# SAGE: Multi-Agent Self-Evolution for LLM Reasoning

Anonymous ACL submission

## Abstract

Reinforcement learning with verifiable rewards improves reasoning in large language models (LLMs), but many methods still rely on large human-labeled datasets. While self-play reduces this dependency, it often lacks explicit planning and strong quality control, limiting stability in long-horizon multi-step reasoning. We present **SAGE** (**S**elf-evolving **A**gents for **G**eneralized reasoning **E**volution), a closed-loop framework where four agents: *Challenger, Planner, Solver, and Critic*, co-evolve from a shared LLM backbone using only a small seed set. The Challenger continuously generates increasingly difficult tasks; the Planner converts each task into a structured multi-step plan; and the Solver follows the plan to produce an answer, whose correctness is determined by external verifiers. The Critic scores and filters both generated questions and plans to prevent curriculum drift and maintain training signal quality, enabling stable self-training. Across mathematics and code-generation benchmarks, SAGE delivers consistent gains across model scales, improving the Qwen-2.5-7B model by 8.9% on LiveCodeBench and 10.7% on OlympiadBench.

## 1 Introduction

Large language models (LLMs) have achieved remarkable advancements in reasoning tasks such as mathematics and coding through reinforcement learning (RL) techniques (Guo et al., 2025; Sheng et al., 2025; Sun et al., 2024). However, these methods often depend on large-scale human-curated datasets for verifiable rewards, posing scalability challenges and limiting autonomous adaptation as models approach superhuman capabilities (Zhao et al., 2025a; Chen et al., 2025).

Recent efforts have explored self-play and multi-agent frameworks to enable self-evolution without extensive external data. For instance, self-play paradigms like SPIRAL (Liu et al., 2025) and Absolute Zero (Zhao et al., 2025a) leverage verifiable environments for autonomous improvement, while multi-agent systems such as MARS (Yuan et al., 2025) and MAE (Chen et al., 2025) facilitate collaborative reasoning through role specialization. Despite these advances, existing approaches struggle with open-ended domains lacking robust verification and often fail to integrate planning for complex, multi-step tasks (Huang et al., 2025; Gao et al., 2025; Yue et al., 2025).

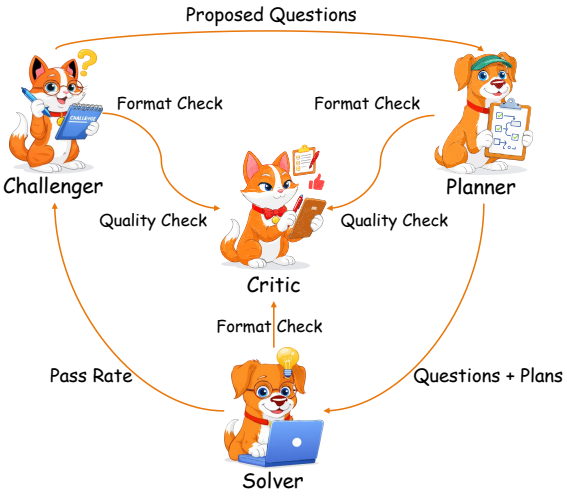


Figure 1: **Overview of the SAGE framework.** Four specialized agents—Challenger, Planner, Solver, and Critic—interact through quality filtering and format validation to enable closed-loop self-evolution.

To address these gaps, we propose **SAGE** (**S**elf-evolving **A**gents for **G**eneralized reasoning **E**volution), a closed-loop multi-agent framework that enables LLMs to co-evolve in verifiable domains like math and coding using only minimal seed examples. As illustrated in Figure 1, SAGE instantiates four specialized agents: a Challenger for task generation, a Planner for strategy outlining, a Solver for solution execution, and a Critic for quality assessment and format calibration. These agents interact adversarially, with the Challenger rewarded

for difficulty and the Solver optimized via verifier-based correctness, forming a self-rewarding cycle trained end-to-end using task-relative policy gradients.

Through experiments on mathematics and coding benchmarks, SAGE demonstrates significant performance gains, outperforming baselines trained on human-curated datasets in sample efficiency and generalization. We outline our contribution as follows:

- We design a scalable multi-agent framework for self-evolving LLMs in reasoning tasks.
- We propose a dual-role Critic mechanism ensuring task quality and solution verification.
- We conduct empirical evidence of effective co-evolution in math and code domains under few-example settings.

## 2 Related Work

**Reinforcement Learning for LLM Reasoning.** Early work applied RL (e.g., PPO (Schulman et al., 2017)) to language tasks, but recent research focuses on reinforcement learning with verifiable rewards (RLVR) for reasoning (Wan et al., 2025). For example, DeepSeek-R1 (Guo et al., 2025) shows that RLVR can extend an LLM's reasoning capabilities on math by training from correctness signals. WebAgent-R1 (Wei et al., 2025) is an end-to-end multi-turn RL framework that significantly boosts web navigation success using binary success rewards. Critic-free RL variants (e.g., GRPO (Guo et al., 2025)) reduce training overhead, but typically still rely on human-curated or grounded environments. Recent work has systematically characterized agentic RL for LLMs, emphasizing capabilities like planning and self-improvement (Zhang et al., 2025; Wen et al., 2025; Wu et al., 2025). In contrast, SAGE learns from self-generated, verifiable tasks with little external data.

**Multi-Agent LLM Systems.** LLM-based multi-agent frameworks facilitate complex tasks via role specialization. MetaGPT (Hong et al., 2024) encodes human-like workflows into a multi-agent assembly line, breaking down large tasks into subtasks among collaborating agents. CAMEL (Li et al., 2023) uses inception prompting to guide a society of role-playing agents, enabling study of cooperative behaviors in instruction-following tasks. MARS (Yuan et al., 2025) introduces a reinforcement learning framework where multi-agent self-play enhances strategic reasoning capabilities across cooperative and competitive tasks. These systems demonstrate that coordinating multiple LLM agents can enhance performance on complex tasks (Zhao et al., 2025b; Zhu et al., 2025). MARFT (Liao et al., 2025) applies multi-agent reinforcement fine-tuning to optimize LLM-based systems, and MALT (Motwani et al., 2025), which divides reasoning into generation, verification, and refinement steps using heterogeneous agents. SAGE extends this line by instantiating distinct agents (Challenger, Planner, Solver, Critic) within one LLM and jointly training them with shared feedback.

**Self-Play and Self-Evolving Agents.** Recent works explore self-play and self-evolution to improve LLMs autonomously. The SPIRAL (Liu et al., 2025) framework shows that self-play on zero-sum games can automatically induce generalizable reasoning strategies without human data. Absolute Zero (Zhao et al., 2025a) generates its own coding problems and uses a code executor as a verifier to self-critique and solve them, achieving strong math and coding reasoning without external data. Agentic Self-Learning (Sun et al., 2025) is a closed-loop framework unifying task generation, policy execution, and reward modelling for LLM agents in search environments. Additional approaches include AgentEvolver (Zhai et al., 2025) enables efficient self-evolving through curiosity-driven task generation and experience reuse, and Agent0 (Xia et al., 2025), which unleashes self-evolving agents via tool-integrated reasoning in a co-evolutionary curriculum-executor loop. While prior work has explored various components of self-evolving agents such as planning and task generation (Gao et al., 2025; Fang et al., 2025; Belle et al., 2025), SAGE is distinguished by integrating planning and critic roles to decompose reasoning and jointly train all agents for improved stability and depth in math and code domains.

## 3 Preliminaries

**Multi-Agent Reasoning in Verifiable Domains.** Let $\mathcal{M}_\theta$ denote an LLM parameterized by $\theta$. In *role-based multi-agent reasoning*, multiple agents share a backbone model. Still, they are conditioned on different role instructions (e.g., proposer, planner, solver, evaluator) to enhance robustness via

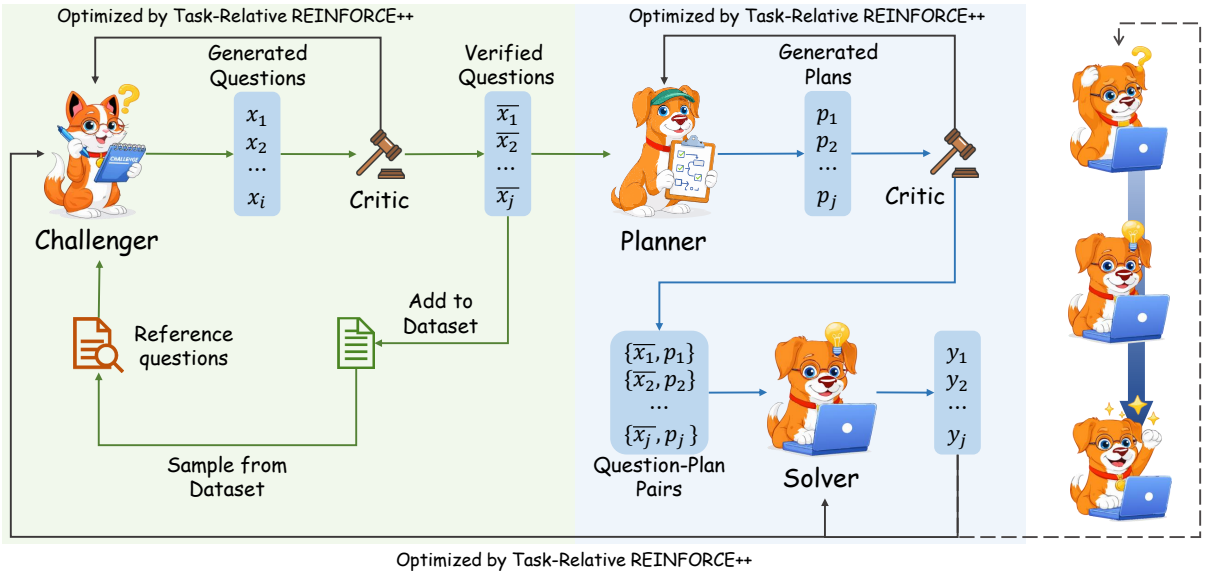


Figure 2: **The SAGE training pipeline.** (1) The Challenger generates questions from reference examples, filtered by the Critic for quality; (2) verified questions expand the dataset; (3) sampled questions are processed by the Planner and Solver to produce solutions; (4) all agents are jointly updated using Task-Relative REINFORCE++ with per-role advantage normalization.

collaboration and decomposition (Du et al., 2023; Liang et al., 2024). For a question $q$, agents produce structured answers $a$. In verifiable domains (mathematics, programming), a domain-specific verifier $V_{\text{gt}}(q, a, v) \in [0, 1]$ evaluates answer correctness given a reference $v$ (ground-truth or unit tests), enabling automatic reward computation without human annotation.

**Policy Gradient Optimization.** To enable self-evolution, we frame agent optimization as reinforcement learning, maximizing $J(\theta) = \mathbb{E}_{q\sim\mathcal{D}, o\sim\pi_\theta}[R(q, o)]$ where $\mathcal{D}$ is the task distribution, $R$ is the reward signal, and $o$ is the output. REINFORCE++ (Hu et al., 2025) is a critic-free method that computes the advantage as $A^t_{q,o} = r(q, o) - \beta_{\text{kl}} \sum_{i=t}^{T} \text{KL}(\pi_\theta \| \pi_{\text{ref}})_i$ with KL penalty to a reference policy, and applies global-batch normalization: $A^{\text{norm}} = (A - \mu_{\mathcal{B}})/(\sigma_{\mathcal{B}} + \epsilon)$. This stabilizes training and improves robustness across prompt distributions. To coordinate multiple agents with heterogeneous objectives, we adopt Task-Relative REINFORCE++ (Huang et al., 2025), which applies per-role advantage normalization:

$$A^{\text{role}}_{\text{norm}} = \frac{r - \mu_{\text{role}}}{\sigma_{\text{role}} + \epsilon}, \tag{1}$$

where $\mu_{\text{role}}$ and $\sigma_{\text{role}}$ are the mean and standard deviation computed over the corresponding role-specific batch.

## 4 The SAGE Framework

SAGE is a fully automated, self-iterative evolution framework requiring only a small seed set with automatic verification signals. SAGE instantiates four agents from a shared LLM backbone $\mathcal{M}_\theta$: (1) **Challenger** generates challenging tasks with verifiers; (2) **Planner** produces solution plans; (3) **Solver** outputs final answers; and (4) **Critic** evaluates quality and format compliance. These agents engage in continuous co-evolution, with the training workflow illustrated in Figure 2.

In verifiable domains such as mathematics and programming, SAGE forms a closed-loop pipeline (challenge–plan–solve–criticize) that combines multi-agent interactions with verifier-based reward signals. The Challenger and Solver co-evolve adversarially: the Solver is rewarded for verified correctness, while the Challenger receives difficulty rewards when the Solver fails under verification, pushing the curriculum toward harder yet still solvable tasks. Quality filtering and verifier validation are applied to prevent dataset degradation and improve training stability.

### 4.1 Reward Design and Normalization

**Format reward.** Across phases, SAGE applies a format reward $r_f \in [0, 1]$ to stabilize self-training by enforcing required tags (e.g., `<question>`, `<answer>`, `<type>`, `<score>`). In practice, $r_f$ is

a soft score (not strictly binary): missing tags yield low reward, redundant tags may receive partial credit, and empty outputs fall back to a neutral value (e.g., 0.5).

**Score normalization.** The Critic outputs scalar scores typically on a 1–10 scale inside `<score></score>`, which are normalized to $[0, 1]$ by

$$\mathrm{Norm}(s) = \begin{cases} s, & 0 \le s \le 1, \\ \frac{s-1}{9}, & 1 < s \le 10, \\ 0.5, & \text{otherwise.} \end{cases} \tag{2}$$

### 4.2 Challenger Agent Training

The Challenger proposes verifiable tasks to drive the Solver's learning. During training, the Challenger policy $\pi_c$ is prompted with reference problems sampled from a small human-curated seed set $\mathcal{D}$ (about 500 examples across datasets), where each seed item includes a problem statement and its verifier (ground-truth answer or executable tests). Given a reference item $(q_{\mathrm{ref}}, v_{\mathrm{ref}})$, the Challenger generates a new problem $q$ and an associated verifier $v$ in a constrained format:

$$(q, v) \sim \pi_c(\cdot \mid q_{\mathrm{ref}}, v_{\mathrm{ref}}; \theta), \tag{3}$$

where $\theta$ represents the shared LLM parameters.

**Composite reward.** The Challenger receives (i) a quality score $s_q \in [0, 1]$ from the Critic (clarity, relevance, well-formedness), (ii) a difficulty reward computed from the Solver's verified success rate, and (iii) a format reward. Concretely, we estimate the Solver success by sampling $N_s$ answers and verifying them with $V_{\mathrm{gt}}$:

$$\begin{aligned} a_j &\sim \pi_s(\cdot \mid q; \theta), \quad j = 1, \ldots, N_s, \\ \bar{s}_{\mathrm{gt}}(q, v) &= \frac{1}{N_s} \sum_{j=1}^{N_s} V_{\mathrm{gt}}(q, a_j, v), \\ r_d(q, v) &= 1 - \bar{s}_{\mathrm{gt}}(q, v). \end{aligned} \tag{4}$$

Here, $V_{\mathrm{gt}}(q, a, v) \in [0, 1]$ denotes the domain-specific verifier (e.g., exact-match/symbolic grading for math or test pass rate for code), $\pi_s$ denotes the Solver policy (formally introduced in Section 4.4).

The Challenger reward is computed as

$$r_c(q, v) = \tfrac{1}{3} s_q(q) + \tfrac{1}{3} r_d(q, v) + \tfrac{1}{3} r_f(o_c), \tag{5}$$

where $o_c$ (resp. $o_p$, $o_s$, $o_{cr}$) denotes the raw textual output of the Challenger (resp. Planner, Solver, Critic).

**Algorithm 1** Training Process of SAGE

**Require:** Base LLM $\pi_{\text{base}}$, iterations $T$, thresholds $\alpha, \beta$, sample size $N_s$

Init agents $\pi_c, \pi_p, \pi_s, \pi_{cr}$ from $\pi_{\text{base}}$
Init dataset $\mathcal{D} \leftarrow \mathcal{D}_0$ (each item has verifier)
**for** $t = 1$ to $T$ **do**
  Sample $(q_{\mathrm{ref}}, v_{\mathrm{ref}}) \sim \mathcal{D}$ ▷ **(1) Challenge Phase**
  $(q_t, v_t) \leftarrow \pi_c(\cdot \mid q_{\mathrm{ref}}, v_{\mathrm{ref}})$
  $s_q \leftarrow \mathrm{Norm}(\pi_{cr}(q_t))$; validate $v_t$
  Sample $a_j \sim \pi_s(\cdot \mid q_t)$ for $j = 1, \ldots, N_s$
  $\bar{s}_{\mathrm{gt}} \leftarrow \frac{1}{N_s} \sum_{j=1}^{N_s} V_{\mathrm{gt}}(q_t, a_j, v_t)$; $r_d \leftarrow 1 - \bar{s}_{\mathrm{gt}}$
  **if** $s_q \ge \alpha$ **and** $v_t$ valid **then**
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(q_t, v_t)\}$; $r_c \leftarrow \frac{1}{3} s_q + \frac{1}{3} r_d + \frac{1}{3} r_f(o_c)$
  **else**
    $r_c \leftarrow \frac{1}{2} s_q + \frac{1}{2} r_f(o_c)$
  **end if**
  ▷ **(2) Plan–Solve Phase**
  Sample $(q, v) \sim \mathcal{D}$; $p_t \leftarrow \pi_p(\cdot \mid q)$
  $s_p \leftarrow \mathrm{Norm}(\pi_{cr}(q, p_t))$
  **if** $s_p \ge \beta$ **then**
    $a_t \leftarrow \pi_s(\cdot \mid q, p_t; \theta)$; $\tilde{s}_p \leftarrow s_p$
  **else**
    $a_t \leftarrow \pi_s(\cdot \mid q, \emptyset; \theta)$; $\tilde{s}_p \leftarrow 0$
  **end if**
  $s_{\mathrm{gt}} \leftarrow V_{\mathrm{gt}}(q, a_t, v)$
  $r_p \leftarrow \lambda_{\mathrm{plan}} s_p + \lambda_f r_f(o_p)$;
  $r_s \leftarrow w_p \tilde{s}_p + w_c s_{\mathrm{gt}} + w_f r_f(o_s)$
  $r_{cr} \leftarrow r_f(o_{cr})$ ▷ **(3) Joint Update**
  Update $\pi_c, \pi_p, \pi_s, \pi_{cr}$ using $r_c, r_p, r_s, r_{cr}$
**end for**

**Quality filtering and difficulty suppression.** To prevent dataset degradation, we filter low-quality questions with a threshold $\alpha$ (in this paper, $\alpha = 0.7$), and also validate the generated verifier (e.g., parsable and executable for code tests). Only candidates that satisfy both criteria are added to $\mathcal{D}$. Moreover, for $s_q < \alpha$, we suppress the difficulty term to avoid rewarding "hard" but ill-posed tasks and use

$$r_c(q, v) = \tfrac{1}{2} s_q(q) + \tfrac{1}{2} r_f(o_c). \tag{6}$$

This stabilizes long-horizon self-training and mitigates reward collapse.

### 4.3 Planner Agent Training

The Planner $\pi_p$ generates a structured plan $p$ for a given question $q$, encapsulated in `<plan></plan>` tags. The Critic evaluates the plan quality to produce a normalized score $s_p \in [0, 1]$.

$$p \sim \pi_p(\cdot \mid q; \theta), \quad s_p = \mathrm{Norm}\big(\mathrm{Critic}(q, p)\big). \tag{7}$$

If $s_p$ meets a gating threshold (in this paper, $\beta = 0.3$), the plan is provided to the Solver; otherwise, the Solver answers directly.

For optimizing the Planner, we use a composite reward that combines plan quality and format

compliance:

$$r_p = \lambda_{\text{plan}}\, s_p + \lambda_f\, r_f(o_p), \tag{8}$$

where $\lambda_{\text{plan}}$ and $\lambda_f$ are weighting coefficients (we use $\lambda_{\text{plan}} = \lambda_f = 0.5$ by default).

### 4.4 Solver Agent Training

The Solver agent is tasked with generating final answers based on the given question $q$ and the plan $p$ (if the plan passes Critic gating). The Solver policy $\pi_s$ produces an answer $a$, typically wrapped in `<answer></answer>` tags or Markdown blocks:

$$a \sim \pi_s(\cdot \mid q, \tilde{p}; \theta), \quad \tilde{p} = \begin{cases} p, & s_p \geq \beta, \\ \emptyset, & s_p < \beta. \end{cases} \tag{9}$$

**Verifier-based composite reward (plan, correctness, format).** Solver correctness is computed by automatic verification in the target domain (symbolic/metric-based grading for math, or execution-/test-based validation for code), yielding $s_{\text{gt}} \in [0, 1]$. We combine plan quality, verified correctness, and format adherence as

$$\tilde{s}_p = \begin{cases} s_p, & s_p \geq \beta, \\ 0, & s_p < \beta, \end{cases} \tag{10}$$

$$r_s = w_p\, \tilde{s}_p + w_c\, s_{\text{gt}} + w_f\, r_f(o_s), \quad w_p + w_c + w_f = 1. \tag{11}$$

In this paper, we use $(w_p, w_c, w_f) = (0.2, 0.6, 0.2)$ as the default setting. If the plan score is unavailable (e.g., when the planning module is disabled), we fall back to a simpler mixture of verified correctness and format (e.g., $\frac{1}{2} s_{\text{gt}} + \frac{1}{2} r_f$) to maintain robustness.

In adversarial interaction with the Challenger, Solver failures under ground-truth verification contribute to the Challenger's difficulty reward (Eq. 5), forming a co-evolutionary loop that progressively pushes the curriculum toward harder yet solvable problems.

### 4.5 Critic: Scoring and Format Calibration

The Critic provides two types of signals: (1) soft format rewards $r_f \in [0, 1]$ by checking required tags, and (2) quality scores for Challenger questions ($s_q$) and Planner plans ($s_p$), normalized via Eq. 2. Importantly, in the verifiable setting, correctness is determined by the external verifier $V_{\text{gt}}$ rather than the Critic.

The Critic policy $\pi_{cr}$ outputs a scalar score deterministically:

$$s \sim \pi_{cr}(\cdot \mid x; \theta), \tag{12}$$

where $x \in \{(q, \cdot), (q, p)\}$ denotes the evaluation context (either a question alone or a question-plan pair). Optionally, we calibrate the Critic with a lightweight format-consistency objective

$$r_{cr} = r_f(o_{cr}), \tag{13}$$

which reduces parsing failures and improves stability of downstream reward computation.

### 4.6 Multi-Agent Co-Training

A training step in SAGE comprises: (1) **Challenger Phase** to generate verifiable candidate tasks and expand $\mathcal{D}$ with quality-and-verifier filtering; (2) **Plan–Solve Phase** where the Planner generates a single plan scored by the Critic and the Solver is optimized using the verifier-based reward in Eq. 10; (3) **Critic Phase** (optional) for format calibration; and (4) **Synchronized Update** that jointly updates the shared backbone using Task-Relative REINFORCE++ with per-role advantage normalization (see Section 3).

## 5 Experiments

### 5.1 Experimental Setup

**Training details** Our framework is implemented based on VeRL(Sheng et al., 2025), and we evaluate it using the Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen3-4B-Base models(Yang et al., 2025b,a). All agents are initialized from their corresponding base models. We apply LoRA (Hu et al., 2021) with rank 128 and a learning rate of 3e-6. Additional hyperparameter settings are provided in Table 4.

**Baseline Methods.** To comprehensively assess the effectiveness of the proposed SAGE framework, we conduct experiments on several representative foundation models, including Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen3-4B-Base. For each model, we report results for both the original checkpoint and the corresponding variant fine-tuned with SAGE. In addition, we include Absolute-Zero-Reasoning (AZR) (Zhao et al., 2025a) and Multi-Agent Evolve (MAE) (Chen et al., 2025) as alternative training baselines. Specifically, each model is trained for 200 steps under AZR. For MAE, we adopt the half-reference setting and train each model for 200 steps.

| Method | HEval+ | MBPP+ | LCB$^{v1-5}$ | GSM8K | Math | AI24 | AI25 | AMC | Olympiad | C Avg. | M Avg. | O Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Qwen-2.5-3B-Instruct** | | | | | | | | | | | | |
| Base Model | 68.3 | 60.6 | 12.0 | 84.6 | 60.4 | 3.3 | **6.7** | **40.0** | 28.0 | 46.9 | 37.2 | 40.4 |
| AZR | **68.9** | 61.4 | 15.0 | 81.2 | 62.4 | 3.3 | 3.3 | 35.0 | 28.9 | 48.4 | 35.7 | 39.9 |
| MAE | 68.3 | 61.1 | 15.9 | 82.2 | 65.8 | 3.3 | 3.3 | 32.5 | **32.5** | 48.4 | 36.6 | 40.5 |
| SAGE | **68.9** | **62.4** | **16.9** | **85.5** | **66.2** | **6.7** | **6.7** | 35.0 | 29.8 | **49.4** | **38.3** | **42.0** |
| **Qwen-2.5-7B-Instruct** | | | | | | | | | | | | |
| Base Model | 73.2 | 65.3 | 17.5 | 91.7 | 75.1 | **13.3** | 6.7 | **57.5** | 28.0 | 52.0 | 45.4 | 47.6 |
| AZR | 71.3 | **69.1** | 25.3 | **92.8** | **76.2** | 10.0 | **13.3** | 50.0 | 38.5 | 55.2 | 46.8 | 49.6 |
| MAE | **76.2** | 65.3 | 23.3 | 91.7 | **76.2** | **13.3** | **13.3** | 42.5 | 32.7 | 54.9 | 45.0 | 48.3 |
| SAGE | **76.2** | 64.0 | **26.4** | 92.2 | 74.7 | **13.3** | **13.3** | 52.5 | **38.7** | **55.5** | **47.5** | **50.1** |
| **Qwen-3-4B-Base** | | | | | | | | | | | | |
| Base Model | **76.8** | **65.3** | 21.5 | **94.5** | 87.0 | **16.7** | **13.3** | **77.5** | **49.0** | 54.5 | **56.3** | 55.7 |
| AZR | 74.4 | 65.0 | 26.1 | 89.3 | 76.2 | 10.0 | **13.3** | 50.0 | 41.5 | 55.2 | 46.7 | 49.5 |
| MAE | 76.2 | **65.3** | 24.2 | **94.5** | **92.0** | 13.3 | 10.0 | 70.0 | 43.7 | 55.2 | 53.9 | 54.4 |
| SAGE | 75.6 | 62.4 | **30.6** | 94.3 | 91.0 | **16.7** | 10.0 | 75.0 | 47.9 | **56.2** | 55.8 | **55.9** |

Table 1: **Main results on reasoning benchmarks.** Comparison of post-training methods across three model scales. We report pass@1 accuracy (%) on code generation (HumanEval+, MBPP+, LiveCodeBench) and mathematical reasoning (GSM8K, MATH, AIME 2024, AIME 2025, AMC, and OlympiadBench). C Avg., M Avg., and O Avg. denote the mean scores over code, math, and all benchmarks. SAGE achieves the best overall performance across all three model backbones. Bold indicates best per LLM backbone.

**Training and Evaluation Datasets.** Our training set comprises 500 instances sampled from MATH (Hendrycks et al., 2021a), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021), with detailed statistics in Appendix B. We evaluate on two domains: (1) *Mathematical Reasoning*: GSM8K and MATH (in-distribution, ID), along with four competition-level benchmarks—AIME'24, AIME'25, OlympiadBench (He et al., 2024), and AMC'23 (Hendrycks et al., 2021b)—as out-of-distribution (OOD) tests. (2) *Code Generation*: HumanEval+ and MBPP+ evaluated via Evalplus (Liu et al., 2023) (ID), and LiveCodeBench (Jain et al., 2024) v1–v5 (May 2023–February 2025) for OOD assessment. We report the accuracy (pass@1) based on greedy decoding across all benchmarks.

## 5.2 Main Results

Table 1 presents the performance of SAGE and baseline methods across code generation and mathematical reasoning benchmarks on three model backbones.

**Consistent Improvements Across Model Scales.** SAGE achieves the highest Overall Avg. on both Qwen-2.5-3B-Instruct (42.0%) and Qwen-2.5-7B-Instruct (50.1%), outperforming all baselines including AZR and MAE. On the 3B model, SAGE improves upon the base model by 1.6% overall, with notable gains on in-distribution benchmarks (GSM8K: 84.6% → 85.5%; MATH: 60.4% → 66.2%). Similarly, on the 7B model, SAGE yields a 2.5% improvement over the base model in Overall Avg., demonstrating consistent effectiveness across model scales.

| Backbone | Method | ID Avg. | OOD Avg. |
|---|---|---|---|
| Qwen-2.5-3B | Base Model | 68.4 | 18.0 |
| | AZR | 68.5 | 17.1 |
| | MAE | 69.4 | 17.5 |
| | SAGE | **70.8** | **19.0** |
| Qwen-2.5-7B | Base Model | 76.3 | 24.6 |
| | AZR | **77.4** | 27.4 |
| | MAE | 76.8 | 25.0 |
| | SAGE | **77.4** | **28.8** |
| Qwen-3-4B | Base Model | 80.9 | 35.6 |
| | AZR | 76.2 | 28.2 |
| | MAE | **82.0** | 32.2 |
| | SAGE | 80.8 | **36.0** |

Table 2: **ID and OOD generalization comparison.** SAGE consistently improves OOD performance (+4.2% on 7B) without sacrificing in-distribution accuracy.

**Strong Out-of-Distribution Generalization.** A key strength of SAGE lies in its generalization to out-of-distribution benchmarks. As shown in Table 2, SAGE achieves the best or near-best OOD Avg. across all three backbones (19.0%, 28.8%, and 36.0% respectively), while maintaining competitive ID Avg. scores. This balanced improvement is particularly evident on Qwen-2.5-7B, where SAGE improves OOD Avg. by 4.2%

| Method | HEval+ | MBPP+ | LCB$^{v1-5}$ | GSM8K | Math | AI24 | AI25 | AMC | Olympiad | C Avg. | M Avg. | O Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SAGE$^{\text{(full implementation)}}$ | **68.9** | 62.4 | **16.9** | 85.5 | **66.2** | **6.7** | **6.7** | 35.0 | **29.8** | **49.4** | **38.3** | **42.0** |
| SAGE$^{\text{(w/o challenger training)}}$ | 66.5 | 61.3 | 9.0 | **86.7** | 65.5 | 0.0 | 3.3 | 35.0 | 28.0 | 45.6 | 36.4 | 39.5 |
| SAGE$^{\text{(w/o solver training)}}$ | 67.7 | **64.3** | 9.0 | 81.2 | 60.4 | 3.3 | 0.0 | 30.0 | 28.0 | 47.0 | 33.8 | 38.2 |
| SAGE$^{\text{(w/o critic training)}}$ | 66.5 | 53.7 | 14.1 | 86.0 | 65.9 | 3.3 | **6.7** | **40.0** | 27.4 | 44.8 | 38.2 | 40.4 |

Table 3: **Ablation study of SAGE components on Qwen-2.5-3B.** We evaluate the impact of removing individual agent training while keeping other components active.

over the base model while preserving strong in-distribution performance. On LiveCodeBench specifically, SAGE achieves the best performance across all three backbones (16.9%, 26.4%, and 30.6%), substantially outperforming both base models and other post-training methods. For mathematical reasoning, SAGE maintains competitive performance on competition-level benchmarks such as OlympiadBench, where it achieves 38.7% (+10.7% over base) on Qwen-2.5-7B.

**Comparison with Baselines.** While AZR and MAE show improvements on certain individual benchmarks, they exhibit inconsistent gains and occasional performance degradation. For instance, AZR on Qwen-3-4B-Base leads to a significant drop in Math Avg. (56.3% $\rightarrow$ 46.7%). In contrast, SAGE maintains more balanced improvements across both domains without sacrificing performance on any benchmark group.

**Results on Qwen-3-4B.** On this stronger backbone, the base model already achieves high performance (Overall Avg. 55.7%). Nevertheless, SAGE attains the highest Code Avg. (56.2%) and remains competitive overall (55.9%), with particularly strong gains on LiveCodeBench (21.5% $\rightarrow$ 30.6%, +9.1%). This suggests that SAGE continues to provide meaningful improvements even when applied to capable base models.

### 5.3 Ablations Studies and Analyses

**Ablation Study.** To understand the contribution of each agent, we conduct ablation experiments by selectively disabling the training of individual roles while keeping the remaining components active. As shown in Table 3, the full SAGE implementation achieves the highest overall average (42.0%), and removing any single agent leads to performance degradation.

Disabling Challenger training results in a notable drop in code benchmarks, particularly on LiveCodeBench (16.9% $\rightarrow$ 9.0%), indicating that curriculum generation is essential for out-of-distribution generalization. Similarly, removing Solver training causes the largest overall decline (O Avg. 38.2%), with substantial drops on both GSM8K (85.5% $\rightarrow$ 81.2%) and MATH (66.2% $\rightarrow$ 60.4%), confirming that the Solver is the primary driver of reasoning capability. Interestingly, excluding Critic training yields competitive math performance (M Avg. 38.2%) but degrades code benchmarks (C Avg. 44.8%), suggesting that the Critic's quality filtering is more critical for code generation where output format and correctness are tightly coupled.

These results validate that all three trainable agents contribute complementarily to SAGE's overall effectiveness, with the Challenger–Solver interaction forming the core co-evolutionary loop and the Critic providing essential quality control.

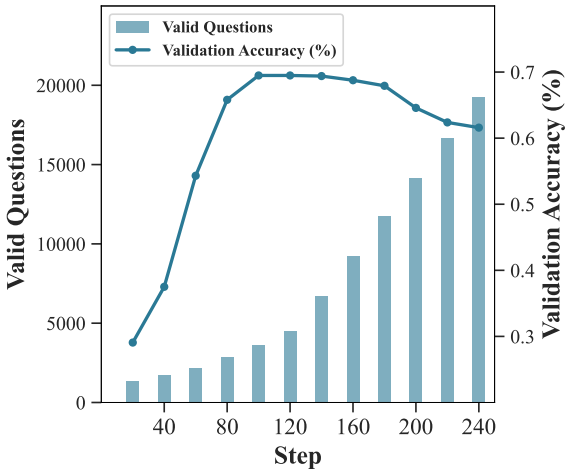


Figure 3: **Training dynamics on Qwen-2.5-3B.** The Challenger steadily expands the question pool (bars) throughout training, while validation accuracy (line) reaches peak performance around step 100–120 before gradual decline, suggesting potential over-specialization on the self-generated curriculum.

**Training Dynamics Analysis.** To gain deeper insights into the self-evolution process, we analyze the training dynamics of SAGE on Qwen-2.5-3B-Instruct, as shown in Figure 3.

The validation accuracy (line) exhibits a charac-

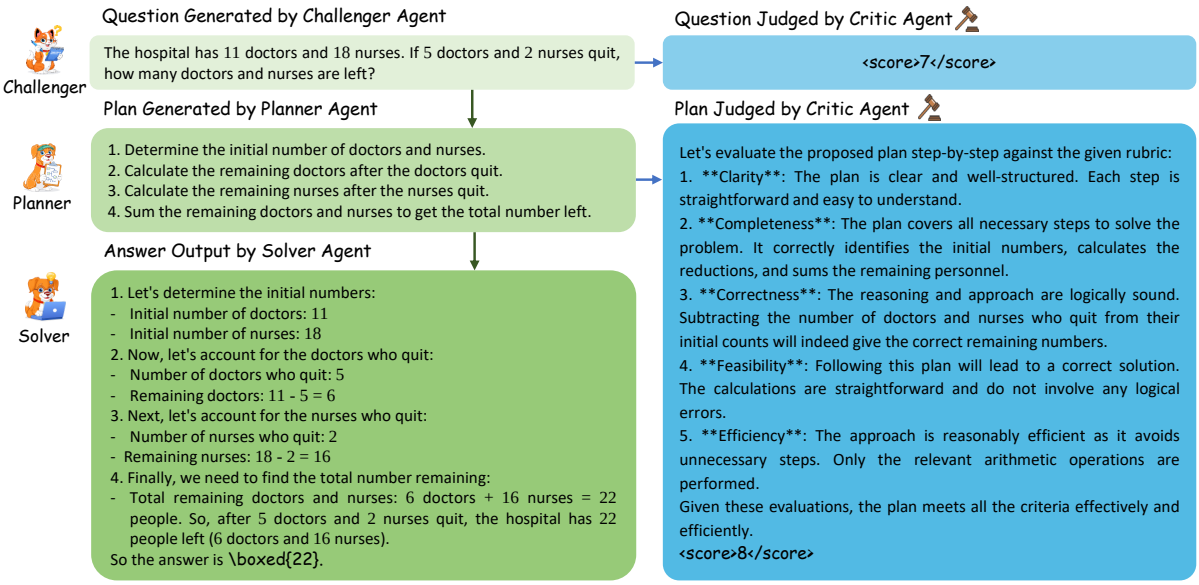


Figure 4: **Qualitative case study.** The Challenger generates a math word problem, the Planner decomposes it into structured steps, the Solver executes the plan to produce the final answer, and the Critic provides quality scores for both the question and the plan.

teristic learning curve. During the initial phase (steps 0–80), the model demonstrates rapid improvement from 29.1% to 65.8%, reflecting efficient knowledge acquisition from the multi-agent co-evolutionary training. The accuracy reaches its peak of 69.5% around step 100–140, representing the optimal balance between task difficulty and model capability. Beyond this point, we observe a gradual decline to 61.6% by step 240, suggesting that prolonged training may lead to over-specialization on the self-generated curriculum. This motivates our choice of reporting results around step 100 in the main experiments.

Meanwhile, the cumulative number of valid questions (bars) grows steadily throughout training, expanding from 1,136 to 20,532 by step 250, an 18-fold increase from the seed set. Notably, the growth rate accelerates around step 120–130, coinciding with peak validation accuracy, suggesting that a well-trained Challenger produces questions that pass the quality threshold $\alpha = 0.7$ at an increasing rate. The continued growth of the question pool despite declining accuracy after step 120 suggests that increased quantity alone does not ensure better performance, highlighting the importance of curriculum diversity and difficulty calibration. Nevertheless, this trend demonstrates SAGE's ability to autonomously scale its training data without human intervention.

**Qualitative Analysis.** Figure 4 illustrates the collaborative reasoning process of SAGE. The Challenger generates a well-formed arithmetic problem involving subtraction across two categories. The Planner decomposes this into four sequential steps, progressing from initial value identification to final summation. Guided by this structured plan, the Solver executes each step systematically and arrives at the correct answer. The Critic evaluates both outputs, assigning scores of 7 and 8 based on clarity, completeness, and logical soundness. This example highlights how role specialization enables effective division of labor: task generation, strategic planning, solution execution, and quality assessment operate as distinct yet coordinated functions within a unified training loop.

## 6 Conclusion

We introduce SAGE, a multi-agent self-evolution framework where four specialized agents: Challenger, Planner, Solver, and Critic, co-evolve through adversarial yet collaborative dynamics. Starting from minimal seed examples, SAGE autonomously expands its training curriculum while maintaining quality via critic-based filtering. Experiments demonstrate consistent improvements across model scales, with strong out-of-distribution generalization on competition-level benchmarks. These results highlight a scalable and effective pathway for evolving capable reasoning agents while reducing dependency on human-curated supervision.

## 7 Limitations

Among the limitations of our work, firstly, SAGE operates in verifiable domains where correctness can be automatically determined through ground-truth answers or executable tests. Extending the framework to open-ended tasks with subjective evaluation criteria, potentially through learned reward models, remains an interesting direction for future work. Secondly, although SAGE significantly reduces reliance on large-scale annotations, it still requires a small seed set (500 examples) to bootstrap the self-evolution process. Investigating strategies to further minimize seed requirements could broaden applicability to extremely low-resource scenarios. Thirdly, our evaluation focuses on mathematical reasoning and code generation benchmarks. Future exploration of other structured reasoning domains, such as logical reasoning or scientific problem solving, could offer valuable insights and validate the generalizability of our multi-agent architecture. Additionally, as with standard self-training approaches, monitoring training dynamics and applying early stopping is advisable to ensure optimal performance.

## References


Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program Synthesis with Large Language Models. *Preprint*, arXiv:2108.07732.

Nikolas Belle, Dakota Barnes, Alfonso Amayuelas, Ivan Bercovich, Xin Eric Wang, and William Wang. 2025. Agents of change: Self-evolving llm agents for strategic planning. *Preprint*, arXiv:2506.04651.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating Large Language Models Trained on Code. *Preprint*, arXiv:2107.03374.

Yixing Chen, Yiding Wang, Siqi Zhu, Haofei Yu, Tao Feng, Muhan Zhang, Mostofa Patwary, and Jiaxuan You. 2025. Multi-Agent Evolve: LLM Self-Improve through Co-evolution. *Preprint*, arXiv:2510.23595.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *Preprint*, arXiv:2110.14168.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. *Preprint*, arXiv:2305.14325.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. 2025. A Comprehensive Survey of Self-Evolving AI Agents: A New Paradigm Bridging Foundation Models and Lifelong Agentic Systems. *Preprint*, arXiv:2508.07407.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, and 8 others. 2025. A Survey of Self-Evolving Agents: On Path to Artificial Super Intelligence. *Preprint*, arXiv:2507.21046.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633–638.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring Massive Multitask Language Understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. *Preprint*, arXiv:2308.00352.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,

and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *Preprint*, arXiv:2106.09685.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. REINFORCE++: Stabilizing Critic-Free Policy Optimization. *Preprint*, arXiv:2501.03262.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025. R-zero: Self-evolving reasoning llm from zero data. *Preprint*, arXiv:2508.05004.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *Preprint*, arXiv:2403.07974.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pages 51991–52008. Curran Associates, Inc.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2024. Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18335–18345, Miami, Florida. Association for Computational Linguistics.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. 2025. MARFT: Multi-Agent Reinforcement Fine-Tuning. *Preprint*, arXiv:2504.16129.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. 2025. SPIRAL: Self-Play on Zero-Sum Games Incentivizes Reasoning via Multi-Agent Multi-Turn Reinforcement Learning. *Preprint*, arXiv:2506.24119.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Advances in Neural Information Processing Systems*, volume 36, pages 21558–21572. Curran Associates, Inc.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. 2025. MALT: Improving Reasoning with Multi-Agent LLM Training. *Preprint*, arXiv:2412.01928.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *Preprint*, arXiv:1707.06347.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297, Rotterdam, The Netherlands. ACM.

Chuanneng Sun, Songjun Huang, and Dario Pompili. 2024. Llm-based multi-agent reinforcement learning: Current and future directions. *Preprint*, arXiv:2405.11106.

Wangtao Sun, Xiang Cheng, Jialin Fan, Yao Xu, Xing Yu, Shizhu He, Jun Zhao, and Kang Liu. 2025. Towards Agentic Self-Learning LLMs in Search Environment. *Preprint*, arXiv:2510.14253.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. 2025. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *Preprint*, arXiv:2503.09501.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. 2025. WebAgent-R1: Training Web Agents via End-to-End Multi-Turn Reinforcement Learning. *Preprint*, arXiv:2505.16421.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. 2025. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *Preprint*, arXiv:2506.14245.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025. EvolveR: Self-Evolving LLM Agents through an Experience-Driven Lifecycle. *Preprint*, arXiv:2510.16079.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. 2025. Agent0: Unleashing Self-Evolving Agents from Zero Data via Tool-Integrated Reasoning. *Preprint*, arXiv:2511.16043.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 Technical Report. *Preprint*, arXiv:2412.15115.

Huining Yuan, Zelai Xu, Zheyue Tan, Xiangmin Yi, Mo Guang, Kaiwen Long, Haojia Hui, Boxun Li, Xinlei Chen, Bo Zhao, Xiao-Ping Zhang, Chao Yu, and Yu Wang. 2025. MARSHAL: Incentivizing Multi-Agent Reasoning via Self-Play with Strategic LLMs. *Preprint*, arXiv:2510.15414.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *Preprint*, arXiv:2504.13837.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. 2025. AgentEvolver: Towards Efficient Self-Evolving Agent System. *Preprint*, arXiv:2511.10395.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Yue Liao, Hongru Wang, Mengyue Yang, and 6 others. 2025. The Landscape of Agentic Reinforcement Learning for LLMs: A Survey. *Preprint*, arXiv:2509.02547.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. Absolute zero: Reinforced self-play reasoning with zero data. *Preprint*, arXiv:2505.03335.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. 2025b. Stronger-mas: Multi-agent reinforcement learning for collaborative llms. *Preprint*, arXiv:2510.11062.

Guobin Zhu, Rui Zhou, Wenkang Ji, and Shiyu Zhao. 2025. Lamarl: Llm-aided multi-agent reinforcement learning for cooperative policy generation. *Preprint*, arXiv:2506.01538.

# A Hyperparameter Settings

Table 4: Training Hyperparameters of our experiments.

| Hyperparameter | Value |
|---|---|
| **Training Configuration** | |
| Batch Size | 128 |
| Learning Rate | $3 \times 10^{-6}$ |
| Training Steps | 200 |
| **Generation Settings** | |
| Maximum Prompt Length | 8192 |
| Maximum Response Length | 8192 |
| Challenger Temperature | 0.6 |
| Planner Temperature | 0.6 |
| Solver Temperature | 0.6 |
| Critic Temperature | 0.1 |
| **Algorithm Settings** | |
| Learning Algorithm | Task-Relative REINFORCE++ |
| KL Regularization | Disabled |
| **LoRA Configuration** | |
| LoRA Rank | 128 |
| LoRA Alpha | 256 |
| LoRA Dropout | 0.95 |
| Target Modules | $q_{proj}$, $k_{proj}$, $v_{proj}$, $o_{proj}$, $gate_{proj}$, $up_{proj}$, $down_{proj}$ |

# B Training Data Composition

Table 5 presents the composition of the 500 training instances sampled from four benchmark datasets. These samples are drawn from the official training splits and serve as the foundation for our training procedure.

Table 5: Distribution of Training Samples Across Benchmarks

| Benchmark | Count |
|---|---|
| MATH (Hendrycks et al., 2021a) | 156 |
| GSM8K (Cobbe et al., 2021) | 148 |
| HumanEval (Chen et al., 2021) | 87 |
| MBPP (Austin et al., 2021) | 109 |
| **Total** | **500** |

## C Prompts for Agents

Here, we list the prompt of each agent as follows.

**Challenger Agent Prompt**

**Role:** Task Designer Agent

**Description:**

You are a task generation specialist. Your goal is to create a single, high-quality evaluation task that challenges complex reasoning abilities.

**Design Constraints:**

- Self-contained with clear problem statement
- Non-trivial: requires multiple reasoning steps or constraint satisfaction
- Deterministic or tightly bounded (avoid subjective judgment)
- Culturally neutral, no real-time data dependency
- Difficult but solvable

**Avoid:**

- Trivia or opinion-based prompts
- Ambiguous success criteria
- Web-dependent or time-sensitive content
- Unsolvable or ill-defined problems

**Respond using:**

<question>

[Your generated task here]

</question>

Figure 5: The prompt of the Challenger Agent.

**Planner Agent Prompt**

**Role:** Planner Agent

**Description:**

You will review the user problem and propose a concise plan that a solver can follow.

**Problem:**{question}

**Respond using:**

<plan>

1. ...

2. ...

</plan>

Figure 6: The prompt of the planner Agent.

**Solver Agent Prompt**

**Role:** Solver Agent

**Description:**

You will solve the problem by following the verified plan and prioritizing correct, well-reasoned content over formatting tricks.

**Input:**

- Problem: {question}

- Verified Plan: {plan}

**Instructions:**

- Explain the key reasoning steps clearly

- Follow the answer-format instruction in the problem statement exactly

- Do not introduce additional wrappers/tags unless explicitly required

Figure 7: The prompt of the Solver Agent.

**Critic Agent Prompt(Question)**

**Role:** Question Quality Critic Agent

**Description:**

You are an expert evaluator. Your task is to assess the quality of a generated question for reasoning benchmarks.

**Input:**

- Question to evaluate:{question}

Evaluate the question based on the following criteria:

<think>

**Evaluation Criteria:**

- Solvability: Is the question solvable with sufficient information? No internal contradictions?

- Logical Soundness: Is the question logically coherent and does not violate common sense?

- Clarity: Is the wording unambiguous with clear objectives and constraints?

- Appropriateness: Is it safe, relevant, and actually in the form of a question?

- Conciseness: Is it free from redundant repetition or unnecessary complexity?

**Scoring Guidelines:**

- 8-10: Excellent question - fully clear, logically sound, solvable, self-contained, and concise. Appropriate for evaluation purposes.

- 4-7: Acceptable question - has some ambiguity or missing details but no fatal flaws in solvability or logic.

- 1-3: Poor question - unsolvable, contradictory, violates common sense, unsafe, too open-ended, or not a valid question.

**Critical Rule:**

If any unsolvability or commonsense violation exists, score must be [1-3].

[Write your detailed analysis here, addressing each criterion.]

<\think>

Provide your final score.

**Important:**

Output only one tag like <score>7</score> (replace 7 with your integer score 1-10).

Figure 8: The prompt of the Critic Agent(question).

**Critic Agent Prompt(Plan)**

**Role:** Plan Critic Agent

**Description:**

You are an expert evaluator. Your task is to assess the quality of a proposed plan for solving a problem.

**Input:**

- Problem: {question}

- Proposed Plan: {plan}

Evaluate the plan based on the following criteria:

<think>

**Evaluation Criteria:**

- Clarity: Is the plan clear, structured, and easy to follow?

- Completeness: Does it cover all necessary steps to solve the problem?

- Correctness: Is the reasoning and approach logically sound?

- Feasibility: Can following this plan lead to a correct solution?

- Efficiency: Is the approach reasonably efficient, avoiding unnecessary steps?

**Scoring Guidelines:**

- 8-10: Excellent plan - clear, complete, logically sound, and feasible. Following it should lead to a correct solution.

- 4-7: Acceptable plan - has some gaps or minor issues but the general direction is correct.

- 1-3: Poor plan - unclear, incomplete, logically flawed, or unlikely to lead to a correct solution.

[Write your detailed analysis here, addressing each criterion.]

</think>

Provide your final score.

**Important:**

Output only one tag like <score>7</score> (replace 7 with your integer score 1-10).

Figure 9: The prompt of the Critic Agent(plan).

**Critic Agent Prompt(Answer)**

**Role:** Solution Quality Critic Agent

**Description:**

You are an expert evaluator. Your task is to assess the quality of a generated solution to a given question or problem.

**Input:**

- Question: {question}

- Generated Solution: {answer}

**Evaluate the solution based on the following criteria:**

<think>

**Evaluation Criteria:**

- Accuracy: Is the solution factually correct with no errors in reasoning, arithmetic, units, or assumptions?

- Completeness: Does it fully address the question with all necessary steps and derivations?

- Coherence: Is the reasoning logical and free from contradictions or hallucinations?

- Conciseness: Is the answer direct without meaningless repetition, rambling, or filler?

- Instruction Following: Does the solution follow any explicit formatting or structural requirements?

**Scoring Guidelines:**

- 8-10: Excellent solution - entirely correct, complete, logically sound, concise, and follows all instructions.

- 4-7: Acceptable solution - generally on-topic and partially correct, but has omissions or clarity issues (no factual errors).

- 1-3: Poor solution - contains any factual/logic/calculation error, hallucinated content, excessive repetition, or severe irrelevance.

**Critical Rules:**

- Any factual error (arithmetic, reasoning, common sense, units, invalid assumptions) → score must be [1-3]

- Hallucinated references, fabricated data, or unsupported claims → score must be [1-3]

- Meaningless repetition or excessive rambling → score must be [1-3]

[Write your detailed analysis here, addressing each criterion. If any critical issue exists, note that the score must be [1-3].]

<\think>

Provide your final score.

**Important:**

Output only one tag like <score>7</score> (replace 7 with your integer score 1-10).

Figure 10: The prompt of the Critic Agent(answer).