SKILLFLOW: BENCHMARKING CONTINUAL LEARNING THROUGH SELF-EVOLVING AGENT SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

What distinguishes an agent that merely completes tasks from one that improves by completing them? Existing benchmarks primarily evaluate task success across isolated episodes or assess the utilization of pre-supplied skills, leaving unaddressed whether agents can convert their own experience into reusable procedural knowledge. We introduce SKILLFLOW, a benchmark designed for continual learning via self-evolving agent skills. Across 20 workflow families, its 166 tasks share execution structures while varying domains, artifacts, and operational conditions. Within each family, an agent begins with an empty skill library and can create, refine, and reuse skills derived from execution trajectories and verifier feedback. Using a unified Terminus-2 evaluation harness, we evaluate vanilla execution against an evolving-library setting across ten models. This comparison demonstrates a pronounced capability gap: models exhibiting comparable vanilla performance can differ substantially in their capacity to benefit from skill evolution, and frequent skill access does not necessarily yield effective adaptation. Trajectory analysis indicates that successful skill transfer relies on portable procedures and objective-aligned verification, whereas overly specific instructions and self-confirming checks are associated with negative transfer. SKILLFLOW renders these learning dynamics observable and provides a controlled testbed to investigate how and when agent skills foster continual adaptation.

1 INTRODUCTION

Frontier LLM systems increasingly operate as autonomous agents that solve realistic multi-step tasks in interactive environments. Yet these agents still behave as *session-local problem solvers*: they complete a task, discard most of the procedural lessons, and subsequently solve related future tasks again from scratch. This inefficiency highlights a deeper limitation. Existing benchmarks measure whether an agent can *use* a provided skill, but they offer little insight into whether an agent can *discover* a useful skill from task completion, *repair* that skill after failure, and *maintain* a coherent library over time.

Prior work addresses several facets of this problem. Reflexion and ExpeL retain feedback or trajectory-derived experience (Shinn et al., 2023; Zhao et al., 2024); Voyager develops an executable skill library (Wang et al., 2024a); and Agent Workflow Memory extracts reusable workflows (Wang et al., 2025). LifelongAgentBench evaluates broader lifelong behavior (Zheng et al., 2025b), whereas SkillsBench evaluates pre-supplied skills (Li et al., 2026), and frameworks such as SkillWeaver, SkillRL, and MemSkill derive or optimize procedural knowledge (Zheng et al., 2025a; Xia et al., 2026; Zhang et al., 2026). Variations in supervision, memory representation, and learning mechanisms nevertheless obscure whether agents can extract, revise, and benefit from procedures under a unified, controlled protocol.

Addressing this question requires an evaluation framework organized around the **skill lifecycle**. We introduce **SKILLFLOW**, a benchmark comprising 166 runnable tasks across 20 task families to evaluate continual learning through self-evolving agent skills. Figure 1 provides an architectural overview of this setting. Each family shares a common **Domain-Agnostic Execution Flow (DAEF)**, a workflow skeleton abstracting away domain-specific entities while preserving operational topology. DAEF serves as the structural foundation of the benchmark: it defines task families, enables

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.

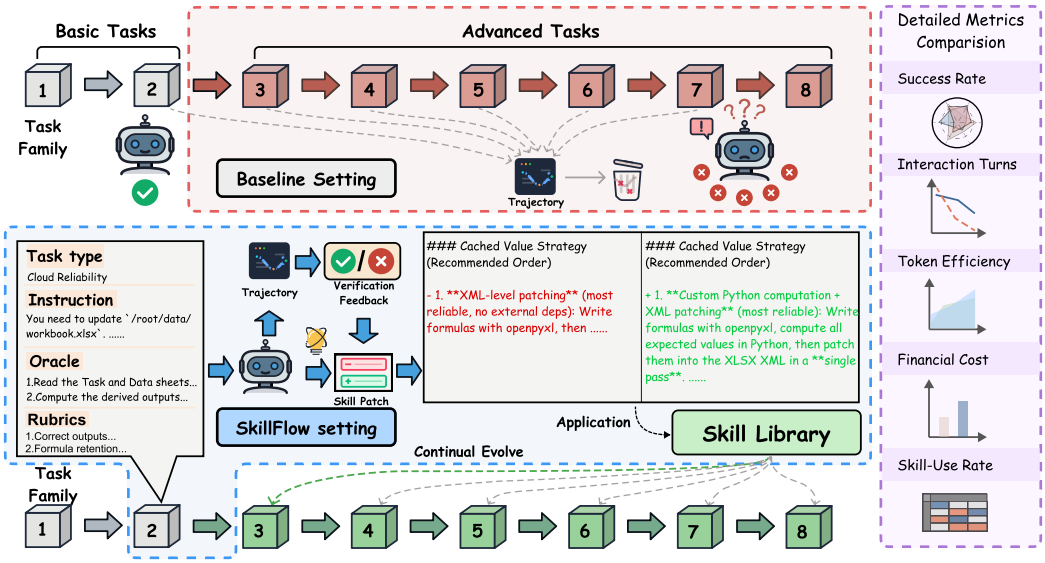


Figure 1: Overview of SKILLFLOW. For a sequence of tasks that share a DAEF, the baseline discards completed-task trajectories, whereas SKILLFLOW converts trajectories and verifier feedback into patches that revise a family-local skill library for subsequent tasks. The evaluation compares task success, interaction turns, token use, cost, and skill use.

controlled within-family progression, and provides the basis for evaluating whether an agent can transfer procedural knowledge across variations in domain surface features.

Task construction employs layered verification for workflow fidelity, environment validity, family consistency, and difficulty calibration, followed by expert human review. A dual-agent generation pipeline synthesizes and critiques Harbor (Harbor Framework Team, 2026) tasks. The evaluation protocol implements a **continual skill learning** framework: after each attempted task, including failed attempts, an agent can leverage its execution trajectory and verifier-derived feedback to update a family-local skill library for subsequent tasks.

Each evaluated model executes tasks via the Terminus-2 harness. We evaluate vanilla execution against a skill-evolution condition within each model, keeping evaluation tasks, presentation order, and verifier identical. Across the ten model configurations in the primary experiment, skill evolution improves the macro-average completion rate across models by 3.01 percentage points, although the gain varies across models and workflows. Models exhibiting comparable baseline performance derive disparate benefits; performance gains typically emerge after prior task experience becomes available; and frequent skill retrieval does not necessarily yield downstream improvement. These findings shift the central inquiry from whether a library is actively retrieved to whether an agent can translate raw experience into a procedure that successfully transfers.

Our primary contributions are summarized as follows:

- We introduce **SkillFlow**, a benchmark for evaluating skill evolution across 166 tasks organized into 20 workflow families. By structuring related tasks around shared domain-agnostic execution flows (DAEFs) with escalating difficulty, SkillFlow enables controlled assessment of whether agents can discover, refine, and reuse procedural knowledge.
- We establish a **workflow-scoped continual skill evolution protocol**. Agents begin each family without prior skills and iteratively update a persistent library using execution traces and verifier feedback. Paired comparisons against vanilla execution isolate skill-evolution gains while holding the underlying model, task sequence, verifier, and environment invariant.
- We present a **systematic empirical study of skill evolution dynamics**. Evaluating multiple models under a unified harness reveals pronounced heterogeneity in adaptation gains, identifying key mechanisms of successful procedural transfer as well as failure modes that trigger negative transfer.

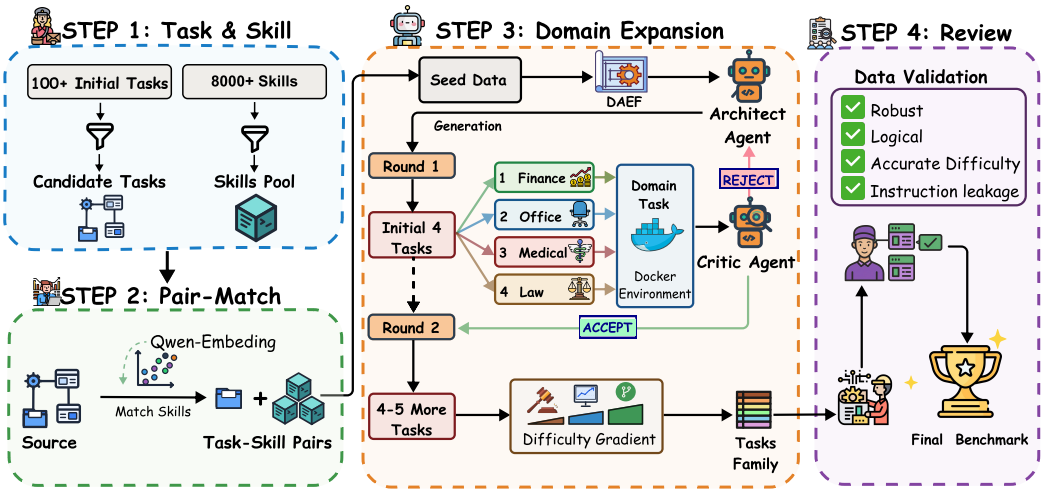


Figure 2: Task-construction pipeline of SKILLFLOW. Candidate tasks and skills are matched into reference packages, expanded under a fixed DAEF through an architect–critic loop, and screened by human reviewers.

2 SKILLFLOW

2.1 BENCHMARK OVERVIEW

SKILLFLOW comprises 20 workflow families and 166 tasks spanning five broad domains: *Finance & Economics*, *Operations & Supply Chain*, *Healthcare & Life Sciences*, *Governance & Strategy*, and *Data & Document Intelligence*. Each benchmark instance is defined by a family-local task sequence, a fixed within-family difficulty progression, and an associated verifier specification. Family sizes range from 8 to 9 tasks. Verifier implementations encompass exact programmatic checkers for structured outputs as well as format-sensitive verification for document-editing tasks. At a high level, the benchmark encompasses spreadsheet-centric planning, OCR and PDF extraction, office-document editing, compliance analysis, and quantitative reasoning workflows.

2.2 DOMAIN-AGNOSTIC EXECUTION FLOW

The core design principle of SKILLFLOW is the Domain-Agnostic Execution Flow (DAEF), a workflow abstraction that groups tasks according to shared procedural structure rather than domain-specific content. Formally, we represent a task as a domain-grounded workflow graph:

$$\mathcal{T} = (V, E, \lambda, \gamma), \tag{1}$$

where V denotes executable operations, $E \subseteq V \times V$ represents their dependency relations, $\lambda(v)$ assigns each node a domain-agnostic operation type, and $\gamma(v)$ specifies its task-specific grounding, such as files, entities, fields, or business objects. Projecting out this grounding yields the corresponding DAEF:

$$\mathcal{F} = \phi(\mathcal{T}) = (V_F, E_F, \lambda_F), \tag{2}$$

where the mapping $\phi(\cdot)$ preserves operation types and dependency topology while abstracting away task-specific grounding. Tasks belong to the same family when they share this abstract workflow structure, even if they diverge in domain context, input artifacts, instruction phrasing, or difficulty level. DAEF therefore defines the fundamental unit over which SKILLFLOW evaluates procedural transfer.

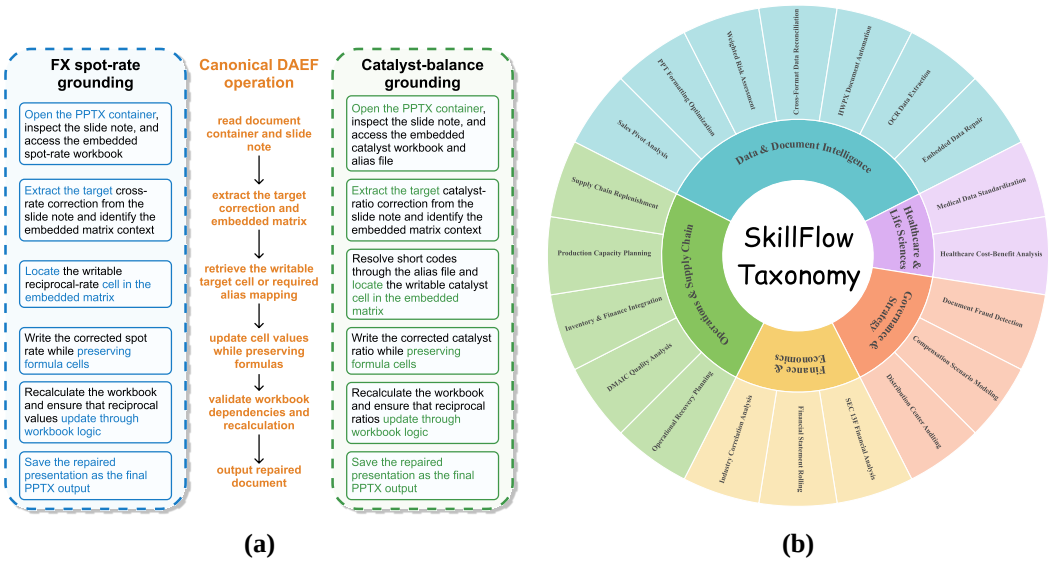


Figure 3: Task example and benchmark taxonomy. (a) Two tasks with different domain grounding share a canonical DAEF operation sequence. (b) The five domains contain 20 DAEF-defined workflow families.

Figure 3(a) illustrates this correspondence. For instance, an FX spot-rate repair task and a catalyst-balance repair task involve distinct entities and supporting artifacts, yet both follow an identical sequence of operations: inspect the document and embedded workbook, identify the required correction, locate the writable target, update the embedded data while preserving formulas, validate the resulting workbook, and save the repaired presentation. Figure 3(b) delineates the 20 DAEF-defined families across the five broad domains of the benchmark.

We standardize DAEFs prior to task-family expansion. Annotators first decompose each seed task into 5–8 executable meta-steps alongside their dependencies, utilizing a controlled vocabulary of operation types. Task-specific grounding is subsequently removed to derive a canonical workflow that retains only invariant operations and dependency structures. A family is retained only if this structure remains coherent across task variants while permitting sufficient variation in grounding. Two domain experts independently annotate candidate workflows in the final screening pool, achieving 85% inter-annotator agreement on decomposition, node labels, and dependency edges. Appendix A.4 describes the annotation vocabulary and screening criteria, and Appendix A.5 outlines the permitted axes of variation.

2.3 Task Construction Protocol

Figure 2 illustrates the benchmark construction pipeline. The process initiates with 64 seed tasks selected from SkillsBench (Li et al., 2026) and GDPval (Patwardhan et al., 2025), paired with a curated repository of 2,318 skills sampled from over 8,000 open-source candidates collected across public repositories and platforms (ComposioHQ, n.d.; VoltAgent, n.d.; Anthropic, n.d.; SkillsMP, n.d.). Seed tasks are selected based on realistic execution workflows and reliable programmatic verification. Candidate skills are matched to seeds via semantic similarity using Qwen3 Embedding (Zhang et al., 2025) and serve, alongside the original task assets and annotated DAEFs, as references for family generation.

Each reference package undergoes an architect–critic synthesis loop. The architect synthesizes task variants, execution environments, solver assets, and verifiers while preserving the target DAEF. The critic verifies execution reliability, workflow consistency, oracle availability, and difficulty progression; failing candidates are refined or discarded. Families passing this phase are expanded under the identical DAEF, typically yielding 8–9 related tasks with escalating procedural complexity.

Human reviewers subsequently evaluate each family for instruction leakage, logical consistency, environment validity, and intended difficulty progression. The resulting benchmark encompasses 20 workflow families and 166 tasks across five domains. Appendix Table 3 provides the corresponding heterogeneity audit, detailing an average of 7.6 domain groundings and 3.5 asset configurations per family. Verification relies primarily on deterministic programmatic checkers for structured outputs such as spreadsheets and JSON, complemented by format-sensitive validation for document-editing tasks.

2.4 Continual Learning through Self-Evolving Skills

Sequential protocol and feedback. Evaluation proceeds within each workflow family according to a fixed difficulty progression. The agent begins the first task with an empty skill library, retains the evolving library across subsequent tasks in that family, and resets it before entering the next family. After every attempted task, regardless of outcome, the agent receives its execution trajectory and verifier feedback and may generate a post-task patch. Each patch becomes available only to later tasks and has no retrospective effect on the task that produced it.

Skill updates and observed use. A patch can create, revise, or remove library artifacts through three auditable fields: `summary`, `upsert_files`, and `delete_paths`. These operations record the abstracted takeaway, maintain `SKILL.md` files and auxiliary scripts, preserve revision lineage, and prune obsolete procedures. During subsequent execution, skill use is registered when a trace command reads a file in the family-local skill directory; prompt-based directory listings and bootstrap placeholders are excluded. This file-level protocol jointly exposes how agents construct, maintain, retrieve, and apply procedural knowledge over a task sequence.

2.5 Metrics

We assess agent performance across three metric categories:

- **Task success rate**: whether the recorded verifier reward is at least one.
- **Efficiency**: interaction turns, task-agent output tokens, and reported monetary cost per task. Trajectory-derived token counts exclude the post-task patch-generation call.
- **Skill generation and reuse**: statistics capturing both the frequency of generated skills during sequential execution and the rate of subsequent skill retrieval across later tasks.

We evaluate task completion across the identical 166 tasks per model in a fixed within-family sequence. For the macro-average performance gain derived from skill evolution, we report a 95% nonparametric bootstrap confidence interval computed by resampling the ten model-level gains, summarizing cross-model variation in skill-evolution benefits. Family-level outcomes and paired model–workflow analyses characterize workflow heterogeneity.

3 Experiments

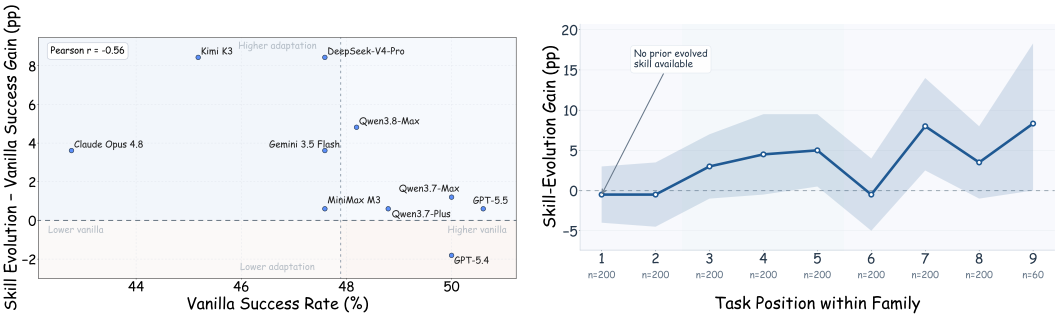
3.1 Experimental Setup and Analysis Protocol

We evaluate SkillFlow under the **continual skill learning** setting. Every model executes tasks through the unified Terminus-2 execution harness; Appendix Table 7 records the runtime configuration. Within each task family, an agent begins with an empty skill library and follows the fixed task sequence. We compare *vanilla* execution without an evolving library against *skill evolution*, in which the agent receives its execution trajectory and verifier-derived feedback after each task and updates a family-local library via file-level patches. The skill library is reset between families. Each within-model comparison holds task assets, the verifier, task order, and the execution harness invariant.

To evaluate verifier validity, we manually audit a domain-stratified, balanced random sample of 56 verifier-accepted and verifier-rejected trajectory outputs. Human evaluations align with the verifier on 91.07% of outputs; all five discrepancies constitute false negatives, and deliberately corrupted versions of correct outputs are detected in 100% of cases (Appendix Table 8).

Table 1: Terminus-2 run summaries under vanilla execution and skill evolution. Completion and skill-use rates use the same 166 tasks per model; turns, cost (USD), and output tokens (thousands) are reported per-task averages. In the Δ block, completion is a percentage-point difference calculated from task counts before rounding, turns is an absolute difference, and cost and output tokens are relative percentage changes computed from the displayed values. Green highlights higher-is-better completion change; blue highlights lower-is-better resource changes. **Bold** marks the best value in each performance column.

	Vanilla				Skills Evolve						Δ			
Model	Comp.	Turns	Cost	Out Tok.	Comp.	Turns	Cost	Out Tok.	#Skills	Skill Use	Comp.	Turns	Cost (%)	Out Tok. (%)
GPT 5.4	50.00	6.42	0.18	5.83	48.19	6.74	0.22	6.52	1.25	52.41	-1.81	+0.32	+22.22	+11.84
GPT 5.5	**50.60**	6.93	0.43	6.73	51.20	**6.34**	0.38	**5.98**	1.05	53.61	+0.60	**-0.59**	**-11.63**	**-11.14**
Claude Opus 4.8	42.77	10.65	0.40	6.14	46.39	11.86	0.44	6.51	1.00	79.52	+3.61	+1.21	+10.00	+6.03
Gemini 3.5 flash	47.59	**6.17**	0.09	**5.53**	51.20	6.54	0.12	6.54	1.15	53.01	+3.61	+0.37	+33.33	+18.26
Qwen 3.8Max	48.19	10.90	0.15	17.34	53.01	12.14	0.18	18.09	1.05	87.95	+4.82	+1.24	+20.00	+4.33
Qwen 3.7Max	50.00	12.53	0.19	9.01	51.20	12.42	0.19	9.02	1.15	50.60	+1.20	-0.11	0.00	+0.11
qwen3dot7-plus	48.80	8.04	**0.03**	13.43	49.40	8.66	**0.04**	15.91	1.70	66.27	+0.60	+0.62	+33.33	+18.47
Minimax-M3	47.59	16.41	**0.03**	11.17	48.19	17.58	**0.04**	11.92	1.55	66.27	+0.60	+1.17	+33.33	+6.71
deepseek4-pro	47.59	8.49	0.10	21.40	**56.02**	10.48	0.12	24.11	1.35	59.64	**+8.43**	+1.99	+20.00	+12.66
kimi K3	45.18	9.81	0.14	6.73	53.61	9.58	0.19	8.37	1.25	79.52	**+8.43**	-0.23	+35.71	+24.37



(a) **(b)**

Figure 4: Skill-evolution gains. (a) Vanilla completion versus gain across ten models. (b) Gain by position in the fixed workflow-family order on the common 166-task grid.

The evaluation encompasses the identical 166 tasks for each of the ten models. We define **skill-evolution gain** as the difference in task completion rates between the two conditions:

$$\Delta_{\text{skill}} = 100\,(S_{\text{evolve}} - S_{\text{vanilla}})\,, \tag{3}$$

where each S denotes the number of successful tasks divided by 166. This denominator is applied across all model-, workflow-, position-, and skill-access analyses. Appendix Table 9 reports the model-level success counts.

Table 1 reports the model-level completion and resource metrics. The subsequent analyses utilize the reported model values for cross-model comparisons and the underlying task records for intra-family diagnostics.

3.2 Does Skill Evolution Improve Task Completion?

Across the ten model evaluations reported in Table 1, mean vanilla completion is 47.83% and mean skill-evolution completion is 50.84%, yielding a macro-average gain across models of 3.01 percentage points. Nine models exhibit performance gains, whereas GPT-5.4 regresses by 1.81 points. A nonparametric bootstrap over the ten model gains produces a 95% confidence interval of [1.08, 5.12] points, characterizing cross-model variation in the utility of skill evolution.

Workflow-level results localize where adaptation is most pronounced. Weighted-Risk-Assessment achieves a gain of 28.75 points and Embedded-Data-Repair gains 21.25 points, whereas Industry-Correlation-Analysis and OCR-Data-Extraction each decline by 5.00 points (-5.00). Appendix C.2 reports all family-level results, and corresponding domain and family heatmaps (Figures 6 and 7)

illustrate model-specific specialization across broad domains. This structured heterogeneity makes skill evolution informative at both the model and workflow levels.

3.3 Is Skill-Evolution Ability Captured by Vanilla Performance?

If certain agents benefit more from skill evolution, does this gain merely stem from their capacity to solve more tasks without it? Figure 4(a) provides empirical evidence against this explanation. GPT-5.4 and GPT-5.5 each solve approximately 50% of vanilla tasks, yet their gains diverge markedly (-1.81 versus $+0.60$ points). DeepSeek-V4-Pro and Kimi K3 exhibit lower baseline completion rates (47.59% and 45.18%) but achieve the largest performance gains ($+8.43$ points). Skill evolution therefore alters the relative capability rankings established by vanilla performance.

The model-level correlation between baseline success rates and subsequent gains is $r = -0.56$. Together with the observed variance among models exhibiting comparable vanilla completion, this result demonstrates that skill-evolution efficacy is not tightly coupled to baseline task-solving capability. The capacity to translate prior execution trajectories into reusable procedural knowledge therefore represents a distinct dimension of agent proficiency.

3.4 Do Benefits Emerge as Experience Accumulates?

The initial task within each family begins without a previously evolved skill: any generated patch can benefit only subsequent tasks. Figure 4(b) illustrates average gains of approximately -0.5 points at both positions 1 and 2, followed by consistently higher gains once prior tasks provide execution experience. Appendix Figure 8 decomposes this aggregate trend into model-specific trajectories. Across the common 166-task benchmark grid, gains average -0.5 points in early stages (positions 1–2), $+4.17$ in middle stages (positions 3–5), and $+4.09$ in late stages (position 6 onward). Appendix Figure 9 groups these outcomes into early, middle, and late evaluation stages.

The transition from negligible early gains to substantial middle- and late-stage gains exhibits an emergence-and-plateau pattern, accompanied by localized fluctuations such as the dip at position 6, rather than strictly monotonic progress. A modest number of prior interactions appears sufficient for effective operational procedures to materialize, after which additional experience sustains their average benefit. The model-specific breakdown in Appendix Figure 8 shows enhanced post-early gains across eight of ten models, demonstrating that this accumulation dynamic is broadly shared across models rather than driven by an isolated subset of high-gain systems.

3.5 Does Skill Access Imply Useful Learning?

Subsequent tasks can query the evolving skill library, establishing skill retrieval as a directly observable behavioral marker of adaptation. On the common 166-task evaluation grid, the 1,070 tasks with an observed read of an evolved skill demonstrate a $+5.33$-point completion increase, compared with a -1.19-point decrease for 590 tasks executed without skill access. This 6.52-point divergence confirms skill retrieval as a strong behavioral indicator of successful adaptation, reflecting the combined capacity of an agent to select and apply retained procedures.

Sequence-level results clarify the conditions under which skill access translates into downstream improvement. In 89 of 200 model–family sequences, evolved skills are retrieved on at least 75% of tasks while the completion gain remains non-positive, and the final skill count exhibits virtually no linear correlation with all-task gains ($r \approx 0.01$). Appendix Figure 13 further illustrates substantial variation in library size and file distribution. Together, these findings indicate that effective skill evolution depends on three interconnected operations: formulating a portable procedure, selecting that procedure for an aligned task, and validating execution against the target objective. Retrieval frequency and library volume measure agent activity, whereas these underlying procedural characteristics dictate whether that activity translates into performance gains.

3.6 What Makes Skill Evolution Succeed or Fail?

While aggregate comparisons identify when performance gains emerge, recorded trajectories reveal the specific skill content and verification patterns converting retrieval into empirical gains (Appendix C.4).

Portable procedure and independent checks. DeepSeek-V4-Pro fails the initial Weighted-Risk-Assessment task under both conditions. Its post-task patch generates an Excel skill introducing formula recalculation, cached-value inspection, and an independent expected-value check. All seven subsequent tasks retrieve this skill and convert vanilla failures into skill-evolution successes, shifting family completion from 0/8 to 7/8. In a second instance, Qwen3.7-Max accesses an OCR procedure on a subsequent task, succeeding where vanilla execution fails while reducing execution length from 28 to 11 turns. These cases illustrate how a reusable procedure facilitates execution through targeted retrieval on compatible tasks.

Over-specific instructions and misplaced validation. The inverse pattern arises when retained instructions encode local operational assumptions or verify inappropriate targets. DeepSeek-V4-Pro accesses and refines a PowerPoint-editing skill across all seven subsequent tasks, yet family-level performance drops from 3/8 vanilla successes to zero: its validator verifies package integrity without confirming specified slide formatting. In the Operational-Recovery-Planning sequence of Qwen3.7-Max, an overly narrow skill constructed after a successful run hard-codes purchase-order dates and specific quantities; a subsequent task retrieves this skill, regressing from vanilla success to failure. Paired task outcomes and revision histories isolate two governing determinants of transfer efficacy: the operational generality of the stored procedure and the independence of its validation mechanism. Together with successful trajectories, these findings establish portable abstraction, context-appropriate selection, and objective-aligned validation as foundational pillars of effective skill evolution.

Efficiency trade-offs. Skill evolution also alters resource utilization during task execution; Appendix C.3 reports corresponding frontiers across completion–cost, completion–turns, and completion–output-tokens. The core investigation above focuses specifically on adaptation dynamics and functional correctness.

4 Discussion

Skill evolution as a distinct capability. Our results establish skill evolution as an orthogonal capability to episodic task completion. Models with comparable vanilla success diverge substantially in their gains under an identical trajectory-feedback interface, while certain weaker baseline agents improve markedly once experience persists. This distinction demonstrates that agent competence comprises two distinct dimensions: executing the immediate task and translating execution traces into transferable procedural knowledge. Existing benchmarks prioritize immediate execution while overlooking longitudinal procedural learning; SkillFlow bridges this gap through paired, controlled evaluations under a unified execution harness.

Emergence and consolidation. Position-level dynamics corroborate a threshold account of procedural acquisition. Early tasks supply the supervisory signals necessary to crystallize an initial routine, after which performance relies primarily on selective retrieval, adaptation, and library maintenance rather than raw experiential accumulation. This mechanism accounts for the observed emergence-and-plateau trajectory, where eight of ten models exhibit marked gains after early stages. While the capacity for experiential accumulation is broadly shared, differences in procedural abstraction and library maintenance govern the net performance improvement realized by each model.

Productive skill retrieval. Although skill retrieval strongly correlates with task success, frequent access alone is insufficient: high-retrieval sequences often yield negligible gains, and library volume exhibits near-zero linear correlation with performance. Trajectory analysis reveals that effective transfer requires procedures invariant to domain grounding, contextual retrieval matched to task specifications, and objective-aligned validation. Conversely, negative transfer emerges when skills overfit local constants or validate surrogate targets. Skill evolution therefore represents a composite capability of abstraction, selective retrieval, iterative revision, and external verification.

Implications for benchmark and agent design. Continual-agent benchmarks must evaluate the complete procedural lifecycle: synthesis, revision, retrieval, and downstream impact. Paired vanilla-versus-evolving protocols cleanly isolate learning gains from baseline proficiency under invariant execution stacks. For agent architectures, these findings advocate for skill managers that preserve update provenance, enforce selective retrieval, validate outputs against ultimate task objectives, and

prune over-specialized routines. Auditable file-level patches provide direct supervision signals for downstream research on automated skill refinement.

Future directions and limitations. Several frontiers warrant further investigation. Evaluating over extended horizons and utilizing counterbalanced task sequences would probe long-term procedural stability and decouple cumulative experience from difficulty progression. In addition, assessing cross-family transfer across related DAEFs and comparing explicit file-based skills against parameter-based updates or dense trajectory retrieval remain promising directions. Finally, while intra-family evaluations and single-pass runs isolate workflow-scoped learning within available computational resources, conducting multi-trial evaluations will offer deeper insights into stochastic stability and negative-transfer reproducibility.

5 Related Work

Agent evaluation and continual learning. Agent benchmarks measure tool use, multi-step reasoning, efficiency, and reliability (Patil et al., 2025; Liu et al., 2023; Trivedi et al., 2024; Wang et al., 2024b; Huang et al., 2025; Shi et al., n.d.; Su et al., 2026), including reproducible long-horizon tasks (Merrill et al., 2026; Xia et al., 2024; Deng et al., 2025; Chou et al., 2025). LifelongAgentBench evaluates agents across evolving task streams (Zheng et al., 2025b), Evo-Memory studies test-time learning with self-evolving memory (Wei et al., 2025), and StuLife studies experience-driven self-evolution (Cai et al., 2025). SkillFlow evaluates a more specific capability through paired vanilla and skill-evolution runs holding the model, harness, tasks, order, and verifier fixed.

Experience-based learning. Reflexion stores feedback as verbal reflections (Shinn et al., 2023); ExpeL distills reusable lessons from task histories (Zhao et al., 2024); and Agent-Pro improves policies through reflection and optimization (Zhang et al., 2024). Meta-experience can also be internalized as memory for guided learning (Huang et al., 2026). Rather than prescribing one learning method, SkillFlow evaluates the resulting capability: how well different models convert trajectories and verifier feedback into persistent procedural knowledge.

Workflow and procedural memory. Voyager maintains a growing library of executable skills (Wang et al., 2024a). Agent Workflow Memory extracts reusable workflows from trajectories (Wang et al., 2025), and Hierarchical MetaFlows abstracts experience across workflow levels (Fan et al., 2025). ProcMEM learns reusable procedures (Mi et al., 2026), while Managing Procedural Memory studies their control, adaptation, and evaluation (Belikova et al., 2026). These works motivate the workflow abstraction in SkillFlow alongside its analysis of selection, revision, validation, and negative transfer. SkillFlow exposes file-level patches and observed reads to audit these processes along controlled task sequences.

Skill discovery and evolution. Skill benchmarks test supplied skills or skill use (Li et al., 2026; Han et al., 2026; PinchBench Team, 2026); skill-management and trajectory-distillation methods represent experience as reusable procedures (Liang et al., 2026; Ni et al., 2026). Other systems derive or optimize skills through interaction and feedback (Zheng et al., 2025a; Xia et al., 2026; Zhang et al., 2026; Chen et al., 2026; Yang et al., 2026; Alzubi et al., 2026; Xu et al., 2023; Zhou et al., 2026; Cao et al., 2026). SkillFlow complements them by recording outcomes, skill reads, and file revisions under fixed within-family sequences, revealing when evolving skills transfer, fail, or interfere with later tasks.

6 Conclusion

SkillFlow turns skill evolution into a measurable question about continual agent adaptation. Across ten models under one harness, evolving skills improves model-macro completion by 3.01 percentage points on the common 166-task grid, but gains differ by model and workflow and are not predicted by vanilla performance alone. Benefits tend to appear after prior experience becomes available, while frequent skill access or library growth does not ensure improvement. The contrasting trajectories identify portable procedures and checks against the task objective as the practical distinction between useful reuse and negative transfer.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the SKILLFLOW benchmark construction procedure, DAEF annotation protocol, and task-family definitions in Section 2 and Appendix A. The continual skill-evolution protocol, skill-patch format, execution environment, runtime limits, decoding settings, and verifier audit are documented in Appendix B. Appendix C reports the statistical procedures, task-level analyses, and trajectory cases underlying the main results. We will release the benchmark tasks, execution environments, verifiers, task-level evaluation records, skill-evolution traces, and analysis scripts to support reproduction of the reported experiments.

AI USE STATEMENT

Generative AI models are an explicit component of the benchmark construction and evaluation pipeline described in this paper. In particular, model-based architect and critic agents are used during task-family construction as described in Section 2.3, and the evaluated language models constitute the agents studied in our experiments. For manuscript preparation, generative AI tools were used only for language polishing and copyediting. The authors reviewed the resulting text and take full responsibility for the scientific claims, experimental results, analysis, and conclusions presented in the paper.

REFERENCES

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

Anthropic. Anthropic skills repository. `https://github.com/anthropics/skills`, n.d.

Julia Belikova, Rauf Parchiev, Evgeny Egorov, Grigorii Davydenko, Gleb Gusev, Andrey Savchenko, and Maksim Makarenko. Managing procedural memory in llm agents: Control, adaptation, and evaluation. *arXiv preprint arXiv:2606.23127*, 2026.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zhen, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, et al. Building self-evolving agents via experience-driven lifelong learning: A framework and benchmark. *arXiv preprint arXiv:2508.19005*, 2025.

Weili Cao, Xunjian Yin, Bhuwan Dhingra, and Shuyan Zhou. Coding agents are effective long-context processors. *arXiv preprint arXiv:2603.20432*, 2026.

Shiqi Chen, Jingze Gai, Ruochen Zhou, Jinghan Zhang, Tongyao Zhu, Junlong Li, Kangrui Wang, Zihan Wang, Zhengyu Chen, Klara Kaleb, et al. Skillcraft: Can llm agents learn to use tools skillfully? *arXiv preprint arXiv:2603.00718*, 2026.

Jason Chou, Ao Liu, Yuchi Deng, Zhiying Zeng, Tao Zhang, Haotian Zhu, Jianwei Cai, Yue Mao, Chenchen Zhang, Lingyun Tan, et al. Autocodebench: Large language models are automatic code benchmark generators. *arXiv preprint arXiv:2508.09101*, 2025.

ComposioHQ. Awesome claude skills. `https://github.com/ComposioHQ/awesome-claude-skills`, n.d.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Shengda Fan, Xin Cong, Zhong Zhang, Yuepeng Fu, Yesai Wu, Hao Wang, Xinyu Zhang, Enrui Hu, and Yankai Lin. Generalizing experience for language agents with hierarchical metaflows. In *Advances in Neural Information Processing Systems*, 2025.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering? *arXiv preprint arXiv:2603.15401*, 2026.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments, January 2026. URL https://github.com/harbor-framework/harbor.

Shiting Huang, Zhen Fang, Zehui Chen, Siyu Yuan, Junjie Ye, Yu Zeng, Lin Chen, Qi Mao, and Feng Zhao. Critictool: Evaluating self-critique capabilities of large language models in tool-calling error scenarios. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26683–26692, 2025.

Shiting Huang, Zecheng Li, Yu Zeng, Qingnan Ren, Zhen Fang, Qisheng Su, Kou Shi, Lin Chen, Zehui Chen, and Feng Zhao. Internalizing meta-experience into memory for guided reinforcement learning in large language models. *arXiv preprint arXiv:2602.10224*, 2026.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect ai skills. *arXiv preprint arXiv:2603.04448*, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.

Mike A Merrill, Alexander G Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Procmem: Learning reusable procedural memory from experience via non-parametric ppo for llm agents. *arXiv preprint arXiv:2602.01869*, 2026.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simón Posada Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. *arXiv preprint arXiv:2510.04374*, 2025.

PinchBench Team. Pinchbench: Skill-based benchmark for llm agents. https://github.com/pinchbench/skill, 2026.

Kou Shi, Ziao Zhang, Shiting Huang, Avery Nie, Zhen Fang, Qiuchen Wang, Lin Chen, Huaian Chen, Zehui Chen, and Feng Zhao. Asynctool: Evaluating the asynchronous function calling capability under multi-task scenarios, n.d.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

SkillsMP. Skillsmp platform. https://skillsmp.com/, n.d.

Qisheng Su, Shiting Huang, Zhen Fang, Ziyan Chen, Zehui Chen, and Feng Zhao. Beyond accuracy: Unveiling inefficiency patterns in tool-integrated reasoning, 2026. URL https://arxiv.org/abs/2604.05404.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

VoltAgent. Awesome openclaw skills. https://github.com/VoltAgent/awesome-openclaw-skills, n.d.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. Officebench: Benchmarking language agents across multiple applications for office automation. *arXiv preprint arXiv:2407.19056*, 2024b.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 63897–63911, 2025.

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.

Chunqiu Steven Xia, Yinlin Deng, and Lingming Zhang. Top leaderboard ranking= top coding proficiency, always? evoeval: Evolving coding benchmarks via llm. *arXiv preprint arXiv:2403.19114*, 2024.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Mengda Xu, Zhenjia Xu, Cheng Chi, Manuela Veloso, and Shuran Song. Xskill: Cross embodiment skill discovery. In *Conference on robot learning*, pp. 3536–3555. PMLR, 2023.

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

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. Agent-pro: Learning to evolve via policy-level reflection and optimization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5348–5375, 2024.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025a.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners. *arXiv preprint arXiv:2505.11942*, 2025b.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

A Task Details

A.1 Comparison with Existing Benchmarks

Table 2 compares benchmark design along dimensions of the skill lifecycle. The comparison concerns what each benchmark explicitly evaluates, rather than its overall breadth or difficulty.

Table 2: Comparison with representative benchmarks from the perspective of skill-lifecycle evaluation. **Skill Eval** denotes explicit evaluation of skill impact; **Self-Gen** self-generated skills; **Revision** iterative skill updates; **Continual** sequential accumulation and reuse; **Workflow Transfer** transfer across related task instances that share a workflow; **Traj.-Grounded** extraction from interaction trajectories; and **Usage Eval** evaluation of skill utility together with observed use.

Benchmark	Skill Eval	Self-Gen	Revision	Continual	Workflow Transfer	Traj.-Grounded	Usage Eval
Terminal-Bench (Merrill et al., 2026)	×	×	×	×	×	×	×
SWE-bench (Jimenez et al., 2023)	×	×	×	×	×	×	×
SkillsBench (Li et al., 2026)	✓	✓	×	×	×	×	×
PinchBench (PinchBench Team, 2026)	×	×	×	×	×	×	×
SWE-Skills-Bench (Han et al., 2026)	✓	×	×	×	×	×	×
SkillFlow (Ours)	✓	✓	✓	✓	✓	✓	✓

A.2 Benchmark Statistics

The final benchmark contains 20 task families and 166 tasks organized under five broad domains: *Finance & Economics*, *Operations & Supply Chain*, *Healthcare & Life Sciences*, *Governance & Strategy*, and *Data & Document Intelligence*. These five domains cover 20 workflow families in total, with family sizes ranging from 8 to 9 tasks.

Table 3 reports the benchmark-wide task-heterogeneity audit across all 166 tasks. Families exercise 7.6 domain groundings, 3.5 asset configurations, 2.5 task-relevant file types, and 6.7 of eight prespecified variation dimensions on average. Mean token-Jaccard overlap between distinct families is 0.135 across 190 family pairs; the table separately reports within-family lexical overlap. These statistics characterize variation within a shared workflow and separation between families.

Table 3: Task heterogeneity across all 20 workflow families (166 tasks). Counts record distinct domain groundings, input-asset configurations, task-relevant file types, and the number of eight prespecified variation dimensions exercised within each family. Lexical overlap is the mean pairwise token-Jaccard overlap of task instructions within the family. The macro row averages family-level values; the mean overlap across the 190 distinct pairs of families is 0.135 (range 0.061–0.463).

Workflow family	Tasks	Groundings	Asset configs.	File types	Variation dims.	Lexical overlap
Compensation Scenario Modeling	8	5	3	1	6	0.32
Cross-Format Data Reconciliation	8	7	1	2	5	0.74
DMAIC Quality Analysis	9	7	2	2	7	0.44
Distribution Center Auditing	8	8	2	1	6	0.66
Document Fraud Detection	8	8	3	4	7	0.57
Embedded Data Repair	8	7	4	4	7	0.53
Financial Statement Rolling	9	8	6	4	8	0.61
HWPX Document Automation	8	8	3	3	6	0.49
Healthcare Cost-Benefit Analysis	9	9	7	3	8	0.51
Industry Correlation Analysis	8	8	5	2	6	0.44
Inventory & Finance Integration	8	7	2	4	8	0.46
Medical Data Standardization	9	9	3	2	7	0.42
OCR Data Extraction	8	8	8	4	8	0.43
Operational Recovery Planning	8	8	1	1	6	0.46
PPT Formatting Optimization	8	8	2	2	5	0.58
Production Capacity Planning	9	6	4	3	7	0.78
SEC 13F Financial Analysis	8	6	5	3	7	0.32
Sales Pivot Analysis	8	8	4	3	7	0.35
Supply Chain Replenishment	9	8	3	1	7	0.63
Weighted Risk Assessment	8	8	1	1	5	0.84
Overall / macro mean	166	7.6	3.5	2.5	6.7	0.53

A.3 WORKFLOW DETAILS

Table 4 provides a compact view of the 20 workflow families. Internal source identifiers are omitted. Instead, the table reports the workflow definition, seed-benchmark lineage, a short task summary, and the corresponding DAEF description. The **Source** column indicates whether the family is derived from a seed task selected from SkillsBench or GDPval.

Table 4: Workflow families and seed-benchmark lineage

Workflow definition	Source	Task summary	DAEF
Industry Correlation Analysis	SkillsBench	Detrend revenue series for paired industries, compute Pearson correlations, and judge cyclical association.	read source series → extract target fields → normalize and align time series → compute annualized and deflated values → compute filtered correlation signals → compare cyclical association → output scalar result.
Financial Statement Rolling	GDPval	Update monthly financial workbooks with reserve merges, accrual rolling, deferred revenue, liabilities, warranties, project cost, rebates, and commissions.	read multi-source ledgers → extract target entries → filter duplicate and invalid rows → align override and adjustment items → compute rolling balances → compare against control totals → update and output summary workbook.
SEC 13F Financial Analysis	SkillsBench	Analyze quarterly SEC 13F filings for cross-period reconciliation, fund categorization, holding changes, issuer ownership, and manager comparison.	read multi-period filings → extract entity records → normalize identifiers → align cross-period holdings → compute target aggregates → compare across periods → output structured analysis.
Supply Chain Replenishment	GDPval	Produce replenishment plans across categories such as bakery, beauty, frozen meals, household routes, pet care, produce, and snacks.	read inventory and transit data → extract target SKUs → filter duplicate and invalid rows → align master data with arrivals → compute coverage and shortage timing → compare against replenishment thresholds → output detail and action sheets.
Production Capacity Planning	GDPval	Build and vary a manufacturing catch-up capacity plan under demand, backlog, overtime, and buffer constraints.	read demand and parameters → retrieve initial state → compute period-level demand and backlog updates → compare candidate resource actions → compute overtime and buffer outcomes → output planning workbook.
Inventory & Finance Integration	GDPval	Integrate replenishment planning with financial rolling scenarios across mixed business contexts.	read multi-source data → extract and normalize keys and time windows → align inventory and finance records → compute rolling deltas and coverage → compare against exception rules → output structured results.
DMAIC Quality Analysis	GDPval	Perform DMAIC Analyze-stage diagnostics across logistics, hospitals, SOC alerts, university IT, DevOps, and field service settings.	read process data → extract analysis window → filter valid samples → compute grouped statistics → compare trend and significance signals → detect anomalies → output structured diagnostics.
Operational Recovery Planning	GDPval	Create recovery plans under multi-scenario operational disruptions such as harvesting, data centers, warehouses, manufacturing, radiology, and returns.	read baselines and constraints → extract planning horizon → compute candidate recovery plans → compare capacity and calendar outcomes → detect backlog gaps → output plan sheets and summaries.

Workflow definition	Source	Task summary	DAEF
Healthcare Cost-Benefit Analysis	GDPval	Compare healthcare supply-chain scenarios under different cycle and batch settings to assess cost-benefit trade-offs.	read inputs and mappings → extract and normalize aliases and keys → filter valid records and exclusions → align coverage with adjustments → compute revenue cost and margin scenarios → compare trade-off thresholds → output structured conclusions.
Medical Data Standardization	SkillsBench	Standardize medical laboratory data across units, precision rules, and formatting conventions.	read raw tables and templates → extract valid rows → align with target schema → normalize numeric formats and units → detect anomalies or missing values → validate required precision → output standardized file.
Distribution Center Auditing	GDPval	Audit distribution-center records for inventory discrepancies, outbound lists, receiving anomalies, return handling, SLA compliance, labor, and trailer detention.	read source data → extract audit fields → compute row-wise rule checks → detect violations and anomalies → compare totals across dimensions → output audit artifacts.
Compensation Scenario Modeling	GDPval	Build and refresh multi-year compensation models for organizations such as orchestras, universities, property firms, airlines, and construction teams.	read assumptions and master data → retrieve model structure and references → align base records across sheets → compute period-wise compensation values → update and compare scenario indicators → output completed model.
Document Fraud Detection	SkillsBench	Review commercial documents across clinics, trials, maintenance logs, travel, and cold-chain records to detect inconsistencies and suspicious items.	read source documents → extract candidate records and references → normalize names and identifiers → align cross-document evidence → compare claimed facts with authorizations → detect anomaly types → output structured flags.
Embedded Data Repair	SkillsBench	Repair or refresh embedded spreadsheet objects inside PowerPoint files for domains such as chemistry, catalysts, or foreign exchange.	read document container → extract embedded tables and target cells → retrieve label mappings → update cell values and formulas → validate dependencies → output repaired document.
OCR Data Extraction	SkillsBench	Extract structured records from scanned images such as legal settlements, measurements, orders, invoices, fuel slips, and medicine labels.	read scanned images → extract OCR fields and target records → normalize dates amounts and codes → align with template rows → validate supplemented references → output structured results.
HWPX Document Automation	SkillsBench	Fill Korean HWPX templates for clinics, announcements, inventory, proposals, renewals, audits, vendor directories, and training feedback.	read templates and data sources → extract placeholders and values → normalize transformed values and enumerations → update filled content and structure → validate document integrity → output completed file.
Cross-Format Data Reconciliation	SkillsBench	Compare archived PDF snapshots with current Excel data to find added, deleted, or modified records across domains.	read archived snapshot and current spreadsheet → extract primary keys and records → normalize and align across formats → compare additions deletions and modifications → detect field-level differences → output structured results.

Workflow definition	Source	Task summary	DAEF
Weighted Risk Assessment	SkillsBench	Implement weighted risk formulas in Excel for scenarios such as API SLA, campus budgeting, factories, hospitals, energy, cloud reliability, and ports.	read workbook and lookup tables → retrieve input and output ranges → update formulas with target cells → compute weighted indicators and risk scores → validate workbook consistency → output result workbook.
PPT Formatting Optimization	SkillsBench	Normalize image-title formatting in PowerPoint decks by unifying fonts, positions, and index pages.	read presentation objects → extract target titles and tags → normalize text font and layout attributes → update positions and alignment → validate summary-page consistency → output formatted document.
Sales Pivot Analysis	SkillsBench	Build pivot-style analysis from PDF catalogs and Excel transaction data for sales, budgets, payroll, registration, inventory, libraries, quality, and grades.	read multi-source data → extract and normalize fields and dimensions → align supplementary joins → compute pivot indicators → compare grouped summaries → output report workbook.

A.4 DAEF Node Vocabulary

Table 5 lists the controlled node labels used when annotating seed-task workflow graphs. The vocabulary is intentionally compact and coarse-grained so that it captures transferable procedural roles rather than low-level commands or domain-specific business concepts.

Table 5: Controlled DAEF node vocabulary

Node label	Operational meaning	Typical examples in released workflows
read	Load or inspect source files, tables, documents, images, or structured records	Read ledgers, filings, scanned images, templates, workbooks, PPTX containers, and PDF snapshots
extract	Identify and isolate target fields, records, entities, or patterns from complex or unstructured sources	Extract OCR fields from scanned images, extract candidate records from documents, extract embedded tables from containers
retrieve	Locate or select target states, ranges, placeholders, or action candidates from available context	Retrieve initial state, retrieve target entries, retrieve audit fields, retrieve target cells, retrieve placeholders, retrieve input/output ranges
normalize	Standardize schema, keys, units, formatting, aliases, or transformed values before downstream reasoning	Normalize entity identifiers, numeric formats, lab units, dates, codes, font attributes, and transformed enumerations
filter	Retain valid records and remove irrelevant, duplicate, malformed, or excluded cases	Filter duplicate ledger rows, filter excluded healthcare cases, filter invalid samples, filter malformed source entries
align	Join, reconcile, or map multiple sources under a shared key, structure, or layout	Align time series, align cross-period holdings, align inventory with arrivals, align finance records, align label mappings, align records across formats

Table 5 (continued)

Node label	Operational meaning	Typical examples in released workflows
compute	Derive intermediate or final quantities through formulas, aggregation, scoring, iterative updates, or planning operations	Compute rolling balances, compute coverage windows, compute backlog updates, compute revenue and margin scenarios, compute weighted risk scores, compute pivot indicators
compare	Contrast values, scenarios, plans, grouped summaries, or cross-period states against a threshold or reference target	Compare cyclical association, compare control totals, compare replenishment thresholds, compare candidate resource actions, compare trade-off thresholds, compare grouped summaries, compare record changes
detect	Identify anomalies, violations, exceptions, gaps, or field-level changes that require action	Detect suspicious claims, detect audit violations, detect missing values, detect backlog gaps, detect anomaly types, detect field-level differences
update	Modify, fill, repair, or refresh existing artifacts such as cells, documents, or configurations	Update embedded cell values, update workbook formulas, update document formatting, update template fields, update configuration values
validate	Check whether outputs satisfy structural, numeric, dependency, formatting, or integrity constraints before release	Validate required precision, validate workbook consistency, validate formulas and dependencies, validate cache integrity, validate reference supplements, validate summary-page consistency
output	Write the final artifact in the required structure, file format, workbook, document, or report layout	Output scalar results, output structured diagnostics, output linked summary workbooks, output audit artifacts, output repaired documents, output formatted presentations

DAEF annotation rules. Each task is annotated with 5–8 meta-step nodes whose labels come from the controlled vocabulary in Table 5. Dependency edges run from prerequisites to subsequent operations without cycles; parallel operations may share a downstream node. Annotators also provide a brief operational rationale for each node label.

DAEF agreement screening. Two expert annotators independently produce DAEF annotations for each candidate task in the final screening pool, from which 30 seed tasks are retained. Each annotator reviews the other's graph for agreement on meta-step decomposition, node labels, and dependency structure. Candidate-level pre-screening agreement is 85% under this joint criterion. A candidate passes only when both annotators judge the annotations mutually consistent; otherwise it is excluded from family construction.

A.5 Allowed Surface Variation Types

Table 6 summarizes the variation dimensions that annotators may allow within a family while p-reserving a shared DAEF. These dimensions are intended to alter surface realization and difficulty without introducing a new workflow core.

B Skill Evolution Details

B.1 Evaluation Configuration

All ten model comparisons use the same task assets, task order, verifier, and execution harness. The configuration used for the Terminus-2 runs is summarized below.

Table 6: Allowed surface variation types within a DAEF family

Variation type	What may change	Representative examples in the release
Domain entities	Business objects, named fields, organizational units, or scenario semantics	Vendor vs. patient vs. warehouse records; holdings vs. inventory vs. clinical measurements
File format	Surface container or artifact type while preserving the same procedural core	XLSX vs. CSV vs. JSON inputs; PPTX embedded spreadsheets vs. standalone workbooks; OCR image vs. PDF table
Data scale	Number of rows, time periods, scenarios, or cross-file joins	More SKUs, longer time horizons, larger audit logs, additional quarterly filings
Noise and quality	Missing values, alias ambiguity, OCR noise, inconsistent formatting, or extra distractors	Misspelled headers, low-quality scans, inconsistent units, partial records, cluttered slides
Constraint regime	Thresholds, policy rules, reconciliation targets, or validation tolerances	Different SLA cutoffs, coverage rules, accounting constraints, exception thresholds
Instruction surface	Wording, ordering, or presentation of the task request without changing the workflow core	Domain-specific phrasing, reformulated reporting goals, alternate output wording
Difficulty level	The amount of bookkeeping, exception handling, or multi-step reasoning required	Additional scenario branches, more exception rules, harder reconciliation chains, denser formatting checks

Table 7: Evaluation configuration for the Terminus-2 skill-evolution experiments.

Setting	Configuration
Agent / harness	Terminus-2 with Harbor 0.18.0 (`harbor[daytona]`)
Task prompt	Terminus-2 default terminal-task prompt; no additional system prompt
Skill evolution	Post-task patcher updates `SKILL.md` through `summary`, `upsert_files`, and `delete_paths`
Turns / timeout	Maximum 500 turns; agent timeout 1,800 s; verifier timeout 300–1,800 s
Retry	One sample per task; up to two retries on execution errors
Decoding	Task-agent decoding uses model-service defaults; skill patcher uses temperature 0
Tools	Shell commands through tmux; no additional MCP tools
Environment	Daytona sandbox; `python:3.12-bookworm`; 1–4 CPU, 2–4 GB RAM, 10 GB storage

B.2 Verifier Audit

We conduct a domain-stratified manual audit of 56 trajectory outputs, randomly sampling a balanced set of verifier-accepted and verifier-rejected outputs across the five benchmark domains. Human review adjudicates whether each output satisfies the task requirements. We report false positives (FP), false negatives (FN), and agreement with the verifier. We additionally construct corrupted variants from correct outputs by deliberately changing a correct result into an incorrect one, then test whether the verifier rejects the corrupted output. This separate stress test is reported as corruption detection.

Table 8 shows 91.07% human–verifier agreement. The audit finds no false positives and five false negatives: within this sample, verifier disagreements reject human-judged valid outputs rather than accept incorrect ones. Corruption detection is 100% in every domain.

Table 8: Manual verifier audit by benchmark domain. FP denotes a verifier-accepted output judged incorrect by the human audit; FN denotes a verifier-rejected output judged correct. Human–verifier agreement is the fraction of audited outputs receiving the same correctness judgment. Corruption detection measures the fraction of deliberately corrupted variants of correct outputs that the verifier rejects.

Domain	Audited Outputs	FP	FN	Human–Verifier Agreement ↑	Corruption Detection ↑
Finance & Economics	12	0	1	91.67%	100.00%
Operations & Supply Chain	14	0	2	85.71%	100.00%
Healthcare & Life Sciences	8	0	0	100.00%	100.00%
Governance & Strategy	8	0	0	100.00%	100.00%
Data & Document Intelligence	14	0	2	85.71%	100.00%
Overall	**56**	**0**	**5**	**91.07%**	**100.00%**

B.3 Skill Patch Format

Each skill patch in the current framework is represented as a JSON object with three top-level fields:

- `summary`: a natural-language description of the newly extracted lesson or repair.
- `upsert_files`: a mapping from file paths to new or updated file contents.
- `delete_paths`: a list of obsolete files to remove from the skill library.

This schema makes file-level updates inspectable: patch history supports tracking how skills change over time and where malformed or redundant content accumulates in the library.

C EXPERIMENTS AND ANALYSIS DETAILS

C.1 ALL-TASK SCOPE AND STATISTICAL PROCEDURES

The same Daytona/Terminus-2 task records underpin all model summaries and trajectory diagnostics, excluding E2B runs. Every analysis uses the common grid of ten models and 166 tasks per model. Table 9 reports the model-level success counts, and the analysis script and task-level records are under `analysis/trajectory_analysis`.

Table 9: Model-level completion counts on the common 166-task grid. Completion in Table 1 is the corresponding success count divided by 166.

Model	N	Vanilla successes	Evolved successes	Δ successes
GPT 5.4	166	83	80	-3
GPT 5.5	166	84	85	+1
Claude Opus 4.8	166	71	77	+6
Gemini 3.5 Flash	166	79	85	+6
Qwen3.8 Max	166	80	88	+8
Qwen3.7 Max	166	83	85	+2
Qwen3.7 Plus	166	81	82	+1
MiniMax M3	166	79	80	+1
DeepSeek V4 Pro	166	79	93	+14
Kimi K3	166	75	89	+14

A task is successful when its verifier reward is ≥ 1. Skill access requires an observed command that reads a file under `/harbor/skills/<skill>/`; prompt listings and the `shared-skills-bootstrap` placeholder do not count.

The model-macro effect averages the ten displayed model differences in Table 1; its 95% interval resamples those ten model effects 100,000 times and summarizes variation across evaluated models. The model-level correlation uses the same table. Across the common grid, 115 outcomes change from failure to success and 65 from success to failure. Position and skill-access summaries characterize adaptation under the fixed task progression and agent-selected retrieval policy.

Trajectory-derived turns and input/output tokens cover the task agent and exclude the post-task skill-patch call. The main table's cost entries are reported summary values rounded to two decimal places, so its percentage cost differences are calculated at that displayed precision. A complete comparison of learning cost would additionally account for patch generation and any infrastructure charges.

C.2 WORKFLOW AND POSITION HETEROGENEITY

Table 10 gives all 20 family-level all-task outcomes. Figure 5 makes the directional spread visible. The domain heatmap in Figure 6 summarizes broad patterns, while the adjacent family heatmap in Figure 7 exposes model–family sign changes hidden by domain aggregation. Ceiling and floor families contextualize the measured gains: Document-Fraud-Detection remains at 100% in both conditions, while four families remain at 0%, concentrating the observable transfer signal in workflows with meaningful performance headroom.

Table 10: Task-level outcomes by workflow family, pooled over ten models on the same 166-task-per-model grid as Table 1. Skill use is based on observed file reads.

Workflow family	N	Vanilla (%)	Evolved (%)	Δ (pp)	Use (%)
Weighted Risk Assessment	80	11.25	40.00	+28.75	82.50
Embedded Data Repair	80	48.75	70.00	+21.25	86.25
SEC 13F Financial Analysis	80	18.75	32.50	+13.75	77.50
Production Capacity Planning	90	76.67	88.89	+12.22	63.33
Distribution Center Auditing	80	72.50	76.25	+3.75	42.50
Operational Recovery Planning	80	50.00	52.50	+2.50	35.00
Supply Chain Replenishment	90	98.89	98.89	+0.00	41.11
Sales Pivot Analysis	80	0.00	0.00	+0.00	77.50
Inventory & Finance Integration	80	61.25	61.25	+0.00	71.25
Financial Statement Rolling	90	0.00	0.00	+0.00	80.00
Document Fraud Detection	80	100.00	100.00	+0.00	71.25
DMAIC Quality Analysis	90	0.00	0.00	+0.00	51.11
Compensation Scenario Modeling	80	0.00	0.00	+0.00	78.75
Healthcare Cost Benefit Analysis	90	44.44	43.33	-1.11	27.78
Cross Format Data Reconciliation	80	100.00	98.75	-1.25	66.25
Medical Data Standardization	90	36.67	34.44	-2.22	76.67
PPT Formatting Optimization	80	11.25	8.75	-2.50	81.25
HWPX Document Automation	80	96.25	92.50	-3.75	85.00
OCR Data Extraction	80	37.50	32.50	-5.00	58.75
Industry Correlation Analysis	80	96.25	91.25	-5.00	41.25

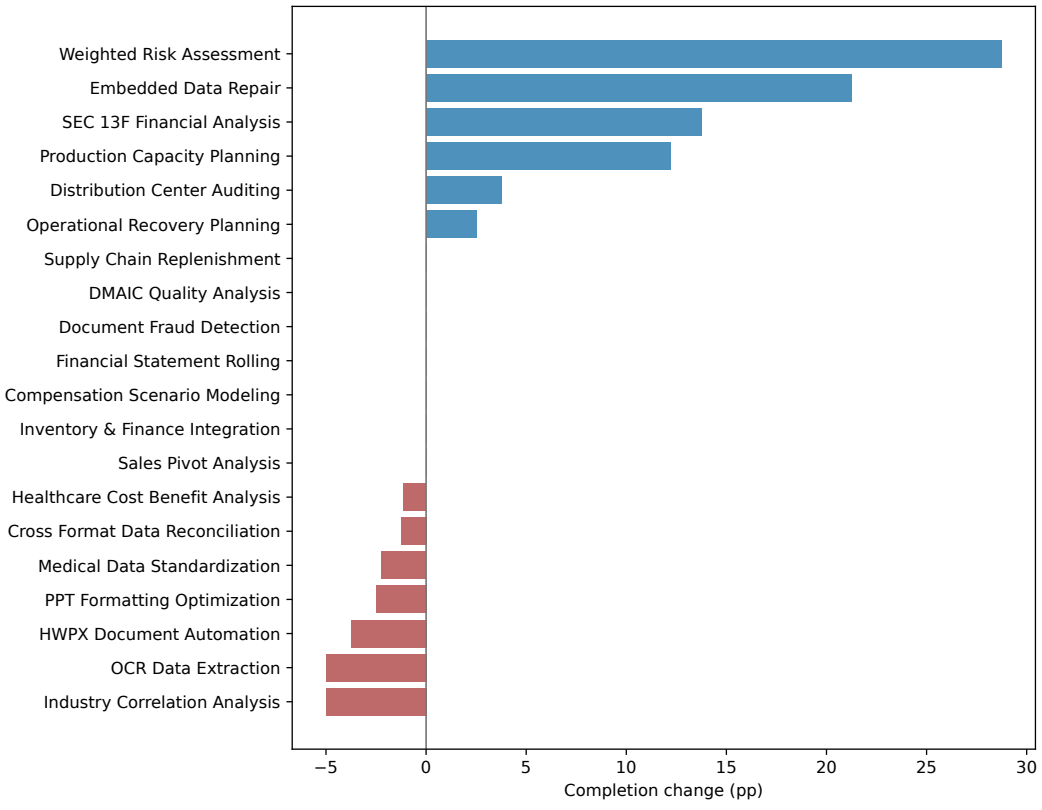


Figure 5: Pooled completion change across the 20 workflow families on the common 166-task-per-model grid.

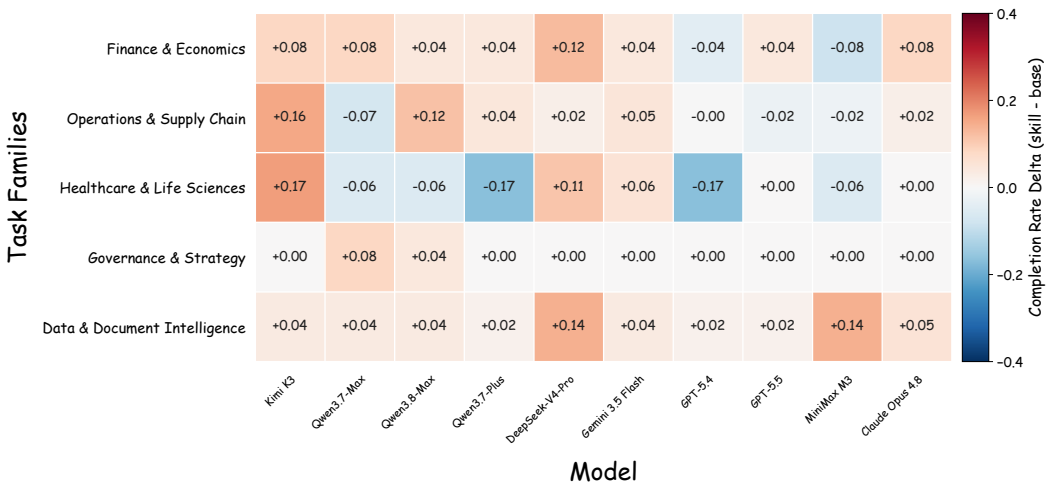


Figure 6: Completion change aggregated by model and benchmark domain on the common 166-task grid. Cell values are percentage-point differences.

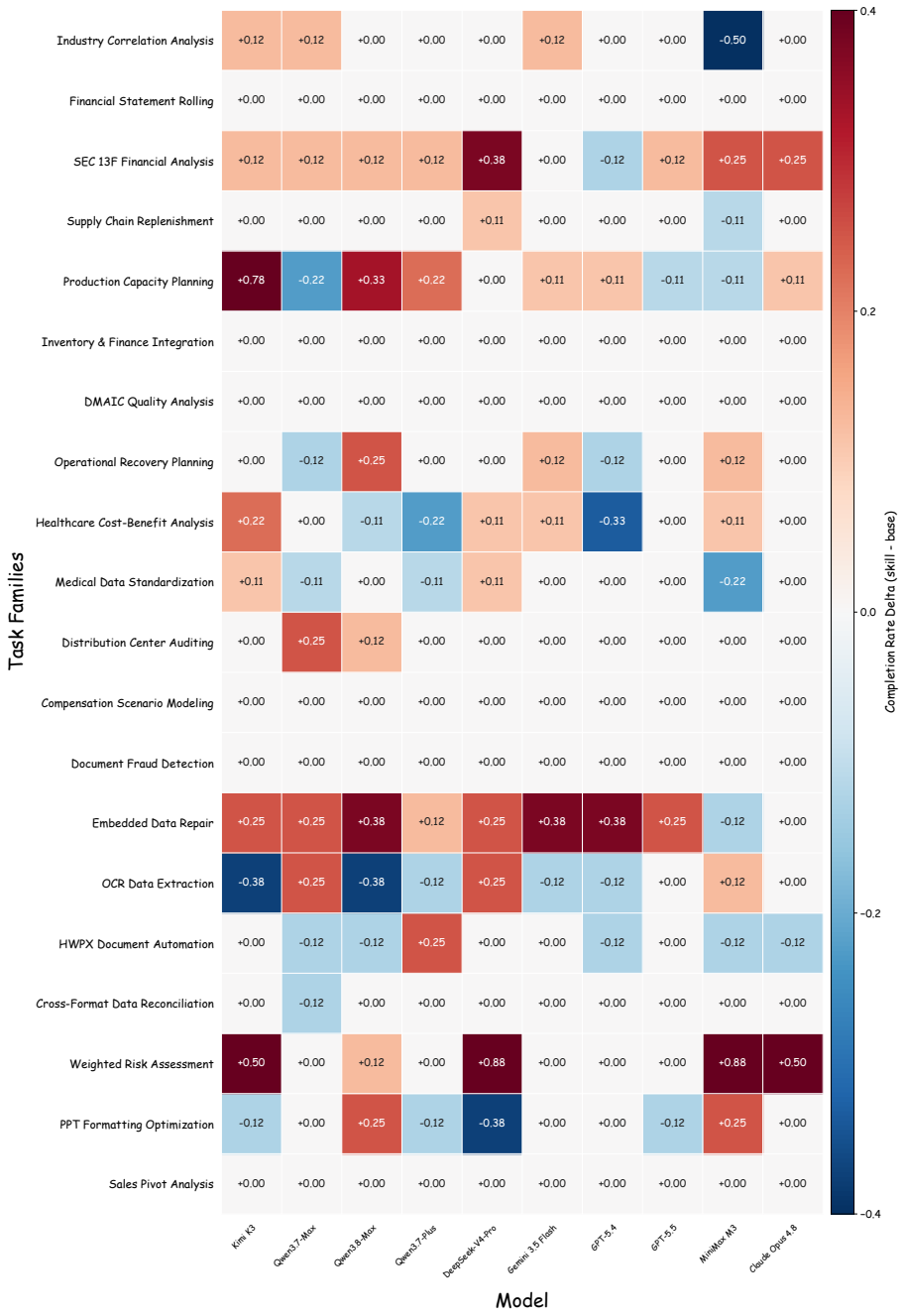


Figure 7: Completion change for each model–family sequence on the common 166-task grid. Cell values are percentage-point differences.

Table 11 reports the observed effect and skill-access rate by absolute task position. Figure 8 expands the pooled position result from Figure 4(b) into model-specific trajectories, while Figure 9 groups positions into early, middle, and late stages. There are fewer observations at position nine because only six families contain nine tasks. All other positions include 200 model–task observations. The fixed difficulty order makes position a joint measure of elapsed learning opportunity and planned task progression.

Table 11: Task-level results by absolute position in the fixed family order. Each of the 166 tasks per model contributes once. Position 1 has no previously evolved skill available.

Position	N	Vanilla (%)	Evolved (%)	Δ (pp)	Use (%)
1	200	54.50	54.00	-0.50	0.00
2	200	49.00	48.50	-0.50	77.00
3	200	52.50	55.50	+3.00	79.00
4	200	52.00	56.50	+4.50	72.00
5	200	51.00	56.00	+5.00	73.50
6	200	41.50	41.00	-0.50	73.50
7	200	44.50	52.50	+8.00	73.50
8	200	41.50	45.00	+3.50	66.50
9	60	35.00	43.33	+8.33	66.67

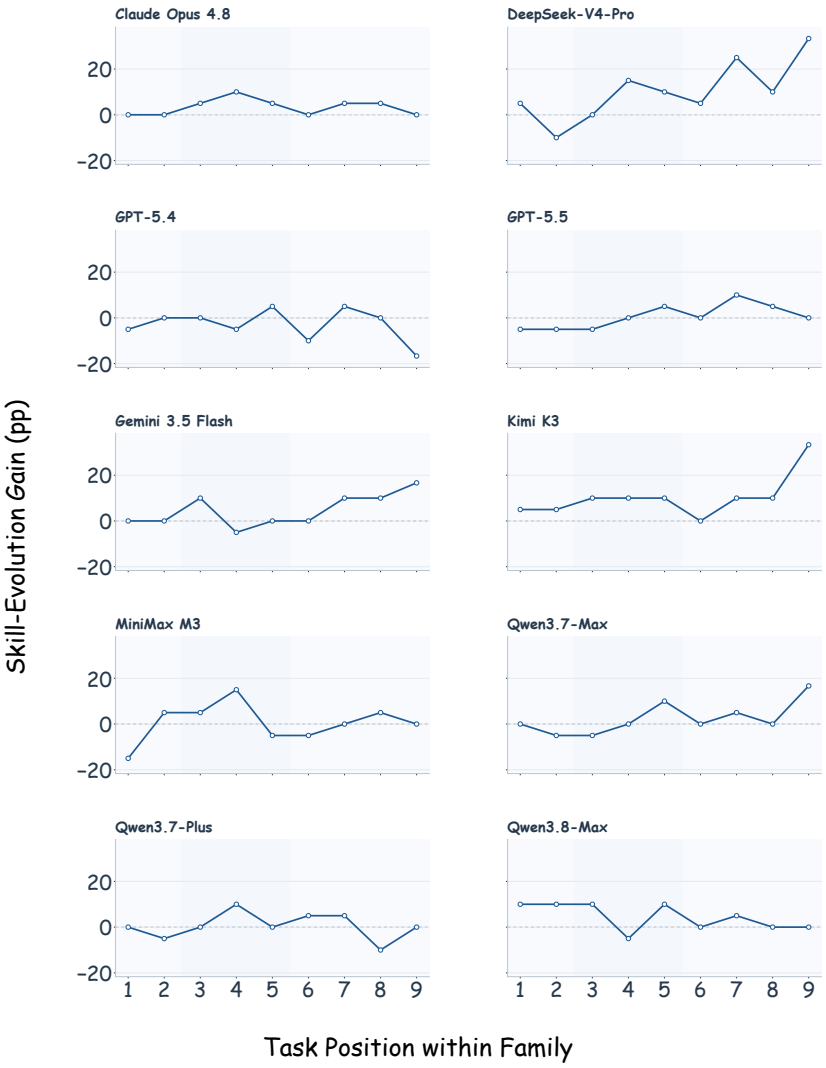


Figure 8: Model-specific skill-evolution gain by absolute task position within each workflow family. The ten panels expand the pooled position view in Figure 4(b), showing that the aggregate temporal pattern masks substantial variation across models.

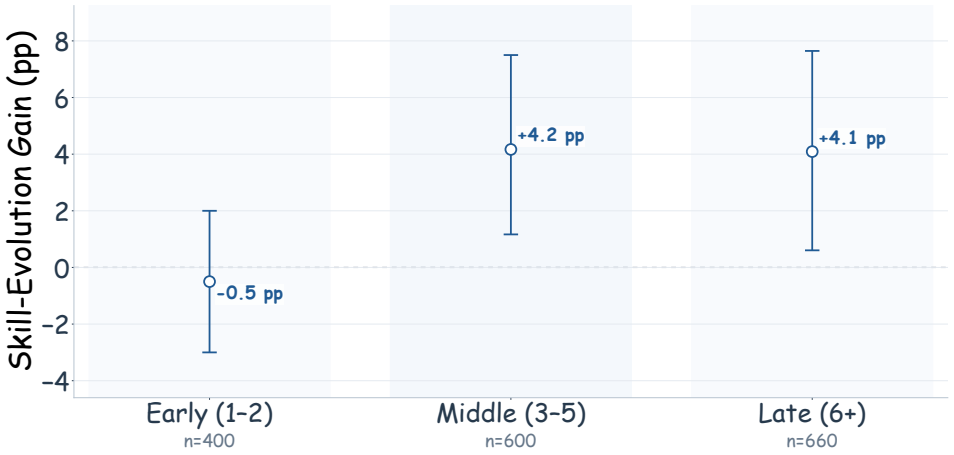


Figure 9: All-task completion gain grouped by family position: early (tasks 1–2), middle (3–5), and late (6 onward). Labels give the number of model–task observations.

C.3 Performance–Resource Frontiers

Figures 10–12 place the reported benchmark-level completion rates against cost, interaction turns, and task-agent output tokens, respectively. Dashed segments connect each model's vanilla and skill-evolution conditions. The frontier traces the non-dominated points in each two-metric projection, preserving separate views of cost, interaction turns, and output tokens. These benchmark-level summaries retain the reporting precision of Table 1.

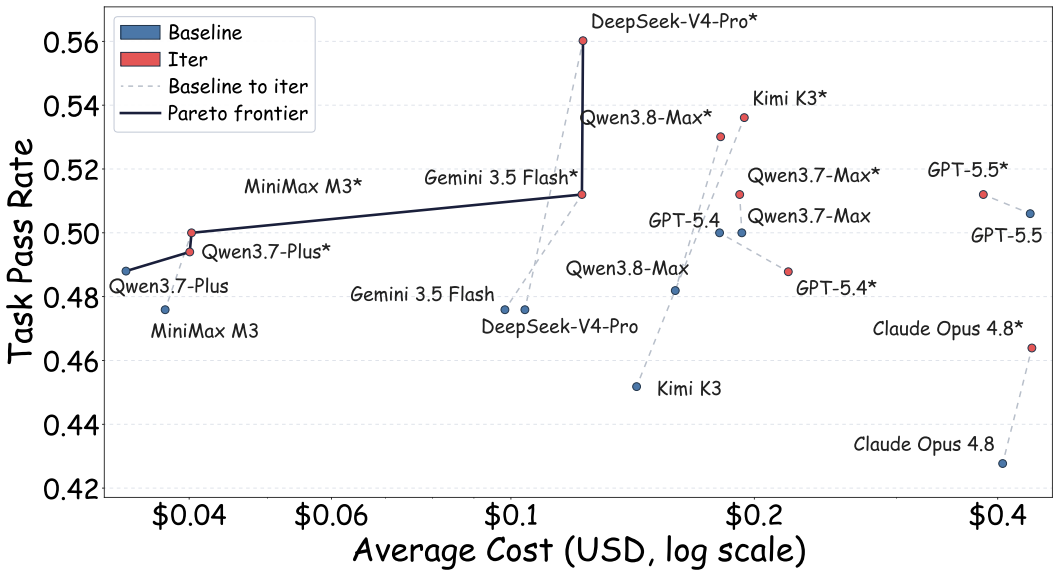


Figure 10: Benchmark-level completion versus reported average cost per task (USD, logarithmic horizontal axis). Blue and red points denote vanilla and skill evolution; dashed segments connect conditions for the same model, and the solid line traces the plotted Pareto frontier.

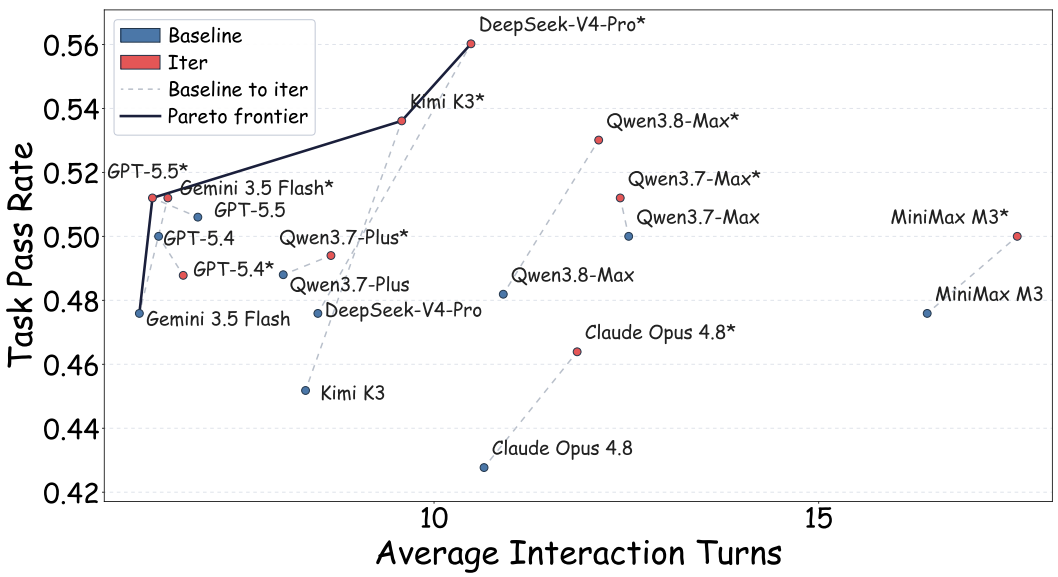


Figure 11: Benchmark-level completion versus mean task-agent interaction turns. Point colors, connecting segments, and frontier follow Figure 10.

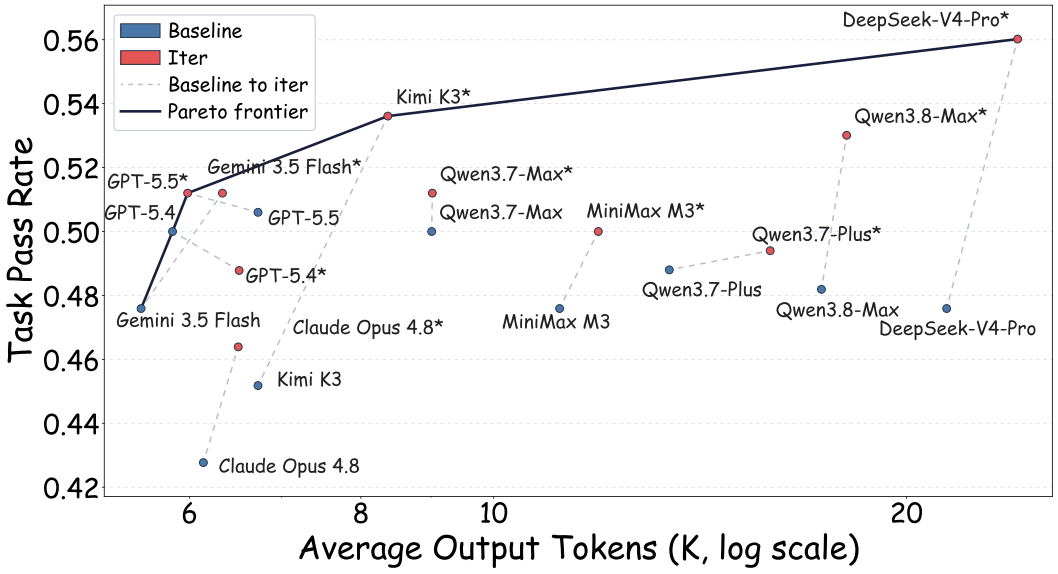


Figure 12: Benchmark-level completion versus mean task-agent output tokens (thousands; logarithmic horizontal axis). Point colors, connecting segments, and frontier follow Figure 10. Post-task patch-generation tokens are outside this measure.

C.4 Patch Events and Trajectory Cases

Across 200 model–family sequences, 247 tasks create a skill, 1,359 revise one, and three delete one. File-level changes comprise 687 creations, 1,773 updates, and nine deletions. The number of final skills has almost no linear association with all-task gain across sequences ($r \approx 0.01$). Together with outcome metrics, these counts distinguish the scale of library activity from the quality of the resulting procedures. The released analysis tables `per_task.csv` and `skill_evolution_events.csv` link each task to its trajectory, patch, and change record.

Figure 13 shows the number and types of files present in each model's final family-local skill libraries, aggregated across families. All models produce `SKILL.md` files alongside varying numbers of helper scripts and other artifacts. DeepSeek-V4-Pro and MiniMax M3 have the largest file totals (119 and 116), while the sequence-level correlation between final skill count and all-task gain remains near zero. File production and successful transfer are distinct measurements.

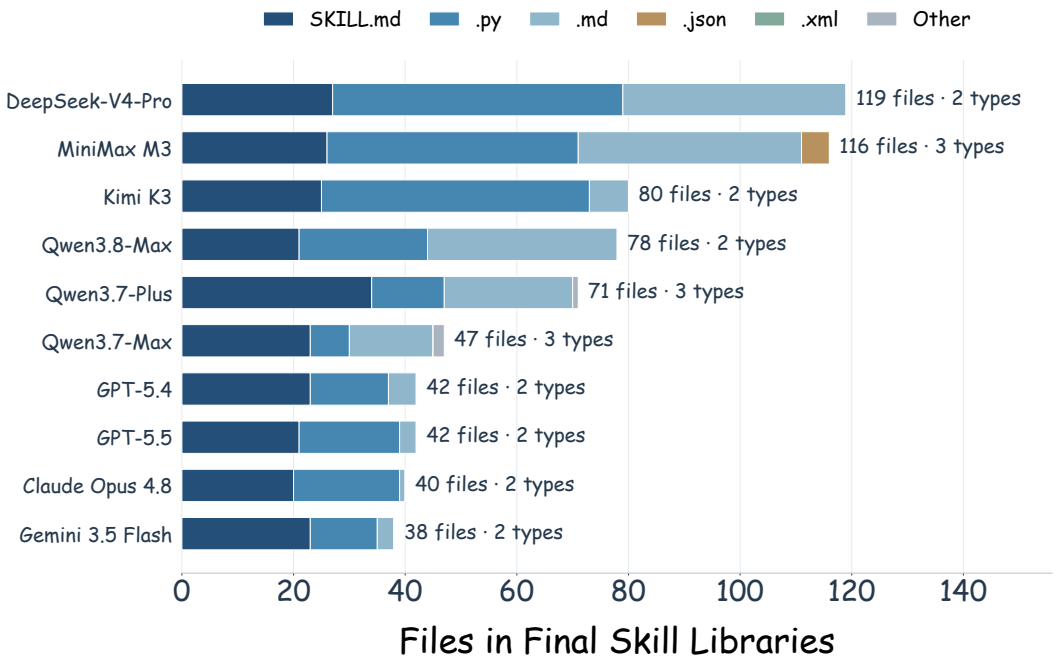


Figure 13: File-type composition of final evolved skill libraries, summed over workflow families for each model. Labels give total files and the number of non-`SKILL.md` file types; the color key distinguishes `SKILL.md`, Python scripts, other Markdown, JSON, XML, and remaining types.

Reusable procedure after failure. For DeepSeek-V4-Pro on Weighted-Risk-Assessment, the first task fails in both conditions. Its post-task patch creates `xlsx-formula-automation`, adding LibreOffice recalculation, cached-value checks, and independent expected-value checks to an Excel workflow. All seven later tasks read that skill and change from vanilla failure to iterative success: the family moves from 0/8 to 7/8. The final task takes 12 rather than seven turns and produces 21,203 rather than 12,247 output tokens, illustrating that completion gains and resource efficiency form complementary dimensions of skill quality.

Selective reuse with lower task-agent cost. For Qwen3.7-Max on OCR-Data-Extraction, the family moves from 5/8 to 7/8 even though only four tasks read an evolved skill. On `task_family_nested_fuel_packets`, the iterative agent reads `ocr-data-extraction`, succeeds where vanilla fails, and uses 11 rather than 28 turns. The stored procedure covers recursive file discovery, mandatory-keyword filtering, and OCR identifier normalization. This paired case concretely illustrates how reusable guidance can focus exploration and improve task-agent efficiency.

Over-specific skill and negative transfer. Qwen3.7-Max's Operational-Recovery-Planning family moves from 6/8 to 5/8. After a successful fifth task, a patch creates a narrower `production-recovery-planning` skill. The next task reads it and fails despite vanilla success; its generated script hard-codes purchase-order dates and quantities and checks them against a matching hard-coded dictionary. A later task again reads the skill and times out without a valid artifact. The observed hard-coding and self-confirming checks connect over-specific procedural memory with negative transfer in this sequence.

High access, wrong validation target. DeepSeek-V4-Pro's PPT-Formatting-Optimization family falls from 3/8 to 0/8 while its `pptx-openxml-editing` skill is read on all seven later tasks. On `wildlife-field-guide-caption-cleanup`, the skill-supplied validator confirms package integrity, but the task verifier rejects the result because the requested formatting was not established by that check. The case shows why skill access and repeated revision can coexist with negative transfer.

Additional cases show Qwen3.7-Max gradually revising a SEC-13F analysis skill before one later rescue, and Qwen3.7-Plus creating eight narrow Distribution-Center-Auditing skills without a single observed read. These cases center evaluation on procedural abstraction, appropriate selection, and independent outcome checks as the outcome-bearing properties of learned skills, complementing file-count and access-count measures.

C.5 Interpretation and Scope

This evaluation measures end-to-end adaptation through a file-based patch mechanism within fixed workflow-family sequences, where trajectory feedback and skill content jointly constitute the learning process. Fixed task order aligns experience accumulation with planned difficulty progression, so position effects characterize the full sequential adaptation process. Family-local resets isolate within-workflow procedural transfer, while task diversity and DAEF annotations support comparisons across grounded variants. These design choices establish a controlled account of workflow-scoped skill evolution and provide a foundation for extensions to alternative memory mechanisms and cross-family transfer.