
Claude-3-Haiku	82.9	59.5	45.6
GPT-5-nano	88.0	65.2	55.0
GPT-5.4-nano	94.6	73.9	52.0
GPT-5-mini	97.9	70.9	38.7
Grok-4.20	99.4	73.0	44.4
Seed-2.0-mini-260428	98.2	76.3	42.3
GLM-5V-Turbo	98.5	78.4	53.8
GPT-5.4	97.0	70.9	58.3
Seed-2.0-pro-260215	**100.0**	82.6	60.1
Seed-2.0-lite-260428	98.5	**84.1**	63.1
Gemini-3-Flash-Preview	99.7	83.2	**67.9**

Table 6: Resolution comparison for the eight Qwen models with complete 512- and 1024-pixel sweeps. Each cell reports 512→1024 accuracy; Long denotes T3 long-skip accuracy. Values are percentages. Bold and underline mark the best and second-best values within each side of a metric column, respectively.

Model	T1	T2 top-1	T2 joint	T3	Long
Qwen3.5-0.8B	8.1→6.3	49.6→54.1	47.4→48.9	57.7→62.7	50.8→50.8
Qwen3.5-2B	12.3→12.9	52.1→54.9	47.1→49.3	71.3→73.2	49.9→51.1
Qwen3.5-4B	27.6→31.3	55.8→59.7	49.0→52.7	69.6→74.1	47.2→49.0
Qwen3.5-9B	28.9→37.2	59.9→66.2	51.4→58.4	70.3→73.4	45.4→45.1
Qwen3.5-27B	29.7→34.2	**62.3→69.1**	54.8→62.6	74.1→73.8	**51.1**→46.3
Qwen3.5-35B-A3B	31.1→**41.0**	60.9→68.2	54.8→62.5	**74.7**→75.6	**51.1**→50.2
Qwen3.6-27B	29.2→35.2	59.1→67.2	55.1→62.6	72.6→72.3	49.0→47.5
Qwen3.6-35B-A3B	**33.3**→40.4	61.9→69.0	**56.6→63.2**	**74.7→77.8**	49.9→**53.8**

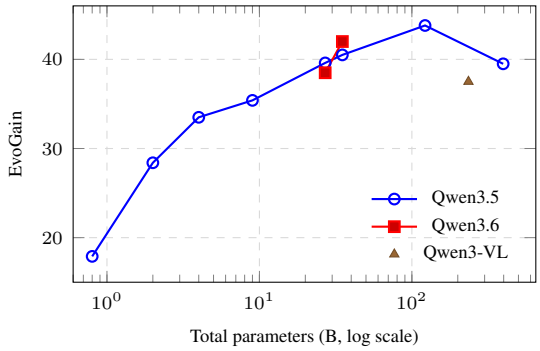

Figure 2: Qwen-family total parameter count versus EvoGain on EvoGUI-Bench. The trend is not monotonic: Qwen3.5-397B-A17B scores below Qwen3.5-122B-A10B, and Qwen3-VL-235B-A22B scores below several smaller Qwen variants.

and T2 joint accuracy when increasing the maximum image side from 512 to 1024. Ordering improves for seven of the eight rows, whereas T3 and long-skip reachability are less monotonic. This supports using 512 pixels for the main cross-model table while treating the 1024-pixel results as a controlled resolution analysis rather than as directly comparable main scores.

A.7 Qwen Parameter Trend

Figure 2 supports the main-text claim that parameter count is an insufficient proxy for GUI state-evolution ability. Larger models often improve over smaller ones in the same family, but the relationship is not deterministic; architectural choices, training data, and transition-oriented supervision likely matter in addition to scale.

A.8 Interpretation Boundaries

The appendix tables should be read as task-local diagnostics rather than calibrated measures of intrinsic task difficulty. T1, T2, and T3 use different answer formats, chance levels, and scoring rules, so absolute accuracies are not directly comparable across tasks. EvoGain normalizes chance performance for the main ranking, but it remains a diagnostic summary rather than a utility measure of deployed agent performance.

The additional slices refine the main claims without broadening them beyond the benchmark construction. T2 exact-value scoring tests whether the model can recover the logged typed string or selected option from the visual transition; errors may reflect failures in OCR, state comparison, or value binding. T3 negatives are sampled contrastive candidates from trajectory structure, so a correct answer demonstrates discrimination against the sampled distractor, not proof that the rejected screen is impossible under every application state or user policy. These boundaries support the narrower conclusions used in the main text: exact temporal ordering remains difficult, inverse action/value prediction is limited by value recovery, and long-skip negatives make one-step reachability discrimination harder

than coarse cross-trajectory rejection.

A.9 LLM Usage Statement

Generative AI assistants were used for manuscript language polishing and partial code framework optimization. All outputs were manually reviewed and verified by the authors, who take full responsibility for the entire work.

A.10 Prompt Templates

Tables 7–9 report the complete zero-shot prompt templates used in the experiments. We typeset them as full-width role-separated tables without vertical rules, so that the model-facing instructions, instance-specific attachments, and parsed output fields are easy to audit. T1 parses `order`, T2 parses `action`/`value`/`confidence`, and T3 parses `answer`/`confidence`. Placeholders such as the user goal, candidate labels, and image slots are filled from each instance. The prompts ask the model to inspect the screens before answering, but the required output is a JSON object only; no free-form reasoning trace is parsed or evaluated.

Table 7: Prompt template for T1 temporal ordering. The model receives shuffled screenshots from one session and returns their chronological order.

Task	T1: Temporal Ordering
System	You are an expert evaluator of GUI screenshots. You will reason carefully about temporal causality between screens, and answer in the requested JSON format only.
User	I will show you K screenshots from the same user session, but presented in SHUFFLED order. Each screenshot is labeled by its position in the shuffled sequence. The task is to recover the chronological order, namely which screenshot happened first, second, and so on. Think step by step about UI affordances, what action would plausibly lead from one screen to the next, and the user's apparent goal when provided. Respond with a single JSON object of the form `{"order": [. . . ]}`, where positions are integers from 1 to K and each position is used exactly once. Output only the JSON, no markdown, and no commentary.
Attachments	Each shuffled screenshot is attached after its position label.
Parsed output	`order`: a complete permutation of the shown screenshot indices.

Table 8: Prompt template for T2 inverse action/value prediction. The model receives two consecutive screenshots and predicts the action that caused the transition.

Task	T2: Inverse Action/Value Prediction
System	You are an expert evaluator of GUI screenshots. Given two consecutive screens of a user session, infer the single user action that caused the change. Answer in the requested JSON format only.
User	Below are two screenshots. The first is before the user's action, and the second is after the action. Pick the single user action that most likely caused the change from the provided options. If the action is `type`, predict the EXACT typed text. If the action is `select`, predict the EXACT option chosen from the dropdown or radio group. If the action is `click`, set the value to an empty string. Use the user goal when it is provided. Reason briefly about visible differences such as cursor, focus, new content, URL change, or dialog appearance before committing. Answer with `{"action": "<one option>", "value": "<typed/selected text or empty>", "confidence": <0..1>}`. Output only the JSON.
Attachments	The before and after screenshots are attached in order.
Parsed output	`action`: one option from the task vocabulary; `value`: the typed or selected value, or an empty string for CLICK; `confidence`: a number in $[0, 1]$.

Table 9: Prompt template for T3 one-step reachability discrimination. The model chooses which candidate screen is reachable from the start screen by one action.

Task	T3: One-Step Reachability Discrimination
System	You are an expert evaluator of GUI screenshots. Given a starting screen and candidate next screens, decide which candidate could be reached from the starting screen by a single user action. Answer in the requested JSON format only.
User	Below is a starting screen, followed by two candidate next screens labeled A and B. Exactly one of A and B is reachable from the starting screen by a SINGLE action, such as one click, one type, one scroll, or one back operation. The other is not reachable in one step. Use the user goal when it is provided. Think about which UI elements are visible and clickable in the starting screen, and what their immediate effect would be. Reject candidates that imply multiple navigation steps, a different page entirely, or a state that requires intermediate screens to appear first. Answer with `{"answer": "A" | "B", "confidence": <0..1>}`. Output only the JSON.
Attachments	The starting screen and the two candidates are attached in order.
Parsed output	`answer`: either `A` or `B`; `confidence`: a number in $[0, 1]$.