LEARNING TO EVOLVE: SKILL–POLICY CO-EVOLUTION FOR SELF-EVOLVING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent self-evolution has garnered increasing attention, where LLM agents improve their policies by extracting and internalizing reusable skills from their own interactions. However, we find that current methods can exhibit *evolutionary stagnation*, where policy improvement gradually diminishes over successive evolution rounds as skill-extraction quality degrades and skill extraction becomes increasingly misaligned with policy optimization. In this paper, we introduce a skill-policy co-evolution method (SCALE) that evaluates candidate skills by their effects on long-term policy evolution. Specifically, SCALE learns a skill-conditioned velocity field to model how each skill transforms the trajectory distribution and constructs an evolution-aware reward based on the Wasserstein distance to a reward-reweighted target distribution. It enables the agent to continuously extract skills aligned with its evolving policy and sustain improvement across evolution rounds. We evaluate SCALE on nine challenging benchmarks spanning scientific reasoning, coding, agentic tasks, GUI control, and real-world environment learning. The results show that SCALE consistently surpasses existing methods across diverse model scales and exhibits strong active learning capability from interaction.

1 INTRODUCTION

Recent advances in agentic RL have demonstrated transformative impact across a wide range of real-world domains, where agents continuously interact with dynamic environments and improve their decision-making policies (Zhu et al., 2026a; Geng et al., 2026). Yet, existing methods still struggle in long-horizon and subjective tasks, such as AI coding (Wang et al., 2025; Kong et al., 2025; Zhang et al., 2025a), web navigation (Deng et al., 2023; Kong et al., 2026), and embodied control (Zhang et al., 2025b; Liu et al., 2026), because it is difficult to accurately evaluate the value of each intermediate step or generated action, causing agents to over-optimize superficial reward signals, exploit imperfect verifiers, and converge to behaviors that appear successful under the feedback but fail to reflect genuine task-solving ability (Sun et al., 2026; Chen et al., 2026).

Very recently, research has shifted towards *agentic self-evolution*, where agents can enhance their problem-solving capability without relying on external knowledge or guidance (Yu et al., 2025; Xia et al., 2025; Yue et al., 2026; Zhao et al., 2026; Yang et al., 2026b). In a typical self-evolving framework, an agent interacts with the environment and summarizes reusable skills from its own trajectories. Equipped with these skills, agent can generate dense training guidance to refine its policy, without depending on human annotations or external teachers (Zhang et al., 2026b; Ju et al., 2026; Cheng et al., 2026). Such self-improving paradigms are expected to continuously improve task-solving capability by accumulating interaction experience and correcting previous mistakes, and have shown great potential for building agents that can autonomously evolve their capabilities, including agentic search (Jin et al., 2025; Chen et al., 2025) and scientific discovery (Zhu et al., 2026b; Du et al., 2026).

Albeit achieving promising performance, we find that current self-evolution methods typically suffer from *evolutionary stagnation*, that is, the effectiveness of self-evolution gradually diminishes as the policy improves (as illustrated in Fig. 1), easily trapping the agent in a plateau where further self-improvement becomes increasingly difficult. The underlying reason primarily lies in *insufficient skill extraction*. On the one hand, self-generated skills are often low-informative because current general models are mainly trained goal-oriented rather than skill-oriented (Li et al., 2026a). As a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

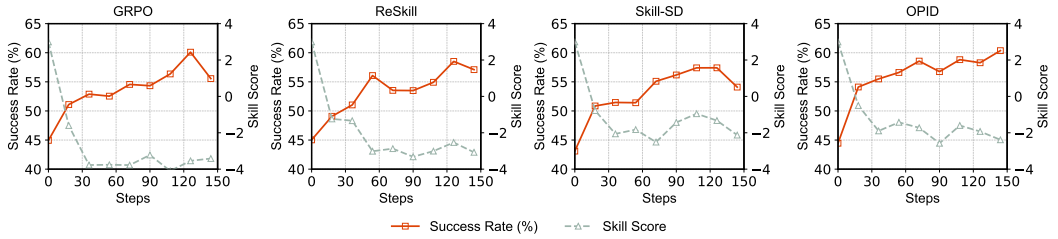


Figure 1: Efficiency of self-evolution and skill extraction evaluation in SkillsBench (Li et al., 2026a).

result, the generated skills may only capture superficial patterns that help the current trajectory, but fail to encode general principles that can serve as reliable contexts for evolution. On the other hand, the agent's skill extraction capability may gradually degrade during policy improvement. This is because policy updates cause the reasoning patterns to gradually become specialized toward executing the desired behavior without extra context, weakening the ability to identify skills that signal further opportunities for improvement (see dashed lines in Fig. 1).

This paper aims to answer: *"how to enhance an agent's skill extraction capability for self-evolution?"* A straightforward solution for skill extraction training is to maximize the performance gain induced by the skills. It, however, suffers from *skill-policy misalignment*, where the skill-extraction gradient conflicts with the policy-optimization gradient, potentially driving the policy away from improvement (see details in Section 4). This is because the model may improve the skill-induced performance gain by degrading the performance of current policy, rendering the training process unstable for policy improvement. Instead, a more reasonable approach is to train skill extraction from a perspective of evolution, such that the agent can learn to continuously extract skills that support sustained policy improvement. Although promising, it is highly challenging to obtain an evolved policy that faithfully reflects the effect of the extracted skill and evaluate it without external guidance .

To tackle this challenge, we introduce Skill–Policy Co-Evolution via Flow-Based Assessment for Long-Term Self-Evolution (SCALE), which continuously extracts reusable skills and internalizes them to sustain long-term improvement, capable of maintaining skill extraction capability throughout policy optimization and effectively leveraging environmental interaction for test-time adaptation. First, we introduce a skill-induced velocity field, which characterizes the distributional effect of extracted skills, thereby enabling us to simulate policy evolution without repeated rollouts. To accurately evaluate the utility of skills, we design an evolution-aware reward that measures how well the skill can produce meaningful long-term policy evolution, effectively reducing skill-policy misalignment. Furthermore, we develop a skill–policy co-evolution method, which iteratively trains skill extraction and skill internalization, leading to continual and steady policy improvement.

We systematically evaluate the proposed method across 9 challenging environments, including scientific reasoning, coding, and GUI control. The results demonstrate that SCALE significantly improves self-evolution performance across all environments, consistently outperforming existing methods by 30% to 50% in terms of relative improvement. Moreover, to the best of our knowledge, SCALE is the first self-evolution framework demonstrated to scale to models with up to 397B parameters, highlighting its scalability to large-scale agentic models. Further, SCALE preserves its skill extraction capability throughout policy optimization, continuously producing informative and reusable skills that provide a reliable foundation for sustained evolution. In addition, we observe that SCALE more effectively leverages environmental interaction at test time to actively learn from accumulated feedback, summarize new experience, and progressively refine its subsequent decisions.

2 Related Work

Agent Self-evolution. Agent self-evolution has recently attracted significant attention, as it enables agents to improve their problem-solving capability by learning from their own interactions, feedback, and generated trajectories (Wu et al., 2025; Ye et al., 2026; Fan et al., 2026). One line of work focuses on reward-driven policy optimization, where agents use internal or external rewards to reinforce successful behaviors and suppress failed ones (Sun et al., 2026; Chen et al., 2026; Hua et al., 2026). However, such methods usually require task-specific reward engineering and primarily optimize policy behavior under scalar feedback. Beyond direct policy optimization, some recent methods

propose to extract reusable skills from trajectories or feedback (Yang et al., 2026c; Xia et al., 2026; Zhang et al., 2026a; Lu et al., 2026b; Yang et al., 2026a). He et al. (2026) propose to create new skills from interaction, so that gathered skills can be aligned with the evolving policy. Zheng et al. (2026) utilize an extra LLM judge to score the extracted skills for self-reflection training. Wu et al. (2026) utilize SFT to equip the agent with the capability to analyze completed trajectories. Albeit with promising results, these methods primarily focus on improving the quality of extracted skills at the current policy stage, without explicitly considering how the skill extraction capability itself should co-evolve with the continuously improving policy. Consequently, the agent may improve its current policy while failing to enhance or preserve its ability to extract informative skills, thereby limiting the efficiency of long-term self-evolution.

Model Distillation. Model distillation refers to training the student to mimic the teacher's behavior, and has recently attracted significant attention as it provide dense training signal for stable policy optimization (Agarwal et al., 2024; Xu et al., 2024; Gu et al., 2024; Xiao et al., 2026). However, these approaches largely rely on powerful teacher model to transfer valuable knowledge, which inherently imposes an upper bound on the performance achievable during training. More recently, researches have attempt to overcome this limitation by utilizing LLMs' powerful mechanism for using feedback (Snell et al., 2022), where a model conditioned on additional information acts as a teacher for a version of itself without that information (Yang et al., 2024; Zhao et al., 2026). Shenfeld et al. (2026) rephrase the answer into an extra context to construct the teacher model for knowledge transfer without modifying the model parameters. Qi et al. (2025) utilize on-policy sample from student and internalize these context-induced knowledge via soft distillation, offering a more robust alternative to traditional fine-tuning. Hübotter et al. (2026) incorporate environmental feedback (i.e. code execution results) as additional input to the teacher, which enables the teacher to extract and transfer the reasoning induced by reflection. Li et al. (2026b) aim to extract more value from environmental textual feedback via variational inference, ensuring the teacher's assessment capabilities scale alongside the student's reasoning. Albeit with promising results, these methods cannot be directly applied to agentic tasks, as such tasks typically lack a definitive ground-truth answer or clear environmental feedback, and their multi-step nature makes it difficult to obtain reliable, immediate correctness signals from intermediate states.

3 PRELIMINARIES

Agentic Self-Evolution In agentic self-evolution, agent interacts with the environment, summarizes reusable skills and internalizes these capabilities to enrich agent's understanding of the dynamic. At each discrete time step $t = 1, 2, \ldots, T$, the agent observes the current state $s_t \in \mathcal{S}$ and produces a textual action $a_t \in \mathcal{V}^n$, where $\mathcal{V}$ denotes the token vocabulary and n is the maximum generation length. The environment then returns a scalar reward $r_t \in [0, R_{max}]$ along with the next state s_{t+1}. An episode is thus represented as a trajectory:

$$\tau = \{(s_1, a_1, r_1), \ldots, (s_T, a_T, r_T)\}. \tag{1}$$

The agent's behavior is generated from a policy $\pi_\theta(a_t|s_t)$, parameterized by θ, which defines a distribution over textual actions conditioned on the current state s_t. We utilize trajectory-level reward $R(\tau) \in \{0, 1\}$, as rewards are sparse and terminal in practical agentic environments. Therefore, the learning objective is to maximize the expected trajectory reward:

$$\max_\theta J(\pi_\theta) \doteq \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \tag{2}$$

Under this formulation, the agent interacts with the environment and samples a group of trajectories, denoted as $\{\tau_i\}_{i=1}^G$. Then, we prompt the agent with instruction p_{ext} (see details in Section B.5) to summarize reusable skill c from the sampled trajectories:

$$c \sim \pi_\theta(\cdot|p_{\text{ext}}, \{\tau_i\}_{i=1}^G). \tag{3}$$

With the extracted skill, we then internalize it into model's parameters via on-policy distillation, which provides token-level guidance for efficient training (Tang & Munos, 2025). To be specific, current studies (Xiao et al., 2026) approximate the gradient as:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\Big[\sum_{l=1}^{L}(\log \pi_\theta(a_l|s, a_{<l}) - \log \pi_\theta(a_l|s, c, a_{<l}))\nabla_\theta \log \pi_\theta(a_l|s, a_{<l})\Big]. \tag{4}$$

Through repeated interaction, skill extraction, and internalization, the agent progressively transforms reusable internal feedback and skills into persistent parametric knowledge, enabling continual policy improvement in dynamic environments.

Flow Matching Flow matching provides a simulation-free framework for learning continuous transformations between probability distributions (Lipman et al., 2023). Let $p_0(x)$ denote a source distribution and $p_1(x)$ denote a target distribution over a continuous space $\mathcal{X} \subseteq \mathbb{R}^d$. The goal is to learn a time-dependent velocity field $v_\phi(x,t) : \mathcal{X} \times [0,1] \to \mathbb{R}^d$ that transports samples from p_0 to p_1 through the ordinary differential equation

$$\frac{dx_t}{dt} = v_\phi(x_t, t), \quad x_0 \sim p_0, \quad x_1 \sim p_1. \tag{5}$$

The distribution of x_t at time t is denoted by p_t, where p_t defines a probability path satisfying $p_{t=0} = p_0$ and $p_{t=1} = p_1$. Therefore, learning the velocity field v_ϕ amounts to learning how probability mass should move from the source distribution to the target distribution. Flow matching trains v_θ by matching it to a prescribed target velocity field $u_t(x)$ associated with a predefined probability path $\{p_t\}_{t\in[0,1]}$. The ideal flow matching objective is:

$$\mathcal{L}_{\mathrm{FM}}(\phi) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, x_t\sim p_t}\left[\|v_\phi(x_t,t) - u_t(x_t)\|_2^2\right]. \tag{6}$$

4 CHALLENGE

In this section, we emphasize the difficulty of evaluating the extracted skill during self-evolution. First, we introduce a naive objective, which maximizes the performance gain caused by the skill c:

$$\max_\theta \mathbb{E}_{\tau_i\sim\pi_\theta, c\sim\pi_\theta(\cdot|p_{\mathrm{ext}},\{\tau_i\}_{i=1}^G)}[J(\pi_\theta(\cdot|\cdot,c)) - J(\pi_\theta)]. \tag{7}$$

Although Eq. (7) directly optimizes the improvement brought by the extracted skill, it is extremely difficult to optimize in practice. As illustrated in Fig. 2, we empirically measure the cosine similarity between the gradients of skill extraction and policy distillation throughout training, and observe that the two gradients become increasingly conflicting as the policy improves.

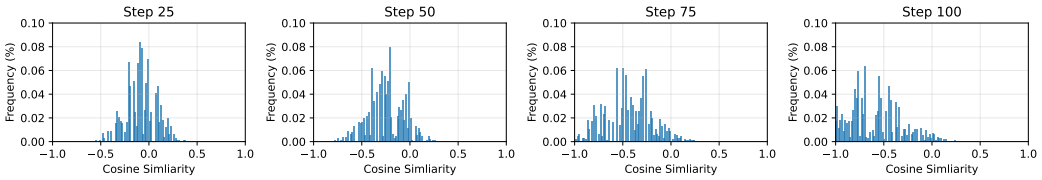


Figure 2: Distribution of cosine similarity between skill gradient and policy gradient at different training step. We experiment on EdgeBench (Zhu et al., 2026a) with Qwen3.5-397B.

The reason for this conflict is that the objective only evaluates the performance difference after conditioning on the extracted skill, providing limited guidance for identifying and optimizing reusable behavioral changes from the interaction experience. The limited guidance makes it difficult to reliably optimize skill extraction and leaves room for an easier optimization shortcut: increasing the skill improvement by degrading the original (skill-free) policy. Consequently, skill extraction can favor updates that are detrimental to the policy being optimized, leading to conflicting gradient directions between skill extraction and skill internalization.

To this end, rather than evaluating a skill by its performance improvement over the current policy, we assess whether the policy evolved by the skill π_c approaches a desirable target policy π_{tar}:

$$\max_\theta \mathbb{E}_{\tau_i\sim\pi_\theta, c\sim\pi_\theta(\cdot|p_{\mathrm{ext}},\{\tau_i\}_{i=1}^G)}[-\mathcal{D}(\pi_{\mathrm{tar}}, \pi_c)]. \tag{8}$$

This formulation directly evaluates the quality of the skill-induced policy against a desirable target, such that degrading the current policy cannot increase the skill objective. However, realizing this objective faces two key challenges. First, accurately obtaining the evolved policy π_c is non-trivial, as it should characterize the policy resulting from sufficiently evolving under the extracted skill rather than merely its immediate effect on the current policy. Second, an explicit target policy π_{tar} is generally unavailable in self-evolution, since there is no external teacher provided.

5 SKILL EXTRACTION AND POLICY OPTIMIZATION CO-EVOLUTION

In this section, we introduce a skill extraction and policy co-evolution method that can consistently improve agent's task-solving capability through interaction with the environment, while preserving its skill extraction capability. Fig. 3 illustrates the full pipeline, which consists of three stages: skill-induced velocity field, skill extraction evaluation, and policy optimization.

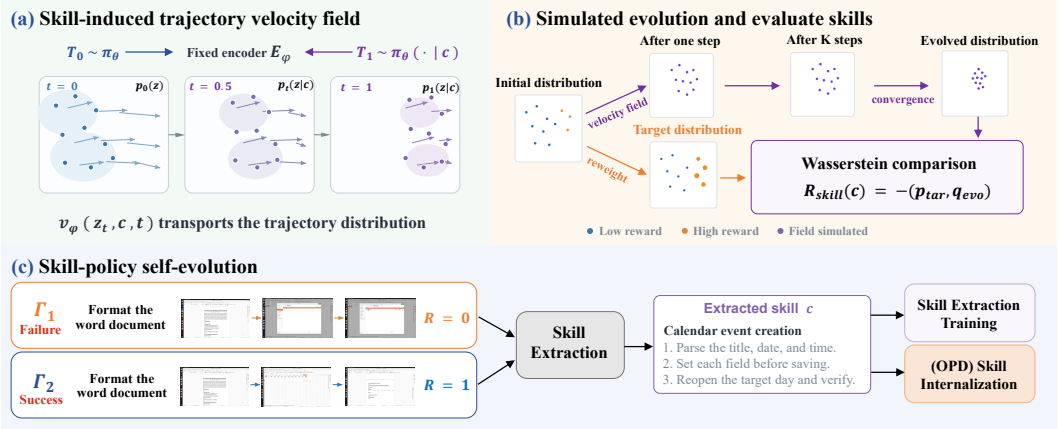


Figure 3: Overview of SCALE.

5.1 SKILL-INDUCED TRAJECTORY VELOCITY FIELD

As the policies involved in Eq. (8) are not directly accessible, we reformulate it from a distributional perspective, where a generated skill should induce a meaningful transformation of the trajectory distribution produced by the current policy. To characterize this transformation, we aim to model how each trajectory should be evolved when conditioned on c. Inspired by the generalization capability of neural fields (Gao & Li, 2026), which can extend skill-induced transformations from finitely sampled trajectory pairs to nearby regions of the trajectory space (see empirical validation in Section C.6), we model the distributional change before and after conditioning on the extracted skill as a velocity field.

Specifically, given a group of trajectories $\{\tau_i\}_{i=1}^G \sim p_\theta(\tau)$, the agent generates a skill $c \sim \pi_\theta(\cdot|p_{\text{ext}}, \{\tau_i\}_{i=1}^G)$. Then, we denote the initial distribution $p_0 = p_\theta(\tau)$ and the final distribution $p_1 = p_\theta(\tau|c)$. Since trajectories are discrete, we perform flow matching in the trajectory representation space, where we encode the trajectory τ into $z = E_\psi(\tau)$ with a fixed representation model (parameterized by ψ). The interpolation path is defined as:

$$z_t = (1-t)z_0 + tz_1, \qquad t \sim \mathcal{U}(0,1). \tag{9}$$

The target velocity is $u_t = z_1 - z_0$. Therefore, the skill-conditioned flow matching loss is:

$$\mathcal{L}_{\text{FM}}^{\text{skill}}(\phi) = \mathbb{E}_{c\sim\pi_\theta(\cdot|p_{\text{ext}},\{\tau_i\}_{i=1}^G),\tau_0\sim p_0,\tau_1\sim p_1,t\sim\mathcal{U}(0,1)}\left[\|v_\phi(z_t,c,t) - (z_1 - z_0)\|^2\right]. \tag{10}$$

Remark 5.1. The learned velocity field specifies how each trajectory representation should be transported toward its skill-conditioned counterpart (see Fig. 18). More importantly, by learning a continuous transformation from a finite set of sampled trajectories, the velocity field generalizes the skill-induced behavioral change to the broader trajectory-representation space. As a result, the evolved trajectory representations produced after one transformation step can still be fed into the same velocity field for further evolution, even though they may lie outside the originally sampled trajectories. This allows us to trace how a trajectory would be progressively transformed by the skill and thereby estimate its eventual behavioral outcome.

5.2 EVALUATION OF EXTRACTED SKILLS

Once we obtain the learned skill-induced velocity field, we can derive the trajectory representation distribution evolved from $p(z)$ via an operator $\mathcal{F}_c : \mathcal{P}(\mathcal{Z}) \to \mathcal{P}(\mathcal{Z})$, where $\mathcal{P}(\mathcal{Z})$ is the space of probability distributions over trajectory representation, defined as:

$$\mathcal{F}_c(p)(z') = \int_{\mathcal{Z}} \delta(z' - \Phi_c(z))\, p(z)\, dz, \quad \Phi_c(z_0) = z_0 + \int_0^1 v_\phi(z_t, c, t)\, dt, \tag{11}$$

where $\delta(\cdot)$ denotes the Dirac delta function centered at a trajectory representation.

To represent the eventual behavioral outcome induced by the skill, where further conditioning on the skill no longer changes the trajectory distribution, we consider the stationary trajectory distribution, denoted by $p^*(z)$, which satisfies:

$$p^*(z) = \mathcal{F}_c(p^*(z)). \tag{12}$$

To obtain this stationary distribution, we iteratively apply $\mathcal{F}_c$ via the velocity field to simulate the evolution process. Specifically, given a set of trajectories sampled from the current policy $\{\tau_i\}_{i=1}^G$, we first encode them into the same representation space used in training velocity field and construct the initial distribution p_{init}:

$$z_i^0 = E_\psi(\tau_i), \quad p_{\text{init}} = \frac{1}{G}\sum\nolimits_{i=1}^{G} \delta(z = z_i^0). \tag{13}$$

Starting from each initial representation z_i^0, we evolve it through the learned skill-induced velocity field. Specifically, at the j-th evolution step, the representation of trajectory τ_i is updated by:

$$z_i^j(c) = z_i^{j-1}(c) + \int_0^1 v_\phi(z_{i,t}^{j-1}, c, t)\, dt, \tag{14}$$

where $z_{i,t}^{j-1}$ denotes the intermediate ODE state initialized from $z_i^{j-1}(c)$. We repeat this process until the difference between two consecutive evolution steps becomes sufficiently small:

$$\left\| z_i^j(c) - z_i^{j-1}(c) \right\|_2 \leq \epsilon_0, \tag{15}$$

where ϵ_0 is a predefined convergence threshold. The converged representation is regarded as the final evolved trajectory representation, denoted as $z_i^{\text{evo}}(c)$. Therefore, the evolved trajectory-representation distribution is constructed by the the evolved trajectories, denoted as:

$$q_{\text{evo}}(z) = \frac{1}{G}\sum\nolimits_{i=1}^{G} \delta(z = z_i^{\text{evo}}(c)). \tag{16}$$

We next construct the target distribution to evaluate this evolved distribution. Our key insight is that agentic self-evolution is fundamentally the process of *training the agentic model to correct its trajectory*, where low-reward behaviors should be progressively transformed toward desirable high-reward behaviors. Therefore, we construct the target distribution via reweighting agent's sampled rollouts, where trajectories with higher reward will be assigned higher weights. To be specific, the reweighting parameter is defined as:

$$\omega_i^{\text{tar}} = \frac{\exp(\beta \hat{A}_i)}{\sum_{j=1}^{G} \exp(\beta \hat{A}_j)}, \tag{17}$$

where $\hat{A}_i$ is the group-normalized advantage of each trajectory, defined as:

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R(\tau_i)\}_{i=1}^G)}{\text{std}(\{R(\tau_i)\}_{i=1}^G) + \epsilon_1}. \tag{18}$$

Proposition 5.2. *The reweighting parameter in Eq.* (17) *is the solution to the following KL-regularized problem:*

$$p_{\text{tar}}(z) = \sum_{i=1}^{G} w_i^{\text{tar}} \delta(z = z_i^0), \qquad \omega^{\text{tar}} = \arg\max_{\omega} \left\{ \sum_{i=1}^{G} \omega_i \hat{A}_i - \frac{1}{\beta} D_{\text{KL}}\left(p_{\text{tar}} \| p_{\text{init}}\right) \right\}. \tag{19}$$

Remark 5.3. We present the detailed derivation in Theorem A.1. By balancing the maximization of expected group-normalized advantage and the constraint on distributional deviation from the initial data, this objective yields a soft target distribution that enables a gradual correction process and stable policy improvement (see impact of β in Section C.4).

Finally, we introduce the evolution-aware skill reward, which quantifies the distance from the target distribution to the skill-evolved distribution:

$$R_{\text{skill}}(c) = -\mathcal{D}_{\text{W}}(p_{\text{tar}}(z), q_{\text{evo}}(z)). \tag{20}$$

Here, we use Wasserstein distance $\mathcal{D}_{\text{W}}$, which measures the optimal transport cost required to correct the evolved trajectory distribution toward the target distribution.

Remark 5.4. This reward evaluates a skill from the perspective of policy evolution. Specifically, it measures how well the evolved trajectory distribution induced by the skill approaches the self-improved target distribution: a skill receives a high reward only when its induced behavioral change can be integrated toward a desirable trajectory distribution. Importantly, the reward is evaluated directly on the quality of the evolved distribution $q_{\text{evo}}(z)$ rather than on the performance improvement relative to the current policy. Since $q_{\text{evo}}(z)$ is obtained through long-term evolution under the skill-induced transformation, artificially degrading the current policy cannot directly create a larger performance gap and thereby increase the skill reward. Therefore, the skill instead induces an evolved trajectory distribution that is genuinely closer to the desirable target distribution. With this skill reward, the gradient conflict can be reduced (as shown in Fig. 25).

5.3 Skill-Policy Co-Evolution

Based on the introduced evaluation of skills, we propose a skill-policy co-evolution algorithm that alternates between skill extraction optimization and policy optimization.

In skill extraction optimization, the objective is to maximize the skill extraction reward in Eq. (20):

$$\max_{\theta} J_{\text{skill}}(\theta) = \mathbb{E}_{c\sim\pi_\theta(\cdot|p_{\text{ext}},\{\tau_i\}_{i=1}^{G})}\left[R_{\text{skill}}(c)\right]. \tag{21}$$

In practice, we optimize this skill extraction objective with GRPO, where we sample M skills and define the training objective as:

$$\mathcal{L}_{\text{skill}}(\theta) = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{|c_m|}\sum_{l=1}^{|c_m|}\min\left(\rho_{m,l}(\theta)\hat{A}^{\text{skill}}_{m,l}, \text{clip}\left(\rho_{m,l}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\right)\hat{A}^{\text{skill}}_{m,l}\right), \tag{22}$$

where the advantage and importance weight are defined as:

$$\hat{A}^{\text{skill}}_{m,l} = \frac{R_{\text{skill}}(c_m) - \text{mean}(\{R_{\text{skill}}(c_j)\}_{j=1}^{M})}{\text{std}(\{R_{\text{skill}}(c_j)\}_{j=1}^{M}) + \epsilon}, \quad \rho_{m,l}(\theta) = \frac{\pi_\theta(c_{m,l}|p_{\text{ext}},\{\tau_i\}_{i=1}^{G}, c_{m,<l})}{\pi_{\theta_{\text{old}}}(c_{m,l}|p_{\text{ext}},\{\tau_i\}_{i=1}^{G}, c_{m,<l})}. \tag{23}$$

After the training of skill extraction, we use the extracted skills to improve the policy through distillation. Specifically, we apply the policy distillation objective in Eq. (4) to internalize the behavior induced by the external skill into the base policy. Repeating skill extraction training and policy distillation forms a co-evolution loop, where the skill extractor becomes better at generating useful skills and the policy becomes better at solving tasks. Overall, we name our proposed algorithm SCALE, with its pseudocode detailed in Algorithm 1.

6 Experiment

In this section, we conduct experiments to evaluate the performance of SCALE by answering the following questions:

- How does SCALE perform compare to existing self-evolving baselines and models across various benchmarks?
- How effective is the model in summarizing reusable reasoning skills from sampled trajectories?
- What are the effects of key parameters and components?
- How does the model actively learn from environment interaction?

6.1 Experiment Setup

Environments. We run experiments with 5 domains including 9 benchmarks: **1) Science Reasoning**, including GPQA (Rein et al., 2023) and AIME26 (Mathematical Association of America, 2026). **2) Code Reasoning**, including SWE-Bench (Jimenez et al., 2024) and LiveCodeBench (Jain et al., 2025). **3) Agentic Reasoning**, including ALFWorld (Shridhar et al., 2021) and GAIA (Mialon et al., 2024). **4) GUI Reasoning**, including AndroidWorld (Rawles et al., 2025) and OSWorld (Xie et al., 2024). **5) Real-world Reasoning**, EdgeBench (Zhu et al., 2026a). Detailed description on environments can be found in Section B.1.

Table 1: Comparison of frontier agents on EdgeBench across six real-world environment-learning domains. The best and second-best results are highlighted.

Model	EdgeBench Category Scores (51 tasks)					
	Scientific	Systems	Optimization	Knowledge	Math	Games
Opus 4.8 (Anthropic, 2026)	23.3	53.4	28.9	27.8	22.7	31.1
GPT-5.5 (OpenAI, 2026b)	26.4	47.5	27.5	31.0	22.7	24.8
GPT-5.4 (OpenAI, 2026a)	22.7	40.0	25.8	27.1	5.1	21.0
GLM-5.1 (Z.ai, 2026)	19.9	32.6	21.2	21.3	5.9	20.1
DS-V4 (DeepSeek-AI, 2026)	20.6	28.0	16.5	24.8	4.4	9.6
Qwen3.5-397B	8.4	16.8	5.7	12.1	7.6	6.9
SCALE (397B) @2h	19.6	40.5	20.8	26.4	15.6	23.1
SCALE (397B) @12h	**38.3**$^{+11.9}$	**65.5**$^{+12.1}$	**42.4**$^{+13.5}$	**48.0**$^{+17.0}$	**37.0**$^{+14.3}$	**43.0**$^{+11.9}$

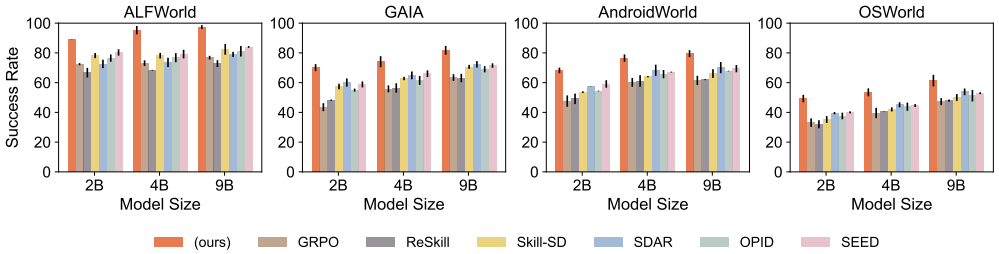


Figure 4: Performance with varying model size. SCALE exhibits higher performance, demonstrating the effectiveness of skill-policy co-evolution compared to existing methods.

Baselines. We evaluate our method against six strong baseline methods: **1)** GRPO (Shao et al., 2024) optimizes LLM policies by computing relative advantages within trajectory groups without training a critic. **2)** ReSkill (He et al., 2026) generates new skills from interaction to adapt the skill repertoire to the evolving policy. **3)** Skill-SD (Wang et al., 2026) summarizes completed trajectories into natural-language skills for internalization. **4)** SDAR (Lu et al., 2026a) a verifier-guided agent optimization method that combines GRPO with gated self-distillation. **5)** OPID (Yang et al., 2026b) extracts episode- and step-level skills from on-policy trajectories and converts their guidance into self-distillation signals. **6)** SEED (Wu et al., 2026) converts on-policy trajectories into hindsight skills and jointly optimizes their skill-conditioned distillation signals.

Reproducibility. All details of our experiments are provided in Section B.3 in terms of the tasks, network architectures, hyperparameters, etc. We conduct experiments on different model size suitable for real-world deployment, which are Qwen3.5 (Qwen Team, 2026) with 5 model sizes (2B, 4B, 9B, 35B-A3B, and 397B-A17B). Model training employs VeRL (Sheng et al., 2025) and the sampling temperature is set to 1, with a maximum generation length of 81920 tokens. All the experiments are run on Ubuntu 22.04.4 LTS with NVIDIA H800 GPUs.

6.2 Experiment Results

Comparative Results To answer the first question, we compare SCALE with all baselines across 9 benchmarks and 5 model scales. Selected results are presented in Fig. 4 and Table 1, with the full comparison reported in Tables 3 and 4 and Figs. 11 to 16. We find that SCALE consistently outperforms the baselines across all 9 benchmarks, often by substantial margins, while sustaining continuous improvement throughout training rather than plateauing after initial gains. Notably, SCALE exhibits consistent evolution gains across models of different scales, which is the first self-evolution method to scale up to 397B to the best of our knowledge, demonstrating that our approach can effectively scale with the underlying model capacity. Skill-SD utilizes a prompting strategy to extract skills for distillation, but the extracted skills are not informative for evolution due to the model's limited skill extraction capability. In contrast, SEED first collects training data to equip the model with the ability to extract reusable skills, and then distills the behavioral effects of these skills into the policy. However, it separates skill training and policy optimization, where the skills become less informative as the policy improves, slowing down the evolution process. Compared with both, SCALE simultaneously evolves skill extraction and policy evolution by evaluating candidate skills according to their effects on future policy improvement, thereby continuously providing evolution-aware learning signals and maintaining skill–policy alignment throughout training.

Skill Extraction Capability To answer the second question, we track the skill extraction metric throughout training in Figs. 22 and 23. The quality of skills extracted by baseline methods rapidly decreases toward zero or negative values as their policies improve, indicating that agents' skill extraction efficiency becomes gradually redundant and misaligned with evolving policy. In contrast, SCALE maintains a consistently positive skill metric across model scales and benchmarks, showing that it continues to extract skills capable of inducing beneficial behavioral changes. In addition, the gradient distributions in Figs. 24 and 25 further show that naive skill extraction training method shifts toward negative cosine similarity, whereas SCALE preserves a substantial positively aligned skill-policy update. These observations demonstrate that evolution-aware skill evaluation effectively alleviates skill-policy misalignment.

Visualization of Velocity Field To better understand how an extracted skill affects subsequent policy evolution, we visualize the learned velocity field in the trajectory-representation space. As illustrated in Fig. 18, the results show that the learned velocity field captures both the direction and magnitude of the skill-induced distributional shift, effectively reflecting how different skills transform agent trajectories with limited training data. This visualization provides an intuitive explanation for why the proposed skill reward can anticipate multi-round evolution without repeatedly sampling complete trajectories from the environment. Therefore, it provides a reliable representation of each skill's behavioral effect and can be subsequently used to evaluate skill quality.

Ablation Studies To answer the third question, we assess the effect of key components by ablating them on all benchmarks under the same setting. 1) *skill extraction training* (w/o skill reward). 2) *evolution-aware skill reward* (w/o velocity). As illustrated in Figs. 20 and 21, removing *skill extraction training* causes the agent's skill extraction capability to gradually degrade during policy optimization. Consequently, the quality of the extracted skills progressively decreases, providing increasingly weak and uninformative guidance for policy optimization and thereby diminishing subsequent policy improvement. Without *evolution-aware skill reward*, the agent is trained using the intuitive reward based on immediate skill-conditioned gains. This objective induces increasingly conflicting gradients between skill extraction and policy optimization: skill extraction favors skills that produce large immediate policy changes, whereas policy internalization reduces the policy's dependence on these skills. This gradient misalignment prevents the two objectives from improving jointly and leads to unstable long-term self-evolution. In contrast, SCALE maintains the quality of extracted skills and aligns skill extraction with policy optimization, enabling sustained performance improvement throughout training.

Environment Learning Capability To answer the fourth question, we evaluate the environment learning capability of the trained agent on EdgeBench. During inference, the agent continuously interacts with the environment, and its performance is measured under different interaction time. This setting examines whether the agent can learn from accumulated interaction trajectories, rather than relying solely on the knowledge acquired during training. As shown in Tables 5 and 6 and Fig. 17, the performance of SCALE consistently improves with longer interaction time, demonstrating that it can summarize newly acquired experience into reusable skills and use them to improve subsequent decisions. In contrast, baseline methods exhibit weaker improvement or plateau earlier. These results show that SCALE retains an effective environment learning capability during inference, enabling continuous adaptation through interaction.

7 Conclusion and Limitation

In this paper, we propose a novel agentic self-evolution method that extracts and internalizes reusable skills from agent's interaction with the environment, aiming to consistently enhance agent's decision-making capability. By modeling skill-induced trajectory distribution velocity field and characterizing the utility of extracted skill in policy evolution, SCALE enables agents to sustain continual self-evolution without prematurely plateauing. Extensive experiments corroborate the effectiveness of SCALE in maintaining skill extraction quality and achieving sustained performance gains across diverse environments. A limitation of SCALE lies in the need to learn a velocity field with strong generalization beyond the sampled trajectories, which is crucial for reliably simulating long-term skill-induced evolution but can be challenging in complex and highly diverse trajectory spaces.

AI USE STATEMENT

We use large language models (LLMs) as an assistive tool for polishing the writing, monitoring the process of experiments, and providing feedback on experiments. All content generated by LLMs was carefully reviewed, verified, and, where necessary, revised by the authors. The authors take full responsibility for the correctness and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

The full algorithmic details of SCALE are presented in the main paper, with additional implementation details and hyperparameter settings included in Section 6. For theoretical results, we include complete proofs in Section A.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The twelfth international conference on learning representations*, 2024.

Anthropic. Claude Opus 4.8 System Card. https://www-cdn.anthropic.com/0b4915911bb0d19eca5b5ee635c80fef830a37ea.pdf, May 2026. System card. Released May 28, 2026; updated June 3, 2026.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2025.

Zizhe Chen, Jiqian Dong, Yizhou Tian, Garry Yang, Yongqiang Chen, Zhitang Chen, and James Cheng. Hista and numca: Estimate state value effectively for large language model reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem2evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20784–20831, 2026.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, April 2026. Technical report.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, et al. Mlevolve: A self-evolving framework for automated machine learning algorithm discovery. *arXiv preprint arXiv:2606.06473*, 2026.

Zhiyuan Fan, Wenwei Jin, Feng Zhang, Bin Li, Yihong Dong, Yao Hu, and Jiawei Li. Evolving-rl: End-to-end optimization of experience-driven self-evolving capability within agents. *arXiv preprint arXiv:2605.10663*, 2026.

Weiguo Gao and Ming Li. How do flow matching models memorize and generalize in sample data subspaces. *npj Artificial Intelligence*, 2026.

Xinyu Geng, Xuanhua He, Sixiang Chen, Yanjing Xiao, Fan Zhang, Shijue Huang, Haitao Mi, Zhenwen Liang, Tianqing Fang, and Yi R. Fung. Deepsearch-world: Self-distillation for deep search agents in a verifiable environment. *arXiv preprint arXiv:2607.07820*, 2026.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Bernie Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic rl. *arXiv preprint arXiv:2606.01619*, 2026.

Xingyuan Hua, Sheng Yue, and Ju Ren. Learning to explore: Scaling agentic reasoning via exploration-aware policy optimization. In *The Twelfth International Conference on Learning Representations*, 2026.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

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

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. MobileWorld: Benchmarking autonomous mobile agents in agent-user interactive, and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*, 2025.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, et al. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6142–6167, 2026.

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

Yang Li, Erik Nijkamp, Semih Yavuz, and Shafiq Rayhan Joty. Learning from language feedback via variational policy distillation. *arXiv preprint arXiv:2605.15113*, 2026b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Chengzhi Liu, Yuzhe Yang, Sophia Xiao Pu, Yepeng Liu, Lin Long, Yichen Guo, Nuo Chen, Zhaotian Weng, Elena Kochkina, Simerjot Kaur, et al. Worldmemarena: Evaluating multimodal agent memory through action-world interaction. *arXiv preprint arXiv:2605.29341*, 2026.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026a.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*, 2026b.

Mathematical Association of America. American Invitational Mathematics Examination (AIME). `https://maa.org/maa-invitational-competitions/`, 2026. Accessed: 2026-05-07.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

OpenAI. GPT-5.4 Thinking System Card. `https://deploymentsafety.openai.com/gpt-5-4-thinking/gpt-5-4-thinking.pdf`, March 2026a. System card. Published March 5, 2026.

OpenAI. GPT-5.5 System Card. `https://deploymentsafety.openai.com/gpt-5-5/gpt-5-5.pdf`, April 2026b. System card. Published April 23, 2026.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. In-context editing: Learning knowledge from self-induced distributions. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. AndroidWorld: A dynamic benchmarking environment for autonomous agents. In *International Conference on Learning Representations*, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

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.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Wei Sun, Wen Yang, Pu Jian, Qianlong Du, Fuwei Cui, Shuo Ren, and Jiajun Zhang. Ktae: A model-free algorithm to key-tokens advantage estimation in mathematical reasoning. *Advances in Neural Information Processing Systems*, 38:161880–161908, 2026.

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025.

Chenyu Wang, Weixin Luo, Sixun Dong, Xiaohua Xuan, Zhengxin Li, Lin Ma, and Shenghua Gao. Mllm-tool: A multimodal large language model for tool agent learning. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6678–6687. IEEE, 2025.

Hao Wang, Guozhi Wang, Han Xiao, Yufeng Zhou, Yue Pan, Jichao Wang, Ke Xu, Yafei Wen, Xiaohu Ruan, Xiaoxin Chen, et al. Skill-SD: Skill-conditioned self-distillation for multi-turn llm agents. *arXiv preprint arXiv:2604.10674*, 2026.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-swe-agent: Can software engineering agents self-evolve on the fly? *arXiv preprint arXiv:2511.13646*, 2025.

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

Bangjun Xiao, Bingquan Xia, Bo Yang, Bofei Gao, Bowen Shen, Chen Zhang, Chenhong He, Chiheng Lou, Fuli Luo, Gang Wang, et al. Mimo-v2-flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Wenda Xu, Rujun Han, Zifeng Wang, Long Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *The Thirteenth International Conference on Learning Representations*, 2024.

Chenxu Yang, Chuanyu Qin, Qingyi Si, Minghui Chen, Naibin Gu, Dingyu Yao, Zheng Lin, Weiping Wang, Jiaqi Wang, and Nan Duan. Self-distilled RLVR. *arXiv preprint arXiv:2604.03128*, 2026a.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, et al. OPID: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026b.

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

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1028–1043, 2024.

Tianzhu Ye, Li Dong, Qingxiu Dong, Xun Wu, Shaohan Huang, and Furu Wei. Online experiential learning for language models. *arXiv preprint arXiv:2603.16856*, 2026.

Wenhao Yu, Zhenwen Liang, Chengsong Huang, Kishan Panaganti, Tianqing Fang, Haitao Mi, and Dong Yu. Guided self-evolving llms with minimal human supervision. *arXiv preprint arXiv:2512.02472*, 2025.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*, 2026.

Z.ai. GLM-5.1: Towards long-horizon tasks. `https://z.ai/blog/glm-5.1`, 2026. Model release blog and model card.

Ding-Chu Zhang, Yida Zhao, Jialong Wu, Liwen Zhang, Baixuan Li, Wenbiao Yin, Yong Jiang, Yu-Feng Li, Kewei Tu, Pengjun Xie, et al. Evolvesearch: An iterative self-evolving search agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13134–13147, 2025a.

Genrui Zhang, Erle Zhu, Jinfeng Zhou, Caiyan Jia, and Hongning Wang. Skillevolver: Skill learning as a meta-skill. *arXiv preprint arXiv:2605.10500*, 2026a.

Qi Zhang, Zhaopeng Feng, Xiaonan Shi, Xiaomeng Hu, Chu Liu, Pengjun Xie, Xiaobin Wang, Jieping Ye, Bryan Hooi, Haobo Wang, et al. Skillcomposer: Learning to evolve agent skills for specification and generalization. *arXiv preprint arXiv:2606.06079*, 2026b.

Yizheng Zhang, Zhenjun Yu, Jiaxin Lai, Cewu Lu, and Lei Han. Agentworld: An interactive simulation platform for scene construction and mobile robotic manipulation. *arXiv preprint arXiv:2508.07770*, 2025b.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026.

Deyao Zhu, Xin Zhou, Shengling Qin, Xuekai Zhu, Hangliang Ding, Shu Zhong, Zixin Wen, Zhonglin Xie, Chenhui Gou, Linxuan Ren, et al. EdgeBench: Unveiling scaling laws of learning from real-world environments. *arXiv preprint arXiv:2607.05155*, 2026a.

Xinyu Zhu, Yuzhu Cai, Zexi Liu, Bingyang Zheng, Cheng Wang, Rui Ye, Jiaao Chen, Hanrui Wang, Wei-Chen Wang, Yuzhi Zhang, et al. Toward ultra-long-horizon agentic science: Cognitive accumulation for machine learning engineering. *arXiv preprint arXiv:2601.10402*, 2026b.

A CONSTRUCTION OF TARGET DISTRIBUTION

Proposition A.1 (KL-Regularized Target Distribution)**.** *Let $p_{\text{init}}(\tau_i) = 1/G$ denote the initial distribution over the sampled trajectory group, and $p_\omega(\tau_i) = \omega_i$ denote the reweighted distribution. Then ω is the unique solution to:*

$$\omega^{\text{tar}} = \arg\max_{\omega} \left\{ \sum_{i=1}^{G} \omega_i \hat{A}_i - \frac{1}{\beta} D_{\text{KL}} \left(p_\omega \| p_{\text{init}} \right) \right\}. \tag{24}$$

Proof. The Lagrangian is

$$\mathcal{L}(\omega, \lambda) = \sum_{i=1}^{G} \omega_i \hat{A}_i - \frac{1}{\beta} \sum_{i=1}^{G} \omega_i \log \frac{\omega_i}{1/G} + \lambda \left(\sum_{i=1}^{G} \omega_i - 1 \right). \tag{25}$$

Taking derivative with respect to ω_i gives

$$\hat{A}_i - \frac{1}{\beta} \left(\log \omega_i + 1 + \log G \right) + \lambda = 0. \tag{26}$$

Therefore, we have

$$\omega_i^{\text{tar}} \propto \exp(\beta \hat{A}_i). \tag{27}$$

After normalization, the optimal solution is

$$\omega_i^{\text{tar}} = \frac{\exp(\beta \hat{A}_i)}{\sum_{j=1}^{G} \exp(\beta \hat{A}_j)}. \tag{28}$$

Hence, we complete the proof. □

B EXPERIMENT SETUP

B.1 ENVIRONMENTS

We evaluate our method on five areas with 9 benchmarks, which are widely used in prior studies. We elaborate on what follows.

- **GPQA** Rein et al. (2023), a benchmark of graduate-level multiple-choice science questions that assesses deep domain knowledge and reasoning under uncertainty.
- **AIME26** Mathematical Association of America (2026), a benchmark of challenging mathematical problems from the 2026 American Invitational Mathematics Examination (AIME), designed to evaluate advanced mathematical reasoning and problem-solving ability.
- **SWE-Bench** Jimenez et al. (2024), a benchmark of real-world GitHub issues from open-source repositories, where agents are required to understand codebases and generate patches that resolve the specified issues.
- **LiveCodeBench** Jain et al. (2025), a contamination-free benchmark of recently released competitive programming problems that evaluates code generation, execution, and reasoning across diverse programming tasks.
- **ALFWorld** Shridhar et al. (2021), a text-based embodied environment featuring household tasks, where agents navigate and interact with objects via natural language commands.
- **GAIA** Mialon et al. (2024), a benchmark of real-world question answering tasks that integrate knowledge retrieval, reasoning, and multi-step tool use.
- **AndroidWorld** Rawles et al. (2025), an environment with 116 dynamic tasks across 20 real-world Android apps, designed to evaluate mobile agents' capabilities in app navigation and system-level control.
- **OSWorld** Xie et al. (2024), a large-scale web benchmark with 369 tasks that span real-world web and desktop applications, requiring long-horizon planning and multi-window tool coordination.
- **EdgeBench** Zhu et al. (2026a), a multi-domain benchmark designed to evaluate agents' environment-learning and adaptive decision-making capabilities across diverse real-world edge scenarios.

B.2 BASELINES

We test our method against six baselines. We implement them based on their publicly available implementations.

- Group Relative Policy Optimization (GRPO) Shao et al. (2024), a group-based, critic-free reinforcement learning method that estimates advantages over trajectory groups, providing stable optimization for reasoning-oriented LLM training.
- Reconciling Skill Creation (ReSkill) He et al. (2026), a self-evolving framework that continually generates new skills from interaction trajectories and updates the skill repertoire to adapt to the evolving policy.
- Skill Self-Distillation (Skill-SD) Wang et al. (2026), a method that adapts self-distillation to multi-turn agent tasks, where completed trajectories are summarized into compact natural-language skills and the student internalizes the teacher-side guidance rather than relying on the skill at test time.
- Self-Distillation Agent Reinforcement (SDAR) Lu et al. (2026a), a method that keeps verifier-driven GRPO as the primary optimization backbone and adds a separately gated self-distillation objective for multi-turn agents.
- Offline Policy Improvement with Distillation (OPID) Yang et al. (2026b), a framework that extracts episode- and step-level skill supervision directly from completed on-policy trajectories.
- Self-Evolving On-Policy Distillation (SEED) Wu et al. (2026), a method that extracts hindsight skills from completed on-policy trajectories and converts their effects on action probabilities into dense token-level supervision jointly optimized with outcome-based reinforcement learning.

B.3 Implementation Details

We provide detailed hyperparameter settings in Table 2.

Table 2: Hyperparameter settings of SCALE. Unless otherwise stated, these settings are shared across benchmarks.

Component	Hyperparameter	Value
Policy training and skill extraction		
	Total self-evolution iterations	150
	Skill and policy updates per evolution iteration	50
	Trajectory group size G	16
	Number of candidate skills M	8
	Sampling temperature	1.0
	Maximum generation length	81,920 tokens
	Policy optimizer	AdamW
	Policy learning rate	1×10^{-6}
	GRPO clipping coefficient ϵ_{clip}	0.2
	KL coefficient λ	0.01
	Gradient clipping norm	1.0
Trajectory representations		
	Trajectory encoder E_ψ	T5
	Trajectory representation dimension d_z	1024
	Representation normalization	ℓ_2 normalization
Skill-induced velocity field v_ϕ		
	Backbone	FLUX-style MMDiT
	Number of parameters	1.3B
	Optimizer	AdamW
	Learning rate	2×10^{-5}
	AdamW β_1, β_2	$(0.9, 0.95)$
	Weight decay	0.05
	Gradient clipping norm	1.0
	Flow-matching batch size	16 trajectory pairs
	Velocity updates per evolution iteration	50
Evolution-aware skill evaluation		
	Reweighting parameter β	5.0
	Distribution distance	Sinkhorn Wasserstein distance
	ODE solver	Fourth-order Runge–Kutta
	ODE integration steps	20
	Convergence threshold ϵ_0	1×10^{-2}

We implement our code using Pytorch 2.8.0, built upon the open-source framework of verl Sheng et al. (2024), provided at `https://github.com/volcengine/verl`. All the experiments are run on Ubuntu 22.04.4 LTS with NVIDIA H800 GPUs.

B.4 PSEUDOCODE

We present the Pseudocode of SCALE in Algorithm 1.

Algorithm 1: Skill-Policy Co-Evolution

Initialize policy π_θ, trajectory encoder E_ψ, and velocity field v_ϕ.
for each iteration **do**
  # Phase 1: Velocity Field Training
  for each training iteration **do**
    Rollout trajectories $\{\tau_i\}_{i=1}^{G} \sim \pi_\theta$.
    Sample skill c via Eq. (3) and construct the skill-conditioned policy.
    Sample trajectory pair (τ_0, τ_1) and optimize the velocity field v_ϕ via Eq. (10).
  end for
  # Phase 2: Skill Extraction Training
  for each training iteration **do**
    Rollout trajectories $\{\tau_i\}_{i=1}^{G} \sim \pi_\theta$ and compute rewards $\{R_i\}_{i=1}^{G}$.
    Sample M skills $\{c\}_{i=1}^{M}$ via Eq. (3) and compute the skill reward R_{skill} via Eq. (20).
    Update the skill extraction process π_θ by minimizing $\mathcal{L}_{\text{skill}}(\theta)$ in Eq. (22).
  end for
  # Phase 3: Policy Optimization
  for each training iteration **do**
    Rollout trajectories $\{\tau_i\}_{i=1}^{G} \sim \pi_\theta$.
    Generate skill via Eq. (3) and construct teacher policies $\pi_\theta(\cdot|s, c)$.
    Update the policy by distilling the skill-conditioned teacher behavior using Eq. (4).
  end for
end for

B.5 INSTRUCTION TEMPLATE

The instruction template for skill extraction consists of two components: *system prompt* in Fig. 5 and *basic prompt* in Fig. 6. The system prompt contains all fixed instructions for skill extraction, specifying how the agent should analyze trajectories, identify key success or failure patterns, filter noisy feedback, abstract reusable decision rules, and format the final skill. The basic prompt only provides task-specific input fields, including the task goal, visited states, executed actions, observations, and environmental feedback. It contains no additional instructions, so that the skill extraction behavior is consistently governed by the fixed system prompt across different tasks.

System Prompt for Skill Extraction

You are a skill extraction module. Given the task goal, the agent's trajectory, the executed actions, the observations, and the environment feedback, your goal is to extract a reusable skill that can help the agent avoid similar mistakes and improve future decision making.
The extracted skill should not simply repeat the final answer or copy the environment feedback. Instead, it should summarize the underlying principle, decision rule, or correction strategy that explains why the previous behavior succeeded or failed. The skill should be general enough to transfer to similar tasks, but specific enough to guide the agent's next action.
When extracting the skill, follow these requirements:

- Identify the key mistake, bottleneck, or useful pattern in the trajectory.
- Ignore noisy, redundant, or task-irrelevant information in the feedback.
- Explain what the agent should pay attention to in similar future states.
- Provide an actionable correction strategy rather than a task-specific answer.
- Keep the skill concise, reusable, and suitable to be used as additional context.

The reasoning process and the extracted skill must be enclosed within `<think> </think>` and `<skill> </skill>` tags respectively.

Output Format:

```
<think>
Analyze the trajectory and feedback. Identify the core reason for
success or failure, remove irrelevant information, and derive a
reusable decision rule.
</think>

<skill>
A concise reusable skill that can guide future decisions.
</skill>
```

Figure 5: System prompt template for extracting reusable skills from agent trajectories.

Basic Prompt for Skill Extraction

You are a skill extraction agent. Your role is to analyze the agent's interaction experience and extract a reusable skill that can help improve future decision making.

Task Goal:

```
{TASK_GOAL}
```

Interaction Trajectory:

```
{TRAJECTORY}
```

Skill Example:

```
{Skill.md}
```

Figure 6: Basic prompt template defining the role and input fields of the skill extraction agent.

The instruction templates consist of three components: *system prompt* in Fig. 7, *basic prompt* in Fig. 8, and *action guidance* in Fig. 9. The system prompt specifies the high-level cognitive process that the agent should follow, including how it reasons about the environment, conducts exploration, and maintains intermediate memory. This prompt will be input in system prompt of instruction-tuned models and remains fixed across different tasks. The basic prompt defines the agent's role and capabilities, describing what it means to act as an agent in the given environment. It also includes the set of valid operations for agents. The action guidance contains the task goal, detailed semantics of each action, and illustrative usage examples. We take the prompts of AndroidWorld as an example.

System Prompt for Instruction-Tuned Model

Now output an action from the action list in the correct JSON format, following the reason why you do that. During thinking, you can explore the current page by executing possible actions and memorize the pages you have explored.
The reasoning process and the action are enclosed within `<think> </think>` and `<action> </action>` tags respectively.

Example:

```
<think> This is the reasoning process. </think>
<action> The final action is \boxed{action here} </action>
```

In the last part of the action, the final exact action must be enclosed within `\boxed{}` with the action format.

Figure 7: System prompt template specifying the reasoning and action generation protocol.

Basic Prompt for Instruction-Tuned Model

You are an agent who can operate an Android phone on behalf of a user. Based on the user's goal or request, you may:

- Answer back if the request is a question or a chat message (e.g., "What is my schedule for today?").
- Complete tasks described in the request by performing actions step by step on the phone.

When given a user request, you will attempt to complete it step by step. At each step, you are provided with the current screenshot (including the original screenshot and the annotated version with bounding boxes and numeric indexes) and a textual history of previously executed actions. Based on this information and the goal, you must output exactly one action in the correct JSON format.

Termination Conditions:

- If the task is completed, output: `{{action_type: status, goal_status: complete}}`
- If the task is infeasible, output: `{{action_type: status, goal_status: infeasible}}`
- If the request is a question, output: `{{action_type: answer, text: <answer_text>}}`

Action Space: `{{action_type: click, index: <target_index>}}`

```
{{action_type: long_press, index: <target_index>}}
{{action_type: input_text, text: <text_input>, index: <target_index>}}
{{action_type: keyboard_enter}}
{{action_type: navigate_home}}
{{action_type: navigate_back}}
{{action_type: scroll, direction: <up, down, left, right>, index:
<optional_target_index>}}
{{action_type: open_app, app_name: <name>}}
{{action_type: wait}}
```

Figure 8: Basic prompt defining the agent role, task completion protocol, and available action space.

Action Guidance for Agentic Decision Making

This section provides detailed guidance on task completion and action execution. The action guidance specifies the task objective, the semantics of each available action, and concrete usage examples to ensure consistent and executable outputs from the agentic model.

Task Objective:
Given a user request and the current environment observation, the agent must select exactly one action at each step to make progress toward completing the task. The agent should reason about the goal, current UI state, and historical actions before choosing the next action.

General Action Constraints:
- At each step, output exactly one action in valid JSON format.
- The selected action must be executable under the current UI state.
- Do not hallucinate UI elements or indexes that are not present.
- If the task has been completed, terminate using a `status` action.

Action Definitions and Examples:
- `click`: Tap a UI element specified by its index.

  `{{action_type: click, index: 5}}`
- `long_press`: Perform a long press on a UI element.

  `{{action_type: long_press, index: 3}}`
- `input_text`: Input text into a text field.

  `{{action_type: input_text, text: meeting notes, index: 2}}`
- `keyboard_enter`: Press the enter key on the virtual keyboard.

  `{{action_type: keyboard_enter}}`
- `scroll`: Scroll the screen in a specified direction.

  `{{action_type: scroll, direction: down}}`
- `navigate_back`: Navigate to the previous screen.

  `{{action_type: navigate_back}}`
- `navigate_home`: Return to the home screen.

  `{{action_type: navigate_home}}`
- `open_app`: Open an application by name.

  `{{action_type: open_app, app_name: Calendar}}`
- `wait`: Take no action and wait for the UI to update.

  `{{action_type: wait}}`
- `status`: Indicate task completion or infeasibility.

  `{{action_type: status, goal_status: complete}}`

Prompt Composition:
The system prompt is provided to the model as a system-level instruction. The basic prompt and the action guidance are concatenated and supplied as the user prompt, together with task-specific observations, to guide step-by-step decision making.

Figure 9: Action guidance specifying task objectives, action semantics, and usage examples for agentic models.

B.6 Example of Extracted Skill

In this section, we provide an example of the extracted skill. The extracted skill describes a reusable code-review behavior distilled from successful review trajectories. Rather than memorizing a task-specific solution, it abstracts the review process into a structured policy that specifies its intended usage, step-by-step execution procedure, exclusion criteria, and output format. This allows the agent to consistently perform defect-oriented reviews on new code changes while avoiding common failure modes such as speculative comments, style-only suggestions, or incomplete diff inspection.

Example of an Extracted Skill: Code Review

Extracted Skill:

```
<skill>
Name: Review Agent

Purpose: Perform a read-only, defect-first review of a specified code
change and report every actionable issue that the author would likely
fix.

When to use: Review uncommitted changes, a base-branch diff, a commit,
or a target specified by custom review instructions.

Procedure:
1. Read all applicable AGENTS.md instructions.
2. Determine the actual change to be merged. For a base-branch review,
resolve the upstream comparison reference when available, compute
`git merge-base HEAD <comparison-ref>`, and inspect the diff from
that merge base.
3. Inspect every changed hunk together with sufficient surrounding
code, call sites, and relevant tests.
4. Continue reviewing the complete diff after finding an issue.
5. Report an issue only if it is newly introduced, demonstrable from a
concrete execution path, materially affects correctness, security,
performance, or maintainability, and is actionable.

Do not report speculative concerns, style nits, intentional changes,
or pre-existing issues. Do not modify files, create commits, push
branches, post comments, or delegate the review.

Output: Present findings first in severity order as
[P0--P3] Imperative finding title --- path/to/file:line.
For each finding, cite the smallest changed line range and explain the
affected scenario. If no qualifying issue exists, output "No findings."
Then summarize the overall assessment and any material test gaps or
residual risks.
</skill>
```

Figure 10: An example of a reusable skill for code reviews.

C Additional Results

C.1 Comparison Results

We evaluate SCALE's performance with varying model's parameter. The comparative results are shown in Tables 3 and 4 and Figs. 11 to 16

Summary of key findings. The results show that SCALE significantly improves self-evolution performance, delivering performance gains over the base model that are 30%-50% more than the performance gains achieved by baseline methods across diverse benchmarks and model scales, particularly on long-horizon reasoning tasks. This highlights its ability to extract informative skills that continuously support policy improvement in complex interactive environments. SCALE also exhibits favorable scaling behavior, maintaining consistent advantages as model capacity increases while enabling smaller models to remain competitive with larger base models. Moreover, its performance continues to improve throughout training rather than plateauing prematurely, demonstrating more effective and sustained self-evolution.

Table 3: Performance comparison on representative reasoning and long-horizon benchmarks. The best and second-best results are highlighted. Improvement denotes the relative increase in the evolution gain achieved by SCALE over the strongest baseline.

Model	Method	Science		Code		Agent		GUI	
		GPQA	AIME26	SweBench	LiveCode	ALFWorld	GAIA	Android	OSWorld
Qwen3.5-2B	Base	51.6	34.8	16.9	28.6	31.8	22.7	27.9	14.5
	GRPO	54.1	37.0	18.7	30.8	72.4	43.5	47.6	33.1
	ReSkill	53.8	36.1	19.3	30.0	66.7	48.2	49.1	32.0
	Skill-SD	54.3	38.0	19.0	30.7	78.2	57.4	53.6	35.2
	SDAR	54.7	36.8	21.1	32.3	72.6	60.1	57.2	39.5
	OPID	56.2	39.1	20.5	32.8	76.5	55.0	54.4	37.6
	SEED	55.6	38.5	20.8	32.1	80.3	58.8	59.1	40.0
	SCALE	**63.7**$^{+7.5}$	**46.1**$^{+7.0}$	**28.0**$^{+6.9}$	**40.2**$^{+7.4}$	**89.0**$^{+8.7}$	**70.1**$^{+10.}$	**68.2**$^{+9.1}$	**49.3**$^{+9.3}$
	Improve	163.0%	162.8%	164.3%	176.2%	17.9%	26.7%	29.2%	36.5%
Qwen3.5-4B	Base	76.2	55.4	34.6	55.8	58.3	43.7	58.6	35.6
	GRPO	78.6	57.7	36.2	58.9	73.1	55.8	60.2	39.6
	ReSkill	78.1	56.9	36.8	57.8	68.3	56.4	61.1	40.4
	Skill-SD	78.4	59.1	37.0	58.7	78.1	62.8	64.0	42.0
	SDAR	79.2	58.0	39.2	60.0	73.6	64.9	68.3	45.1
	OPID	80.5	60.0	38.5	61.1	76.8	61.5	65.7	43.8
	SEED	79.8	59.4	39.0	60.7	79.2	66.0	67.1	44.6
	SCALE	**86.1**$^{+5.6}$	**67.2**$^{+7.2}$	**46.0**$^{+6.8}$	**68.3**$^{+7.2}$	**95.3**$^{+16.}$	**74.1**$^{+8.1}$	**76.5**$^{+8.2}$	**53.5**$^{+8.4}$
	Improve	130.2%	156.5%	147.8%	135.8%	77.0%	36.3%	84.5%	88.4%
Qwen3.5-9B	Base	81.7	69.6	49.7	65.6	66.2	55.1	57.8	41.8
	GRPO	83.8	71.6	51.0	68.3	76.8	63.4	61.5	47.2
	ReSkill	83.4	71.2	51.7	67.5	72.8	63.1	62.0	47.9
	Skill-SD	83.9	73.3	52.3	68.6	82.4	70.6	66.1	50.0
	SDAR	84.2	72.0	54.5	70.4	79.0	72.3	70.1	53.8
	OPID	86.0	74.1	53.7	70.8	81.1	69.0	67.8	51.2
	SEED	85.2	73.5	54.0	70.0	84.0	71.5	69.4	52.9
	SCALE	**91.2**$^{+5.2}$	**81.0**$^{+6.9}$	**62.1**$^{+7.6}$	**78.6**$^{+7.8}$	**97.4**$^{+13.}$	**81.7**$^{+9.4}$	**79.5**$^{+9.4}$	**61.4**$^{+7.6}$
	Improve	120.9%	153.3%	158.3%	150.0%	75.3%	54.7%	76.4%	63.3%
Qwen3.5-35B	Base	84.2	91.0	54.6	74.6	72.4	63.2	71.1	54.5
	GRPO	86.3	92.7	56.1	77.1	83.6	71.4	74.9	65.0
	ReSkill	85.9	92.2	57.0	76.2	79.7	72.0	76.3	65.7
	Skill-SD	86.1	93.8	57.2	77.0	89.0	79.4	76.0	68.2
	SDAR	86.6	92.8	60.0	79.0	86.5	81.6	80.2	72.1
	OPID	88.5	94.5	59.1	79.6	88.1	78.1	78.0	69.4
	SEED	87.7	93.9	59.7	78.8	90.2	80.5	79.4	71.0
	SCALE	**94.0**$^{+5.5}$	**97.2**$^{+2.7}$	**78.6**$^{+18.}$	**87.4**$^{+7.8}$	**98.1**$^{+7.9}$	**90.3**$^{+8.7}$	**88.6**$^{+8.4}$	**80.2**$^{+8.1}$
	Improve	127.9%	77.1%	344.4%	156.0%	44.4%	47.3%	92.3%	46.0%
Qwen3.5-397B	Base	88.4	91.3	76.4	83.6	79.8	72.5	66.8	62.2
	GRPO	90.0	92.8	77.7	85.8	91.4	80.9	71.5	66.3
	ReSkill	89.8	92.4	78.4	85.0	88.5	81.3	71.9	67.2
	Skill-SD	90.1	94.1	78.7	85.7	96.0	87.3	75.4	69.5
	SDAR	90.4	93.0	81.2	87.6	93.4	89.2	80.1	73.8
	OPID	92.0	94.8	80.5	88.0	95.1	86.4	77.6	71.0
	SEED	91.3	94.0	81.0	87.2	96.7	88.5	79.0	72.6
	SCALE	**96.1**$^{+4.1}$	**98.0**$^{+3.2}$	**88.4**$^{+7.2}$	**93.6**$^{+5.6}$	**99.0**$^{+2.3}$	**95.2**$^{+6.0}$	**89.3**$^{+9.2}$	**82.1**$^{+8.3}$
	Improve	113.9%	91.4%	150.0%	127.3%	13.6%	35.9%	69.2%	71.6%

Table 4: Comparison on EdgeBench across six real-world environment-learning domains.

	Method	EdgeBench Domains					
		Scientific	Systems	Optimization	Knowledge	Math	Games
Pretrained	Opus 4.8	23.3	53.4	28.9	27.8	22.7	31.1
	GPT-5.5	26.4	47.5	27.5	31.0	22.7	24.8
	GPT-5.4	22.7	40.0	25.8	27.1	5.1	21.0
	GLM-5.1	19.9	32.6	21.2	21.3	5.9	20.1
	DS-V4	20.6	28.0	16.5	24.8	4.4	9.6
Baseline (397B)	Base	8.4	16.8	5.7	12.1	7.6	6.9
	GRPO	11.6	20.3	5.8	14.7	9.4	7.2
	ReSkill	13.9	18.8	7.1	13.9	8.0	6.8
	Skill-SD	15.7	19.6	13.4	17.3	9.8	15.1
	SDAR	17.8	32.2	18.7	21.5	14.9	18.8
	OPID	19.4	34.1	17.5	23.2	13.7	20.5
	SEED	18.7	33.5	18.1	22.7	14.2	19.9
	SCALE	19.6	40.5	20.8	26.4	15.6	23.1

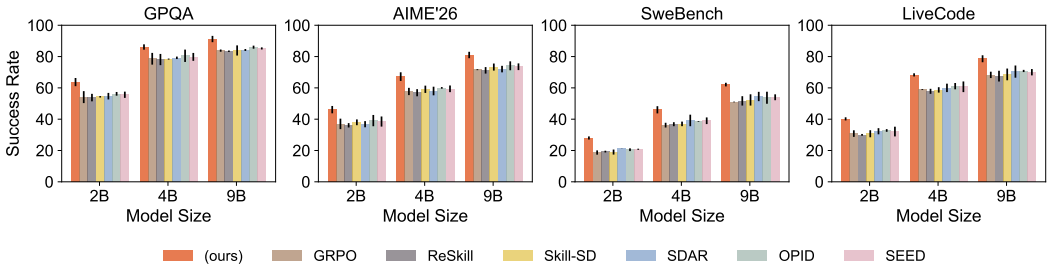


Figure 11: Performance with varying model size. SCALE exhibits higher performance, demonstrating the effectiveness of skill-policy co-evolution compared to existing methods.

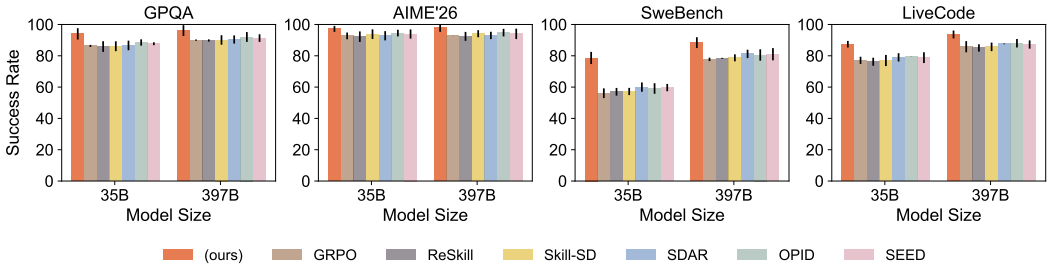


Figure 12: Performance with varying model size. SCALE exhibits higher performance, demonstrating the effectiveness of skill-policy co-evolution compared to existing methods.

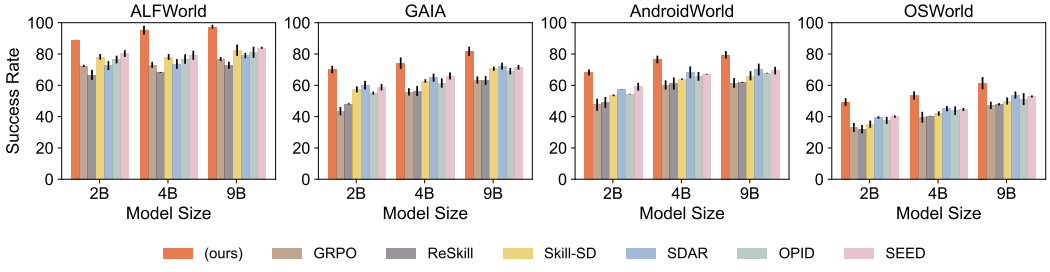


Figure 13: Performance with varying model size. SCALE exhibits higher performance, demonstrating the effectiveness of skill-policy co-evolution compared to existing methods.

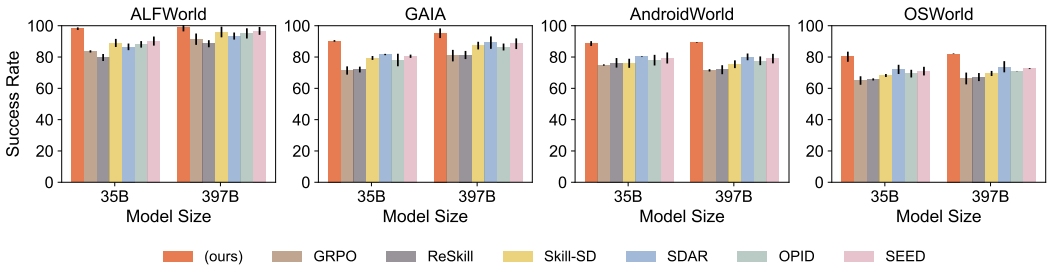


Figure 14: Performance with varying model size. SCALE exhibits higher performance, demonstrating the effectiveness of skill-policy co-evolution compared to existing methods.

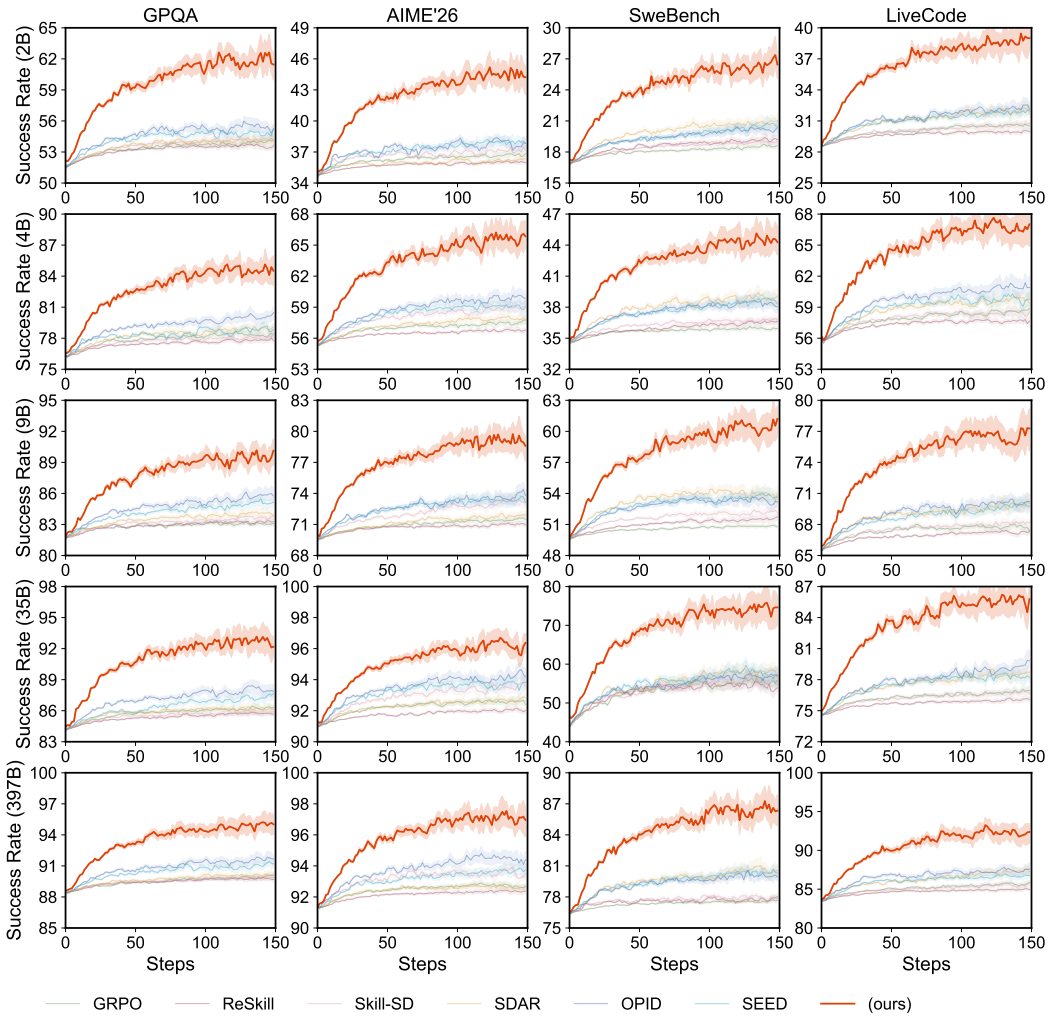


Figure 15: Training convergence with varying model size. Uncertainty intervals depict standard deviation over three seeds. SCALE consistently and significantly surpasses existing methods in terms of evolution efficiency.

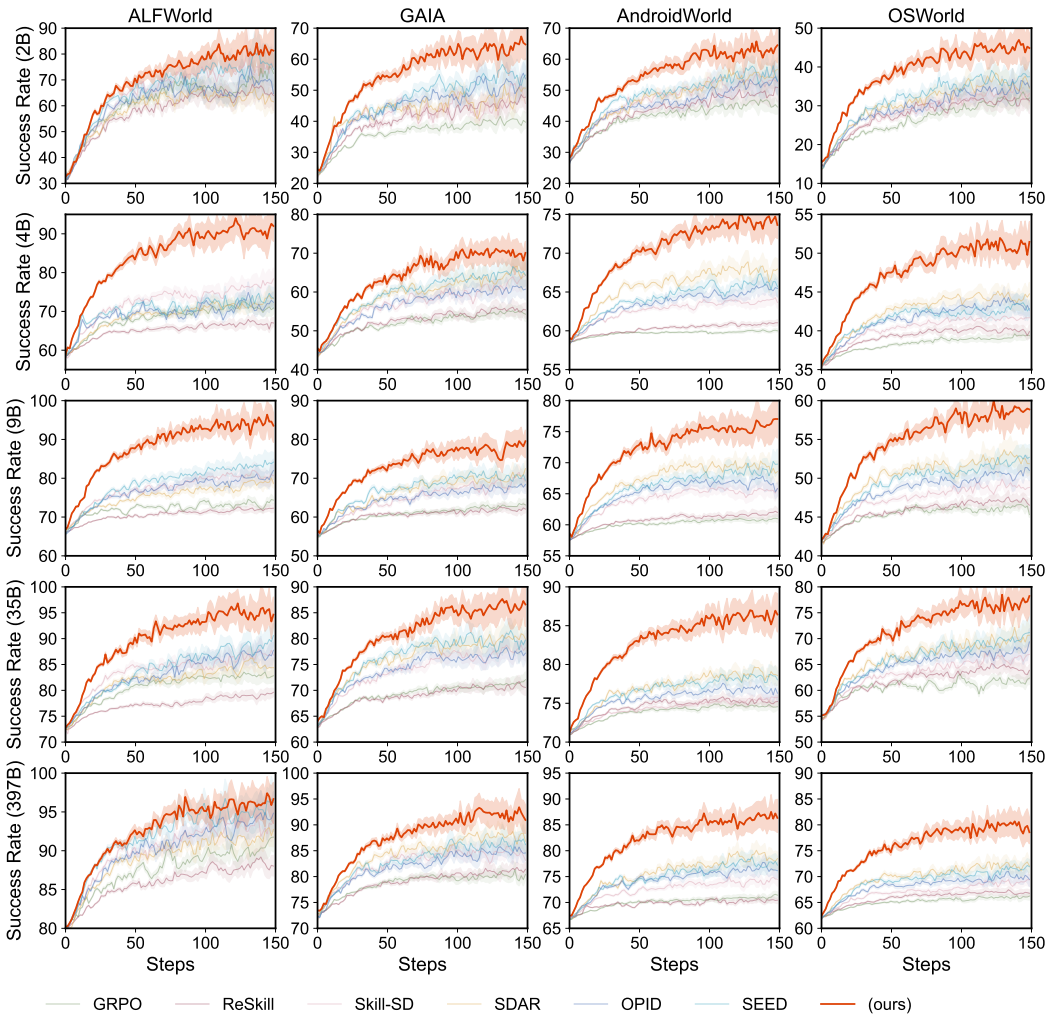


Figure 16: Training convergence with varying model size. Uncertainty intervals depict standard deviation over three seeds. SCALE consistently and significantly surpasses existing methods in terms of evolution efficiency.

C.2 Environment Learning Capability

We provide a detailed evaluation of environment learning capability on EdgeBench. After training, each agent continuously interacts with the environment during inference, and its performance is recorded under different interaction time. Longer runtimes allow the agent to accumulate more trajectories and feedback, enabling us to measure whether it can learn from its interaction history and progressively improve subsequent behavior.

As shown in Tables 5 and 6 and Fig. 17, SCALE starts with a moderate performance at 2h, comparable to GPT-5.4, but improves substantially as the interaction time increases. The improvement does not merely result from executing more actions; it indicates that the agent can summarize accumulated experience, extract reusable skills, and apply them to later decisions. In comparison, baseline models improve more slowly or plateau earlier, suggesting that they make less effective use of additional interaction experience. These results demonstrate that the skill extraction capability learned during training remains effective at inference time. Consequently, SCALE can continuously acquire knowledge from the environment and refine its behavior through interaction, exhibiting stronger environment learning capability across different interaction time.

Table 5: Model performance across different interaction time.

Model	@2h	@4h	@6h	@8h	@10h	@12h
Claude Opus 4.8	33.2	38.5	40.8	42.1	43.3	44.2
GPT-5.5	31.2	36.0	38.2	40.3	42.1	43.1
GPT-5.4	25.0	28.2	30.3	32.1	33.3	34.2
GLM-5.1	21.4	24.2	26.8	28.2	29.1	30.4
DS-V4-Pro	17.1	21.1	22.9	23.8	25.1	25.7
SCALE	25.3	28.7	33.2	38.4	43.0	47.2

Table 6: Performance of SCALE across increasing interaction time on EdgeBench.

Domain	@2h	@4h	@6h	@8h	@10h	@12h
Scientific	19.6	22.5	26.2	31.2	34.5	38.3
Systems	40.5	45.2	48.2	52.7	58.5	65.5
Optimization	20.8	23.9	29.2	34.4	39.0	42.4
Knowledge	26.4	29.6	34.2	40.3	45.0	48.0
Formal	15.6	18.4	23.4	29.0	33.5	37.0
Games	23.1	25.9	31.0	36.5	40.0	43.0

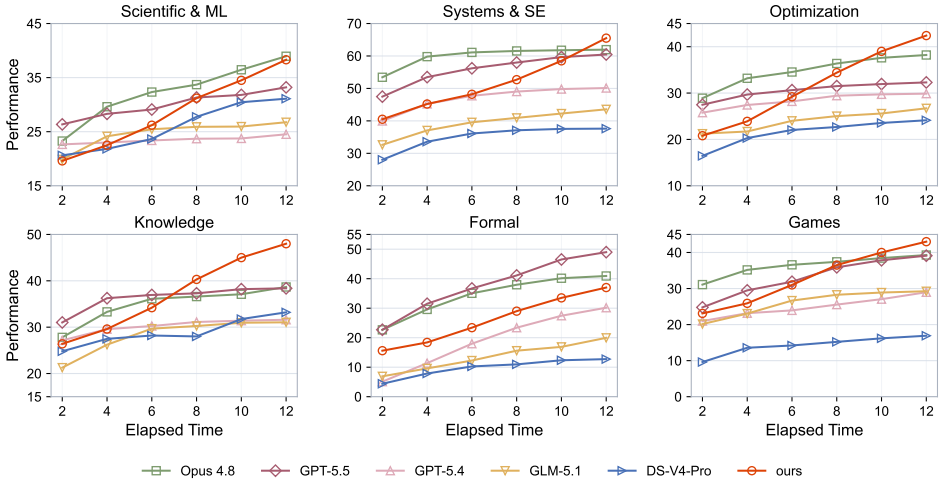


Figure 17: Performance of SCALE across increasing interaction time on EdgeBench.

C.3 Visualization of Velocity Field

We visualize the learned velocity field in the trajectory-representation space to examine whether it faithfully captures the behavioral effect of an extracted skill. Starting from the same initial trajectory distribution, we repeatedly apply a fixed skill to obtain the ground-truth sequence of policies and their corresponding rollout distributions. In addition, we recursively evolve the initial distribution using the learned skill-induced velocity field. We utilize PCA (Abdi & Williams, 2010) to project the high-dimensional trajectory representations into a two-dimensional space for visualization.

As illustrated in Fig. 18, we first observe that the learned velocity field contains attracting fixed points in the trajectory-representation space. Around these regions, the integrated velocity becomes negligible, so recursively evolved representations are gradually drawn into the corresponding basins of attraction. This indicates that the velocity field does not induce unbounded distributional drift, but instead drives trajectory representations toward stable, absorbing-like states.

Second, the predicted final distribution closely follows the direction and magnitude of the ground-truth distributional shift. This demonstrates that the velocity field captures the cumulative behavioral effect of repeatedly applying a skill and provides an effective representation of long-term policy evolution without repeatedly collecting full rollouts.

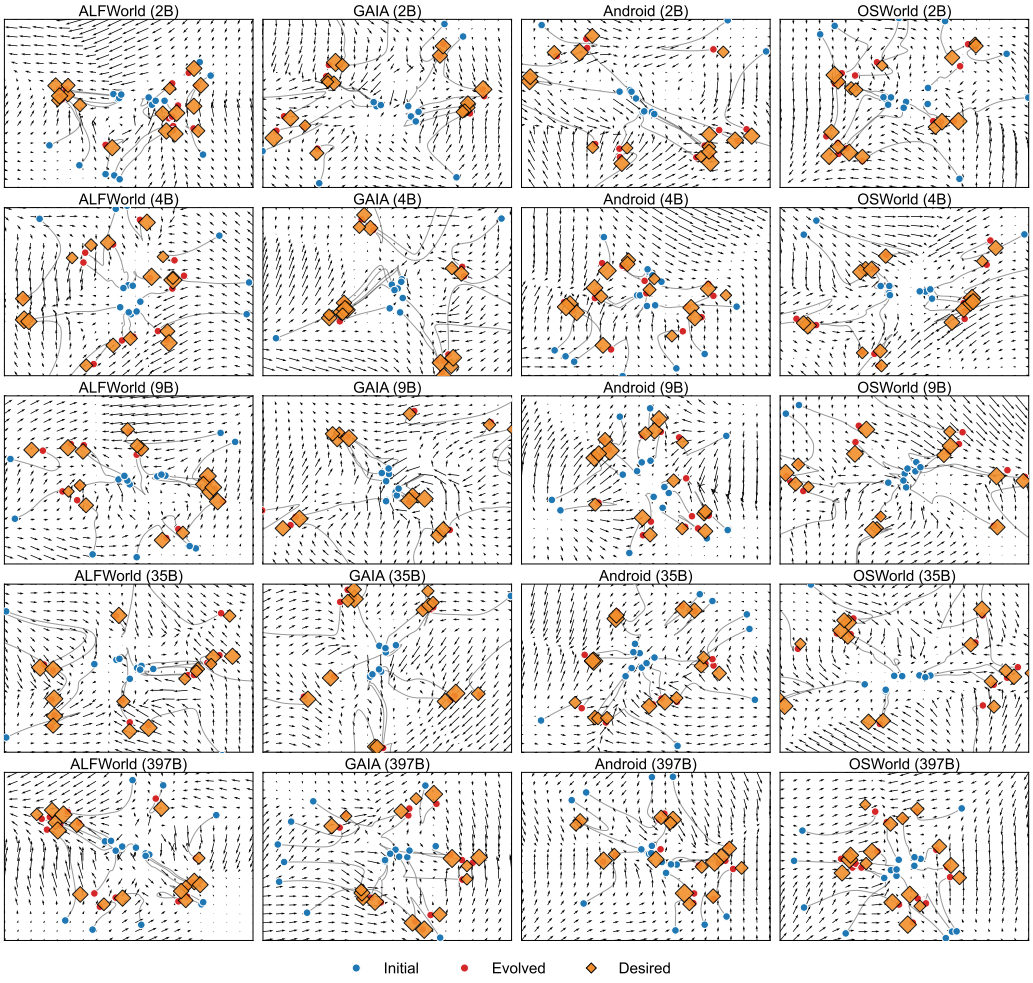


Figure 18: Long-term evolution fidelity of the learned skill-induced velocity field. We compare trajectory distributions obtained by repeatedly internalizing the same skill into the policy with distributions predicted through multi-step velocity-field rollout. The close agreement demonstrates that the learned field faithfully captures the accumulated long-term effect of skills on policy evolution.

C.4 Impact of Reweighting Parameter β

We investigate the impact of the reweighting parameter β, which controls the extent to which high-reward trajectories are emphasized when constructing the target distribution. As shown in Fig. 19, performance is maximized at a moderate value of β. A small β produces insufficient separation between high- and low-quality trajectories, resulting in a weak evolution signal. In contrast, an excessively large β over-concentrates the target distribution on a small set of trajectories, reducing diversity and leading to less stable optimization. These results validate the use of a moderate reweighting strength to balance exploitation of promising behaviors and coverage of diverse trajectories.

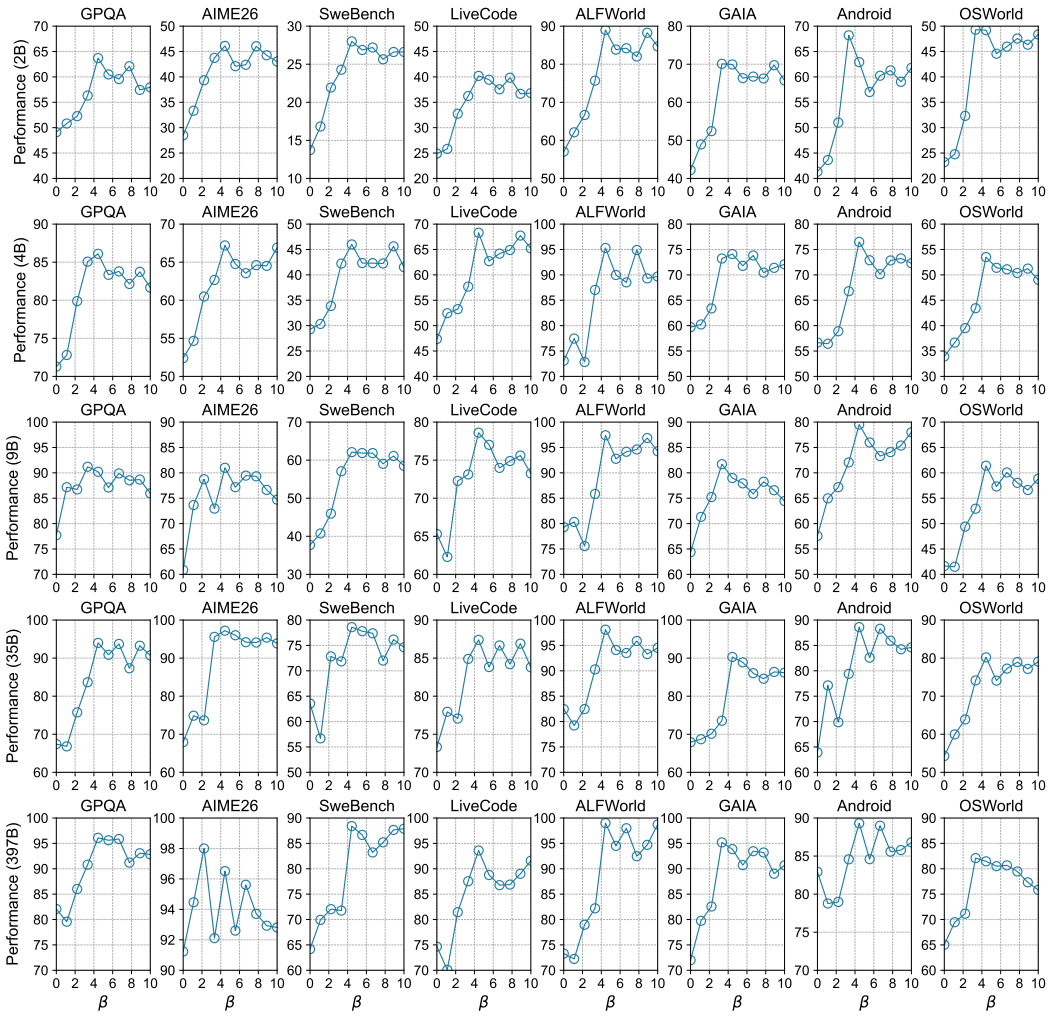


Figure 19: Performance with varying reweighting parameter β.

C.5 Ablation Studies

In this section, we assess the effect of key components by ablating them under the same setting.

Without skill extraction training, the agent's skill extraction capability gradually deteriorates as its policy improves. Although the policy initially benefits from internalizing previously extracted skills, these skills progressively become redundant, while the agent fails to continuously extract new skills that match its evolving improvement demands. Consequently, the quality of the extracted skills decreases, providing increasingly weak supervision for policy optimization and causing performance improvement to plateau or regress. Figs. 20 and 21 underscore the importance of jointly training skill extraction and policy optimization for sustained self-evolution.

Without the evolution-aware skill reward, the agent evaluates skills using their immediate skill-conditioned performance gains. As analyzed previously in Section 4, this intuitive objective induces gradient misalignment between skill extraction and policy optimization: skill extraction favors skills that produce large immediate policy changes, whereas policy internalization aims to reduce the policy's dependence on these skills. This conflict prevents the two objectives from improving jointly and leads to unstable long-term evolution. Figs. 20 and 21 highlight the necessity of evaluating skills from the perspective of long-term trajectory evolution to maintain skill-policy alignment.

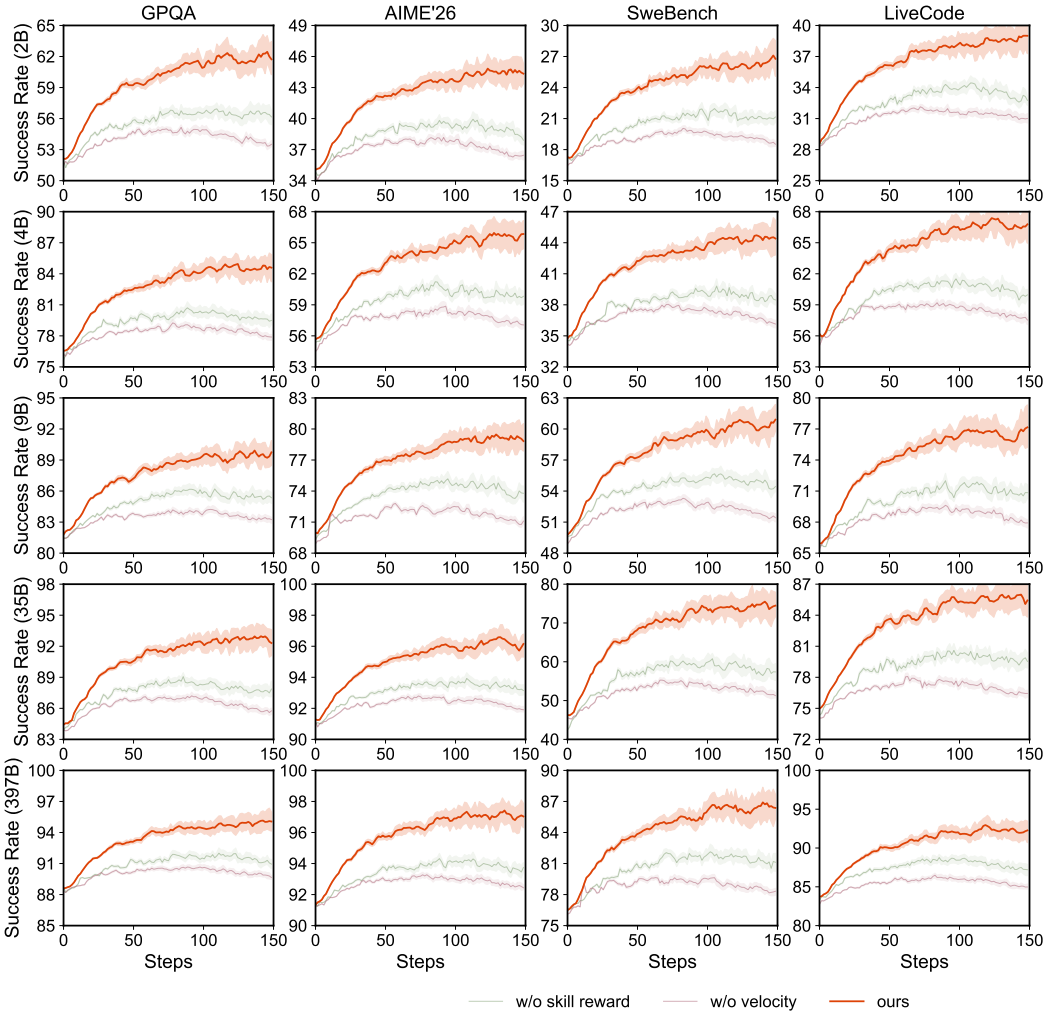


Figure 20: Performance comparison between SCALE and ablating key components, including skill reward and skill training.

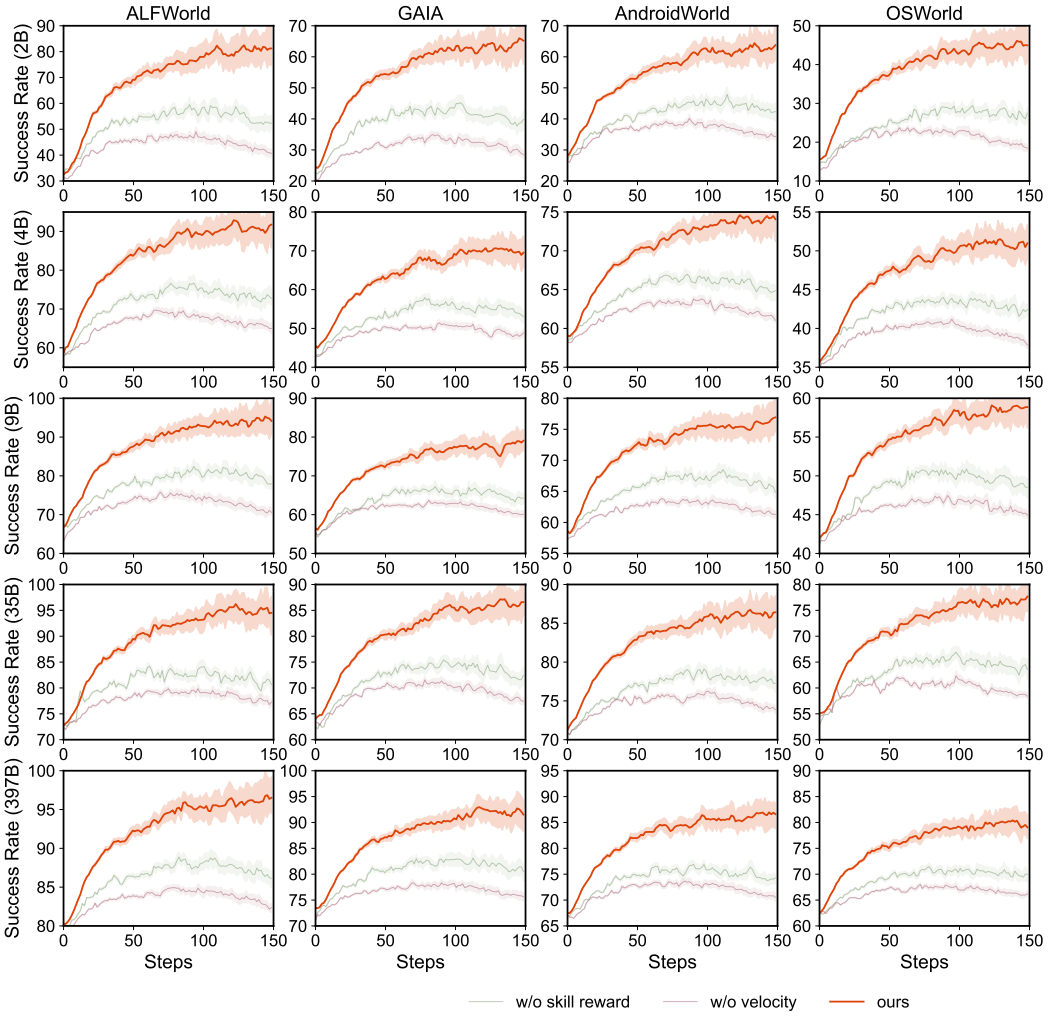


Figure 21: Performance comparison between SCALE and ablating key components, including skill reward and skill training.

C.6 Impact of Trajectory Evolution Step

We investigate the impact of trajectory evolution steps by replacing our convergence-based evolution with a fixed number of steps. As shown in Table 7, the result remains stable across different step choices, indicating that the learned velocity field generalize well in a larger region of the trajectory space to simulate evolution.

In addition, increasing the fixed step number generally improves performance, since additional evolution better captures the long-term behavioral effect of a skill. However, the gains gradually diminish, and all fixed-step variants remain inferior to SCALE across benchmarks and model scales. This indicates that different trajectories and skills evolve at different rates, making a single fixed step number unsuitable for all cases.

Importantly, SCALE does not determine the final evolved trajectory representation using a predefined number of evolution steps. Instead, it repeatedly applies the learned velocity field until two consecutive representations satisfy the convergence criterion $|z_j^i(c) - z_j^{i-1}(c)|_2 \leq \epsilon$. This adaptive strategy provides sufficient evolution for trajectories undergoing substantial changes while avoiding unnecessary updates for trajectories that have already stabilized. It therefore produces a more faithful evolved distribution and enables more reliable evaluation of the evolution-aware skill reward.

Table 7: Performance comparison with different numbers of fixed evolution steps. The best and second-best results are highlighted.

Model	Step	Science		Code		Agent		GUI	
		GPQA	AIME26	SweBench	LiveCode	ALFWorld	GAIA	Android	OSWorld
Qwen3.5-2B	Base	51.6	34.8	16.9	28.6	31.8	22.7	27.9	14.5
	1	57.4	40.2	21.3	33.2	53.5	40.7	41.2	26.0
	2	59.5	42.1	23.3	35.3	64.4	49.7	48.9	32.6
	3	61.0	44.3	25.1	37.9	73.8	57.6	56.1	38.8
	4	62.4	45.7	26.5	39.5	81.6	64.2	62.1	44.0
	5	63.2	45.4	27.4	39.2	86.7	68.5	66.0	47.4
	6	62.9	45.0	27.1	38.8	85.6	67.6	65.2	46.7
	SCALE	**63.7**	**46.1**	**28.0**	**40.2**	**89.0**	**70.1**	**68.2**	**49.3**
Qwen3.5-4B	Base	76.2	55.4	34.6	55.8	58.3	43.7	58.6	35.6
	1	81.0	61.1	39.2	60.8	72.4	55.3	64.5	41.5
	2	82.6	63.1	41.2	63.0	79.4	61.0	67.9	44.9
	3	84.4	64.6	43.7	65.0	85.4	68.2	70.3	47.9
	4	85.6	65.9	45.3	66.6	90.4	72.8	72.5	50.5
	5	85.3	66.7	45.0	67.7	93.6	72.2	74.2	52.1
	6	84.9	66.4	44.6	67.4	92.9	71.5	75.4	51.7
	SCALE	**86.1**	**67.2**	**46.0**	**68.3**	**95.3**	**74.1**	**76.5**	**53.5**
Qwen3.5-9B	Base	81.7	69.6	49.7	65.6	66.2	55.1	57.8	41.8
	1	86.3	75.1	54.7	70.8	78.1	65.2	65.0	48.3
	2	87.9	77.0	56.9	73.1	84.0	70.3	69.1	52.0
	3	89.1	79.1	58.8	76.0	87.9	74.6	74.6	55.5
	4	90.1	80.5	60.5	77.9	91.3	78.2	78.1	58.3
	5	90.7	80.2	61.5	77.6	94.2	80.5	77.7	60.2
	6	90.4	79.8	61.2	77.2	96.0	80.0	77.2	59.8
	SCALE	**91.2**	**81.0**	**62.1**	**78.6**	**97.4**	**81.7**	**79.5**	**61.4**
Qwen3.5-35B	Base	84.2	91.0	54.6	74.6	72.4	63.2	71.1	54.5
	1	88.9	94.0	64.2	79.7	82.2	73.5	76.9	63.0
	2	90.6	95.0	68.5	82.0	87.0	78.6	80.2	67.9
	3	92.4	96.9	72.3	84.9	91.2	82.0	83.2	74.6
	4	93.5	96.6	75.5	86.7	94.7	85.1	85.8	78.9
	5	93.2	96.2	77.6	86.4	96.9	87.6	87.4	78.4
	6	92.8	95.9	77.1	86.0	96.4	89.2	87.0	77.8
	SCALE	**94.0**	**97.2**	**78.6**	**87.4**	**98.1**	**90.3**	**88.6**	**80.2**
Qwen3.5-397B	Base	88.4	91.3	76.4	83.6	79.8	72.5	66.8	62.2
	1	92.1	94.5	81.2	87.6	87.1	81.1	74.2	68.8
	2	93.4	95.7	83.4	89.4	90.7	85.4	78.5	72.5
	3	95.7	96.9	86.1	91.0	95.3	89.1	81.7	77.7
	4	95.4	97.7	87.8	92.3	98.2	92.2	84.5	81.0
	5	95.0	97.4	87.5	93.1	97.8	94.2	86.8	80.6
	6	94.7	97.0	87.1	92.8	97.4	93.7	88.3	80.1
	SCALE	**96.1**	**98.0**	**88.4**	**93.6**	**99.0**	**95.2**	**89.3**	**82.1**

C.7 Skill Extraction Capability

We further examine the skill extraction capability throughout self-evolution using the skill metric introduced in SkillsBench (Li et al., 2026a). This metric evaluates whether an extracted skill provides effective and transferable guidance for solving downstream tasks.

As shown in Figs. 22 and 23, all methods initially achieve positive skill metrics because useful knowledge can be readily extracted from the trajectories generated by the base policy. As training proceeds, however, the skill metrics of baseline methods decrease rapidly and frequently approach zero or become negative. This degradation consistently occurs across model scales and is particularly pronounced on long-horizon agentic and GUI tasks. Although SEED and SDAR occasionally retain moderately positive values, their skill quality remains unstable and substantially lower than that of SCALE.

This degradation occurs because the reasoning patterns of the evolving policy gradually specialize towards problem-solving, while baseline methods fail to adapt their skill extraction capability accordingly. Their later skills therefore become less informative for the policy's current needs, weakening subsequent policy improvement. In contrast, SCALE maintains a consistently positive skill metric, demonstrating its ability to continuously extract informative skills that support sustained self-evolution.

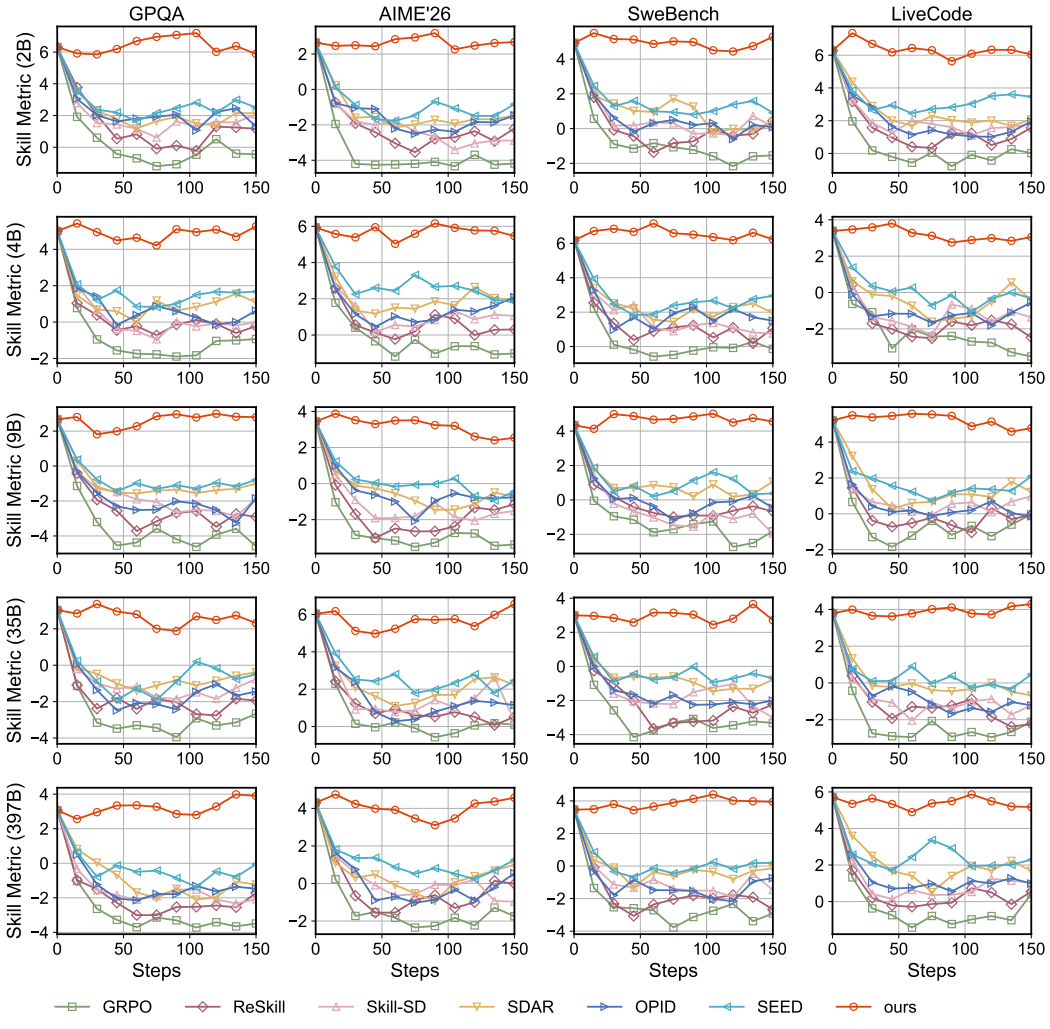


Figure 22: The value of skill metric of diverse methods during training in SkillsBench.

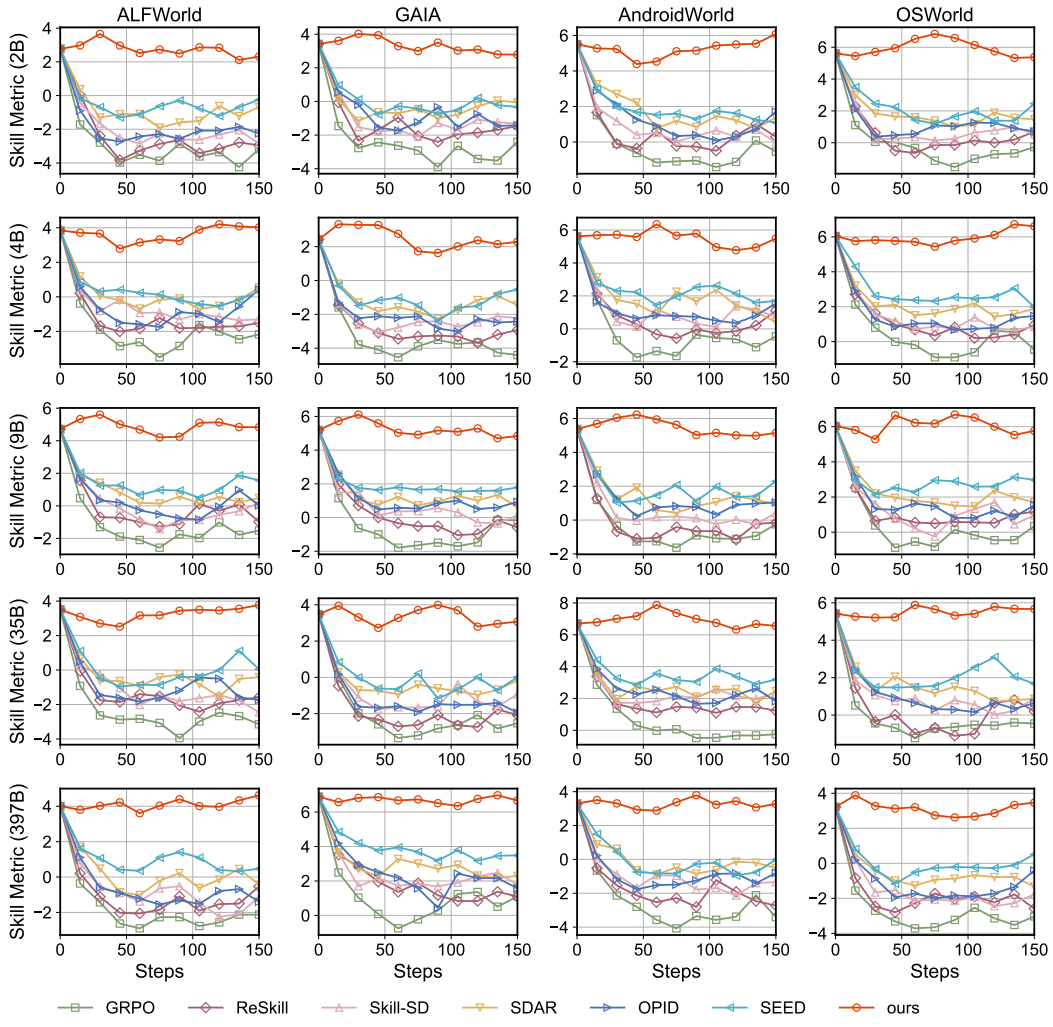


Figure 23: The value of skill metric of diverse methods during training in SkillsBench.

C.8 SKILL-POLICY GRADIENT SIMILARITY DISTRIBUTION

We validate the skill-policy misalignment by tracking the cosine similarity between the gradients of skill extraction and policy optimization. To clarify, we do not treat the performance gain in Eq. (7) as a detached reward for training skill extraction. In fact, skill extraction and policy optimization share the same parameters , and we differentiate the entire objective, including $(J(\pi_\theta))$. As shown in Fig. 24, the distribution of gradient similarity between skill extraction and policy improvement progressively shift toward negative values during training, indicating that skill extraction increasingly conflicts with policy improvement, as this objective may bring undesirable optimization behavior by degrading the policy performance for higher skill measurement.

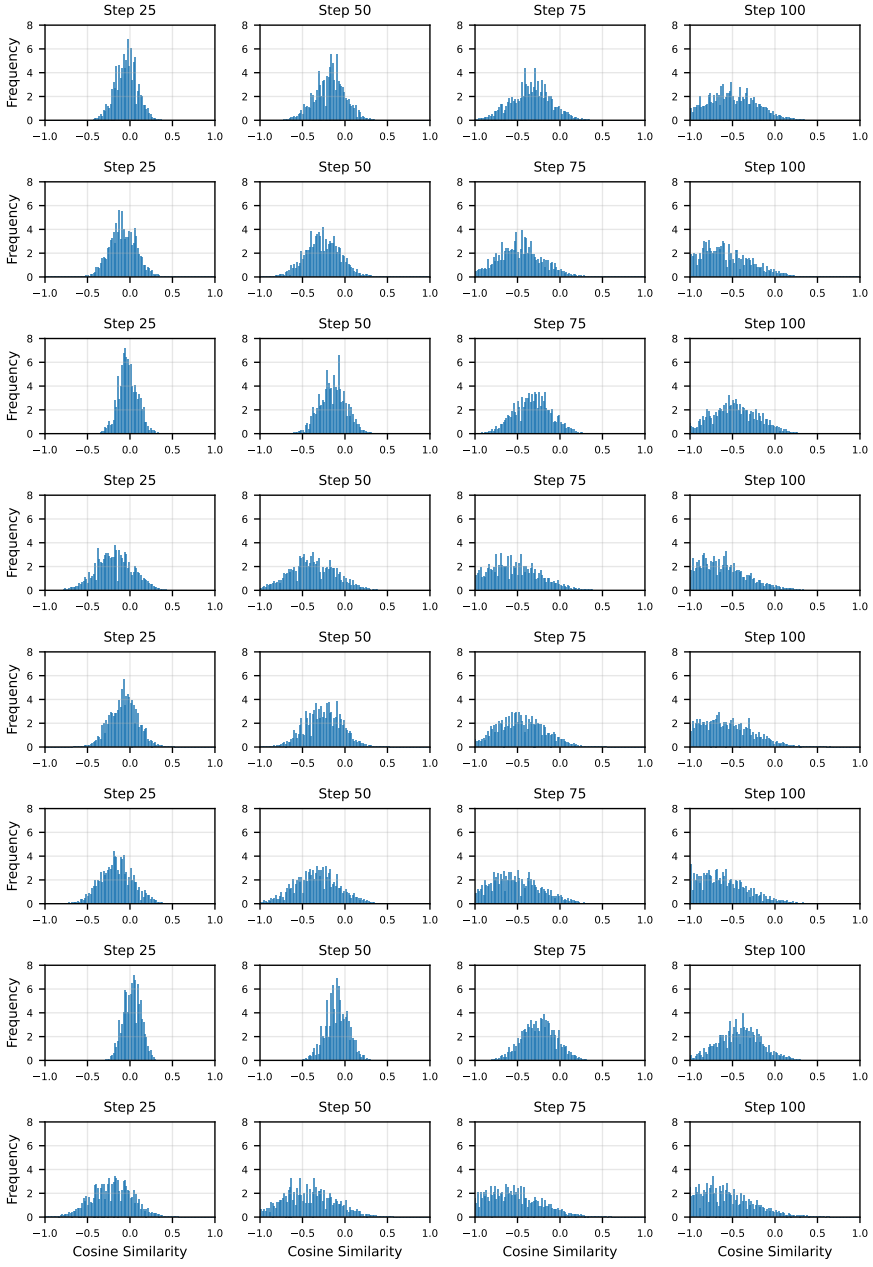


Figure 24: Illustration of the cosine similarity distributions of the naive skill extraction objective at different training steps across diverse environments. As training progresses, the cosine similarity increasingly shifts toward -1, indicating that skill extraction and policy optimization become progressively more conflicting and interfere with each other's optimization.

In contrast, as illustrated in Fig. 25, the distribution of SCALE gradually concentrates around 0 and 1. This indicates that skill extraction updates are either approximately orthogonal to policy optimization, thereby avoiding interference, or positively aligned with it, thereby jointly improving the policy. These results empirically validate the identified skill-policy misalignment and demonstrate that our evolution-aware skill training objective effectively alleviates gradient conflicts during self-evolution.

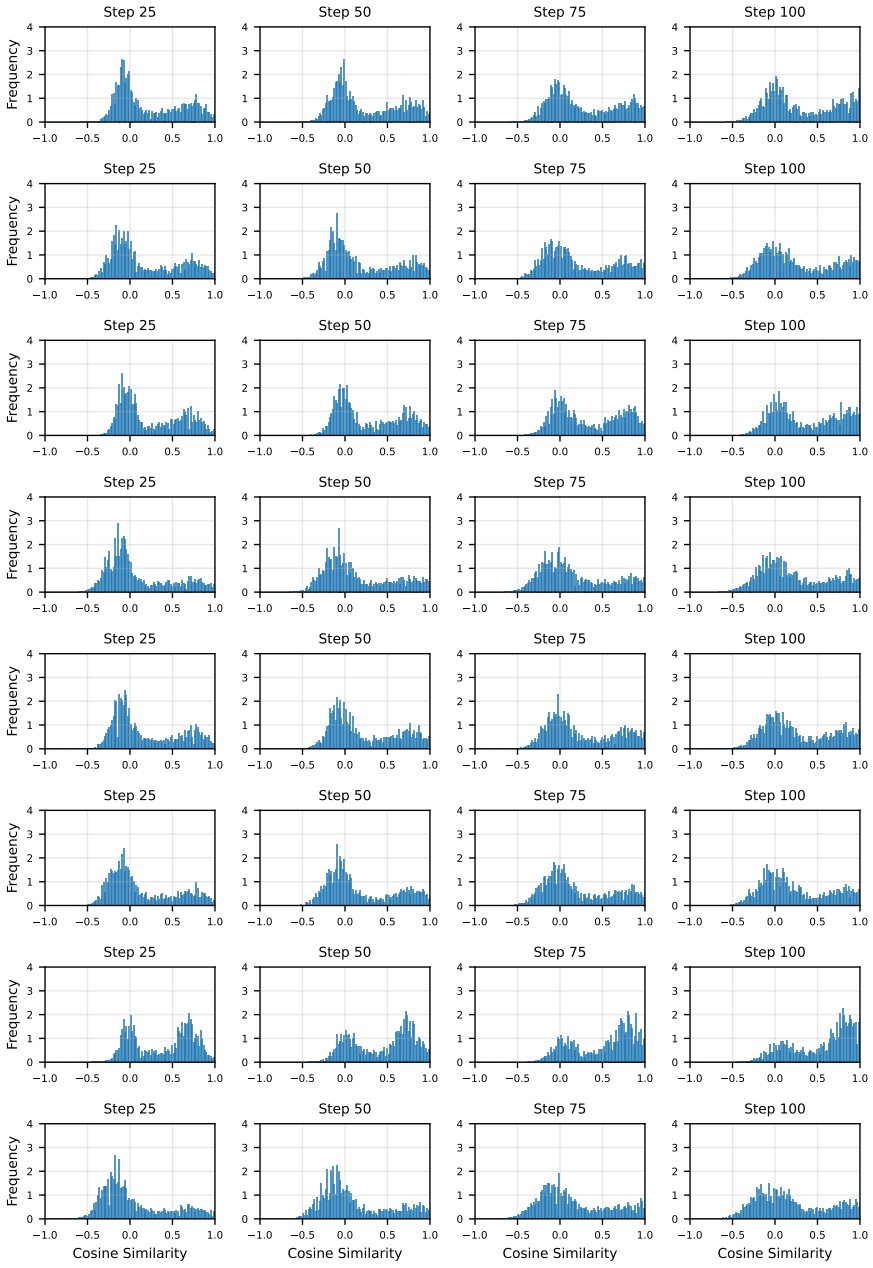


Figure 25: Illustration of the cosine similarity distributions of SCALE at different training steps across diverse environments. The distributions remain concentrated around 0 and 1, indicating that skill extraction is either largely non-interfering with or positively aligned with policy optimization.

C.9 CONVERGENCE

C.9.1 OPD LOSS

As shown in Figs. 26 and 27, the OPD loss decreases rapidly during the early training stage and subsequently stabilizes at a low value across benchmarks and model scales. This indicates that the policy can effectively internalize the behavioral guidance provided by the extracted skills. The mild fluctuations at later stages are expected because both the on-policy trajectories and skill-conditioned targets continuously change as the policy evolves.

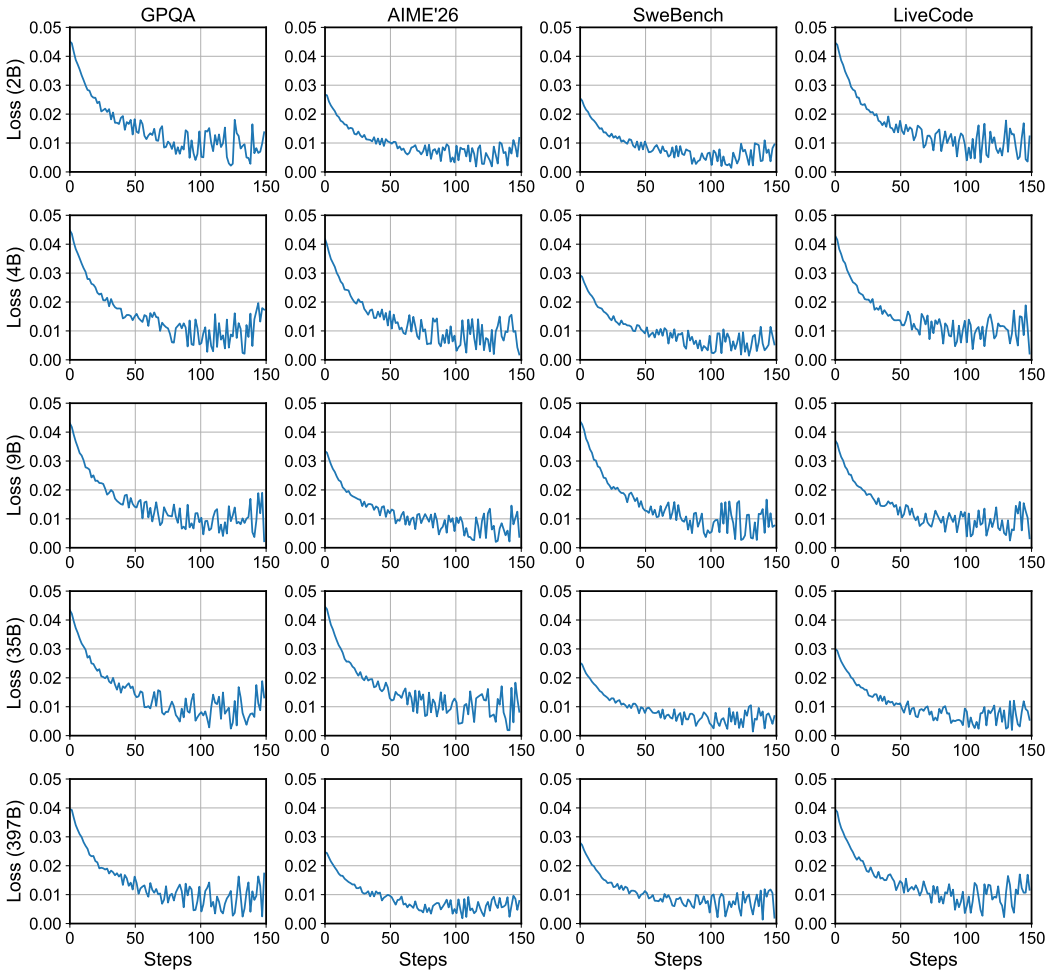


Figure 26: The value of the OPD loss. Its stable convergence demonstrates reliable optimization of the skill-induced evolution, which is crucial for ensuring consistent policy improvement throughout self-evolution.

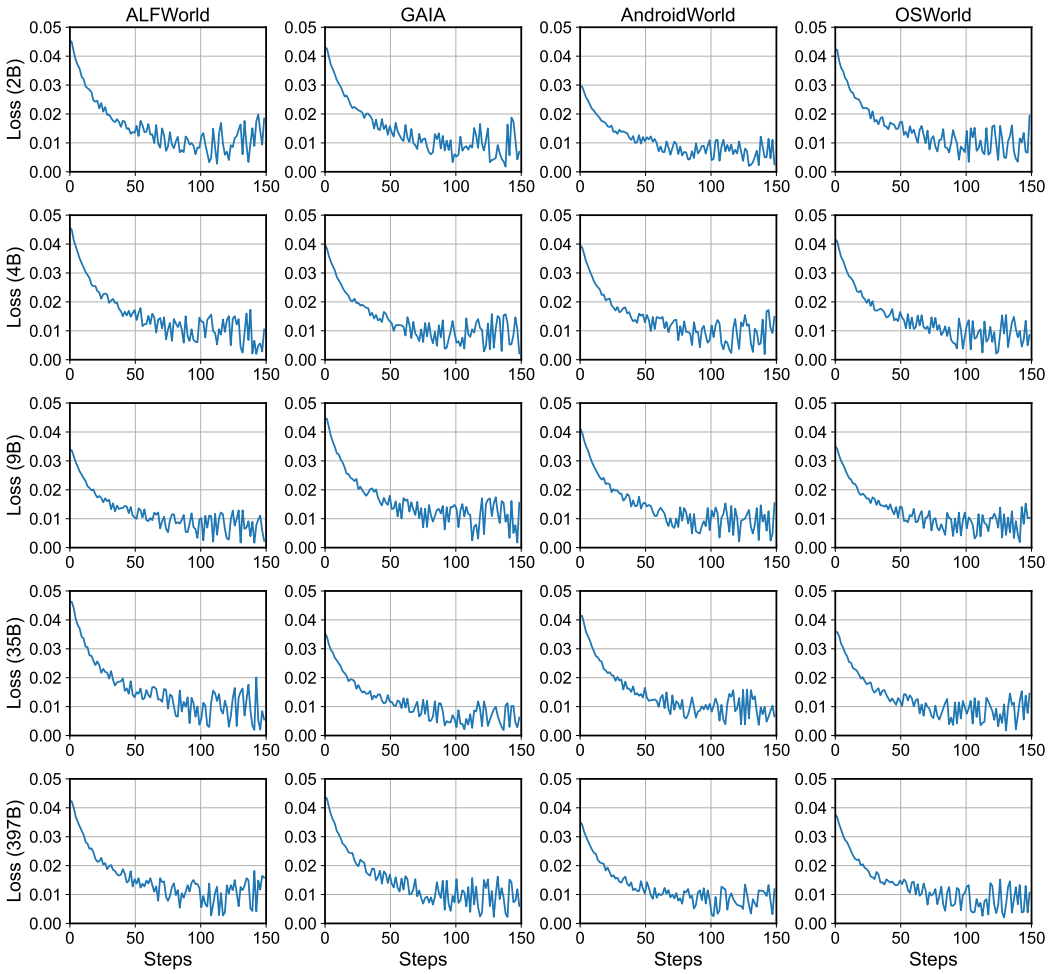


Figure 27: The value of the OPD loss.

C.9.2 SKILL LOSS

As illustrated in Figs. 28 and 29, the skill extraction loss exhibits consistent convergence across all settings. This demonstrates that the evolution-aware skill reward provides a stable learning signal for improving the agent's skill extraction capability, even though the policy and the distribution of useful skills continuously evolve during training.

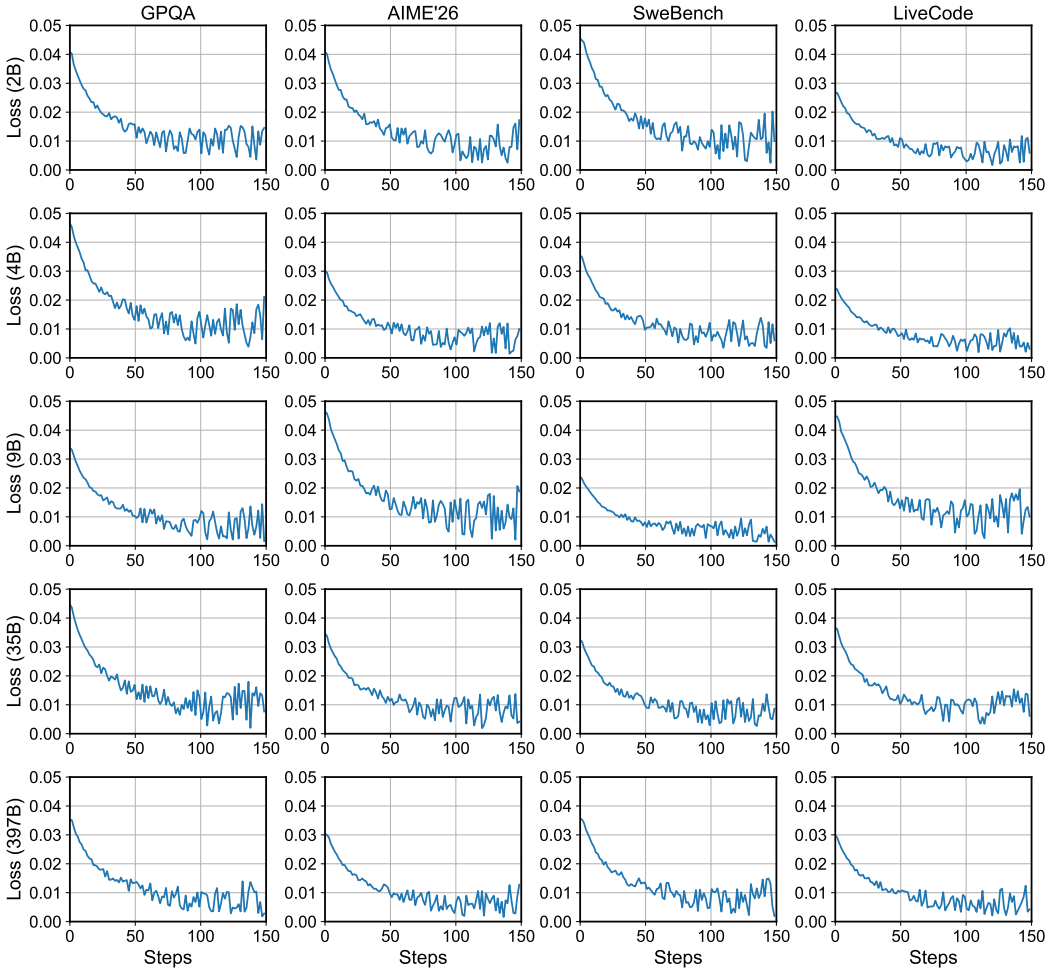


Figure 28: The value of the skill extraction loss. Its stable convergence suggests that the velocity field learned by SCALE generalizes well to simulate policy evolution, enabling the proposed evolution-aware reward to provide a reliable learning signal for stable skill extraction training.

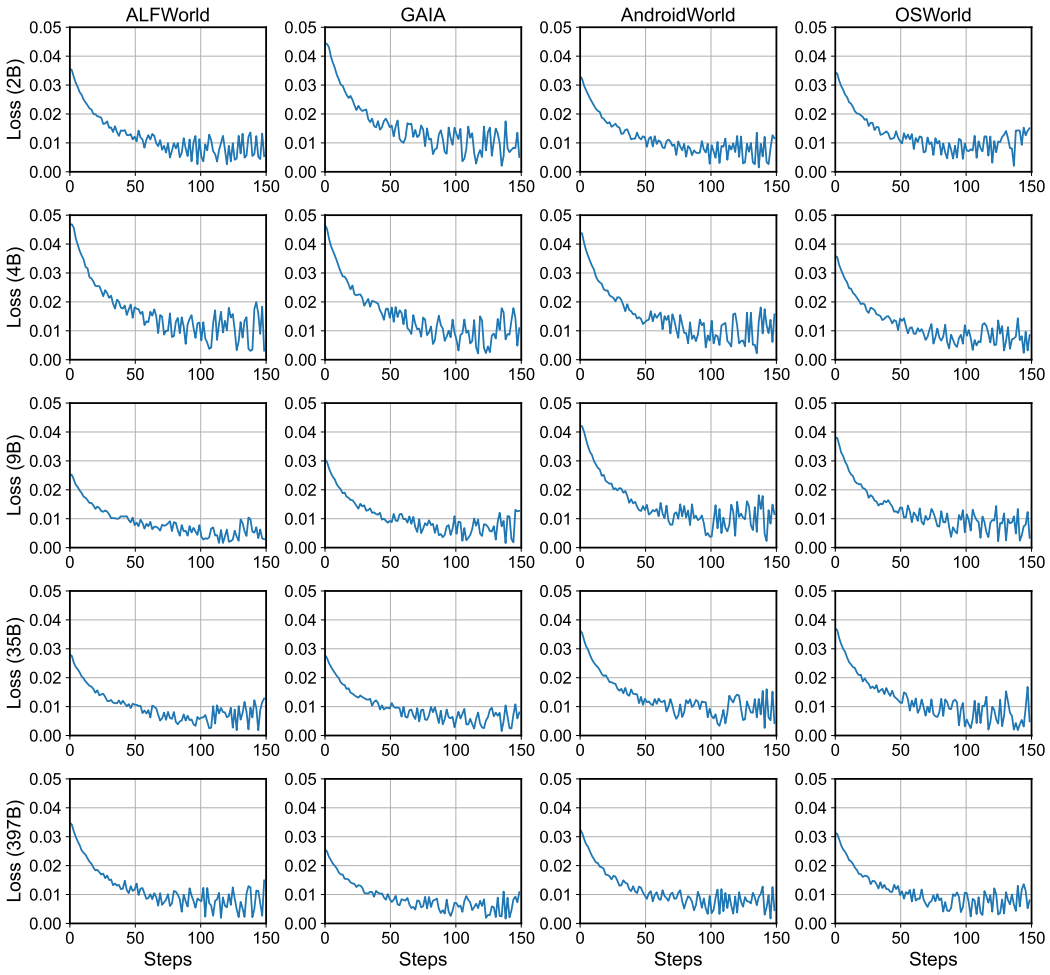


Figure 29: The value of the skill extraction loss.

C.9.3 VELOCITY LOSS

As shown in Figs. 30 and 31, the flow-matching loss decreases steadily and remains stable at a low level across tasks and model scales. The convergence of this objective is particularly important because the learned velocity field is subsequently used to estimate the skill-induced trajectory distribution and compute the evolution-aware skill reward. An inaccurate or unstable velocity field would introduce noise into skill evaluation. Therefore, the stable convergence observed here provides a reliable foundation for evaluating skill quality and jointly optimizing skill extraction and policy evolution.

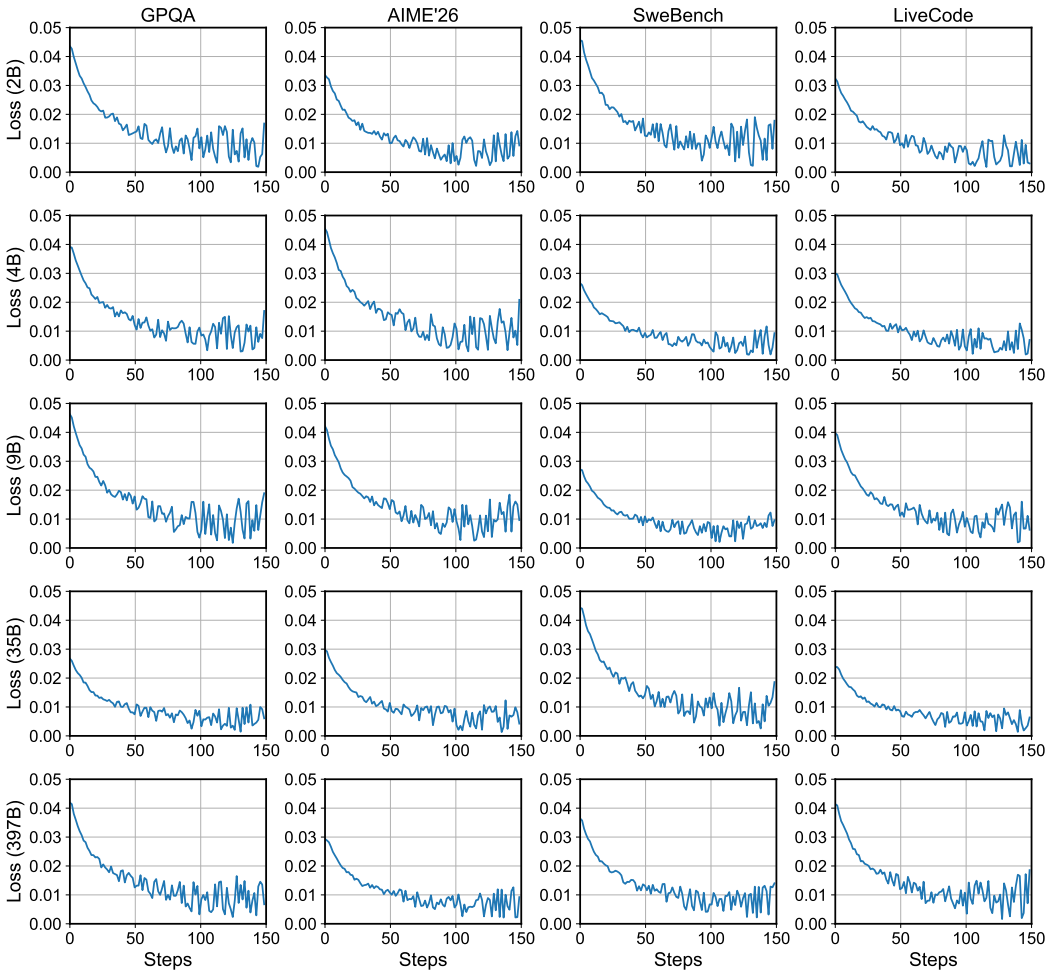


Figure 30: The value of the velocity field loss. Its stable convergence ensures that the learned velocity field can generalize the skill-induced transformation from limited trajectory samples to a broader trajectory space, thereby enabling reliable simulation of long-term policy evolution.

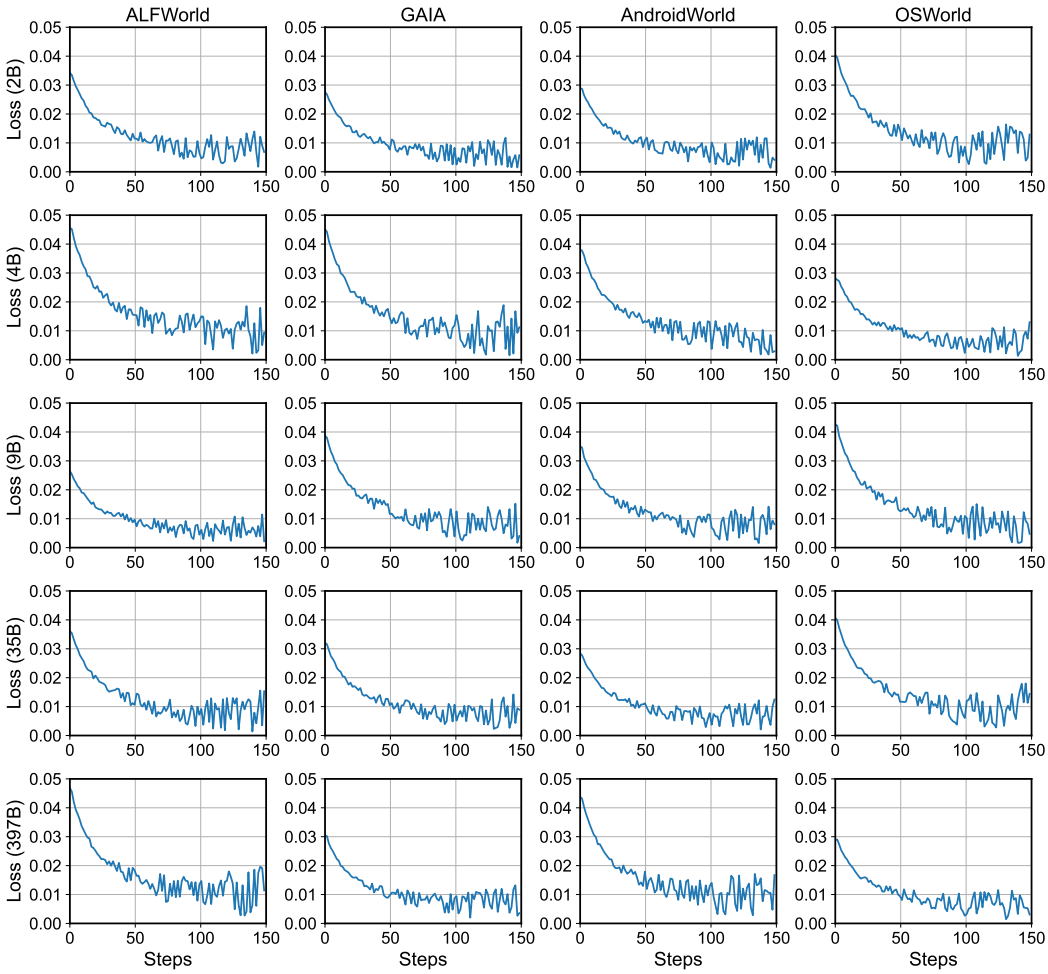


Figure 31: The value of the velocity field loss.

C.10 RUN TIME

We compare the training time of SCALE with all baselines on four representative benchmarks spanning mathematical reasoning (AIME), software engineering (SWE-bench), embodied decision making (ALFWorld), and GUI control (AndroidWorld). The runtime increases with task horizon and interaction complexity for all methods, with AndroidWorld requiring the largest cost due to its long-horizon visual interaction.

Compared with GRPO, SCALE incurs a moderate average overhead of $1.48\times$, mainly due to skill-conditioned trajectory modeling and velocity-field optimization. Importantly, its cost remains close to that of existing skill-based self-evolution methods, being only 12.2% slower than SEED on average. Since velocity-field evolution is performed in the trajectory-representation space, it avoids repeatedly performing expensive policy internalization and environment rollouts for long-term skill evaluation. Thus, SCALE maintains a practical per-iteration runtime while enabling evolution-aware skill evaluation across diverse agentic environments.

Table 8: Training time (minutes per self-evolution iteration) across four representative benchmarks. All methods use the same model (397B) and hardware configuration.

Method	AIME	SWE-bench	ALFWorld	AndroidWorld
GRPO	13.5	37.6	50.0	64.4
ReSkill	15.7	43.8	58.4	75.2
Skill-SD	14.5	40.2	53.7	69.1
SDAR	17.5	48.8	65.2	84.0
OPID	18.8	52.5	70.2	90.5
SEED	17.7	49.3	65.9	85.1
SCALE	19.8	55.3	74.1	95.7