Self-Improving Large Language Models via Progressive Experience Evolution

Shijie Ren[1], Xiting Wang[1]*, Meng Li[1], Yujie Guo[1], Yunhang Yao[1], Ziheng Peng[1]
Xunlong Wang[1], Yuetan Chen[1], Haoyang Zhou[1], Yunlong Liang[2], Fandong Meng[2]

[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] WeChat AI, Tencent Inc, China

Abstract

Large language models (LLMs) capable of self-improvement require not only effective policy optimization, but also a principled mechanism for transforming transient interaction experience into persistent model capabilities. Existing self-improvement paradigms remain fragmented: test-time methods can explicitly extract experience but cannot internalize it into model parameters, whereas training-time optimization methods can update model parameters but lack an explicit mechanism for accumulating transferable experience. Bridging these two paradigms requires a critical intermediate stage that remains underexplored, namely *experience distillation*. To address this gap, we propose **SPEE** (**S**elf-**P**rogressive **E**xperience **E**volution), a unified post-training framework that sequentially performs explicit experience evolution followed by implicit policy optimization. During explicit experience evolution, SPEE reflects on trajectories collected from multiple interactions to extract, verify, and progressively evolve transferable experience, which is subsequently internalized into the policy through privilege-guided On-Policy Self-Distillation (OPSD). During implicit policy optimization, reward-driven reinforcement learning leverages these internalized priors to explore novel solution strategies. In the experience evolution stage, a continuously evolving global experience pool consolidates knowledge from both successful and failed trajectories, filters out low-utility experience, and mitigates post-hoc rationalization induced by individual trajectories. Experiments on five mathematical reasoning benchmarks demonstrate that SPEE consistently outperforms both test-time and training-time self-evolution baselines across three model scales. The source code is available at https://github.com/rrrsj/SPEE.

Introduction

Recent advances in LLMs have achieved remarkable progress across a wide range of tasks, including mathematical reasoning, code generation, and autonomous agents (Achiam et al. 2023; Wei et al. 2022; Yao et al. 2022). However, as model scales continue to grow, offline training pipelines have become increasingly insufficient, as they heavily rely on large amounts of high-quality data (Kaplan et al. 2020; Ouyang et al. 2022). This limitation has motivated a shift toward self-evolving systems, where models are expected to autonomously adapt to new environments and acquire capabilities beyond their original training distributions (Wang et al. 2023a, 2026). Despite rapid progress, how models achieve stable and continual self-evolution remains unclear. Existing approaches are mainly categorized by optimization objectives rather than the mechanisms underlying capability improvement (Tao et al. 2024). We argue that the essence of self-evolution lies in transforming transient interaction signals into reusable and persistent model capabilities. In this work, we define *experience* as compact and transferable knowledge abstracted from interaction trajectories, including reusable reasoning strategies, task-invariant constraints, and recurring failure patterns. Unlike raw trajectories, effective experience should remove instance-specific details, generalize across problems, and evolve with the policy.

From the perspective of experience utilization, existing self-evolution paradigms can be broadly divided into test-time and training-time approaches. Test-time methods preserve demonstrations, reflections, or retrieved knowledge as additional context during inference (Mohamed, El Rashid, and Shaalan 2025; Debnath et al. 2025; Wang et al. 2023b). Although effective, such experience remains external to model parameters and is constrained by context capacity, retrieval accuracy, and generalization ability (Liu et al. 2024; Mueller et al. 2024). In contrast, training-time methods, particularly reinforcement learning (RL), internalize rewarded behaviors through parameter updates and enable exploration of new behaviors (Guo et al. 2025; Yu et al. 2026; Shao et al. 2024). However, in reinforcement learning, the experience contained in successful and failed trajectories typically influences parameter updates only indirectly through sparse reward signals. Consequently, the model may require more extensive exploration to achieve better performance and can be highly sensitive to the quality of the initial model.

Together, these limitations point to a missing intermediate stage in self-evolution: Before reward-based reinforcement learning, textual experience should first be explicitly extracted from trajectories and internalized into the policy through dense supervision. This provides a stronger initialization for reinforcement learning, thereby improving exploration efficiency and potentially enabling a higher performance ceiling. Existing online policy self-distillation methods (Hübotter et al. 2026) provide a natural mechanism for such internalization by conditioning a teacher policy

*Corresponding author.

on instance-level trajectories and distilling the resulting enhanced behaviors into a student policy. However, these methods directly rely on complete reference trajectories rather than explicitly constructing transferable experience. Because such trajectories are tied to individual problems and often contain solution-specific details or answer information, the teacher may overly rely on instance-specific cues, increasing the risk of post-hoc rationalization rather than acquiring strategies that generalize across problems. Consequently, existing methods face two limitations: (1) complete trajectories entangle transferable knowledge with instance-specific content, making it difficult to form compact experience representations; and (2) they do not provide a complete pipeline for acquiring, refining, internalizing, and further optimizing experience, thereby limiting systematic experience accumulation across interactions.

To address this challenge, we propose **SPEE** (**S**elf-**P**rogressive **E**xperience **E**volution), a unified framework that transforms transient interaction experience into persistent model capabilities through progressive experience evolution and policy optimization. During explicit experience evolution, SPEE maintains a continuously evolving global experience pool that extracts and refines transferable knowledge from successful and failed trajectories, reducing post-hoc rationalization caused by instance-specific details. The evolved experience is then internalized into the policy through OPSD, enabling the accumulation of reusable knowledge without human intervention. During implicit policy optimization, reinforcement learning further improves the policy by encouraging exploration beyond existing experience. By coupling experience evolution with policy optimization, SPEE establishes the basis for a self-reinforcing loop: accumulated experience improves the policy, while the enhanced policy can generate richer experience for future rounds of evolution. Experiments on five mathematical benchmarks show that SPEE consistently outperforms both test-time and training-time baselines across three model scales, achieving up to 6.96% improvement on qwen3-4b-base compared with the base model. These results demonstrate that SPEE provides an effective paradigm for continual self-improvement through progressive experience evolution and internalization.

Method

In this section, we first present an experience-centered perspective of LLMs' self-evolution, which serves as the conceptual foundation of SPEE. We then introduce explicit experience evolution, where transferable experience is extracted, evolved, and distilled into the policy. Finally, we present implicit policy optimization, where reward-driven exploration enables the policy to discover improved behaviors and generate richer experience for subsequent evolution, thereby closing the progressive self-improvement pipeline. The overall framework is illustrated in Figure 1.

Background: Experience-Centered Self-Evolution

We revisit LLMs' self-evolution from the perspective of experience, viewing its objective as transforming transient interaction trajectories into persistent model capabilities. An effective self-evolution framework should not only extract experience more effectively, but also make better use of it.

Let $\mathcal{D}$ denote the input distribution, π_θ denote the policy parameterized by θ. Given an input $q \sim \mathcal{D}$, the policy generates an output $y \sim \pi_\theta(\cdot \mid q)$, thereby forming an interaction trajectory $\mathbf{T} = \{\tau | \tau = (q, y)\}$. We further introduce $\mathbf{E}$ to represent the experience extracted from $\mathbf{T}$ by an experience extractor Σ, and $\mathbf{C}$ to represent the competence required to solve future tasks. To characterize the information relationships among these variables, we adopt mutual information as the measure. For two random variables X and Y, $I(X;Y)$ represents how much information one variable provides about the other. Mutual information increases as the statistical dependence between the two variables becomes stronger and equals zero when they are independent. The extracted experience should preserve information that is predictive of competence on future tasks while discarding instance-specific details that do not generalize. This objective can be revisited as:

$$\min_{p_\Sigma(\mathbf{E}|\mathbf{T})} I(\mathbf{T};\mathbf{E}) - \lambda I(\mathbf{E};\mathbf{C}), \tag{1}$$

where the first term encourages experience representations by penalizing unnecessary trajectory information. The second term preserves information predictive of future competence.

Under this view, different self-evolution methods can be interpreted as different approximations to the experience extractor Σ, differing primarily in how the extracted experience is represented and accumulated. Test-time methods explicitly preserve experience e and provide as additional context during inference: $\pi_\theta(y \mid q, e)$. This enables utilization of experience, but the knowledge remains external to the model and is never internalized into parameters. Consequently, test-time methods improve experience utilization without transforming external experience into persistent model capabilities. Reinforcement learning instead attempts to compress interaction experience directly into model parameters through reward-driven optimization. Formally, the policy maximizes the expected trajectory reward: $J_{\mathrm{RL}}(\theta) = \mathbb{E}_{\tau\sim p_\theta}\left[r(\tau)\right]$, and is updated via gradient ascent:

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta J_{\mathrm{RL}}(\theta), \tag{2}$$

where η denotes the learning rate. Rewarded behavioral patterns are gradually internalized into the policy. Standard reinforcement learning internalizes experience solely through reward signals, meaning that the information contained in trajectories can only be absorbed implicitly. When the sampled trajectories are of low quality, they may contain little information relevant to future competence: $I(\mathbf{T};\mathbf{C}) \approx 0$. As a result, the resulting supervision becomes weak, leading to sparse effective learning signals, inefficient exploration, and slow capability improvement.

The above analysis suggests that internalizing textual experience into the model before implicit policy optimization can not only improve sampling efficiency but also potentially lead to better performance. We refer to this process as *experience distillation*, whose objective is:

$$\pi_{\theta'}(y \mid q) \leftarrow \pi_\theta(y \mid q, e), \tag{3}$$

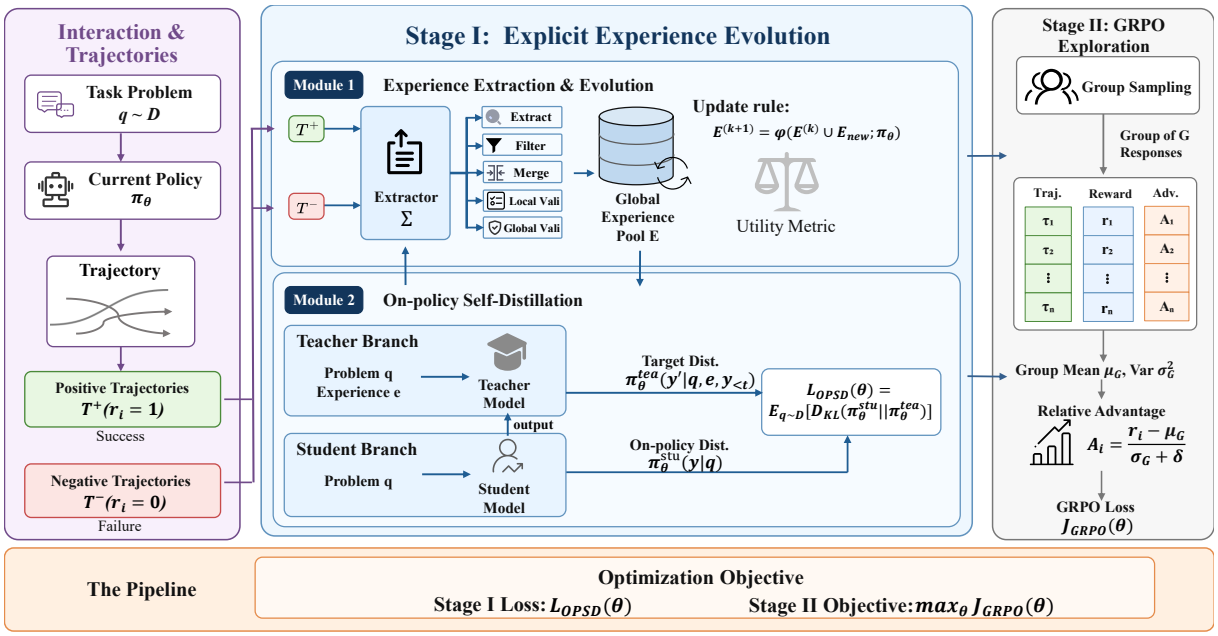

Figure 1: Overview of SPEE. Interaction trajectories are used to evolve a global experience pool, whose distilled knowledge improves the policy and supports more effective reward-driven exploration in subsequent iterations.

thereby bridging explicit experience utilization and implicit policy optimization. Motivated by this perspective, we design SPEE to first explicitly evolve transferable experience and then progressively internalize it into the policy.

Stage I: Explicit Experience Evolution

Rather than mechanically memorizing every detail, effective learners distill stable and generalizable principles from repeated attempts and reflection. Following this intuition, SPEE achieves the experience-distillation objective in Eq. 3 by maintaining a global experience pool rather than a static cache of trajectories or supervision. Specifically, SPEE first extracts transferable experience from trajectories and iteratively improves its reliability and transferability as the policy evolves. It then treats the refined experience as privileged information and internalizes it into the model parameters through OPSD. Accordingly, this stage consists of two components: experience evolution and experience distillation.

Experience evolution. SPEE evolves experience over multiple rounds, at round k, for each problem $q \sim \mathcal{D}$, the current policy π_{θ_k} samples G candidate responses:

$$y_i \sim \pi_{\theta_k}(\cdot \mid q), \quad i = 1, \ldots, G. \tag{4}$$

Each response induces a trajectory $\tau_i = (q, y_i)$ and receives a binary reward $r_i = r(q, y_i)$. We partition the resulting trajectories into successful and failed sets:

$$\mathcal{T}^+(q) = \{\tau_i \mid r_i = 1\}, \ \mathcal{T}^-(q) = \{\tau_i \mid r_i = 0\}. \tag{5}$$

To accelerate the process, we randomly select only S trajectories while preserving the empirical proportion of successful and failed responses. The experience extractor Σ_{θ_k}, implemented by the same model as the current policy under an experience-extraction prompt, converts each selected trajectory into a transferable experience item:

$$e_i^{(s_i)} = \Sigma_{\theta_k}(\tau_i, s_i), \ i \in \mathcal{I}_k(q). \tag{6}$$

Positive items summarize effective reasoning patterns and reusable solution strategies, whereas negative items identify invalid reasoning patterns and recurrent failure modes. Retaining both is particularly important: successful trajectories reveal behaviors to reinforce, while failed trajectories help prevent repeated exploration of ineffective paths.

Since these items are extracted from the current policy's behavior, they are treated as provisional hypotheses rather than final knowledge and may initially be incomplete, biased, or incorrect. We subsequently refine them through iterative evolution, allowing experience quality to improve alongside the policy. The newly extracted candidate set at round k is

$$\mathcal{E}_{\text{new}}^{(k)} = \bigcup_{q \sim \mathcal{D}} \left\{ e_i^{(s_i)} \mid i \in \mathcal{I}_k(q) \right\}. \tag{7}$$

We maintain the experience pool as a dynamic global memory. The newly extracted candidate set is integrated into a provisional pool:

$$\hat{\mathcal{E}}^{(k+1)} = \Phi\left(\mathcal{E}^{(k)} \cup \mathcal{E}_{\text{new}}^{(k)}; \pi_{\theta_k}\right), \tag{8}$$

where Φ is an evolution operator implemented by the current model. It merges overlapping items, resolves redundant or conflicting content, and abstracts trajectory-specific observations into more transferable strategies.

To ensure experience quality, we measure each experi-

ence's marginal utility on a held-out probe set $\mathcal{Q}_{\mathrm{pb}}$:

$$w^{(k)}(e) = \frac{1}{|\mathcal{Q}_{\mathrm{pb}}|} \sum_{q \in \mathcal{Q}_{\mathrm{pb}}} \left[R_k\left(q, \hat{\mathcal{E}}_e^{(k+1)}\right) - R_k\left(q, \mathcal{E}^{(k)}\right) \right], \tag{9}$$

where $R_k(q, \mathcal{E}) = \mathbb{E}_{y \sim \pi_{\theta_k}(\cdot|q,\mathcal{E})}\left[r(q,y)\right]$ denotes the expected reward obtained by the current policy when conditioned on $\mathcal{E}$. To balance effectiveness and efficiency, we apply a two-stage filter. Candidate items are first screened on a small subset of the probe set, and only those with positive utility are evaluated on the full probe set. Let

$$\widetilde{\mathcal{E}}^{(k)} = \left\{ e \in \hat{\mathcal{E}}_{\mathrm{new}}^{(k)} \mid w^{(k)}(e) > \epsilon \right\} \tag{10}$$

denote the accepted candidates, where $\epsilon \geq 0$ is a validation threshold. The global pool is then updated as

$$\mathcal{E}^{(k+1)} = \Phi\left(\mathcal{E}^{(k)} \cup \widetilde{\mathcal{E}}^{(k)}; \pi_{\theta_k}\right). \tag{11}$$

Repeated consolidation and validation yield the path

$$\mathcal{E}^{(0)} \to \mathcal{E}^{(1)} \to \cdots \to \mathcal{E}^{(M)}, \tag{12}$$

producing an increasingly reliable and transferable experience pool for subsequent distillation.

Privileged experience distillation. After several rounds of evolution, the pool $\mathcal{E}$ contains refined and high-utility experience. SPEE treats this experience as privileged information available only during training. We construct two branches with different information conditions. The teacher branch receives both the original problem and the global experience, while the student branch only observes the original problem. Both branches share the same underlying model parameters, while the teacher branch is detached from gradient computation. The student policy first generates trajectories following its current on-policy distribution:

$$y \sim \pi_\theta^{\mathrm{stu}}(\cdot|q). \tag{13}$$

The sampled trajectories are then rescored at the token level by the experience-augmented teacher policy in an on-policy manner, thereby providing a stronger target distribution:

$$\pi_{\bar{\theta}}^{\mathrm{tea}}(y'|q) \triangleq \pi_{\bar{\theta}}(y'|q, e, y_{<t}). \tag{14}$$

The student is optimized to improve its own on-policy behaviors by matching the teacher distribution. Specifically, SPEE minimizes the reverse KL divergence:

$$\mathcal{L}_{\mathrm{OPSD}}(\theta) = \mathbb{E}_{q \sim \mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi_\theta^{\mathrm{stu}}(\cdot|q) \| \pi_{\bar{\theta}}^{\mathrm{tea}}(\cdot|q)\right)\right]. \tag{15}$$

Stage II: Implicit Policy Optimization

Experience distillation internalizes the evolved experience into the policy, providing a strong initialization for further self-improvement. However, because this process learns from experience already represented in the pool, its gains are inherently limited by the pool's current coverage. To further expand the policy's capabilities, we apply Group Relative Policy Optimization (GRPO) to the distilled policy. Through reward-driven exploration, this stage discovers high-reward behaviors beyond the existing experience pool while directly improving the policy.

For each problem $q \sim \mathcal{D}$, we sample a group of G responses from the behavior policy. Let $r_i = r(q, y_i)$ denote the reward assigned to response y_i. GRPO computes the mean and variance of the rewards within each group as

$$\mu_G = \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad \sigma_G^2 = \frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_G)^2. \tag{16}$$

The group-relative advantage associated with response y_i is then defined as

$$A_i = \frac{r_i - \mu_G}{\sigma_G + \delta}, \tag{17}$$

where $\delta > 0$ is a small constant ensuring numerical stability. For each generated token $y_{i,t}$, the importance-sampling ratio between the current policy and the behavior policy is

$$\rho_{i,t}(\theta) = \frac{\pi_\theta\left(y_{i,t} \mid q, y_{i,<t}\right)}{\pi_{\theta_{\mathrm{old}}}\left(y_{i,t} \mid q, y_{i,<t}\right)}. \tag{18}$$

Following the clipped policy optimization objective, the token-level surrogate term is:

$$\ell_{i,t}^{\mathrm{clip}}(\theta) = \min\left\{\rho_{i,t}(\theta)A_i, \bar{\rho}_{i,t}(\theta)A_i\right\}, \tag{19}$$

where the clipped importance ratio is given by $\bar{\rho}_{i,t}(\theta) = \mathrm{clip}\left(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)$, and ϵ denotes the clipping coefficient. The overall GRPO objective is:

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{\substack{q \sim \mathcal{D}, \\ \mathbf{y} \sim \pi_{\theta_{\mathrm{old}}}(\cdot|q)}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\ell_{i,t}^{\mathrm{clip}}(\theta)\right], \tag{20}$$

The policy is optimized by maximizing $\mathcal{J}_{\mathrm{GRPO}}(\theta)$.

Unlike directly applying GRPO to the base checkpoint, SPEE initializes policy optimization from the policy obtained through self-evolving experience distillation, increases the likelihood of sampling rewarded trajectories and, more importantly, groups whose responses receive different reward values. Such groups produce non-zero relative advantages and therefore provide informative policy-gradient signals. In contrast, when all responses in a group receive the same reward, their normalized advantages become zero, resulting in an uninformative optimization step. By moving the policy toward high-reward regions before reinforcement learning, experience distillation improves exploration efficiency. We further analyze this effect in our experiments. It is worth noting that the second stage can also employ other variants of GRPO, which may yield better performance. For simplicity in both method design and experimental setup, we use only the standard GRPO algorithm in this work.

Experiments

Setting

We evaluate SPEE on three scales of the Qwen3 family: `qwen3-1.7b/4b/8b-base`. We control the number of training steps and the total number of sampled rollouts across different methods. We train models on DAPO-math-17k (Yu et al. 2026) and report results on five widely used mathematical benchmarks: AIME24/25,

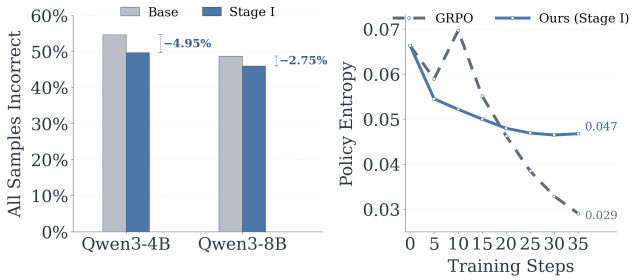

Figure 2: (Left) Proportion of problems for which all eight sampled responses are incorrect before and after Stage I. Experience distillation reduces the frequency of all-incorrect response groups. (Right) Policy entropy throughout training.

GSM8K, MATH500, and Minerva Math (Cobbe et al. 2021; Lightman et al. 2023; Zhang and Math-AI 2024; Zhang and Team 2025; Lewkowycz et al. 2022). These benchmarks cover competition-level olympiad problems, grade-school word problems, and general mathematical reasoning tasks. For GSM8K, MATH500, and Minerva Math, we report pass@1 results, while for AIME 2024 and AIME 2025, we report pass@16 average results, along with the performance change Δ relative to the base model checkpoint. We compare SPEE with four representative baselines: (i) the Base model; (ii) Domain Prompt, a test-time baseline which applies a manually designed mathematical reasoning prompt during inference; (iii) GRPO (Shao et al. 2024), a reinforcement learning baseline; and (iv) SDPO (Hübotter et al. 2026), a self-distillation-based method.

Main Results

Table 1 presents the main results. Across three model scales, SPEE achieves the highest average accuracy. Compared with the base model checkpoints, SPEE yields improvements of +4.87, +6.96, and +6.53 percentage points on the 1.7B, 4B, and 8B models, respectively. Compared with GRPO, SPEE achieves superior performance at all model scales, with an average improvement of 1.16%. Although GRPO can discover high-quality behaviors through reward-driven exploration, its effectiveness is constrained by the capability of the initial policy. In contrast, SPEE internalizes experience into the model parameters through experience distillation, thereby improving the efficiency of subsequent policy optimization. SPEE also consistently outperforms SDPO. Unlike directly distilling reference answers, which may cause the model to overfit specific reasoning paths and produce post-hoc rationalizations, SPEE employs a continuously evolving global experience pool to extract more abstract and transferable reasoning patterns. The comparison with Domain Prompt further highlights the importance of parameter-level experience accumulation. Overall, these results validate SPEE as an effective framework for supporting the continual self-evolution of large language models.

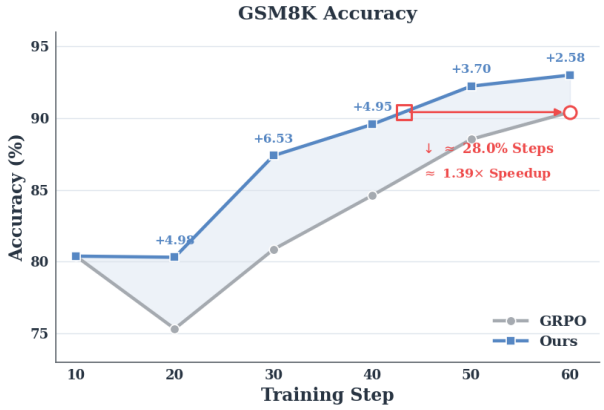

Figure 3: Data efficiency comparison between SPEE and GRPO. SPEE achieves comparable or superior performance with substantially fewer training trajectories.

Ablation Study

To validate the effectiveness of each component in our method, we conduct an ablation study. Table 2 evaluates the contributions of the two main components of SPEE. Removing Stage II decreases the average performance from 36.67% to 35.16% on the 4B model and from 38.71% to 36.14% on the 8B model, demonstrating the importance of reward-driven policy optimization. Removing the shared experience pool leads to a further performance drop, with the average accuracy decreasing to 34.13% and 35.48% on the 4B and 8B models, respectively. These results indicate that both components are essential to SPEE, while the shared experience pool plays a particularly important role in accumulating and transferring reusable experience across different problems. Overall, the complete SPEE framework achieves the best average performance at both model scales.

Training dynamics

Rollout Diversity. A key premise of our method is that Stage I improves policy performance without prematurely reducing policy entropy. To validate this premise, we compare the base policy with the policy obtained after Stage I in terms of both the proportion of sampled responses receiving positive rewards and the policy entropy during generation. We perform 8 rollouts for each problem. As shown in Figure 2 (left), Stage I substantially reduces the proportion of problems for which all sampled responses are incorrect. Consequently, the policy is more likely to generate response groups containing both correct and incorrect solutions, thereby providing more informative gradient signals for GRPO. As shown in Figure 2 (right), the policy after Stage I still maintains an entropy level comparable to that of the base model. These results indicate that experience distillation shifts the policy toward higher-reward regions while avoiding premature policy collapse or excessive restriction of response diversity.

Data Efficiency. As shown in Figure 3, benefiting from the dense supervision provided in Stage I and the higher effective sampling efficiency achieved in Stage II, SPEE attains performance comparable to or even better than GRPO

Model	Method	AIME24	AIME25	GSM8K	MATH500	MINA	Average	Δ
qwen3-1.7b	Base	1.45	4.16	71.44	39.90	11.49	25.69	–
	Domain Prompt	1.67	2.50	74.07	39.30	13.20	26.15	+0.46
	GRPO	5.00	6.67	80.81	42.50	**14.07**	29.81	+4.12
	SDPO	0.00	1.67	71.92	39.40	9.56	24.51	-1.18
	SPEE (Ours)	**5.21**	**7.71**	**82.13**	**43.80**	13.97	**30.56**	**+4.87**
qwen3-4b	Base	7.91	6.46	73.10	45.53	15.54	29.71	–
	Domain Prompt	7.29	9.17	85.59	48.00	14.06	32.82	+3.11
	GRPO	10.00	4.79	89.73	49.75	19.49	34.75	+5.04
	SDPO	6.67	**10.83**	83.57	48.65	18.75	33.69	+3.98
	SPEE (Ours)	**11.04**	8.75	**90.83**	**52.80**	**19.94**	**36.67**	**+6.96**
qwen3-8b	Base	7.50	6.87	80.38	46.85	19.30	32.18	–
	Domain Prompt	4.17	6.87	89.82	48.50	21.97	34.27	+2.09
	GRPO	9.17	11.29	92.47	52.80	23.86	37.92	+5.74
	SDPO	3.33	11.25	90.26	49.15	21.88	35.17	+2.99
	SPEE (Ours)	**10.60**	**12.08**	**93.14**	**53.70**	**24.06**	**38.71**	**+6.53**

Table 1: Performance comparison of different post-training methods across mathematical benchmarks.

Model	Method	AIME24	AIME25	GSM8K	MATH500	MINA	Average	Δ
qwen3-4b	Base	7.91	6.46	73.10	45.53	15.54	29.71	–
	w/o Stage2	6.46	**8.75**	89.61	50.00	**20.96**	35.16	+5.45
	w/o Shared Experience Pool	6.67	8.33	87.32	49.74	18.57	34.13	+4.42
	SPEE (Ours)	**11.04**	**8.75**	**90.83**	**52.80**	19.94	**36.67**	**+6.96**
qwen3-8b	Base	7.50	6.87	80.38	46.85	19.30	32.18	–
	w/o Stage2	7.08	6.67	92.27	52.60	22.06	36.14	+3.96
	w/o Shared Experience Pool	5.00	8.13	88.38	52.35	23.53	35.48	+3.30
	SPEE (Ours)	**10.60**	**12.08**	**93.14**	**53.70**	**24.06**	**38.71**	**+6.53**

Table 2: Ablation study of different components in SPEE. Δ denotes the improvement over the corresponding base model.

while using substantially fewer training trajectories. SPEE first extracts transferable knowledge from both successful and failed trajectories during the experience evolution stage and internalizes it into the policy in advance, enabling the initialized policy to generate more effective responses. Experimental results show that, at the same performance level, SPEE requires approximately 28% less training data than GRPO, demonstrating superior data utilization efficiency.

Effect of Experience Evolution Iterations

To investigate how progressive experience evolution affects the sampling capability of the policy, we conduct an additional controlled experiment by varying the number of experience evolution iterations from 0 to 4 while keeping the remaining configuration unchanged. We report *sampling accuracy*, defined as the proportion of sampled responses that are correct and receive a positive reward. Therefore, this metric reflects the ability of the current policy to generate correct trajectories during sampling, rather than its final evaluation accuracy on downstream benchmarks. Iteration 0 denotes sampling without experience evolution, while each subsequent iteration introduces an additional round of experience extraction, consolidation, filtering, and validation. As shown in Figure 4, the sampling accuracy consistently increases as the experience pool evolves. Specifically, it improves from 18.83% at iteration 0 to 20.55%, 21.33%, 21.80%, and 22.66% after one to four evolution iterations, respectively. After four iterations, the sampling accuracy increases by 3.83 percentage points, corresponding to a relative improvement of approximately 20.34% over the setting without experience evolution. These results indicate that repeated experience evolution progressively improves the policy's ability to sample correct trajectories. Rather than merely accumulating additional textual information, the evolving experience pool consolidates overlapping knowledge, filters unreliable items, and preserves transferable reasoning patterns that provide more effective guidance during generation. As the quality of the experience pool improves, the policy is more likely to produce positively rewarded responses, thereby providing higher-quality trajectories and more informative learning signals for subsequent policy optimization. Although the main experiments adopt a single evolution round for computational efficiency, the consistent improvement in sampling accuracy suggests that multiple rounds of experience evolution may further strengthen the self-improvement loop.

Qualitative Analysis

Figure 5 illustrates how evolved experience improves reasoning behavior. For a problem requiring the last two digits to satisfy both a digit-sum constraint and the divisibility-by-4

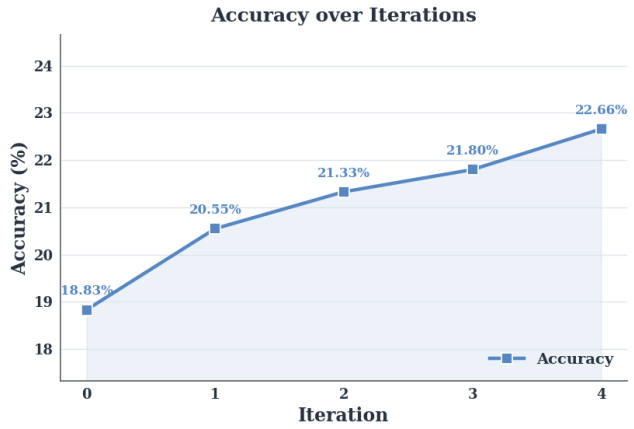

Figure 4: Accuracy under different numbers of experience evolution iterations. Iteration 0 denotes the setting without experience evolution. Performance consistently improves as the experience pool undergoes additional rounds of evolution.

rule, the baseline model fails to systematically enumerate all candidates whose digits sum to 13, introduces several invalid candidates, and ultimately produces the incorrect answer. After incorporating the experience, the model adopts a more structured two-stage reasoning procedure. It first exhaustively enumerates all ordered digit pairs, and then verifies whether each corresponding two-digit number is divisible by 4. This procedure yields the correct result $7 \times 6 = 42$. Importantly, the injected experience contains neither the answer 42 nor any candidates specific to this problem. Instead, it provides transferable reasoning principles, such as decomposing the problem into smaller steps. Therefore, the improvement does not result from answer memorization, but from the adoption of a more reliable reasoning procedure.

Related Work

Prompt-Based Experience Utilization

LLMs exhibit strong in-context learning capabilities, allowing them to adapt their behaviors by leveraging external information provided in prompts. Methods such as chain-of-thought prompting and self-consistency demonstrate that demonstrations and reasoning trajectories can serve as explicit experience to guide model reasoning (Yao et al. 2023; Wang et al. 2022; Besta et al. 2024). Retrieval-augmented generation further extends this paradigm by incorporating additional knowledge from external databases to enhance model capabilities (Lewis et al. 2020; Jin et al. 2025). Recent studies on agents introduce long-term memory mechanisms that store historical observations, actions, and reflection results, enabling long-horizon interactions (Park et al. 2023; Zhao et al. 2024; Shinn et al. 2023). However, such experience remains external to model parameters, and its effectiveness depends on retrieval quality and context capacity. Once the relevant information is removed from the context, the acquired knowledge cannot be directly preserved as intrinsic model capabilities (Liu et al. 2024).

Reinforcement Learning

Reinforcement learning (RL) improves the capabilities of large language models (LLMs) by optimizing model policies through environmental feedback. Early approaches, such as reinforcement learning from human feedback (RLHF), align model behaviors with human preferences by learning reward signals and applying policy optimization algorithms (Ouyang et al. 2022; Stiennon et al. 2022; Ziegler et al. 2020). With the emergence of automatically verifiable tasks, reinforcement learning has been widely applied to reasoning tasks, where rewards can be obtained through verifiers, or rule-based evaluators (Luong et al. 2024; Shao et al. 2024; Le et al. 2022). Large-scale reward-driven training enables models to autonomously develop complex behaviors, including longer reasoning processes, backtracking, and strategy adjustment. These advances have gradually transformed reinforcement learning from a preference alignment technique into an important post-training paradigm for enhancing reasoning and exploration capabilities (Team et al. 2025; Jin et al. 2025; Wang et al. 2025). However, reinforcement learning heavily relies on the strength of the initial model, while experience is primarily absorbed through scalar reward signals. When the model capability is limited, sparse rewards and inefficient exploration can lead to suboptimal performance.

On-Policy Self-Distillation

Knowledge distillation transfers knowledge from a teacher model to a student model, enabling effective knowledge transfer between models (Hinton, Vinyals, and Dean 2015). On-policy distillation further improves this process by sampling trajectories from the current policy, ensuring that the supervision signal matches the behavior distribution of the student model itself (Agarwal et al. 2024; Gu et al. 2026). In self-evolving scenarios, existing methods typically construct teacher models using privileged information and directly distill complete answers, reasoning trajectories, or teacher response distributions (Penaloza et al. 2026; Zhao et al. 2026). However, such trajectory-level supervision signals often contain substantial problem-specific details, causing student models to imitate particular solution processes rather than learn generalizable strategies shared across different examples (Shi et al. 2026). Compared with these methods, we introduce a continuously evolving experience representation that extracts, integrates, and filters transferable knowledge, thereby enabling more effective experience internalization.

Conclusion and Limitation

In this work, we introduce SPEE, a unified post-training framework for transforming transient interaction experience into persistent model capabilities. SPEE closes the loop between explicit experience utilization and implicit policy optimization by extracting and refining transferable experience, internalizing it through self-distillation, and further improving the policy through reinforcement learning. Experiments across five mathematical reasoning benchmarks and three model scales demonstrate consistent gains over test-time and training-time self-evolution baselines. Future work may extend this experience-evolution paradigm to a broader range of

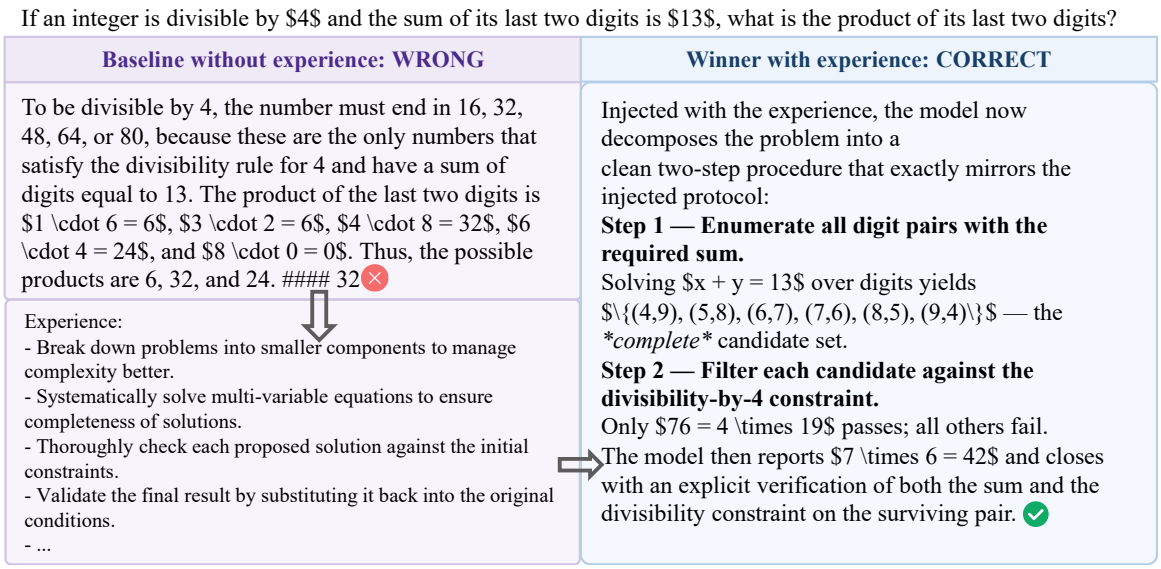

Figure 5: A representative case illustrating the effect of experience-guided reasoning. Without experience, the model constructs an incomplete candidate set and produces an incorrect answer. After incorporating evolved experience, the model systematically enumerates all feasible digit pairs and verifies each candidate against the divisibility constraint, obtaining the correct answer.

task domains and application scenarios with sparser, less reliable, or more complex environmental feedback. The framework supports closed-loop iteration, future work may further explore multiple rounds of methods.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Agarwal, R.; Vieillard, N.; Zhou, Y.; Stanczyk, P.; Ramos, S.; Geist, M.; and Bachem, O. 2024. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. arXiv:2306.13649.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Podstawski, M.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Niewiadomski, H.; Nyczyk, P.; et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 17682–17690.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Debnath, T.; Siddiky, N. A.; Rahman, M. E.; Das, P.; and Guha, A. K. 2025. A Comprehensive Survey of Prompt Engineering Techniques in Large Language Models. *Authorea Preprints*.

Gu, Y.; Dong, L.; Wei, F.; and Huang, M. 2026. MiniLLM: On-Policy Distillation of Large Language Models. arXiv:2306.08543.

Guo, D.; Yang, D.; Zhang, H.; Song, J.; Wang, P.; Zhu, Q.; Xu, R.; Zhang, R.; Ma, S.; Bi, X.; et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531.

Hübotter, J.; Lübeck, F.; Behric, L. D.; Baumann, A.; Bagatella, M.; Marta, D.; Hakimi, I.; Shenfeld, I.; Buening, T. K.; Guestrin, C.; and Krause, A. 2026. Reinforcement Learning via Self-Distillation. In *Forty-third International Conference on Machine Learning*.

Jin, B.; Zeng, H.; Yue, Z.; Yoon, J.; Arik, S.; Wang, D.; Zamani, H.; and Han, J. 2025. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv:2503.09516.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Le, H.; Wang, Y.; Gotmare, A. D.; Savarese, S.; and Hoi, S. C. H. 2022. CodeRL: Mastering Code Generation through Pretrained Models and Deep Reinforcement Learning. arXiv:2207.01780.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 9459–9474. Curran Associates, Inc.

Lewkowycz, A.; Andreassen, A.; Dohan, D.; Dyer, E.; Michalewski, H.; Ramasesh, V.; Slone, A.; Anil, C.; Schlag, I.; Gutman-Solo, T.; et al. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35: 3843–3857.

Lightman, H.; Kosaraju, V.; Burda, Y.; Edwards, H.; Baker, B.; Lee, T.; Leike, J.; Schulman, J.; Sutskever, I.; and Cobbe, K. 2023. Let's Verify Step by Step. *arXiv preprint arXiv:2305.20050*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173.

Luong, T. Q.; Zhang, X.; Jie, Z.; Sun, P.; Jin, X.; and Li, H. 2024. ReFT: Reasoning with Reinforced Fine-Tuning. arXiv:2401.08967.

Mohamed, A.; El Rashid, M.; and Shaalan, K. 2025. In-context learning in large language models (LLMs): Mechanisms, capabilities, and implications for advanced knowledge representation and reasoning. *IEEE Access*, 13: 95574–95593.

Mueller, A.; Webson, A.; Petty, J.; and Linzen, T. 2024. In-context Learning Generalizes, But Not Always Robustly: The Case of Syntax. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 4761–4779. Mexico City, Mexico: Association for Computational Linguistics.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

Park, J. S.; O'Brien, J.; Cai, C. J.; Morris, M. R.; Liang, P.; and Bernstein, M. S. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Penaloza, E.; Vattikonda, D.; Gontier, N.; Lacoste, A.; Charlin, L.; and Caccia, M. 2026. Privileged Information Distillation for Language Models. arXiv:2602.04942.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y. K.; Wu, Y.; and Guo, D. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Shi, T.; Huang, C.; Li, B.; Chen, X.; Quan, X.; Wang, J.; and Wang, Q. 2026. Beyond Trajectory Imitation: Strategy-Guided Policy Optimization for LLM Reasoning. arXiv:2606.24064.

Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36: 8634–8652.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D. M.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. 2022. Learning to summarize from human feedback. arXiv:2009.01325.

Tao, Z.; Lin, T.; Chen, X.; Li, H.; Wu, Y.; Li, Y.; Jin, Z.; Huang, F.; Tao, D.; and Zhou, J. 2024. A survey on self-evolution of large language models. arXiv. 240414387 (2024).

Team, K.; Du, A.; Gao, B.; Xing, B.; Jiang, C.; Chen, C.; Li, C.; Xiao, C.; Du, C.; Liao, C.; Tang, C.; Wang, C.; Zhang, D.; Yuan, E.; Lu, E.; Tang, F.; Sung, F.; Wei, G.; Lai, G.; Guo, H.; Zhu, H.; Ding, H.; Hu, H.; Yang, H.; Zhang, H.; Yao, H.; Zhao, H.; Lu, H.; Li, H.; Yu, H.; Gao, H.; Zheng, H.; Yuan, H.; Chen, J.; Guo, J.; Su, J.; Wang, J.; Zhao, J.; Zhang, J.; Liu, J.; Yan, J.; Wu, J.; Shi, L.; Ye, L.; Yu, L.; Dong, M.; Zhang, N.; Ma, N.; Pan, Q.; Gong, Q.; Liu, S.; Ma, S.; Wei, S.; Cao, S.; Huang, S.; Jiang, T.; Gao, W.; Xiong, W.; He, W.; Huang, W.; Xu, W.; Wu, W.; He, W.; Wei, X.; Jia, X.; Wu, X.; Xu, X.; Zu, X.; Zhou, X.; Pan, X.; Charles, Y.; Li, Y.; Hu, Y.; Liu, Y.; Chen, Y.; Wang, Y.; Liu, Y.; Qin, Y.; Liu, Y.; Yang, Y.; Bao, Y.; Du, Y.; Wu, Y.; Wang, Y.; Zhou, Z.; Wang, Z.; Li, Z.; Zhu, Z.; Zhang, Z.; Wang, Z.; Yang, Z.; Huang, Z.; Huang, Z.; Xu, Z.; Yang, Z.; and Lin, Z. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. arXiv:2501.12599.

Wang, F.; Ma, Y.; Guan, T.; Wang, Y.; and Chen, J. 2026. Autonomous Learning through Self-Driven Exploration and Knowledge Structuring for Open-World Intelligent Agents. In *2026 International Conference on Embedded Systems, Mobile Communication and Computing (EMC²)*, 301–306.

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023b. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171.

Wang, Z.; Wang, K.; Wang, Q.; Zhang, P.; Li, L.; Yang, Z.; Jin, X.; Yu, K.; Nguyen, M. N.; Liu, L.; Gottlieb, E.; Lu, Y.; Cho, K.; Wu, J.; Fei-Fei, L.; Wang, L.; Choi, Y.; and Li, M. 2025. RAGEN: Understanding Self-Evolution in LLM Agents via Multi-Turn Reinforcement Learning. arXiv:2504.20073.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yu, Q.; Zhang, Z.; Zhu, R.; Yuan, Y.; Zuo, X.; Yue, Y.; Dai, W.; Fan, T.; Liu, G.; Liu, L.; et al. 2026. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38: 113222–113244.

Zhang, Y.; and Math-AI, T. 2024. American Invitational Mathematics Examination (AIME) 2024.

Zhang, Y.; and Team, M.-A. 2025. American Invitational Mathematics Examination (AIME) 2025.

Zhao, A.; Huang, D.; Xu, Q.; Lin, M.; Liu, Y.-J.; and Huang, G. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19632–19642.

Zhao, S.; Xie, Z.; Liu, M.; Huang, J.; Pang, G.; Chen, F.; and Grover, A. 2026. Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models. arXiv:2601.18734.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P.; and Irving, G. 2020. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593.