Trace2Skill: Distill Trajectory-Local Lessons into Transferable Agent Skills

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents increasingly rely on domain-specific skills, yet manually authoring such skills does not scale, and skills generated purely from parametric knowledge often miss critical operational pitfalls. We introduce Trace2Skill, a framework that consolidates broad execution trajectories in parallel into a unified skill directory through inductive reasoning over agent experience. Trace2Skill supports both deepening existing human-written skills and creating useful skills from weak LLM-generated drafts. Experiments demonstrate the effectiveness of Trace2Skill across diverse domains, including office workflows, math reasoning, and vision QA. Importantly, the evolved skills are not merely memorized artifacts of the trajectories used to create them: they often transfer across model scales, across model families, and to out-of-distribution settings. For example, skills evolved from Qwen3.5-35B trajectories improve a Qwen3.5-122B agent by up to 57.65 percentage points on WikiTableQuestions. Further analyses show that Trace2Skill outperforms sequential skill editing and ReasoningBank-style retrieval memories, compresses recurring failures and workarounds into standard operating procedures (SoPs), and yields portable skills that can be reused without parameter updates or test-time retrieval.[1]

1 Introduction

LLM-based agents increasingly rely on *skills*: reusable documents that encode task procedures, domain knowledge, and operational guidelines (Anthropic, 2026d; Zhou et al., 2026b). As agents move into specialized file- and tool-use workflows, the bottleneck is no longer only model capability, but also the ability to create and maintain high-quality skills for each domain (Han et al., 2026; Li et al., 2026a; Anthropic, 2026a; Liang et al., 2026; Zhou et al., 2026b). Human-written skills can help, but they are not uniformly beneficial across agents: in Table 1, the official `xlsx` skill improves a 122B spreadsheet agent while hurting a 35B agent on the same benchmark. Generating skills purely from parametric knowledge is also brittle, because such drafts often lack the concrete failure modes, workarounds, and operational details exposed by actual execution traces (Li et al., 2026b; Jiang et al., 2026; Zhou et al., 2026b).

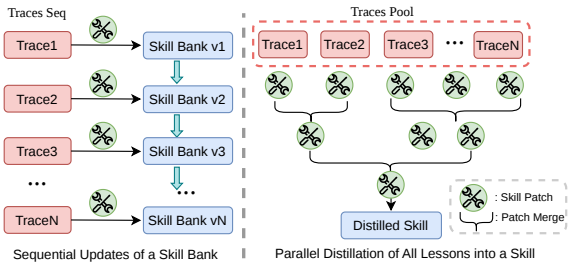

Figure 1: **Left:** sequential online skill evolution edits the skill after each incoming trace. **Right:** Trace2Skill analyzes many traces in parallel and hierarchically consolidates recurring lessons into one portable skill.

Recent systems therefore use agent experience to evolve skills or memories online (Yang et al., 2026; Xia et al., 2026a; Alzubi et al., 2026; Zhou et al., 2026a; Jiang et al., 2026; Zhou et al., 2026b). This direction is promising, but many existing approaches either store trajectory-local lessons for retrieval or edit skills sequentially as new traces arrive (Fig. 1, left). Such designs can fragment reusable knowledge across a large memory or skill collection, and sequential editing can make later skills depend on the order of earlier updates (Li, 2026). **Human experts usually work differently:** they inspect broad traces, abstract recurring patterns, and write compact procedures that are reusable across cases.

We introduce Trace2Skill, a framework for turning execution traces into portable skills. Rather

[1] Code: https://anonymous.4open.science/r/anonymous-skill-evolver-E41B

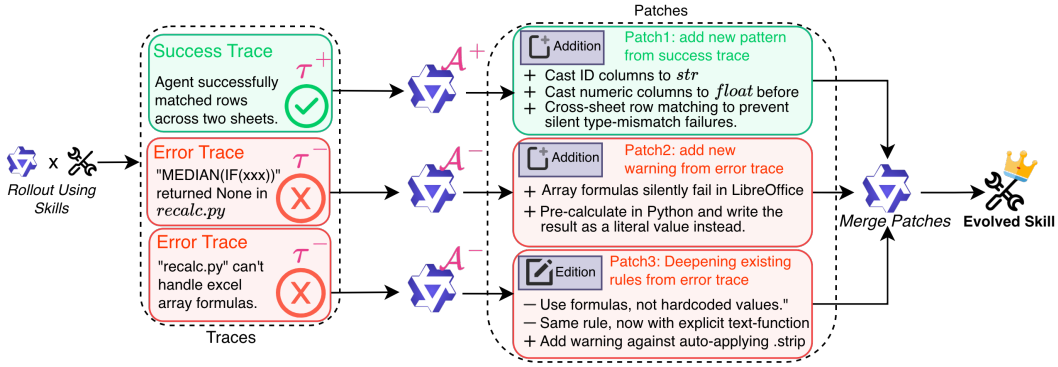

Figure 2: Overview of Trace2Skill's three-stage pipeline: (1) roll out a frozen agent to collect labeled success and failure trajectories, (2) propose trajectory-level skill patches in parallel with separate error and success analysts, and (3) hierarchically merge all patches into one portable skill directory.

than retrieving per-episode memories at test time or absorbing traces through order-dependent sequential edits, Trace2Skill analyzes many traces jointly and consolidates recurring lessons into a single skill directory (Fig. 1, right). The same mechanism supports two common use cases: *deepening* an existing human-written skill and *creating* a useful skill from a weak LLM-generated draft. This many-to-one consolidation acts as an inductive reasoning step over agent experience (Xiong et al., 2025; Li et al., 2025; Lin et al., 2025), while preserving the portability of ordinary skill files.

Experiments show that trajectory-grounded skills improve performance while remaining portable across models, benchmarks, and task domains. In spreadsheet workflows, Trace2Skill both strengthens Anthropic's official `xlsx` skill and creates useful skills from scratch. The resulting skills transfer across model scales and families: for example, a skill evolved with Gemma-4-31B-it (Google, 2026) improves Qwen3.5-122B (Qwen, 2026). They also generalize to out-of-distribution (OOD) data and tasks, such as transferring from spreadsheet editing to table QA. The benefits extend beyond spreadsheets: Trace2Skill improves math reasoning and DocVQA, and further strengthens Anthropic's official PDF, DOCX, and PPTX workflow skills. Further analyses show that: parallel consolidation is much faster and generally stronger than order-dependent skill editing; a single distilled skill outperforms ReasoningBank-style episodic retrieval (Ouyang et al., 2026); and agentic analysts produce better patches by inspecting artifacts and validating fixes. Qualitatively, the learned patches are not trajectory-specific tips: they coalesce into reusable SoPs, while patch-selection studies show that their value is often combinatorial, making holistic consolidation more reliable than greedy local selection.

We contribute: (1) **Trace2Skill**, a framework for automatic skill creation and deepening. It mirrors human skill writing: building broad prior knowledge through extensive trajectory analysis before drafting skills (§ 2). (2) **Broad empirical evidence** that trajectory-grounded evolution yields skills that transfer effectively across LLM scales, families, and OOD tasks (§ 3). (3) **Comprehensive analysis** showing why consolidation works: parallel merging improves over sequential updates, one consolidated skill beats retrieval-based reasoning memories, agentic diagnosis improves patch quality, and patch value is often combinatorial (§ 4).

2 Trace2Skill

Fig. 2 shows the three-stage Trace2Skill pipeline: collect trajectories, propose trajectory-level patches in parallel, and consolidate them into one portable skill. We first define the skill-evolution objective, then describe each stage.

2.1 Skill and Problem Formalization

A skill is a human-readable directory $\mathcal{S} = (M, \mathcal{R})$, where M is the root `SKILL.md` and $\mathcal{R}$ contains auxiliary references, scripts, or assets. The root document stores broadly applicable procedural knowledge, while auxiliary files provide deterministic tools or lower-frequency details. Let π_θ be a fixed LLM agent using skill $\mathcal{S}$ at inference time, and let $\mathcal{P}(\mathcal{S}; \pi_\theta, \mathcal{D})$ denote the pass rate of that agent on task set $\mathcal{D}$. Given an evolving set $\mathcal{D}_{\text{evolve}}$ and a disjoint test set $\mathcal{D}_{\text{test}}$, skill evolution constructs

a new skill from evolving-set trajectories without updating θ:

$$\mathcal{S}^* = \mathcal{E}(\mathcal{S}_0, \mathcal{D}_{\text{evolve}}; \pi_\theta), \quad \mathcal{P}(\mathcal{S}^*; \pi_\theta, \mathcal{D}_{\text{test}}) > \mathcal{P}(\mathcal{S}_0; \pi_\theta, \mathcal{D}_{\text{test}}). \tag{1}$$

We evaluate two initializations for $\mathcal{S}_0$: a human-expert skill and an LLM-generated draft from parametric knowledge alone.

2.2 Stage 1: Trajectory Generation

We use a ReAct-style harness (Yao et al., 2023). For each evolving-set task, the fixed agent runs with $\mathcal{S}_0$ and produces a trajectory τ_i containing the query, reasoning/tool-use history, final output, and a binary correctness outcome. The resulting corpus $\mathcal{T}$ is split into failures $\mathcal{T}^-$ and successes $\mathcal{T}^+$. Trajectory generation is parallel across evolving-set problems, so the trace corpus can be collected independently before patch proposal. See prompt templates in § I.1.

2.3 Stage 2: Parallel Patch Proposal

A group of analyst sub-agents independently proposes skill patches from individual trajectories. Failures are sent to an error analyst $\mathcal{A}^-$, successes to a success analyst $\mathcal{A}^+$, and the analysts read $\mathcal{S}_0$ and propose a patch for it, leading to a patch pool $\mathcal{P} = \mathcal{P}^- \cup \mathcal{P}^+$.

The analyst roles are intentionally asymmetric. $\mathcal{A}^+$ uses a single-pass workflow to identify reusable behavior patterns from successful trajectories. $\mathcal{A}^-$ uses a ReAct-style loop that can inspect traces and artifacts, compare outputs against ground truth, and validate candidate fixes before proposing a patch. Failures that cannot be causally explained are excluded, ensuring $\mathcal{P}^-$ is grounded in verified failure mechanisms rather than log-only guesses (Ouyang et al., 2026). Both roles are prompted to write concise, actionable patches following skill-writing guidance (Anthropic, 2026a). Analyst prompt templates and representative example patches are provided in § I.2.

2.4 Stage 3: Patch Consolidation

Stage 3 consolidates the patch pool into one coherent update p^* and applies it to $\mathcal{S}_0$. Patches are merged hierarchically for $L = \lceil \log_{B_{\text{merge}}} |\mathcal{P}| \rceil$ levels; at each level ℓ, up to B_{merge} patches are synthesized into one patch:

$$p^{(\ell+1)} = \mathcal{M}\Big(\pi_\theta,\ \mathcal{S}_0,\ \{p_1^{(\ell)}, \ldots, p_{B_{\text{merge}}}^{(\ell)}\}\Big), \tag{2}$$

where $\mathcal{M}$ deduplicates, resolves conflicts, and preserves non-overlapping insights. π_θ itself serves as trajectory generator, analyst, and merge operator, so no external evolution/teacher model is required. The final p^* is translated into diff-style edits and applied with deterministic guardrails: reject edits to missing files, withhold line-range conflicts, and validate the updated skill format.

The merge also performs inductive generalization. Because each patch comes from one trajectory, recurring edits across independent patches are evidence of systematic task properties rather than one-off fixes. $\mathcal{M}$ is therefore instructed to prefer prevalent patterns and discard idiosyncratic edits, producing a compact skill that replaces $\mathcal{S}_0$ and is used directly at inference without retrieval. The merge operator prompt template and an example consolidated patch p^* are given in § I.3.

Trace2Skill supports two modes that cover scenarios with and without expert-written skills: *Skill deepening* starts with a human-written skill, and *Skill creation* starts from a skill generated from LLM parametric knowledge. Both improve the target skill using patches induced from trace analysis.

3 Experiments

3.1 Experimental Setup

Spreadsheet setup. Our main experiments use SpreadsheetBench-Verified (Ma et al., 2024), where agents manipulate `xlsx` files through tool use. We split its 400 samples into 200 evolution problems and 200 held-out test problems, and also report Soft/Hard scores on full SpreadsheetBench (excluding all evolving-set problems) plus OOD transfer to WikiTableQuestions (Pasupat and Liang, 2015) and HiTab (Cheng et al., 2022) converted into spreadsheet format. All spreadsheet results are averaged over three random seeds using each benchmark's official evaluation criteria.

Skill settings. We compare **No Skill**, the Anthropic `xlsx` skill (**Human-Written**), LLM-generated skills using **Parametric** knowledge, and three Trace2Skill variants: **+Error**, **+Success**, and **+Combined**, which consolidate patches from failed trajectories only, from successful trajectories only, and from all trajectories, respectively. *Skill Deepening* starts from Human-Written, while *Skill Creation* starts from Parametric. We evaluate Qwen3.5-122B-A10B and Qwen3.5-35B-A3B as both skill authors and skill users. We do 100% self-evolution:

	Skill User: Qwen3.5-122B-A10B					*Skill User: Qwen3.5-35B-A3B*					
	SpreadsheetBench			*OOD*		*SpreadsheetBench*			*OOD*		
Condition	**Vrf↑**	**Soft↑**	**Hard↑**	**WikiTQ↑**	**HiTab↑**	**Vrf↑**	**Soft↑**	**Hard↑**	**WikiTQ↑**	**HiTab↑**	**Avg↑**
Baseline (absolute scores)											
No Skill	27.67	28.90	17.57	21.50	14.42	19.00	18.00	4.60	13.33	19.12	18.19
Human-Written	48.33	36.30	17.03	74.68	41.31	9.67	13.03	3.37	9.02	5.30	26.93
Parametric	26.17	36.60	17.50	23.73	17.36	20.17	13.70	3.87	20.14	13.94	19.23
Skill Author: Qwen3.5-122B-A10B											
Deepening (Delta from init: Human-Written)											
+Error	+17.50	+10.30	+10.40	+1.62	+2.14	**+27.00**	+9.44	+2.86	+9.26	+9.55	+9.28
+Success	+18.00	+8.60	+8.70	-10.35	-0.95	+19.66	+6.84	+1.33	+12.09	+22.33	+8.15
+Combined	**+21.50**	**+10.87**	**+12.50**	+4.56	+2.97	+21.16	+8.84	+1.80	+6.64	+11.94	+9.65
Creation (Delta from init: Parametric)											
+Error	+22.83	+3.77	+5.87	+7.89	-1.70	+8.66	+9.53	+4.00	+2.06	+9.64	+6.79
+Success	+15.33	-0.93	+4.33	+23.70	+19.89	+12.83	+11.57	+6.13	+30.36	**+28.90**	+16.96
+Combined	+14.00	-0.63	+3.53	+32.32	+23.11	+15.50	**+14.50**	**+7.23**	+29.70	+27.71	+18.62
Skill Author: Qwen3.5-35B-A3B											
Deepening (Delta from init: Human-Written)											
+Error	+16.67	+8.50	+8.14	-6.36	-2.38	+17.33	+9.17	+4.83	+2.71	+8.08	+5.64
+Success	+2.17	+2.73	+3.30	+1.46	+1.70	+12.33	+5.87	+1.23	**+43.23**	+34.70	+12.44
+Combined	+6.67	+3.87	+4.17	+2.65	+2.44	+20.00	+5.77	+2.36	+42.20	+36.46	**+14.04**
Creation (Delta from init: Parametric)											
+Error	+1.00	-7.70	+1.03	**+57.65**	**+28.25**	+3.83	+7.30	+2.66	+12.66	+17.81	+15.22
+Success	+5.33	-4.57	+2.43	+9.09	+2.34	+5.66	+5.80	+2.63	+3.31	+18.94	+5.65
+Combined	+8.33	-5.83	+2.00	+30.82	+16.93	+4.33	+9.73	+4.73	+18.00	+25.22	+13.31

Table 1: Main spreadsheet results across skill-author (evolves the skill) and skill-user (runs it at inference) models. SpreadsheetBench reports Vrf (verified), Soft (problem pass rate), and Hard (task pass rate); WikiTQ and HiTab are out-of-distribution (OOD) table-QA transfer, and $\mathrm{Avg}=(\mathrm{ID}_{\mathrm{avg}}+\mathrm{OOD}_{\mathrm{avg}})/2$ equally weights in-distribution (ID, SpreadsheetBench) and OOD. Reference rows are absolute scores; evolved rows are signed deltas from the Human-Written (Deepening) or Parametric (Creation) baseline. **Bold** marks the largest absolute score per column. Per-seed standard deviations are in Table 7.

the same model generates trajectories, proposes patches, and edits skills. Details of dataset construction, scoring, and model serving are in § A; the external `skill-creator` baseline is in § H.

3.2 Main Results

Table 1 reports all spreadsheet results across skill conditions, author models, user models, and transfer tasks. Baseline rows give absolute scores; evolved rows give signed deltas from the relevant baseline, with Deepening compared to Human-Written and Creation compared to Parametric. We use **Avg** as the primary summary metric because it equally weights ID and OOD performance across both user models, rewarding skills that generalize across models and tasks.

Baselines reveal why both settings matter. Human-Written is a useful handcrafted prior, but it is not reliably portable across model scales; Parametric remains close to No Skill, confirming that parametric knowledge alone does not yield actionable spreadsheet skills (Han et al., 2026). These baselines motivate the two evaluation regimes: Deepening tests whether a strong manual skill can be made more transferable, while Creation tests whether trajectory-grounded distillation can build a useful skill from a weak seed.

Both Deepening and Creation produce generalizable skills. Starting from Human-Written, evolved skills consistently strengthen in-distribution spreadsheet performance and often transfer to other model scales and OOD table tasks; starting from Parametric, Creation can match or exceed Human-Written quality in favorable settings. The gains are broad: they appear across author models, user models, and task families, not only on the model that produced the traces. Notably, 35B Deepening +Combined attains the best absolute Avg, showing that skills authored by a small LLM can also generalize. Across analysts, both +Error

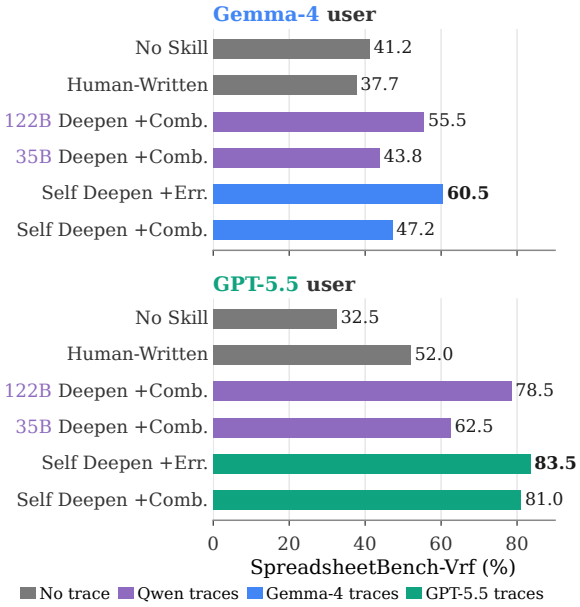

Figure 3: Cross-family generalization on SpreadsheetBench. Gemma-4-31B-it and GPT-5.5-high each (i) evolve a skill from their own traces and (ii) run Qwen3.5-authored skills.

Condition	*Skill User: 122B* D-Test↑	*Skill User: 122B* AIME↑	*Skill User: 35B* D-Test↑	*Skill User: 35B* AIME↑
Reference (absolute scores)				
No Skill	91.0	90.4	89.0	83.3
Skill Author: Qwen3.5-122B-A10B				
Creation (init: No Skill)				
+Error	**+4.0**	**+2.9**	+5.0	+5.0
+Success	+2.0	+0.4	+2.0	+4.6
+Combined	+3.0	-1.2	**+6.0**	**+5.5**
Skill Author: Qwen3.5-35B-A3B				
Creation (init: No Skill)				
+Error	+3.0	+1.3	+4.0	+0.5
+Success	-1.0	-1.2	+0.0	-1.2
+Combined	+1.0	-0.4	+1.0	+0.0

Table 2: Math-reasoning transfer to held-out DAPO and AIME. D-Test is the DAPO-Math-Test pass rate (%) and AIME is avg@8 (%). Layout and presentation follow Table 1.

and +Success help spreadsheet tasks, and +Combined usually gives the largest Avg improvement.

3.3 Model-Family Generalization

We test generalization across model families with Gemma-4-31B-it and GPT-5.5-high in two settings: each model (i) deepens the human-written `xlsx` skill from its own traces via Trace2Skill, and (ii) runs the Qwen3.5-122B/35B +Combined deepened skills. Fig. 3 shows that both models self-improve from their own traces and also benefit from Qwen3.5-authored skills, so Trace2Skill generalizes across families. § B.4 reports additional results and implementation details.

3.4 Math Reasoning

We apply Trace2Skill to math domain to test its domain-agnosticism. As in the spreadsheet setting (§ 3.1), the math agent runs in a ReAct loop with a command-line Python interpreter to write and execute code for each question. We create skills from scratch on 400 DAPO questions (Yu et al., 2025) and evaluate on 100 disjoint held-out DAPO questions (ID) and AIME 2026 (OOD competition mathematics; avg@8 over 30 problems), following the cross-model protocol of § 3.2. Table 2 reports deltas from No Skill. **Trace2Skill works for math domain:** the distilled skills improve both held-out DAPO and OOD AIME rather than overfitting the source distribution. **+Error is the most stable signal**, transferring cleanly between the 122B and 35B users.

3.5 Visual Question Answering

To test multimodal generalization, we apply Trace2Skill to DocVQA (Mathew et al., 2020), where agents answer questions over document images such as forms, tables, invoices, letters, and reports. Again, each agent runs in a ReAct loop with a command-line + Python environment. We use 50 DocVQA examples only for skill evolution and remove them from evaluation; the remaining 5,299 examples form the held-out test set. We report the official ANLS and Accuracy (ANLS ≥ 0.5, %) for both same-model use and cross-model transfer between 122B- and 35B-authored skills. Results in Table 3 show that **all evolved DocVQA skills improve performance over No Skill. +Combined is the strongest setting**, giving the clearest same-model gains and positive cross-model transfer.

4 Analysis

This section analyzes why Trace2Skill works and its broader application. We first isolate its three core design choices under a shared trace pool and execution harness (§ 4.1), then characterize the standard operating procedures it distills (§ 4.2), study how patch value composes (§ 4.3), and test transfer in broader real-world tasks (§ 4.4).

Condition	*Skill User: 122B* ANLS↑	*Skill User: 122B* Acc↑	*Skill User: 35B* ANLS↑	*Skill User: 35B* Acc↑
Reference (absolute scores)				
No Skill	0.6300	70.27	0.6582	73.17
Skill Author: Qwen3.5-122B-A10B				
Creation (init: No Skill)				
+Error	+0.1949	+17.69	+0.1974	+16.64
+Success	+0.1639	+14.89	+0.1668	+14.23
+Combined	**+0.2534**	**+22.25**	+0.2049	+17.22
Skill Author: Qwen3.5-35B-A3B				
Creation (init: No Skill)				
+Error	+0.0884	+6.79	+0.1267	+10.15
+Success	+0.1572	+14.53	+0.2132	+18.27
+Combined	+0.0958	+7.73	**+0.2158**	**+18.83**

Table 3: DocVQA transfer from trajectory-distilled visual-reasoning skills. ANLS is Average Normalized Levenshtein Similarity and Acc is the share of instances with ANLS ≥ 0.5 (%). Layout and presentation follow Table 1.

Condition	*Skill User: 122B* Vrf↑	Soft↑	Hard↑	*Skill User: 35B* Vrf↑	Soft↑	Hard↑	Time↓
Seq-B=4	59.00	40.63	20.63	26.17	22.37	7.47	∼15 min
Seq-B=1	61.83	44.40	25.40	26.00	**23.83**	**10.57**	∼60 min
Parallel (ours)	**65.83**	**46.60**	**27.43**	**27.00**	22.20	8.20	**∼3 min**

Table 4: Parallel consolidation versus sequential editing on SpreadsheetBench (same trace pool, +Error). Parallel outperforms Seq in effectiveness and efficiency. Table 8 and Table 10 show similar trend in math/VQA.

4.1 Core Design Comparisons

Trace2Skill rests on three design choices, which we isolate here by comparing each against its natural alternative under the same trace pool and execution harness: parallel many-to-one consolidation versus sequential editing (speed without quality loss), a single consolidated skill versus experience memory retrieval (reuse without test-time retrieval), and agentic versus single-call error analysis (more accurate root-cause patches).

Parallel Consolidation. Online skill evolution typically edits the skill as trajectory batches arrive, so later analyses depend on earlier edits. We isolate the effect of our parallel many-to-one consolidation by comparing against two sequential baselines: Seq-B=1, which updates after every trajectory, and Seq-B=4, which updates after every four trajectories. All conditions use error analysts only and initialize from the Human-Written skill. Table 4 reports the resulting quality and wall-clock tradeoff.

Parallel consolidation is better on all 122B SpreadsheetBench metrics and on 35B Vrf; the exception is that Seq-B=1 is modestly higher on 35B Soft and Hard. This small quality tradeoff comes with a large efficiency gap: parallel produces the skill in about 3 min, compared with 15 min for Seq-B=4 and 60 min for Seq-B=1. Math reasoning and DocVQA show the same broad trend that parallel consolidation preserves or improves quality while being much faster; § B gives the latency analysis and similar comparisons on math/VQA.

Setting	User	Vrf↑	Soft↑	Hard↑
ReasoningBank (Ouyang et al., 2026)	122B	56.00	40.10	21.30
	35B	20.50	17.30	4.97
Human-Written	122B	**69.83**	**47.17**	**29.53**
+Combined (ours)	35B	**29.67**	**18.80**	**5.73**

Table 5: Trace2Skill outperforms ReasoningBank-style retrieval on SpreadsheetBench (same trajectory pool). Table 9 and Table 11 show similar trend in math/VQA.

Holistic Skill vs. Retrieval. Past-experience retrieval is a common paradigm for reusing agent trajectories: experience-memory systems store reflections, memories, or procedural lessons from previous executions and retrieve relevant items at inference time (Shinn et al., 2023; Wang et al., 2023a; Ouyang et al., 2026; Fang et al., 2026; Wang et al., 2024b; Liu et al., 2025). We instantiate this paradigm with ReasoningBank (Ouyang et al., 2026): following its original setting, we store lessons from both success and failure trajectories and retrieve the top-1 memory with Qwen3-Embedding-8B. We compare it against +Combined, which uses the same trajectory pool. Results on same-model Deepening are shown in Table 5. +Combined is consistently better than ReasoningBank. This supports the advantage of consolidating trajectory evidence into a compact skill rather than retrieving isolated memories at test time. We exclude OOD evaluations, as retrieval is not applicable for WikiTQ and HiTab whose queries are too semantically distant from the queries of SpreadsheetBench.

Agentic Error Analysis. Related work derives transferable lessons or skills from error trajectories via a single non-interactive LLM call (Ouyang et al., 2026; Xia et al., 2026a; Yang et al., 2026; Jiang et al., 2026). We ablate this design with **+Error LLM**, where one LLM call reads each failed trajectory and proposes a patch without inspecting artifacts, querying ground truth, or validating

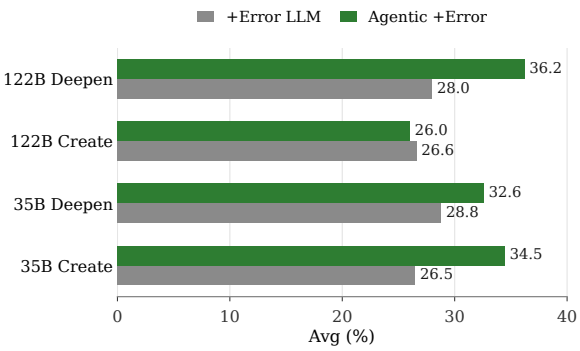

Figure 4: Agentic +Error versus single-call +Error LLM on Avg, the balance of ID and OOD performance (same as Avg in Table 1). The agentic analyst inspects artifacts and validates fixes, whereas +Error LLM reads only the execution log.

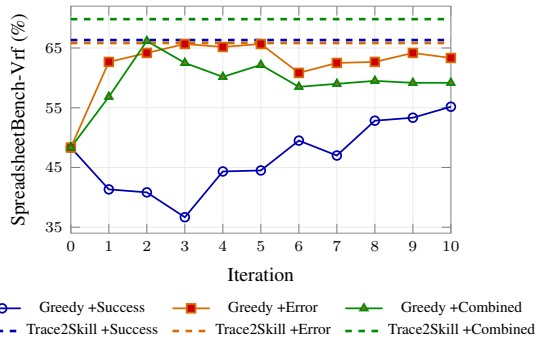

Figure 5: Greedy selective patch aggregation on SpreadsheetBench-Vrf (pass rate, %). Each curve adds one validation-selected patch per iteration (x-axis) from the Error, Success, or Combined pool; the flat line is full Trace2Skill aggregation of all patches.

fixes; Fig. 4 compares it with our agentic error analyst. Agentic +Error achieves higher Avg than +Error LLM in most settings, showing that interactive diagnosis is more useful than log-only patch generation; § D reports and discusses the full per-dataset results in Table 12. The qualitative analysis in § C.1 shows the mechanism: artifact access and fix validation help the agentic analyst identify root causes more precisely.

Apples-to-apples vs. head-to-head comparison. The above three comparisons are apples-to-apples, isolating each core design without the data, model, harness, and engineering confounders that complicate whole-system evaluation. As a complementary view, § G reports a direct head-to-head against three concurrent systems (XSkill (Jiang et al., 2026), EvoSkill (Alzubi et al., 2026), and SkillGen (Ma et al., 2026)) under a shared open model and benchmark, where Trace2Skill still shows its advantage.

4.2 SoPs Learned

Many-to-one merging keeps the operations that recur across trajectories and turns them into standard operating procedures (SoPs), rather than collecting one-off tricks. Across the 323 map patches from the 122B Deepening +Combined run, four learned SoPs dominate, each cited by between 16% and 55% of patches (a single patch can cite more than one SoP), including: recalculating and reading back formulas after writes, verifying target cells before submitting, and deleting rows in a corruption-safe order. Task-specific quirks are routed to on-demand `references/` files, keeping the SKILL.md focused on broadly reusable procedures (Details in § C.3).

4.3 Selective Patch Aggregation

Trace2Skill aggregates all learned patches into the target skill, avoiding the cost of per-patch validation. If patches vary in quality, can selecting a subset do better? We test two selectors against full aggregation: **greedy top-1**, which at each iteration adds the locally best patch according to validation on evolving set, and **Bayesian optimization (BO)**, which searches binary patch-inclusion vectors and reuses validation scores from previously tried subsets to bias later proposals toward promising combinations. Both use the 200-task evolving set (§ 3.2) and a 32-task validation set from evolving set; § E gives the full algorithmic details.

Greedy rises quickly but plateaus below full aggregation. Fig. 5 shows greedy +Error and +Combined rising for a few iterations and then plateauing, with +Success dipping and only partially recovering; every greedy curve stays below full Trace2Skill aggregation. Two mechanisms drive the plateau (detailed in § C.2). (1) *patch-irrelevant regression*: the patches flip some target tasks from wrong to correct, but their side effects also flip previously-correct tasks from correct to wrong, so net accuracy stalls. (2) *semantic overlap*: recurring errors lead the pipeline to propose patches that repeatedly target the same behaviors (e.g., formula recalculation, validation checklists), so a new patch largely restates existing guidance and adds little marginal gain. Patch value is therefore combinatorial rather than a sum of singletons, which makes optimizing the combined effect of patches necessary.

BO is useful but computationally heavy. BO searches patch subsets directly, so it can model

Skill	Evolve	Vrf	Soft	Hard	WikiTQ	HiTab
No Selection (+Error)	68.00	65.83	46.60	27.43	76.30	**43.45**
BO Selection (+Error)	**72.83**	**69.83**	**47.27**	**28.90**	**77.51**	43.38

Table 6: BO-selected +Error patches versus applying all of them (No Selection), in the 122B Deepening spreadsheet setting. Evolve is the accuracy on evolving-set.

complementarity and interference that greedy misses. Table 6 shows that, compared with applying all +Error patches, BO improves most selected metrics, especially the SpreadsheetBench columns. The cost is validation: each candidate subset must be materialized as a skill and run on the validation set before it can inform the posterior, and larger patch universes increase this cost quickly. We therefore view BO as useful when the target distribution and validation budget are known, rather than as a replacement for the default full aggregation in Trace2Skill.

4.4 Broader Application

We test broader application of Trace2Skill in PDF extraction, PPTX editing, and DOCX editing. These settings use Anthropic's official `pdf`, `pptx`, and `docx` skills as starting points (Anthropic, 2026b) and retain the same execution style as the spreadsheet experiments: agents operate over files with command-line tools and are scored by task-specific local verifiers.

Across the three domains, evolution on source traces improve performance on separate held-out targets. For PDF, VRDU traces transfer to VAREX, raising pass rate from 76.9% to 85.3% (Wang et al., 2023b; Barzelay et al., 2026). For PPTX, TSBench traces transfer to a deck-disjoint TSBench OOD split, improving 72.5% to 88.8% (Jung et al., 2026). For DOCX, evolving on synthetic tasks that cover generic DOCX operations transfers to the OfficeBench DOCX subset, improving 79.7% to 87.5% (Wang et al., 2024a). § F provides setup and implementation details for each domain.

5 Related Work

Agent skills. Agent skills package task procedures, domain knowledge, and operational guardrails into loadable artifacts (Anthropic, 2026d), but ecosystems and benchmarks show the abstraction is delicate: focused, well-matched skills improve performance, while broad, stale, or mismatched ones can distract or even hurt the agent (Zhou et al., 2026b; Li et al., 2026b; Han et al., 2026; Li, 2026; Li et al., 2026a; Liang et al., 2026). Trace2Skill keeps this view of skills as portable SoPs, but asks a narrower question: how to compress broad execution evidence into guidance that stays useful across models and task distributions.

Experience memory for agent self-evolution. Another line improves agents by storing execution experience for reuse, from verbal reflection and accumulated behaviors to retrieval, procedural, workflow, or replay memories queried at test time (Shinn et al., 2023; Wang et al., 2023a; Ouyang et al., 2026; Fang et al., 2026; Wang et al., 2024b; Qian et al., 2024; Nottingham et al., 2024; Liu et al., 2025). These methods share our premise that experience contains reusable structure but retain an external memory or retrieval module, whereas Trace2Skill distills many local observations into one static skill directory. It favors inductive compression over nearest-neighbor reuse and avoiding test-time dependence on retrieval quality.

Skill and policy evolution. The closest concurrent line evolves skills from agent interaction or guided refinement (Zheng et al., 2025; Yang et al., 2026; Jiang et al., 2026; Alzubi et al., 2026; Zhou et al., 2026a; Ma et al., 2026; Anthropic, 2026a), while a broader family co-evolves policies, skills, or agent loops online through skill-augmented reinforcement learning, intrinsic skill evolution, or meta-learning (Xia et al., 2026a; Li et al., 2026c; Xia et al., 2026b). Trace2Skill asks a complementary question: whether traces collected with one model can be compressed into a static skill directory that directly benefits other models and tasks. Instead of maintaining a test-time memory, retrieval index, or updated policy, it consolidates trajectory-local evidence into reusable SoPs that can be loaded unchanged. Thus, our focus is not adaptive reuse within a particular agent loop, but portability of the learned artifact across deployment settings.

6 Conclusion

We introduced Trace2Skill, a framework that distills agent execution traces into a portable skill: parallel analyst sub-agents propose targeted patches from disjoint trajectory batches, and a consolidation step merges them at once into one declarative skill directory. Skills distilled from a single model's traces transfer across model scales, families, and OOD tasks. Trace2Skill also exhibits strong applicability in various domains.

Limitations

Trace2Skill applies all consolidated patches by default; selecting a higher-quality subset with Bayesian optimization can help (§ 4.3), but it is costly. A reliable selection signal needs a large, representative validation set and BO must materialize and score a new skill for every candidate subset, so cost grows quickly with both the validation set and the patch universe. Ideally, using the whole evolving set for validation would better reflect patch quality than our sampled validation with only 32 questions, but it would be much more computationally expensive. We therefore leave a more thorough exploration of combinatorial patch selection to future work.

Ethics Statement

We use publicly available datasets, which have no data privacy issues. All artifacts we use are under licenses allowing research usage. Human annotation in qualitative analyses were conducted by the authors of this paper. We do not identify any other ethical risks associated with this study. AI coding tools (Codex/Claude Code) are used to assist with coding. We confirm that all such coding is under careful human supervision. Unit tests are implemented to avoid hacking or unaligned behavior. We will also open-source the code for full reproducibility. We also leverage ChatGPT for grammar check and fix, fully supervised by human authors.

References

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

Anthropic. 2026a. How to create a skill with claude through conversation. Claude Tutorials.

Anthropic. 2026b. PDF, DOCX, and PPTX document skills. GitHub repository. Accessed: 2026-05-01; document-skill subfolders: `pdf`, `docx`, and `pptx`.

Anthropic. 2026c. skill-creator. GitHub repository.

Anthropic. 2026d. What are skills? Claude Help Center. Access Date: 2026-03-22.

Udi Barzelay, Ophir Azulai, Inbar Shapira, Idan Friedman, Foad Abo Dahood, Madison Lee, and Abraham Daniels. 2026. VAREX: A benchmark for multi-modal structured extraction from documents. *Preprint*, arXiv:2603.15118.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2026. Memp: Exploring agent procedural memory. *Preprint*, arXiv:2508.06433.

Google. 2026. Gemma 4: Byte for byte, the most capable open models. `https://blog.google/innovation-and-ai/technology/developers-tools/gemma-4/`. Accessed: 2026-05-18.

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

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. 2026. Xskill: Continual learning from experience and skills in multimodal agents. *Preprint*, arXiv:2603.12056.

Kyudan Jung, Hojun Cho, Jooyeol Yun, Soyoung Yang, Jaehyeok Jang, and Jaegul Choo. 2026. Talk to your slides: High-efficiency slide editing via language-driven structured data manipulation. *Preprint*, arXiv:2505.11604.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Hao Li, Chunjiang Mu, Jianhao Chen, Siyue Ren, Zhiyao Cui, Yiqun Zhang, Lei Bai, and Shuyue Hu. 2026a. Organizing, orchestrating, and benchmarking agent skills at ecosystem scale. *Preprint*, arXiv:2603.02176.

Jiachun Li, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Mirage: Evaluating and explaining inductive reasoning process in language models. *Preprint*, arXiv:2410.09542.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, and 22 others. 2026b. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *Preprint*, arXiv:2602.12670.

Xiaoxiao Li. 2026. When single-agent with skills replace multi-agent systems and when they fail. *Preprint*, arXiv:2601.04748.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. 2026c. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning. *Preprint*, arXiv:2603.16060.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Xin Xu, Tongtong Wu, Kun Wang, Yang Liu, Zhen Bi, Jungang Lou, Yuchen Eleanor Jiang, Hangcheng Zhu, and 30 others. 2026. Skillnet: Create, evaluate, and connect ai skills. *Preprint*, arXiv:2603.04448.

Brian S. Lin, Jiaxin Yuan, Zihan Zhou, Shouli Wang, Shuo Wang, Cunliang Kong, Qi Shi, Yuxuan Li, Liner Yang, Zhiyuan Liu, and Maosong Sun. 2025. On llm-based scientific inductive reasoning beyond equations. *Preprint*, arXiv:2509.16226.

Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. 2025. Contextual experience replay for self-improvement of language agents. *Preprint*, arXiv:2506.06698.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. 2026. Skillgen: Verified inference-time agent skill synthesis. *Preprint*, arXiv:2605.10999.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. *Preprint*, arXiv:2406.14991.

Minesh Mathew, Dimosthenis Karatzas, R Manmatha, and CV Jawahar. 2020. Docvqa: A dataset for vqa on document images. corr abs/2007.00398 (2020). *arXiv preprint arXiv:2007.00398*.

Kolby Nottingham, Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Sameer Singh, Peter Clark, and Roy Fox. 2024. Skill set optimization: Reinforcing language model behavior via transferable skills. *Preprint*, arXiv:2402.03244.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. 2026. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Preprint*, arXiv:1508.00305.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *Preprint*, arXiv:2401.13996.

Qwen. 2026. Qwen3.5: Accelerating productivity with native multimodal agents.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

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

Zilong Wang, Yichao Zhou, Wei Wei, Chen-Yu Lee, and Sandeep Tata. 2023b. VRDU: A benchmark for visually-rich document understanding. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *Preprint*, arXiv:2409.07429.

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

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026b. Metaclaw: Just talk – an agent that meta-learns and evolves in the wild. *Preprint*, arXiv:2603.17187.

Chenfei Xiong, Jingwei Ni, Yu Fan, Vilém Zouhar, Donya Rooein, Lorena Calvo-Bartolomé, Alexander Hoyle, Zhijing Jin, Mrinmaya Sachan, Markus Leippold, Dirk Hovy, Mennatallah El-Assady, and Elliott Ash. 2025. Co-DETECT: Collaborative discovery of edge cases in text classification. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 354–364, Suzhou, China. Association for Computational Linguistics.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

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

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. 2026a. Memento-skills: Let agents design agents. *Preprint*, arXiv:2603.18743.

Yingli Zhou, Shu Wang, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. 2026b. A comprehensive survey on agent skills: Taxonomy, techniques, and applications. *Preprint*, arXiv:2605.07358.

A Experimental Details

Random seeds and compute. Unless otherwise noted, all reported results are averaged over three random seeds: 41, 42, and 43. Experiments are run on nodes with 8 NVIDIA A100 GPUs. We spent roughly a total of 20,000 GPU hours for all experiments and exploration.

Spreadsheet data and scoring. SpreadsheetBench Verified is split into 200 evolution problems and 200 held-out test problems; no held-out sample is used during skill evolution. We additionally report Soft, the sub-problem pass rate, and Hard, where all sub-problems must pass, on full SpreadsheetBench, from which we remove every evolving-set problem so that no problem seen during skill evolution is scored at test time. For OOD transfer, WikiTableQuestions is converted into spreadsheet tasks over compositional Wikipedia tables, and HiTab is converted into spreadsheet tasks that require hierarchical indexing plus implicit calculation and semantic reasoning. The evaluation sets contain 200 SpreadsheetBench-Verified held-out examples, 2,529 full SpreadsheetBench examples for Soft-/Hard (all evolving-set problems removed), 2,810 WikiTableQuestions examples, and 1,585 HiTab examples.

Baseline skills. Human-Written is Anthropic's official `xlsx` skill, and Parametric is an `xlsx-basic` seed generated by prompting Qwen3.5-122B-A10B from parametric knowledge alone, with no trajectory grounding.

Model serving and evolution protocol. The two main author/user models are Qwen3.5-122B-A10B and Qwen3.5-35B-A3B. We use the Hugging Face checkpoints Qwen/Qwen3.5-122B-A10B and Qwen/Qwen3.5-35B-A3B. Both are instruct/think hybrid MoE models: we use instruct mode for multi-turn ReAct-style agents and thinking mode for single-call steps such as hierarchical merging, success analysis, and patch conversion. Models are served with vLLM (Kwon et al., 2023); for reasoning-mode calls, we follow Qwen's official recommended decoding settings. Stage 1 generates one trajectory per problem; Stage 2 runs 128 subagents in parallel with merge batch size 32. We do not impose a separate tool-call budget.

B Additional Analysis Results

B.1 Latency Analysis

With $W{=}128$ workers and $N{\approx}70$ error lessons, all analysts execute in a single parallel round. With merge batch size $B_{\text{merge}}{=}32$, the hierarchical merge adds only $\lceil \log_{B_{\text{merge}}} N \rceil {\approx} 2$ further sequential rounds, one per merge layer, yielding ≈ 3 sequential LLM-call rounds in total. The sequential baselines require N and $\lceil N/B \rceil$ rounds respectively, since each skill edit depends on the preceding one. In practice this translates to 3 min for parallel consolidation, 60 min for Seq-$B{=}1$ ($20\times$ slower), and 15 min for Seq-$B{=}4$ ($5\times$ slower), with the gap scaling linearly in N. All times are wall-clock skill-generation times measured on the same 8-GPU A100 node configuration.

B.2 Math Reasoning

Tables 8 and 9 extend the main analysis ablations to math reasoning. We use the +Error setting for these math ablations because it is the strongest math setting in Table 2. Parallel consolidation is best or tied for best on all reported math metrics while also having the lowest skill-generation wall time.

B.3 DocVQA

Tables 10 and 11 extend the main DocVQA results to sequential and retrieval-memory comparisons. We use the +Combined setting for these

	Skill User: Qwen3.5-122B-A10B					*Skill User: Qwen3.5-35B-A3B*					
	SpreadsheetBench			*OOD*		*SpreadsheetBench*			*OOD*		
Condition	**Vrf↑**	**Soft↑**	**Hard↑**	**WikiTQ↑**	**HiTab↑**	**Vrf↑**	**Soft↑**	**Hard↑**	**WikiTQ↑**	**HiTab↑**	**Avg↑**
Baseline (absolute scores)											
No Skill	3.75	0.90	0.70	0.81	0.67	1.80	2.77	1.25	1.27	5.28	0.42
Human-Written	8.10	1.39	0.40	0.65	0.71	2.93	0.57	0.40	0.36	2.68	0.46
Parametric	0.58	1.04	0.75	6.52	8.93	1.44	1.56	0.50	13.32	18.31	3.28
Skill Author: Qwen3.5-122B-A10B											
Deepening (Delta from init: Human-Written)											
+Error	4.82	1.10	0.78	0.78	0.60	6.76	3.36	2.06	2.62	4.16	1.02
+Success	12.53	0.75	1.35	1.68	0.75	3.69	3.78	2.23	1.43	4.01	1.43
+Combined	9.53	1.68	0.61	1.22	0.85	5.92	2.92	1.73	2.53	2.74	1.57
Creation (Delta from init: Parametric)											
+Error	1.53	1.10	0.55	7.07	6.98	3.62	1.63	1.22	13.30	18.93	3.24
+Success	1.89	0.91	1.45	5.25	9.60	4.86	2.39	1.76	14.66	19.38	3.00
+Combined	3.04	0.90	1.10	6.07	9.69	3.04	1.48	1.13	13.52	19.72	2.99
Skill Author: Qwen3.5-35B-A3B											
Deepening (init: Human-Written)											
+Error	11.50	1.31	2.31	1.40	0.63	5.11	3.00	1.12	0.67	1.46	1.10
+Success	15.81	4.63	2.42	1.69	0.84	0.76	2.47	1.16	1.18	2.90	1.32
+Combined	9.00	2.14	1.33	1.94	0.85	0.50	3.62	2.54	3.27	2.71	0.88
Creation (init: Parametric)											
+Error	2.18	1.04	0.58	6.96	8.84	1.76	1.77	1.08	14.60	17.83	3.36
+Success	1.61	1.00	0.90	6.94	8.64	4.51	1.10	1.19	14.24	19.02	3.21
+Combined	2.75	1.02	1.28	6.37	8.19	5.51	1.62	1.91	10.98	19.64	3.16

Table 7: Standard deviations across seeds 41, 42, and 43 for the main spreadsheet results in Table 1. Reference rows are standard deviations of absolute scores; evolved rows are standard deviations of the paired deltas defined in Table 1. The summary metric Avg is stable (standard deviation ≤ 3.4 in every row), and the mean gains in Table 1 stay large relative to the per-cell spread, so the comparisons are robust. The higher variance in individual columns (Vrf and the OOD WikiTQ/HiTab transfer) is expected rather than a sign of instability: each score aggregates long agentic rollouts that often exceed 30 turns of environment interaction, so single-benchmark run-to-run stochasticity accumulates while averaging out in Avg. For conciseness, we omit additional standard-deviation tables for the paper's other results, whose deviations are similarly small: std ranges for spreadsheet tasks are similar to this table. Std ranges of DAPO-Test, AIME'26, and DocVQA ANLS are 1.1-2.4, 1.8-3.4, and 0.034-0.139 correspondingly.

Condition	*Runner: 122B*		*Runner: 35B*		
	D-Test↑	**AIME↑**	**D-Test↑**	**AIME↑**	**Time↓**
No Skill	92.0	90.4	89.0	83.3	0
Seq-B=4	93.0	**91.7**	89.0	68.3	3.8 min
Seq-B=1	94.0	85.8	90.0	81.7	25.9 min
Parallel	**95.0**	**91.7**	**94.0**	**88.3**	**2.0 min**

Table 8: Parallel consolidation versus sequential editing on math reasoning (same trace budget). D-Test/AIME are as in Table 2 (122B and 35B users).

Condition	*Runner: 122B*		*Runner: 35B*	
	D-Test↑	**AIME↑**	**D-Test↑**	**AIME↑**
No Skill	92.0	90.4	89.0	83.3
ReasoningBank (error-only)	94.0	90.8	91.0	80.8
ReasoningBank (combined)	94.0	90.4	91.0	80.4
Trace2Skill +Error (ours)	**95.0**	**91.7**	**94.0**	**88.3**

Table 9: Trace2Skill outperforms ReasoningBank retrieval on math reasoning. D-Test/AIME are as in Table 2.

DocVQA ablations because it is the strongest DocVQA setting in Table 3. Table 10 isolates the 122B combined-protocol batch ablation, where all non-baseline rows use the same 25 failure and 25 success trajectories and differ only in sequential batch size versus parallel consolidation. These 50 DocVQA examples are used only for evolution and are excluded from evaluation; all DocVQA scores are computed on the remaining 5,299 held-out examples. The efficiency gap between parallel and

Setting	*Runner: 122B*		
	ANLS↑	**Acc↑**	**Time↓**
No Skill	0.6300	70.27	0
Seq-B=4	0.8674	90.80	5.2 min
Seq-B=1	0.8694	91.05	35.9 min
Parallel	**0.8833**	**92.52**	**4.8 min**

Table 10: Parallel consolidation versus sequential editing on DocVQA using the +Combined trace pool. ANLS/Acc are as in Table 3.

Setting	*Runner: 122B*		*Runner: 35B*	
	ANLS↑	**Acc↑**	**ANLS↑**	**Acc↑**
No Skill	0.6300	70.27	0.6582	73.17
ReasoningBank (combined)	0.8668	90.90	0.8568	89.62
Trace2Skill +Combined (ours)	**0.8833**	**92.52**	**0.8740**	**92.00**

Table 11: Trace2Skill outperforms ReasoningBank retrieval on DocVQA. ANLS/Acc are as in Table 3.

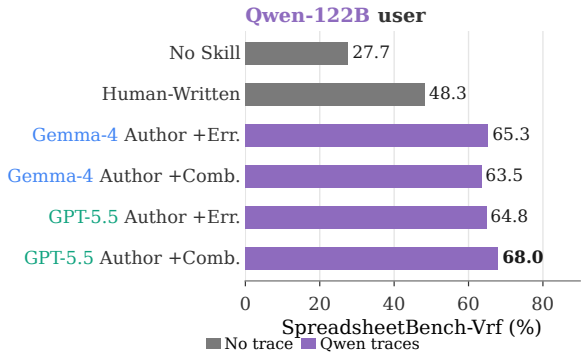

Figure 6: Cross-model trace induction on SpreadsheetBench-Vrf (pass rate, %). Gemma-4 and GPT-5.5-high author skills from Qwen traces, which are then run by the Qwen3.5-122B user.

sequential paradigms will grow with the number of traces as analyzed in § B.1.

B.4 Cross-Model Trace Induction

Fig. 6 visualizes this cross-model trace-induction setting. Using Qwen traces, Gemma-4-31B-it and GPT-5.5-high-authored Deepening skills improve the Qwen3.5-122B user over both No Skill (27.67% Vrf) and Human-Written (48.33% Vrf). The strongest setting is GPT-5.5-high-authored Deepening +Combined at 68.00% Vrf, showing that cross-model trace evidence can still induce meaningful portable skills.

Implementation details. We serve Gemma-4-31B-it with vLLM (Kwon et al., 2023) using the official recommended generation configuration from its Hugging Face model card (google/gemma-4-31B-it); reasoning is disabled for the ReAct agent runs. GPT-5.5-high is accessed through the official OpenAI API with only the reasoning effort set to high and all other settings left at their defaults.

C Qualitative Analyses

C.1 Agentic vs. LLM Error Analysis

We qualitatively audit 33 shared error cases analyzed by both the agentic error analyst $\mathcal{A}^-$ and the single-call +Error LLM baseline. The two pipelines reach strong agreement on only 4 cases (12.1%), while 18 cases (54.5%) show clear disagreement about the root cause or the appropriate skill patch. The main difference is access to evidence: $\mathcal{A}^-$ can inspect input/output artifacts, compare the submitted answer with ground truth, and validate candidate fixes, whereas +Error LLM must infer the failure from the execution log alone.

This limitation makes the LLM-only analyzer prone to log-level false positives. Among cases where parse-error messages appear, +Error LLM attributes the parse error as the primary root cause in 57% of cases, compared with 14% for the agentic analyst. In one representative trajectory, +Error LLM hallucinated three distinct failure causes even though artifact evaluation showed the output was already correct. By contrast, the agentic loop can reject such explanations after checking the generated file and rerunning the relevant validation steps.

These qualitative differences explain why the agentic patches transfer more reliably in § 4.1. Rather than encoding surface symptoms from logs, $\mathcal{A}^-$ anchors patches to verified failure mechanisms: wrong target ranges, stale formula values, corrupted workbook structure, type conversion, or missing read-back verification. The resulting edits are more likely to become domain-general guardrails and less likely to degrade ID, cross-model, or OOD settings when applied as a portable skill.

C.2 Patch Selection Qualitative Analysis

Patch-irrelevant regression. Later iterative patches do change agent behavior on some tasks, flipping them from wrong to correct as the patch intends. However, the same edits also change behavior that is irrelevant to the targeted failure, flipping other, previously-correct tasks from correct to wrong. The flips are largely off-source: across the ten-iteration greedy +Combined path (seed 41), the

selected patches' source tasks have zero overlap with the evaluation split, so the previously-correct tasks a step breaks are ones the selected patch never targeted: side effects rather than targeted regressions. Accordingly, every materialized step records substantial flips in both directions (for example, 57 fail-to-pass against 21 pass-to-fail at one step, and 26 against 46 at another), so each addition simultaneously fixes and breaks tasks. We closely inspect those flips, finding that the majority of fail-to-pass flips are patch-relevant while pass-to-fail flips are usually irrelevant Because each greedy step both fixes and breaks tasks, the net accuracy plateaus instead of rising. A locally best single patch therefore does not compose into a global gain: patch value depends on how additions interact, so optimizing the combinatorial effect of patches rather than their isolated marginal contributions is necessary.

Semantic overlap. The same failure modes recur across iterations, so the pipeline repeatedly proposes semantically overlapping patches that target the same spreadsheet behaviors—formula recalculation, workbook-structure preservation, reference consolidation, and validation checklists. The greedy path makes this concrete: across its ten iterations the selector keeps returning to a few themes—recalculation and verification (iterations 1 and 6), structure/header/sheet handling (iterations 2 and 9), and formatting/type/date handling (iterations 4, 5, 7, and 8). A newly selected patch from an already-covered theme then largely restates guidance the skill already encodes and brings little marginal gain.

C.3 Learned SoP Details

We inspect the 323 map patches produced by the 122B Deepening +Combined run. The four most prevalent SoPs are cited by 55.1%, 54.8%, 42.7%, and 16.4% of the 323 patches, respectively[2]; these shares sum above 100% because a single patch can cite multiple themes.

Formula recalculation and write-back verification (55.1% of patches). Run `recalc.py` after every formula write and reopen with `data_only=True` to confirm evaluation; skipping this step leaves cells stale and is the single most common error mode in the run.

Tool selection: `openpyxl` over `pandas.to_excel()` (54.8% of patches). Use `pandas` for read/transform logic and `openpyxl` for write-back; copy the input file to the output path first to preserve all structural anchors. `pandas.to_excel()` silently destroys formula relationships and named ranges.

Explicit read-back verification (42.7% of patches). After writing, reopen the output file and confirm every target cell holds the expected value before submitting; error trajectories that fail characteristically omit this check.

Structural-edit safety (16.4% of patches). Delete rows in descending order to prevent index-shift corruption; copy the input workbook before editing to preserve formatting and formulas. Error trajectories document both failure modes; success trajectories confirm the protective workflow.

Niche quirks are routed to `references/`. Low-support observations are not discarded but routed into 13 supplementary reference files rather than the main `SKILL.md`. For example, cell color extraction and FIFO vs. LIFO mismatch under special business logic are placed in on-demand references. This mirrors established skill-design practice: procedural guidance flows from general to case-specific, with the main document encoding universal workflow rules and `references/` serving as an on-demand look-up layer for infrequent edge cases. Trace2Skill recovers this hierarchy automatically from trajectory evidence rather than requiring manual curation.

Additional moderate-support SoPs. The following SoPs appear in the same run with moderate support (about 3.1%–4.6% of patches) and are also encoded in the evolved skill.

Target-range and answer-position validation (4.6% of patches). Before writing, verify the exact target sheet name, cell range, and `answer_position` field from the task metadata. Misreading these fields — writing to the wrong sheet or an off-by-one range — causes silent failures that produce no error message but score zero.

Datatype and datetime preservation (4.6% of patches). Write dates and numeric values as native Python types, not strings. Both `pandas` date parsing and `openpyxl` cell assignment can silently stringify datetime values; inspect each column's dtype before writing and use `openpyxl`'s native datetime assignment.

Workbook structure exploration before editing (success-dominant, ∼4.0% of patches). List

[2] Trace2Skill merging process tracks the original patches where the merged patch (SoPs) are induced from.

	Skill User: Qwen3.5-122B-A10B					*Skill User: Qwen3.5-35B-A3B*					
	SpreadsheetBench			*OOD*		*SpreadsheetBench*			*OOD*		
Condition	**Vrf**	**Soft**	**Hard**	**WikiTQ**	**HiTab**	**Vrf**	**Soft**	**Hard**	**WikiTQ**	**HiTab**	**Avg**
Skill Author: Qwen3.5-122B-A10B											
Deepening											
+Error (ours)	65.83	**46.60**	**27.43**	**76.30**	**43.45**	**36.67**	**22.47**	**6.23**	**18.28**	**14.85**	**36.21**
+Error LLM	**67.00**	43.93	25.23	39.81	36.68	25.00	22.43	6.23	11.24	9.74	28.00
Creation											
+Error (ours)	**49.00**	**40.37**	**23.37**	31.62	15.66	**28.83**	**23.23**	**7.87**	22.20	23.58	26.02
+Error LLM	27.17	27.73	16.20	**47.26**	**30.26**	19.83	17.60	4.70	**23.30**	**36.53**	**26.60**
Skill Author: Qwen3.5-35B-A3B											
Deepening											
+Error (ours)	**65.00**	**44.80**	**25.17**	68.32	38.93	27.00	**22.20**	8.20	**11.73**	**13.38**	**32.58**
+Error LLM	37.83	22.93	12.83	**77.05**	**42.05**	**30.50**	20.17	**8.73**	9.95	12.73	28.80
Creation											
+Error (ours)	**27.17**	**28.90**	**18.53**	**81.38**	**45.61**	**24.00**	**21.00**	**6.53**	**32.80**	31.75	**34.45**
+Error LLM	22.00	27.67	16.60	54.61	37.93	23.50	16.87	4.93	11.24	**33.75**	26.49

Table 12: Agentic error analysis (+Error) versus single-call +Error LLM, across Deepening and Creation and all 122B/35B author–user pairs. Vrf/Soft/Hard, the OOD splits (WikiTQ/HiTab), and Avg are as in Table 1. **Bold** marks the better of the two methods per column.

all sheets, inspect row/column layout, and verify header positions before any write. This pre-edit exploration prevents wrong-sheet and wrong-range failures and accounts for a substantial share of the 151 success-leaning patches in the run.

D Full Agentic Error Analysis Results

Table 12 expands the main-text Avg comparison into per-dataset results for both skill authors, both evolution modes, and both skill-user models. The main pattern is that agentic error analysis is more consistent: +Error (ours) has the higher Avg in three of the four author–mode blocks and wins most SpreadsheetBench columns. The exception is 122B-authored Creation, where +Error LLM is slightly higher on Avg because it does better on the OOD table benchmarks; even there, the agentic analyst remains stronger on the in-distribution SpreadsheetBench metrics. Overall, the full table supports the main claim that artifact inspection and fix validation produce error patches that transfer more reliably than log-only analysis.

E Selective Patch Aggregation Details

The main experiments apply all learned patches after hierarchical consolidation; selective aggregation instead asks whether some subset of patches yields a better skill. We reuse the formalization of § 2.1: $\mathcal{S}$ is a skill, π_θ the fixed agent, and $\mathcal{P}$ the pool of trajectory-level patches from Stage 2 (§ 2.3), with $c(p)$ the number of source trajectories supporting patch $p \in \mathcal{P}$. We write $\mathcal{S} \oplus \mathcal{P}'$ for the skill obtained by applying a patch subset $\mathcal{P}' \subseteq \mathcal{P}$ to $\mathcal{S}$ through the Stage 3 diff application (§ 2.4), and $\mathrm{acc}_V(\mathcal{S}) \triangleq \mathcal{P}(\mathcal{S}; \pi_\theta, V)$ for its validation pass rate on a 32-task validation set V. Both selectors below search for a subset whose materialized skill maximizes acc_V.

Greedy top-1 selection. At iteration t, we evaluate the current skill $\mathcal{S}_t$, collect the newly proposed patches $\mathcal{P}_t$, and form a candidate set $\mathcal{C}_t$ of the five highest-coverage patches with $c(p) \geq 5$. For each $p \in \mathcal{C}_t$, we estimate its contribution by an add-one and remove-one intervention:

$$\begin{aligned}\Delta^+(p) &= \mathrm{acc}_V(\mathcal{S}_t \oplus p) - \mathrm{acc}_V(\mathcal{S}_t),\\ \Delta^-(p) &= \mathrm{acc}_V(\mathcal{S}_t \oplus \mathcal{P}_t)\\ &\quad - \mathrm{acc}_V\big(\mathcal{S}_t \oplus (\mathcal{P}_t \setminus \{p\})\big).\end{aligned}$$

Patches with $\Delta^+(p) < 0$ or $\Delta^-(p) < 0$ are discarded; the remainder are ranked by $r(p) = \Delta^+(p) + \Delta^-(p)$, with coverage as the tie-breaker, and the next skill is $\mathcal{S}_{t+1} = \mathcal{S}_t \oplus p_t^\star$ for the top-ranked patch $p_t^\star$. § 4.3 reports the SpreadsheetBench-Vrf results.

Bayesian optimization over patch subsets. BO fixes a patch universe $\mathcal{P} = \{p_j\}_{j=1}^m$ from the Stage-2 pool (§ 2.3), keeping the top $m \leq 15$ patches

by coverage. It searches binary inclusion vectors $x \in \{0,1\}^m$, writing $\mathcal{P}_x = \{p_j : x_j = 1\}$ for the selected subset and starting from the initial skill $\mathcal{S}_0$. The objective is

$$f(x) = \mathrm{acc}_V(\mathcal{S}_0 \oplus \mathcal{P}_x) - \lambda\|x\|_0, \quad \lambda = 0.$$

The initial design evaluates the empty subset, all singletons, five random mixed subsets, and the full subset. After each batch, evaluated subsets are split into a good set G containing the top $\gamma = 0.2$ fraction by $f(x)$ and a bad set B containing the rest. With Beta smoothing $\alpha = 1$, the per-patch Bernoulli inclusion rates are

$$\mu_j^G = \frac{\sum_{x \in G} x_j + \alpha}{|G| + 2\alpha}, \qquad \mu_j^B = \frac{\sum_{x \in B} x_j + \alpha}{|B| + 2\alpha}.$$

Candidate subsets are sampled from $\phi_j = (1 - \rho)\mu_j^G + \rho/2$ with $\rho = 0.1$ and ranked by the TPE acquisition

$$A(x) = \sum_{j=1}^{m} \big[\log q(x_j; \mu_j^G) - \log q(x_j; \mu_j^B)\big],$$
$$q(b; \mu) = \mu^b (1-\mu)^{1-b}.$$

We evaluate the top eight candidates per round from a pool of 500, stop after four rounds or three rounds without improvement, and return the best observed subset. § 4.3 compares the BO-selected +Error skill against applying all +Error patches.

F Broader Application Details

We extend Trace2Skill to three broader document-agent settings: PDF extraction, PPTX editing, and DOCX editing. All settings use Anthropic's official document skills as the underlying skill family (Anthropic, 2026b). Source traces are used only for skill evolution, while held-out target tasks are used only for evaluation. Each evaluation uses the task's exact local verifier and reports the same task-level pass-rate metric as the main experiments. Table 13 summarizes the transfer setting used for each modality in the main text; "Base" denotes the immediate skill before the listed source traces are applied.

PDF extraction. The PDF study adapts VRDU and VAREX into local PDF-to-JSON tasks (Wang et al., 2023b; Barzelay et al., 2026). We use VRDU Registration Forms as the source corpus because it provides a coherent real collection of visually rich form-extraction traces with recurring field and layout conventions. We evaluate on VAREX Flat as a separate held-out structured-extraction benchmark, testing whether those induced form-extraction procedures transfer across datasets rather than to more examples from the same corpus. The reported PDF score uses the strongest validated VRDU Registration source split; other Registration splits are also positive but yield smaller gains.

PPTX editing. The PPTX study uses TSBench presentation-editing tasks from Talk-to-Your-Slides (Jung et al., 2026). The PPTX result evolves from TSBench training traces on one set of decks and evaluates on a deck-disjoint TSBench OOD split of held-out decks, testing transfer across presentation files rather than more edits from the same decks.

DOCX editing. The DOCX study uses OfficeBench as the held-out target because it contains realistic office-automation tasks with verifiable Word/DOCX outputs (Wang et al., 2024a). We reserve all 64 convertible OfficeBench DOCX subtasks only for evaluation at `max_turns=100`. Because the remaining real, verifiable DOCX agent data is too small after reserving this target set, we use generated office-document traces as the source distribution. These traces cover reusable operations such as editing templates, preserving document structure, appending content across files, and producing required sidecar files.

G Head-to-Head Comparison with Concurrent Skill-Evolution Systems

This appendix is the head-to-head complement to the apples-to-apples comparisons in § 4.1: it pits Trace2Skill against three full concurrent skill-evolution systems—XSkill (Jiang et al., 2026), EvoSkill (Alzubi et al., 2026), and SkillGen (Ma et al., 2026)—rather than isolating one design choice at a time. We keep it out of the main text for three reasons. **Attribution:** a whole-system score conflates a design *idea* with its base model, harness, and engineering, so it cannot say which factor drove a difference; § 4.1 instead varies a single design choice under a shared trace pool, model, and harness, which is what licenses our causal claims. **Scope:** these systems pursue different goals and were built around specific (often proprietary) base models, bespoke harnesses, and different task domains, whereas we deliberately study open models

Domain	Source traces	Held-out evaluation	Base	Trace2Skill	Gain
PDF	VRDU Registration Forms	VAREX Flat validation	76.9%	85.3%	+8.4 pp
PPTX	TSBench training traces	TSBench deck-disjoint OOD	72.5%	88.8%	+16.3 pp
DOCX	Generated document-operation tasks	OfficeBench DOCX holdout	79.7%	87.5%	+7.8 pp

Table 13: Broader document-agent transfer. Each row evolves an official Anthropic document skill from source traces and evaluates on a held-out target domain. Base and Trace2Skill are target pass rates (%); Gain is their difference in percentage points (pp).

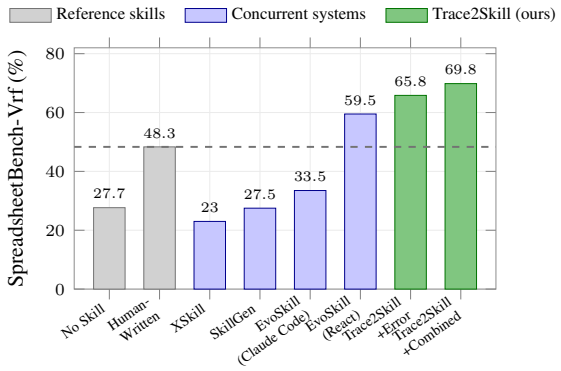

Figure 7: Head-to-head comparison on SpreadsheetBench-Verified (Vrf, pass rate %) with all systems using the same open base model (Qwen3.5-122B-A10B) and the held-out 200:400 test slice. Bars cover the No Skill floor, the Human-Written starting skill (dashed line), the three reproduced concurrent systems, and Trace2Skill Deepening. EvoSkill's two bars are the same faithful reproduction differing only in agent harness (Claude Code, following original paper, vs. React, matching our setting).

that self-evolve; transplanting them onto one benchmark shows behavior *in our setting*, not a ceiling on their design, so we read the numbers conservatively.

We reproduce these pipelines following their official codebase adapted to one shared protocol: the same open base model Qwen3.5-122B-A10B served with vLLM, evolution on the SpreadsheetBench-Verified 0:200 slice, and evaluation on the held-out 200:400 slice scored by official instance accuracy (our Vrf metric). Fig. 7 reports the result against the No Skill floor, the Human-Written skill that several systems also start from, and Trace2Skill Deepening +Error/+Combined. Trace2Skill attains the highest Vrf (65.83 +Error, 69.83 +Combined), above all three systems; the per-system settings and the deviations from each native pipeline are detailed below.

XSkill. XSkill is a non-parametric memory method that accumulates task-level procedure skills and action-level experiences, then retrieves, adapts, and injects them at inference; it was designed for multimodal tool-using agents. We collect 200 trajectories on the 0:200 slice (same evolving set as Trace2Skill), merge their per-trajectory skills into one 865-word global `SKILL.md` (batched LLM merge), and distill 20 curated action-level experiences. At test time, for each instance we retrieve the top-3 experiences by lexical overlap, have the LLM adapt the global skill to the task, inject it into the system prompt, and run the agent for up to 100 turns. This reproduction reaches 23.0 Vrf. We replaced the lexical retriever with Qwen3-Embedding-0.6B. However, this achieves a slightly worse performance at 20.0.

EvoSkill. EvoSkill is a self-evolving loop—base agent, failure proposer, skill generator, validation evaluator, and a bounded frontier—designed for the Claude Code harness with Opus 4.5. We mine failures on 0:160, validate candidates on 160:200, and evaluate the selected program on 200:400 (our SpreadsheetBench Vrf test subset), with a frontier of size 3; each iteration proposes and generates a replacement `xlsx/SKILL.md` and scores it on the validation slice, with Qwen3.5-122B-A10B served by vLLM. We run two harnesses: a React-style local harness matching our setting reaches 59.5 Vrf (5 iterations), while Claude Code accessed through an Anthropic-compatible proxy (closest to the native harness) reaches 33.5 (8 workers, early-stopped near iteration 11).

SkillGen. SkillGen induces a compact skill from contrastive failure/success analysis behind a paired verification gate before held-out evaluation. We collect baseline trajectories on 0:200, cluster failure and success summaries with Qwen3-Embedding-0.6B (k-means) to synthesize contrastive observations, generate a compact `SKILL.md`, and verify it on 0:200 behind a net-gain acceptance gate (we use $\mathrm{min_net_gain_abs}=1$; the upstream default is ≥ 3) for up to 8 refinement rounds, evaluating the accepted skill on 200:400. From a 52.5 baseline on

User/Trace source	Deepening			Creation		
	Base	skill-creator	Trace2Skill +Error	Base	skill-creator	Trace2Skill +Error
122B/122B	48.33	27.33	65.83	26.17	26.67	49.00
122B/35B	48.33	19.50	65.00	26.17	18.33	27.17
35B/35B	9.67	18.50	27.00	20.17	17.67	24.00
35B/122B	9.67	28.00	36.67	20.17	27.33	28.83
Average	**29.00**	**23.33**	**48.63**	**23.17**	**22.50**	**32.25**

Table 14: SpreadsheetBench-Verified (Vrf) comparison with Anthropic's `skill-creator` baseline (pass rate, %). For each skill-user/trace-source pair, `skill-creator` is compared with the corresponding Base skill and Trace2Skill +Error under Deepening and Creation; the final row averages each column.

the training slice, round 1 nets 0 (rejected), round 2 nets -4 (rejected), and round 3 nets $+1$ (accepted), giving 27.5 Vrf on the held-out slice. This improvement on training set but regression on test set might be attributed to the distribution shift: the training slice mixes 125 sheet-level and 75 cell-level tasks while the test slice is cell-only.

These head-to-head comparisons complement, and do not replace, the confounder-free apples-to-apples comparisons in § 4.1, on which our claims rest.

H Skill-Creator Baseline

For the external Vrf-only baseline, we used Anthropic's official `skill-creator` skill for skill drafting and improvement (Anthropic, 2026c) through Claude Code with Opus 4.6 medium. Table 14 reports SpreadsheetBench-Verified Vrf scores for the same skill-user/trace-source pairings used in the main spreadsheet evaluation, alongside the corresponding Table 1 baselines and Trace2Skill +Error results. The resulting skills do not improve the corresponding base skill on average in either Deepening or Creation, while Trace2Skill +Error remains consistently stronger.

The prompts used for this baseline were:

Skill Creator Baseline Prompt: Deepening

```
I have an agent running spreadsheet jobs
using an xlsx skill. The agent's traces are
provided in /path/to/traces/, where error and
success traces are annotated with
*_FAILURE.md and *_SUCCESS.md. Your job is to
deepen the xlsx (/path/to/xlsx) to improve
the agent's future performance when using the
new skill. You should first induce the common
and generalizable failure and success
patterns from the traces and patch the xlsx
skill using the skill-creator skill.
```

Skill Creator Baseline Prompt: Creation

```
I have an agent running spreadsheet jobs. The
agent's traces are provided in
/path/to/traces/, where error and success
traces are annotated with *_FAILURE.md and
*_SUCCESS.md. Your job is to create a
spreadsheet skill to improve the agent's
future performance equipped with the skill.
You should first induce the common and
generalizable failure and success patterns
from the traces and then create the skill
using the skill-creator skill.
```

I Prompt Templates and Intermediate Outputs

This appendix reproduces the key prompt templates used in each pipeline stage and illustrates representative intermediate outputs to make the pipeline fully transparent and reproducible.

I.1 Stage 1: Agent System Prompt Template

The agent π_θ operates under the following system prompt during trajectory collection. The skill $\mathcal{S}_0$ is prepended to the user context at inference time. Note that this differs from the standard skill-loading process, where the agent initially has access only to skill descriptions. We simplify this by preloading the SKILL.md content into the system prompt because Trace2Skill focuses on improving a fixed target skill that is known relative to the task. Therefore, there is no need for the standard skill-selection step. Importantly, the Trace2Skill skill-using agent still needs to procedurally discover resources referenced by the preloaded SKILL.md (e.g., resources and scripts), which are not preloaded.

Stage 1 — Agent System Prompt (abbreviated)

Role: You are an expert role (e.g., spreadsheet analysis) agent.
Skill context: `[Contents of $\mathcal{S}_0$ inserted here]`
Task: Describing tasks and input files.
Tools available: Describing tools and environment. E.g., `bash` (shell execution) for ReAct with file system access.
Format: ReAct-style interaction — alternate between reasoning traces and tool calls until the task is complete.

I.2 Stage 2: Analyst Prompt Templates and Example Patches

In Stage 2, the patch proposing agents first draw error and success memory items similar to (Ouyang et al., 2026), which are generalizable trajectory-level knowledge that might be helpful for future task executions. Next, the agents read the original skill directory and then propose a patch to encode the memory items into the skill.

I.2.1 Error Analyst Prompt ($\mathcal{A}^-$)

Error Analyst System Prompt (abbreviated)

Role: You are an expert failure-analysis agent for {domain} tasks.

Mission: Given an agent's execution artifacts (logs + produced files) and the ground-truth solution, diagnose *why the agent failed*, identify *causal failure reasons*, and *validate* your diagnosis by implementing a minimal fix that makes the agent output match the ground truth. Your analysis must be **systematic**, **evidence-driven**, and **reproducible**. **Do not guess** when you can verify.

Required Workflow (MANDATORY):
1. Understand the task and failure surface — identify exactly what is wrong in the output.
2. Trace the failure to agent behavior — locate the decision or code step that produced the mismatch.
3. Validate the root cause with a minimal fix — write fixed output and re-evaluate against the ground truth.
4. Re-evaluate — if still failing, return to steps 1–3 and revise your diagnosis.

Output: Produce (1) *Failure Cause Items* — systematic, causal reasons grounded in observable agent behavior; (2) *Failure Memory Items* ($\leq$3) — generalizable insights the agent should remember to avoid similar failures.

I.2.2 Success Analyst Prompt ($\mathcal{A}^+$)

Success Analyst System Prompt (abbreviated)

Role: You are an expert in success pattern analysis for AI agent systems.

Mission: Given a successful agent trajectory, identify *generalizable* behavior patterns that contributed to the correct answer.

Requirements:
Broad Coverage — every effective behavior in the trajectory must be captured by a pattern.
Frequency Awareness — patterns covering more instances should be listed first; rare behaviors should be absorbed into the nearest broader pattern.
Generalization — each pattern must describe a general mechanism, not a single task-specific detail.

Output: A compact set of *Success Memory Items* with title, description, and concrete examples of the effective behaviors observed.

I.2.3 Example Trajectory-Level Patch (Error Analyst Output)

Each analyst produces a structured patch for a single trajectory. The following example is from an error analyst ($\mathcal{A}^-$) processing one failure trajectory where the agent deleted rows outside the specified range. The patch proposes adding a new `SKILL.md` section on row-manipulation safety; this patch, along with 52 others citing related structural-edit failures, is later consolidated into the "Structural-edit safety" SoP (§ 4.2).

Example Stage 2 Trajectory-Level Patch (error analyst, one trajectory)

Analyst reasoning: Single error record reveals a clear failure pattern: the agent deletes rows outside the specified range and fails to distinguish partially-empty data rows from completely empty padding rows. The skill lacks any guidance on row/column manipulation or range boundary enforcement.

```
--- a/SKILL.md
+++ b/SKILL.md
@@ -55,0 +55,26 @@ ## Formula Verification Checklist

+## Row and Column Operations
+
+### Range Boundary Enforcement
+- When a task specifies a cell range (e.g., F14:M83),
      operations
+  MUST stay within those exact boundaries
+- Parse range format: start_col + start_row : end_col +
      end_row
+- Never apply operations to rows/columns outside the
      range
+
+### Row Deletion Logic
+Before deleting any row, verify BOTH conditions:
+1. Target columns are empty (as specified by task)
+2. Row falls within the explicit deletion range
+
+**Do NOT delete** completely empty rows across ALL
      columns unless
+explicitly instructed -- these often serve as structural
       padding.
+
+### Post-Operation Validation
+After row/column operations:
+- [ ] Count remaining rows matches expected count
+- [ ] No header/footer/padding rows outside range were
      affected
+- [ ] Data integrity is maintained in adjacent columns
+- [ ] Range boundaries were respected (no off-by-one
      errors)
```

I.3 Stage 3: Merge Operator Prompt and Example Consolidated Patch

I.3.1 Merge Operator Prompt ($\mathcal{M}$)

Merge Operator System Prompt (full)

You are a skill edit coordinator. You receive multiple independently-proposed patches that each suggest changes to a skill folder. Your job is to merge them into one coherent, non-redundant patch.

Guidelines:
1. **Deduplicate**: When multiple patches propose the same or very similar edits, keep the best version (most specific, best worded).
2. **Resolve conflicts**: If patches propose contradictory edits to the same section, choose the one with stronger justification or synthesize both into a better edit.
3. **Preserve unique insights**: Different patches address different failures — include all unique, non-redundant edits.
4. **Maintain conciseness**: The merged patch should have $\leq$ the sum of unique edits across all input patches. Remove redundancy.
5. **Ensure independence**: Edits in the merged patch MUST be line-level independent — no two edits may target overlapping lines or the same passage of text, even across different operations.
6. **Atomic create/link pairs**: A `create` operation for `references/*.md` and the `SKILL.md` edit that inserts a link to it are an inseparable pair — keep both or drop both.

Prevalent pattern bias: When multiple patches independently propose similar edits addressing the same class of failure or success pattern, treat this recurrence as evidence of a *systematic* property of the task. Preserve such prevalent edits with higher priority and express them as general principles rather than instance-specific fixes.

I.3.2 Example Final Consolidated Patch p^* (After Full Merge Hierarchy)

The following excerpt shows the reasoning and representative edits from the final consolidated patch p^* produced after four levels of hierarchical merging over 323 individual trajectory patches on SpreadsheetBench-Verified.

Example Stage 3 Final Patch Output (excerpt)

```
{
  "reasoning": "Merged 3 patches addressing mixed failure/
      success
    evidence. Key consolidation decisions: (1)
      Synthesized recalc.py
    workflow from all patches using the most prominent
      CRITICAL WARNING
    placement, CSV fallback validation, and a
      verification loop;
    (2) Consolidated library selection guidance into a
      comprehensive
    decision tree; (3) Combined row deletion guidance
      emphasizing
    bottom-up/right-to-left deletion order from all
      patches;
    (4) Merged formula validation checklists without
      redundancy.",
  "edits": [
    {
      "file": "SKILL.md",
      "op": "insert_after",
      "target_section": "# Requirements for Outputs",
      "content": "## Important Automation Guidelines\n\n
        **Prefer Python over VBA for Automation**: When
      tasks request
        VBA macros or spreadsheet automation, implement
      the logic in
        Python using openpyxl/pandas instead. This
      provides better
        error handling, easier debugging, cross-platform
      compatibility,
        and avoids macro security issues."
    },
    {
      "file": "SKILL.md",
      "op": "insert_after",
      "target_section": "## Important Requirements",
      "content": "### When to Compute Directly in Python\
      n
        If LibreOffice/recalc.py cannot evaluate certain
      formula
        patterns (complex array formulas, advanced INDEX/
      MATCH
        combinations), compute the logic directly in
      Python and write
        results as values. Use this fallback when formula
       returns errors
        after recalculation despite correct syntax."
    }
  ],
  "changelog_entries": [
    "Added automation guidelines preferring Python over
      VBA",
    "Consolidated recalc.py warnings into a single
      CRITICAL WARNING"
  ]
}
```

I.3.3 Example Translated Diff: Final Consolidated Patch p^*

The JSON patch p^* produced by Stage 3 is translated programmatically into a unified diff and applied directly to the skill directory. The following excerpt shows two hunks from the final applied diff on SpreadsheetBench-Verified, corresponding to the two most prevalent SoPs: formula recalculation enforcement (55.1% of patches) and tool-selection guidance (54.8% of patches).

Example Stage 3 Applied Diff (excerpt from final p^*)

```
--- a/SKILL.md
+++ b/SKILL.md
@@ -126,3 +261,18 @@ ## Common Workflow

 ## Common Workflow

+### CRITICAL WARNING: Formula Recalculation Is Mandatory
+
+**If you write ANY formulas to an Excel file using
      openpyxl,
+you MUST run recalc.py before considering the task
      complete.**
+
+Formulas written via openpyxl exist only as text strings
       until
+recalculated. Without running recalc.py:
+- Cells return None/empty when read with data_only=True
+- Evaluation fails even if formulas are syntactically
      correct
+- The output file is incomplete
+
+This is non-negotiable. Do not proceed to verification
      or
+delivery until recalc.py confirms success.
```

```
+
@@ -138,3 +285,9 @@ ### Working with openpyxl

+### Tool Selection Warning
+
+**CRITICAL**: When modifying spreadsheets that contain
      existing
+formulas you need to preserve:
+- Use openpyxl (load_workbook() then save()) -- formulas
       remain
+  as strings
+- Avoid pandas (to_excel()) -- silently converts
      formulas to
+  static values permanently
```