EVOMOSKILL: EVOLVING A MIXTURE-OF-SKILLS FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Enabling agent skills to autonomously refine reusable knowledge is a central challenge in adapting large language model (LLM) agents to real-world applications. Recent work treats skills as trainable external parameters, following the iterative optimization paradigm of deep learning. However, these methods typically optimize a single skill across tasks, so revisions that benefit one task group can disrupt others. A skill library can limit this interference, but raises questions across skill creation, task routing, failure attribution, and evaluation. We introduce EvoMoSkill, a mixture-of-skills framework that jointly optimizes a share skill, expert skills, and a router. Each optimization step applies Credit-Assigned Gradient Descent (CreAGD) in five stages: (1) **Forward execution and credit assignment** attributes execution failures to skill text or router scope; (2) **Textual and routing gradient descent** edits skills, creates experts, and revises routing; (3) **Scope-conditioned paired evaluation** compares candidates with the original system on tasks they would serve; (4) **History-calibrated step acceptance** checks gains against past outcomes and accepts revisions after validation-set checks; and (5) **Optimizer memory accumulation** uses rejected drafts and feedback to improve later updates. Experiments with five LLMs on five benchmarks demonstrate state-of-the-art performance, substantially outperforming human-authored skills, LLM-generated skills, and baselines that optimize only a single skill. These results show that evolving mixture-of-skills yields more adaptable agents.

1 INTRODUCTION

Recent advances in large language models (LLMs) are expanding the scope of autonomous problem solving, enabling agents to coordinate multistep planning, tool execution, and environmental feedback within increasingly complex workflows (Wang et al., 2023; Li et al., 2026b). Agent skills consolidate reusable procedures into external documents, enabling adaptation without model parameter updates (Zhang et al., 2025; Jiang et al., 2026b). However, manual authoring and one-shot generation alone cannot incorporate lessons from subsequent executions. Enabling autonomous skill self-improvement as task demands evolve therefore remains a central challenge.

A promising approach treats skills as trainable external parameters, drawing on the iterative optimization paradigm of deep learning. Batch execution and scoring provide the analogue of a forward pass, while an optimizer derives textual update directions from trajectory errors to guide skill edits (Liu et al., 2026). Evaluation selects revisions for subsequent batches (Alzubi et al., 2026). These methods typically train a single skill shared across tasks (Yang et al., 2026a; Lin et al., 2026). Every revision therefore changes the guidance supplied to all tasks, even when their procedural requirements differ. A repair for one group can consequently disrupt behavior that already works elsewhere. A library of separately selectable and editable skills can localize such adaptations, but raises questions beyond the single-skill training loop: *When should new skills be introduced, and which tasks should use them? How should failures be attributed to skills or routing decisions, and what constitutes an improvement to the library as a whole?*

To this end, we propose EvoMoSkill, a mixture-of-skills framework that jointly optimizes a share skill, expert skills, and their router (Figure 1). Training tasks are organized into families of related problems, providing a common basis for revision and evaluation. Each optimization step applies Credit-Assigned Gradient Descent (CreAGD) in five stages. (1) **Forward execution and credit**

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.

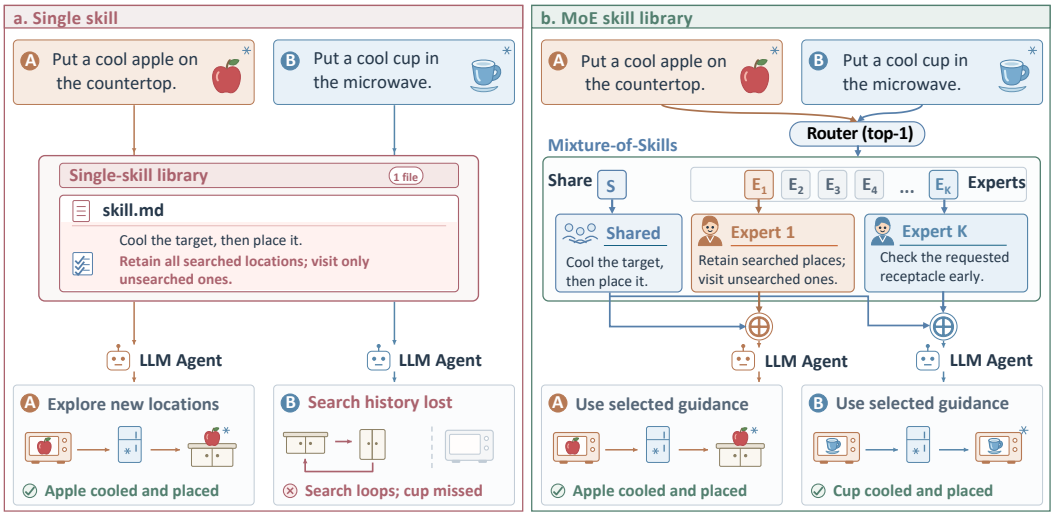

Figure 1: Different tasks can require different skill guidance. (a) A single search rule helps task A explore and place the apple, but task B loses its search history, revisits locations, and misses the cup. (b) The router combines the share skill with distinct expert skills: task A retains searched locations, while task B checks the requested receptacle early. Both tasks succeed.

assignment maps failures to responsible skills or router scopes. (2) **Textual and routing gradient descent** proposes bounded edits, new experts, and scope changes. (3) **Scope-conditioned paired evaluation** compares candidate skills with the prior system on tasks they would serve. (4) **History-calibrated step acceptance** evaluates gains across repeated runs against historical performance, then validates promising revisions on validation tasks. (5) **Optimizer memory accumulation** uses rejected drafts and feedback to guide later revisions. Accepted experts join the library and improve through subsequent feedback. Persistently declining experts retire, making room for new candidates.

Across five diverse agent benchmarks, EvoMoSkill improves average performance by 14.9–23.8 percentage points over no skill for five student models. Ablation studies further attribute these gains to individual components and support the overall design of EvoMoSkill.

Our contributions are threefold:

- We formulate skill evolution as joint optimization of skill content and applicability. EvoMoSkill evolves a share skill, expert skills tailored to task families, and their routing scopes, adapting both the agent's procedures and where each procedure applies.
- We introduce Credit-Assigned Gradient Descent (CreAGD), which attributes execution errors to skills or routing, generates multiple targeted edits, and accepts skill updates only after deliberative validation of gains and cross-task regressions.
- Across five benchmarks and five students, EvoMoSkill outperforms the strongest baseline by 4.0–6.8 points on average. Ablations confirm the value of routing and guarded updates.

2 Related Work

Agent Skills as Reusable Knowledge. An agent skill is a reusable package of procedural knowledge, typically organized as a `SKILL.md` document with optional scripts and references. It guides task planning, tool use, and result verification (Jiang et al., 2026b; Zhang et al., 2025; Destefanis et al., 2026). Human authors translate domain expertise into instructions agents can follow (Destefanis, 2026). To accelerate authoring, automated skill generation systems translate requirements into skills, as in `Skill Creator` (Anthropic, 2026b), or construct them from documents and scientific resources (Si et al., 2026; Shen et al., 2026). With this design, systems consolidate experience into libraries, retrieving, composing, and selectively loading skills (Liang et al., 2026; Li et al., 2026a; Zheng et al., 2026; Gao et al., 2026) for document processing, software development,

scientific analysis, and embodied interaction (Li et al., 2026b; Han et al., 2026; Ju et al., 2026; Wang et al., 2023). However, manual authoring and one-shot generation alone cannot accumulate lessons from subsequent executions (Ni et al., 2026; Yang et al., 2026a).

Evolution of Agentic Skills. Skill evolution continually revises reusable procedures through execution experience (Ding et al., 2026; Zhong et al., 2026). Feedback determines the direction of evolution. *Execution signals* use task scores, runtime errors, and verification results to localize failures (Wang et al., 2023; Ma et al., 2026; Tian et al., 2026). *Trajectory comparisons* contrast successful and failed runs or align teacher demonstrations with student failures to isolate missing behaviors (Chen et al., 2026; Shi et al., 2026a). *User and model judgments* draw on dialogue preferences, critiques, and simulated interactions to expose knowledge and procedural gaps (Yang et al., 2026b; Zhang et al., 2026a; Shinn et al., 2023; Madaan et al., 2023; Yan et al., 2026). Turning feedback into lasting gains requires deciding how to update skills, reuse them, and verify improvements in task performance. *Model and skill coevolution* uses task rewards to train models for skill selection, execution, or curation, while new rollouts drive skill creation and refinement (Xia et al., 2026; Tu et al., 2026; Shi et al., 2026b; Ouyang et al., 2026; Zhang et al., 2026b). *Skill-library evolution* distills reusable procedures from trajectories, consolidates redundant knowledge, and revises stored skills (Zhao et al., 2024; Wang et al., 2024; Ni et al., 2026; Jiang et al., 2026a; Wang et al., 2026), with retrieval and routing governing reuse (Zhou et al., 2026). Candidate search evaluates which edits to retain (Alzubi et al., 2026; Yuksekgonul et al., 2025; Agrawal et al., 2026). Within this direction, *stepwise skill optimization* repeatedly executes training tasks, diagnoses shared failures, and revises skills across batches or epochs (Liu et al., 2026; Yang et al., 2026a; Lin et al., 2026). Analogous to supervised training, successive data-driven updates refine an external skill, with edit budgets, validation checks, or regression constraints controlling revisions. This approach accumulates experience across tasks and supports measurable improvements without retraining the executor, producing reusable domain skills. Yet single-skill optimizers such as SkillOpt and SkillBoost apply one skill throughout a domain, where heterogeneous task families may require incompatible procedures (Li et al., 2026b; Han et al., 2026). EvoMoSkill extends stepwise optimization to a mixture-of-skills library, jointly evolving the share skill, expert skills, and routing scopes.

3 METHOD

In this section, we introduce EvoMoSkill, a skill optimization framework that jointly optimizes a mixture-of-skills library and router from execution feedback. As shown in Figure 2, EvoMoSkill (1) organizes training batches by task family, (2) uses Credit-Assigned Gradient Descent (CreAGD) to assign failure feedback, propose skill and routing revisions, and evaluate skill updates, and (3) refines the share skill and manages expert skill lifecycles across optimization steps.

3.1 PROBLEM FORMULATION

EvoMoSkill extends skill training inspired by deep learning (Yang et al., 2026a; Lin et al., 2026) to jointly optimize a library $\mathcal{L} = (s_{\text{sh}}, \{s_j\}_{j \in I})$ and router g, with a frozen executor M. The share skill s_{sh} provides general guidance, while expert skills s_j supply specialized procedures, with $|I| \leq K_{\max}$. The LLM router matches task descriptions to expert scopes and returns $g(x; \mathcal{L}) \in I \cup \{\varnothing\}$, where $s_\varnothing$ denotes empty context. For task x, M samples trajectory ξ from the task and skill context, and Eval returns a binary signal y indicating whether the task was completed successfully:

$$\xi \sim M(\cdot \mid x \oplus s_{\text{sh}} \oplus s_{g(x;\mathcal{L})}), \qquad y = \text{Eval}(x, \xi) \in \{0, 1\}. \tag{1}$$

Here, $\oplus$ denotes context concatenation. Given disjoint splits $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$, our objective is to jointly learn the library and router that maximize mean validation success:

$$\begin{aligned} (\mathcal{L}^\star, g^\star) &\in \arg \max_{(\mathcal{L},g) \in \mathcal{P}(\mathcal{D}_{\text{train}})} J_{\mathcal{D}_{\text{val}}}(\mathcal{L}, g), \\ J_{\mathcal{D}}(\mathcal{L}, g) &= \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \mathbb{E}[y \mid x, \mathcal{L}, g]. \end{aligned} \tag{2}$$

Here, $\mathcal{P}(\mathcal{D}_{\text{train}})$ contains library and router pairs obtained during training. $J_{\mathcal{D}}$ averages task success over sampled trajectories. We report $J_{\mathcal{D}_{\text{test}}}(\mathcal{L}^\star, g^\star)$ for the frozen pair on test tasks.

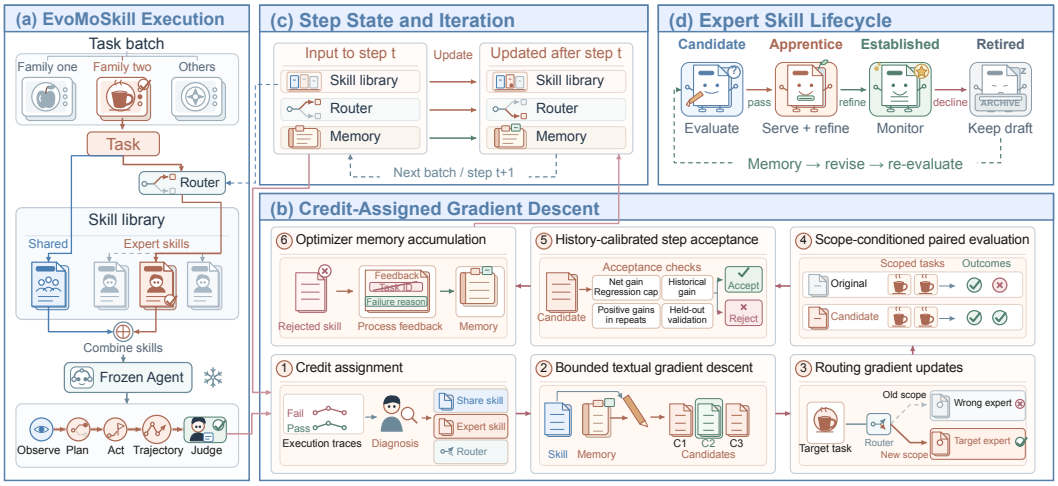

Figure 2: Overview of EvoMoSkill. Tasks use the share skill and at most one routed expert skill. CreAGD assigns failure feedback, proposes skill revisions, and selects them through repeated paired evaluation and validation. Routing errors trigger scope repair. Memory guides future proposals, and successive updates govern expert skill admission, refinement, and retirement.

3.2 Training Data and Batches

We partition $\mathcal{D}_{\text{train}}$ into F task families $\{\mathcal{D}_f\}_{f=1}^{F}$ using task-type annotations or, when unavailable, semantic clustering of task descriptions. We merge related families to keep $F \leq K_{\max}$ and reserve one expert skill per family. Each epoch comprises multiple optimization steps. At step t, we sample a batch $\mathcal{D}_{\text{batch},t}$ across all families, denoting its family-f tasks by $\mathcal{D}_{\text{batch},f,t}$. Tasks execute with the previous library $\mathcal{L}_{t-1}$ and router g_{t-1}. Family labels also partition validation tasks into $\mathcal{D}_{\text{val},f}$ for skill evaluation and library selection. The test set $\mathcal{D}_{\text{test}}$ remains unavailable during training and library selection and is used only for final evaluation.

3.3 Credit-Assigned Gradient Descent

Credit-Assigned Gradient Descent (CreAGD) assigns textual gradients to responsible skills or the router at each step. An LLM optimizer uses the assigned gradients and stored feedback to propose revisions, which undergo paired evaluation before acceptance.

Forward execution and credit assignment. The frozen executor M runs each $x_i \in \mathcal{D}_{\text{batch},t}$ for $R \geq 2$ repetitions, producing trajectories $\{\xi_{i,r}\}_{r=1}^{R}$ and binary outcomes $\{y_{i,r}\}_{r=1}^{R}$.

Table 1: Failure diagnoses and update targets.

Diagnosis	Evidence from execution	Update target
Coverage	No dedicated expert skill covers the family.	New expert skill
Misguide	Following a skill rule contributes to failure.	Skill content
Missing	The skill omits an action needed for the task.	Skill content
Routing	The selected expert skill does not fit the task.	Router scope
Capability	Correct actions fail or the cause is uncertain.	Defer revision

A more capable teacher LLM compares failed trajectories from M with successful demonstrations on the same tasks and examines the applicable skill rules (Shi et al., 2026a). Table 1 maps the resulting diagnoses to updates. Missing or misleading instructions are assigned to the share skill when failures span families, or to an expert skill when confined to one family. A family without coverage calls for a new expert, while selection of an unsuitable expert calls for a router-scope revision. Intermittent failures can motivate reliability edits. If appropriate actions still fail or the cause is unclear, the teacher defers revision and reassesses the case when new trajectories become available. Each actionable diagnosis is assigned to an editable component $u \in \mathcal{U}_t$. The value $u = \text{sh}$ denotes share-skill text, and $u = j \in I$ denotes the text of existing expert s_j. A proposed expert receives a fresh index $u = |I| + 1$, while $u = \text{route}$ denotes router scope. Diagnoses assigned to the same component form its textual gradient:

$$G_{u,t} = \text{Aggregate}(\{d_{i,t} : a(d_{i,t}) = u\}) . \tag{3}$$

Here $d_{i,t}$ is the diagnostic evidence for task x_i, and a assigns it to the responsible skill text or router scope. Deferred cases receive no update target.

Bounded textual gradient descent. For each skill-text target $u \in \mathcal{U}_t \setminus \{\text{route}\}$, the gradient identifies failures to correct and successful behavior to retain. The optimizer proposes N drafts $\{\tilde{s}_{u,t}^{(c)}\}_{c=1}^{N}$ from different perspectives. For an existing skill, each draft applies at most B local edits:

$$\begin{aligned} E_{u,t}^{(c)} &\sim \text{Optimizer}(G_{u,t}, s_{u,t-1}, \mathcal{M}_{u,t-1}), \quad |E_{u,t}^{(c)}| \le B, \\ \tilde{s}_{u,t}^{(c)} &= \text{Apply}(s_{u,t-1}, E_{u,t}^{(c)}), \qquad\qquad\qquad c = 1, \dots, N. \end{aligned} \tag{4}$$

Here N is the number of drafts per skill-text target at each step. Each $E_{u,t}^{(c)}$ contains insertions, deletions, or replacements. Memory $\mathcal{M}_{u,t-1}$ supplies past drafts and feedback for the same skill text, helping avoid repeated failures. New experts are drafted in full without the edit bound. We discard instance-specific rules and duplicate procedures. Share skill candidates contain cross-family guidance, while expert candidates contain family-specific procedures. Batch screening on $\mathcal{D}_{\text{batch},t}$ selects one draft $\tilde{s}_{u,t}$ per skill-text target for paired evaluation.

Routing gradient descent. For $u = \text{route}$, $G_{\text{route},t}$ revises expert scopes without changing skill text. The new scope enters g_t and is tested at the next step without extra rollouts. For a new expert, the optimizer drafts a scope alongside the skill text and adds both to a temporary library $\tilde{\mathcal{L}}_t$ and router $\tilde{g}_t$. Before admission, we check that the scope routes enough target-family tasks to the new expert without routing checked tasks from other families to it. If either check fails, we revise the scope and repeat the checks.

Scope-conditioned paired evaluation. We compare each candidate system with the unchanged original system $(\mathcal{L}_{t-1}, g_{t-1})$ on a paired-evaluation set Q. For share-skill edits, $Q = \mathcal{D}_{\text{batch},t}$. For an existing expert assigned to family f, $Q = \mathcal{D}_{\text{batch},f,t}$. The candidate system replaces only that expert text in $\mathcal{L}_{t-1}$, preserving all other skill texts and the original router g_{t-1}.

For a new expert assigned to family f, the temporary router $\tilde{g}_t$ selects Q from $\mathcal{D}_{\text{batch},f,t}$ by retaining only tasks routed to that expert. On Q, the candidate system uses the share skill and new expert in $\tilde{\mathcal{L}}_t$, while the original system uses its share skill and any expert selected by g_{t-1} in $\mathcal{L}_{t-1}$. The preceding scope check separately tests tasks from other families.

Each system performs R runs on every task in Q, yielding original outcomes $\{y_{i,r}\}_{r=1}^{R}$ and candidate outcomes $\{\tilde{y}_{i,r}\}_{r=1}^{R}$. The net gain Δ_Q counts additional successes. The stable regression count q_Q counts tasks that succeed in all original runs but fail in all candidate runs:

$$\Delta_Q = \sum_{x_i \in Q} \sum_{r=1}^{R} (\tilde{y}_{i,r} - y_{i,r}), \qquad q_Q = \sum_{x_i \in Q} \mathbf{1}\left[\sum_r y_{i,r} = R \wedge \sum_r \tilde{y}_{i,r} = 0\right]. \tag{5}$$

History-calibrated step acceptance. Expert skill candidates first undergo three checks. **(1) Current gain and regression:** we require $\Delta_Q \ge \tau > 0$ and $q_Q \le \rho_{\text{exp}}$. **(2) Historical calibration:** we guard against an unusually weak current baseline. Past successes H_i and executions N_i on x_i under unchanged skills and routing give $p_i^{\text{hist}} = H_i/N_i$. Replacing current baseline counts with Rp_i^{hist} expected successes yields Δ_Q^{hist}; both gains must reach τ. For tasks without history, we use the current baseline rate. Large discrepancies between current and historical rates trigger additional executions. **(3) Rollout consistency:** at least ηR repetitions must yield positive gains across Q. Writing the gain in repetition r as $\Delta_Q^{(r)}$, the three checks become:

$$\begin{aligned} \Delta_Q^{\text{hist}} &= \sum_{x_i \in Q} \left(\sum_{r=1}^{R} \tilde{y}_{i,r} - R p_i^{\text{hist}} \right), \qquad \Delta_Q^{(r)} = \sum_{x_i \in Q} (\tilde{y}_{i,r} - y_{i,r}), \\ &\min\{\Delta_Q, \Delta_Q^{\text{hist}}\} \ge \tau, \quad q_Q \le \rho_{\text{exp}}, \quad \frac{1}{R} \sum_{r=1}^{R} \mathbf{1}[\Delta_Q^{(r)} > 0] \ge \eta. \end{aligned} \tag{6}$$

Share skill updates pass training acceptance when $q_Q \le \rho_{\text{sh}}$ on the full batch, restricting persistent harm across task families without imposing a positive gain threshold.

Passing candidates then undergo **(4) Held-out validation:** paired tests on $\mathcal{D}_{\text{val},f}$ (experts) or $\mathcal{D}_{\text{val}}$ (share skill). Excessive stable regressions reject both types, while a substantial success loss also rejects experts. Accepted updates and scope repairs yield the next library $\mathcal{L}_t$ and router g_t.

Optimizer memory accumulation. Memory $\mathcal{M}_{u,t}$ stores rejected drafts, outcomes, and rejection reasons for each target u. Records remove task identifiers and replace entities with roles, preserving errors such as missing prerequisites. The optimizer retrieves these records when proposing revisions for the same target, using past outcomes to refine drafts and avoid repeating unsuccessful repairs.

3.4 Skill Library Lifecycle

Starting with an empty library $\mathcal{L}_0$, CreAGD drafts the first skills from execution failures. Once introduced, the share skill remains active throughout training and accumulates cross-family guidance through revisions that pass global evaluation. Newly drafted experts remain *candidates* while their text and scope are evaluated. Those that pass training and validation acceptance enter $\mathcal{L}_t$ as *apprentices* and begin serving tasks at step $t + 1$. Their subsequent execution feedback guides further CreAGD revisions to skill text and routing scope, enabling them to become *established* experts.

We monitor each expert on fixed tasks, comparing current and admission success rates over the same number of executions. Sustained deterioration beyond a tolerance triggers *retirement*. Retired skills remain inactive *work-in-progress drafts* in optimizer memory for later revision and reevaluation. Repeated admission failures require a cooldown before retrying.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate EvoMoSkill on five complementary benchmarks. ALFWorld (Shridhar et al., 2020) tests household task execution in text environments, while ScienceWorld (Wang et al., 2022) requires interactive scientific experimentation. SearchQA (Dunn et al., 2017) evaluates question answering with retrieved snippets, SpreadsheetBench (Ma et al., 2024) tests real-world workbook manipulation, and SkillLearnBench (Zhong et al., 2026) evaluates learning reusable procedures for real-world tasks. We use disjoint training, validation, and test splits. ALFWorld, SpreadsheetBench, and SkillLearnBench use task success, ScienceWorld uses its official best score, and SearchQA uses exact match.

Baselines. We compare six baselines. No skill runs the student without skill guidance, following Yang et al. (2026a). LLM Skill uses GPT-5.6-Sol to create one skill document with Skill Creator (Anthropic, 2026b) from each benchmark's task description, example tasks, and answer trajectories. It does not revise the document. Trace2Skill (Ni et al., 2026) consolidates lessons extracted from individual trajectories into reusable skills. SkillOpt (Yang et al., 2026a) makes bounded skill-text edits from batch feedback and selects updates through validation. EvoSkill (Alzubi et al., 2026) analyzes failures to propose and select skill revisions. SKILL-KD (Shi et al., 2026a) contrasts teacher and student trajectories, tests proposed patches through student reruns, and consolidates the resulting rules.

Models and protocol. For EvoMoSkill and every baseline, we test all five benchmarks with five frozen students: DeepSeek-V4-Flash (DeepSeek-AI, 2026), Claude-Opus-4.6 (Anthropic, 2026a), Kimi-K2.6 (Kimi Team, 2026), Qwen3-32B (Qwen Team, 2025), and Qwen3.5-4B (Qwen Team, 2026). In our controlled comparison, all skill-learning methods use GPT-5.6-Sol (OpenAI, 2026) as the common teacher and skill optimizer. The ReAct execution harness, tools, test tasks, and scorer remain fixed for each benchmark and student. Further experimental settings appear in Appendix A3.

4.2 Main Results

Table 2 reports each benchmark result and the average across benchmarks. On unseen ALFWorld and ScienceWorld tasks, EvoMoSkill achieves the highest score for all five students. It leads outright in 23 of 25 benchmark–student comparisons and improves the average by 14.9–23.8 percentage points over No skill and by 4.0–6.8 points over the strongest competing baseline for each student. These gains across distinct task types show the benefit of evolving a selectively routed skill library.

Table 2: Main results on five benchmarks across five student models (%). Metrics are task success, SearchQA exact match, and ScienceWorld official best score. Green/red subscripts show percentage-point change from No skill. Bold/underline mark the best/second-best result.

Model	Method	ALFWorld	SciWorld	SearchQA	Spreadsheet	SkillLearn	Avg.
DeepSeek-V4-F.	No skill	58.2	61.7	77.6	49.6	15.4	52.5
	LLM Skill	59.7$_{+1.5}$	53.0$_{-8.7}$	73.2$_{-4.4}$	46.8$_{-2.9}$	15.4$_{+0.0}$	49.6$_{-2.9}$
	SkillOpt	76.9$_{+18.7}$	<u>71.1</u>$_{+9.4}$	82.8$_{+5.2}$	67.5$_{+17.9}$	23.1$_{+7.7}$	64.3$_{+11.8}$
	Trace2Skill	64.2$_{+6.0}$	65.8$_{+4.0}$	75.9$_{-1.7}$	69.3$_{+19.7}$	19.2$_{+3.8}$	58.9$_{+6.4}$
	EvoSkill	78.4$_{+20.2}$	54.4$_{-7.4}$	<u>83.5</u>$_{+5.9}$	55.0$_{+5.4}$	<u>26.9</u>$_{+11.5}$	59.6$_{+7.1}$
	SKILL-KD	<u>82.1</u>$_{+23.9}$	61.7$_{+0.0}$	83.1$_{+5.4}$	<u>75.4</u>$_{+25.7}$	<u>26.9</u>$_{+11.5}$	<u>65.8</u>$_{+13.3}$
	EvoMoSkill	**90.3**$_{+32.1}$	**78.5**$_{+16.8}$	**84.4**$_{+6.7}$	**79.3**$_{+29.7}$	**30.8**$_{+15.4}$	**72.6**$_{+20.1}$
Claude-Opus-4.6	No skill	80.6	75.2	82.8	56.8	15.4	62.1
	LLM Skill	84.3$_{+3.7}$	73.2$_{-2.0}$	83.0$_{+0.2}$	53.9$_{-2.9}$	11.5$_{-3.8}$	61.2$_{-1.0}$
	SkillOpt	88.8$_{+8.2}$	81.2$_{+6.0}$	<u>86.9</u>$_{+4.1}$	71.8$_{+15.0}$	<u>26.9</u>$_{+11.5}$	71.1$_{+9.0}$
	Trace2Skill	85.8$_{+5.2}$	78.5$_{+3.4}$	84.8$_{+2.0}$	67.1$_{+10.4}$	19.2$_{+3.8}$	67.1$_{+5.0}$
	EvoSkill	87.3$_{+6.7}$	81.9$_{+6.7}$	84.1$_{+1.3}$	66.1$_{+9.3}$	15.4$_{+0.0}$	66.9$_{+4.8}$
	SKILL-KD	<u>90.3</u>$_{+9.7}$	<u>82.6</u>$_{+7.4}$	85.8$_{+3.0}$	<u>78.2</u>$_{+21.4}$	23.1$_{+7.7}$	<u>72.0</u>$_{+9.8}$
	EvoMoSkill	**95.5**$_{+14.9}$	**89.9**$_{+14.8}$	**88.1**$_{+5.4}$	**80.7**$_{+23.9}$	**30.8**$_{+15.4}$	**77.0**$_{+14.9}$
Kimi-K2.6	No skill	61.2	66.4	80.7	32.5	7.7	49.7
	LLM Skill	69.4$_{+8.2}$	66.4$_{+0.0}$	80.4$_{-0.4}$	38.6$_{+6.1}$	7.7$_{+0.0}$	52.5$_{+2.8}$
	SkillOpt	79.9$_{+18.7}$	73.8$_{+7.4}$	**85.6**$_{+4.9}$	<u>47.5</u>$_{+15.0}$	19.2$_{+11.5}$	61.2$_{+11.5}$
	Trace2Skill	67.2$_{+6.0}$	65.8$_{-0.7}$	80.2$_{-0.5}$	38.9$_{+6.4}$	11.5$_{+3.8}$	52.7$_{+3.0}$
	EvoSkill	74.6$_{+13.4}$	68.5$_{+2.0}$	83.1$_{+2.4}$	41.8$_{+9.3}$	15.4$_{+7.7}$	56.7$_{+7.0}$
	SKILL-KD	<u>81.3</u>$_{+20.2}$	<u>78.5</u>$_{+12.1}$	83.2$_{+2.5}$	45.4$_{+12.9}$	<u>23.1</u>$_{+15.4}$	<u>62.3</u>$_{+12.6}$
	EvoMoSkill	**90.3**$_{+29.1}$	**84.6**$_{+18.1}$	<u>84.6</u>$_{+3.9}$	**48.6**$_{+16.1}$	**26.9**$_{+19.2}$	**67.0**$_{+17.3}$
Qwen3-32B	No skill	40.3	30.9	72.3	22.5	0.0	33.2
	LLM Skill	59.7$_{+19.4}$	26.9$_{-4.0}$	69.6$_{-2.7}$	25.0$_{+2.5}$	0.0$_{+0.0}$	36.2$_{+3.0}$
	SkillOpt	76.1$_{+35.8}$	<u>46.3</u>$_{+15.4}$	<u>80.1</u>$_{+7.8}$	39.6$_{+17.1}$	<u>15.4</u>$_{+15.4}$	<u>51.5</u>$_{+18.3}$
	Trace2Skill	64.2$_{+23.9}$	31.5$_{+0.7}$	72.1$_{-0.2}$	24.6$_{+2.1}$	3.8$_{+3.8}$	39.3$_{+6.1}$
	EvoSkill	72.4$_{+32.1}$	40.3$_{+9.4}$	76.1$_{+3.8}$	31.1$_{+8.6}$	7.7$_{+7.7}$	45.5$_{+12.3}$
	SKILL-KD	<u>82.8</u>$_{+42.5}$	43.0$_{+12.1}$	78.7$_{+6.4}$	<u>40.4</u>$_{+17.9}$	11.5$_{+11.5}$	51.3$_{+18.1}$
	EvoMoSkill	**86.6**$_{+46.3}$	**47.7**$_{+16.8}$	**81.0**$_{+8.7}$	**42.9**$_{+20.4}$	**19.2**$_{+19.2}$	**55.5**$_{+22.3}$
Qwen3.5-4B	No skill	32.1	22.2	68.2	9.6	0.0	26.4
	LLM Skill	30.6$_{-1.5}$	27.5$_{+5.4}$	64.9$_{-3.4}$	13.6$_{+3.9}$	0.0$_{+0.0}$	27.3$_{+0.9}$
	SkillOpt	<u>77.6</u>$_{+45.5}$	<u>36.9</u>$_{+14.8}$	74.3$_{+6.1}$	14.3$_{+4.7}$	<u>7.7</u>$_{+7.7}$	42.2$_{+15.7}$
	Trace2Skill	54.5$_{+22.4}$	28.2$_{+6.0}$	61.9$_{-6.3}$	11.1$_{+1.4}$	**11.5**$_{+11.5}$	33.4$_{+7.0}$
	EvoSkill	66.4$_{+34.3}$	34.2$_{+12.1}$	68.6$_{+0.4}$	19.3$_{+9.7}$	3.8$_{+3.8}$	38.5$_{+12.1}$
	SKILL-KD	70.2$_{+38.1}$	34.9$_{+12.8}$	<u>78.8</u>$_{+10.6}$	<u>23.9</u>$_{+14.3}$	**11.5**$_{+11.5}$	<u>43.9</u>$_{+17.4}$
	EvoMoSkill	**88.1**$_{+56.0}$	**41.6**$_{+19.5}$	**81.1**$_{+12.9}$	**28.9**$_{+19.3}$	**11.5**$_{+11.5}$	**50.3**$_{+23.8}$

The one-shot LLM Skill baseline is less reliable: relative to No skill, it improves 9 of 25 comparisons, reduces performance in 11, and leaves 5 unchanged. On SearchQA, four of five students perform worse with that skill. By contrast, EvoMoSkill improves on No skill in all 25 comparisons. For Qwen3.5-4B, ALFWorld rises from 32.1% to 88.1% and SpreadsheetBench from 9.6% to 28.9%. The contrast supports execution-guided skill revision and selective reuse as a more reliable route to gains across tasks and students with different no-skill performance.

Learning cost and benchmark performance. On ALFWorld, EvoMoSkill improves across learning budgets for both students and attains the highest final success (Figure 3). Later checkpoints outperform other iterative skill methods at comparable or lower cost. Continued library optimization thus turns additional learning budget into stronger unseen-task performance.

Cross-model transfer. We evaluate frozen libraries on students other than those used for training (Table 3). All 18 transfers across three benchmarks exceed both No skill and target-model LLM Skill. Gains across every source–target pair show that learned skill texts and routing scopes remain useful with a different executor, extending the benefit of optimization beyond the training student.

4.3 Ablation Studies: Why Does the Mixture-of-Skills Help?

Q1: Does the mixture-of-skills organization improve performance? We train three alternatives from scratch. *Single skill* optimizes one document for all tasks. *Experts only* learns routed expert

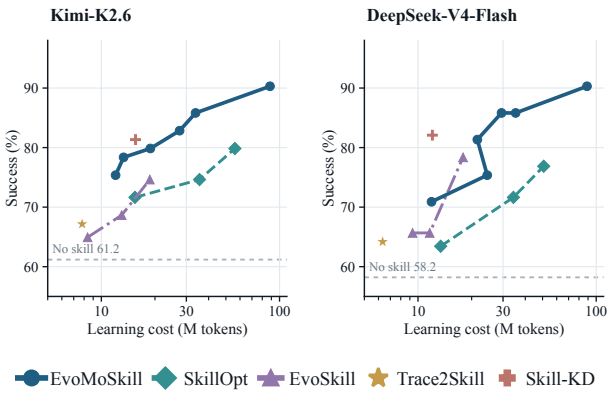

Figure 3: ALFWorld unseen success versus learning cost (M tokens). Gray dashed lines show No skill.

Benchmark	Source	Target	LLM	Transfer
ALFWorld	Qwen3-32B	Qwen3.5-4B	30.6	82.1$_{+51.5}$
		Opus-4.6	84.3	87.3$_{+3.0}$
	Qwen3.5-4B	Qwen3-32B	59.7	76.9$_{+17.2}$
		Opus-4.6	84.3	91.0$_{+6.7}$
	Opus-4.6	Qwen3-32B	59.7	80.6$_{+20.9}$
		Qwen3.5-4B	30.6	64.9$_{+34.3}$
SearchQA	Qwen3-32B	Qwen3.5-4B	64.9	77.4$_{+12.6}$
		Opus-4.6	83.0	86.9$_{+3.9}$
	Qwen3.5-4B	Qwen3-32B	69.6	78.1$_{+8.5}$
		Opus-4.6	83.0	87.4$_{+4.4}$
	Opus-4.6	Qwen3-32B	69.6	75.1$_{+5.5}$
		Qwen3.5-4B	64.9	72.4$_{+7.6}$
Spreadsheet Bench	Qwen3-32B	Qwen3.5-4B	13.6	27.5$_{+13.9}$
		Opus-4.6	53.9	74.3$_{+20.4}$
	Qwen3.5-4B	Qwen3-32B	25.0	36.4$_{+11.4}$
		Opus-4.6	53.9	69.3$_{+15.4}$
	Opus-4.6	Qwen3-32B	25.0	39.3$_{+14.3}$
		Qwen3.5-4B	13.6	23.6$_{+10.0}$

Table 3: Cross-model transfer (%). Green subscripts: gains vs. LLM Skill.

skills without a share skill. *Static partition* trains a separate skill for each task family and routes tasks by their descriptions without updating those assignments. EvoMoSkill exceeds the first two in all nine benchmark–student settings and Static partition in seven (Table 4). On ALFWorld and SearchQA, it leads every alternative for all three students; on SpreadsheetBench, it leads Single skill and Experts only for all three. These results favor joint optimization of the share skill and routed experts over simpler library designs across the tested tasks and students.

Table 4: Skill-library organization ablation. Black values are benchmark scores (%). Gray subscripts show percentage-point changes from EvoMoSkill. Blue/red shading marks gains/losses.

	ALFWorld			SearchQA			SpreadsheetBench		
Method	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**
EvoMoSkill	**86.6**	**88.1**	**95.5**	**81.0**	**81.1**	**88.1**	**42.9**	**28.9**	**80.7**
Single skill	76.1$_{-10.5}$	79.1$_{-9.0}$	89.6$_{-6.0}$	79.9$_{-1.1}$	73.6$_{-7.6}$	85.3$_{-2.9}$	37.5$_{-5.4}$	18.2$_{-10.7}$	72.5$_{-8.2}$
Experts only	78.4$_{-8.2}$	81.3$_{-6.7}$	93.3$_{-2.2}$	78.9$_{-2.1}$	79.1$_{-2.0}$	85.7$_{-2.4}$	34.6$_{-8.2}$	22.9$_{-6.1}$	76.4$_{-4.3}$
Static partition	83.6$_{-3.0}$	82.8$_{-5.2}$	94.8$_{-0.7}$	80.1$_{-0.9}$	79.6$_{-1.6}$	86.9$_{-1.3}$	43.9$_{+1.1}$	25.7$_{-3.2}$	81.4$_{+0.7}$

Q2: Which acceptance checks make skill updates reliable? We ablate the four acceptance checks of Section 3.3 in turn: *rollout consistency*, *historical calibration*, *the regression bound*, and *held-out validation*. Removing the regression bound lowers test-set scores in all nine benchmark–student settings by 6.4 points on average (Table 5). Weakening or removing each other check lowers scores in eight of nine settings. Together, these results indicate that repeated trials and historical calibration guard against chance gains, while regression and validation checks screen harmful edits. The checks therefore make updates more reliable across tasks and students.

Table 5: Update acceptance ablation. Black values are benchmark scores (%). Gray subscripts show percentage-point changes from EvoMoSkill. Blue/red shading marks gains/losses.

	ALFWorld			SearchQA			SpreadsheetBench		
Method	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**
EvoMoSkill	**86.6**	**88.1**	**95.5**	**81.0**	**81.1**	**88.1**	**42.9**	**28.9**	**80.7**
One rollout	76.9$_{-9.7}$	81.3$_{-6.7}$	95.5$_{+0.0}$	77.4$_{-3.6}$	77.3$_{-3.8}$	86.6$_{-1.6}$	40.0$_{-2.9}$	24.6$_{-4.3}$	72.1$_{-8.6}$
No history	79.1$_{-7.5}$	86.6$_{-1.5}$	90.3$_{-5.2}$	76.9$_{-4.1}$	75.4$_{-5.7}$	88.3$_{+0.2}$	38.6$_{-4.3}$	28.2$_{-0.7}$	77.9$_{-2.9}$
No regression bound	82.8$_{-3.7}$	81.3$_{-6.7}$	94.0$_{-1.5}$	72.3$_{-8.7}$	72.3$_{-8.9}$	86.0$_{-2.1}$	35.7$_{-7.2}$	21.8$_{-7.1}$	69.3$_{-11.4}$
No validation gate	76.9$_{-9.7}$	80.6$_{-7.5}$	91.8$_{-3.7}$	78.1$_{-2.9}$	77.6$_{-3.6}$	91.1$_{+3.0}$	30.7$_{-12.2}$	26.4$_{-2.5}$	74.3$_{-6.4}$

Q3: Does learned routing matter beyond skill content? To isolate routing, we fix the final library and compare four deployment rules. *Admission scopes* restores the scope recorded when each expert was admitted. *Random router* selects an expert at random. *Share skill only* omits experts. *Concatenated skills* loads the share skill and every expert for each task. In *Frozen scopes*, each new expert retains its admission scope throughout training, while only skill content is revised.

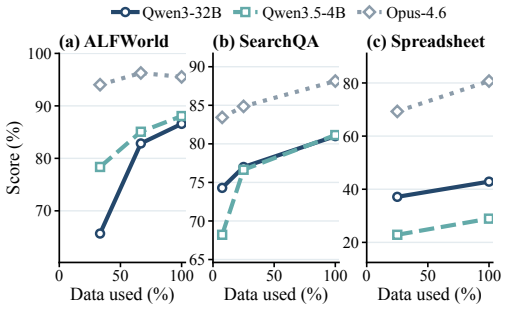

Figure 4: Test-set scores by training-data fraction. Panels (a)–(c) show ALFWorld, SearchQA, and SpreadsheetBench, respectively.

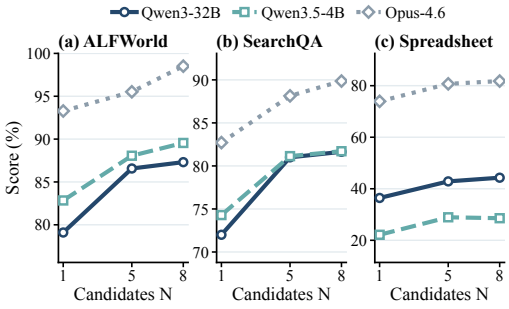

Figure 5: Test-set scores by candidate count (N). Panels (a)–(c) show ALFWorld, SearchQA, and SpreadsheetBench.

EvoMoSkill leads random routing, share-skill-only execution, and frozen scopes in all nine settings, concatenation in eight with one tie, and admission scopes in seven (Table 6). The largest losses occur when experts are selected at random or omitted. These results support task-aware expert selection and continued adaptation of expert scopes.

Table 6: Expert routing ablation. Black values are benchmark scores (%). Gray subscripts show percentage-point changes from EvoMoSkill. Blue/red shading marks gains/losses.

	ALFWorld			SearchQA			SpreadsheetBench		
Method	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**	**Qwen3-32B**	**Qwen3.5-4B**	**Opus-4.6**
EvoMoSkill	**86.6**	**88.1**	**95.5**	**81.0**	**81.1**	**88.1**	**42.9**	**28.9**	**80.7**
Frozen scopes	79.9$_{-6.7}$	82.1$_{-6.0}$	94.0$_{-1.5}$	74.3$_{-6.7}$	79.0$_{-2.1}$	85.3$_{-2.9}$	35.7$_{-7.2}$	24.3$_{-4.6}$	69.3$_{-11.4}$
Admission scopes	85.1$_{-1.5}$	86.6$_{-1.5}$	97.0$_{+1.5}$	78.7$_{-2.3}$	82.6$_{+1.4}$	84.7$_{-3.4}$	41.1$_{-1.8}$	26.4$_{-2.5}$	80.0$_{-0.7}$
Concatenated skills	84.3$_{-2.2}$	84.3$_{-3.7}$	95.5$_{+0.0}$	77.0$_{-4.0}$	73.4$_{-7.7}$	85.6$_{-2.6}$	36.4$_{-6.4}$	25.0$_{-3.9}$	75.7$_{-5.0}$
Share skill only	78.4$_{-8.2}$	82.8$_{-5.2}$	92.5$_{-3.0}$	72.0$_{-9.0}$	68.7$_{-12.4}$	81.3$_{-6.9}$	38.9$_{-3.9}$	22.9$_{-6.1}$	71.4$_{-9.3}$
Random router	77.6$_{-9.0}$	79.1$_{-9.0}$	88.1$_{-7.5}$	73.0$_{-8.0}$	67.0$_{-14.1}$	82.6$_{-5.6}$	37.5$_{-5.4}$	20.0$_{-8.9}$	64.3$_{-16.4}$

Q4: How do training data and unseen task families affect performance? We train on increasing fractions of the largest tested task subset per benchmark and evaluate each run on the same test set (Figure 4). The largest subset outperforms the smallest in all nine benchmark–student settings. On SpreadsheetBench, moving from 25% to 100% improves Qwen3.5-4B by 6.0 points and Claude-Opus-4.6 by 11.4 points. We also exclude an entire ALFWorld task family during training and evaluate it afterward (Table A8). All three students retain substantial success on both excluded families. These results show that greater data use generally improves test-set performance and that learned procedures remain useful on unseen task families without family-specific training examples.

Q5: How many candidate drafts are needed? We test $N \in \{1, 5, 8\}$ candidate drafts per target (Equation 4), with $N = 5$ as the default and other settings fixed (Figure 5). Increasing N from one to five improves all nine benchmark–student settings by 6.2 points on average. At $N = 8$, the average increases by another 1.1 points. Comparing several drafts, rather than relying on one proposal, therefore accounts for most of the benefit of deliberative skill revision.

5 Conclusion

In this work, we introduce EvoMoSkill, a mixture-of-skills framework that jointly optimizes a share skill, task-family experts, and their router. Each CreAGD optimization step follows five stages. First, *forward execution and credit assignment* traces failures to their source. Second, *textual and routing gradient descent* proposes revisions. Third, *scope-conditioned paired evaluation* measures their effects. Fourth, *history-calibrated step acceptance* selects updates. Finally, *optimizer memory accumulation* guides later revisions. Across five benchmarks and five student models, EvoMoSkill improves average performance by 14.9–23.8 points over no skill and surpasses the strongest baseline. Ablations further validate the roles of its main components in these gains. These results highlight that evolving mixture-of-skills yields more adaptable agents.

AI USE STATEMENT

In this work, we used generative AI tools to help plan experimental comparisons and interpret and check results. We also used LLMs as the teacher, optimizer, router, and student agents in our experiments. We did not use generative AI to generate synthetic datasets or develop theoretical proofs, as these activities are not part of this study. Additionally, we used generative AI to polish author-written drafts, assist with subsequent manuscript revisions, and prepare figures and tables. We checked numerical claims, figures, and analyses against the experimental records, reviewed all AI-assisted work, and take responsibility for the final text, claims, and artifacts of this paper.

ETHICS STATEMENT

We study LLM-agent skill optimization using existing benchmarks. We did not recruit human participants or collect sensitive personal data. We are not aware of ethical concerns specific to this study beyond the general risks associated with deploying LLM agents.

REPRODUCIBILITY STATEMENT

We describe the method and experimental protocol in Sections 3.3 and 4.1. Appendices A2 and A3 document benchmark splits, hyperparameters, and additional results. We will release the implementation and experiment scripts in a code repository to support reproduction of our results.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

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. Introducing Claude Opus 4.6, 2026a. URL `https://www.anthropic.com/news/claude-opus-4-6`. February 5, 2026.

Anthropic. Improving skill-creator: Test, measure, and refine agent skills, September 2026b. URL `https://claude.com/blog/improving-skill-creator-test-measure-and-refine-agent-skills`.

Kunfeng Chen, Qihuang Zhong, Juhua Liu, and Bo Du. SkillCAT: Contrastive assessment and topology-aware skill self-evolution for llm agents. *arXiv preprint arXiv:2606.13317*, 2026.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Giuseppe Destefanis. Authoring agent skills: A software-engineering approach. *arXiv preprint arXiv:2607.25032*, 2026.

Giuseppe Destefanis, Daniel Graziotin, Matteo Vaccargiu, and Marco Ortu. GitSkills: A dataset of agent skills on github. *arXiv preprint arXiv:2608.10906*, 2026.

Kexin Ding, Yang Zhou, Can Jin, Feng Tong, Mu Zhou, and Dimitris N Metaxas. Agent skill evaluation and evolution: Frameworks and benchmarks. *arXiv preprint arXiv:2606.11435*, 2026.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A new Q&A dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*, 2017.

Yudong Gao, Zongjie Li, Yuanyuan Yuan, Zimo Ji, Pingchuan Ma, and Shuai Wang. SkillReducer: Optimizing llm agent skills for token efficiency. *arXiv preprint arXiv:2603.29919*, 2026.

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.

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

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. SoK: Agentic skills–beyond tool use in llm agents. *arXiv preprint arXiv:2602.20867*, 2026b.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, et al. EmbodiSkill: Skill-aware reflection for self-evolving embodied agents. *arXiv preprint arXiv:2605.10332*, 2026.

Kimi Team. Kimi K2.6: Advancing open-source coding, February 2026. URL https://www.kimi.ai/blog/kimi-k2-6/.

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

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026b.

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.

Hongqiang Lin, Chao Liu, Xiaofan Bai, Xuan Jin, Yuhong Li, Nenggan Zheng, and Xipeng Cao. Rethinking Self-Evolution: A constrained exploration-exploitation process for mitigating skill overfitting. *arXiv preprint arXiv:2607.26643*, 2026.

Xingyan Liu, Xiyue Luo, Linyu Li, Ganghong Huang, Jianfeng Liu, and Honglin Qiao. SkillForge: Forging domain-specific, self-evolving agent skills in cloud technical support. *arXiv preprint arXiv:2604.08618*, 2026.

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

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadSheetBench: Towards challenging real world spreadsheet manipulation. *Advances in Neural Information Processing Systems*, 37:94871–94908, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-Refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

OpenAI. GPT-5.6 system card, July 2026. URL https://openai.com/index/gpt-5-6/.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. SkillOS: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. SkillFoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources. *arXiv preprint arXiv:2604.03964*, 2026.

Qiming Shi, Yibo Dou, Jiawen Zhu, Yulong Tao, Linbo Jin, Zhaolu Kang, Yunfan Zhou, and Di Weng. SKILL-KD: Contrastive skill distillation for llm agents. *arXiv preprint arXiv:2607.28048*, 2026a.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026b.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, et al. From context to skills: Can language models learn from context skillfully? *arXiv preprint arXiv:2604.27660*, 2026.

Yu Tian, Jiawei Chen, Lifan Zheng, Mingxiang Tao, Xinyi Zeng, Zhaoxia Yin, Hang Su, and Xian Sun. Skills-Coach: A self-evolving skill optimizer via training-free grpo. *arXiv preprint arXiv:2604.27488*, 2026.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, Dong Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl. *arXiv preprint arXiv:2603.28716*, 2026.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, et al. SkillX: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*, 2026.

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. *arXiv preprint arXiv:2305.16291*, 2023.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

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

Qianxi Yan, Chunrong Chen, Jiuzhou Zhao, Min Zhang, Yongzhou Xu, and Xiaochuan Xu. SkillEvo: Self-renewing evolution gradients from multi-turn interaction feedback. *arXiv preprint arXiv:2608.13120*, 2026.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026a.

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*, 2026b.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping Agents for the Real World with Agent Skills, October 2025. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026a.

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*, 2026b.

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.

YanZhao Zheng, ZhenTao Zhang, Chao Ma, YuanQiang Yu, JiHuai Zhu, Yong Wu, Tianze Xu, Baohua Dong, Hangcheng Zhu, Ruohui Huang, et al. SkillRouter: Skill routing for llm agents at scale. *arXiv preprint arXiv:2603.22455*, 2026.

Shanshan Zhong, Yi Lu, Jingjie Ning, Yibing Wan, Lihan Feng, Yuyi Ao, Leonardo FR Ribeiro, Markus Dreyer, Sean Ammirati, and Chenyan Xiong. SkillLearnBench: Benchmarking continual learning methods for agent skill generation on real-world tasks. *arXiv preprint arXiv:2604.20087*, 2026.

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.

A1 LIMITATIONS

While EvoMoSkill performs well across diverse agent tasks, several limitations remain. First, generating and validating multiple candidate skills requires repeated LLM calls and agent rollouts, increasing training time and token cost. More efficient candidate screening could reduce this overhead. Second, expert creation and routing rely on a partition of training tasks into families. When task boundaries overlap or shift, the resulting scopes may not identify where specialized guidance is needed. Third, a stronger teacher model diagnoses failures and proposes revisions. Errors in these diagnoses can propagate into skill updates, motivating further study of sensitivity to the teacher. Finally, our capacity study tests at most $K_{\max} = 12$ experts. Larger libraries may make routing and expert maintenance more difficult, so their scalability remains to be evaluated.

A2 EXPERIMENTAL DETAILS

A2.1 BENCHMARKS AND DATA SPLITS

The five benchmarks cover different task interfaces and evaluation objectives. ALFWorld (Shridhar et al., 2020) presents household goals in a text environment, where an agent must complete a sequence of actions to succeed. We train on 90 tasks, use 24 tasks for validation, and report success on all 134 tasks in the unseen test split. ScienceWorld (Wang et al., 2022) requires interactive scientific experiments in a simulated environment. We use 60 training tasks and 18 validation tasks, and report the official best score on its 149-task test split. This progress-based score is reported separately from the binary task-success metric used for ALFWorld.

SearchQA (Dunn et al., 2017) provides fixed retrieved snippets with each question and evaluates the resulting answer by exact match. SpreadsheetBench (Ma et al., 2024) pairs natural-language instructions with real workbooks. A task counts as successful only when the modified workbook passes its checks. SkillLearnBench (Zhong et al., 2026) contains verifiable real-world tasks that test whether learned procedures improve execution. Table A1 lists the train, validation, and test sizes and primary metrics for all five benchmarks. All methods use the same test set. Training tasks support skill updates, validation tasks guide candidate screening and library selection, and test tasks are used only for final evaluation.

Table A1: Benchmark splits and primary evaluation metrics. ALFWorld uses the unseen test split, and ScienceWorld uses its test split.

Benchmark	Train	Validation	Test	Metric
ALFWorld	90	24	134	Task success
ScienceWorld	60	18	149	Official best score
SearchQA	400	200	1,400	Exact match
SpreadsheetBench	80	40	280	Hard accuracy
SkillLearnBench	51	18	26	Task success

A2.2 HYPERPARAMETER SETTINGS

Table A2 summarizes the settings most relevant to agent execution and skill optimization. Student models remain frozen during skill learning and use the same decoding temperature for training rollouts and final evaluation. The teacher uses a higher temperature when proposing revisions, whereas student execution, routing, and paired evaluation use the lower temperature shown in the table. Execution budgets follow the task interface: ALFWorld and ScienceWorld require multiple environment actions, SearchQA uses a single answer, and SpreadsheetBench permits multi-turn code generation and workbook editing before the resulting file is checked.

Skill optimization processes training tasks in batches of 32, with $N = 5$ candidate drafts proposed for each update target by default. Candidate and original systems are compared over three paired trials before history-calibrated checks and held-out validation determine acceptance. ALFWorld and ScienceWorld train for up to six epochs, while SearchQA and SpreadsheetBench train for up to four. Expert capacity also varies by benchmark to accommodate differences in task-family structure. Together, these settings specify the budgets for candidate search, validation, and final evaluation.

Table A2: Selected execution and optimization hyperparameters. Benchmark-specific values are ordered as ALFWorld / ScienceWorld / SearchQA / SpreadsheetBench.

Hyperparameter	Value	Notes
LLM decoding		
Student temperature	0.1	Training and final evaluation
Teacher temperature	1.0	Skill proposal and revision
Router and gate temperature	0.1	Routing and paired evaluation
Maximum output tokens	16,384	Student and teacher calls
Random seed	42	Training and evaluation
Agent execution		
Maximum execution steps	50 / 40 / 1 / 30	Benchmark-dependent agent budget
Skill optimization		
Training epochs	6 / 6 / 4 / 4	Full training runs
Batch size	32	Tasks per optimization batch
Candidate drafts N	5	Per skill-update target
Paired gate trials	3	Candidate comparison
Expert capacity $K_{\max}$	7 / 11 / 6 / 4	Benchmark-specific expert limit

A3 Additional Experimental Results

In this section, we report additional experiments and more detailed ablations of the components of EvoMoSkill. Unless stated otherwise, each comparison uses the test set for the same benchmark and student while retaining the execution and evaluation conditions of the full method.

A3.1 Skill Library Organization

To assess skill-library organization, we train three alternatives from scratch: (1) **Single skill** optimizes one skill document for all tasks; (2) **Experts only** trains routed expert skills without a share skill; and (3) **Static partition** optimizes one skill per training family and assigns tasks by a fixed, description-based router. Table A3 reports test-set scores for three benchmarks and three students. EvoMoSkill exceeds **Single skill** and **Experts only** in all nine benchmark–student settings, by 6.8

Table A3: Skill-library organization: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Full**	**86.57**	**88.06**	**95.52**
	Single skill	76.12	79.10	89.55
	Experts only	78.36	81.34	93.28
	Static partition	83.58	82.84	94.78
SearchQA	**Full**	**81.00**	**81.14**	**88.14**
	Single skill	79.93	73.57	85.29
	Experts only	78.86	79.14	85.71
	Static partition	80.07	79.57	86.86
Spreadsheet	**Full**	**42.86**	**28.93**	**80.71**
	Single skill	37.50	18.21	72.50
	Experts only	34.64	22.86	76.43
	Static partition	43.93	25.71	81.43

and 4.7 percentage points on average, respectively. It also surpasses **Static partition** in seven settings and leads all three alternatives throughout ALFWorld and SearchQA. These results indicate that jointly optimizing the share skill, expert skills, and task assignments improves performance more consistently than the simpler library organizations.

A3.2 Interference from Accepted Skill Updates

To examine whether accepted updates disrupt previously solved tasks, we audit expert-skill and share-skill edits on ALFWorld. For expert edits, we count both stable regressions, in which the pre-

vious version succeeds and the revised version fails across repeated trials, and any-trial regressions on previously stable tasks. For each share-skill edit, we compare family-level success under the original and revised systems on $Q = \mathcal{D}_{\mathrm{batch},t}$, the paired-evaluation batch spanning all six families. Table A4 separates update-level from family-level measurements.

Table A4: Effects of accepted ALFWorld skill updates. Expert regressions are counted across accepted edits. Share-skill regressions are counted across family evaluations.

Update lane	Accepted edits	Audit units	Observed regression
Expert skill	27	27 edits	0 stable, 8 any-trial
Share skill	7	42 family checks	2 one-task regressions

None of the 27 accepted expert edits caused a stable regression, although eight showed an any-trial regression. Of the 42 family-level comparisons for seven accepted share-skill edits, only two were negative, each by one successful task. The acceptance checks rejected candidates with larger family-level losses. Together, these results show that scoped expert edits preserve stable behavior and that regression checks limit interference from global share-skill revisions.

A3.3 Reliability of Update Acceptance

To measure the contribution of the acceptance checks in Section 3.3, we retrain after changing one check at a time. **One rollout** replaces repeated evaluation with a single run. **No history** removes historical calibration. **No regression bound** removes the stable-regression criterion. **No validation gate** removes held-out validation. Table A5 reports the final test-set scores.

Table A5: Update-acceptance ablations: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Full**	**86.57**	**88.06**	**95.52**
	One rollout	76.87	81.34	95.52
	No history	79.10	86.57	90.30
	No regression bound	82.84	81.34	94.03
	No validation gate	76.87	80.60	91.79
SearchQA	**Full**	**81.00**	**81.14**	**88.14**
	One rollout	77.43	77.29	86.57
	No history	76.86	75.43	88.29
	No regression bound	72.29	72.29	86.00
	No validation gate	78.14	77.57	91.14
Spreadsheet	**Full**	**42.86**	**28.93**	**80.71**
	One rollout	40.00	24.64	72.14
	No history	38.57	28.21	77.86
	No regression bound	35.71	21.79	69.29
	No validation gate	30.71	26.43	74.29

Removing the regression bound lowers scores in all nine settings by 6.4 percentage points on average. Weakening or removing each of the other checks lowers scores in eight settings. The regression bound has the clearest effect, while repeated trials, historical calibration, and held-out validation also contribute consistently. The combined checks support reliable updates across tasks and students.

A3.4 The Role of Learned Routing

To distinguish the effects of routing and skill text, we evaluate five alternatives. **Frozen scopes** fixes the scope of each new expert at admission while continuing to update skill text during training. The other four hold the final skill texts fixed: **Admission scopes** restores the scopes recorded when experts were admitted; **Random router** selects an expert at random; **Share skill only** omits experts; and **Concatenated skills** loads the share skill and every expert for each task. Together, these settings test scope adaptation during training and expert selection at deployment. EvoMoSkill exceeds **Frozen scopes**, **Random router**, and **Share skill only** in all nine settings. It leads **Concatenated skills** in eight settings with one tie, and **Admission scopes** in seven. Random selection or omission

Table A6: Routing interventions: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Full**	**86.57**	**88.06**	**95.52**
	Frozen scopes	79.85	82.09	94.03
	Random router	77.61	79.10	88.06
	Share skill only	78.36	82.84	92.54
	Concatenated skills	84.33	84.33	95.52
	Admission scopes	85.07	86.57	97.01
SearchQA	**Full**	**81.00**	**81.14**	**88.14**
	Frozen scopes	74.29	79.00	85.29
	Random router	73.00	67.00	82.57
	Share skill only	72.00	68.71	81.29
	Concatenated skills	77.00	73.43	85.57
	Admission scopes	78.71	82.57	84.71
Spreadsheet	**Full**	**42.86**	**28.93**	**80.71**
	Frozen scopes	35.71	24.29	69.29
	Random router	37.50	20.00	64.29
	Share skill only	38.93	22.86	71.43
	Concatenated skills	36.43	25.00	75.71
	Admission scopes	41.07	26.43	80.00

of experts produces the largest average losses. These comparisons show the value of task-aware expert selection, while the lower scores under **Frozen scopes** also support scope adaptation during training.

A3.5 Training-Data Efficiency and Unseen-Family Generalization

To assess data efficiency, we vary the fraction of training tasks while keeping validation and test sets fixed. For ALFWorld, 30, 60, and 90 tasks represent 33.3%, 66.7%, and 100% of the largest subset tested. For SearchQA, 30, 100, and 400 tasks represent 7.5%, 25%, and 100%. For SpreadsheetBench, 20 and 80 tasks represent 25% and 100%. We also test unseen-family generalization by excluding either the ALFWorld two-object or look family during training and evaluating on the excluded family. Tables A7 and A8 report the aggregate and family-specific scores. The largest training subset outperforms the smallest in all nine benchmark–student settings, although the ALFWorld score of Claude-Opus-4.6 peaks at the intermediate fraction. When an entire ALFWorld family is excluded, success rates remain 0.529–0.882 on two-object and 0.722–1.000 on look (Table A8). Students solve at most three fewer tasks than when the family is included in training. Thus, more data generally improves test-set performance, and procedures learned from other families remain useful on unseen tasks.

Table A7: Test-set scores (%) by training-data fraction and ALFWorld family holdout. Fractions are relative to the largest training subset tested for each benchmark.

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**33.3% data**	65.67	78.36	94.03
	66.7% data	82.84	85.07	96.27
	100% data	86.57	88.06	95.52
	Hold out two-object	82.84	85.82	94.03
	Hold out look	83.58	85.07	95.52
SearchQA	**7.5% data**	74.29	68.21	83.43
	25% data	77.00	76.64	84.86
	100% data	81.00	81.14	88.14
Spreadsheet	**25% data**	37.14	22.86	69.29
	100% data	42.86	28.93	80.71

Table A8: ALFWorld success rates on task families excluded from training. Decimal values are successful unseen tasks divided by the family test size ($n = 17$ for two-object and $n = 18$ for look). Full includes the family during training, whereas Family holdout excludes it.

Excluded family	Training	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
Two-object ($n = 17$)	**Full**	0.706	1.000	1.000
	Family holdout	0.529	0.824	0.882
Look ($n = 18$)	**Full**	1.000	0.889	1.000
	Family holdout	0.889	0.722	1.000

A3.6 Cross-Student Transfer of Skill Libraries

To test whether a skill library transfers between student models, we train a separate source library for each of three students on each benchmark. At evaluation, the source skill texts and router scopes are frozen and used by a different target student without further training. We compare each transfer with both no-skill execution and the library trained directly for the target student, shown on the matrix diagonal of Table A9. The SpreadsheetBench libraries use two training epochs.

Table A9: Cross-student transfer: target test-set scores (%). Rows identify the source library, and columns identify the executing student. Diagonal cells use libraries trained for the target student. Bold marks the highest score in each benchmark–target column. ALFWorld and SpreadsheetBench use hard accuracy ($n = 134$ and 280). SearchQA uses exact match ($n = 1{,}400$).

Source library	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld			
No skill	40.30	32.09	80.60
Qwen3-32B	**86.57**	82.09	87.31
Qwen3.5-4B	76.87	**88.06**	91.04
Claude-Opus-4.6	80.60	64.93	**95.52**
SearchQA			
No skill	72.29	68.21	82.79
Qwen3-32B	**81.00**	77.43	86.86
Qwen3.5-4B	78.14	**81.14**	87.36
Claude-Opus-4.6	75.14	72.43	**88.14**
SpreadsheetBench			
No skill	22.50	9.64	56.79
Qwen3-32B	**42.86**	27.50	74.29
Qwen3.5-4B	36.43	**28.93**	69.29
Claude-Opus-4.6	39.29	23.57	**80.71**

All 18 cross-student deployments improve on the No skill score of the target student, with gains of 2.85–50.00 percentage points. In every benchmark–target setting, the library trained directly for the target achieves the highest score, exceeding all 18 transferred alternatives. These results show that skill texts and routing scopes learned with one student remain useful to another without additional training, while target-specific optimization provides the strongest test-set performance.

A3.7 Expert Lifecycle and Optimizer Memory

To test the contribution of library maintenance, we retrain after removing one mechanism at a time. **No retirement** keeps experts despite sustained degradation. **No draft reuse** permits retirement but does not reuse retired drafts in later proposals. **No rejected memory** retains other optimizer records but omits rejected candidates and their feedback (Table A10). Removing any of the three mechanisms lowers scores in all nine settings. The full method leads **No retirement**, **No draft reuse**, and **No rejected memory** by 9.7, 7.2, and 7.5 percentage points on average, respectively. Retirement has the largest mean effect, while draft reuse and rejected-candidate memory also provide consistent

Table A10: Expert-lifecycle and memory ablations: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Full**	**86.57**	**88.06**	**95.52**
	No retirement	76.87	73.88	92.54
	No draft reuse	80.60	81.34	91.04
	No rejected memory	81.34	82.09	91.79
SearchQA	**Full**	**81.00**	**81.14**	**88.14**
	No retirement	70.57	73.50	83.29
	No draft reuse	72.86	69.29	84.43
	No rejected memory	74.29	72.29	84.86
Spreadsheet	**Full**	**42.86**	**28.93**	**80.71**
	No retirement	35.71	12.86	66.07
	No draft reuse	33.57	23.21	72.14
	No rejected memory	33.57	17.14	68.21

gains. These results show that lifecycle maintenance and optimizer memory support sustained skill improvement.

A3.8 Contribution of Paired Teacher–Student Evidence

To test whether task-matched teacher–student evidence improves skill revision, we vary the trajectories available to the optimizer. **Full** compares teacher successes with student executions on the same tasks. **Failures only** provides student failure trajectories, **Successes only** provides teacher success trajectories, and **Unpaired evidence** provides both types without task-level matching (Table A11).

Table A11: Teacher–student evidence ablations: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Full**	**86.57**	**88.06**	**95.52**
	Failures only	76.87	75.37	94.03
	Successes only	78.36	81.34	92.54
	Unpaired evidence	81.34	85.07	89.55
SearchQA	**Full**	**81.00**	**81.14**	**88.14**
	Failures only	73.71	76.29	83.29
	Successes only	75.71	72.14	86.29
	Unpaired evidence	73.57	75.57	80.86
Spreadsheet	**Full**	**42.86**	**28.93**	**80.71**
	Failures only	34.29	20.00	70.71
	Successes only	38.57	22.86	71.43
	Unpaired evidence	39.29	15.71	66.43

Full exceeds all three restricted-evidence variants in every setting, by 7.6, 6.0, and 7.3 percentage points on average relative to **Failures only**, **Successes only**, and **Unpaired evidence**. The consistent advantage over both single-source and unpaired evidence shows that matching teacher successes with student executions on the same task provides more useful information for skill revision.

A3.9 Candidate Count and Expert Capacity

To assess candidate search and expert capacity, we vary one setting at a time. The number of drafts per update target is $N = 5$ by default, compared with $N = 1$ and $N = 8$. The default expert limits for ALFWorld, SearchQA, and SpreadsheetBench are 7, 6, and 4, respectively. The capacity scan sets each limit to $K_{\max} = 12$ while retaining five drafts (Table A12). Reducing the candidate count from five to one lowers all nine scores by 6.2 percentage points on average. Increasing it to eight improves eight settings and adds a further 1.1 points on average. Raising each expert-capacity limit to twelve yields five gains, three declines, and one tie, with a 0.3-point mean gain. Multiple

Table A12: Candidate-count and expert-capacity scans: test-set scores (%).

Benchmark	Variant	Qwen3-32B	Qwen3.5-4B	Claude-Opus-4.6
ALFWorld	**Default** ($N = 5$, $K_{\max} = 7$)	**86.57**	**88.06**	**95.52**
	$N = 1$	79.10	82.84	93.28
	$N = 8$	87.31	89.55	97.01
	$K_{\max} = 12$	87.31	88.06	97.76
SearchQA	**Default** ($N = 5$, $K_{\max} = 6$)	**81.00**	**81.14**	**88.14**
	$N = 1$	72.00	74.29	82.71
	$N = 8$	81.64	81.71	89.86
	$K_{\max} = 12$	81.57	78.43	84.00
Spreadsheet	**Default** ($N = 5$, $K_{\max} = 4$)	**42.86**	**28.93**	**80.71**
	$N = 1$	36.43	22.14	73.93
	$N = 8$	44.29	28.57	81.79
	$K_{\max} = 12$	43.21	26.79	83.21

candidate drafts drive the main improvement, while the small and uneven return from more experts suggests that the benchmark-specific default capacities are sufficient for these tasks.

A3.10 SearchQA Performance Across Learning Budgets

To assess learning-cost performance on question answering, we evaluate the same five skill methods on SearchQA with Kimi-K2.6 and DeepSeek-V4-Flash. Figure A1 shows test-set exact match against cumulative learning tokens, with dashed No skill references.

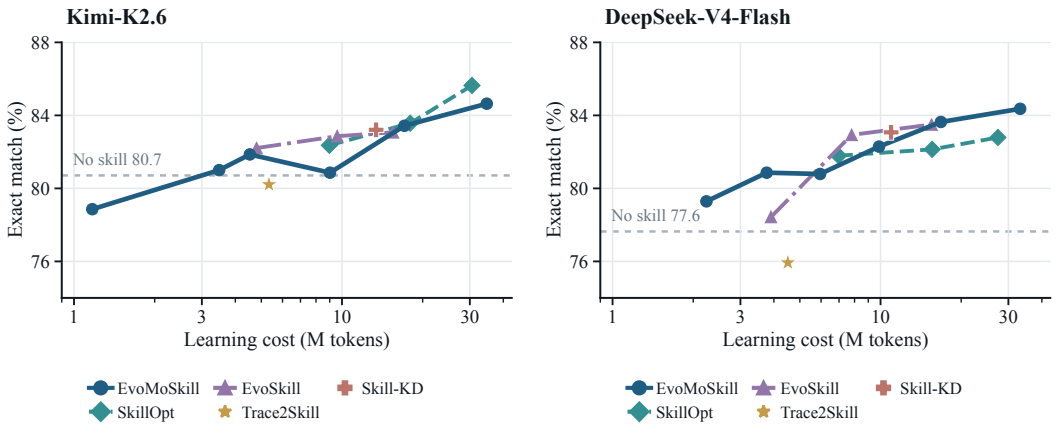

Figure A1: SearchQA exact match versus learning cost for Kimi-K2.6 and DeepSeek-V4-Flash.

Across both students, EvoMoSkill rises overall with more learning and ends above No skill. SkillOpt has the best final Kimi-K2.6 score, whereas EvoMoSkill leads on DeepSeek-V4-Flash. The curves show sustained gains from iterative skill optimization on SearchQA.