KNOWACT-MEMORY: SELF-EVOLVING SKILL MEMORY FOR EFFICIENT MOBILE GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Mobile GUI agents powered by multimodal large language models have been used to automate complex tasks. However, existing agents typically suffer from memoryless planning and regenerate action sequences from scratch for every task. This inefficiency increases inference latency and token cost while limiting the ability to learn from experience. We propose KNOWACT-MEMORY, a self-evolving framework that shifts mobile GUI agents from heavy online planning to efficient state-grounded execution. Instead of replaying raw trajectories, KNOWACT-MEMORY abstracts interaction traces into structured parametric skills, merges and refines them through semantic consolidation, and maintains a dual library distilled from both successful executions and repaired failures. Experiments on AndroidWorld, MobileWorld, and real-device HarmonyOS evaluations show that KNOWACT-MEMORY improves measured success rates while reducing inference cost. With Qwen3-VL-8B on AndroidWorld, it improves success from 45.68% to 52.58% and reduces per-step inference time by 25.4%.

1 INTRODUCTION

Recent advances in multimodal large language models (MLLMs) have enabled agents to interpret and interact with graphical user interfaces (GUIs) through visual understanding (Cheng et al., 2024; Wen et al., 2024; Qin et al., 2025). State-of-the-art methods such as Mobile-Agent (Wang et al., 2024; Ye et al., 2025) and AutoDroid (Wen et al., 2024) demonstrate competitive performance on benchmarks such as AndroidWorld (Rawles et al., 2024), highlighting the potential of AI-powered mobile automation. However, a key limitation persists: current agents operate under a "plan-from-scratch" paradigm, regenerating action plans regardless of prior experience. By treating each task in isolation, these agents tend to repeat past errors (Ouyang et al., 2025) and redundantly reason about routine operations such as authentication, search, or navigation (Wang et al., 2024; 2025; Zheng et al., 2025), leading to substantial inference latency and token cost as shown in Figure 1. This inefficiency motivates memory-aware systems that can learn from accumulated experience.

Recent efforts on agent memory store past interactions for retrieval (Tang et al., 2025; Sun et al., 2025). HiAgent (Hu et al., 2025) manages working memory through subgoals, and Synapse (Zheng et al., 2023) retrieves trajectory exemplars, while other approaches learn from failures (Sun et al., 2024; Ouyang et al., 2025) or abstract reusable workflows and skills (Wang et al., 2025; Zheng et al., 2025; Cao et al., 2025; Yang et al., 2025). Our contribution builds on these ideas: mobile GUI reuse requires the complete integration of visual precondition checks, fixed and dynamic actions, and skill lifecycle management. Coordinates, pop-ups, search results, and app layouts can vary across sessions, so a retrieved procedure must be validated and grounded against the current screen before execution.

Motivated by how humans abstract common routines from past experiences and apply such knowledge to guide future activities (Chi et al., 1981), we introduce KNOWACT-MEMORY, a self-evolving skill memory framework for mobile GUI agents operating through test-time learning. Our core insight is that complex mobile tasks can be decomposed into recurring sub-procedures, which can then be abstracted as parametric skills. We propose shifting from online planning to a *State-Grounded Execution* pipeline. To realize this, the framework integrates three components: (i) a parametric skill representation that decouples high-level intent from low-level actions; (ii) a dual-library management system that balances verified skills with exploratory procedures; and (iii) a dual-path skill extraction mechanism that learns from both successful and failed trajectories.

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.

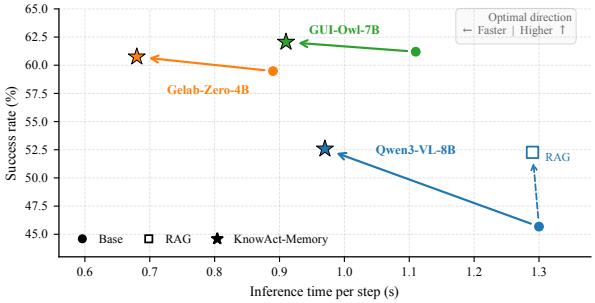


Figure 1: KNOWACT-MEMORY (⋆) enables agents to achieve higher task success rates while reducing inference time across backbones, compared with baselines without memory (●) and with trajectory-as-memory (■). Results are shown on AndroidWorld.

Central to KNOWACT-MEMORY is a structured parametric skill representation, implemented as a JSON object with metadata, action sequences, and parameter slots for dynamic grounding. Skills are organized in a Primary Library for high-reliability procedures and a Candidate Library for experimental ones. During execution, hybrid retrieval combines dense similarity with LLM-based verification to select relevant skills. Static actions are executed directly after visual validation, while dynamic parameters are grounded through lightweight inference. After task completion, the dual-path extraction mechanism either refines skills from successful trajectories or repairs valid prefixes from failures, enabling continuous self-evolution without updating the backbone.

We conduct experiments on mobile-agent benchmarks (Rawles et al., 2024; Kong et al., 2025) and real-device HarmonyOS automation. Results show that KNOWACT-MEMORY reduces inference time and token consumption while improving measured task success rates across GUI backbones. With Qwen3-VL-8B, the framework reduces average inference time by over 25% while achieving a 6.9-percentage-point improvement on AndroidWorld. Further analysis highlights the importance of memory consolidation and extracting candidate skills from unsuccessful trajectories. Across four AndroidWorld evolution rounds, per-step inference time drops from 1.34s to 0.76s while success improves from 75% to 85%. A separate 117-task MobileWorld stream tests evolution at a larger scale, and repeated evaluations characterize run-to-run reliability.

Contributions. We propose KNOWACT-MEMORY, a self-evolving skill memory framework that abstracts both successful and failed mobile GUI interactions into structured parametric skills, combining visual precondition checks, dynamic grounding, and skill lifecycle management in one system. We introduce a *State-Grounded Execution* pipeline that integrates dual-library skill management with semi-parametric execution for efficient and generalizable action planning. Experiments on benchmarks and real devices show that KNOWACT-MEMORY improves task success rates while reducing inference time, supporting continuous self-evolution through test-time learning.

2 RELATED WORK

GUI Agents for Mobile Automation. Recent advances in MLLMs have contributed to generalist agents that interact with digital interfaces (Qin et al., 2025; Yan et al., 2025; Liu et al., 2025). Early approaches relied on accessibility trees or HTML DOMs for element grounding (Wen et al., 2024; Deng et al., 2023). To address noisy metadata, visual-centric agents such as SeeClick (Cheng et al., 2024) and FocusUI (Ouyang et al., 2026) use high-resolution screenshots and refined token selection. For long-horizon tasks, Mobile-Agent-v3 (Ye et al., 2025) adopts multi-role architectures to separate reasoning from execution. AutoGLM (Liu et al., 2024) and ComputerRL (Lai et al., 2025) introduce reinforcement learning paradigms for decision-making in complex environments. Despite these advances, generative GUI agents often face high inference latency. V-Droid (Dai et al., 2025) addresses this with verifier-driven action scoring, while our work focuses on reducing repeated planning through accumulated procedural memory.

Test-time Learning and Self-Evolution. Traditional agents usually execute each task in isolation. Recent research has moved toward test-time learning (TTL), where agents evolve through ongoing interaction. At the architecture level, Titans (Behrouz et al., 2024) introduces a neural long-term memory module that updates weights at test time based on surprise signals. At the agent workflow level, self-evolving memory systems refine procedural knowledge across tasks. Evo-Memory (Wei et al., 2025) evaluates agents' ability to refine memory across streaming tasks. ReMe (Cao et al., 2025) and MUSE (Yang et al., 2025) distill raw trajectories into structured SOPs or parametric skills, and SAGE (Liang et al., 2025) combines reflection with forgetting curves for memory retention. Unlike

static retrieval systems, these approaches support recursive self-improvement. KNOWACT-MEMORY follows this direction but targets mobile GUI execution, where skills must remain executable under visual preconditions, dynamic slots, and evolving app states.

3 METHOD

We present KNOWACT-MEMORY, a self-evolving framework designed to transition mobile GUI agents from ephemeral per-task reasoning to persistent, evolving procedural knowledge. As illustrated in Figure 2, the system operates as a closed-loop cycle: *Observation* → *Retrieval* → *State-Grounded Execution* → *Self-Evolution*.

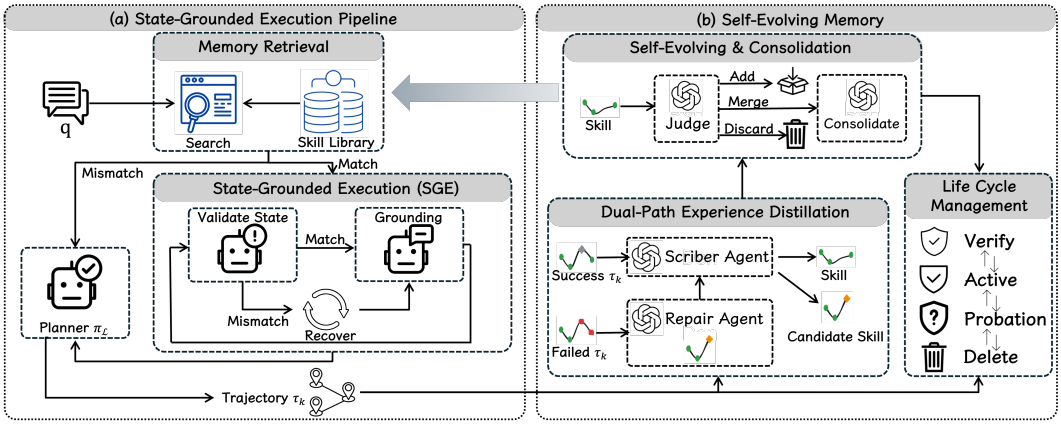


Figure 2: **Overview of the KNOWACT-MEMORY framework.** The online loop retrieves skill schemas, validates preconditions, and grounds dynamic slots for state-grounded execution. The offline loop distills reusable procedures from both successful and failed trajectories, then applies lifecycle management and consolidation to refine the skill library.

3.1 PROBLEM FORMULATION

Agent configuration. We formalize mobile GUI automation as a partially observable control problem. At each time step t, the agent observes a screenshot o_t derived from latent state s_t and generates action a_t given a natural language instruction q. Unlike static agents, KNOWACT-MEMORY conditions the policy on a dynamic skill memory $\mathcal{M}_k$:

$$a_t \sim \pi_\mathcal{L}(a \mid o_{0:t}, a_{0:t-1}, q; \mathcal{M}_k). \tag{1}$$

The memory evolves after each task by distilling interaction history into reusable skills:

$$\mathcal{M}_{k+1} \leftarrow \text{Update}(\mathcal{M}_k, \mathcal{E}_\phi(\tau_k, r_k)), \tag{2}$$

where τ_k is the task trajectory, r_k is the success/failure feedback, and $\mathcal{E}_\phi$ denotes the VLM/LLM-based extraction module. This test-time learning paradigm (Behrouz et al., 2024; Wei et al., 2025) enables inter-task evolution (Gao et al., 2025): the agent transforms ephemeral histories into persistent procedural knowledge without updating backbone parameters.

3.2 PARAMETRIC SKILL SCHEMA

A skill s is a structured, executable procedure represented by the tuple $s = (\Omega, \mathcal{A}, \mathcal{P}, \mathcal{V})$. Metadata Ω contains the name, description, app, status, and success/failure streaks. The action sequence $\mathcal{A} = (a_1, \ldots, a_n)$ stores atomic actions; parameter slots $\mathcal{P}$ store variables such as `{text}`, `{coord}`, and `{app}`; and visual preconditions $\mathcal{V}$ describe the expected GUI state for each step. Each action is formalized as

$$a_i = (\text{type}, \text{element}, \text{value}, \text{valid_state}, \text{desc}, \text{fixed}). \tag{3}$$

The `valid_state` field defines the visual precondition required for execution. The `fixed` field governs execution semantics: fixed actions are executed directly after validation, while dynamic actions require runtime VLM grounding. This semi-parametric design balances the efficiency of invariant operations with the flexibility needed for context-dependent parameters.

3.3 STATE-GROUNDED EXECUTION

Context-aware retrieval. Given a user instruction q, we compute the dense embedding $e_q = \phi(q)$ with KaLM-Embedding-V2 (Zhao et al., 2025). The system retrieves top-k candidates by cosine similarity and then applies LLM-based semantic verification to filter functionally irrelevant matches. If no valid skill is found, the agent executes with standard planning.

Precondition validation and grounding. For a selected skill, SGE validates the current observation o_t against the action precondition v_i:

$$\ell_i = \mathbb{I}\left[\mathcal{L}_{\text{verify}}(o_t, v_i) = \text{true}\right]. \tag{4}$$

This validation is used for grounding correctness, anomaly avoidance, and temporal alignment. If $\ell_i = 1$, fixed actions execute immediately and dynamic actions call a lightweight grounding model. If $\ell_i = 0$, the executor suspends the skill and triggers subgoal recovery: a constrained planner seeks a short action sequence from the current state to the required precondition v_i. After a skill finishes, control is yielded back to the main planner-executor, which checks completion against the original instruction. Thus, skills act as accelerators for recurring prefixes while leaving long-tail task-specific actions to online planning.

3.4 SELF-EVOLVING TEST-TIME LEARNING

Success: semantic densification. For successful trajectories τ_{success}, a Scriber Agent performs trajectory refinement and semantic densification: it removes non-causal actions, rewrites app launch behavior as `AWAKE(app)`, abstracts literals into slots, and annotates visual preconditions:

$$s_{\text{new}} = \mathcal{L}_{\text{scriber}}(\tau_{\text{success}}, q). \tag{5}$$

The resulting skill is a compact executable sequence rather than a raw transcript.

Failure: counterfactual repair. Execution failures are converted into candidate skills through a repair-then-abstract pipeline. A Repair Agent identifies the first divergence step t_{err}, keeps the valid prefix $\tau_{0:t_{\text{err}}-1}$, and synthesizes a corrective suffix τ_{fix} for the implied subgoal:

$$\tau_{\text{fix}} = \mathcal{L}_{\text{repair}}(\tau_{0:t_{\text{err}}-1}, o_{\text{err}}, q). \tag{6}$$

The hypothetical corrected trajectory

$$\tau'_{\text{corrected}} = \tau_{0:t_{\text{err}}-1} \oplus \tau_{\text{fix}} \tag{7}$$

is then passed through the Scriber Agent for the same de-redundancy, parameter abstraction, and state annotation as successful traces. Repaired skills enter the Candidate Library rather than the trusted Primary Library until later executions validate them. The suffix is an offline hypothesis; promotion requires $\theta = 3$ successful real executions, and repeated failures remove a candidate.

Lifecycle management and consolidation. To address the stability-flexibility trade-off, skills are managed by a Primary Library $\mathcal{M}_P$ for trusted procedures and a Candidate Library $\mathcal{M}_C$ for experimental recovery skills. Each skill maintains success and failure streaks (k^+, k^-). With threshold θ, repeatedly successful candidates are promoted, repeatedly failing candidates are discarded, and primary skills with repeated failures are demoted or revised. Incoming skills are compared with existing skills above a semantic threshold σ; a Consolidation Agent returns MERGE, KEEP, or DISCARD. MERGE generalizes shared action topology with different literals or parameters, while DISCARD suppresses lower-quality duplicates.

This lifecycle policy is deliberately conservative. A repaired skill may identify the right local correction while still failing to complete the full instruction because of later uncertainty. Storing it in $\mathcal{M}_C$ gives the system a way to reuse and test the repair without granting it the same authority as a repeatedly validated skill. Conversely, merge decisions are not based only on embedding similarity: the Consolidation Agent checks whether two procedures share action topology and differ mainly in literals, parameters, or binary state. This distinction is important for mobile GUIs, where semantically similar instructions can require incompatible screens, and visually similar screens can serve different tasks.

4 EXPERIMENTS

We evaluate KNOWACT-MEMORY to address three core research questions: **(RQ1) Efficiency vs. efficacy:** can KNOWACT-MEMORY enable efficient test-time scaling while maintaining or improving success rates? **(RQ2) Evolution dynamics:** how does the skill library evolve over continuous task streams, and does it support inter-task transfer? **(RQ3) Component attribution:** how do consolidation and failure-derived skills affect the gains from state-grounded execution?

4.1 SETUP

Benchmarks. We use three benchmarks across different complexity regimes. **AndroidWorld** (Rawles et al., 2024) is a dynamic execution testbed with 116 programmatic tasks across 20 real-world apps and reward signals derived from system state changes. **MobileWorld** (Kong et al., 2025) contains high-complexity mobile tasks with long horizons and frequent cross-application workflows; we adopt its GUI tasks for evaluation. To assess robustness in commercial mobile environments, we also construct a **HarmonyOS real-device benchmark** with 50 instructions covering system utilities and third-party apps, including multimedia, social commerce, knowledge sharing, and service tools. The self-evolution study uses a fixed 20-task AndroidWorld stream across 17 apps, while a disjoint 20-task held-out set tests transfer beyond the source stream. A separate MobileWorld evolution study uses 117 GUI-only tasks across 20 apps with Qwen3.5-397B-A17B, native evaluators, and a 50-step limit. Its no-memory baseline is followed by two evolution rounds; AndroidWorld instead counts its cold-start execution as Round 1.

Models and baselines. We benchmark KNOWACT-MEMORY with a spectrum of GUI backbones: Qwen3-VL-8B-Instruct (Bai et al., 2025), GUI-Owl-7B (Ye et al., 2025), and GELab-Zero-4B (Team, 2025; Yan et al., 2025). For each applicable backbone, we compare three configurations: (1) **Base**, which executes tasks with standard Chain-of-Thought planning and no persistent memory (Yao et al., 2023); (2) **Standard RAG**, which injects relevant trajectory traces into the prompt as in-context examples (Zheng et al., 2023); and (3) **Memory-Augmented**, where the same executor is equipped with KNOWACT-MEMORY. We report success rate (SR), mean output tokens, per-step inference time, and average steps. On HarmonyOS, task success is judged by Qwen3-VL-Plus and validated against manual labels as summarized in Table 7.

Implementation details. For Qwen3-VL and GUI-Owl, we follow the Mobile-Agent-v3 prompt structure and action protocol (Ye et al., 2025). For GELab-Zero, we follow the public implementation and preserve its summary-only history format. The online executor is always the evaluated GUI backbone; the default offline Scriber, Repair, and Consolidation agents use GPT-4o. We set the lifecycle streak threshold to $\theta = 3$, consolidation similarity to $\sigma = 0.75$, and retrieval top-k to 3. We test multiple RAG prompts and report Qwen3-VL RAG results in the main table. In our tests, specialized GUI backbones sometimes interpret retrieved trajectory examples as the current state. Sections A.3, A.4 and B.3 report auxiliary-model sensitivity, transfer examples, and execution diagnostics.

4.2 MAIN RESULTS

Table 1 presents the comparative results across benchmarks. Overall, KNOWACT-MEMORY improves the reported success–cost trade-off. The clearest efficiency effect appears on AndroidWorld. For Qwen3-VL-8B, per-step inference time drops from 1.30s to 0.97s, a 25.4% reduction, while SR improves from 45.68% to 52.58%. This differs from the RAG baseline, which slightly improves SR but increases tokens and steps because the backbone still needs to reason over retrieved trajectory text. GELab-Zero-4B similarly reduces latency from 0.89s to 0.68s, and GUI-Owl-7B reduces token usage while improving SR. These gains are consistent with the intended role of parametric skill instantiation: recurring sub-tasks can be executed through validated action schemas instead of full visual replanning.

The trend varies by benchmark. MobileWorld contains longer-horizon tasks and more cross-app workflows, so KNOWACT-MEMORY does not remove the need for planning; it mainly accelerates reusable subgoals and slightly reduces trajectory length. The GELab-Zero-4B results on MobileWorld show the same direction of improvement in SR, tokens, time, and steps. On HarmonyOS, latency gains are smaller because the baseline prompt is already compressed for real-device testing, but

Table 1: Main results across AndroidWorld, MobileWorld, and real HarmonyOS devices. KNOWACT-MEMORY improves success while reducing token and latency costs in these evaluations.

Backbone	Memory	SR (%) ↑	Tokens ↓	Time (s) ↓	Steps ↓
AndroidWorld					
Qwen3-VL-8B	None	45.68	1910.98	1.30	11.93
	RAG	48.27	2387.68	1.29	14.16
	KNOWACT-MEMORY	**52.58**	**1873.54**	**0.97**	**11.84**
GELab-Zero-4B	None	59.48	1800.98	0.89	13.08
	KNOWACT-MEMORY	**60.75**	**1681.01**	**0.68**	**12.94**
GUI-Owl-7B	None	61.20	1623.42	1.11	10.98
	KNOWACT-MEMORY	**62.06**	**1483.47**	**0.91**	**10.84**
MobileWorld					
Qwen3-VL-8B	None	10.20	3010.20	5.39	25.82
	KNOWACT-MEMORY	**12.67**	**2739.17**	**4.73**	**25.64**
GELab-Zero-4B	None	9.45	4364.16	5.57	28.02
	KNOWACT-MEMORY	**10.95**	**4159.43**	**5.04**	**27.74**
HarmonyOS					
Qwen3-VL-8B	None	64.00	766.28	0.81	9.29
	RAG	64.00	745.19	0.80	9.23
	KNOWACT-MEMORY	**68.00**	**719.18**	**0.78**	**8.94**
GELab-Zero-4B	None	74.00	724.46	1.37	8.69
	KNOWACT-MEMORY	**78.00**	**683.73**	**1.34**	**7.89**

KNOWACT-MEMORY improves success and token cost for both Qwen3-VL-8B and GELab-Zero-4B. These results support the main claim that skill memory can reduce repeated reasoning while preserving online grounding for task-specific decisions. Offline extraction cost, auxiliary-model sensitivity, and SGE mismatch diagnostics are reported in Sections A.2 to A.4.

4.3 MEMORY BASELINES AND REPEATED-RUN EVALUATION

Table 2: Memory baselines on AndroidWorld with the same Qwen3-VL-8B-Instruct executor.

Method	SR (%) ↑	Steps ↓
SkillWeaver (Zheng et al., 2025)	37.07	11.64
ReasoningBank (Ouyang et al., 2025)	46.55	12.12
Agent-Workflow-Memory (Wang et al., 2025)	48.28	12.43
KNOWACT-MEMORY (Ours)	**52.58**	11.84

Comparison with procedural and failure-aware memory. Table 2 compares SkillWeaver (Zheng et al., 2025), ReasoningBank (Ouyang et al., 2025), and Agent-Workflow-Memory (Wang et al., 2025) under the same AndroidWorld setting. KNOWACT-MEMORY achieves the highest SR of 52.58%, outperforming the three baselines by 15.51, 6.03, and 4.30 percentage points, respectively. SkillWeaver takes slightly fewer steps, whereas KNOWACT-MEMORY uses fewer steps than both text-memory baselines. These results demonstrate that KNOWACT-MEMORY achieves stronger task success while maintaining competitive step counts, supporting the effectiveness of combining state-grounded execution with skill management. Adaptation details of baselines are in Section A.5.

Repeated-run evaluation. We evaluate each setting over three runs on 116 AndroidWorld tasks with GUI-Owl-7B and 117 MobileWorld GUI-only tasks with Qwen3.5-397B-A17B (Table 3). On

Table 3: Three-run evaluation. SR is mean ± sample standard deviation in percentage points. Pass@3 and Pass^3 are the percentages of tasks solved in at least one and in all three runs, respectively.

Benchmark / Executor	Memory	SR (%)	Pass@3	Pass^3
AndroidWorld / GUI-Owl-7B	None	58.91 ± 2.17	66.38	45.69
	KnowAct-Memory	60.63 ± 1.32	68.97	49.14
MobileWorld / Qwen3.5-397B-A17B	None	43.30 ± 1.97	58.1	29.1
	KnowAct-Memory	46.15 ± 2.26	59.8	32.5

AndroidWorld, KnowAct-Memory raises mean SR from $58.91 \pm 2.17\%$ to $60.63 \pm 1.32\%$. On MobileWorld, mean SR increases from $43.30 \pm 1.97\%$ to $46.15 \pm 2.26\%$, a gain of 2.85 percentage points. Both settings also improve Pass@3 and Pass^3, showing that memory increases the proportion of tasks solved at least once and the proportion solved in all three runs.

4.4 Self-evolution dynamics

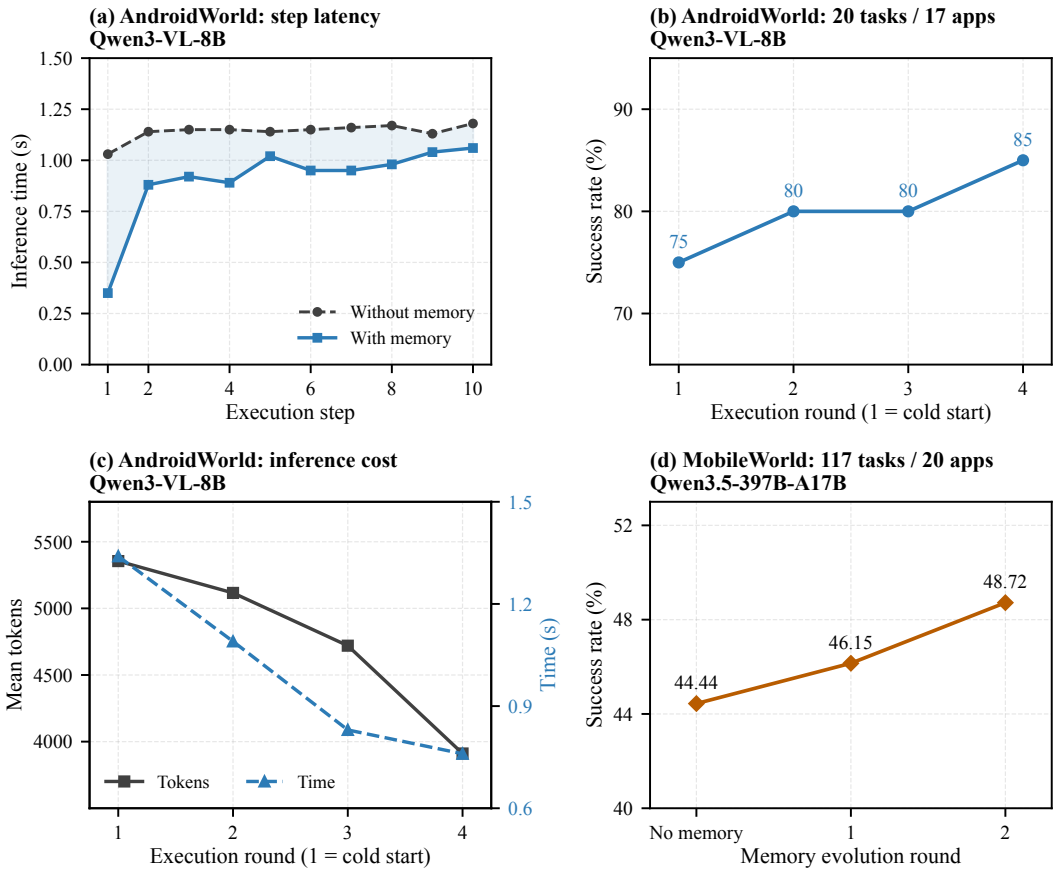


Figure 3: Self-evolution in two separate settings. (a–c) AndroidWorld uses Qwen3-VL-8B on 20 tasks across 17 apps, with Round 1 as the cold start. (d) MobileWorld uses Qwen3.5-397B-A17B on 117 tasks across 20 apps, comparing no memory with two evolution rounds. These curves track evolving libraries, not the repeated evaluations in Table 3.

Figure 3(a–c) evaluates test-time learning over four continuous execution rounds on a fixed 20-task AndroidWorld stream spanning 17 apps. Round 1 is a cold-start run, while later rounds iteratively refine the skill library by consolidating fresh traces with prior procedural knowledge.

Table 4: Component ablation on AndroidWorld in separate runs (SR, %). Cons.: consolidation; Fail. Ext.: failure extraction.

SGE	Cons.	Fail. Ext.	GUI-Owl SR	Qwen3-VL SR
✗	✗	✗	60.34	45.68
✓	✗	✗	53.45	45.25
✓	✓	✗	56.90	49.14
✓	✓	✓	**62.06**	**52.58**

Stepwise efficacy growth. The success rate rises from 0.75 in the cold-start phase to 0.85 in Round 4. The curve does not only stabilize; it continues to improve across later rounds, suggesting that the library moves from accumulating frequent skills to refining more complex procedures.

Inverse cost correlation. As efficacy improves, resource consumption decreases. Mean token consumption drops from roughly 5.36k to 3.91k, and average inference time falls from 1.34s to 0.76s. This inverse relation indicates that the mature skill library replaces repeated CoT reasoning with parametric execution rather than adding extra prompt context.

Granular latency analysis. The step-wise latency profile localizes the speedup to early task prefixes. Reusable routines such as app launching and navigation accelerate the first several steps, while later task-specific steps remain closer to the baseline because grounding and verification are still required. The schema fields in Table 10 explain why this is not raw replay: fixed actions execute only after visual validation, dynamic slots are grounded online, and lifecycle evidence governs promotion, merging, and discard decisions. During evolution, the library performs 28 add, 26 merge, and 1 discard operations, ending with 25 primary and 7 candidate skills. To test whether the gains are only self-task replay, we apply the evolved library to a disjoint 20-task held-out set, where SR improves from 45% to 65% (see reuse examples in Section B.3).

Larger task stream. In the separate MobileWorld evolution experiment, Qwen3.5-397B-A17B improves from 44.44% without memory to 46.15% and 48.72% after two evolution rounds across 117 tasks and 20 apps (Figure 3(d)). The 4.28-point gain complements the smaller AndroidWorld stream.

4.5 Ablation Study

Table 4 investigates the contribution of individual components on AndroidWorld. The results show a U-shaped performance trajectory, indicating that memory augmentation is non-trivial. Here, SGE alone uses unmerged success-only skills. Adding SGE without consolidation reduces GUI-Owl SR from 60.34% to 53.45%, and gives little benefit for Qwen3-VL. In this naive-memory setting, the library contains redundant or under-specified skills, and the retrieved procedure can push the agent into a suboptimal trajectory. Consolidation recovers performance by merging redundant procedures and discarding inferior ones. Failure extraction then adds candidate repairs for recurring execution pitfalls, producing the highest SR for both GUI-Owl and Qwen3-VL.

The failure audit indicates that partially applicable, overly coarse skills are a more common problem than unrelated retrieval. Local repair success also need not imply task completion, motivating execution-based candidate promotion (Section A.4).

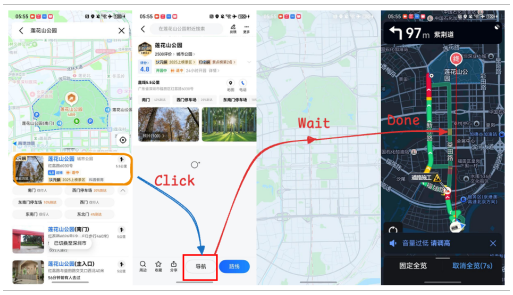
4.6 Case Study: Skill Evolution Across Rounds

Figure 4 follows the request *"Navigate to Lianhuashan Park by subway using Amap"* across three executions. The destination stays the same, while the reused procedure is refined to satisfy the requested travel mode.

Initial exploration and incomplete reuse. In Round 1, exploration under visual complexity fails to complete the request. Round 2 retrieves a related navigation procedure and reaches the destination page, but proceeds through the navigation entry without selecting the subway route. Reaching a navigation screen is insufficient: the requested mode is part of the task goal. This case exposes a limitation of a partially applicable skill even when its destination and app match the instruction.



(a) Round 1: initial exploration fails under visual complexity.



(b) Round 2: incomplete skill reuse misses route selection.



(c) Round 3: repaired skill completes navigation.

Figure 4: Self-evolution on an Amap navigation task.

Repair and validated reuse. In Round 3, the repaired procedure opens route selection, chooses a subway itinerary, and starts the corresponding navigation flow. The correction preserves the useful search and destination-selection prefix while refining the action sequence after it. Failure-derived procedures enter the Candidate Library and require subsequent successful executions before promotion. The example connects the repair mechanism to the multi-round improvements in Figure 3.

Complementary execution cases. Appendix Section C.1 shows how precondition checks handle temporary pop-ups and how dynamic slots reuse a search procedure across different requests. These mechanisms support reuse within a run, while the Amap case shows how the stored procedure changes across runs. Repair diagnostics and transfer examples appear in Sections A.4 and B.3.

5 Limitations

KnowAct-Memory is evaluated on a finite set of mobile tasks and environments. Broader generalization across applications and evolving interfaces, together with sustained learning over longer deployment periods, remains to be established. Skill reuse also depends on the backbone's visual grounding and planning capabilities: stored procedures can reduce repeated reasoning, but unfamiliar states still require online adaptation. As experience accumulates, maintaining a compact, reliable library will require further study of retrieval and forgetting. These directions motivate future work on robust, long-term skill memory for mobile agents.

6 Conclusion

We present KnowAct-Memory, a self-evolving skill memory framework that converts mobile GUI interaction traces into semi-parametric, state-grounded skills. By combining validated execution, failure-derived candidate skills, and dual-library consolidation, KnowAct-Memory improves measured success while lowering online inference cost across the evaluated benchmarks and devices. These findings suggest that experience-driven skill memory complements increasingly capable GUI-agent backbones and provides a foundation for further research on long-term memory and continual self-evolution in mobile GUI agents.

AI USE STATEMENT

AI tools assisted with manuscript preparation, including language polishing, LaTeX formatting, and consistency checks. The research itself uses LLMs for mobile GUI task execution and for skill extraction, repair, and consolidation, as described in the method and experimental protocol. The authors take full responsibility for the accuracy and integrity of the manuscript, including its experimental results, interpretations, and conclusions.

REPRODUCIBILITY STATEMENT

The benchmark settings, model configurations, skill lifecycle parameters, and evaluation protocol are described in Section A.5 and Appendix B.1. The appendix also provides the task stream, skill schema, prompts, transfer checks, and qualitative cases.

REFERENCES

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, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. *arXiv preprint arXiv:2512.10696*, 2025.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Li YanTao, Jianbing Zhang, and Zhiyong Wu. Seeclick: Harnessing gui grounding for advanced visual gui agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024.

Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5:121–152, 1981. doi: 10.1207/s15516709cog0502_2. URL https://doi.org/10.1207/s15516709cog0502_2.

Gaole Dai, Shiqi Jiang, Ting Cao, Yuanchun Li, Yuqing Yang, Rui Tan, Mo Li, and Lili Qiu. Advancing mobile gui agents: A verifier-driven approach to practical deployment. *arXiv preprint arXiv:2503.15937*, 2025.

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. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025.

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

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. *arXiv preprint arXiv:2508.14040*, 2025.

Xuechen Liang, Meiling Tao, Yinghui Xia, Jianhui Wang, Kun Li, Yijin Wang, Yangfan He, Jingsong Yang, Tianyu Shi, Yuantao Wang, et al. Sage: Self-evolving agents with reflective and memory-augmented abilities. *Neurocomputing*, pp. 130470, 2025.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, et al. Autoglm: Autonomous foundation agents for guis. *arXiv preprint arXiv:2411.00820*, 2024.

Zhaoyang Liu, Jingjing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Xuan Dong, Yue Yu, Chenyu Lu, YunXiang Mo, Yao Yan, Zeyue Tian, Xiao Zhang, Yuan Huang, Yiqian Liu, Weijie Su, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. Scalecua: Scaling open-source computer use agents with cross-platform data, 2025. URL https://arxiv.org/abs/2509.15221.

Mingyu Ouyang, Kevin Qinghong Lin, Mike Zheng Shou, and Hwee Tou Ng. Focusui: Efficient ui grounding via position-preserving visual token selection. *arXiv preprint arXiv:2601.03928*, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. Ui-tars: Pioneering automated gui interaction with native agents, 2025. URL https://arxiv.org/abs/2501.12326.

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

Hao Sun, Yong Jiang, Bo Wang, Yingyan Hou, Yan Zhang, Pengjun Xie, and Fei Huang. Retrieved in-context principles from previous mistakes. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8155–8169, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.465. URL https://aclanthology.org/2024.emnlp-main.465/.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. Seagent: Self-evolving computer use agent with autonomous learning from experience, 2025. URL https://arxiv.org/abs/2508.04700.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi Wang, and Wangchunshu Zhou. Agent kb: Leveraging cross-domain experience for agentic problem solving, 2025. URL https://arxiv.org/abs/2507.06229.

GELab Team. Gelab-zero: An advanced mobile agent inference system, 2025. URL https://github.com/stepfun-ai/gelab-zero.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710, 2024.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=NTAhi2JEEE.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 543–557, 2024.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, Shiliang Yang, Zhirui Wang, Brian Li, Kang An, Chenyang Li, Lei Lei, Mengmeng Duan, Danxun Liang, Guodong Liu, Hang Cheng, Hao Wu, Jie Dong, Junhao Huang, Mei Chen, Renjie Yu, Shunshan Li, Xu Zhou, Yiting Dai, Yineng Deng, Yingdan Liang, Zelin Chen, Wen Sun, Chengxu Yan, Chunqin Xu, Dong Li, Fengqiong Xiao, Guanghao Fan, Guopeng Li, Guozhen Peng, Hongbing Li, Hang Li, Hongming Chen, Jingjing Xie, Jianyong Li, Jingyang Zhang, Jiaju Ren, Jiayu Yuan, Jianpeng Yin, Kai Cao, Liang Zhao, Liguo Tan, Liying Shi, Mengqiang Ren, Min Xu, Manjiao Liu, Mao Luo, Mingxin Wan, Na Wang, Nan Wu, Ning Wang, Peiyao Ma, Qingzhou Zhang, Qiao Wang, Qinlin Zeng, Qiong Gao, Qiongyao Li, Shangwu Zhong, Shuli Gao, Shaofan Liu, Shisi Gao, Shuang Luo, Xingbin Liu, Xiaojia Liu, Xiaojie Hou, Xin Liu, Xuanti Feng, Xuedan Cai, Xuan Wen, Xianwei Zhu, Xin Liang, Xin Liu, Xin Zhou, Yingxiu Zhao, Yukang Shi, Yunfang Xu, Yuqing Zeng, Yixun Zhang, Zejia Weng, Zhonghao Yan, Zhiguo Huang, Zhuoyu Wang, Zheng Ge, Jing Li, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, and Daxin Jiang. Step-gui technical report, 2025. URL https://arxiv.org/abs/2512.15431.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, et al. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

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

Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, Youcheng Pan, Yang Xiang, Meishan Zhang, Haofen Wang, Jun Yu, Baotian Hu, and Min Zhang. Kalm-embedding-v2: Superior training techniques and data inspire a versatile embedding model, 2025. URL https://arxiv.org/abs/2506.20923.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. *arXiv preprint arXiv:2306.07863*, 2023.

A Additional Analyses and Evaluation Protocol

A.1 Same-extractor controls

Table 5 fixes the offline extractor to Qwen3.5-35B-A3B. The Qwen controls compare skill reuse under fixed surrounding agent configurations; in the 35B setting, both conditions retain host orchestration and textual experience memory.

Table 5: Qwen skill-reuse controls on 117 MobileWorld tasks with Qwen3.5-35B-A3B extraction. SR is reported in percent.

Online executor	Without skills	With skills	Δ
Qwen3.5-35B-A3B	34.5	37.9	+3.4
Qwen3.5-397B-A17B	43.3	46.2	+2.9

A.2 Offline extraction cost

Using Qwen3.5-35B-A3B with `thinking=False`, offline skill extraction consumes an average of 7,133.89 tokens per trajectory, equivalent to 3.15% of the average token usage for executing a complete MobileWorld task (226,702.45 tokens).

A.3 Auxiliary LLM sensitivity

Table 6: Auxiliary-model sensitivity on a 25-task subset spanning 17 apps. Results are averaged over repeated runs with the same online GUI backbone.

Offline auxiliary LLM	SR (%)	Avg. steps	Time (s/step)
GPT-5.4	66	9.92	1.03
GPT-4o	64	9.64	1.27
Qwen3.5-Plus	62	9.76	1.12
Qwen3.5-27B	60	9.32	1.33

The auxiliary LLM performs offline extraction, repair, and consolidation; the online executor is fixed to Qwen3-VL-8B. Model versions are `gpt-5.4-2026-03-05`, `gpt-4o-2024-11-20`, `qwen3.5-plus-2026-02-15`, and self-deployed `Qwen/Qwen3.5-27B`.

A.4 Additional validation

Retrieval and repair. Among 48 failed tasks with naive memory, 23 involve retrieved skills: 2 are clear mis-retrievals and 21 use related but overly coarse procedures. In a separate check of 20 failed Qwen3-VL trajectories, repaired skills achieve 95% local subgoal matching but recover 5/20 complete tasks. This gap motivates execution-based candidate promotion.

State validation. SGE detects observation–precondition mismatches in 15.54% of checked skill steps, mainly due to pop-ups, notifications, layout changes, or grounding deviations.

A.5 Implementation and evaluation protocol

Table 7: Implementation and evaluation protocol.

Item	Setting
Online deployment	Open-weight GUI backbones are served with vLLM on H100 GPUs. The online executor is always the evaluated backbone; no model parameters are updated during test-time evolution.
Offline memory construction	The default Scriber, Repair, and Consolidation agents use GPT-4o. Auxiliary alternatives are evaluated in Table 6. The separate extraction-token measurement uses Qwen3.5-35B-A3B (Section A.2).
Lifecycle settings	Candidate promotion uses success streak threshold $\theta = 3$, consolidation uses semantic threshold $\sigma = 0.75$, and retrieval uses top-$k = 3$.
HarmonyOS judge	Qwen3-VL-Plus judges final success. On 100 author-labeled checks, agreement with the judge gives Pearson correlation 0.8316 and Cohen's $\kappa = 0.8271$.

Memory-baseline adaptation. ReasoningBank and Agent-Workflow-Memory inject top-k retrieved memories into the prompt. In the SkillWeaver adaptation, 34 of 49 skills contain fixed-pixel actions, and the executor selects skills from function names and descriptions. AndroidWorld success is measured using programmatic reward signals.

B Additional Experimental Details

B.1 Real-device HarmonyOS benchmark

We evaluate robustness on a physical HarmonyOS device with 50 instructions across 11 commercial applications. Table 8 shows representative tasks.

Table 8: Representative tasks from the HarmonyOS real-device benchmark.

Category	App	Instruction
System	Settings	Turn on Wi-Fi.
	Settings	Set screen brightness to maximum in Settings.
Social commerce	XiaoHongShu	Search for "Beijing Travel Guide" and open the first note.
	Taobao	Search for "Mechanical Keyboard" and filter for items under 500 yuan.
	Meituan	Find the highest-rated BBQ restaurant nearby.
	Meituan	Search for eggs in Xiaoxiang Supermarket and add a "Worry-Free Price" item to cart.
	VMall	Check the price of Mate 80; if below 4500 yuan after discount, add it to cart.
Multimedia	Bilibili	Search "AI Painting Tutorial" and sort by most played.
	Music	Search for Jay Chou's "Dao Xiang" and play it.
	Video	Search for the popular movie "The Wandering Earth 2".
Service tools	Amap	Find nearby restaurants, sort by distance, and navigate to the nearest one.
	Ctrip	Search for flight tickets from Shanghai to Guangzhou for the day after tomorrow.
	Railway 12306	Check high-speed train tickets from Beijing to Tianjin for tomorrow.
	Zhihu	View the Hot List and report the titles of the top 3 items.

B.2 AndroidWorld evolution task stream

The 20-task evolution stream covers 17 apps and multiple domains. The fixed stream is used to measure longitudinal improvement, while the held-out transfer set is disjoint from this stream.

Table 9: AndroidWorld task subset used for self-evolution.

Category	Task ID	Description
System settings	SystemBluetoothTurnOn	Turn on Bluetooth via system settings.
	SystemWifiTurnOff	Turn off Wi-Fi via system settings.
	TurnOffWifiAndTurnOnBluetooth	Turn off Wi-Fi and then turn on Bluetooth.
Communication	ContactsAddContact	Add a new contact.
	SimpleSmsSend	Send a text message to a specific number.
	SimpleSmsResend	Resend the last message in a conversation.
	SimpleCalendarDeleteOneEvent	Delete a specific calendar event.
Media	CameraTakePhoto	Open camera and capture a photo.
	CameraTakeVideo	Open camera and record a video clip.
	AudioRecorderRecordAudio	Record a voice memo.
Productivity	ClockStopWatchRunning	Start the stopwatch.
	FilesMoveFile	Move a file between directories.
	MarkorCreateNote	Create a new text note in Markor.
	SaveCopyOfReceiptTaskEval	Scan or save a receipt copy.
Third-party apps	BrowserMaze	Navigate and complete a browser maze.
	OsmAndFavorite	Mark a location as favorite in OsmAnd.
	ExpenseAddSingle	Log a single expense entry.
	ExpenseDeleteDuplicates	Remove duplicate expense entries.
	RecipeAddSingleRecipe	Add a new recipe.
	RecipeDeleteDuplicateRecipes	Remove duplicate recipes.

B.3 TRANSFER EXAMPLES

The evolved library shows cross-task reuse rather than only self-task replay. In Round 4, `SystemBluetoothTurnOn` succeeds by retrieving a skill evolved from `TurnOffWifiAndTurnOnBluetooth`. We observe similar transfer for `MarkorCreateNoteFromClipboard` from `MarkorCreateNote`, and `SimpleSmsResend` from `SimpleSmsSend`.

C Additional Qualitative Cases

C.1 State recovery and parameter grounding

Figure 5 illustrates two execution mechanisms. In the takeaway task, SGE detects privacy and location pop-ups as precondition violations, handles them as temporary subgoals, and resumes the retrieved procedure only after the entry state is restored. In the video-search task, a single topology with a dynamic `{text}` slot supports different intents by regrounding the parameter from the current instruction and screenshot.



Figure 5: Qualitative execution cases. Left: subgoal recovery handles pop-ups before resuming a takeaway skill. Right: a video-search skill supports different intents through dynamic grounding.

C.2 MULTI-APP TASK EXECUTION VIA SKILL CHAINING

For the composite request *"Order braised pork rice on Meituan, then turn on Do Not Disturb mode, and finally watch 'Let the Bullets Fly' on Bilibili and like it"*, KNOWACT-MEMORY decomposes the instruction into three sub-tasks and chains app-specific skills.



Stage-1: Skill-Meituan_Order_Takeaway

(a) Meituan ordering skill with parameter "Braised Pork Rice".



Stage-2: Skill-Turn_on_no_Disturb_Mode

(b) System setting skill for Do Not Disturb mode.



Stage-3: Skill-Bilibili_Play_Movie_and_Thumb

(c) Bilibili play-and-like skill with a movie parameter.

Figure 6: Cross-app task chain executed through sequential skill retrieval and parameter grounding.

D SKILL REPRESENTATION AND PROMPTS

D.1 SKILL SCHEMA EXAMPLE

Table 10: Execution-facing skill schema fields. The full JSON form appears in Appendix Figure 7.

Field	Execution role
`name`, `description`	Retrieve a candidate procedure for the current instruction and app state.
`valid_state`	Gate each cached action with a visual precondition before execution.
`fixed`	Execute invariant clicks or swipes after the gate succeeds.
`parameters`	Ground `{text}` or `{coord}` from the current task and screenshot.
`success_streak`, `failure_streak`	Promote, merge, demote, or discard skills using execution evidence.

Figure 7 provides a compact example of the semi-parametric skill schema used by KNOWACT-MEMORY, including metadata, lifecycle evidence, dynamic slots, visual preconditions, and executable action fields.

Example: a reusable contact-search skill

```
{
  "name": "search_contact",
  "description": "Search for a contact in messaging app",
  "app": "Messages",
  "success_streak": 3, "failure_streak": 0,
  "parameters": ["coord", "text"],
  "action_sequence": [
    {
      "type": "click", "coordinate": "{coord}",
      "valid_state": "search bar visible",
      "desc": "click the search bar", "fixed": false
    },
    {
      "type": "type", "text": "{text}",
      "valid_state": "search bar activated",
      "desc": "type the contact name", "fixed": false
    },
    {
      "type": "click", "coordinate": "{coord}",
      "valid_state": "contact found",
      "desc": "tap the target contact", "fixed": false
    }
  ]
}
```

Figure 7: Example semi-parametric skill schema.

D.2 VALIDATION AND GROUNDING PROMPTS

(a) State validation prompt

System. You are a helpful GUI agent assistant.
User. Determine whether the current screenshot matches the valid-state description: {valid_state} without interruption or notification.
Output. Return exactly <answer>true</answer> or <answer>false</answer>.

(b) Action grounding prompt

Input. Task: {instruction}. History: {history}.
If completed. Return: <tool>{"arguments": {"action": "terminate", "status": "success"}}</tool>
If not completed. Return: <tool>{"arguments": {"action": "ACTION_TYPE", "point": [x, y]}}</tool>
Coordinates must be integer lists, not strings. Output only the <tool> tag.

D.3 Skill extraction and repair prompts

(c) Skill extraction and generalization prompt

Role. You are an expert in evaluating UI automation tasks to extract generalized, reusable skills.
Input. Instruction {instruction} and observed action responses {action_responses}.
Procedure. Filter redundant actions; convert app-opening behavior to AWAKE(app_name); mark fixed elements and replace dynamic values with slots; generate a skill object with name, description, parameters, valid states, and action sequence.

(d) Failure diagnosis and correction prompt

Role. You are an expert in analyzing failed UI automation tasks.
Input. Instruction {instruction} and failed trajectory {action_responses}.
Procedure. Identify the first divergence step, discard cascading failures, infer the corrective action, and define a valid-state condition for the correction. Output the failure index, corrective action, reason, and correction hint.

(e) Skill comparison and consolidation prompt

Role. Maintain a GUI Agent Skill Library.
Input. Target app {app}, old skill {old_skill}, and new skill {new_skill}.
Decision. Return merge, keep_old, or keep_new. Merge if action topology is shared but literals differ; replace if the new skill is shorter or fixes a failure; keep separate if goals are distinct.