SKILLER: Language-Level Reinforcement Learning for Reusable Skill Extraction in Small Language Models

Chenhao Dang[1,2*], **Siyuan Xiong**[3*], **Conghui He**[2†], **Weijia Li**[2,4†]

[1]Shanghai Jiao Tong University, [2]Shanghai Artificial Intelligence Laboratory, [3]Harbin Institute of Technology, Shenzhen, [4]Tsinghua Shenzhen International Graduate School, Tsinghua University,
dangchenhao@pjlab.org.cn, xiongsiyuan@stu.hit.edu.cn, heconghui@pjlab.org.cn, liweijia@sz.tsinghua.edu.cn

Abstract

Agent skills represent a standardized format for packaging procedural knowledge and domain expertise, serving within agent harness systems as an essential mechanism to continually constrain a language model's behavior space for repeatable, high-quality task execution. However, because strong closed-source models entail high inference costs, current popular agent harnesses, such as Codex and OpenClaw, remain prohibitively expensive when deploying these skills to accomplish real-world tasks. The rapid capability enhancement of open-source models deployable on consumer-grade GPUs presents a compelling opportunity to drastically reduce these costs by leveraging skill-based behavioral constraints. Nevertheless, automatically generating effective skills tailored specifically for such compact models remains a significant practical challenge. To address this, we propose SKILLER, a natural-language-driven reinforcement learning framework designed to automatically generate executor-specific skills for small models, which employs a strong model as the actor and critic, treats the small-model agent system as the environment, and propagates all reinforcement learning signals entirely via natural language. Extensive experimental evaluations across five relevant benchmarks using Qwen3.5-9B and Qwen3.5-4B demonstrate that SKILLER outperforms three open-source and one closed-source skill generation or evolution methods, achieving absolute gains ranging from 4.3 to 20.4 percentage points for the 9B model and 1.8 to 13.3 points for the 4B model, while remarkably matching the performance of strong closed-source models on single-skill tasks in SkillsBench. The project is available at https://github.com/DANG-ai/SKILLER.

Introduction

In the rapidly evolving landscape of autonomous agents, agent skills have emerged as a foundational primitive (Zhang, Lazuka, and Murag 2025; Ling, Zhong, and Huang 2026). Conceptually aligned with recent frameworks formalized by AI research organizations such as Anthropic, an agent skill is not merely a prompt, but a standardized format for packaging procedural knowledge, tool-use conventions, and domain expertise (Anthropic 2026a; Li et al. 2026). Within advanced agent harness systems, these skills serve as an essential mechanism to continually constrain the behavior space of Large Vision-Language Models (LVLMs), thereby ensuring that complex tasks are executed in a repeatable and high-quality manner. However, achieving this reliability traditionally relies on strong, closed-source frontier models. Because these models entail exorbitant inference costs, current popular agent harnesses, such as Codex, Claude Code, OpenCode, and OpenClaw, remain prohibitively expensive when deploying skills to accomplish real-world tasks at scale (Anthropic 2026b; OpenAI 2026a; OpenCode 2026; OpenClaw Foundation 2026).

A promising alternative to this cost bottleneck lies in the rapid capability enhancement of open-source compact models deployable on consumer-grade GPUs, such as the Qwen3.5, Qwen3.6, and Gemma 4 series (Qwen Team 2026; Google DeepMind 2026). When augmented with specific, well-defined skill constraints, these compact models exhibit surprising proficiency (Xu et al. 2026; Li et al. 2026). For simple or highly repetitive real-world tasks, guiding these small-scale LVLMs with strict procedural boundaries not

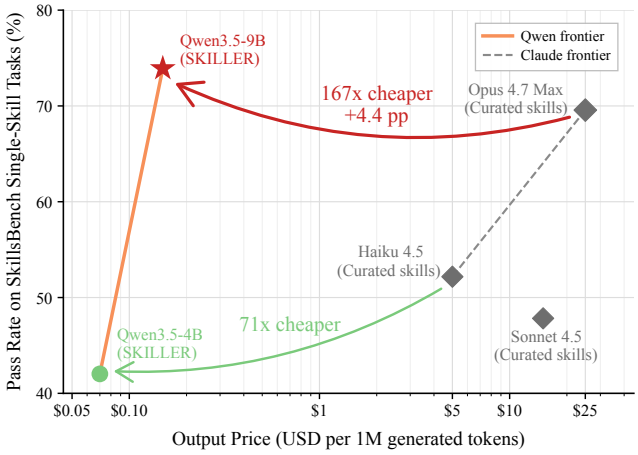


Figure 1: Cost–performance on single-skill tasks of SkillsBench. Skills generated by SKILLER enable Qwen3.5-based agent loops to approach the frontier at much lower output-token prices. On specific tasks, Qwen3.5-9B with SKILLER skills even outperforms closed-source models using curated skills, suggesting that well-matched agent skills can make lower-cost models more effective than more expensive alternatives, as in the case of Haiku 4.5 outperforming Sonnet 4.5 (Li et al. 2026).

*These authors contributed equally. † Corresponding authors.

only can maintain high success rates but also drastically reduce operational costs (Xu et al. 2026; Li et al. 2026). As illustrated in Figure 1, when equipped with appropriate skills, compact models like Qwen3.5-9B and Qwen3.5-4B can achieve task-specific performance that rivals that of their massive, closed-source counterparts, unlocking a highly cost-effective paradigm for agentic deployment.

Despite this immense potential, a critical model-mismatch problem hinders direct deployment. Skills crafted for strong frontier models do not natively transfer to small-scale LVLMs (Huang et al. 2026; Liu et al. 2026). The behavioral constraints, implicit reasoning steps, and error-recovery assumptions that successfully guide a massive model are often completely ineffective for a compact model. A small-scale LVLM might easily hallucinate arguments, skip essential verification steps, or become derailed by overly complex instructions that feature multiple branching paths. Consequently, directly porting existing high-end skills to these compact models frequently leads to catastrophic task failure. How to automatically and reliably generate effective skills tailored specifically to the unique behavioral spaces of such small-scale LVLMs remains a practical challenge.

To overcome this challenge, we propose SKILLER, a novel natural-language-driven reinforcement learning framework designed to automatically generate and optimize executor-specific skills for small-scale LVLMs. Unlike traditional reinforcement learning paradigms that update neural weights, SKILLER treats the textual skill itself as the optimizable policy. The framework employs a state-of-the-art frontier model, such as GPT-5.5 or Claude opus 4.8, to serve as the actor and critic. The environment within this reinforcement learning formulation operates as an agent loop driven by the target open-source, small-scale LVLM. This environment functions under a progressive skill disclosure mechanism that prevents the compact model from being overwhelmed by long textual contexts. Crucially, all reinforcement learning signals in SKILLER, including states, diagnostic rewards, and policy update actions, are propagated entirely via structured natural language, bridging the gap between strong-model reasoning and small-scale LVLM execution.

We comprehensively evaluate SKILLER using Qwen3.5-9B and Qwen3.5-4B across a diverse suite of benchmarks. Our evaluation encompasses four general-purpose benchmarks, namely SkillsBench (Li et al. 2026), SkillLearnBench (Zhong et al. 2026), SWE-Skills-Bench (Han et al. 2026), and GAIA (Mialon et al. 2024), alongside one specialized domain benchmark designated as EarthBench (Feng et al. 2025). Furthermore, we benchmark our framework against three open-source skill generation and evolution methods, specifically EvoSkill (Alzubi et al. 2026), AutoSkill (Yang et al. 2026), and SkillX (Wang et al. 2026), as well as one closed-source skill-generation baseline known as Manus (Manus 2026). Empirical results demonstrate that the skills generated by SKILLER yield substantial and consistent performance improvements within the agent loops of both the 9B and 4B compact models. These findings confirm that language-level reinforcement learning effectively unlocks the autonomous capabilities of cost-efficient small-scale LVLMs.Our main contributions of this work are summarized as follows.

- **Novel Skill Generation Framework.** We introduce SKILLER, a natural-language-driven reinforcement learning framework designed to automatically generate executor-specific skills for small-scale LVLMs. Treating textual skills as optimizable policies, our approach leverages a strong frontier model to serve as the actor and critic within the interactive environment of the target compact model's agent loop.
- **Language-Level Policy Iteration.** We formulate a language-level policy iteration mechanism that propagates all reinforcement learning signals, including states, diagnostic rewards, and policy update actions, entirely via structured natural language. This feedback loop resolves the model-mismatch gap by diagnosing and repairing the unique failure modes of compact models, generating tailored constraints without any neural weight updates.
- **Comprehensive Empirical Validation.** We extensively evaluate SKILLER across five diverse benchmarks using the Qwen3.5-9B and Qwen3.5-4B compact models, consistently outperforming four baseline skill evolution methods. The generated skills empower these small-scale LVLMs to rival the task-specific performance of massive closed-source counterparts, establishing a highly cost-effective paradigm for agentic deployment.

Related Work

Agent skills. Agent systems frequently interleave reasoning with tool execution (Yao et al. 2023; Schick et al. 2023), and agent skills package these procedural choices into reusable memory artifacts. The evaluation of these skills across diverse benchmarks highlights both their utility and their uneven transferability. For instance, curated skills can improve execution success (Li et al. 2026), whereas mismatched or partially relevant skills often degrade performance (Han et al. 2026; Liu et al. 2026). Furthermore, comprehensive benchmarks systematically evaluate continual skill generation (Zhong et al. 2026), tool composition abilities (Chen et al. 2026), and lifelong library maintenance (Zhang et al. 2026b). Because well-crafted skills provide strict procedural boundaries that restrict the action space, they are exceptionally well-suited for guiding small-scale LVLMs to complete tasks efficiently in specific structured scenarios.

Skill acquisition and evolution. Prior literature explores skill acquisition from diverse sources, including public repositories (Bi et al. 2026), large-scale networks (Liang et al. 2026), interaction histories (Yang et al. 2026), collective trajectories (Ma et al. 2026), and successful agent rollouts (Alzubi et al. 2026; Wang et al. 2026). Advanced frameworks facilitate this evolution by maintaining executable libraries (Wang et al. 2023), storing verbal feedback (Shinn et al. 2023), distilling local lessons (Ni et al. 2026), managing dual expertise forms (Qiu et al. 2026), and utilizing co-evolutionary verification (Zhang et al. 2026a). Recent methods also integrate skills with reinforcement learning by modifying agent policies, optimizing skill banks, or inducing programmatic constraints (Wang et al. 2025a; Xia et al. 2026; Mi et al. 2026; Wang et al. 2025b). However, these existing

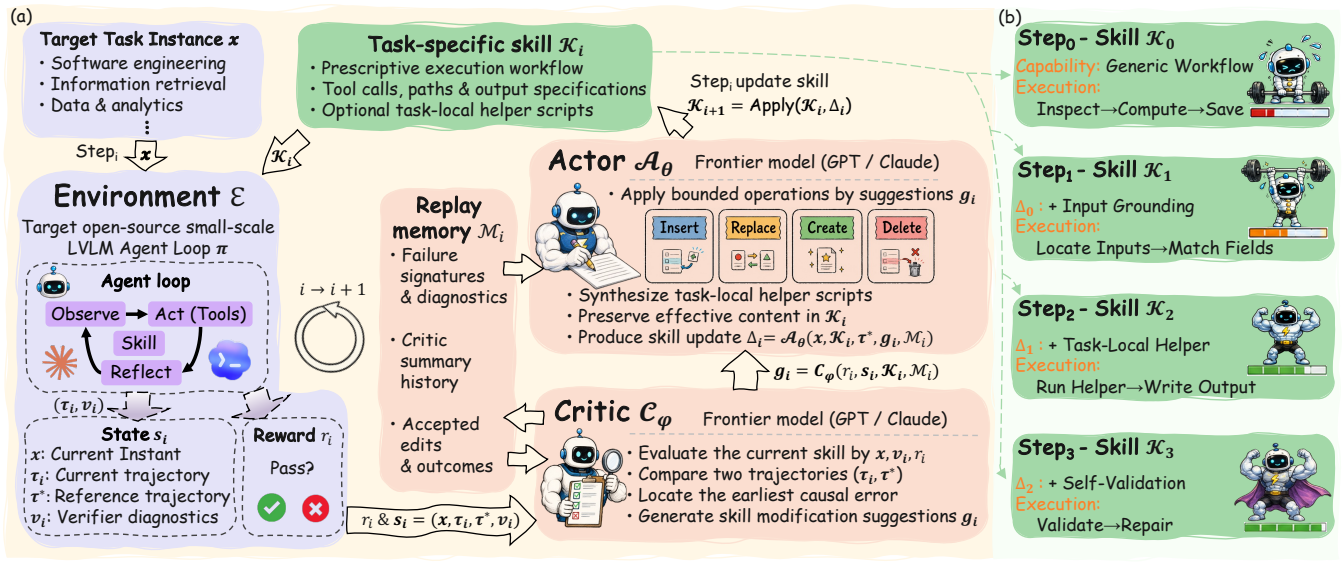


Figure 2: Overview of SKILLER. **(a)** Automated generation and iterative refinement of task-specific skills tailored for compact models. At each optimization step, the framework takes a target task instance and the current skill, treating the small-scale LVLM agent loop as an interactive environment to produce a structured state quadruple and a scalar reward. Driven by a frontier model, the critic analyzes the current state alongside historical records from the replay memory to evaluate execution outcomes, isolate causal errors, and formulate modification suggestions before saving these diagnostics back into memory. The actor then executes bounded editing operations to update the skill content. Crucially, all information transfer across this optimization loop is conducted entirely via natural language. **(b)** The progressive performance enhancement of the small-scale LVLM agent loop as the underlying skill iteratively evolves and strengthens across successive optimization steps.

generation and evolution techniques are predominantly optimized for massive frontier models, and there are currently no skill generation methods specifically tailored to address the unique constraints and execution paradigms of compact models.

Method

SKILLER is a natural-language policy optimization framework that translates frontier-model reasoning into task-specific skills for small-scale LVLMs, without updating the parameters of the compact model itself. As illustrated in Figure 2, SKILLER treats the textual skill as the optimization variable, the small-scale LVLM agent loop as an interactive environment, and the official benchmark verifier as the source of task reward. Driven by a frontier model, a critic module converts execution evidence into causal and localized feedback, while an actor module implements this feedback through bounded edits to the skill artifact. A structured replay memory preserves failure signatures, critic diagnoses, and previously effective modifications across sequential optimization steps. This formulation concentrates computational expense at skill-construction time, yields an interpretable and directly deployable policy artifact, and propagates all diagnostic feedback and editing instructions entirely through natural language rather than gradient updates.

Problem Definition

Let $\mathbf{x} \in \mathcal{X}$ denote a target task instance, including its instruction, runtime inputs, available tools, and output contract. A frozen compact model π interacts with a task environment $\mathcal{E}$ under a task-specific skill $\mathcal{K}_i$ at optimization step i. The skill conditions the compact model's action distribution, inducing the effective policy

$$\pi_{\mathcal{K}_i}(a_t \mid h_t, \mathbf{x}) \triangleq \pi(a_t \mid h_t, \mathbf{x}, \mathcal{K}_i), \tag{1}$$

where h_t is the interaction history and a_t is an action selected by the model. One execution produces an execution trajectory $\tau_i = (o_{i,0}, a_{i,0}, \ldots, o_{i,T})$. The environment then exposes the benchmark-native performance signal directly as a scalar reward $r_i \in [0,1]$, instantiated as task success or test pass rate, together with verifier diagnostics $\mathbf{v}_i$, such as per-test outcomes and error messages. The complete environment transition is

$$(\tau_i, r_i, \mathbf{v}_i) = \mathcal{E}(\mathbf{x}, \mathcal{K}_i; \pi). \tag{2}$$

During optimization, a reference trajectory $\tau^\star$ supplies privileged evidence about a successful solution process, but it is never provided as a runtime input to the compact model. Let $\mathbb{K}$ denote the space of natural-language skills, which includes optional task-local helper programs. For a fixed π, our objective is to find a skill that maximizes verifier performance,

$$\begin{aligned} \mathcal{K}^\star_{\mathbf{x}} &\in \arg\max_{\mathcal{K} \in \mathbb{K}} J_{\mathbf{x}}(\mathcal{K}), \\ J_{\mathbf{x}}(\mathcal{K}) &= \mathbb{E}_{(\tau, r, \mathbf{v}) \sim p_{\mathcal{E},\pi}(\cdot \mid \mathbf{x}, \mathcal{K})}\big[r\big]. \end{aligned} \tag{3}$$

Here, $p_{\mathcal{E},\pi}(\cdot \mid \mathbf{x}, \mathcal{K})$ denotes the joint distribution over trajectories, rewards, and verifier diagnostics induced by the environment and compact model when conditioned on $\mathbf{x}$ and

$\mathcal{K}$; their realizations at optimization step i are $(\tau_i, r_i, \mathbf{v}_i)$. Rather than differentiating through π or $\mathcal{E}$, SKILLER explores $\mathbb{K}$ through verifier-grounded, natural-language policy updates $\mathcal{K}_0, \mathcal{K}_1, \ldots, \mathcal{K}_I$.

Natural-Language Skill Optimization

At each optimization step i, SKILLER takes the fixed target instance $\mathbf{x}$ and the current task-specific skill $\mathcal{K}_i$ as input, as shown at the upper left of Figure 2(a). At step 0, it instantiates an initial skill $\mathcal{K}_0$ that captures a coarse execution workflow. Each skill serves as an executable language artifact that specifies a prescriptive procedure, concrete tool calls and paths, output contracts, and, when beneficial, task-local helper scripts. The system then traverses an iterative optimization loop connecting the environment, state, critic, replay memory, and actor to specialize this artifact for the small-scale LVLM.

Environment. The environment $\mathcal{E}$ wraps the benchmark-specific tool interface, workspace, and official verifier around the compact model π. Conditioned on $\mathcal{K}_i$, the model repeatedly observes the current workspace, selects a tool action, reads the resulting observation, and reflects before initiating its next action. The environment records this interaction as τ_i and invokes the official verifier to obtain the scalar reward r_i and diagnostics $\mathbf{v}_i$. Thus, despite differences in tools and verifiers across benchmarks, each adapter implements the same transition

$$\mathcal{E} : (\mathbf{x}, \mathcal{K}_i; \pi) \longmapsto (\tau_i, r_i, \mathbf{v}_i). \tag{4}$$

State & Reward. The controller combines the rollout with the optimization-side reference to form the structured state quadruple

$$\mathbf{s}_i = \big(\mathbf{x}, \tau_i, \tau^\star, \mathbf{v}_i\big). \tag{5}$$

Here, $\mathbf{x}$ has no step subscript because the target instance remains fixed, whereas τ_i and $\mathbf{v}_i$ describe the behavior and diagnostic outcome induced by the current skill. Pairing the current trajectory with $\tau^\star$ exposes not only whether execution failed, but also where its action sequence first departed from a successful strategy; the verifier diagnostics $\mathbf{v}_i$ ground that comparison in the task's actual acceptance criterion. Alongside $\mathbf{s}_i$, the environment returns a separate scalar reward $r_i \in [0, 1]$, which represents binary task success for pass/fail benchmarks and the normalized test pass rate when partial credit is available. Thus, r_i quantifies how well the current skill performs, while $\mathbf{v}_i$ provides the diagnostic evidence needed to explain that outcome. The analysis and ablation of the state are provided in Appendix A.

Critic. A frontier-model critic $\mathcal{C}_\phi$ evaluates the current skill from the state, scalar reward, and prior optimization evidence:

$$\mathbf{g}_i = \mathcal{C}_\phi\big(\mathbf{s}_i, r_i, \mathcal{K}_i, \mathcal{M}_i\big), \tag{6}$$

where $\mathbf{g}_i$ denotes natural-language skill-modification suggestions. The scalar r_i indicates how well the current skill performed, while $\mathbf{v}_i$ explains the verifier outcome. By anchoring its judgment in both signals, the critic contrasts τ_i with $\tau^\star$ to locate the earliest causal divergence. It systematically distinguishes missing procedural guidance from tool misuse, output-contract violations, and non-actionable infrastructure failures. Furthermore, it identifies content that already supports successful behavior and converts the diagnosis into concrete, localized editing instructions. Consequently, the actor receives an evidence-backed repair plan rather than an unconstrained request to rewrite the entire skill. The prompts and ablation of the critic are provided in Appendix B.

Replay Memory. The replay memory $\mathcal{M}_i$ is a compact textual history of completed optimization steps, not a store of raw token-level transitions. As highlighted in Figure 2(a), it retains three forms of reusable evidence, namely failure signatures with verifier diagnostics, critic-summary history, and accepted edits with their observed outcomes. The current critic diagnosis is recorded before actor execution, and once the resulting skill is evaluated, the corresponding edit and outcome are permanently archived in memory. Relevant records are supplied to both $\mathcal{C}_\phi$ and $\mathcal{A}_\theta$. This memory structure discourages repeated failures, protects previously effective behavior, and provides explicit evidence for retaining or rolling back a modification following a performance regression.

Actor. Given the critic's suggestions, a frontier-model actor $\mathcal{A}_\theta$ produces a bounded skill update

$$\begin{aligned} \Delta_i &= \mathcal{A}_\theta\big(\mathbf{x}, \mathcal{K}_i, \tau^\star, \mathbf{g}_i, \mathcal{M}_i\big), \\ \mathcal{K}_{i+1} &= \mathrm{Apply}(\mathcal{K}_i, \Delta_i). \end{aligned} \tag{7}$$

The update Δ_i is realized through four explicit editing operations, namely INSERT, REPLACE, CREATE, and DELETE, applied over the skill bundle. The actor preserves content identified as effective, adds precise behavioral constraints, and may synthesize deterministic task-local helpers when a small-scale LVLM would struggle with a long or error-prone procedure. Reference evidence is distilled strictly into runtime-input-dependent guidance and is never introduced as a direct runtime dependency or a precomputed answer. The prompts and ablation of the actor are provided in Appendix C.

SKILLER repeats this process for $i = 0, \ldots, I-1$, yielding the final composed policy $\mathcal{K}_I = \mathrm{Apply}(\cdots \mathrm{Apply}(\mathcal{K}_0, \Delta_0), \ldots, \Delta_{I-1})$. It also supports batch optimization. For $\mathcal{B} = \{\mathbf{x}^{(n)}\}_{n=1}^N$, parallel environment executions form $\mathbf{S}_i^{\mathcal{B}} = \{\mathbf{s}_i^{(n)}\}_{n=1}^N$ and the empirical verifier objective $\widehat{J}_i(\mathcal{B}) = \frac{1}{N}\sum_{n=1}^N r_i^{(n)}$. Skills may be updated independently as $\{\mathcal{K}_i^{(n)}\}_{n=1}^N$ or tied across related instances by setting $\mathcal{K}_i^{(n)} = \mathcal{K}_i$ and aggregating their evidence before one update. As optimization advances, successive updates Δ_i incorporate progressively finer constraints, including input grounding, task-local computation, and self-validation, that effectively narrow the error-prone behaviors of the small-scale LVLM. Figure 2(b) illustrates this evolution from a generic workflow to a grounded, executable, and self-correcting skill, with increasing verifier performance across steps.

Base model	Method		SkillsBench	SkillLearnBench	SWE-Skills-Bench	GAIA	EarthBench
Qwen3.5-9B	–	No-skill	1.45	23.83	26.20	44.58	59.68
	Closed-source	Human-authored	10.14	30.00	52.00	–	–
		Manus	57.97	27.56	62.40	46.18	71.24
	Open-source	AutoSkill	53.62	25.61	44.10	44.98	71.77
		EvoSkill	50.72	24.22	45.90	48.19	62.90
		SkillX	60.87	27.78	58.80	**49.40**	66.13
		SKILLER (ours)	**73.91**	**32.11**	**82.80**	**49.40**	**76.08**
Qwen3.5-4B	–	No-skill	0.00	24.50	17.60	40.16	52.69
	Closed-source	Human-authored	5.80	28.94	42.70	–	–
		Manus	36.23	31.17	53.40	41.37	**71.51**
	Open-source	AutoSkill	31.88	24.33	35.50	40.56	66.13
		EvoSkill	40.58	22.89	37.30	42.17	67.47
		SkillX	**43.48**	30.22	48.40	**44.18**	65.86
		SKILLER (ours)	42.03	**33.00**	**66.70**	43.78	**71.51**

Table 1: Main results on five benchmarks. All scores are three-run average score, accuracy or pass rate (%). The best result in each model–benchmark group is shown in bold.

Experiments and Analysis

In this section, we present comprehensive empirical comparisons against existing skill-generation methods across five benchmark families, analyze the dynamic evolution process of SKILLER, examine the structural differences between automatically generated and human-authored skills, and evaluate the generation costs of various approaches. Detailed ablation studies of our framework are provided in Appendices A-C.

Experimental Setup

Benchmarks. We evaluate our method across five diverse benchmark families. SkillsBench measures whether agents can use task-paired skills across diverse domains (Li et al. 2026), from which we select a subset of 26 tasks where each instance is resolved using a single skill. SWE-Skills-Bench evaluates skill utility on software-engineering tasks with execution-based tests (Han et al. 2026), where we report performance on 117 instances covered by 10 high-difficulty skills. SkillLearnBench evaluates continual skill generation across 100 verified task instances (Zhong et al. 2026). GAIA covers multi-step information-seeking tasks (Mialon et al. 2024), where we evaluate on 165 tasks from the validation set. EarthBench covers Earth-science and data-processing workflows (Feng et al. 2025), where we conduct testing across 248 evaluation samples. Specifically, GAIA and EarthBench are each treated as a single task where half of the instances are used for generating a skill, allowing us to report additional zero-shot test performance on the remaining samples, whereas for the remaining three benchmarks, a single instance from each task is used to generate the corresponding skill before evaluating across all instances of that task. Further dataset details and evaluation settings are provided in Appendix D.

Models and baseline methods. The target small-scale LVLMs evaluated in our experiments are Qwen3.5-9B and Qwen3.5-4B (Qwen Team 2026), which operate within the skill-executing agent loop of OpenCode (OpenCode 2026). A strong frontier model, specifically GPT-5.4 (OpenAI 2026b), is utilized exclusively during the offline skill-generation phase to serve as the actor, and critic, and all downstream evaluation tokens consumed by the compact models are excluded from the reported generation costs. We compare our framework against standard no-skill execution, three open-source automated skill evolution methods consisting of AutoSkill (Yang et al. 2026), EvoSkill (Alzubi et al. 2026), and SkillX (Wang et al. 2026), as well as skills generated by the closed-source Manus system (Manus 2026). Human-authored skill baselines are reported only when officially provided by the benchmark (Li et al. 2026; Han et al. 2026; Zhong et al. 2026). All reported results represent the average performance across three executions of the same skill, and SKILLER is configured with a five-step reinforcement learning schedule across all tasks. Further baseline methods' details are provided in Appendix E.

Main Results and Analysis

As reported in Table 1, SKILLER achieves superior overall performance across both model scales, demonstrating that natural-language reinforcement learning effectively adapts procedural knowledge to small-scale LVLMs. Notably, human-authored skills and generic skills generated by systems such as Manus frequently yield suboptimal gains or even degrade performance compared to standard automated evolution baselines. This phenomenon directly confirms our core motivation regarding the model-mismatch bottleneck. Skills composed by human experts or strong frontier models implicitly assume expansive reasoning capacities, broad context tolerance, and robust error recovery. When these high-level instructions are injected into a compact model, they frequently induce cognitive overload and trigger severe argument hallucinations. In contrast, by treating the target compact model as the interactive environment, SKILLER diagnoses specific execution failures and injects localized,

Base model	Method	GAIA	EarthBench
Qwen3.5-9B	No-skill	37.00	58.87
	Manus	45.53	70.43
	AutoSkill	47.97	70.16
	EvoSkill	45.12	66.94
	SkillX	42.28	67.20
	SKILLER (ours)	**49.59**	**72.31**
Qwen3.5-4B	No-skill	34.55	50.54
	Manus	33.33	**70.97**
	AutoSkill	33.74	69.89
	EvoSkill	34.15	66.67
	SkillX	28.86	65.59
	SKILLER (ours)	**40.65**	69.62

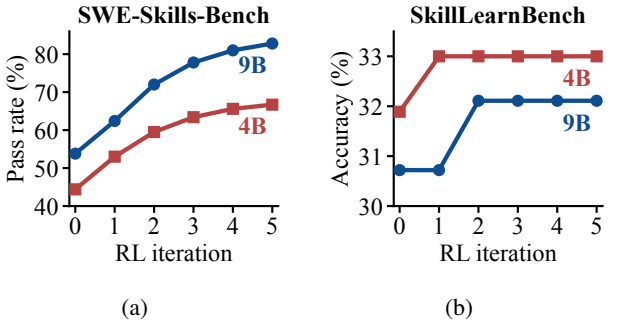
Table 2: Zero-shot results on the rest of GAIA and EarthBench. Scores are three-run average score and accuracy (%).

prescriptive boundaries that small-scale LVLMs can reliably execute.

The advantages of executor-specific policy optimization are most pronounced on benchmarks characterized by complex procedural workflows and strict tool-use conventions, such as SWE-Skills-Bench and SkillsBench. On SWE-Skills-Bench, SKILLER outperforms all open-source and closed-source baselines by substantial margins on Qwen3.5-9B and achieves a dominating pass rate on Qwen3.5-4B. Software engineering tasks strictly penalize unverified file edits, out-of-order tool invocations, and ungrounded argument fabrication. Generic skill generators frequently produce high-level advice that fails to prevent these mechanical errors. Our critic-actor loop directly repairs these failure modes by inserting mandatory verification steps and input-grounding constraints, enabling compact models to execute multi-step engineering pipelines with unprecedented precision.

On benchmarks requiring open-ended information seeking, continual task adaptation, and complex domain workflows, including SkillLearnBench, GAIA, and EarthBench, SKILLER consistently matches or surpasses the strongest baseline methods. While procedural constraints cannot substitute for missing factual knowledge or complex mathematical reasoning during multi-hop retrieval, our learned skills effectively regulate the search process, enforce systematic evidence verification, and prevent premature task finalization. Consequently, SKILLER secures leading accuracy on SkillLearnBench and EarthBench while matching the top-performing baseline on GAIA. These results confirm that natural-language policy optimization provides reliable behavioral control even when task success heavily depends on external data interpretation.

Perhaps most remarkably, the empirical analysis reveals that executor-specific skill optimization can effectively bridge substantial model capacity gaps. As shown in Table 1, Qwen3.5-4B equipped with SKILLER achieves a pass rate on SWE-Skills-Bench that surpasses the performance of the larger Qwen3.5-9B model when deployed with human-authored skills, AutoSkill, EvoSkill, SkillX, or even Manus-generated skills. This finding demonstrates that for structured real-world tasks, an optimized natural-language policy that constrains error-prone tendencies is more valuable than raw parameter scaling. By tailoring procedural control to the exact action distribution of the underlying small-scale LVLM, our framework enables lightweight compact models to outperform unoptimized models of more than twice their parameter scale, establishing a compelling paradigm for cost-efficient agent deployment.



(a) (b)

Figure 3: Learning dynamics through five SKILLER iterations. The vertical axis is the three-repeat average pass rate and accuracy.

Zero-Shot Results and Analysis

To evaluate the out-of-distribution generalization of our learned policies, Table 2 reports the zero-shot transfer performance on the held-out half of GAIA and EarthBench. On Qwen3.5-9B, SKILLER outperforms all baselines by substantial margins, indicating that our optimization loop extracts truly reusable procedural rules rather than overfitting to the surface features of the skill-generation instances. By enforcing systematic evidence gathering and strict output normalization, the generated skills provide robust behavioral scaffolding for small-scale LVLMs across unseen problem distributions. Under tighter capacity constraints on Qwen3.5-4B, SKILLER maintains remarkable resilience by securing the highest accuracy on GAIA and remaining highly competitive on EarthBench. Notably, while closed-source Manus skills slightly lead on the EarthBench evaluation, they severely degrade performance on GAIA, where both Manus and SkillX fall below the standard no-skill baseline. This divergence highlights a fundamental trade-off where verbose domain context frequently triggers cognitive overload and error propagation during complex multi-hop reasoning. In contrast, SKILLER constructs concise and executor-specific procedural boundaries that consistently mitigate hallucinations, confirming that lightweight compact models benefit most from disciplined behavioral control rather than exhaustive context injection.

Learning Dynamics

To understand how natural-language reinforcement learning iteratively refines policy artifacts over time, Figure 3 illustrates the optimization trajectories across five steps on SWE-Skills-Bench and SkillLearnBench. The contrasting curves

Method	Words	TF-IDF	Scripts	LOC	Refs.
Human	1162.52	0.07	1.48	8,819	0.43
Manus	346.57	0.06	2.42	11,168	0.34
AutoSkill	1887.41	0.93	0.54	10,527	0.22
EvoSkill	1256.98	0.89	1.69	7,236	0.45
SkillX	427.48	0.15	2.06	14,701	0.43
SKILLER	534.33	0.07	2.96	15,747	0.47

Table 3: Structural statistics of generated skills on SkillsBench. Evaluated metrics include natural-language verbosity denoted by Words, inter-task instruction diversity measured via TF-IDF cosine similarity, mean script and references count per task, total physical lines of code denoted by LOC.

reveal that the pace of policy convergence is fundamentally governed by the procedural complexity of the target domain. As shown in Figure 3(a), both small-scale LVLMs exhibit substantial and continuous performance gains across all five iterations on SWE-Skills-Bench, demonstrating that complex software engineering workflows benefit from cumulative policy specialization where early steps resolve coarse execution failures and later iterations inject finer constraints like input grounding and self-validation. Conversely, on SkillLearnBench in Figure 3(b), both compact models achieve rapid convergence within the first two optimization steps, indicating that when task bottlenecks stem primarily from output contract formatting or basic tool routing, our critic module quickly isolates the causal errors to establish an effective behavioral boundary.

Structural Analysis of Generated Skills

Table 3 details structural statistics on SkillsBench to uncover fundamental differences between human-authored and automated policies. Open-source baselines consistently produce verbose prompts with elevated TF-IDF similarity, indicating a reliance on repetitive boilerplate templates that induce cognitive overload for compact models. In stark contrast, SKILLER yields concise instructions with remarkably low TF-IDF scores matching human developers, proving our optimization loop synthesizes highly specialized behavioral constraints rather than generic patterns. Furthermore, SKILLER registers the highest number of discrete scripts and total lines of code. This distribution highlights a deliberate paradigm shift that offloads complex procedural reasoning from natural language into deterministic external tools. By minimizing prompt verbosity while maximizing code-level execution capabilities, the generated skills perfectly accommodate the limited context windows of small-scale LVLMs. Generation examples are provided in Appendix F.

Cost-Effectiveness Analysis

Table 4 presents a cost-effectiveness analysis alongside average performance on Qwen3.5-9B, revealing a fundamental trade-off between optimization depth and computational expenditure. While baselines like AutoSkill incur minimal financial costs, their shallow prompt rewriting fails to resolve the model-mismatch bottleneck and yields constrained accuracy. Conversely, SkillX consumes an excessive volume of output tokens and incurs the highest monetary penalty, indicating that open-ended generation without precise environment grounding causes inefficient textual bloat. SKILLER optimally resolves this trade-off by utilizing precise execution feedback to drive highly targeted policy updates. By restricting unnecessary text generation steps, our framework achieves a commanding performance advantage while remaining substantially more cost-efficient than exhaustive generation methods, proving that targeted behavioral alignment offers the highest return on investment for empowering small-scale LVLMs. The cost details of each benchmark and steps are provided in Appendix G.

Method	Input	Output	Cost ($)	Avg. Score
AutoSkill	0.86M	0.02M	2.53	48.02
EvoSkill	0.34M	0.07M	1.95	46.39
SkillX	3.61M	0.37M	14.55	52.60
SKILLER	2.68M	0.15M	8.95	**62.86**

Table 4: Cost analysis and average performance of skill generation on Qwen3.5-9B across five benchmarks. Reported metrics include cumulative strong-model input and output token consumption in millions, monetary cost by GPT-5.4, and the average performance.

Conclusion

In this work, we introduced SKILLER, a natural-language-driven reinforcement learning framework designed to resolve the model-mismatch bottleneck and automatically generate executor-specific skills for small-scale LVLMs. By treating the target compact model's agent loop as an interactive environment and propagating all diagnostic signals entirely through structured text, our framework iteratively refines textual policies based on precise execution feedback without requiring neural weight updates. This language-level policy iteration successfully offloads complex procedural reasoning into deterministic external tools, thereby establishing robust behavioral boundaries that prevent cognitive overload and mechanical errors. Extensive empirical evaluations across five diverse benchmarks demonstrate that SKILLER consistently outperforms both open-source skill evolution methods and closed-source generation systems. Most remarkably, these tailored behavioral constraints enable lightweight compact models to surpass the task-specific performance of unoptimized models at more than twice their parameter scale, establishing a highly cost-effective and scalable paradigm for real-world agent deployment.

References

Alzubi, S.; Provenzano, N.; Bingham, J.; Chen, W.; and Vu, T. 2026. EvoSkill: Automated Skill Discovery for Multi-Agent Systems. arXiv:2603.02766.

Anthropic. 2026a. Agent Skills: Claude Platform Docs. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview. Accessed: 2026-07-29.

Anthropic. 2026b. Overview: Claude Code Docs. https://code.claude.com/docs/en/overview. Accessed: 2026-07-29.

Bi, S.; Wu, M.; Hao, H.; Li, K.; Liu, W.; Song, S.; Zhao, H.; and Zhou, A. 2026. Automating Skill Acquisition through Large-Scale Mining of Open-Source Agentic Repositories: A Framework for Multi-Agent Procedural Knowledge Extraction. arXiv:2603.11808.

Chen, S.; Gai, J.; Zhou, R.; Zhang, J.; Zhu, T.; Li, J.; Wang, K.; Wang, Z.; Chen, Z.; Kaleb, K.; Miao, N.; Gao, S.; Lu, C.; Li, M.; He, J.; and Teh, Y. W. 2026. SkillCraft: Can LLM Agents Learn to Use Tools Skillfully? arXiv:2603.00718.

Feng, P.; Lv, Z.; Ye, J.; Wang, X.; Huo, X.; Yu, J.; Xu, W.; Zhang, W.; Bai, L.; He, C.; and Li, W. 2025. Earth-Agent: Unlocking the Full Landscape of Earth Observation with Agents. arXiv:2509.23141.

Google DeepMind. 2026. Gemma 4 Model Card. https://ai.google.dev/gemma/docs/core/model_card_4. Accessed: 2026-07-29.

Han, T.; Zhang, Y.; Song, W.; Fang, C.; Chen, Z.; Sun, Y.; and Hu, L. 2026. SWE-Skills-Bench: Do Agent Skills Actually Help in Real-World Software Engineering? arXiv:2603.15401.

Huang, Z.; Xu, J.; Yang, Y.; Gong, Z.; Yang, Q.; Tian, M.; Wang, X.; Lv, C.; Gao, X.; Dai, Q.; Liu, B.; Qiu, K.; Yang, X.; Chen, D.; Zheng, X.; and Luo, C. 2026. From Raw Experience to Skill Consumption: A Systematic Study of Model-Generated Agent Skills. arXiv:2605.23899.

Li, X.; Liu, Y.; Chen, W.; You, B.; Di, Z.; He, Y.; Zheng, S.; Choe, K. W.; Sun, J.; Wang, S.; Tao, C.; Li, B.; Zhao, X.; Geng, H.; Wu, X.; Zhou, J.; Chen, X.; Xing, H.; Li, Y.; Zeng, Q.; Wang, D.; Wang, Y.; Ben Chaim, R.; Jiang, P.; Shen, H.; Kong, L.; Liu, X.; Wang, R.; Liu, X.; Li, J.; Lan, X.; Lin, Y.; Ye, W.; He, J.; Li, S.; Zhang, Y.; Gao, Y.; Li, Y.; Ma, Z.; Jing, L.; Wang, T.; Li, K.; Xue, Y.; Lyu, H.; He, Y.; Tian, Y.; Wu, S.; Wang, B.; Gao, Y.; Chen, B.; Liu, L.; Cheng, S.; Bao, J.; Tong, S.; Xu, S.; Zhuo, T. Y.; Ye, T.; Qi, Q.; Li, M.; Liao, L.; Tan, Z.; Shi, C.; Tang, X.; Tankasala, S.; Yuan, B.; Qian, Y.; Tu, J.; Wang, C.; Sun, Y.; Wang, W.; Taylor, A.; Yang, Z.; Guan, C.; Dong, Z.; Zhang, X.; Dillmann, S.; Lee, H.-c.; and Song, D. 2026. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. arXiv:2602.12670.

Liang, Y.; Zhong, R.; Xu, H.; Jiang, C.; Zhong, Y.; Fang, R.; Gu, J.-C.; Deng, S.; Yao, Y.; Wang, M.; Qiao, S.; Xu, X.; Wu, T.; Wang, K.; Liu, Y.; Bi, Z.; Lou, J.; Jiang, Y. E.; Zhu, H.; Yu, G.; Hong, H.; Huang, L.; Xue, H.; Wang, C.; Wang, Y.; Shan, Z.; Chen, X.; Tu, Z.; Xiong, F.; Xie, X.; Zhang, P.; Gui, Z.; Liang, L.; Zhou, J.; Wu, C.; Shang, J.; Gong, Y.; Lin, J.; Xu, C.; Deng, H.; Zhang, W.; Ding, K.; Zhang, Q.; Huang, F.; Zhang, N.; Pan, J. Z.; Qi, G.; Wang, H.; and Chen, H. 2026. SkillNet: Create, Evaluate, and Connect AI Skills. arXiv:2603.04448.

Ling, G.; Zhong, S.; and Huang, R. 2026. Agent Skills: A Data-Driven Analysis of Claude Skills for Extending Large Language Model Functionality. arXiv:2602.08004.

Liu, Y.; Ji, J.; An, L.; Jaakkola, T.; Zhang, Y.; and Chang, S. 2026. How Well Do Agentic Skills Work in the Wild: Benchmarking LLM Skill Usage in Realistic Settings. arXiv:2604.04323.

Ma, Z.; Yang, S.; Ji, Y.; Wang, X.; Wang, Y.; Hu, Y.; Huang, T.; and Chu, X. 2026. SkillClaw: Let Skills Evolve Collectively with Agentic Evolver. arXiv:2604.08377.

Manus. 2026. Manus Skills. https://manus.im/docs/features/skills. Accessed: 2026-07-29.

Mi, Q.; Ma, Z.; Yang, M.; Li, H.; Wang, Y.; Zhang, H.; and Wang, J. 2026. Skill-Pro: Learning Reusable Skills from Experience via Non-Parametric PPO for LLM Agents. arXiv:2602.01869.

Mialon, G.; Fourrier, C.; Swift, C.; Wolf, T.; LeCun, Y.; and Scialom, T. 2024. GAIA: A Benchmark for General AI Assistants. In *International Conference on Learning Representations*.

Ni, J.; Liu, Y.; Liu, X.; Sun, Y.; Zhou, M.; Cheng, P.; Wang, D.; Zhao, E.; Jiang, X.; and Jiang, G. 2026. Trace2Skill: Distill Trajectory-Local Lessons into Transferable Agent Skills. arXiv:2603.25158.

OpenAI. 2026a. Codex Cloud. https://learn.chatgpt.com/docs/cloud. Accessed: 2026-07-29.

OpenAI. 2026b. GPT-5.4 Thinking System Card. https://openai.com/index/gpt-5-4-thinking-system-card/. Published: 2026-03-05. Accessed: 2026-07-29.

OpenClaw Foundation. 2026. Skills: OpenClaw Documentation. https://docs.openclaw.ai/tools/skills. Accessed: 2026-07-29.

OpenCode. 2026. Intro: AI Coding Agent Built for the Terminal. https://opencode.ai/docs/. Accessed: 2026-07-29.

Qiu, L.; Gao, Z.; Chen, J.; Ye, Y.; Huang, W.; Xue, X.; Qiu, W.; and Tang, S. 2026. AutoRefine: From Trajectories to Reusable Expertise for Continual LLM Agent Refinement. arXiv:2601.22758.

Qwen Team. 2026. Qwen3.5: Towards Native Multimodal Agents. https://qwen.ai/blog?id=qwen3.5. Published: 2026-02-15. Accessed: 2026-07-29.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, 68539–68551.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 8634–8652.

Wang, C.; Yu, Z.; Xie, X.; Yao, W.; Fang, R.; Qiao, S.; Cao, K.; Zheng, G.; Qi, X.; Zhang, P.; and Deng, S. 2026. SkillX: Automatically Constructing Skill Knowledge Bases for Agents. arXiv:2604.04804.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291.

Wang, J.; Yan, Q.; Wang, Y.; Tian, Y.; Mishra, S. S.; Xu, Z.; Gandhi, M.; Xu, P.; and Cheong, L. L. 2025a. Reinforcement Learning for Self-Improving Agent with Skill Library. arXiv:2512.17102.

Wang, Z. Z.; Gandhi, A.; Neubig, G.; and Fried, D. 2025b. Inducing Programmatic Skills for Agentic Tasks. In *Conference on Language Modeling*.

Xia, P.; Chen, J.; Wang, H.; Liu, J.; Zeng, K.; Wang, Y.; Han, S.; Zhou, Y.; Zhao, X.; Chen, H.; Zheng, Z.; Xie, C.; and Yao, H. 2026. SKILLRL: Evolving Agents via Recursive Skill-Augmented Reinforcement Learning. arXiv:2602.08234.

Xu, Y.; Li, L.; Sleem, L.; Gentile, N.; Song, Y.; Wang, Y.; Ji, S.; Wu, W.; and State, R. 2026. Agent Skill Framework: Perspectives on the Potential of Small to Medium Language Models in Industrial Environments. arXiv:2602.16653.

Yang, Y.; Li, J.; Pan, Q.; Zhan, B.; Cai, Y.; Du, L.; Zhou, J.; Chen, K.; Chen, Q.; Li, X.; Zhang, B.; and He, L. 2026. AutoSkill: Experience-Driven Lifelong Learning via Skill Self-Evolution. arXiv:2603.01145.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*.

Zhang, B.; Lazuka, K.; and Murag, M. 2025. Equipping Agents for the Real World with Agent Skills. https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills. Published: 2025-10-16. Accessed: 2026-07-29.

Zhang, H.; Fan, S.; Zou, H. P.; Chen, Y.; Wang, Z.; Zhou, J.; Li, C.; Huang, W.-C.; Yao, Y.; Zheng, K.; Liu, X.; Li, X.; and Yu, P. S. 2026a. CoEvoSkills: Self-Evolving Agent Skills via Co-Evolutionary Verification. arXiv:2604.01687.

Zhang, Z.; Shi, K.; Huang, S.; Nie, A.; Zeng, Y.; Zhao, Y.; Fang, Z.; Su, Q.; Qiu, H.; Yang, W.; Ren, Q.; Zou, S.; Huang, W.; Chen, L.; Chen, Z.; and Zhao, F. 2026b. SkillFlow: Benchmarking Lifelong Skill Discovery and Evolution for Autonomous Agents. arXiv:2604.17308.

Zhong, S.; Lu, Y.; Ning, J.; Wan, Y.; Feng, L.; Ao, Y.; Ribeiro, L. F. R.; Dreyer, M.; Ammirati, S.; and Xiong, C. 2026. SkillLearnBench: Benchmarking Continual Learning Methods for Agent Skill Generation on Real-World Tasks. arXiv:2604.20087.

Appendix A: State Analysis and Ablation

We examine the contribution of each component in the structured state $\mathbf{s}_i = (\mathbf{x}, \tau_i, \tau^\star, \mathbf{v}_i)$ using Qwen3.5-9B. Each ablation removes one component while retaining the remaining optimization and evaluation settings. Table 5 reports average performance over three runs under the official metric of each benchmark.

The ablation reveals a clear asymmetry between components that ground a policy update and those that refine it. Removing either the task instance $\mathbf{x}$ or the current trajectory τ_i causes substantially greater damage than removing $\tau^\star$ or $\mathbf{v}_i$, particularly on SkillsBench and SWE-Skills-Bench. A successful reference or verifier outcome cannot by itself identify a useful edit: $\mathbf{x}$ specifies what the executor must satisfy, whereas τ_i exposes how the current skill actually shapes its behavior. Together, they turn feedback into an error signal that is conditioned on the task rather than generic advice.

The dependence on this grounding pair is strongest for benchmarks with strict tool sequences, executable artifacts, and output contracts. GAIA degrades less sharply under every ablation. This pattern suggests that information seeking tasks can retain partial utility from broadly applicable search procedures even when one state signal is absent. In contrast, structured skill and software tasks require the editor to connect a precise failure to a precise action or artifact, which makes omissions from the state much harder to compensate for.

The remaining two components play complementary refinement roles. Across all three benchmarks, removing the reference trajectory is more harmful than removing verifier diagnostics. This result indicates that evidence about the execution process offers a richer target for localized edits than outcome feedback alone. Nevertheless, the consistent loss without $\mathbf{v}_i$ shows that a plausible execution path is insufficient unless it is anchored to the benchmark's acceptance criterion. The full state is effective because it combines task intent, observed behavior, a positive execution reference, and correctness grounded in verifier feedback rather than relying on any single feedback source.

Variant	SkillsBench	SWE-Skills-Bench	GAIA
SKILLER	**73.91**	**82.80**	**49.40**
w/o $\mathbf{x}$	1.45	20.51	39.76
w/o τ_i	2.44	24.87	44.58
w/o $\tau^\star$	58.71	62.16	44.98
w/o $\mathbf{v}_i$	60.87	64.50	46.20

Table 5: State component ablation using Qwen3.5-9B as the executor. Scores are means over three runs in percentage points under each benchmark's official metric, with higher values indicating better performance.

Appendix B: Critic Prompt Analysis and Ablation

The critic prompt implements four operations that mirror the critic module in Figure 1 of the main paper. It evaluates the current execution, compares the observed and reference trajectories, locates the earliest causal error, and generates a bounded skill modification. We remove each operation separately while retaining the remaining prompt instructions and experimental settings. Table 6 reports performance with Qwen3.5-9B.

The dominant failure after removing Generation suggests that translation is the critic's decisive function. Evaluation, comparison, and localization can expose a defect, but the learning loop improves only when the diagnosis becomes a bounded edit to the textual policy. Without this operation, rollout evidence remains descriptive and cannot alter the behavior that produced the failure.

Variant	SkillsBench	SWE-Skills-Bench	GAIA
SKILLER	**73.91**	**82.80**	**49.40**
w/o Evaluate	60.87	68.20	46.40
w/o Compare	60.87	64.50	46.40
w/o Locate Error	58.71	64.46	45.00
w/o Generation	36.44	24.90	42.15

Table 6: Critic prompt ablation using Qwen3.5-9B as the executor. Each variant removes one critic operation, and scores follow the official metric of each benchmark, with higher values indicating better performance.

Comparison and error localization provide a second layer of control. The official verifier already supplies a coarse outcome signal, which may partially compensate for the absence of an explicit evaluation instruction. It cannot, however, explain which decision caused the outcome. Comparing the observed path with a successful reference narrows the search to meaningful divergences, and localizing the earliest error discourages the actor from patching downstream symptoms. This distinction is especially important for software tasks, where a late test failure can originate from an earlier choice of file, tool, or verification procedure.

The smaller sensitivity on GAIA suggests a boundary of prompt based skill repair. One plausible explanation is that information seeking tasks may fail because relevant external evidence was not found, rather than because a visible procedure was applied incorrectly. A critic can reshape the search policy, but it cannot supply missing facts. Structured skill and software tasks expose more repeatable causal traces, which give comparison and localization greater leverage. The complete prompt is effective because it composes a verdict, a contrastive reference, a causal diagnosis, and an executable update into one feedback path.

Figure 4 presents the SKILLER critic prompt used for SkillsBench using the four operations examined by the method and ablation.

SKILLER Critic Prompt

Shared role and inputs

You are the Critic in a natural-language RL framework that trains ONE Agent Skill at a time for a single SkillsBench task. The executor is Qwen3.5-9B. You receive the task instruction, the natural-language reference trajectory, the current skill bundle, the latest executor trace with the official verifier verdict, and a short history poll.

1 Evaluate the current skill by x, v_i, r_i

Purpose: judge task success and audit policy validity

The oracle is a reference for the critic and actor, not a runtime dependency for the executor. Audit the current skill for direct oracle use before diagnosing ordinary failures. Treat references to solve.sh, oracle wrappers, pasted solver code, and precomputed answer artifacts as critical defects. If the verifier passes and no such defect exists, default to no substantive edits. Assess task alignment, instruction clarity, script quality, tool protocol, efficiency and hallucination, and regression risk.

2 Compare two trajectories (τ_i, τ^*)

Purpose: contrast observed execution with canonical evidence

Treat the reference trajectory as authoritative for algorithmic intent, constants, paths, and output schema. Walk the executor trace step by step and compare it with the canonical algorithm. Identify the first divergence, then quote the executor's bad output and the corresponding correct algorithmic step. Use only the task instruction, reference trajectory, current skill, executor trace, and verifier verdict as sources of truth. Do not invent oracle numbers or infer script drift without concrete lines.

3 Locate the earliest causal error

Purpose: isolate the earliest causal failure and protect working behavior

Cite the executor tool call with its step index and an observed output snippet. Identify the line in SKILL.md that should have prevented the failure, and reference exact paths, snippets, and tool names. Preserve the sections that produced correct partial steps unless they violate the non-cheating policy. If a passing behavior regressed, identify the disruptive change to undo. Separate the earliest causal error from downstream symptoms so that the next update targets a repairable decision.

4 Generate skill modification suggestions $\mathbf{g}_i$

Purpose: convert diagnosis into a bounded and executable skill update

Emit mechanically applicable INSERT, REPLACE, CREATE, or DELETE instructions. Prefer surgical insertions, keep replacements under four lines, and target no more than 30 changed lines. Never rewrite the whole skill, rename files, or move content between SKILL.md and references. Use the framework tool vocabulary in examples. Return one JSON object containing the reward paragraph, six reward dimensions, one failure signature, and a summary of the top two or three edits.

Figure 4: SKILLER critic prompt used for SkillsBench and organized by its four operations. Each colored panel uses a distinct visual treatment and begins with the purpose of the corresponding operation. The prompt content is drawn from the released implementation and grouped to remove repeated instructions while preserving the operational constraints.

Appendix C: Actor Prompt Analysis and Ablation

The actor prompt implements four operations that mirror the actor module in Figure 1 of the main paper. It applies bounded edit operations, synthesizes task local helper scripts, preserves effective skill content, and emits a complete skill update. We ablate the first three operations separately while retaining the remaining prompt instructions and experimental settings. The final operation defines the update interface itself, so removing it would disable the learning step rather than isolate a comparable prompt component. Table 7 reports performance with Qwen3.5-9B.

Variant	SkillsBench	SWE-Skills-Bench	GAIA
SKILLER	**73.91**	**82.80**	**49.40**
w/o Operations	68.45	77.90	44.98
w/o Scripts	55.34	52.30	43.68
w/o Preserve	59.18	68.20	42.59

Table 7: Actor prompt ablation using Qwen3.5-9B as the executor. Each variant removes one actor operation, and scores follow the official metric of each benchmark, with higher values indicating better performance.

The strongest degradation on SkillsBench and SWE-Skills-Bench occurs when the actor cannot synthesize helper scripts. This pattern identifies executable abstraction as a central bridge between language feedback and reliable action. Long procedures embedded directly in a skill remain vulnerable to truncation, format drift, and inconsistent tool use by a small executor. A task local helper moves deterministic computation into a reusable artifact, leaving the skill to specify when and how that artifact should be invoked. The larger effect on structured skill and software tasks is consistent with their strict output contracts and repeatable computation paths.

Preservation governs a different failure mode. A critic observes one rollout, so its repair signal is necessarily local to the latest error. Without an explicit instruction to retain effective content, the actor can overfit that signal by replacing instructions that supported previously correct behavior. This stability constraint is particularly important on GAIA, where broadly useful search and verification routines must survive updates driven by heterogeneous tasks. Preservation therefore protects accumulated competence that is not visible in the current trajectory.

Bounded operations yield a smaller but consistent contribution. Their role is to control the scope of plasticity rather than introduce new procedural knowledge. Exact insertion, replacement, creation, and deletion primitives make critic feedback mechanically actionable and reduce unintended edits to unrelated files. Together, the three operations form a coherent update policy. Helper synthesis adds executable competence, preservation retains established competence, and bounded editing mediates the tradeoff between the two.

Figure 5 presents the SKILLER actor prompt used for SkillsBench using the four operations represented by the method.

Appendix D: Benchmark Task Selection and Data Splits

We use *task* to denote a benchmark unit associated with a skill and *instance* to denote an individual sample evaluated under that task. The SkillsBench subset contains 26 tasks, each selected because it can be solved with a single skill. For SWE-Skills-Bench, we retain the tasks for which the original benchmark reports that adding a skill changes performance, including both gains and declines. This criterion yields 10 tasks and 117 instances. The subset is designed to study settings in which skill use has a measurable behavioral effect, and its results should not be interpreted as estimates over every task in the original benchmark. SkillLearnBench uses the official configuration and verifier.

GAIA and EarthBench are each treated as one task, and each dataset sample is treated as an instance. For GAIA, we stratify the samples by the three difficulty levels and randomly assign half of each stratum to skill generation. The resulting generation split contains 83 instances. The remaining instances form the zero-shot test split. For EarthBench, we stratify the samples by the three task categories and apply the same procedure. The generation split contains 124 instances, and the remaining half forms the zero-shot test split.

Stratification preserves the benchmark composition across the generation and test partitions. No held-out instance is used to update the skill, so the zero-shot results measure whether a skill induced from one subset transfers to unseen instances from the same benchmark.

Appendix E: Method Implementation Details

The implementation assigns each task an independent skill bundle and experience buffer, then initializes the bundle by refining the baseline skill with the task instruction and reference trajectory while retaining unchanged local files. The released SkillsBench configuration runs three optimization rounds, limits each executor rollout to 30 tool steps, and uses sampling temperatures of 0.2 for the executor and 0.3 for the actor and critic. During each round, the executor runs the current skill in the task environment and the benchmark verifier supplies the success signal, after which the critic conditions on the task instruction, reference trajectory, current skill file tree, rollout state, verifier result, and three most recent history records to produce structured natural language feedback. The actor uses this feedback to apply bounded edits and return the complete contents of modified files, and the merged update is rejected if it directly uses oracle solver content. The system stores skill snapshots before and after every round and restores the most recent passing snapshot when a later update causes failure, which prevents subsequent optimization from discarding an already successful skill.

Appendix F: Qualitative Skill Evolution

We examine three consecutive saved versions of the `springboot-tdd` skill from SWE-Skills-Bench. This ex-

SKILLER Actor Prompt

Shared role and inputs

You are the Actor in a natural-language RL framework that maintains ONE Agent Skill for one SkillsBench task. You receive the task instruction, reference trajectory, critic reward and edit plan, current skill files, and recent history. The Qwen3.5-9B executor follows the revised SKILL.md through the framework tools.

1 Apply bounded operations by suggestions $\mathbf{g}_i$

Purpose: translate critic suggestions into small and controlled file edits

Treat the critic's edit plan as bounded actions. Apply each INSERT, REPLACE, CREATE, or DELETE instruction mechanically at the exact location named, only when it satisfies the rule against direct oracle use. Do not refactor, reorder, rename, or clean up unrelated content. Keep the total diff within 30 lines and prioritize the top two or three high leverage edits when the plan is larger.

2 Synthesize task-local helper scripts

Purpose: externalize deterministic procedures for the small executor

When a task needs custom code, bundle a distilled task local helper under scripts. The helper must read the actual task inputs and write the required output. Invoke it from SKILL.md with one short run_shell call. Do not embed a large payload in a tool call or copy the oracle solver. Preserve necessary constants, formulas, and schemas in a clean implementation.

3 Preserve effective content in $\mathcal{K}_i$

Purpose: retain verified behavior while introducing a localized repair

Keep every line marked DO NOT modify unchanged unless it violates the rule against direct oracle use. Preserve working steps, existing phases, the YAML frontmatter, and the skill name. Omit unchanged files from files_to_write so that they remain intact. After a pass with no substantive defect, return no edits. Apply at most one short optional hardening edit.

4 Produce skill update $\Delta_i = \mathcal{A}_\theta(\mathbf{x}, \mathcal{K}_i, \tau^\star, \mathbf{g}_i, \mathcal{M}_i)$

Purpose: emit a complete and executable update to the skill bundle

Return exactly one JSON object containing summary, target_skill_name, files_to_write, files_to_delete, and experience_entry. Include the complete contents of every file that is modified or created. Keep omitted files unchanged and delete only explicitly listed files. The revised SKILL.md must retain concrete paths, executable tool calls, exact output contracts, and concise recovery guidance. Emit no markdown or prose outside the JSON object.

Figure 5: SKILLER actor prompt used for SkillsBench and organized by its four operations. Each colored panel uses a distinct visual treatment and begins with the purpose of the corresponding operation. The prompt content is drawn from the released implementation and grouped to remove repeated instructions while preserving the operational constraints.

Method or stage	R1	R2	R3	Avg.	Acc. (%)	Input (K)	Output (K)	Total (K)
AutoSkill	28	27	23	26.00	25.61	1352.98	30.31	1383.29
EvoSkill	23	25	24	24.00	24.22	366.65	21.62	388.27
SkillX	26	27	28	27.00	27.78	754.82	104.16	858.98
SKILLER Step 1	30	27	31	29.33	30.33	162.22	73.66	235.88
SKILLER Step 2	31	30	31	30.67	**32.11**	185.34	73.93	259.27
SKILLER Step 3	28	25	33	28.67	30.17	184.98	73.92	258.90
SKILLER Step 4	31	30	27	29.33	30.06	205.06	83.82	288.88
SKILLER Step 5	29	29	31	29.67	30.17	180.16	84.67	264.83

Table 8: Detailed SkillLearnBench results and skill generation tokens using Qwen3.5-9B as the executor. R1, R2, and R3 are the numbers of passed instances out of 100 in three repeated runs, and Avg. is their mean. Acc. is the official accuracy averaged over the same runs and expressed as a percentage. Input, Output, and Total report strong model token consumption in thousands for each baseline or SKILLER stage.

ample was selected after comparing the recorded skills from all five benchmarks because it contains substantive and interpretable updates at both transitions while preserving the same overall workflow. Figure 6 summarizes the semantic changes in the saved rounds, which we refer to as steps in this appendix.

The first transition changes the skill from a broad procedural guide into a budget-aware control policy. Absolute root anchoring, finite exploration limits, bounded evidence collection, and an explicit file checklist define stopping conditions for actions that were previously open ended. These changes reveal that an important source of failure is not missing domain knowledge, but the allocation of a limited interaction budget. Encoding the observed failure as a reusable constraint reduces the chance that a later executor repeats the same navigation loop while leaving the task workflow available for new instances.

The second transition moves the repair boundary from action selection to artifact consistency. Exact database targets, observed method signatures, cross-file dependencies, and valid tool arguments become preconditions for an edit. This distinction matters because a locally plausible Java class can still fail when its imports, repository methods, SQL profile, or tool call does not match the surrounding project. The skill therefore converts trajectory feedback into interface contracts that can prevent an entire family of related failures rather than patching only the latest output.

Across all three steps, the seven execution phases remain unchanged while their transition and completion criteria become more precise. This pattern shows how bounded language-level updates can balance plasticity with retention. New failure evidence is accumulated as local policy constraints, whereas previously useful structure remains available to the executor. The example is qualitative rather than a standalone measure of average behavior, but it exposes the mechanism through which repeated interaction can produce a more executable and regression-resistant skill.

Appendix G: SkillLearnBench Cost and Learning Dynamics

Table 8 reports the detailed SkillLearnBench results for Qwen3.5-9B together with the strong model tokens used to generate each skill or update. We omit No-skill and Human-authored because they do not invoke a strong model for skill generation, and we omit Manus because its token usage is unavailable.

The comparison separates generation volume from skill effectiveness. AutoSkill consumes the most tokens among the reported baselines but does not obtain the strongest baseline accuracy, while SkillX performs better with a smaller token budget. Every reported SKILLER step also exceeds the strongest baseline accuracy. This pattern indicates that generation volume alone does not determine whether the resulting instructions provide useful behavioral control for the target executor.

The SKILLER trajectory is nonmonotonic. Step 2 establishes the best checkpoint, while later updates retain much of the gain but do not improve on that checkpoint. This behavior is consistent with localized edits that repair a sampled failure while occasionally narrowing instructions that remain useful for other instances. It also motivates the snapshot and rollback mechanism described in Appendix E because the final update need not be the most transferable policy.

The five update stages consume similar token budgets despite producing different evaluation outcomes. In particular, the stage with the largest token usage does not yield the highest accuracy. The marginal utility of an update therefore depends more on the information contained in verifier grounded feedback than on raw generation volume. Because the SKILLER rows report stage level rather than cumulative token usage, this table characterizes the efficiency of individual updates rather than the total expense of the full optimization trajectory. The optimal stopping step can also vary with the benchmark and executor.

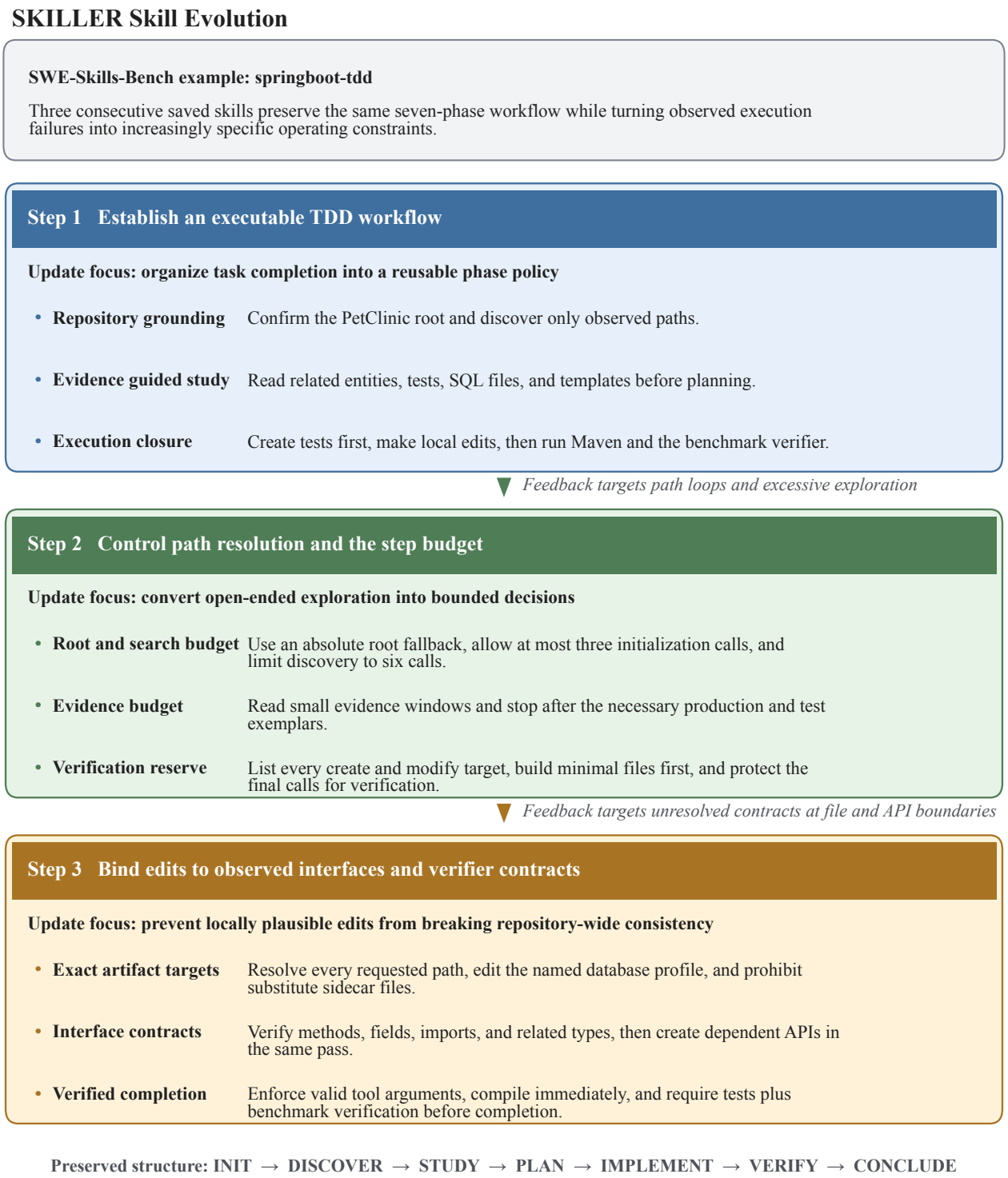


Figure 6: Evolution of the `springboot-tdd` skill on SWE-Skills-Bench across three SKILLER steps. Each panel summarizes the semantic changes introduced by one saved skill version rather than presenting a literal text comparison. The seven-phase workflow is retained while feedback progressively constrains path discovery, edit planning, artifact consistency, and verification.