How Far Can Self-Evolving Agents Go?
A Systematic Evaluation across Models, Harnesses, and Benchmarks

Anonymous ACL submission

Abstract

Self-evolving agents (systems that improve through test-time experience without retraining) have proliferated rapidly, yet the field lacks controlled cross-method evaluation: each work tests only its own method on self-selected benchmarks with limited model coverage, making it impossible to determine which paradigm actually works, for which tasks, and with which models. We present the first systematic, multi-factor evaluation across several representative paradigms, frontier LLMs, and benchmarks spanning domain-specific to extreme-difficulty tasks. Our evaluation reveals: (1) self-evolution reliably benefits structured tasks but shows inconsistent effects on knowledge-intensive reasoning and no detectable effect on extreme-difficulty tasks; (2) the inconsistent effects on GPQA are specific to retrieval-augmented methods and absent from non-retrieval methods, demonstrating a dissociation between retrieval-based and non-retrieval evolution; (3) extending evolution scope from prompts to tool code or scaffold logic triggers catastrophic self-amplifying failure loops across all tested models, revealing a sharp reliability boundary between declarative and procedural evolution targets. Batch ordering sensitivity analysis confirms these findings are robust across alternative orderings. Our results provide the empirical foundation needed to guide principled deployment of self-evolving agents.

1 Introduction

Large language model (LLM) based agents have rapidly advanced from passive text generators to autonomous systems capable of tool use, multi-step reasoning, and complex decision-making (Shunyu Yao, 2023; Lei Wang, 2023; John Yang, 2024). Yet a core architectural limitation constrains their practical deployment: current agents are episodic, processing each task in isolation, unable to retain, organize, or leverage experience from prior interactions. This "stateless" design stands in stark contrast to human expertise, which improves continuously through accumulated practice and reflection.

The emerging paradigm of self-evolving agents, systems that autonomously improve task-solving capability through test-time experience without retraining, promises to bridge this gap. Self-evolution harnesses now span a wide spectrum, from prompt-level evolution that refines system instructions based on task feedback (Noah Shinn, 2023), to memory-augmented reasoning that retrieves and leverages structured experience banks (Siru Ouyang, 2026; Runnan Fang, 2026), to skill-level evolution that creates, refines, and composes reusable tool-augmented modules (Ziyu Ma, 2026).

A parallel line of development has emerged in real-world agent ecosystems. Systems like OpenClaw and Hermes demonstrate that self-evolution is not merely an academic construct but an engineering reality: deployed agents maintain persistent memory, accumulate daily experience logs, and operate through skill-based architectures where reusable modules bundle tool definitions, system prompts, and workflow logic into compound, evolving units. The SkillClaw framework (Ziyu Ma, 2026) takes this further by enabling collective skill evolution across multi-user agent ecosystems. These developments highlight a crucial observation: self-evolution in practice operates at a compound level that interleaves prompt refinement, memory accumulation, and tool-code

creation. However, it remains unclear which subcomponent is actually responsible for the observed improvements.

Despite the rapid proliferation of self-evolution methods, the evaluation practices across existing works remain highly inconsistent, creating a critical barrier to scientific progress. Reflexion (Shinn et al., 2023) evaluates only on ALFWorld and HotpotQA with a single model (GPT-4) using per-task retry; ACE (Wang et al., 2024b) tests on 3 benchmarks with 2 models under sequential online batches; ReasoningBank (Zhang et al., 2025) and Mem-p (Zhao et al., 2024) each select different benchmarks and models with incompatible evolution protocols. More recent works further illustrate this fragmentation: Live-SWE-agent (Chen et al., 2025) and EvoCodeBench (Zhang et al., 2026) focus exclusively on coding tasks; Evo-Memory (Wei et al., 2025) and AutoSkill (Yang et al., 2026) propose new benchmarks but prioritize defining evaluation environments over cross-method comparison. This fragmented landscape-where each work evaluates only its own method on self-selected tasks with limited model coverage-makes it impossible for the community to answer the most basic question: which self-evolution paradigm actually works, for which tasks, and with which models? No existing study places structurally distinct methods under identical experimental conditions to enable fair comparison.

Three critical questions motivate this study: (1) Do specialized self-evolution harnesses deliver consistent improvement across diverse models, tasks, and difficulty levels? (2) What factors within evolution harnesses correlate with improvement? (3) Under what task conditions is self-evolution reliably beneficial versus harmful? These questions remain unanswered because prior work evaluates harnesses in isolation against their own baselines, tests only moderate-difficulty tasks, and lacks cross-harness comparison under controlled conditions with identical models, benchmarks, and harnesses.

Our work is primarily an empirical contribution. We do not propose new self-evolution algorithms; rather, we provide the first large-scale controlled comparison that places structurally distinct self-evolution paradigms under identical experimental conditions. By contrasting retrieval-augmented and non-retrieval harnesses, we isolate inference-time context injection as a key factor driving inconsistent effects, extending and contextualizing the ICL/RAG distraction literature in the self-evolving agent setting, while the structured-task positive results and the non-retrieval/retrieval dissociation go beyond what prior work predicts.

1.1 Contributions

Our primary contribution is empirical: a controlled evaluation framework and the findings it yields, not new algorithms. Specifically:

(1) Controlled evaluation at scale. A sequential batch protocol comparing 3 harnesses $\times$ 3 models $\times$ 4 benchmarks under large-scale evaluation, with a coding-agent baseline (Claude Code), tool/scaffold evolution failure analysis, context-management ablation, and batch-ordering sensitivity analysis.

(2) Mechanistic dissociation. By contrasting retrieval-augmented (Siru Ouyang, 2026; Runnan Fang, 2026) and non-retrieval (Qizheng Zhang, 2026) harnesses, we empirically demonstrate that the large, model-dependent instability on GPQA is exclusive to retrieval-augmented harnesses, while the non-retrieval harness produces only small, mixed effects.

(3) Three empirical observations:

Observation 1 (Task-type dependence): Self-evolution reliably benefits structured tasks but produces inconsistent effects on knowledge-intensive reasoning and no detectable effects on extreme-difficulty tasks.

Observation 2 (Retrieval-augmented instability): On GPQA-Diamond, retrieval-augmented harnesses (Siru Ouyang, 2026; Runnan Fang, 2026) produce large, model-dependent effects (range: -19.7 to $+7.6$ pp), while the non-retrieval harness (Qizheng Zhang, 2026) produces only small, mixed effects (range: -5.6 to $+2.5$ pp). This dissociation identifies inference-time context injection as the distinguishing architectural factor.

Observation 3 (Declarative vs. procedural evolution boundary): Extending evolution scope from prompts to tool code or scaffold logic produces universal failure: 17/18 runs enter infinite debugging loops, and the single completed run shows catastrophic degradation (-21.5 pp). Evolution reliability degrades sharply at the declarative-procedural boundary.

2 Related Work

2.1 Self-Evolution Paradigms

Self-evolution harnesses span three paradigms. **Prompt-level:** ACE (Qizheng Zhang, 2026) refines system prompts via high-level experience extraction; Reflexion (Noah Shinn, 2023) uses verbal self-reflection; EvolveR (Rong Wu, 2026) distills strategic principles. These are lightweight but coarse-grained. **Memory-level:** ReasoningBank (Siru Ouyang, 2026) retrieves structured reasoning traces as few-shot examples; Mem-p (Runnan Fang, 2026) combines memory retrieval with prompt augmentation; MEM-α (Yu Wang, 2025) learns memory construction via RL; MemAgent (Hongli Yu, 2025) synergizes internal and external memory. **Skill/code-level:** SkillClaw (Ziyu Ma, 2026) enables collective skill evolution across users; SKILLRL (Peng Xia, 2026) grows hierarchical skill banks via RL; AlphaEvolve (Alexander Novikov and Balog, 2025) discovers algorithms through evolutionary code generation; Meta-Harness (Yoonho Lee, 2026) evolves evaluation harnesses alongside capabilities. Our protocol differs from Meta-Harness by enforcing strict isolation between evolution-visible and held-out evaluation data.

2.2 Continual Learning and ICL/RAG Distraction

Our protocol draws on continual learning (James Kirkpatrick, 2016; David Lopez-Paz, 2017; Gido M. van de Ven, 2019). Unlike classical forgetting via weight interference, self-evolving agents face *context pollution*: accumulated experience degrades performance by competing for context window capacity. Our GPQA results (Section 5) provide empirical evidence of this failure mode.

Prior work shows injected context can harm capable models: Freda Shi (2023) demonstrate distraction by irrelevant context, Sewon Min (2022) show format effects in ICL, and Florin Cuconasu (2024) find retrieval noise disproportionately degrades strong models. Our GPQA-Diamond findings are directionally consistent with this literature.

2.3 Evaluation Practices in Self-Evolving Agent Work

Although no unified self-evolution benchmark exists, each method-proposing work includes evaluation experiments with incompatible settings. Prompt-level methods (Noah Shinn, 2023; Qizheng Zhang, 2026; Marc-Antoine Allard, 2026) each test 12 models on self-selected benchmarks without cross-method baselines; memory methods (ReasoningBank, Mem-p) use different benchmarks and models with incompatible protocols; skill/coding methods (Chunqiu Steven Xia, 2025; Jia Li, 2024; Yutao Yang, 2026) restrict to narrow domains; and the dedicated EvoMemory benchmark (Tianxin Wei, 2025) covers only memory-augmented methods. Common deficiencies include: (1) single-method evaluation without cross-paradigm baselines; (2) benchmark selection biased toward each method's strengths; (3) limited model coverage (typically 12 models); and (4) absence of robustness analysis on evaluation protocols. Our work addresses all four gaps through controlled multi-factor comparison under identical conditions.

3 Methodology

3.1 Sequential Batch Evaluation Protocol

We adopt a sequential batch evaluation protocol (Figure 1) that processes tasks in fixed-order batches with evolution between batches. Our evaluation uses a fixed dataset processed in sequential segments. While sequential batch evaluation is established in continual learning literature (Section 2.4), we apply it here to enable fair cross-harness comparison of self-evolution harnesses under identical conditions.

Step 1: Batch Presentation. The full evaluation dataset is divided into sequential batches $B_1, B_2, \ldots, B_k$. The harness processes one batch at a time.

Step 2: Evolution. After processing batch B_i (receiving ground-truth feedback), the harness evolves its context, tools, or internal structure based on the accumulated experience from batches B_1 through B_i.

Step 3: Prediction. The evolved harness (checkpoint after processing B_i) predicts on the next batch B_{i+1}.

Step 4: Cumulative Evaluation. The final reported accuracy is computed over all samples across all batches.

This protocol has two important implications: (1) evolution improvements are measured incrementally as they accumulate, providing a realistic estimate of test-time gains; (2) early-stage evo-

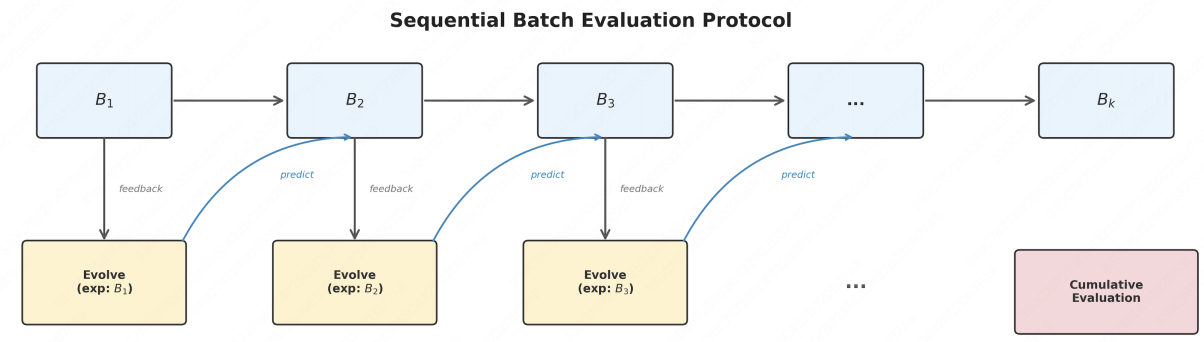

Figure 1: The tested agents process tasks in fixed-order batches with evolution between batches

lution errors (before sufficient experience accumulates) are included in the final metric, penalizing harnesses that require many samples to stabilize. To mitigate overfitting risks, our protocol enforces strict separation between evolution-visible data and evaluation data: within each batch, the harness receives ground-truth feedback only after completing predictions, and the final evaluation metric aggregates performance across all batches—including early batches where the harness has not yet evolved. This design prevents the harness from tailoring its evolution to specific test instances, as each prediction is made before feedback is available for that batch. Additionally, the batch-ordering sensitivity analysis (§4.5) confirms that results are not artifacts of a particular data arrangement.

3.2 Self-Evolution Harnesses

We evaluate three specialized self-evolution harnesses and one coding-agent baseline.

3.2.1 ACE: Prompt Self-Evolution via Experience Extraction

ACE implements prompt evolution through high-level experience extraction. After each batch, a meta-agent analyzes the agent's trajectories to identify success patterns and failure modes, distilling them into general principles that are incorporated into the system prompt.

3.2.2 ReasoningBank: Structured Memory with Few-Shot Retrieval

ReasoningBank maintains an external structured repository of reasoning traces. After each batch, successful and failed reasoning traces are stored with metadata (task type, difficulty, key reasoning steps). At inference time on subsequent batches, the most relevant traces are retrieved as few-shot examples based on task similarity. Critically, this retrieval mechanism alters the effective prompting format by injecting few-shot examples, which changes the baseline prompting condition (see Section 5 for detailed analysis of this confound).

3.2.3 Mem-p: Memory-Augmented Prompt Evolution

Mem-p combines elements of both paradigms: it retrieves relevant few-shot examples from a growing memory bank while simultaneously extracting high-level experience for prompt augmentation.

3.2.4 Coding Agent Baseline (Claude Code)

The coding-agent baseline uses Claude Code with access to online data. Unlike the specialized harnesses above, the coding agent autonomously decides how to evolve. Crucially, Claude Code incorporates sophisticated context management strategies (compaction, summarization, pruning) that prevent context window overflow during extended evolution sessions.

Fairness note: We acknowledge an inherent asymmetry in this comparison. Claude Code is an industrial-grade product with extensive engineering investment in context management, tool integration, and prompt engineering. The specialized harnesses (ACE, ReasoningBank, Mem-p) are academic prototypes designed to isolate specific evolution mechanisms. We include Claude Code not as a "fair" competitor but as an upper-bound reference point that reveals what is achievable with substantial engineering, helping identify which capabilities academic prototypes most need to develop. Readers should interpret Claude Code's results as indicative of the potential ceiling rather than as evidence that specialized harnesses are categorically inferior.

3.3 Benchmarks

We evaluate on four benchmarks representing a difficulty spectrum from domain-specific to extreme-difficulty:

τ2-Telecom(Victor Barres, 2025): Domain-specific agent benchmark with structured action spaces and clear success criteria.

ALFWorld(Mohit Shridhar, 2021): Embodied agent benchmark with sequential decision-making in household environments.

GPQA-Diamond(David Rein, 2023): Graduate level science questions requiring deep domain knowledge.

Humanity's Last Exam(Long Phan, 2025): Extremely challenging questions designed to be beyond current AI capabilities.

4 Experiments

4.1 Experimental Setup

All experiments follow the sequential batch evaluation protocol described in Section 3.1. For each harness $\times$ model $\times$ benchmark combination, the evaluation dataset is divided into K equal-sized sequential batches. After processing each batch with ground-truth feedback, the harness evolves its state before predicting the next batch. The final reported accuracy is cumulative across all batches.

For the coding-agent baseline, we evaluate three evolution scope levels: (1) prompt-only self-evolution (most constrained), (2) prompt + tool evolution, and (3) prompt + scaffold evolution (least constrained).

4.2 Main Results and Key Findings

Table 1 and Table 2 present the complete results. We report cumulative accuracy after evolution with the change from baseline in parentheses. Four patterns emerge: (1) *Task-type dependence:* self-evolution universally benefits structured tasks (τ2-Telecom: 12/12 positive; ALFWorld: 12/12 positive, all $\geq$ +3.4 pp) but shows no detectable effect on extreme-difficulty tasks (HLE: all $|\Delta| \leq$ 2.1 pp) and inconsistent effects on knowledge-intensive reasoning (GPQA: $-$19.7 to +7.6 pp). (2) *Retrieval-augmented instability:* the GPQA inconsistency is specific to retrieval-augmented harnesses (ReasoningBank, Mem-p), which produce large model-dependent swings ($-$19.7 to +7.6 pp), while ACE produces only small mixed effects ($-$5.6 to +2.5 pp). (3) *Harness-model interactions:* optimal harness choice is model-dependent (e.g., ACE best for Claude on ALFWorld at +13.4 pp; ReasoningBank best for DeepSeek at +5.4 pp), requiring model-specific validation. (4) *Declarative-procedural boundary:* extending evolution from prompts to tool code or scaffold logic produces universal failure (Table 2): 17/18 runs enter infinite debugging loops, and the single completed run shows $-$21.5 pp degradation, revealing that procedural self-evolution remains beyond current model capabilities.

4.3 Tool and Scaffold Evolution: Failure Analysis

Beyond prompt-only self-evolution, we evaluated two more ambitious evolution scopes within the coding-agent paradigm: (1) prompt + tool evolution, where the agent may create and modify tool implementations; and (2) prompt + scaffold evolution, where the agent may additionally restructure its execution scaffold.

Of 18 attempted runs (2 modes $\times$ 3 models $\times$ 3 benchmarks except τ2-Telecom), only 1 produced a complete result (Table 2). All other runs entered infinite debugging loops: the agent repeatedly attempted to fix self-introduced code bugs without converging. Analysis of failure trajectories reveals a self-amplifying pattern: the agent modifies tool code based on error feedback, introduces new bugs, attempts cascading patches, and fills the context window with debugging traces. The contrast between prompt-only success and tool/scaffold failure suggests that evolution reliability degrades sharply as the target moves from declarative (prompts) to procedural (code).

4.4 Ablation: Context Management on GPQA-Diamond

The ablation compares Claude Code with sophisticated context management including compaction, summarization, and pruning against R2Egym (Naman Jain, 2025), a minimal coding agent with only bash and file_edit tools, no context management. The directional consistency across both models (R2Egym degrades while Claude Code remains stable or positive) provides correlational evidence that context management contributes to evolution stability in the coding-agent paradigm.

Ablation confounds and scope. R2Egym and Claude Code differ in multiple dimensions beyond context management (tools, prompt engineering, system architecture). We cannot strictly isolate context management as the sole causal factor.

Harness	Model	τ2-Telecom	HLE	GPQA-Diamond	ALFWorld
ACE	Claude Opus 4.6	0.985 (+0.022)	0.373 (−0.005)	0.901 (−0.028)	0.798 (+0.134)
	DeepSeek-3.2	0.972 (+0.028)	0.242 (−0.009)	0.692 (−0.056)	0.716 (+0.034)
	GLM-5	0.996 (+0.001)	0.284 (−0.021)	0.885 (+0.025)	0.734 (+0.082)
ReasoningBank	Claude Opus 4.6	0.982 (+0.020)	0.380 (+0.002)	0.444 (−0.197)†	0.787 (+0.105)
	DeepSeek-3.2	0.971 (+0.027)	0.255 (+0.004)	0.806 (+0.059)	0.718 (+0.054)
	GLM-5	0.997 (+0.001)	0.309 (+0.004)	0.723 (−0.137)	0.739 (+0.087)
Mem-p	Claude Opus 4.6	0.981 (+0.019)	0.369 (−0.009)	0.750 (−0.179)	0.792 (+0.110)
	DeepSeek-3.2	0.966 (+0.021)	0.253 (+0.002)	0.823 (+0.076)	0.723 (+0.059)
	GLM-5	0.997 (+0.002)	0.299 (−0.006)	0.696 (−0.165)	0.726 (+0.074)
CC (Claude Code)	Claude Opus 4.6	0.987 (+0.024)	0.382 (+0.004)	0.924 (−0.005)	0.804 (+0.122)
	DeepSeek-3.2	0.977 (+0.033)	0.248 (−0.003)	0.724 (−0.024)	0.732 (+0.068)
	GLM-5	0.999 (+0.003)	0.313 (+0.008)	0.885 (+0.025)	0.724 (+0.072)

Table 1: Main results: sequential batch evaluation (cumulative accuracy, change from baseline). †ReasoningBank's few-shot retrieval protocol alters the effective baseline for Claude on GPQA (effective baseline: 0.641 vs. standard 0.929). See Section 5 for detailed analysis. CC additionally attempted prompt+tool and prompt+scaffold evolution modes, but all models entered infinite debugging loops and failed to complete evaluation.

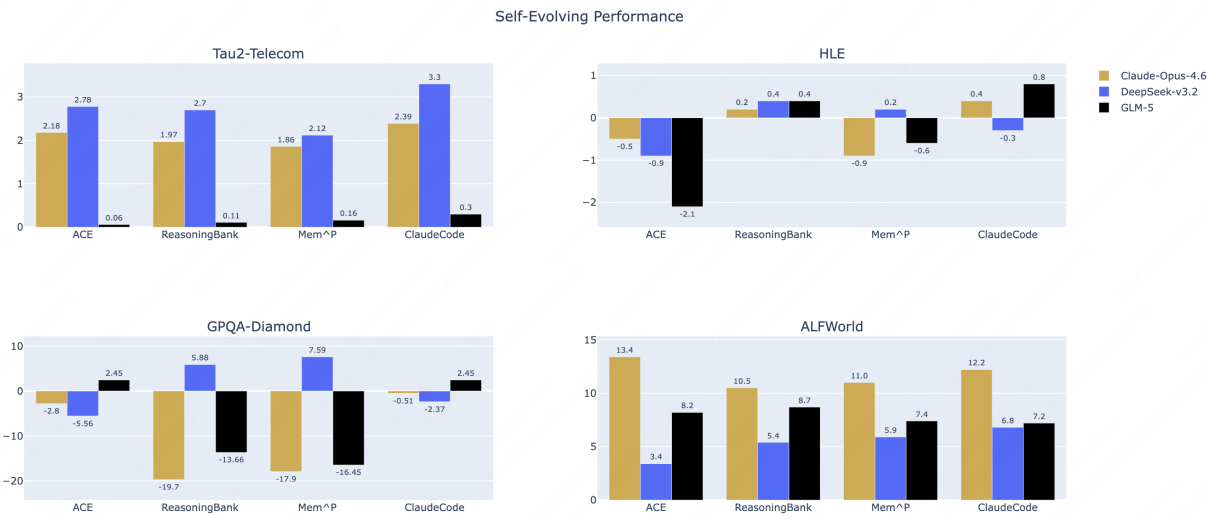

Figure 2: Performance comparison across all harness × model × benchmark combinations. Self-evolution reliably improves structured tasks (τ2-Telecom, ALFWorld) but shows inconsistent effects on knowledge-intensive reasoning (GPQA-Diamond) and no detectable effect on extreme-difficulty benchmarks (HLE).

Nonetheless, the directional consistency across both models (degradation without context management, stability with it) provides correlational evidence motivating further controlled investigation.

4.5 Batch Ordering Sensitivity Analysis

To address the critical limitation of fixed batch ordering, we conduct a dedicated sensitivity analysis on GPQA-Diamond, the benchmark with the most variable results. We re-run all harness × model combinations under two additional random batch orderings (random_shuffle_1, random_shuffle_2) while keeping all other experimental conditions identical to the original run.

Across all 12 harness-model cells, the maximum absolute deviation between any two orderings is 4.8 pp (Mem-p/GLM-5), with 9/12 cells showing variance $<$ 1 pp. No cell changes sign across orderings, confirming that the direction of evolution effects on GPQA-Diamond is robust to batch ordering. The most practically important results including ReasoningBank/Claude (−19.7 pp), Mem-p/Claude (−17.9 pp) and ReasoningBank/GLM-5 (−13.7 pp) show $<$ 0.5 pp variation, confirming these are genuine effects rather than ordering artifacts.

This analysis substantially mitigates the batch ordering concern: on GPQA-Diamond, the primary conclusions (retrieval-specific instability pat-

Evolution Scope	Model	τ2-Telecom	HLE	GPQA-Diamond	ALFWorld
Prompt + Tool	Claude Opus 4.6	—	✗	✗	✗
	DeepSeek-3.2	—	✗	✗	✗
	GLM-5	—	✗	0.645 (-0.215)	✗
Prompt + Scaffold	Claude Opus 4.6	—	✗	✗	✗
	DeepSeek-3.2	—	✗	✗	✗
	GLM-5	—	✗	✗	✗

Table 2: Tool and scaffold evolution results (Claude Code). ✗ = entered infinite debugging loop, failed to complete. — = not attempted due to τ2-Telecom's evaluation depends on specific tool schema. The single completed run (GLM-5/GPQA-Diamond/Tool) produced catastrophic degradation of -21.5 pp.

Coding Agent	Claude Opus 4.6	GLM-5
R2Egym (bash + file_edit, no context management)	$0.929 \rightarrow 0.886$ (-4.7%)	$0.860 \rightarrow 0.813$ (-5.5%)
CC (full context management)	$0.929 \rightarrow 0.924$ (-0.5%)	$0.860 \rightarrow 0.874$ ($+1.6\%$)

Table 3: Ablation: context management effect on GPQA-Diamond.

tern, baseline-strength correlation, ACE contrast) are fully preserved under alternative orderings.

5 Analysis and Discussion

Retrieval-augmented instability. On GPQA-Diamond, retrieval-augmented harnesses produce highly variable effects across models. From Table 1: ReasoningBank yields Claude -19.7 pp†, GLM-5 -13.7 pp, DeepSeek +5.9 pp; Mem-p yields Claude -17.9 pp, GLM-5 -16.5 pp, DeepSeek +7.6 pp. Note that ReasoningBank additionally alters the effective prompting format for Claude (effective baseline: 0.641 vs. standard 0.929; see Table 1 footnote). In contrast, ACE (no retrieval) shows smaller and directionally mixed effects: Claude -2.8 pp, GLM-5 +2.5 pp, DeepSeek -5.6 pp. The key empirical observation is the dissociation: retrieval-augmented harnesses produce large, model-dependent swings on GPQA while the non-retrieval harness does not. We note this pattern is observed across only 3 models, which is insufficient to establish a statistical relationship between baseline strength and effect direction.

Task-type dependence. Structured tasks with transferable patterns (ALFWorld, τ2-Telecom) show universally positive effects. Knowledge-intensive reasoning (GPQA) shows method-dependent effects driven by the retrieval mechanism. Extreme-difficulty tasks (HLE) show no detectable effect due to insufficient baseline success for experience extraction.

Harness-model interactions. Beyond the retrieval-specific pattern, gain magnitudes vary substantially across models (e.g., ACE/Claude +13.4 pp vs. ACE/DeepSeek +3.4 pp on ALFWorld). Optimal harness choice is model-dependent, requiring validation before deployment.

6 Design Hypotheses

Based on our empirical observations, we formulate four testable hypotheses:

H1: Pattern transferability gates self-evolution. Self-evolution helps when task instances share reusable procedural patterns (ALFWorld, τ2-Telecom), shows inconsistency when procedural overlap is minimal (GPQA), and fails when baseline success is too low for experience extraction (HLE).

H2: Context management is necessary for stable evolution. Without explicit context management, accumulated experience causes context pollution. Evidence: the ablation gap (Table 3) and ReasoningBank's severe degradation on Claude/GPQA.

H3: Evolution harness must match model capability. Scaffolding benefits weaker models but harms stronger ones via context competition. Evidence: ReasoningBank helps DeepSeek on GPQA (+5.9 pp) but harms Claude (-19.7 pp).

H4: Evolution reliability degrades from declarative to procedural targets. Prompt-only evolution succeeds reliably, but extending evolution scope to tool code or scaffold logic triggers self-amplifying failure loops (§4.3). Compound

Harness	Model	Shuffle 1	Shuffle 2
ACE	Claude Opus 4.6	−0.028	−0.026
ACE	DeepSeek-3.2	−0.056	−0.056
ACE	GLM-5	+0.025	+0.025
ReasoningBank	Claude Opus 4.6	−0.197	−0.197
ReasoningBank	DeepSeek-3.2	+0.059	+0.059
ReasoningBank	GLM-5	−0.137	−0.136
Mem-p	Claude Opus 4.6	−0.179	−0.181
Mem-p	DeepSeek-3.2	+0.076	+0.075
Mem-p	GLM-5	−0.165	−0.160
CC (prompt)	Claude Opus 4.6	−0.005	−0.007
CC (prompt)	DeepSeek-3.2	−0.024	−0.024
CC (prompt)	GLM-5	+0.025	+0.026

Table 4: Batch ordering sensitivity on GPQA-Diamond (Δ from baseline across 3 orderings). No cell changes sign; 9/12 cells have max range < 1 pp.

systems permitting code-level evolution require robust guardrails (syntax validation, rollback, sandboxed execution).

7 Limitations

Statistical: Each cell is a single run without confidence intervals or significance tests. Batch ordering sensitivity is confirmed only on GPQA-Diamond. However, our core conclusions rest on consistent directional patterns across multiple cells (e.g., 12/12 positive for ALFWorld, retrieval/non-retrieval dissociation on GPQA), which are unlikely to arise from noise alone.

Experimental design: The ablation (Table 3) cannot strictly isolate context management from other system differences. Claude Code as an industrial product creates asymmetric comparison with academic prototypes. ReasoningBank's retrieval changes the effective baseline. These confounds affect the strength of specific causal claims but do not undermine the primary empirical patterns (task-type dependence, the retrieval/non-retrieval dissociation, and the declarative-procedural failure boundary), which are observed independently of the ablation design.

8 Conclusion

We presented a systematic empirical evaluation of self-evolving agent harnesses across three models, four harnesses, and four benchmarks. Our primary empirical contributions are:

A controlled multi-factor comparison establishing that self-evolution reliably benefits structured tasks with transferable patterns (ALFWorld: universally positive; τ2-Telecom: majority meaningfully positive) but produces no detectable benefit on extreme-difficulty tasks (HLE) and shows inconsistent effects for retrieval-augmented harnesses on knowledge-intensive reasoning (GPQA).

Identification of a retrieval-specific instability pattern (§5): retrieval-augmented harnesses (ReasoningBank, Mem-p) produce large, model-dependent effects on GPQA (−19.7 to +7.6 pp), while the non-retrieval harness (ACE) produces only small, mixed effects (−5.6 to +2.5 pp). This dissociation identifies inference-time context injection as the distinguishing architectural factor.

Directional consistency (4/4 conditions) that context management is associated with evolution stability in the coding-agent paradigm: degradation without it, stability with it.

A sharp reliability boundary between declarative and procedural evolution: extending evolution scope to tool code or scaffold logic triggers self-amplifying failure loops (17/18 runs diverged), demonstrating that current models cannot reliably evolve their own procedural components without robust guardrails.

Future work. Several broad directions emerge from this study. First, establishing formal statistical confidence through repeated-trial evaluation would strengthen the empirical foundation. Second, extending evaluation to a wider range of benchmarks and task types would refine our understanding of when self-evolution helps versus harms. Third, investigating robustness mechanisms represents a crucial engineering challenge for real-world deployment. Finally, developing principled methods for matching evolution strategies to model capabilities and task characteristics would move the field from post-hoc analysis toward predictive deployment guidelines.

References

Marvin Eisenberger Emilien Dupont Po-Sen Huang Adam Zsolt Wagner Sergey Shirobokov Borislav Kozlovskii Francisco J. R. Ruiz Abbas Mehrabian M. Pawan Kumar Abigail See Swarat Chaudhuri George Holland Alex Davies Sebastian Nowozin Pushmeet Kohli Alexander Novikov, Ngân Vu and Matej Balog. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Yan Yang Yuxiang Wei-Lingming Zhang Chunqiu Steven Xia, Zhe Wang. 2025. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*.

Marc'Aurelio Ranzato David Lopez-Paz. 2017. Gradient episodic memory for continual learning. In *Proceedings of NeurIPS*.

Asa Cooper Stickland Jackson Petty Richard Yuanzhe Pang Julien Dirani Julian Michael Samuel R. Bowman David Rein, Betty Li Hou. 2023. GPQA: A graduate-level google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*.

Federico Siciliano Simone Filice Cesare Campagnano Yoelle Maarek Nicola Tonellotto Fabrizio Silvestri Florin Cuconasu, Giovanni Trappolini. 2024. The power of noise: Redefining retrieval for RAG systems. *arXiv preprint arXiv:2401.14887*.

Kanishka Misra Nathan Scales David Dohan Ed Chi Nathanael Schärli Denny Zhou Freda Shi, Xinyun Chen. 2023. Large language models can be easily distracted by irrelevant context. In *Proceedings of ICML*.

Andreas S. Tolias Gido M. van de Ven. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Jiangtao Feng Jiangjie Chen Weinan Dai Qiying Yu Ya-Qin Zhang Wei-Ying Ma Jingjing Liu Mingxuan Wang Hao Zhou Hongli Yu, Tinghong Chen. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Neil Rabinowitz Joel Veness Guillaume Desjardins Andrei A. Rusu-Kieran Milan John Quan Tiago Ramalho Agnieszka Grabska-Barwinska Demis Hassabis Claudia Clopath Dharshan Kumaran Raia Hadsell James Kirkpatrick, Razvan Pascanu. 2016. Overcoming catastrophic forgetting in neural networks. *arXiv preprint arXiv:1612.00796*.

Xuanming Zhang Yihong Dong Zhi Jin Jia Li, Ge Li. 2024. Evocodebench: An evolving code generation benchmark aligned with real-world code repositories. *arXiv preprint arXiv:2404.00599*.

Alexander Wettig Kilian Lieret Shunyu Yao Karthik Narasimhan-Ofir Press John Yang, Carlos E. Jimenez. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Xueyang Feng Zeyu Zhang Hao Yang Jingsen Zhang Zhiyuan Chen Jiakai Tang Xu Chen Yankai Lin Wayne Xin Zhao Zhewei Wei Ji-Rong Wen Lei Wang, Chen Ma. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Ziwen Han Nathaniel Li Josephina Hu Hugh Zhang Chen Bo Calvin Zhang Mohamed Shaaban John Ling Sean Shi Michael Choi Anish Agrawal-Arnav Chopra Adam Khoja Ryan Kim Richard Ren Jason Hausenloy Oliver Zhang Mantas Mazeika Dmitry Dodonov Tung Nguyen Jaeho Lee Daron Anderson Mikhail Doroshenko Alun Cennyth Stokes Mobeen Mahmood Oleksandr Pokutnyi Oleg Iskra Jessica P. Wang John-Clark Levin Mstyslav Kazakov Fiona Feng Steven Y. Feng Haoran Zhao Michael Yu Varun Gangal Chelsea Zou Zihan Wang Serguei Popov Robert Gerbicz Geoff Galgon Johannes Schmitt Will Yeadon Yongki Lee Scott Sauers Alvaro Sanchez Fabian Giska Marc Roth Søren Riis Saiteja Utpala Noah Burns Gashaw M. Goshu Mohinder Maheshbhai Naiya Chidozie Agu Zachary Giboney Antrell Cheatom Francesco Fournier-Facio Sarah-Jane Crowson Lennart Finke Zerui Cheng Jennifer Zampese Ryan G. Hoerr Mark Nandor Hyunwoo Park Tim Gehrunger Jiaqi Cai Ben McCarty Alexis C Garretson Edwin Taylor Damien Sileo Qiuyu Ren Usman Qazi Lianghui Li Jungbae Nam John B. Wydallis Pavel Arkhipov Jack Wei Lun Shi Aras Bacho Chris G. Willcocks Hangrui Cao Sumeet Motwani Emily de Oliveira Santos Johannes Veith Edward Vendrow Doru Cojoc Kengo Zenitani Joshua Robinson Longke Tang Yuqi Li Joshua Vendrow Natanael Wildner Fraga Vladyslav Kuchkin Andrey Pupasov Maksimov Pierre Marion Denis Efremov Jayson Lynch Kaiqu Liang Aleksandar Mikov Andrew Gritsevskiy Julien Guillod et al. (1020 additional authors not shown) Long Phan, Alice Gatti. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Victor Xing Gautier Viaud Marc-Antoine Allard, Arnaud Teinturier. 2026. Experiential reflective learning for self-improving llm agents. *arXiv preprint arXiv:2603.24639*.

Marc-Alexandre Côté Yonatan Bisk Adam Trischler Matthew Hausknecht Mohit Shridhar, Xingdi Yuan. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *Proceedings of ICLR*.

Manish Shetty Liang Zheng Koushik Sen Ion Stoica Naman Jain, Jaskirat Singh. 2025. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*.

Edward Berman Ashwin Gopinath Karthik Narasimhan Shunyu Yao Noah Shinn, Federico Cassano. 2023. Reflexion: Language agents

with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Hanyang Wang Jiaqi Liu Kaide Zeng Yu Wang Siwei Han Yiyang Zhou Xujiang Zhao Haifeng Chen Zeyu Zheng Cihang Xie Huaxiu Yao Peng Xia, Jianwen Chen. 2026. SKILLRL: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Shubhangi Upasani Boyuan Ma Fenglu Hong Vamsidhar Kamanuru Jay Rainton Chen Wu Mengmeng Ji Hanchen Li Urmish Thakker James Zou Kunle Olukotun Qizheng Zhang, Changran Hu. 2026. Agentic context engineering: Evolving contexts for self-improving language models. In *Proceedings of ICLR*.

Jianbiao Mei Pinlong Cai Daocheng Fu Cheng Yang Licheng Wen Xuemeng Yang Yufan Shen Yuxin Wang Botian Shi Rong Wu, Xiaoman Wang. 2026. Evolver: Self-evolving llm agents through an experience-driven lifecycle. In *Proceedings of ICML*.

Xiaobin Wang Jialong Wu Shuofei Qiao Pengjun Xie Fei Huang Huajun Chen Ningyu Zhang Runnan Fang, Yuan Liang. 2026. Memp: Exploring agent procedural memory. In *Proceedings of ACL Findings*.

Ari Holtzman Mikel Artetxe Mike Lewis Hannaneh Hajishirzi Luke Zettlemoyer Sewon Min, Xinxi Lyu. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of EMNLP*.

Dian Yu Nan Du Izhak Shafran Karthik Narasimhan Yuan Cao Shunyu Yao, Jeffrey Zhao. 2023. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of ICLR*.

I-Hung Hsu Yanfei Chen Ke Jiang Zifeng Wang Rujun Han Long T. Le Samira Daruki Xiangru Tang Vishy Tirumalashetty George Lee-Mahsan Rofouei Hangfei Lin Jiawei Han Chen-Yu Lee Tomas Pfister Siru Ouyang, Jun Yan. 2026. ReasoningBank: Structured reasoning memory for LLM-based agents. In *Proceedings of ICLR*.

Benjamin Coleman Zhankui He Yuanchen Bei Xuying Ning Mengting Ai Yunzhe Li Jingrui He Ed H. Chi Chi Wang Shuo Chen Fernando Pereira Wang-Cheng Kang Derek Zhiyuan Cheng Tianxin Wei, Noveen Sachdeva. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*.

Soham Ray Xujie Si Karthik Narasimhan Victor Barres, Honghua Dong. 2025. τ2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*.

Omar Khattab Yoonho Lee. 2026. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*.

Zhiqi Liang Yuzhen Mao Yuanzhe Hu Julian McAuley Xiaojian Wu Yu Wang, Ryuichi Takanobu. 2025. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*.

Qianjun Pan Bihao Zhan Yuxuan Cai Lin Du Jie Zhou Kai Chen Qin Chen Xin Li Bo Zhang Liang He Yutao Yang, Junsong Li. 2026. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*.

Yuxiang Ji Xucong Wang Yong Wang Yiming Hu Tongwen Huang Xiangxiang Chu Ziyu Ma, Shidong Yang. 2026. SkillClaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*.