ZERO-L: TRI-ROLE CO-EVOLUTION FRAMEWORK FOR SELF-EVOLVING LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving large language models (LLMs) can improve through reinforcement learning on self-generated tasks and feedback, reducing reliance on human-annotated training data. However, existing self-evolution frameworks face a bottleneck of training gain stagnation after first few iterations. This is because of the reward accuracy declination in later iterations. We introduce Zero-L, a framework for long-horizon self-evolution of LLMs. Starting from a single base LLM, Zero-L initializes three independent roles - a **Questioner**, a **Solver**, and a **Thinker**. Questioner and Solver are optimized through reinforcement learning. Questioner is rewarded for providing tasks which challenge the edge of the solver's ability. After questioner update, solver answers the tasks that questioner generates and gets Solver Reward for solving tasks, but this reward is examined to be insustainable. And that's why **Thinker-Solver Reward** comes onboard, reasoning process of solver's answers is fed into thinker and better process gets a higher thinker-solver reward. At the same time, with high solver reward accuracy in early iters, thinker is trained to rate reasoning process that leads to right answers. But this indicates that the ability of thinker is weak in early iters, and that's why we introduce **Iteration Dependent Reward Policy (IDR Policy)** which mix solver reward and thinker reward through one math function which gives solver reward a higher weight in early iterations and a lower weight after accuracy declination. Empirical results show Zero-L outperforms strong baseline at +5.7 % in math reasoning benchmarks and +6.4% in general reasoning benchmarks. More importantly, the mean IDR reward for correct responses remains above 60% across all ten iterations.

1 INTRODUCTION

Self-evolving large language models (LLMs) have attracted growing attention as a means of improving model capabilities while reducing dependence on human supervision. Recent advances in self-play and self-rewarding learning demonstrate that models can improve through iterative training on their own outputs, reducing the need for additional human annotations (Chen et al., 2024; Yuan et al., 2024). This direction is particularly appealing as adapting LLMs to diverse tasks requires high-quality training data that can be costly to curate. More recent approaches extend this principle to autonomous task generation. A question-generating model and a problem-solving model interact to construct an adaptive curriculum. This process requires no externally supplied training questions or labels (Huang et al., 2026). Together, these developments point toward increasingly autonomous post-training, in which models participate in constructing the learning signals that drive their own improvement.

Despite this progress, sustaining improvement over successive rounds of self-evolution is an initial challenge. Empirical analyses of autonomous co-evolution show that performance can peak after only a few iterations and subsequently deteriorate (Huang et al., 2026). These results show that more rounds of self-evolution do not yield further improvements. This motivates a central objective: enabling self-evolving LLMs to sustain meaningful gains across extended training horizons while preserving their zero reliance on external supervision.

To address this challenge, we introduce Zero-L, a framework for long-horizon self-evolution of large language models. Zero-L initializes three distinct roles from the base LLM model: a Questioner, a

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

Solver, and a Thinker. At each iteration, the Questioner proposes questions, and the Solver generates multiple responses to each question. Majority voting over the boxed final answers provides the Solver reward. The remaining text in each response is treated as its solution process and is analyzed by the Thinker.

The Thinker maps the solution processes for the same question into a pairwise cosine-distance matrix. The mean distance of each process from the other processes is transformed into a bounded Thinker-Solver reward. The difference between the Solver reward and the Thinker-Solver reward provides the training signal for the Thinker, allowing its representation space to adapt across iterations without an external reward model. We combine these two rewards using IDR Policy. The first Solver update therefore uses only the Solver reward, while the contribution of the Thinker-Solver reward increases and take place of inaccurate solver rewards. The Questioner and Solver retain group-relative policy optimization, and all three roles maintain separate checkpoints without parameter fusion. Zero-L combines these components to support sustained improvement over extended co-evolutionary training.

We design experiments to examine whether the learned solution-process signal sustains Solver improvement across multiple self-evolution iterations.

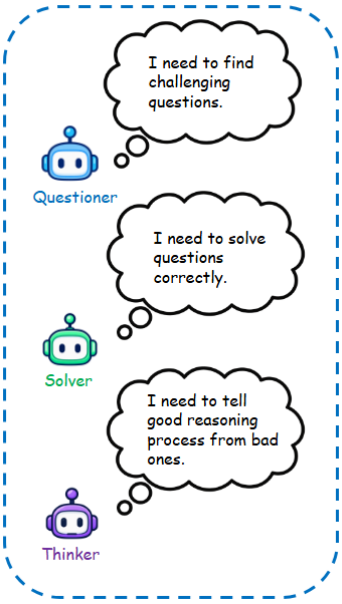

Figure 1: The three functional roles in Zero-L.

2 Motivation

The central learning mechanism in self-evolutionary co-training is the construction of pseudo-labels from Solver responses. For each generated question, final answers are grouped by mathematical equivalence, and the largest equivalence class is treated as the correct class for subsequent optimization. This mechanism enables training without human-annotated answers or external verifiers, but it depends on the majority response remaining a reliable proxy for correctness.

Empirically, the performance gains of self-evolving models often diminish or collapse after a limited number of outer iterations. This behavior is associated with a progressive decline in pseudo-label reliability (Huang et al., 2026). As the Questioner produces increasingly difficult questions, the Solver may no longer improve at the same rate, and its most frequent answer is therefore less likely to be correct. Once this occurs, erroneous pseudo-labels are reused as rewards and can amplify the degradation in subsequent iterations.

To reduce this dependence on late-stage majority voting, we introduce a third trainable role, the Thinker, initialized from the same base model as the Questioner and Solver. The Thinker uses the base model's decoder-only Transformer backbone to map each solution process into a contextual hidden representation. For the responses to the same question, pairwise cosine distances form a relation matrix, and the mean distance of each process from the remaining processes defines its relative isolation. During the early iterations, the reliable Solver reward calibrates this geometry: processes that agree with the pseudo-label are encouraged to receive high Thinker-Solver rewards, whereas inconsistent processes are encouraged to be represented as relatively isolated. The thinker is therefore not to reproduce the final answer, but to learn which structural patterns in a solution process are associated with reliable pseudo-label agreement during the early iterations.

This design induces a gradual transition in the signal used to optimize the Solver. At outer iteration n, let $w_n = e^{-nt}$ and define the IDR mixed reward as

$$R_i^{\text{mix}}(n) = w_n R_i^S + (1 - w_n)\, R_i^{TS}, \qquad w_n = e^{-nt}, \tag{1}$$

where R_i^S is the pseudo-label-based Solver reward and R_i^{TS} is the Thinker-Solver reward. At $n = 0$, $w_0 = 1$. The solver reward is most reliable in this early stage, so the Solver is updated entirely according to R_i^S, while the same reward supplies the calibration target used to obtain candidate

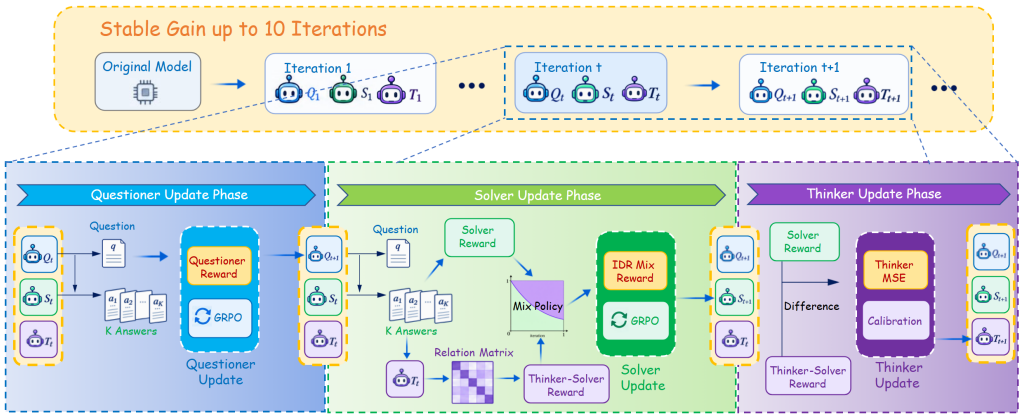

Figure 2: Overview of Zero-L. First of all, the Questioner generates a batch of candidate questions and the Solver samples multiple responses to each question. The boxed final answers are grouped by mathematical equivalence, and the largest class defines a pseudo-label. Then, the questioner is updated by the Questioner reward, and it produces a fresh batch of questions. The Solver samples new responses, and the Thinker computes a pairwise cosine-distance matrix over the solution processes. The mean distance of each process from the others is transformed into a bounded Thinker-Solver reward. The Solver is updated by a mixture policy IDR of the pseudo-label based Solver reward and the Thinker-Solver reward. At last, the Thinker is updated residually by the calibrastion through difference between the Thinker-Solver reward and the Solver reward.

Thinker parameters $\widetilde{\phi}_{n+1}$. As n increases and solver reward reliability declines, w_n decreases: the contribution of R_i^S is reduced, and the Solver is updated primarily according to the increasingly learned R_i^{TS}. To prevent unreliable late-stage solver reward from continuing to alter the Thinker at the same rate, we commit only a residual fraction of the candidate update, $\phi_{n+1} = \phi_n + \alpha w_n(\widetilde{\phi}_{n+1} - \phi_n)$. The hyperparameter $t \geq 0$ controls both the speed at which the Solver reward shifts from R_i^S to R_i^{TS} and the rate at which Thinker updates are attenuated; a larger t produces an earlier and faster transition, whereas $t = 0$ disables it.

3 Method

3.1 Problem Setup and Zero-Supervision Constraint

Let M_{θ_0} denote the initial base language model. Zero-L creates three functional roles from this single checkpoint: a Questioner, a Solver, and a Thinker. The Questioner produces training problems, the Solver produces candidate solutions, and the Thinker estimates response-level scores from relations among reasoning processes. The three roles are initialized independently from M_{θ_0} and then retain separate parameters, denoted by θ_n^Q, θ_n^S, and ϕ_n at the beginning of outer iteration n.

We use three reward terms throughout this section. The Questioner reward R_b^Q measures the learnability and non-redundancy of a candidate question. The Solver reward R_i^S is the binary agreement between response i and the majority-vote pseudo-label. The Thinker-Solver reward R_i^{TS} is derived from the relative geometry of response i in the Thinker's reasoning process representation space. Their round-dependent combination, which is used to optimize the Solver, is denoted by $R_i^{\text{mix}}(n)$.

No externally supplied question, solution, label, correctness verifier, teacher model, or benchmark score enters the optimization loop. Supervision is obtained from questions generated by the Questioner and pseudo-labels produced by majority voting over Solver responses. Mathematical equivalence is used only to group self-generated final answers; it does not provide an independent correctness judgment. External benchmarks are used exclusively for evaluation after training.

For a question q, the Solver samples a group of K responses,

$$\mathcal{A}(q) = \{a_1, \ldots, a_K\}, \qquad a_i \sim \pi_{\theta_n^S}(\cdot \mid q), \tag{2}$$

A group, rather than an individual response, is the basic unit of pseudo-label construction and Thinker scoring because both signals depend on relations among responses to the same question.

Each Solver response is decomposed into a reasoning process r_i and a final answer f_i. We define the final-answer validity indicator

$$z_i = \mathbb{I}[f_i \neq \emptyset] \tag{3}$$

and the set of responses usable by the Thinker as

$$\mathcal{V} = \{i : z_i = 1 \ \wedge \ r_i \neq \emptyset\}, \qquad m = |\mathcal{V}|. \tag{4}$$

Invalid responses are not replaced by rejection sampling during policy optimization.

3.2 Questioner and Solver Rewards[1]

For each response group, valid final answers are partitioned into equivalence classes. Let $\widetilde{f}(q)$ be the representative of the largest class, with ties resolved by the first occurrence in generation order. The corresponding consistency is

$$p(q) = \frac{1}{K} \sum_{i=1}^{K} z_i \mathbb{I}[f_i \equiv \widetilde{f}(q)]. \tag{5}$$

Malformed responses remain in the denominator and do not form an equivalence class.

During Questioner optimization, the current Solver generates K responses for each candidate question. To discourage repeated questions, we compute pairwise smoothed BLEU distances over the current reward batch and apply agglomerative clustering with a fixed distance threshold. For question q_b in a batch of B questions, let

$$\rho_b = \frac{|C(q_b)|}{B}, \tag{6}$$

where $C(q_b)$ denotes its cluster. The Questioner reward is

$$R_b^Q = \begin{cases} \max\left\{0, 1 - 2\left|p(q_b) - \frac{1}{2}\right| - \rho_b\right\}, & q_b \text{ passes the format checks,} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The first term favors questions of intermediate empirical difficulty, while ρ_b penalizes repeated questions. The proposed answer emitted by the Questioner is used only for format validation and is not treated as ground truth.

After the Questioner update, a fresh question pool is generated. The current Solver samples $K = 10$ responses for each question, and questions with majority counts from three through seven are retained for Solver training. The winning class defines a fixed curriculum pseudo-label $\widetilde{f}(q)$. For every fresh Solver rollout used during policy optimization, the Solver reward is

$$R_i^S = z_i \mathbb{I}[f_i \equiv \widetilde{f}(q)] \in \{0, 1\}. \tag{8}$$

The pseudo-label is not recomputed from the fresh optimization rollout.

3.3 Thinker and Thinker-Solver Reward

The Thinker represents only the reasoning process and does not encode the boxed final answer. For each r_i, we form the contextual sequence

$$x_i = \texttt{Question: } q \ \texttt{Reasoning: } r_i \tag{9}$$

and pass it through the decoder-only Transformer backbone T_{ϕ_n}. Let

$$H_{\phi_n}(q, r_i) = T_{\phi_n}(x_i) \in \mathbb{R}^{L_i \times d} \tag{10}$$

[1]The Questioner and Solver reward formulations follow R-Zero (Huang et al., 2026).

denote the contextual token representations after the final decoder block and the model's final normalization, where L_i is the encoded sequence length and d is the hidden dimension. Let $\mathcal{S}_i$ be the positions belonging to r_i. The response representation is the masked mean over only the reasoning process tokens, followed by ℓ_2 normalization:

$$h_i = \frac{1}{|\mathcal{S}_i|} \sum_{\ell \in \mathcal{S}_i} H_{\phi_n,\ell}(q, r_i), \qquad u_i = \frac{h_i}{\|h_i\|_2}. \tag{11}$$

The language-model output head is not used, and no projection, classification, or reward head is introduced. Long sequences are truncated to a fixed context length while retaining tokens from both the beginning and the end of the solution process.

During a Solver optimization stage, all groups are scored by a fixed snapshot of T_{ϕ_n}. The Thinker parameters are updated only after that Solver stage has completed. Thus, the reward function remains stationary within a Solver stage, although the Thinker itself is trainable across outer iterations.

For the m usable solution processes, Zero-L constructs a pairwise cosine-distance matrix

$$D^R_{ij} = \frac{1}{2}\left(1 - u_i^\top u_j\right), \qquad i, j \in \mathcal{V}. \tag{12}$$

For each usable response, we jointly define its mean distance to the remaining solution processes and the corresponding Thinker-Solver reward as

$$\delta^R_i = \frac{1}{m-1} \sum_{\substack{j \in \mathcal{V} \\ j \neq i}} D^R_{ij}, \qquad R^{TS}_i = 1 - \tanh\left(\frac{\delta^R_i}{\tau_d}\right), \quad i \in \mathcal{V}, \tag{13}$$

where δ^R_i measures the relative isolation of response i and $\tau_d > 0$ controls sensitivity to pairwise distance.

3.4 Iteration-Dependent Solver-Training Reward (IDR)

The contribution of the Thinker-Solver reward increases across outer iterations, while solver reward weight should decrease because of the inaccuracy of pseudo labels in later iterations. Let $t \geq 0$ denote the schedule temperature and define

$$w_n = \exp(-nt). \tag{14}$$

The mixed reward used to optimize the Solver is

$$R^{\text{mix}}_i(n) = w_n R^S_i + (1 - w_n) R^{TS}_i. \tag{15}$$

Both terms lie in $[0, 1]$, and therefore $R^{\text{mix}}_i(n) \in [0, 1]$. At the first iteration, $n = 0$ and $w_0 = 1$, so the Solver is optimized entirely by R^S_i and is independent of the untrained Thinker. For $t > 0$, the relative contribution of R^{TS}_i increases monotonically with n.

3.5 Model Update

Having defined the Questioner reward R^Q_b and the mixed Solver-training reward $R^{\text{mix}}_i(n)$, we use group relative policy optimization (GRPO) to update the Questioner and Solver, respectively (Shao et al., 2024).

The Thinker is trained to associate the geometry of a reasoning process with reliable pseudo-label agreement during the early iterations. The Solver reward R^S_i is treated as a fixed calibration target, and we define

$$e_i = R^{TS}_i - \text{stopgrad}(R^S_i). \tag{16}$$

Since $R^{TS}_i, R^S_i \in [0, 1]$, the squared calibration error lies in $[0, 1]$. The Thinker parameters are optimized by minimizing

$$\mathcal{L}_T(\phi) = \frac{1}{m} \sum_{i \in \mathcal{V}} \left(R^{TS}_i - \text{stopgrad}(R^S_i)\right)^2. \tag{17}$$

Gradients propagate through the $tanh$ transformation, row means, cosine-distance matrix, masked pooling, and Transformer backbone. They do not propagate into R_i^S or the Solver trajectories.

We use the same coefficient of IDR to attenuate the committed Thinker update. Let $\widetilde{\phi}_{n+1}$ denote the candidate parameters obtained by optimizing the Thinker at outer iteration n. The checkpoint carried into the next iteration is

$$\phi_{n+1} = \phi_n + \alpha w_n \left(\widetilde{\phi}_{n+1} - \phi_n \right), \tag{18}$$

Consequently, $w_0 = 1$, whereas later rounds retain progressively smaller fractions of the candidate parameter change. Late-stage Solver optimization therefore relies increasingly on R_i^{TS}, while potentially unreliable late-stage pseudo-labels induce progressively smaller changes to the Thinker parameters. α is a hyperparameter that controls the extent to which the Thinker parameters are updated. Responses without a valid boxed final answer receive zero reward regardless of the mixture weight.

3.6 One Outer Iteration

Starting from the three checkpoints $(\theta_n^Q, \theta_n^S, \phi_n)$, one complete Zero-L iteration is implemented as follows.

1. **Questioner inner update.** The Questioner generates candidate questions, and the current Solver samples K responses to each question. Equation 7 supplies one reward per candidate question. GRPO produces the updated checkpoint θ_{n+1}^Q.
2. **Self-generated curriculum.** A fresh pool of questions is sampled from θ_{n+1}^Q. The current Solver generates $K = 10$ responses per question. Majority voting supplies a pseudo-label, and only questions with majority counts from three through seven are retained.
3. **Solver inner update.** The Solver samples a fresh group of K responses for each retained question. The fixed pseudo-label gives R_i^S, while the round-start Thinker T_{ϕ_n} gives R_i^{TS} from the solution-process matrix. Equation 15 supplies one mixed reward per response, and GRPO produces θ_{n+1}^S.
4. **Thinker update.** The response groups, Solver rewards, and solution-process relations collected during Solver optimization are used to minimize Equation 17, producing the candidate checkpoint $\widetilde{\phi}_{n+1}$. Equation 18 then produces the updated Thinker checkpoint ϕ_{n+1}.

The three updated checkpoints are carried independently into iteration $n + 1$. Persistent rollout identifiers are used to reconstruct response groups after token-length balancing; the implementation does not infer group membership from adjacent batch positions.

4 Main Experiments

4.1 Models and training details

We employ Qwen3-4B-Base and Qwen3-8B-Base (Yang et al., 2025) to assess the impact of scale within a single architectural family. In every outer iteration, we apply five Questioner-update steps and 15 Solver-update steps. For a fair comparison, the R-Zero baseline uses the same five Challenger-update and 15 Solver-update steps (Huang et al., 2026). The Questioner generates a candidate pool of $N = 8000$ questions. After format filtering, the current Solver samples $m = 10$ answers per question to update the Questioner. A question is retained for Solver training only when between three and seven of these answers match the majority-vote pseudo-label. The Solver then generates $m = 10$ fresh answers for every retained question. The Thinker assigns a Thinker-Solver reward to each answer with $\tau_d = 0.1$. The IDR schedule uses $t = 0.5$ for the 4B model and $t = 0.4$ for the 8B model, and the residual Thinker update uses $\alpha = 0.7$. Additional hyperparameters and implementation details are provided in Appendix A.

To show the effectiveness of our proposed Zero-L framework, we compare it with R-Zero on mathematical- and general-reasoning benchmarks after $N = 3, 6, 9, 12$ iterations of training. The evaluation code is adopted from General-Reasoner (Ma et al., 2025). All evaluations use the Solver

checkpoints from the corresponding iteration of each framework. All experiments are conducted on 4 NVIDIA RTX PRO 6000 Blackwell GPUs. Results are reported as the average of three independent runs with different random seeds.

4.2 RESULTS

4.2.1 MATH REASONING

Table 1: Mathematical reasoning performance across self-evolution iterations.

Model Name		Qwen3-4B-Base				Qwen3-8B-Base			
		Avg.	**GSM8K**	**AMC**	**AIME25**	**Avg.**	**GSM8K**	**AMC**	**AIME25**
	Base-Model	47.34	86.71	45.01	10.30	50.90	88.92	52.12	11.67
iter-3	R-Zero	**52.87**	**91.59**	**57.42**	9.61	**56.59**	**93.78**	**61.60**	**14.39**
	Zero-L	52.84	90.12	57.17	**11.23**	55.17	91.76	59.28	14.46
iter-6	R-Zero	52.15	90.98	55.24	10.23	56.72	**94.28**	61.63	14.26
	Zero-L	**53.42**	**91.34**	**58.24**	**10.67**	**56.73**	93.98	61.63	**14.57**
iter-9	R-Zero	51.48	88.59	56.42	9.42	55.51	**94.42**	60.72	11.39
	Zero-L	**54.85**	**92.96**	**58.24**	**13.35**	**56.90**	94.23	**62.24**	**14.22**
iter-12	R-Zero	51.72	88.42	56.14	10.61	55.16	93.02	59.12	13.34
	Zero-L	**54.66**	**92.67**	**59.23**	**12.09**	**58.14**	**94.87**	**63.32**	**16.24**

As data shown in Table 1, on Qwen3-4B-Base, the performance of the solver in R-Zero framework reaches its peak after 3 iterations of training and then starts to decline. In contrast, the solver in Zero-L framework continues to improve and reaches its peak after 9 iterations of training. On Qwen3-8B-Base, R-Zero peaks after 6 iterations, while the performance of the solver in Zero-L framework continues to improve and reaches its peak after 12 iterations of training. In the end , the solver in Zero-L framework outperforms the solver in R-Zero framework by 2.94% on Qwen3-4B-Base and 2.98% on Qwen3-8B-Base after 12 iterations of training. And the peak performance of the solver in Zero-L framework is also higher than that in R-Zero framework. The results show that our proposed Zero-L framework can delay the performance stagnation and achieve better performance than R-Zero framework.

4.2.2 GENERAL REASONING

Table 2: General reasoning performance across self-evolution iterations.

Model Name		Qwen3-4B-Base				Qwen3-8B-Base			
		Avg.	**MMLUpro**	**Super-GP**	**BBEH**	**Avg.**	**MMLUpro**	**Super-GP**	**BBEH**
	Base-Model	26.09	50.12	20.14	8.02	32.04	58.87	28.36	8.89
iter-3	R-Zero	**30.98**	**55.24**	**27.25**	**10.46**	**34.89**	**61.42**	**31.28**	**11.96**
	Zero-L	30.05	54.24	26.98	8.92	33.04	59.97	28.57	10.59
iter-6	R-Zero	29.98	54.97	25.73	9.23	**34.86**	**62.25**	**32.22**	10.12
	Zero-L	**31.67**	**57.87**	**27.89**	**9.26**	34.46	61.34	30.92	**11.12**
iter-9	R-Zero	30.93	56.18	26.45	**10.17**	34.81	61.18	30.97	**12.28**
	Zero-L	**32.83**	**59.23**	**29.28**	9.97	**35.38**	**62.89**	**32.24**	11.02
iter-12	R-Zero	31.48	57.26	27.27	9.92	34.03	60.23	32.93	8.92
	Zero-L	**33.31**	**59.57**	**29.14**	**11.23**	**36.22**	**63.34**	**33.12**	**12.19**

As shown in Table 2, Zero-L exhibits sustained improvement in average general-reasoning performance across the evaluated iterations for both model scales. On Qwen3-4B-Base, the average score

of R-Zero fluctuates within a narrow range and reaches 31.48 at iteration 12, whereas Zero-L increases steadily from 30.05 at iteration 3 to 33.31 at iteration 12. On Qwen3-8B-Base, R-Zero peaks at 34.89 after three iterations and subsequently declines to 34.03, while Zero-L continues to improve and reaches 36.22 after 12 iterations. At iteration 12, Zero-L outperforms R-Zero by 1.83 percentage points on Qwen3-4B-Base and 2.19 percentage points on Qwen3-8B-Base. Moreover, the peak performance of Zero-L exceeds the corresponding R-Zero peak by 1.83 and 1.33 percentage points, respectively. These results provide further evidence that Zero-L mitigates performance stagnation and sustains general-reasoning improvements over longer self-evolution horizons.

5 More Experiments and Analysis

5.1 Reward Accuracy Experiment and Analysis

To examine how reward reliability changes during sustained learning, we sample $M = 500$ question–answer groups from the Solver training set. An external GPT-5.5 teacher model (OpenAI, 2026) judges the correctness of each answer for analysis only; these judgments never enter the Zero-L training loop. For each internal reward signal, we report the mean rewards assigned to answers that the teacher judges correct and incorrect. Additional experimental details are provided in Appendix A.4.

Table 3: Mean rewards assigned to teacher-judged correct and incorrect responses by the three reward signals, together with the Solver reward weight, for Qwen3-4B-Base with $t = 0.5$.

Mean reward		**iter1**	**iter2**	**iter3**	**iter4**	**iter5**	**iter6**	**iter7**	**iter8**	**iter9**	**iter10**
Correct	Solver reward (R-Zero)	72.4	57.9	50.7	47.4	48.8	46.9	48.7	47.4	47.8	46.6
	Thinker-Solver reward	49.7	64.2	67.6	71.8	70.4	68.6	66.2	68.7	66.9	69.4
	IDR mix (ours)	72.4	60.4	61.4	66.4	67.5	66.8	65.3	68.1	66.6	69.1
Incorrect	Solver reward (R-Zero)	25.7	40.6	44.8	43.6	46.4	47.2	46.8	47.6	48.8	47.2
	Thinker-Solver reward	47.6	32.2	31.7	26.8	27.5	28.0	28.9	26.3	27.0	29.2
	IDR mix (ours)	25.7	37.3	36.5	30.5	30.1	29.6	29.8	26.9	27.4	29.4
Solver reward weight		1.000	0.607	0.368	0.223	0.135	0.082	0.049	0.030	0.018	0.011

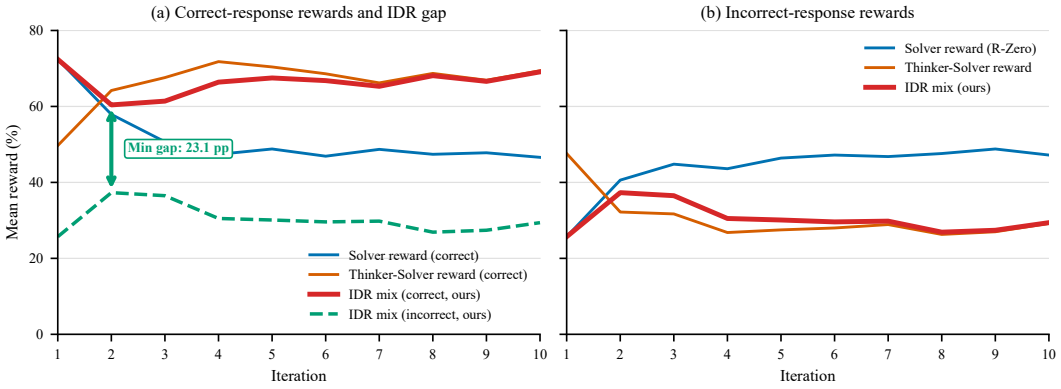

Figure 3: Reward dynamics for Qwen3-4B-Base with $t = 0.5$. Left: mean rewards assigned to teacher-judged correct responses by the Solver, Thinker-Solver, and IDR signals; the dashed green curve shows the IDR reward on incorrect responses, and the arrow marks the minimum correct–incorrect IDR gap. Right: mean rewards assigned to teacher-judged incorrect responses by the three signals.

Table 3 and Figure 3 show that the mean Solver reward assigned to correct responses falls below 50% after iteration 3, whereas the Thinker-Solver signal remains stronger in later rounds. As the IDR schedule shifts weight toward this signal, the mean IDR reward on correct responses remains above 60% from iteration 2 onward. IDR also assigns a lower mean reward to incorrect responses

in every round. The smallest correct–incorrect separation occurs at iteration 2, where the two IDR means are 60.4% and 37.3%, respectively, yielding the 23.1-point gap annotated in Figure 3. These results indicate that IDR preserves a positive reward margin while the pseudo-label-based Solver reward becomes less reliable.

5.2 Ablation Study

Table 4 isolates the adaptive Questioner and IDR on Qwen3-4B-Base after 12 iterations. Freezing the Questioner lowers the Math and General averages by 2.49 and 3.18 points, confirming the value of adapting the curriculum to the evolving Solver. Replacing IDR with $w_n = 0.5$ lowers them by 4.04 and 2.97 points. Thus, a fixed mixture cannot track changing reward reliability, and both components are important for sustained improvement.

Table 4: Ablation results on Qwen3-4B-Base after 12 iterations. **Questioner Frozen**: the Questioner is not updated. **w/o IDR**: the reward-mixing weight is fixed at $w_n = 0.5$.

Method	Math Avg.	General Avg.
Zero-L (Full)	**54.66**	**33.31**
Ablations		
Questioner Frozen	52.17	30.13
w/o IDR ($w_n = 0.5$)	50.62	30.34

6 Related Work

6.1 Self-Evolving LLMs and Autonomous Curriculum Generation

Self-improvement reuses model outputs as training signals. SPIN competes with previous policy iterations but retains human demonstrations as its target distribution (Chen et al., 2024); Self-Rewarding Language Models instead generate and judge their own responses (Yuan et al., 2024). Autonomous curricula also evolve the tasks: Absolute Zero uses a code executor to validate self-proposed tasks and answers (Zhao et al., 2025), whereas R-Zero trains separate Challenger and Solver models using agreement-based pseudo-labels (Huang et al., 2026). These methods reduce external data but remain vulnerable to unreliable self-generated rewards. Zero-L addresses this long-horizon bottleneck with an independent process-aware Thinker and IDR, which gradually shifts Solver training away from degrading pseudo-label rewards.

6.2 Process-Level Reward Modeling and Self-Supervised Verification

Process rewards provide denser reasoning feedback than outcome rewards, but human step labels are costly (Lightman et al., 2024). Math-Shepherd constructs step supervision automatically (Wang et al., 2024a); Rewarding Progress uses a separate prover to estimate step-wise progress (Setlur et al., 2025); and process-based self-rewarding uses an LLM judge (Zhang et al., 2025). These approaches still depend on human or answer-based labels, a prover, or explicit judge decisions. Zero-L instead derives a response-level process reward from the relative geometry of reasoning representations, without a reward head or external verifier. Early pseudo-labels calibrate the Thinker, while IDR increases its contribution as pseudo-label accuracy declines, enabling zero-supervision feedback to remain useful across iterations.

6.3 R-Zero: Self-Evolving Reasoning LLM from Zero Data

R-Zero initializes independent Challenger and Solver roles from one base model and co-evolves them without external tasks or labels (Huang et al., 2026). The Challenger targets the Solver's capability boundary, creating an adaptive self-generated curriculum. However, Solver training relies only on agreement-based pseudo-labels, whose reliability declines over iterations. Zero-L extends this framework with a Thinker and IDR to reduce that dependence and sustain longer-term improvement.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and information retrieval and discovery. We did not use generative AI tools for research ideation, substantive drafting, synthetic-data generation, or the derivation or proof of mathematical claims. We reviewed all AI-assisted outputs, checked retrieved information for errors, and verified the paper's structure and content. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the assistance of generative AI.

REFERENCES

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.01335.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, Kang Zhu, Minghao Liu, et al. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025. URL https://arxiv.org/abs/2502.14739.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=oSGOE5z2iU.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K. Jain, Virginia Aglietti, Disha Jindal, Peter Chen, et al. BIG-Bench Extra Hard. *arXiv preprint arXiv:2502.19187*, 2025. URL https://arxiv.org/abs/2502.19187.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing LLM reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025. URL https://arxiv.org/abs/2505.14652.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026. Accessed September 25, 2026.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024a. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024b. URL https://arxiv.org/abs/2406.01574.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2401.10020.

Shimao Zhang, Xiao Liu, Xin Zhang, Junxiao Liu, Zheheng Luo, Shujian Huang, and Yeyun Gong. Process-based self-rewarding language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18097–18110, 2025. doi: 10.18653/v1/2025.findings-acl.930. URL https://aclanthology.org/2025.findings-acl.930/.

Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/9837dc00ff67d176373268ed48042d49-Abstract-Conference.html.

APPENDIX

A EXPERIMENT DETAILS

A.1 TRAINING HYPERPARAMETERS

A.1.1 SOLVER TRAINING

- **Global Batch Size**: 128
- **Learning Rate**: 1×10^{-6}
- **Weight Decay**: 1×10^{-2}
- **KL Penalty Coefficient** (λ_{KL}): 1×10^{-2}
- **Maximum Update Steps per Outer Iteration**: 15
- **Responses per Question**: 10
- **Rollout Temperature**: 1.0
- **Rollout Top-p**: 0.99

A.1.2 QUESTIONER TRAINING

- **Global Batch Size**: 128
- **Learning Rate**: 1×10^{-6}
- **Weight Decay**: 1×10^{-2}
- **KL Penalty Coefficient** (λ_{KL}): 1×10^{-2}
- **Maximum Update Steps per Outer Iteration**: 5
- **Question Rollouts per Prompt**: 4
- **Rollout Temperature**: 1.0
- **Rollout Top-p**: 0.99

A.1.3 ZERO-L-SPECIFIC SETTINGS

- **Candidate Questions per Outer Iteration**: $N = 8000$
- **Answers for Pseudo-Label Construction**: $m = 10$ per question
- **Curriculum Filter**: retain majority counts from 3 through 7
- **Thinker Distance Temperature**: $\tau_d = 0.1$
- **IDR Schedule Parameter**: $t = 0.5$ (4B) and $t = 0.4$ (8B)
- **Residual Thinker-Update Scale**: $\alpha = 0.7$
- **Reported Outer Iterations**: $3, 6, 9, 12$

The Thinker is held fixed while a Solver stage is being optimized and is updated only after that stage. Its language-model output head is unused; the reward is computed from normalized mean-pooled final-layer representations of the reasoning-process tokens, as described in Section 3.

A.2 PROMPT TEMPLATES

The Solver and Questioner use the R-Zero prompt templates, with "Challenger" renamed to "Questioner" to match the role terminology of Zero-L. The benchmark answer-checking prompt is likewise adopted from R-Zero (Huang et al., 2026).

Solver Prompt Template

System Message:
Please reason step by step, and put your final answer within `\boxed{}`.

User Message:
{*problem_statement*}

Note: {`problem_statement`} *is a placeholder for the actual math problem.*

Questioner Prompt Template

System Message:
You are an expert competition-math problem setter. FIRST, in your private scratch-pad, think step by step to design a brand-new, non-trivial problem. The problem could come from any field of mathematics, including but not limited to algebra, geometry, number theory, combinatorics, prealgebra, probability, statistics, and calculus. Aim for a difficulty such that fewer than 30% of advanced high-school students could solve it. Avoid reusing textbook clichés or famous contest problems.
THEN, without revealing any of your private thoughts, output **exactly** the following two blocks:

<question>
{The full problem statement on one or more lines}
</question>

\boxed{final_answer}

Do NOT output anything else—no explanations, no extra markup.
User Message:
Generate one new, challenging reasoning question now. Remember to format the output exactly as instructed.

GPT-5.5 Judge Prompt

- **Model**: `gpt-5.5`

System Message:
You are a math answer checker.
User Message Template:
Hi, there is an answer: {`answer`},
and the ground truth answer is: {`response`},
please check whether the answer is correct or not, and return **only** Yes or No.

Note: {`answer`} *is a placeholder for the model-generated answer, and* {`response`} *is the benchmark ground-truth answer.*

A.3 Repetition-Penalty Implementation

Following R-Zero (Huang et al., 2026), we discourage the Questioner from producing near-duplicate questions in a reward batch. For questions x_i and x_j, we first compute

$$d_{ij} = 1 - \mathrm{BLEU}(x_i, x_j). \tag{19}$$

We use NLTK's `sentence_bleu` with `SmoothingFunction().method1`; questions are tokenized by whitespace without additional lowercasing or punctuation removal. We then apply agglomerative clustering to the precomputed distance matrix with average linkage and the fixed distance threshold used throughout all experiments. If $C(x_i)$ is the cluster containing x_i and B is the reward-batch size, the penalty is

$$r_{\mathrm{rep}}(x_i) = \frac{|C(x_i)|}{B}. \tag{20}$$

This is the ρ_b term in Equation 7.

A.4 Reward-Accuracy Audit

For the analysis in Section 5.1, we draw $M = 500$ question–answer groups from the Qwen3-4B-Base Solver training data at each reported iteration. GPT-5.5 is used only as an external post-hoc correctness annotator (OpenAI, 2026). Let $\mathcal{C}$ and $\mathcal{I}$ be the sets of responses that the teacher judges correct and incorrect, respectively. For an internal reward R_i, we report

$$\mathrm{RewardAccuracy}(R) = \frac{100}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} R_i. \tag{21}$$

Thus, the reported value is the mean reward assigned to externally judged correct responses, expressed as a percentage. The mixed-reward curve uses $t = 0.5$, matching the 4B training configuration. No GPT-5.5 judgment is exposed to the Questioner, Solver, or Thinker during training. For each internal reward signal, we also report

$$\mathrm{MeanIncorrectReward}(R) = \frac{100}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} R_i, \tag{22}$$

which measures the mean reward assigned to teacher-judged incorrect responses.

B Evaluation Benchmarks

We evaluate whether the improvements produced by the math-focused Questioner transfer both to mathematical problem solving and to general-domain reasoning. The benchmark suite exactly matches the six columns reported in Tables 1 and 2.

Mathematical Reasoning. We use GSM8K (Cobbe et al., 2021), AMC, and AIME-2025. Following the R-Zero evaluation protocol (Huang et al., 2026), the GPT-5.5 judge shown above checks semantically complex final answers against the benchmark ground truth. We report **mean@32** for AMC and AIME-2025, and greedy-decoding accuracy for GSM8K.

General-Domain Reasoning. We evaluate three benchmarks:

- **MMLU-Pro** (Wang et al., 2024b), a more challenging and reasoning-focused extension of MMLU (Hendrycks et al., 2021).
- **SuperGPQA** (Du et al., 2025), a graduate-level benchmark spanning 285 disciplines.
- **BBEH** (Kazemi et al., 2025), which replaces the tasks in BIG-Bench Hard (Suzgun et al., 2023) with substantially more difficult counterparts targeting related reasoning abilities.

For these benchmarks, we follow the prompts and evaluation code released with General-Reasoner (Ma et al., 2025) and report Exact Match accuracy under greedy decoding.