SKILLCOME: GROUP CONTRAST SKILL OPTIMIZATION WITH DUAL MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill evolution improves the capabilities of large language models by analyzing trajectories generated under a given skill and modifying the skill accordingly. Existing approaches typically generate a single trajectory per question. However, this provides insufficient optimization signals since it requires inferring effective skill edits from a solitary path. It is difficult to pinpoint which actions caused the failure in a failed trajectory, or to determine which actions in a successful one should be incorporated into the skill. Furthermore, they rely on a local batch of trajectories for analysis, making the optimization direction susceptible to noisy evidence. To address these, we propose SkillCome, a Skill-evolution method based on group Contrast optimization with dual memory. For each question, SkillCome generates n trajectories and performs group contrast analysis to precisely identify key behavioral divergences between successful and failed trajectories, offering reliable optimization signals. The dual memory system further accumulates evidence from historical steps to track patterns shared across different groups, leading to more generalized optimization directions. Together, SkillCome builds a systematic optimization process that transforms experience from observed successful trajectories into reusable skills. Extensive experiments on six benchmarks spanning question answering, reasoning, and agentic tasks demonstrate the effectiveness of our method. SkillCome consistently outperforms baselines across five models of varying families and scales, with gains up to **+5.69%** points.

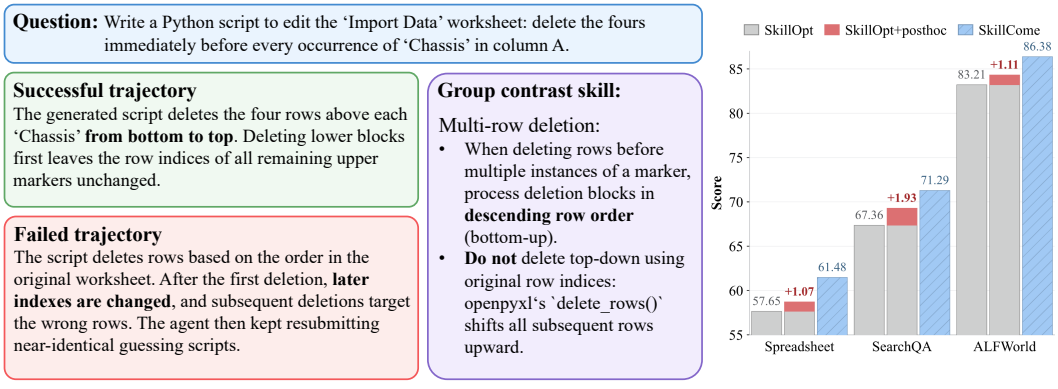


Figure 1: **Left**: An example of a skill learned by SkillCome on SpreadsheetBench. Through group contrast analysis, SkillCome learns to delete rows in descending order to avoid index shifts invalidating subsequent deletions, a skill absent from the single-rollout methods. **Right**: Post-hoc experiment on Qwen3.6-35B-A3B. After SkillOpt training, we perform one round of group contrast analysis. This single post-hoc skill edit can already improve performance on three datasets.

1 INTRODUCTION

Large language models (LLMs) can be improved through parametric optimization of model weights or non-parametric optimization of external context (Ouyang et al., 2022; Wang et al., 2023a). Skills (Xu & Yan, 2026; Li et al., 2026), typically derived from human experience or LLM-generated

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.

trajectories (Jiang et al., 2026; Bai et al., 2026), provide an effective form of non-parametric optimization. Recent works on skill evolution further optimize skills iteratively (Wang et al., 2025; Lin et al., 2026). Specifically, a target model executes tasks under the current skill, and an optimizer model analyzes the resulting trajectories to propose skill revisions, which are then evaluated on validation sets (Zhang et al., 2026; Zhu et al., 2026).

Despite its effectiveness, skill evolution requires more than an iterative loop of generating and evaluating edits. A sound optimization process requires two key components: a reliable optimization signal that indicates how the current solution should be updated, and an optimizer to organize the signals and guide update directions. Parametric optimization has developed sophisticated mechanisms for both training objectives (Wei et al., 2022a; Rafailov et al., 2023) and optimizer states (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). This motivates us to examine both the construction of optimization signals and the design of optimizer state for skill evolution.

At the optimization signal level, the outcome of a trajectory does not directly reveal how the skill should change. Existing approaches typically analyze one trajectory per question (Alzubi et al., 2026; Wu et al., 2026), leaving it unclear which actions caused failure or contributed to success. At the optimizer level, current methods primarily observe trajectories from a local batch, which can provide noisy or incomplete evidence. A valuable skill pattern can appear in trajectories scattered across different steps. Consequently, a single batch lacks sufficient data to substantiate this recurring pattern, leaving the optimizer to overlook it for inclusion in the skill.

These limitations call for an optimization process with reliable learning signals and a stateful optimizer. In parametric optimization, group relative rewards among responses to the same question have been widely adopted as optimization signals (Shao et al., 2024). For skill evolution, comparing successful and failed trajectories within a group similarly reveals behavioral divergences associated with opposite outcomes. The optimizer can then derive reliable optimization signals from observed successful attempts, rather than inferring a potential solution solely from a failed trajectory. Figure 1 illustrates how same-question contrast reveals a row-deletion error in worksheet-operation tasks and yields a practical skill. We further examine this idea by performing a single post-hoc group contrast skill revision using existing SkillOpt trajectories for the same question. As shown on the right of Figure 1, this already improves performance without additional training rollouts. For such evidence to guide optimization direction, a memory system is necessary to preserve historical observations. Analogous to momentum in gradient descent (Sutskever et al., 2013), the memory can retain patterns learned from previous steps. The optimizer can then combine current findings with accumulated information, continually calibrating its update direction as further evidence becomes available. Such a stateful optimization process helps turn scattered evidence into generalizable skills, preventing noise in individual steps from dominating the optimization process.

In this work, we propose SkillCome, a skill-evolution method that provides more reliable optimization signals while guiding optimization directions with historical evidence. Our method reshapes trajectory generation by introducing grouped rollouts, collecting multiple trajectories for each question. We incorporate within-group contrast between successful and failed trajectories to conventional reflection. These comparisons help identify key behavioral differences and precisely transform trajectory evidence into targeted skill revisions, improving optimization reliability and data efficiency. We further introduce a dual-memory system that connects current observations with historical information. Contrast memory preserves contrast patterns learned between successful and failed trajectories within the same group, while failure memory retains failure patterns shared by different groups. By continuously interacting with the dual memory, the optimizer can discover shared patterns across questions and use them to refine the skill update directions. The two synergistic components establish a comprehensive optimization process for skill evolution. Experiments across diverse model-benchmark pairs demonstrate that SkillCome can discover effective skills that are difficult to learn for single-rollout methods, leading to improved skill evolution and performance. In summary, our contributions are threefold:

- We introduce group rollouts and within-group contrast optimization under the same skill. This provides reliable optimization signals by pinpointing the key divergences between successful and failed trajectories, grounding skill revisions in observed experience.
- To calibrate the direction of skill optimization, we propose a dual-memory system storing historical evidence. The system recognizes recurring patterns from different steps and builds support for generalizable skills, preventing noisy updates from individual batches.

- Extensive experiments on six benchmarks validate the effectiveness of our method across five models with varying scales. SkillCome enables effective skill evolution and delivers consistent improvements, with average gains up to **19.55%** compared to no skill.

2 Related Work

From Skills to Skill Evolution. Prompt-based methods (Wei et al., 2022b; Yao et al., 2023; Pryzant et al., 2023) have demonstrated that LLMs can be improved by simply refining the textual instructions (Wang et al., 2023b; Liu et al., 2023). Beyond general prompts, agent skills further include task-specific experience, including tool-use strategies and relevant references (Zheng et al., 2025; Shi et al., 2026). They organize experience derived from human expertise or agent trajectories into natural language, optionally accompanied by executable scripts and supporting resources (Shen et al., 2026a). By making such experience editable, skills can also incorporate new lessons from execution feedback (Yang et al., 2026b). This has motivated various research on skill evolution, where agent trajectories are analyzed to iteratively revise skills and improve subsequent task performance (Liu et al., 2026a; Ouyang et al., 2026). They explicitly optimize skill artifacts through repeated generation and revision (Liu et al., 2026a; Ouyang et al., 2026). Trace2Skill (Ni et al., 2026) extracts local lessons from execution traces in parallel and consolidates them into transferable procedural guidance. SkillOpt (Yang et al., 2026a) develops a structured optimization procedure that converts trajectories into bounded skill edits, with edit aggregation and validation-based acceptance. SkillOpt-Lite (Shen et al., 2026b) and OEO (Liu et al., 2026b) delegate parts of the evolution process to an agent rather than a human-implemented pipeline, allowing the agent to automatically organize the evolution. These studies establish generated trajectories as a basis for skill evolution, with different designs for extracting experience and learning edits. While our method focuses on how to better construct the optimization process based on grouped rollouts.

3 Method

In this section, we first formulate the skill-evolution problem in § 3.1. We then describe grouped rollouts in § 3.2. § 3.3 presents our group contrast skill optimization procedure. The overall framework of SkillCome is illustrated in Figure 2.

3.1 Problem Formulation

In skill evolution, an external skill document is optimized to improve the performance of LLM agents on given tasks. A skill s consists of a natural-language document together with any accompanying executable scripts. The document is incorporated into the agent context during execution.

We distinguish two model roles: the target model M_{tar} executes tasks under the current skill, while the optimizer model M_{opt} analyzes the resulting trajectories and proposes skill revisions. The two roles can be the same or different LLMs, whose parameters remain frozen throughout skill evolution. Given an input task x and a skill document s, a trajectory is sampled as $\tau \sim M_{\mathrm{tar}}(\cdot \mid x, s)$. The trajectory τ records model outputs, tool interactions when applicable, and the final answer. A task-specific evaluator assigns a binary score $R(x, \tau) \in \{0, 1\}$ according to the task's success criterion. Given a dataset $\mathcal{D}$, the objective is to find a skill that maximizes expected performance:

$$\max_{s} \sum_{x \in \mathcal{D}} \mathbb{E}_{\tau \sim M_{\mathrm{tar}}(\cdot|x,s)} \left[R(x, \tau) \right]. \tag{1}$$

In practice, we use three disjoint data splits: $\mathcal{D}_{\mathrm{train}}$ for collecting training trajectories, $\mathcal{D}_{\mathrm{val}}$ for evaluating the updated skills, and $\mathcal{D}_{\mathrm{test}}$ for testing the final skill. Starting from an initial skill s^0, the target model executes training tasks under the current skill s^t at optimization step t. M_{opt} analyzes these trajectories and their evaluation feedback, then proposes and selects edits that add, remove, or replace content from s^t. Applying the selected edits yields a new candidate skill s^t_{cand}.

The candidate is evaluated on the validation split using the same target model and execution environment. We denote the measured validation score of a skill s by:

$$V(s) = \frac{1}{|\mathcal{D}_{\mathrm{val}}|} \sum_{x \in \mathcal{D}_{\mathrm{val}}} R(x, \tau_x(s)), \tag{2}$$

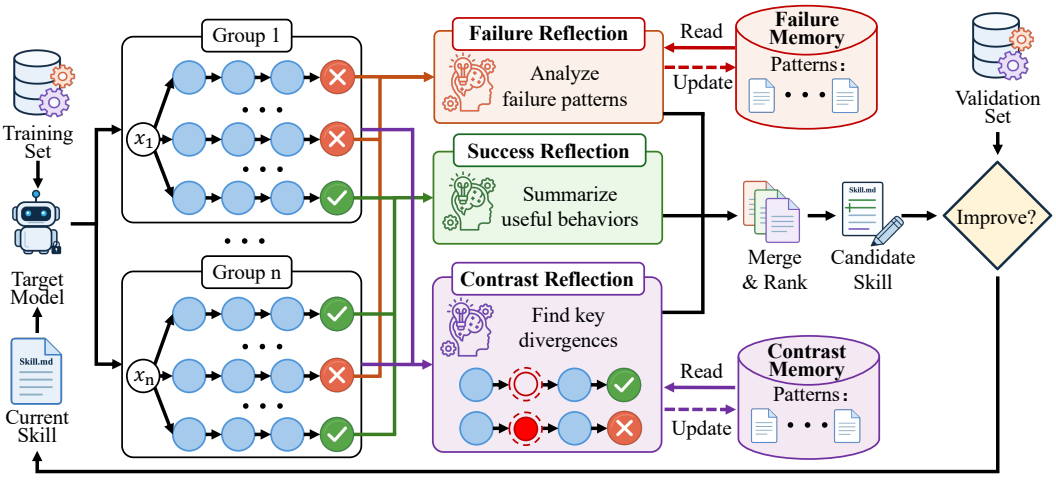


Figure 2: Overview of SkillCome. We first formulate the training as grouped rollouts. The generated trajectories are routed to either "Success Reflection" or "Failure Reflection" based on their outcome correctness. Groups with mixed outcomes undergo "Contrast Reflection". During reflection, the optimizer model reads and updates failure memory and contrast memory to find generalizable optimization directions. The updated skill is accepted if it improves the validation score.

where $\tau_x(s)$ denotes the validation trajectory generated for task x using skill s. The candidate skill s^t_{cand} is accepted only if its score exceeds the recorded validation score of the previous skill:

$$s^{t+1} = \begin{cases} s^t_{\text{cand}}, & V(s^t_{\text{cand}}) > V(s^t)\,, \\ s^t, & \text{otherwise.} \end{cases} \tag{3}$$

After iteratively running the training-updating-validation loop, we retain the skill with the highest validation score as $s^\star$ and evaluate it on $\mathcal{D}_{\text{test}}$.

3.2 Grouped Rollout

To obtain comparable trajectory evidence for each question, we organize the training process in the form of group rollouts. At optimization step t, we sample m distinct training questions $\{x^t_i\}_{i=1}^m \subseteq \mathcal{D}_{\text{train}}$. For each question, the target model M_{tar} independently generates n trajectories under the same current skill s^t:

$$\tau^t_{ij} \sim M_{\text{tar}}\big(\cdot \mid x^t_i, s^t\big)\,, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n, \tag{4}$$

where the subscripts i and j are the question index and its rollout index, respectively. The skill remains fixed throughout the current optimization step. Thus, the trajectories within a group represent alternative executions of the same question. Each trajectory receives a score $r^t_{ij} = R(x^t_i, \tau^t_{ij})$. We organize the scored trajectories for each question into a group $\mathcal{G}^t_i$ and denote the collection of all groups in the batch by $\mathcal{G}^t$:

$$\mathcal{G}^t_i = \big\{(\tau^t_{ij}, r^t_{ij})\big\}_{j=1}^n\,, \qquad \mathcal{G}^t = \{\mathcal{G}^t_i\}_{i=1}^m\,. \tag{5}$$

The batch contains m groups and $B = mn$ rollouts in total.

For each group, let $\mathcal{G}^{t,+}_i$ and $\mathcal{G}^{t,-}_i$ denote the subsets of successful trajectories with $r^t_{ij} = 1$ and failed trajectories with $r^t_{ij} = 0$, respectively. Across the batch, we collect these trajectories into success and failure pools:

$$\mathcal{T}^{t,+} = \bigcup_{i=1}^m \mathcal{G}^{t,+}_i, \qquad \mathcal{T}^{t,-} = \bigcup_{i=1}^m \mathcal{G}^{t,-}_i. \tag{6}$$

These pools provide evidence for success reflection and failure reflection, respectively. And we collect groups containing both successful and failed trajectories into the mixed-group subset:

$$\mathcal{G}^t_{\text{mix}} = \big\{\mathcal{G}^t_i \in \mathcal{G}^t \;\big|\; \mathcal{G}^{t,+}_i \neq \varnothing \text{ and } \mathcal{G}^{t,-}_i \neq \varnothing\big\}\,. \tag{7}$$

Each mixed group is passed to the subsequent group contrast reflection, where successful trajectories serve as concrete references for examining failures and guiding skill edits.

3.3 Group Contrast Skill Optimization

3.3.1 Dual Memory System

Since the optimizer M_{opt} does not have an infinite context length, each optimization step can analyze only a limited set of trajectories. Skill edits derived from one step may overfit to a particular question, while evidence supporting a generalizable pattern may be distributed across multiple steps. To connect these observations, we maintain a failure memory $\mathcal{M}_{\text{f}}$ and a contrast memory $\mathcal{M}_{\text{c}}$, both initialized as empty. $\mathcal{M}_{\text{f}}$ preserves failure patterns extracted through conventional reflection. $\mathcal{M}_{\text{c}}$ contains contrastive patterns learned from the behavioral differences between successful and failed trajectories within the same group. We keep $\mathcal{M}_{\text{f}}$ and $\mathcal{M}_{\text{c}}$ separate to distinguish failure-only patterns from contrast patterns that provide more reliable optimization signals grounded in observed successful alternatives.

Rather than storing lengthy historical trajectories, each memory consists of concise patterns:

$$\mathcal{M} = \{(p_k, u_k)\}_{k=1}^{K}, \tag{8}$$

where K is the number of learned patterns, p_k is the natural-language description of the k-th pattern, and u_k denotes the set of distinct training groups supporting it. The size of u_k indicates how many different questions support the pattern, rather than how many trajectories. Patterns supported by more questions provide stronger evidence for broadly applicable skill edits.

During trajectory analysis, M_{opt} interacts with the corresponding memory. It compares the current trajectories with the memory and proposes memory updates alongside its skill edits. An update may introduce a new pattern, refine an existing description, or associate additional supporting groups with an existing pattern without changing its description. For the latter two cases, the support set is updated as $u_k \leftarrow u_k \cup u_{\text{new}}$, where u_{new} contains the newly associated group identifiers. We denote the proposed memory updates by $\Delta\mathcal{M}$, which may be empty when no update is needed. This read-and-update process allows a pattern initially observed in one step to accumulate evidence from different groups across optimization steps, without requiring additional training rollouts.

3.3.2 Conventional Failure and Success Reflection

At optimization step t, the trajectory pools $\mathcal{T}^{t,-}$ and $\mathcal{T}^{t,+}$ obtained from §3.2 are analyzed separately by M_{opt} to propose skill edits. Failure reflection examines what went wrong and how the skill could prevent similar failures, whereas success reflection extracts useful practices worth retaining. Taking failure reflection as an example, the optimizer uses the current skill, failed trajectories, and failure memory to generate candidate skill edits and optional memory updates:

$$\left(\mathcal{P}_{\text{f}}^{t}, \Delta\mathcal{M}_{\text{f}}^{t}\right) \leftarrow M_{\text{opt}}\left(s^{t}, \mathcal{T}^{t,-}, \mathcal{M}_{\text{f}}\right), \tag{9}$$

where $\mathcal{P}_{\text{f}}^{t}$ contains the proposed skill edits. Historical failure patterns from $\mathcal{M}_{\text{f}}$ help M_{opt} relate current mistakes to problems in previous steps and formulate edits supported by different questions. Success reflection is performed through separate optimizer calls and similarly produces candidate edits $\mathcal{P}_{\text{s}}^{t}$. Note that trajectories from the mixed groups still contribute to conventional reflection, while they are additionally examined through group contrast reflection described next.

3.3.3 Group Contrast Reflection

To accurately determine what to correct and what to retain in the skill, we introduce the group contrast reflection. It compares successful and failed trajectories within the same group to track critical divergences that account for their opposite outcomes. For each mixed group $\mathcal{G}_i^t \in \mathcal{G}_{\text{mix}}^t$, a separate optimizer call generates candidate edits and optional updates to the contrast memory:

$$\left(\mathcal{P}_{\text{c},i}^{t}, \Delta\mathcal{M}_{\text{c},i}^{t}\right) \leftarrow M_{\text{opt}}\left(s^{t}, \mathcal{G}_i^{t,+}, \mathcal{G}_i^{t,-}, \mathcal{M}_{\text{c}}\right). \tag{10}$$

Here, M_{opt} is instructed to first locate critical divergences between the successful and failed trajectories. It then examines which actions in the successful trajectories were omitted or executed

incorrectly in the failures, and relates these differences to missing or insufficiently actionable guidance in s^t. Successful trajectories thus provide concrete references that the target model has actually done, offering reliable optimization signals for skill editing.

The contrast memory $\mathcal{M}_{\mathrm{c}}$ further connects these same-question comparisons with patterns observed in previous groups. A new comparison can then strengthen or refine an existing pattern instead of being interpreted as an isolated case. This guides skill optimization toward generalizable directions supported by multiple questions, rather than fitting a single case. And the proposed memory updates are integrated after each analysis, making them visible to subsequent steps.

3.3.4 Evidence-Guided Skill Revision

We collect the contrast edits as $\mathcal{P}_{\mathrm{c}}^t = \bigcup_{\mathcal{G}_i^t \in \mathcal{G}_{\mathrm{mix}}^t} \mathcal{P}_{\mathrm{c},i}^t$ and combine them with the failure and success edits. Through dedicated merge calls of the optimizer M_{opt}, we consolidate overlapping edits and resolves conflicting suggestions:

$$\mathcal{P}^t = \mathrm{Merge}_{M_{\mathrm{opt}}}\left(\mathcal{P}_{\mathrm{f}}^t \cup \mathcal{P}_{\mathrm{s}}^t \cup \mathcal{P}_{\mathrm{c}}^t\right). \tag{11}$$

When the number of edits in $\mathcal{P}^t$ exceeds the given edit budget L^t, M_{opt} will select at most L^t edits. Ranking prioritizes contrast edits and failure edits over success edits. It also accounts for the number of supporting groups to find more generalizable skill edits.

Based on the above procedure, the optimizer integrates complementary evidence: Conventional reflection captures shared difficulties and potentially useful practices; group contrast reflection provides practical guidance grounded in grouped rollouts. In addition, the dual memory system further extends the supporting evidence across steps. Applying the selected final edits $\Delta^t \subseteq \mathcal{P}^t$ produces the candidate skill:

$$s_{\mathrm{cand}}^t = \mathrm{Apply}(s^t, \Delta^t), \qquad |\Delta^t| \le L^t. \tag{12}$$

The candidate skill s_{cand}^t is evaluated and accepted according to Eq. 3, after which the target model M_{tar} collects the next batch of grouped rollouts under the resulting skill. Memory updates are retained even when a candidate is rejected. Since an unsuccessful edit does not invalidate the underlying observations, which may be supported by different groups of rollouts in future steps. Across iterations, group contrast deepens the analysis of individual questions and provides accurate optimization signals. While the dual-memory system plays a momentum-like role to regulate optimization direction by retaining the patterns observed beyond the current batch. For final evaluation, we test the target model on $\mathcal{D}_{\mathrm{test}}$ using the selected best skill $s^\star$.

4 Experiments

4.1 Experimental Setup

Datasets. The experiments are conducted on six benchmarks. *General question answering* tasks include DocVQA (Mathew et al., 2021) and SearchQA (Dunn et al., 2017). *Mathematical reasoning* task is LiveMathematicianBench (LiveMath; He et al. (2026)). *Agentic tasks* include Spreadsheet-Bench (Ma et al., 2024), OfficeQA (Opsahl-Ong et al., 2026), and ALFWorld (Shridhar et al., 2021). We adopt the data splits from SkillOpt-Lite (Shen et al., 2026b).

Models. We employ GLM-5.3 (Zeng et al., 2026), Qwen3.8-Flash-Next (Qwen Team, 2026a), DeepSeek-V4-Pro-0813 (DeepSeek-AI, 2026), DeepSeek-V4-Flash, and Qwen3.6-35B-A3B (Qwen Team, 2026b) as the target model. For the first three models, we use themselves as the optimizer for a self-evolving setting. For the less capable two models, DeepSeek-V4-Pro is employed as the optimizer to evaluate a setting where a stronger model guides a weaker model.

Settings. We compare our method against the following baselines: running without a skill, the initial skill written by human, skill-evolving methods Trace2Skil (Ni et al., 2026), SkillOpt (Yang et al., 2026a), and SkillOpt-Lite (Shen et al., 2026b). And we follow the training protocol of SkillOpt-Lite. The default number of rollouts n is set to 8. Since all test sets other than SearchQA are on the scale of hundreds of samples, resulting in significant testing volatility, we evaluate these benchmarks over **four independent runs** and report the average accuracy. More details are provided in Appendix B.

Table 1: Performance across six benchmarks and five models. "–" denotes unsupported VQA evaluation for text-only LLMs. Subscripts in green/red represent the change relative to "No skill". The best and second-best scores within each model are in **bold** and underlined, respectively.

Model	Skill	Spreadsheet	SearchQA	LiveMath	ALFWorld	OfficeQA	DocVQA	Avg
Qwen3.8 Flash-Next	No skill	48.58	67.50	33.73	81.53	23.48	90.37	57.53
	Init skill	45.55$_{-3.03}$	67.97$_{+0.47}$	35.14$_{+1.41}$	82.47$_{+0.94}$	30.75$_{+7.27}$	91.38$_{+1.01}$	58.88$_{+1.35}$
	Trace2Skill	69.66$_{+21.08}$	69.93$_{+2.43}$	31.60$_{-2.13}$	85.07$_{+3.54}$	49.83$_{+26.35}$	91.31$_{+0.94}$	66.23$_{+8.70}$
	SkillOpt-Lite	73.75$_{+25.17}$	72.43$_{+4.93}$	41.75$_{+8.02}$	86.19$_{+4.66}$	60.30$_{+36.82}$	91.71$_{+1.34}$	71.02$_{+13.49}$
	SkillOpt	72.51$_{+23.93}$	72.57$_{+5.07}$	49.53$_{+15.80}$	84.70$_{+3.17}$	59.80$_{+36.32}$	91.24$_{+0.87}$	71.73$_{+14.20}$
	SkillCome	**75.71**$_{+27.13}$	**74.14**$_{+6.64}$	**63.68**$_{+29.95}$	**91.79**$_{+10.26}$	**63.68**$_{+40.20}$	**92.78**$_{+2.41}$	**76.96**$_{+19.43}$
DeepSeek-V4 Pro-0813	No skill	47.78	66.57	28.07	66.98	53.71	–	52.62
	Init skill	48.40$_{+0.62}$	67.07$_{+0.50}$	31.37$_{+3.30}$	79.10$_{+12.12}$	54.56$_{+0.85}$	–	56.10$_{+3.48}$
	Trace2Skill	54.98$_{+7.20}$	69.21$_{+2.64}$	33.25$_{+5.18}$	80.04$_{+13.06}$	46.45$_{-7.26}$	–	56.79$_{+4.16}$
	SkillOpt-Lite	59.08$_{+11.30}$	69.29$_{+2.72}$	32.08$_{+4.01}$	80.60$_{+13.62}$	54.73$_{+1.02}$	–	59.16$_{+6.53}$
	SkillOpt	57.38$_{+9.60}$	70.86$_{+4.29}$	48.35$_{+20.28}$	84.32$_{+17.34}$	56.76$_{+3.05}$	–	63.53$_{+10.91}$
	SkillCome	**62.10**$_{+14.32}$	**72.50**$_{+5.93}$	**53.30**$_{+25.23}$	**89.18**$_{+22.20}$	**59.29**$_{+5.58}$	–	**67.27**$_{+14.65}$
GLM-5.3	No skill	45.73	67.93	36.08	86.94	20.78	–	51.49
	Init skill	52.44$_{+6.71}$	68.57$_{+0.64}$	37.03$_{+0.95}$	91.23$_{+4.29}$	27.84$_{+7.06}$	–	55.42$_{+3.93}$
	Trace2Skill	43.15$_{-2.58}$	70.50$_{+2.57}$	36.79$_{+0.71}$	88.81$_{+1.87}$	27.03$_{+6.25}$	–	53.26$_{+1.76}$
	SkillOpt-Lite	68.23$_{+22.50}$	**77.43**$_{+9.50}$	65.09$_{+29.01}$	88.43$_{+1.49}$	46.62$_{+25.84}$	–	69.16$_{+17.67}$
	SkillOpt	66.90$_{+21.17}$	74.29$_{+6.36}$	45.05$_{+8.97}$	92.35$_{+5.41}$	46.28$_{+25.50}$	–	64.97$_{+13.48}$
	SkillCome	**71.53**$_{+25.80}$	77.07$_{+9.14}$	**67.92**$_{+31.84}$	**95.90**$_{+8.96}$	**48.65**$_{+27.87}$	–	**72.21**$_{+20.72}$
DeepSeek-V4 Flash	No skill	47.33	64.85	31.60	59.51	5.41	–	41.74
	Init skill	48.58$_{+1.25}$	65.43$_{+0.58}$	35.38$_{+3.78}$	56.91$_{-2.60}$	11.66$_{+6.25}$	–	43.59$_{+1.85}$
	Trace2Skill	51.07$_{+3.74}$	71.50$_{+6.65}$	33.25$_{+1.65}$	79.10$_{+19.59}$	12.84$_{+7.43}$	–	49.55$_{+7.81}$
	SkillOpt-Lite	57.38$_{+10.05}$	68.00$_{+3.15}$	33.73$_{+2.13}$	61.01$_{+1.50}$	24.66$_{+19.25}$	–	48.96$_{+7.22}$
	SkillOpt	59.16$_{+11.83}$	71.57$_{+6.72}$	68.63$_{+37.03}$	74.44$_{+14.93}$	26.01$_{+20.60}$	–	59.96$_{+18.22}$
	SkillCome	**62.54**$_{+15.21}$	**73.36**$_{+8.51}$	**74.29**$_{+42.69}$	**84.14**$_{+24.63}$	**50.51**$_{+45.10}$	–	**68.97**$_{+27.23}$
Qwen3.6 35B-A3B	No skill	41.99	66.00	26.65	66.04	37.16	89.77	54.60
	Init skill	40.75$_{-1.24}$	66.23$_{+0.23}$	30.19$_{+3.54}$	79.48$_{+13.44}$	38.68$_{+1.52}$	90.17$_{+0.40}$	57.58$_{+2.98}$
	Trace2Skill	44.93$_{+2.94}$	69.21$_{+3.21}$	31.60$_{+4.95}$	75.56$_{+9.52}$	35.64$_{-1.52}$	89.10$_{-0.67}$	57.67$_{+3.07}$
	SkillOpt-Lite	56.93$_{+14.94}$	67.57$_{+1.57}$	29.01$_{+2.36}$	81.34$_{+15.30}$	37.84$_{+0.68}$	90.11$_{+0.34}$	60.47$_{+5.87}$
	SkillOpt	57.65$_{+15.66}$	67.36$_{+1.36}$	69.81$_{+43.16}$	83.21$_{+17.17}$	34.63$_{-2.53}$	89.77$_{+0.00}$	67.07$_{+12.47}$
	SkillCome	**61.48**$_{+19.49}$	**71.29**$_{+5.29}$	**71.46**$_{+44.81}$	**86.38**$_{+20.34}$	**40.03**$_{+2.87}$	**91.24**$_{+1.47}$	**70.31**$_{+15.71}$

4.2 Main Results

As presented in Table 1, SkillCome achieves the highest scores across 26 model–benchmark pairs, outperforming baselines by a large margin. Compared with the second-best method SkillOpt, our method further improves average accuracy by 5.69 points, demonstrating the benefits of learning from grouped trajectories. On both the self-evolving setting and strong-guide-weak setting, SkillCome exhibits consistent gains across LLMs from 35B to 1.6T, with average gains up to 27.23 over "No skill". Notably, the OfficeQA performance of DeepSeek-V4-Flash improves from 5.41% to 50.51% with our method, surpassing the second-best method by 24.50%. This is associated with a recurring format error: the model frequently uses <|DSML|answer>...</|DSML|answer> delimiters instead of the required <answer>...</answer> tags, preventing its responses from being recognized as valid answers. SkillCome retains this repeated pattern in memory and incorporates explicit instructions prohibiting DSML-style output into the skill. This provides a concrete example of how cross-step memory supports the correction of persistent model-specific errors. For more detailed analysis, please refer to Appendix C.1.

4.3 Ablation Study

To investigate the impact of each component of SkillCome, we conduct two ablations. "SkillOpt w/ Memory" augments the strongest baseline, SkillOpt, with the failure memory in our framework. We also evaluate "SkillCome w/o Memory", which removes dual memory while retaining group contrast analysis. As shown in Table 2, compared with SkillOpt, "SkillOpt w/ Memory" improves performance in 8 of the 11 model–benchmark pairs and reduces in the remaining three. These mixed results suggest that adding memory to conventional single-rollout optimization yields gains, but not consistently. In contrast, removing memory from SkillCome decreases performance in all pairs. Despite this, "SkillCome w/o Memory" still exhibits improvement over Skillopt. The contrasting effects of memory can be attributed to differences in the evidence within each update. Under a comparable budget, single-rollout methods observe more distinct questions per step, potentially lim-

Table 2: Ablation experiment of method components on two models.

Model	Method	Spreadsheet	SearchQA	LiveMath	ALFWorld	OfficeQA	DocVQA	Avg
DeepSeek-V4 Flash	SkillOpt	59.16	71.57	68.63	74.44	26.01	–	59.96
	SkillOpt w/ Memory	59.43	71.14	68.86	77.80	47.80	–	65.01
	SkillCome w/o Memory	57.92	71.43	63.92	75.37	37.84	–	61.30
	SkillCome	**62.54**	**73.36**	**74.29**	**84.14**	**50.51**	–	**68.97**
Qwen3.6 35B-A3B	SkillOpt	57.65	67.36	69.81	83.21	34.63	89.77	67.07
	SkillOpt w/ Memory	58.63	69.07	67.92	81.16	35.14	89.91	66.97
	SkillCome w/o Memory	58.72	68.57	70.99	83.40	34.46	89.84	67.66
	SkillCome	**61.48**	**71.29**	**71.46**	**86.38**	**40.03**	**91.24**	**70.31**

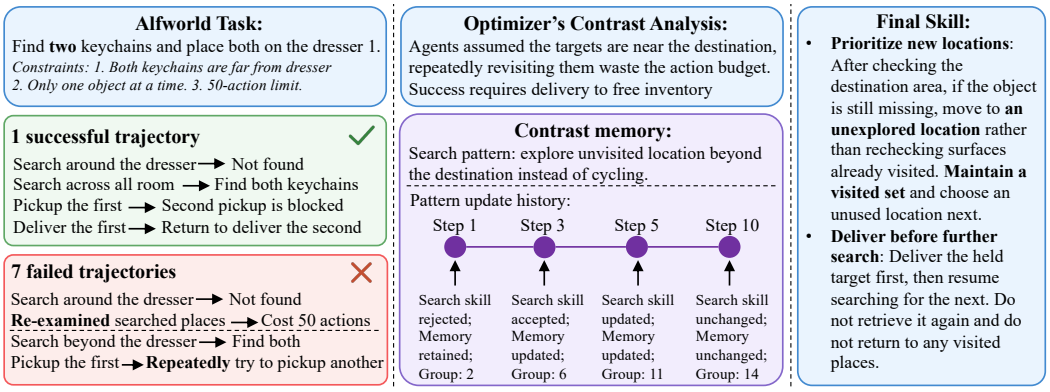


Figure 3: A case study of the evolution process of SkillCome on ALFWorld, where the agent needs to manipulate objects within a simulated embodied environment. SkillCome discovers a valuable pattern. Although the initial candidate skill is rejected, the extracted pattern remains in contrast memory. Its support group accumulates from 2 to 14 distinct tasks across optimization steps. During the process, the skill is accepted and refined multiple times.

iting the benefit of memory. Grouped rollouts instead concentrate on repeated trajectories of fewer questions, making cross-step evidence particularly useful for calibrating the optimization direction. Together, the ablations show that the two components of SkillCome are effective and synergistic.

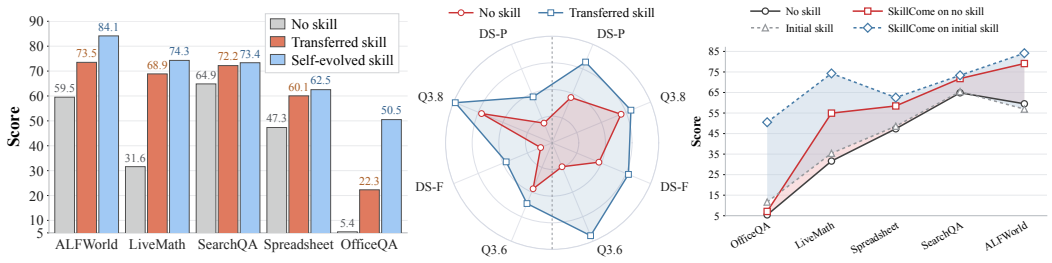
5 Further Analysis

5.1 Case Study

Figure 3 showcases a detailed evolution process of SkillCome. As illustrated, group contrast reveals a useful search pattern: explore unvisited locations beyond the destination rather than repeatedly checking the same area. With the accumulation of supporting groups in the contrast memory, the corresponding skill edit is selected and accepted at step 3. At step 5, a further revision introduces a complementary rule: deliver the held target before searching for the next. The resulting skill includes practical guidance absent from single-rollout baselines and leads to stronger test performance.

5.2 Skill Transferability

Cross-Model Transferability. To evaluate whether the learned skills remain useful beyond the target model, we transfer the skills between DeepSeek-V4-Flash (Flash) and Qwen3.6-35B-A3B (Qwen3.6). For each benchmark, a skill optimized for Qwen3.6 is directly applied to Flash. Figure 4a shows that transferred skills significantly outperform the initial skill in all five benchmarks. However, they still exhibit gaps from self-evolved skills, with the largest differences on OfficeQA. This is consistent with previous discussion in §4.2, since the skill learned on Qwen3.6 does not contain the DSML-style format issue. Overall, these results indicate that our skills contain both broadly reusable procedures and corrections tailored to the behavioral patterns of individual models.



(a) (b) (c)

Figure 4: (a) Cross-model skill transferability on DeepSeek-V4-Flash. (b) Cross-dataset skill transferability on four models (Names are abbreviated). Left are the HotpotQA results, while the right are the Omni-MATH results. (c) Evolution from empty skill on DeepSeek-V4-Flash.

Cross-Dataset Transferability. We further evaluate the cross-dataset transferability of SkillCome on four models. Specifically, we directly transfer skills learned on SearchQA to HotpotQA (Yang et al., 2018) and those learned on OlympiadBench (He et al., 2024) to Omni-MATH (Gao et al., 2024). As shown in Figure 4b, the transferred skills outperform the no-skill baseline on both benchmarks across all four models, demonstrating their generalizability to new datasets with similar task requirements. More results on transferability are in Appendix C.6 and C.7.

5.3 Evolving from Empty Skill

Since initial skills do not always improve performance, we investigate whether evolving from an empty skill provides a better alternative. We compare evolving from no skill with evolving from initial skills on DeepSeek-V4-Flash. As presented in Figure 4c, evolution from an empty skill improves over "No skill" on all five benchmarks, but evolving from the initial skill achieves higher final performance. This shows that SkillCome can learn useful skills from scratch, while predefined guidance generally provides better starting points, even when their immediate benefits are limited.

5.4 Evaluation in Agent Harnesses

To further evaluate our method in agent harnesses, we apply SkillCome on Qwen3.8-Flash-Next and DeepSeek-V4-Pro-0813 within the Codex CLI harness on three datasets. As illustrated in Figure 5, we compare three settings: *Initial skill*, where the agent operates through the chat completion mode under the initial skill; *Direct chat*, where the agent operates through the chat completion mode under the skill learned by SkillCome; and *Codex*, where the agent operates within the Codex CLI harness under the skill learned by SkillCome. For both models, running under Codex consistently outperforms running under direct chat across all datasets. These results demonstrate the effectiveness of our method within more advanced and complex harness environments, highlighting its practical value in real-world workflows.

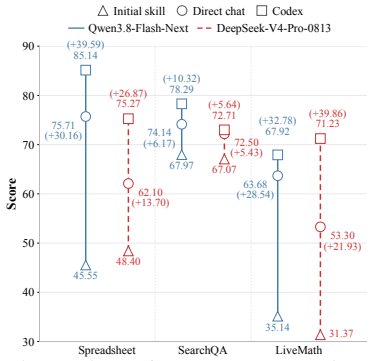


Figure 5: Performance comparison between Codex and Direct chat.

6 Conclusion

In this paper, we identify that existing skill evolution methods lack not only analysis of different trajectories for the same question, but also historical information across training steps. To mitigate these issues, we propose SkillCome, a skill-evolution method based on group contrast analysis and a dual-memory system. Combining the two synergistic components, SkillCome derives precise optimization signals and generalizable optimization directions, learning effective and reusable skills. Comprehensive evaluation across diverse models and benchmarks demonstrates that SkillCome substantially improves agent performance, highlighting its potential for various real-world applications.

7 AI USE STATEMENT

In this work, we used generative AI tools for proofreading and polishing the paper. For example, we use LLMs to check the grammar and improve the readability. We also used LLMs for literature discovery. Specifically, they suggested lists of potentially relevant papers, which the authors then independently located the original publications. The citations and descriptions of cited work were prepared by the authors. All AI-generated content was manually reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

8 REPRODUCIBILITY STATEMENT

In this section, we list any related materials that help to reproduce this paper..

1. **Datasets**: The training and evaluation dataset we used is described in §4.1. The complete description of each dataset and the processing steps are provided in Appendix B.1.
2. **Implementation Details**: The implementation details, such as training&evaluation settings and hyperparameters are described in §4.1. And their details are provided in Appendix B.2.
3. **Code**: The code to reproduce our algorithm is provided in the supplementary materials.

REFERENCES

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

Tong Bai, Zheng-Lin Wan, Pengfei Zhou, Xingrui Yu, Wang-Bo Zhao, Yang You, and Ivor W. Tsang. Skilldag: Self-evolving typed skill graphs for llm skill selection at scale. *ArXiv*, abs/2606.03056, 2026. URL https://api.semanticscholar.org/CorpusID:288863363.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 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.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models, 2024. URL https://arxiv.org/abs/2410.07985.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Linyang He, Qiyao Yu, Hanze Dong, Baohao Liao, Xinxing Xu, Micah Goldblum, Jiang Bian, and Nima Mesgarani. Livemathematicianbench: A live benchmark for mathematician-level reasoning with proof sketches, 2026. URL https://arxiv.org/abs/2604.01754.

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*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

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, Junwei He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Ze Ma, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Kaixin Li, Yuchen Tian, Shutong Wu, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Huawei Lin, Peng Li, Jie Song, Fu-Xin Jiang, and Tieying Zhang. Muse-autoskill: Self-evolving agents via skill creation, memory, management, and evaluation. *ArXiv*, abs/2605.27366, 2026. URL https://api.semanticscholar.org/CorpusID:288671734.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

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*, 2026a.

Yuxuan Liu, Zhaochen Su, Yuhao Zhang, Jiahe Guo, Zhongwei Xie, Huihao Jing, Lingyun Xie, Qing Zong, Yauwai Yim, Zhixiong Zhang, Haoran Li, and Yangqiu Song. Rethinking self-evolving agent skills: Feedback dynamics over multiple rounds. *arXiv preprint arXiv:2608.02636*, 2026b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 2200–2209, 2021.

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.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. Officeqa pro: An enterprise benchmark for end-to-end grounded reasoning. *arXiv preprint arXiv:2603.08655*, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7957–7968, 2023.

Qwen Team. On the design of Qwen3.8-Next architecture: Evaluation, efficiency, and training stability. Technical report, Alibaba Group, August 2026a.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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*, 2026a.

Yifei Shen, Bo Li, and Xinjie Zhang. Skillopt-lite: Better and faster agent self-evolution via one line of vibe. *arXiv preprint arXiv:2607.03451*, 2026b.

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*, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

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*, 2023a.

Jiongxiao Wang, Qiaojing Yan, Ya-Wei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Pan-Pan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *ArXiv*, abs/2512.17102, 2025. URL https://api.semanticscholar.org/CorpusID:284058483.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022b. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Xiyang Wu, Zongxia Li, Guangyao Shi, Alexander Duffy, Tyler Marques, Matthew Lyle Olson, Tianyi Zhou, and Dinesh Manocha. Co-evolving llm decision and skill bank agents for long-horizon tasks. *ArXiv*, abs/2604.20987, 2026. URL https://api.semanticscholar.org/CorpusID:287702239.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. *arXiv preprint arXiv:2602.12430*, 2026.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. 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, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution. *arXiv preprint arXiv:2603.01145*, 2026b.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

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

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gao-Wen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *ArXiv*, abs/2504.07079, 2025. URL https://api.semanticscholar.org/CorpusID:277634081.

Jiayin Zhu, Kelong Mao, Yudong Guo, Deng-Bo He, Sulong Xu, Simiu Gu, and Yutao Yue. Skillcoach: Self-evolving rubrics for evaluating and enhancing agentic skill-use. *ArXiv*, abs/2607.01874, 2026. URL https://api.semanticscholar.org/CorpusID:289749220.

APPENDIX

A Method Details **15**

B Detailed Experimental Setting **17**

B.1 Dataset Details 17

B.2 Implementation Details 18

C Additional Experiments **18**

C.1 Notable Findings in the Experiment 18

C.2 Standard Deviation of Main Results 20

C.3 Statistics of the Mix Group 20

C.4 Ablation on the Components of SkillCome 20

C.5 Evaluation in Agent Harnesses 21

C.6 Cross-Model Transferability 21

C.7 Cross-Dataset Transferability 21

C.8 Evolving from Empty Skill 22

C.9 Token Usage of Optimizer Model 22

A METHOD DETAILS

Prompt A.1: Prompt for Group Contrast Reflection

```
You are an expert same-task contrast analyst for AI agent trajectories.

You receive several valid rollouts of the EXACT SAME task under the same skill.  At least
one rollout succeeded and at least one failed.  Analyze them jointly; do not treat them as
unrelated examples.

Procedure:

     1. Compare successful and failed trajectories and locate the earliest meaningful
        behavioral divergence.
     2. Explain what the successful rollout did that the failed rollout omitted,
        misunderstood, or executed incorrectly.
     3. Check whether the current skill already contains the needed guidance.  If it does,
        sharpen it only when the failed behavior shows the rule is ambiguous or
        insufficiently actionable.
     4. Distinguish skill-relevant reasoning/strategy errors from random wording, evaluator
        noise, tool nondeterminism, and infrastructure behavior.
     5. Propose only generalizable guidance.  Never encode the task id, its answer, entity
        names, file names, cell addresses, or other instance-specific facts.
     6. Prefer the smallest effective edit.  It is correct to return no edits when the
        contrast does not support a causal, reusable lesson.
     7. Do not edit protected content between SLOW_UPDATE markers.
     8. Compare the mechanism with the supplied cross-step hypothesis bank.  Link an
        existing ID only when the causal mechanism is genuinely the same.  Otherwise leave
        the related ID empty.  A causal, generalizable contrast from one task remains
        eligible for an edit; cross-task support is ranking evidence, not an eligibility
        requirement.

Respond ONLY with a valid JSON object:

{
  "contrast_summary":  {
    "first_divergence":  "<earliest important difference>",
    "success_behavior":  "<what worked>",
    "failure_behavior":  "<what went wrong>",
    "causal_hypothesis":  "<why this changed the outcome>",
    "generalizable":  true,
    "confidence":  0.0
  },
  "hypothesis":  {
    "related_hypothesis_id":  "<existing bank ID or empty>",
    "mechanism":  "<abstract reusable behavioral mechanism>"
  },
  "patch":  {
    "reasoning":  "<why these edits follow from the same-task contrast>",
    "edits":  [
      {
        "op":  "append|insert_after|replace|delete",
        "target":  "<if needed>",
        "content":  "<generalizable markdown guidance>"
      }
    ]
  }
}
```

In this section, we provide the prompt for the core component of SkillCome, group contrast reflection. As shown in Prompt A.1, the optimizer jointly analyzes successful and failed trajectories within the same group and returns a structured JSON response containing the contrast analysis, pattern information for contrast memory, and proposed skill edits. Prompt A.2 further presents an example output of group contrast reflection. The example corresponds to the ALFWorld case discussed in Figure 3 of §5.1. It is the output from training step 1, telling the agent to search the target beyond visited places around the destination.

Prompt A.2: Example of Group Contrast Reflection Output


```
{
  "contrast_summary": {
    "first_divergence": "The successful rollout eventually visited both sidetables after
exhaustively searching dresser-adjacent locations, found the two keychains there,
transported them one at a time to the dresser, and placed them. Most failed rollouts never
visited the sidetables at all, or found the keychains late but then wandered back to
already-searched receptacles instead of delivering them immediately.",
    "success_behavior": "The successful agent treated the search as open-ended across all
room furniture rather than only around the destination dresser. Once it found the first
keychain, it handled the full inventory constraint by moving that keychain to the
destination before returning for the second keychain, then delivered the second keychain to
the same destination.",
    "failure_behavior": "Failed trajectories repeatedly cycled through drawers and cabinets
near the dresser, re-examined known-empty receptacles many times, and often never explored
the sidetables containing the goal objects. Even trajectories that found one keychain
frequently continued searching known-empty locations or failed to deliver held goal objects
promptly.",
    "causal_hypothesis": "Agents assumed the target objects would be near the destination
and exhausted only its neighboring receptacles, repeatedly revisiting them instead of
searching unexplored furniture elsewhere. This caused step-limit failure because the
actual target objects were on unvisited sidetables. In addition, when an agent held one
target object, it needed to place it before picking up the second, because inventory holds
only one object.",
    "generalizable": true,
    "confidence": 0.82
  },
  "hypothesis": {
    "related_hypothesis_id": "contrast_v2_cc5a59433591",
    "mechanism": "Agents fail multi-object search tasks when they limit exploration to
receptacles near the stated destination and repeatedly revisit exhausted locations.
Success requires maintaining unexplored-location coverage across the whole environment and,
when inventory is capacity-limited and the task requires two objects, delivering the first
target object before retrieving the second."
  },
  "patch": {
    "reasoning": "The same-task contrast shows failures not because the agents lacked the
locate/take/place sequence but because they searched only near the destination and
revisited known-empty receptacles. The existing skill already warns about revisiting
searched locations, but it does not explicitly instruct agents to broaden exploration
across the entire environment when local receptacles are empty, nor to deliver a held
target object before attempting to pick up a second required object when inventory is
limited. These are actionable, task-type-general extensions.",
    "edits": [
      {
        "op": "append",
        "target": "Common Mistakes to Avoid",
        "content": "- **Destination-tunnel vision**: Do not search only receptacles
adjacent to the final destination. If nearby places are empty, expand exploration to other
furniture in the room before revisiting anything.\n- **Holding one target while trying to
take another**: If the task requires two objects and inventory is full, deliver the first
held target to the destination before picking up the second.",
        "source_type": "contrast",
        "support_count": 2,
        "group_support": 2,
        "trajectory_support": 8,
        "memory_refs": [
          {
            "bank_type": "contrast",
            "hypothesis_id": "contrast_v2_cc5a59433591"
          }
        ],
        "hypothesis_id": "contrast_v2_cc5a59433591"
      }
    ]
  },
  "source_type": "contrast",
  "batch_size": 8,
  "trajectory_support": 8,
  "group_support": 2,
  "group_ids": [
    "train/look_at_obj_in_light-Pen-None-DeskLamp-305/trial_T20190907_115849_734053",
    "train/pick_clean_then_place_in_recep-Lettuce-None-GarbageCan-20/trial_T20190909_
033324_286989",
    "train/pick_two_obj_and_place-KeyChain-None-Dresser-322/trial_T20190908_051114_312895"
  ],
  "hypothesis_id": "contrast_v2_cc5a59433591"
}
```

Table 3: The statistics of the datasets used in our paper. Transfer-only target datasets have no training or validation split.

Dataset	Train	Validation	Test	Total
SearchQA	400	200	1400	2000
SpreadsheetBench	40	79	281	400
OfficeQA	49	49	148	246
DocVQA	107	53	374	534
LiveMath	36	35	106	177
ALFWorld	200	140	134	474
HotpotQA	–	–	1000	1000
OlympiadBench	40	100	60	200
Omni-MATH	–	–	1000	1000

B Detailed Experimental Setting

B.1 Dataset Details

Table 3 summarizes the dataset and splits adopted in our work. The main evaluation covers six benchmarks spanning question answering, mathematical reasoning, and agentic tasks. We mainly follow the splits from SkillOpt-Lite (Shen et al., 2026b). Detailed information on each dataset is as follows:

- **SearchQA** (Dunn et al., 2017) pairs trivia questions with supporting snippets retrieved from the web. Answering these questions requires extracting relevant facts from potentially noisy or redundant evidence. Here, models answer these questions using the provided snippets rather than performing live web searches.
- **SpreadsheetBench** (Ma et al., 2024) evaluates spreadsheet manipulation through practical requests originating from real users. Agents must interpret natural-language instructions and implement Python code to perform the corresponding operations on spreadsheet files.
- **OfficeQA** (Opsahl-Ong et al., 2026) evaluates question answering over lengthy financial documents containing narrative text and tables. We use the 246-question collection in an offline document-tool setting, where agents inspect locally available documents to produce their answers.
- **DocVQA** (Mathew et al., 2021) evaluates visual question answering over document images. The model must read document content and interpret its spatial organization to locate the information requested by a question.
- **LiveMathematicianBench (LiveMath)** (He et al., 2026) assesses research-level mathematical reasoning through multiple-choice questions constructed from mathematical research papers.
- **ALFWorld** (Shridhar et al., 2021) evaluates sequential decision making in household environments through a text-based interaction interface. Agents must search for objects, manipulate them, and coordinate multiple actions to satisfy a specified goal.

We additionally use three datasets for cross-dataset transfer experiments. The search-based benchmark HotpotQA (Yang et al., 2018) is adopted to evaluate the skills evolved on SearchQA. We also transfer the skills evolved on OlympiadBench (He et al., 2024) to Omni-MATH (Gao et al., 2024). OlympiadBench includes training, validation, and test splits, whereas HotpotQA and Omni-MATH are used exclusively for transfer evaluation without training or validation.

- **HotpotQA** (Yang et al., 2018) is a multi-hop question-answering benchmark based on Wikipedia articles. Its questions require combining evidence from multiple passages, including linking related facts and comparing information about different entities. We randomly select 1000 examples from its distractor validation split to evaluate the direct transfer of skills learned on SearchQA.

- **OlympiadBench** (He et al., 2024) contains challenging open-end mathematics and physics problems drawn from competitions and examinations. We use a mathematics subset as the source dataset for mathematical skill transfer.
- **Omni-MATH** (Gao et al., 2024) evaluates Olympiad-level mathematical problem solving across diverse mathematical topics and difficulty levels. We evaluate skills transferred from OlympiadBench on 1000 randomly selected samples on Omni-MATH.

B.2 Implementation Details

In this subsection, we provide additional implementation details. Trace2Skill (Ni et al., 2026) uses its official implementation, which collects trajectories for one training epoch and analyzes them to construct the skill. Our method uses the same training budget to that of SkillOpt. Following the SkillOpt-Lite training protocol, we run SkillOpt and our method for four epochs or ten batches, whereas SkillOpt-Lite is executed for exactly ten batches.

For all models, the maximum length is set to 16384 tokens per response. For grouped rollouts, we sample $n = 8$ trajectories per question. In practice, because group rollouts involve repeated rollouts for the same question, we deduplicate the input trajectories for failure and success reflection. Since the test sets of the main benchmarks other than SearchQA contain only a few hundred examples, their scores can fluctuate across evaluations. We therefore evaluate these benchmarks over four independent runs and report the average score to reduce the impact of stochastic variation.

C Additional Experiments

C.1 Notable Findings in the Experiment

A LiveMath Question with a Meta-Option

Question (excerpt). What is the strongest statement that can be proved about every sufficiently small perturbation $\psi_0 \in \mathcal{E}$ of ϕ measured by ρ_R?

Option A. One of the remaining options is correct, but a stronger result can be proven.

Options B–E (omitted). Mathematical statements about stability with different conditions on the time domain, quantifiers, and phase or spatial shifts.

Ground-truth answer: A.

Figure 6: An example of the meta-option pattern in LiveMath. Option A is the meta-option and the ground-truth answer. The other options are summarized for brevity.

Table 4: Livemath Performance with the Meta and Non-Meta subsets.

Model	Skill	All	Meta	Non-Meta
GLM-5.3	No skill	36.08	14.67	52.50
	Init skill	38.44	16.30	55.42
	SkillCome	**67.92**	**71.74**	**65.00**
DeepSeek-V4 Flash	No skill	31.60	4.35	52.50
	Init skill	35.38	10.33	54.58
	SkillCome	**74.29**	**96.74**	**57.08**

As shown in Table 1 of §4.2, the three more capable models, GLM-5.3, Qwen3.8-Flash-Next, and DeepSeek-V4-Pro-0813 do not always outperform DeepSeek-V4-Flash and Qwen3.6-35B-A3B after skill evolution, especially on LiveMath. This is because some LiveMath questions include the option "One of the remaining options is correct, but a stronger result can be proven." We refer to

(a) Meta-option Related Skill on DeepSeek-V4-Flash

Meta-Option Override (HARD RULE)

- If the question asks for the "strongest," "best possible," or "maximal" statement, OR if any option says "One of the remaining options is correct, but a stronger result can be proven" (or its equivalent), **STOP ALL ANALYSIS IMMEDIATELY**. Do not read the explicit options. Do not compare hypotheses, exponents, group structures, or any mathematical content.

- **YOUR FINAL ANSWER IS THE META-OPTION.** Output its letter and nothing else.

(b) Meta-option Related Skill on GLM-5.3

Meta-Option Priority (UNCONDITIONAL DEFAULT)

When an option of the form "One of the remaining options is correct, but a stronger result can be proven" is present, and the question asks for the strongest provable statement, a complete classification/characterization, the complete list, or "which description gives the complete...", TAKE THE META-OPTION. This is the single highest-yield rule: in the training history, meta-options are correct far more often than any concrete option, including on questions about spectral graph bounds, quantum K-theory, von Neumann algebras, PDE regularity, commutator lengths, quasimodular forms, and clique-sum decompositions.

1. **CRITICAL — a successful concrete derivation does NOT license rejecting the meta-option.** If you reconstruct a classical theorem (Schauder regularity, a spectral extremal bound with equality case, a duality congruence, a structural decomposition) and verify a concrete option matches it, that proves ONLY truth, never maximality. [...]

2. **Only two legitimate exits from the meta-option:** (a) you can name a specific, rigorous obstruction (rigidity theorem, explicit counterexample to EVERY candidate strengthening, matching sharp lower bound) valid under the FULL hypotheses; or (b) the meta-option is internally contradicted. Recall, derivation, confidence, and "the concrete option already seems sharp" are NOT exits.

[...]

Figure 7: Skills related to the Meta-option pattern on DeepSeek-V4-Flash and GLM-5.3. The skill of DeepSeek-V4-Flash instructs the agent to select the meta-option without further analysis. In contrast, the GLM-5.3 skill prioritizes the meta-option, but still requires the model to analyze the math problem itself. [...] indicates omitted text.

this as the *meta-option*, since this option is always labeled as correct whenever it appears. Figure 6 provides an example of questions with meta-option. Of the 106 test samples, 46 contain this option and constitute the *Meta* subset, while we refer to the remaining 60 as the *Non-Meta* subset. This regularity therefore provides a shortcut for answering the Meta subset without resolving the underlying mathematical problem.

The skills learned for DeepSeek-V4-Flash and Qwen3.6-35B-A3B adopt a strict rule for this pattern: once the meta-option is detected, the model should stop further reasoning and return its corresponding label. By contrast, the skills learned for the stronger models also recognize the pattern but retain instructions to analyze the mathematical content before answering. Figure 7 contrasts the relevant skill learned for DeepSeek-V4-Flash and GLM-5.3. These different strategies explain why the weaker target models achieve higher overall scores on LiveMath, since their gains mostly came from the Meta subset.

Table 4 reports the detailed results for each subset on DeepSeek-V4-Flash and GLM-5.3. Relative to the initial skill, Flash improves from 10.33% to 96.74% on Meta, but only from 54.58% to 57.08% on Non-Meta. GLM-5.3 exhibits a more balanced improvement, reaching 71.74% on Meta and 65.00% on Non-Meta. Although Flash achieves a higher overall score, GLM-5.3 remains stronger on questions without the meta-option. Overall, GLM-5.3 exhibits stronger mathematical reasoning capabilities. The meta-option rule improves accuracy under the benchmark's labeling pattern, but its benefit should not be interpreted as an equivalent improvement in mathematical reasoning.

Table 5: Standard deviations of SkillCome. Mean scores are provided in Table 1. "–" denotes unsupported evaluation.

Model	Spreadsheet	LiveMath	ALFWorld	OfficeQA	DocVQA
Qwen3.8-Flash-Next	1.78	1.81	1.83	2.09	0.69
DeepSeek-V4-Pro-0813	1.71	2.50	0.43	0.65	–
GLM-5.3	0.92	2.83	0.43	1.50	–
DeepSeek-V4-Flash	1.77	0.90	1.27	3.46	–
Qwen3.6-35B-A3B	3.39	0.90	1.65	2.01	0.86

C.2 Standard Deviation of Main Results

Table 5 reports the standard deviations of SkillCome over four independent evaluation runs. SearchQA is excluded because it is evaluated only once. The relatively small standard deviations indicate that the performance of SkillCome remains stable.

C.3 Statistics of the Mix Group

Table 6: Mixed-group counts and proportions by target model. A mixed group contains both successful and failed rollouts of the same task. Counts are reported as mixed groups / total groups.

Target model	Mixed groups	Proportion
DeepSeek-V4-Flash	142 / 430	33.02%
Qwen3.6-35B-A3B	192 / 480	40.00%
GLM-5.3	107 / 430	24.88%
DeepSeek-V4-Pro-0813	152 / 430	35.35%
Qwen3.8-Flash-Next	138 / 480	28.75%
Total	**731 / 2250**	**32.49%**

Table 6 summarizes the mixed groups collected during skill evolution. The two multimodal LLMs, Qwen3.6-35B-A3B and Qwen3.8-Flash-Next, have more total group counts since their experiments also include DocVQA. Across the five target models, 731 of the 2,250 groups contain both successful and failed trajectories. The remaining non-mixed groups contain either only successful trajectories or only failed trajectories. These statistics show that mixed groups account for a substantial fraction of the collected groups, supporting group contrast reflection across different models.

C.4 Ablation on the Components of SkillCome

In this subsection, we further analyze our ablation experiment on the components of SkillCome discussed in §4.3. Compared with SkillOpt, "SkillOpt w/ Memory" improves performance in 8 of the 11 model–benchmark pairs and reduces in the remaining six. Although the gains are not consistent, it improves DeepSeek-V4-Flash on OfficeQA by 21.79 points over SkillOpt. This can be attributed to the memory system, which helps to learn explicit guidance against the DSML-style answer format discussed in § 4.2.

Meanwhile, "SkillCome w/o Memory" generally shows better performance than Skillopt. Together, the ablations demonstrate that both components of our method are effective. More importantly, combining them leads to the best results on all model-benchmark pairs, suggesting that the two components are synergistic. Group contrast enables deeper per-question analysis, while dual memory connects these findings with evidence from a broader range of samples.

Table 7: Performance comparison of SkillCome under direct chat and the Codex CLI harness.

Model	Skill	Spreadsheet	SearchQA	LiveMath
Qwen3.8 Flash-Next	No skill	48.58	67.50	33.73
	Init skill	45.55$_{-3.03}$	67.97$_{+0.47}$	35.14$_{+1.41}$
	SkillCome	75.71$_{+27.13}$	74.14$_{+6.64}$	63.68$_{+29.95}$
	Codex-SkillCome	**85.14**$_{+36.56}$	**78.29**$_{+10.79}$	**67.92**$_{+34.19}$
DeepSeek-V4 Pro-0813	No skill	47.78	66.57	28.07
	Init skill	48.40$_{+0.62}$	67.07$_{+0.50}$	31.37$_{+3.30}$
	SkillCome	62.10$_{+14.32}$	72.50$_{+5.93}$	53.30$_{+25.23}$
	Codex-SkillCome	**75.27**$_{+27.49}$	**72.71**$_{+6.14}$	**71.23**$_{+43.16}$

C.5 Evaluation in Agent Harnesses

The detailed results of our evaluation in agent harnesses are presented in Table 7, corresponding to Figure 5 in §5.4. For both models, running under Codex consistently outperforms running under direct chat across all datasets. The performance gains from the harness environment and its tools are particularly pronounced on SpreadsheetBench and Livemath. These results demonstrate the effectiveness of our method within advanced harness environments, highlighting its practical value in real-world workflows.

C.6 Cross-Model Transferability

Table 8: Cross-target skill transferability experiments of SkillCome. "Qwen3.6 → Flash" denotes transferring the skill learned on Qwen3.6-35B-A3B to DeepSeek-V4-Flash. "Flash → Flash" is the original performance of SkillCome on the DeepSeek-V4-Flash.

Target Model	Skill	Spreadsheet	SearchQA	LiveMath	ALFWorld	OfficeQA	Avg
DeepSeek-V4 Flash	No skill	47.33	64.85	31.60	59.51	5.41	41.74
	Init skill	48.58$_{+1.25}$	65.43$_{+0.58}$	35.38$_{+3.78}$	56.91$_{-2.60}$	11.66$_{+6.25}$	43.59$_{+1.85}$
	Qwen3.6 → Flash	60.05$_{+12.72}$	72.21$_{+7.36}$	68.87$_{+37.27}$	73.51$_{+14.00}$	22.30$_{+16.89}$	59.39$_{+17.65}$
	Flash → Flash	**62.54**$_{+15.21}$	**73.36**$_{+8.51}$	**74.29**$_{+42.69}$	**84.14**$_{+24.63}$	**50.51**$_{+45.10}$	**68.97**$_{+27.23}$
Qwen3.6 35B-A3B	No skill	41.99	66.00	26.65	66.04	37.16	47.57
	Init skill	40.75$_{-1.24}$	66.23$_{+0.23}$	30.19$_{+3.54}$	79.48$_{+13.44}$	38.68$_{+1.52}$	51.07$_{+3.50}$
	Flash → Qwen3.6	60.23$_{+18.24}$	70.79$_{+4.79}$	**72.17**$_{+45.52}$	83.02$_{+16.98}$	**40.71**$_{+3.55}$	65.38$_{+17.81}$
	Qwen3.6 → Qwen3.6	**61.48**$_{+19.49}$	**71.29**$_{+5.29}$	71.46$_{+44.81}$	**86.38**$_{+20.34}$	40.03$_{+2.87}$	**66.13**$_{+18.56}$

Figure 4a in § 5.2 presents the transfer of skills learned on Qwen3.6-35B-A3B to DeepSeek-V4-Flash. We also have evaluated the reverse direction and report the complete results in Table 8. Skills learned on DeepSeek-V4-Flash transfer well to Qwen3.6-35B-A3B, even outperforming skills evolved specifically for the target model on LiveMath and OfficeQA. Conversely, as discussed in § 5.2, the transferred skills on OfficeQA of DeepSeek-V4-Flash shows substantially lower OfficeQA performance than self-evolved skills, because the transferred skill does not explicitly address Flash's DSML-style answer format. These results indicate that SkillCome learns not only reusable task-level strategies that generalize across different models, but also model-specific rules that correct distinctive behavioral errors.

C.7 Cross-Dataset Transferability

Table 9 reports the complete results corresponding to Figure 4b in § 5.2. We transfer the skills learned on SearchQA to HotpotQA, a similar search-based task. Following SkillOpt, we also transfer skills evolved from OlympiadBench to Omni-MATH, another benchmark for open-ended mathematical reasoning. The transferred skills improve performance across all eight model–benchmark pairs, with gains ranging from 0.2 to 4.9 points. Although the magnitude of improvement varies across models, the consistent gains on both target datasets indicate that the learned skills provide

Table 9: Full results of cross-dataset skill transferability. Δ denotes the score improvement of transferred skill over no skill.

Source dataset	Target dataset	Model	No skill	Transferred skill	Δ
SearchQA	HotpotQA	DeepSeek-V4-Pro-0813	63.20	64.80	+1.60
		Qwen3.8-Flash-Next	66.30	67.90	+1.60
		DeepSeek-V4-Flash	62.70	64.80	+2.10
		Qwen3.6-35B-A3B	64.80	65.70	+0.90
OlympiadBench	Omni-MATH	DeepSeek-V4-Pro-0813	72.30	74.10	+1.80
		Qwen3.8-Flash-Next	86.40	86.60	+0.20
		DeepSeek-V4-Flash	58.90	60.10	+1.20
		Qwen3.6-35B-A3B	66.70	71.60	+4.90

reusable guidance beyond the datasets on which they were optimized. These results support the cross-dataset generalizability of SkillCome within similar domains.

C.8 Evolving from Empty Skill

Table 10: Performance of SkillCome evolving from empty skills.

Model	Skill	Spreadsheet	SearchQA	LiveMath	ALFWorld	OfficeQA	DocVQA	Avg
DeepSeek-V4 Flash	No skill	47.33	64.85	31.60	59.51	5.41	–	41.74
	SkillCome	58.45$_{+11.12}$	71.79$_{+6.94}$	54.95$_{+23.35}$	79.10$_{+19.59}$	7.09$_{+1.68}$	–	54.28$_{+12.54}$
	Init skill	48.58	65.43	35.38	56.91	11.66	–	43.59
	SkillCome	**62.54**$_{+13.96}$	**73.36**$_{+7.93}$	**74.29**$_{+38.91}$	**84.14**$_{+27.23}$	**50.51**$_{+38.85}$	–	**68.97**$_{+25.38}$
Qwen3.6 35B-A3B	No skill	41.99	66.00	26.65	66.04	37.16	89.77	54.60
	SkillCome	58.01$_{+16.02}$	70.07$_{+4.07}$	64.86$_{+38.21}$	**87.13**$_{+21.09}$	33.28$_{-3.88}$	90.04$_{+0.27}$	67.23$_{+12.63}$
	Init skill	40.75	66.23	30.19	79.48	38.68	90.17	57.58
	SkillCome	**61.48**$_{+20.73}$	**71.29**$_{+5.06}$	**71.46**$_{+41.27}$	86.38$_{+6.90}$	**40.03**$_{+1.35}$	**91.24**$_{+1.07}$	**70.31**$_{+12.73}$

As shown in the main Table 1, initial skills do not consistently improve performance over "No skill". For example, both Qwen models perform worse with the initial skill on Spreadsheet, while several other model–benchmark pairs exhibit only marginal gains. These observations raise the question of whether skill evolution can achieve better results when starting without an initial skill. We therefore investigate the setting of evolving from empty skill on DeepSeek-V4-Flash in Figure 4c of §5.3. Here, we provide more result on Qwen3.6-35B-A3B.

The full results are presented in Table 10. Skill evolution from an empty skill improves over the "No skill" baseline in 10 of the 11 model–benchmark pairs, demonstrating that SkillCome can learn useful guidance without a predefined skill. Nevertheless, evolution from the initial skill achieves higher final accuracy in 10 of the 11 pairs. The only exception is ALFWorld on Qwen3.6-35B-A3B, where starting from an empty skill yields a slight advantage. Notably, on Spreadsheet with Qwen3.6-35B-A3B, the initial skill reduces performance before evolution but leads to a better final result, suggesting that a skill's immediate effectiveness does not fully reflect its value for evolution.

One explanation is that an initial skill provides task-specific procedures and constraints that offer a useful starting point for revision, even when some instructions are ineffective. Under a limited optimization budget, the optimizer model can refine this existing guidance and correct problematic rules, rather than construct and organize the skill from scratch. Such guidance may also influence the trajectories collected during evolution, providing more informative behavioral references for later updates. Overall, these results suggest that skill initialization affects not only the starting performance but also the subsequent evolution process. Although an empty initialization remains a viable alternative, an initial skill generally provides a more effective basis for evolution.

C.9 Token Usage of Optimizer Model

Since the target model rollout budget is matched during training, we compare the optimizer model token usage of SkillCome and SkillOpt. As shown in Table 11, SkillCome consumes fewer optimizer

Table 11: Optimizer token usage in millions (M). The ratio is computed as SkillCome/ SkillOpt.

Target model	Dataset	SkillCome (M)	SkillOpt (M)	Ratio
DeepSeek-V4 Flash	SpreadsheetBench	0.69	1.50	0.460×
	SearchQA	1.69	4.64	0.364×
	OfficeQA	0.97	2.20	0.441×
	LiveMath	0.56	0.90	0.622×
	ALFWorld	1.09	1.57	0.694×
Qwen3.6 35B-A3B	SpreadsheetBench	0.56	1.31	0.427×
	SearchQA	2.15	5.07	0.424×
	OfficeQA	0.75	1.86	0.403×
	DocVQA	0.16	0.45	0.356×
	LiveMath	0.39	1.06	0.368×
	ALFWorld	1.05	1.67	0.629×

tokens across all evaluated datasets for both target models. This reduction primarily comes from lower input token usage through trajectory deduplication under grouped rollouts, as described in Appendix B.2. These results demonstrate that SkillCome achieves substantial performance gains while reducing optimizer token costs.