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

Chengsong Huang[1,2†], **Wenhao Yu**[1†], **Xiaoyang Wang**[1], **Hongming Zhang**[1], **Zongxia Li**[1,3], **Ruosen Li**[1,4], **Jiaxin Huang**[2], **Haitao Mi**[1], **Dong Yu**[1]

[1]Tencent AI Seattle Lab, [2]Washington University in St. Louis,
[3]University of Maryland, College Park, [4]The University of Texas at Dallas
† Core contributors
chengsong@wustl.edu; wenhaowyu@global.tencent.com

Abstract

Self-evolving Large Language Models (LLMs) offer a scalable path toward superintelligence by autonomously generating, refining, and learning from their own experiences. However, existing methods for training such models still rely heavily on vast human curated tasks and labels, typically via fine-tuning or reinforcement learning, which poses a fundamental bottleneck to advancing AI systems toward capabilities beyond human intelligence. To overcome this limitation, we introduce *R-Zero*, a fully autonomous framework that generates its own training data from scratch. Starting from a single base LLM, *R-Zero* initializes two independent models with distinct roles – a **Challenger** and a **Solver**. These models are optimized separately and **co-evolve** through interaction: the Challenger is rewarded for proposing tasks near the edge of the Solver's capability, and the Solver is rewarded for solving increasingly challenging tasks posed by the Challenger. This process yields a targeted, self-improving curriculum without any pre-existing tasks and labels. Empirically, *R-Zero* substantially improves reasoning capability across different backbone LLMs, e.g., boosting the Qwen3-4B-Base by +6.49 on math reasoning benchmarks, and +7.54 on general-domain reasoning benchmarks.

Code: https://github.com/Chengsong-Huang/R-Zero.

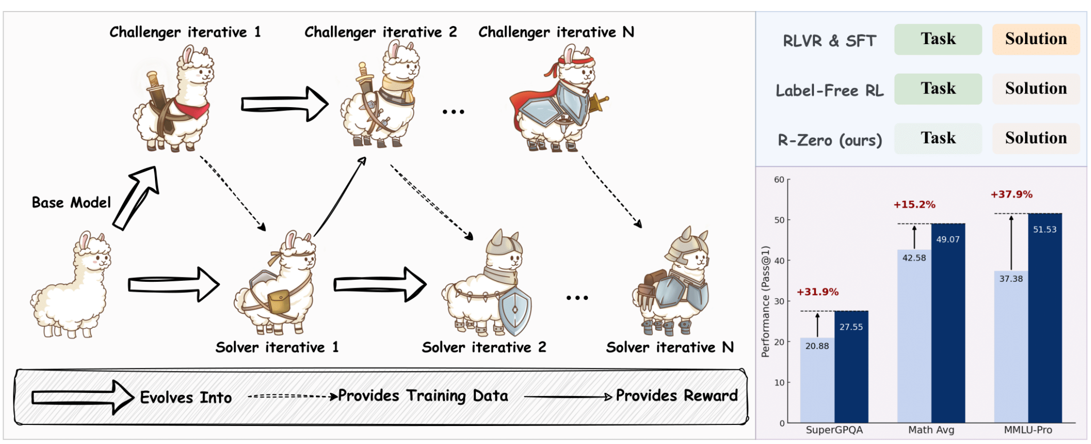

Figure 1: (Left): ***R-Zero*** employs a co-evolutionary loop between Challenger and Solver. (Right): ***R-Zero*** achieves strong benchmark gains without any pre-existing tasks or human labels.

1 Introduction

Self-evolving Large Language Models (LLMs) represent a promising frontier for advancing language intelligence. By autonomously generating, refining, and learning from their own experiences, these models provide a scalable pathway toward artificial superintelligence (Tao et al., 2024; Tan et al., 2024a). A critical requirement for training such self-evolving LLMs is access to large volumes of expertly curated tasks and labels, which serve as supervision signals for fine-tuning or reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; DeepSeek-AI et al., 2025). However, relying on human annotators to create these tasks and labels is not only costly, labor-intensive, and difficult to scale, but also presents a fundamental bottleneck to advancing AI systems toward capabilities that could eventually surpass human intelligence (Su et al., 2025; Zhao et al., 2025a).

To reduce dependence on human-curated data, self-generated and label-free methods have been proposed to eliminate the need for explicit supervision. In particular, label-free RL derives reward signals directly from the model's own outputs, such as sequence-level confidence scores (Li et al., 2025a; Prabhudesai et al., 2025; Huang et al., 2025) and output entropy (Agarwal et al., 2025; Cheng et al., 2025). However, despite removing the need for explicit labels, label-free methods still relies on a pre-existing corpus of tasks, which limits its scalability in truly self-evolving settings. On the other side, self-challenging approaches train LLMs on tasks generated by the models themselves (Zhou et al., 2025a; Wang et al., 2025a; Zhao et al., 2025a), While promising, many of these methods rely on external code executors to ensure that the synthesized tasks are both feasible and verifiable. However, in domains that lack an explicit verification oracle, such as open-ended reasoning, ensuring the quality and correctness of self-generated data remains a significant challenge.

In this paper, we propose *R-Zero*, a framework for training reasoning LLMs that can self-evolve from zero external data. In *R-Zero*, a single base model is initialized with two roles – a **Challenger** and a **Solver** that are independently optimized but **co-evolve** throughout the RL process. During co-evolving, the Challenger is rewarded for generating tasks targeted to be at the edge of Solver's current abilities, while the Solver is rewarded for solving increasingly challenging tasks posed by the Challenger. Framework details are provided in Section 3, but briefly, in the Challenger training phase, the Challenger is trained via Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to generate difficult questions. The reward signal is derived from the uncertainty for the frozen Solver, which is measured by the self-consistency of its multiple generated answers. In the Solver training phase, the Solver is fine-tuned with GRPO on a filtered set of these challenging questions generated by the now-frozen Challenger, using the pseudo-labels voted by itself. This entire process repeats, creating a self-evolving cycle that operates without any human intervention.

Our experiments demonstrate that *R-Zero* is a model-agnostic framework, consistently and iteratively improving the reasoning abilities of different backbone LLMs. For example, Qwen3-4B-Base model's average score on math benchmarks increased by a significant **+6.49** points after three iterations of self-evolution. Moreover, the reasoning skills learned through our math-focused questions can generalize to complex general-domain tasks, with models trained using *R-Zero* showing significant improvements on general domain reasoning benchmarks like MMLU-Pro (Wang et al., 2024) and SuperGPQA (Du et al., 2025). Our further analysis finds that *R-Zero* can act as a mid-training method, as models first improved by our method achieve higher performance after fine-tuned on labeled data. In addition, we provide an in-depth analysis that validates our framework's components, demonstrates its synergy with supervised fine-tuning, and characterizes the co-evolutionary dynamics to identify both strengths and limitations, offering insights for future research.

2 Preliminaries

Our work builds upon recent advancements in reinforcement learning for fine-tuning large language models. We briefly review two key methodologies that are relevant to our framework.

2.1 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a reinforcement learning algorithm that fine-tunes a policy LLM π_θ without a separate, learned value function. Its core idea is to normalize rewards based on the performance of other responses generated from the same prompt.

For a given prompt p, a policy LLM $\pi_{\theta_{\text{old}}}$ generates a group of G complete responses $\{x_1, \ldots, x_G\}$. Each response x_i is evaluated to receive a single scalar reward r_i. The rewards across the group are then normalized using a z-score to compute a response-level advantage:

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G) + \varepsilon_{\text{norm}}},$$

where $\varepsilon_{\text{norm}}$ is a small constant added for numerical stability.

Policy Update. The policy is updated using a clipped surrogate objective, similar to PPO, to ensure stable training. The objective, regularized by a KL-divergence penalty to constrain policy drift, is:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G} \min\Big(\tfrac{\pi_\theta(x_i)}{\pi_{\theta_{\text{old}}}(x_i)}\,\hat{A}_i,\ \text{clip}\big(\tfrac{\pi_\theta(x_i)}{\pi_{\theta_{\text{old}}}(x_i)}, 1-\epsilon, 1+\epsilon\big)\,\hat{A}_i\Big) \;+\; \beta\,\text{KL}\big(\pi_\theta \| \,\pi_{\theta_{\text{old}}}\big).$$

Maximizing the negative of this loss encourages the policy to increase the probability of generating responses with positive relative advantages, while the KL term, controlled by β, limits divergence from the previous policy.

2.2 Reinforcement Learning with Verifiable Rewards

Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2024) is a paradigm for fine-tuning models in domains where response quality can be deterministically verified. This approach relies on a rule-based verifier $v : \mathcal{X} \to \{0, 1\}$ that assigns a binary reward to each generation x_i:

$$r_i \;=\; v(x_i) \;=\; \begin{cases} 1, & \text{if } x_i \text{ satisfies a task-specific correctness check,} \\ 0, & \text{otherwise.} \end{cases}$$

This reward structure is especially effective for tasks like math, code generation with clear correctness criteria, and serves as the foundation for the reward mechanism in our Solver training.

3 Method

3.1 Overview

We propose ***R-Zero***, a fully automated framework featuring a **Challenger** and a **Solver**, both initialized from the same base LLM. The framework operates in an iterative loop. We illustrate the main framework in Figure 2. First, the Challenger (Q_θ) is trained with Group Relative Policy Optimization (GRPO) to generate synthetic questions that are challenging for the current Solver (Sec. 3.2). A training dataset of question-answer pairs is then constructed from these synthetic questions using a filtering strategy and a majority-vote mechanism (Sec. 3.3). Next, the Solver (S_ϕ) is fine-tuned on this new dataset, also using GRPO (Sec. 3.4). This iterative process allows the Challenger and Solver to co-evolve, leading to a progressively more capable Solver. The entire framework is self-supervised, requiring no human intervention.

3.2 Challenger Training

The Challenger, Q_θ, is an autoregressive language model trained to generate challenging questions. We train Q_θ using the GRPO algorithm detailed in Sec. 2. The core of this process lies in designing a reward function that accurately captures the desired properties of a "good" question. This final scalar reward, r_i, is then used in the GRPO advantage calculation. We focus on generating questions specifically within the domain of mathematics, as it provides a convenient and self-contained setting for our framework; the objective nature of mathematical answers allows for the straightforward generation of pseudo-labels via majority voting, without the need for external verification environments like code executors.

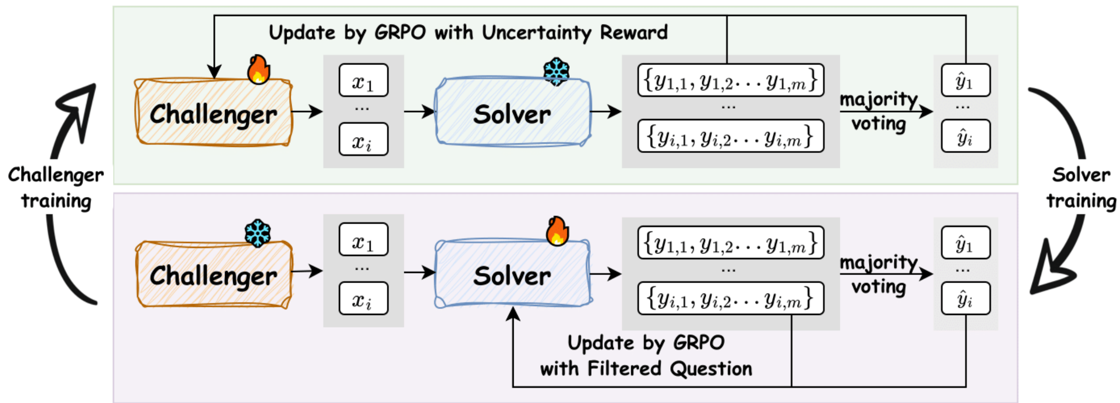

Figure 2: An overview of our *R-Zero* framework, which illustrates the co-evolution of the Challenger and the Solver. **Top:** In the Challenger training phase, the Challenger is trained via GRPO to generate difficult questions. The reward signal is derived from the uncertainty for the frozen Solver, which is measured by the self-consistency of its multiple generated answers. **Bottom:** In the Solver training phase, the Solver is fine-tuned with GRPO on a filtered set of these challenging questions generated by the now-frozen Challenger, using the pseudo-labels voted by itself.

Uncertainty Reward. To guide the Challenger toward producing challenging yet solvable questions, we first define an uncertainty score. For a generated question x, we query the current Solver S_ϕ for m responses $\{y_1, \dots, y_m\}$. The most frequent response is treated as the pseudo-label $\tilde{y}(x)$, and we compute the Solver's empirical accuracy as $\hat{p}(x; S_\phi) = \frac{1}{m}\sum_{j=1}^{m} \mathbb{1}\{y_j = \tilde{y}(x)\}$. The uncertainty reward is then defined as:

$$r_{\text{uncertainty}}(x; \phi) = 1 - 2\left|\hat{p}(x; S_\phi) - \tfrac{1}{2}\right|$$

This function incentivizes questions where the Solver is maximally uncertain (accuracy approaches 50%). We provide a theoretical motivation for this reward function in Sec. 3.5.

Repetition Penalty. To encourage diversity within a training batch $\mathcal{X}$, we introduce a repetition penalty. We could use any similarity metric, but in our case, we specifically use the BLEU score for faster computation, as this calculation must be performed numerous times during the rollout process. We compute pairwise distances using BLEU score similarity, $d_{ij} = 1 - \text{BLEU}(x_i, x_j)$, and group questions where $d_{ij} < \tau_{\text{BLEU}}$ into clusters $\mathcal{C} = \{C_1, \dots, C_K\}$. The penalty for a question x_i in a cluster C_k is proportional to its relative size:

$$r_{\text{rep}}(x_i) = \lambda \frac{|C_k|}{B}$$

where B is the batch size and λ is a scaling factor. In our experiments, we set $\lambda = 1$. The implementation details are shown in Appendix A.4.

Format Check Penalty. A critical first step in the reward pipeline is a structural format check to verify that each generated question is correctly enclosed within `<question>` and `</question>` tags. If the output does not adhere to this required structure, it is immediately assigned a final reward of 0, and no further reward signals are computed.

Composite Reward and Policy Update. For all questions that pass the format check, we calculate a composite reward. The final scalar reward r_i for each valid question x_i combines signals for uncertainty and repetition:

$$r_i = \max\left(0, r_{\text{uncertainty}}(x_i; \phi) - r_{\text{rep}}(x_i)\right)$$

With these rewards $\{r_1, \dots, r_G\}$ for a batch of generated questions, we compute the advantage $\hat{A}_i$ for each question and update the Challenger's policy Q_θ by minimizing the GRPO loss $\mathcal{L}_{\text{GRPO}}(\theta)$.

3.3 Solver Dataset Construction

After updating the Challenger, we use it to generate a new, curated dataset to train the Solver. This process acts as a curriculum generator. We first sample a large pool of N candidate questions from the Challenger's policy, $x_i \sim Q_\theta(\cdot \mid p_0)$. For each question, we obtain m answers from the current Solver, determine the pseudo-label $\tilde{y}_i$ via majority vote, and calculate the empirical correctness $\hat{p}_i$. A question-answer pair $(x_i, \tilde{y}_i)$ is added to the training set $\mathcal{S}$ only if its correctness falls within an informative band, $|\hat{p}_i - \frac{1}{2}| \leq \delta$. This filtering step discards tasks that are either too easy or too hard.

While the primary goal of this filtering is to discard tasks that are too easy or too hard, it also serves as an implicit quality control mechanism. Since our pseudo-labels are derived from a majority vote, a very low empirical correctness $\hat{p}_i$ often indicates that the question itself is ambiguous, ill-posed, or that the resulting pseudo-label is unreliable. By filtering out these low-consistency items, our method simultaneously improves the quality and the uncertainty calibration of the training data.

3.4 Solver Training

The Solver, S_ϕ, is then fine-tuned on the curated dataset of challenging problems $\mathcal{S}$. We also use GRPO for this stage, but with a simpler, verifiable reward signal. For a given question $x_i \in \mathcal{S}$ with its pseudo-label $\tilde{y}_i$, the Solver generates a batch of answers, each assigned a binary reward r_j:

$$r_j = \begin{cases} 1, & \text{if } x_j \text{ is identical to the pseudo-label } \tilde{y}_i, \\ 0, & \text{otherwise.} \end{cases}$$

This verifiable reward is used to compute the advantage $\hat{A}_j$, and the Solver's policy S_ϕ is subsequently updated by minimizing the GRPO loss $\mathcal{L}_{\text{GRPO}}(\phi)$. This process enhances the Solver's ability to correctly answer the difficult questions generated by its co-evolving Challenger.

3.5 Theoretical Analysis

In this section, we provide a theoretical motivation for our uncertainty reward function, $r_{\text{uncertainty}} \propto 1 - 2|\hat{p}(x; S_\phi) - \frac{1}{2}|$, which is maximized when the Solver's success probability, $\hat{p}$, is 50%. Our analysis is grounded in recent work that formally establishes that the most efficient training occurs when a learner is exposed to tasks at the frontier of its capabilities (Shi et al., 2025a; Bae et al., 2025).

The core insight from these studies is that the learning potential of the current Solver, with policy S_ϕ, can be quantified by the KL divergence to an optimal policy S^*. This divergence, $\mathbb{D}_{KL}(S_\phi || S^*)$, is lower-bounded by the variance of the Solver's reward. For the binary reward signal used in our framework, the success probability is $\hat{p}$. This leads to the specific lower bound:

$$\mathbb{D}_{KL}(S_\phi || S^*) \geq \frac{\hat{p}(1 - \hat{p})}{2\beta^2}$$

where β is the temperature parameter controlling entropy regularization. The right-hand side of the inequality, which is proportional to the reward variance, is maximized precisely when $\hat{p} = 0.5$. Therefore, by designing the Challenger's reward to incentivize questions that push the current Solver towards this point of maximum uncertainty, our framework is theoretically motivated to generate a maximally efficient curriculum in each iteration of the co-evolutionary process.

4 Experiments

4.1 Experiments Setting

4.1.1 Models

We employ the Qwen3-4B-Base (Yang et al., 2025) and Qwen3-8B-Base models to assess the impact of scale within a single architectural family. Second, to ensure our approach is effective on a distinct lineage, we utilize the OctoThinker-3B and OctoThinker-8B models (Wang et al., 2025b).This choice is particularly relevant as Wang et al. (2025b) reported that applying RL training directly to Llama models yielded suboptimal results. As the OctoThinker series is continually trained from the Llama-3.1 models (Dubey et al., 2024), this comprehensive selection allows us to test our framework across different foundational architectures – Qwen vs. Llama.

4.1.2 Evaluation Benchmark

We assess our framework on a comprehensive suite of benchmarks. Although the question-generator prompt for our method is primarily focused on mathematical problem-solving, a key objective of our evaluation is to explore whether the resulting improvements in reasoning ability can generalize to other domains. Therefore, our evaluation is divided into two main categories.

Mathematical Reasoning. We use seven challenging benchmarks: AMC, Minerva (Lewkowycz et al., 2022), MATH-500 (Hendrycks et al., 2021b), GSM8K (Cobbe et al., 2021), Olympiad-Bench (He et al., 2024), AIME-2024, and AIME-2025. For these tasks, where answers can be complex, we employ GPT-4o as a programmatic judge to semantically verify the correctness of the final answer against the ground truth (Zhao et al., 2025c). For the difficult AMC and AIME benchmarks, we report the **mean@32** metric. For all other math benchmarks, we report accuracy based on greedy decoding.

General Domain Reasoning. To test for the generalization of reasoning ability, we evaluate on the following challenging benchmarks:

- **MMLU-Pro** (Wang et al., 2024): An enhanced version of the MMLU (Hendrycks et al., 2021a) benchmark, featuring a more challenging suite of multi-task questions designed to provide a stricter evaluation of language model capabilities.
- **SuperGPQA** (Du et al., 2025): A large-scale benchmark focused on graduate-level reasoning. It comprises questions across 285 distinct disciplines that have been verified as unsearchable on the web, thereby isolating true reasoning ability from simple knowledge recall.
- **BBEH** (shoaa kazemi et al., 2025): This benchmark builds upon the foundation of BIG-Bench Hard (Suzgun et al., 2023) by incorporating a new selection of tasks specifically engineered to be more difficult, thus providing a more accurate measure of complex reasoning skills.

For this category, we follow the experimental setup, prompts, and evaluation codes from (Ma et al., 2025), reporting Exact Match (EM) accuracy obtained via greedy decoding.

4.1.3 Training Details

Our entire framework is implemented based on the EasyR1 codebase (Zheng et al., 2025b). In each iteration of the *R-Zero* co-evolutionary loop, we follow a specific set of hyperparameters. The Challenger (Q_θ) first generates a candidate pool of $N = 8,000$ questions. To construct the training dataset for the Solver, these questions are filtered based on consistency. For each candidate question, we sample $m = 10$ answers from the current Solver (S_ϕ). A question is retained for the training set only if the number of answers matching the majority-vote pseudo-label is between 3 and 7, inclusive ($\delta = 0.25$). This numerical range is consistent with the methodology used in previous research (Zhang & Zuo, 2025; Li et al., 2025b; Bercovich et al., 2025). When training the Challenger, the uncertainty reward $r(x; \phi)$ is calculated by sampling $m = 10$ responses from the Solver. For the intra-batch repetition penalty, we set the clustering distance threshold to $\tau_{\text{BLEU}} = 0.5$. Further implementation details and prompts can be found in Appendix A.

4.2 Results in Mathematical Reasoning

The comprehensive results of our experiments are presented in Table 1. The findings confirm that our proposed framework, *R-Zero*, is a highly effective, model-agnostic method for enhancing the performance of language models on mathematical tasks across different architectures and scales.

Table 1: Comprehensive results on mathematical reasoning benchmarks. We compare each base model against a **Base Challenger** baseline (where the Solver is trained on questions from an untrained Challenger) and our iterative method, *R-Zero*. The peak performance achieved during each model's training process is highlighted in **bold**.

Model Name	AVG	AMC	Minerva	MATH	GSM8K	Olympiad	AIME25	AIME24
Qwen3-4B-Base								
Base Model	42.58	45.70	38.24	68.20	87.79	41.04	6.15	10.94
Base Challenger	44.36	45.00	45.22	72.80	87.87	41.19	7.29	11.15
R-Zero (Iter 1)	48.06	51.56	51.47	78.60	91.28	43.85	**9.17**	10.52
R-Zero (Iter 2)	48.44	52.50	51.47	**79.80**	91.66	44.30	4.27	**15.10**
R-Zero (Iter 3)	**49.07**	**57.27**	**52.94**	79.60	**92.12**	**44.59**	4.27	12.71
Qwen3-8B-Base								
Base Model	49.18	51.95	50.00	78.00	89.08	44.74	16.67	13.85
Base Challenger	51.87	60.70	57.72	81.60	92.56	46.44	13.44	10.62
R-Zero (Iter 1)	53.39	61.56	59.93	**82.00**	93.71	48.00	14.17	14.37
R-Zero (Iter 2)	53.84	61.56	59.93	**82.00**	93.93	48.30	17.60	13.54
R-Zero (Iter 3)	**54.69**	**61.67**	**60.66**	**82.00**	**94.09**	**48.89**	**19.17**	**16.35**
OctoThinker-3B								
Base Model	26.64	17.19	24.26	**55.00**	73.69	16.15	0.21	0.00
Base Challenger	27.51	20.19	24.63	54.60	74.98	15.70	0.10	**2.40**
R-Zero (Iter 1)	27.76	20.39	25.74	54.60	**75.51**	16.30	0.10	1.67
R-Zero (Iter 2)	28.20	24.06	25.37	54.80	74.45	17.48	0.00	1.25
R-Zero (Iter 3)	**29.32**	**27.03**	**27.57**	54.20	74.98	**18.22**	**3.23**	0.00
OctoThinker-8B								
Base Model	36.41	32.11	41.91	65.20	86.96	26.52	1.56	0.62
Base Challenger	36.98	29.30	42.28	66.20	**88.10**	27.56	1.04	4.38
R-Zero (Iter 1)	37.80	32.97	45.22	65.60	86.96	**28.44**	**1.98**	3.44
R-Zero (Iter 2)	38.23	32.58	**48.53**	67.20	87.11	27.26	0.00	**4.90**
R-Zero (Iter 3)	**38.52**	**34.03**	48.22	**68.80**	87.19	27.56	0.42	3.44

Our iterative training process consistently and substantially improves upon the performance of the base models. This holds true for large models like Qwen3-8B-Base, where three iterations of *R-Zero* raise the average performance from a baseline of 49.18 to 54.69, a significant gain of **+5.51** points. Similarly, on the smaller OctoThinker-3B, our method improves the average score from 26.64 to 29.32 (**+2.68** points), demonstrating the broad applicability of our self-supervised training loop.

This improvement is progressive, with the results showing a clear trend of performance gains across iterations. For instance, the Qwen3-8B-Base model's average score climbs from a base performance of 49.18 to 53.39 (Iter 1) and ultimately reaches 54.69 (Iter 3). A similar monotonic improvement is observed on OctoThinker-3B, which progresses from its base score of 26.64 to 29.32 after three iterations. This consistent growth underscores the benefits of the co-evolutionary dynamic, where the progressively more capable Solver learns from an increasingly challenging curriculum.

The critical role of the Challenger's RL-based training is validated by the immediate performance leap from the Base Challenger to the first iteration of *R-Zero*. On Qwen3-8B-Base, this first iteration provides a +1.52 point gain over the baseline, and the improvement is even more pronounced on Qwen3-4B-Base at +3.7 points. This confirms that the intelligent curriculum generated by the RL-trained Challenger is significantly more effective than that of a non-trained generator.

4.3 Results in General Reasoning

Previous work has demonstrated that training language models on reasoning-intensive domains, such as mathematics, can lead to improvements in general-domain capabilities (Huan et al., 2025). A

Table 2: Results on general-domain reasoning benchmarks. The table compares the Base Model, a **Base Challenger** baseline, and our iterative *R-Zero*. The peak performance achieved during each model's training process is highlighted in **bold**.

Model Name	Overall AVG	MATH AVG	SuperGPQA	MMLU-Pro	BBEH
Qwen3-4B-Base					
Base Model	27.10	42.58	20.88	37.38	7.57
Base Challenger	30.83	44.36	24.77	47.59	6.59
R-Zero (Iter 1)	34.27	48.06	**27.92**	51.69	9.42
R-Zero (Iter 2)	**34.92**	48.44	27.72	**53.75**	9.76
R-Zero (Iter 3)	34.64	**49.07**	27.55	51.53	**10.42**
Qwen3-8B-Base					
Base Model	34.49	49.18	28.33	51.80	8.63
Base Challenger	36.43	51.87	30.12	54.14	9.60
R-Zero (Iter 1)	37.93	53.39	31.26	57.17	9.91
R-Zero (Iter 2)	38.45	53.84	**31.58**	58.20	10.20
R-Zero (Iter 3)	**38.73**	**54.69**	31.38	**58.23**	**10.60**
OctoThinker-3B					
Base Model	12.27	26.64	10.09	10.87	1.46
Base Challenger	14.41	27.51	11.19	14.53	**4.40**
R-Zero (Iter 1)	14.93	27.76	12.21	15.72	4.05
R-Zero (Iter 2)	15.11	28.20	12.43	16.08	3.74
R-Zero (Iter 3)	**15.67**	**29.32**	**12.44**	**16.71**	4.20
OctoThinker-8B					
Base Model	16.81	32.11	13.26	20.21	1.64
Base Challenger	25.08	36.41	16.99	41.46	5.46
R-Zero (Iter 1)	26.44	37.80	19.15	**42.05**	6.77
R-Zero (Iter 2)	26.77	38.23	19.27	41.34	**8.25**
R-Zero (Iter 3)	**26.88**	**38.52**	**19.82**	40.92	**8.25**

key question, however, is whether this generalization effect still holds when the training curriculum is not human-labeled, but entirely self-generated through *R-Zero*.

As shown in Table 2, this transfer of skills is evident across all tested models. For instance, three iterations of our math-focused training improve the average general-domain score of Qwen3-8B-Base by +3.81 points and OctoThinker-3B by +3.65 points. This generalization also extends to the key performance patterns observed in the mathematical results, with progressive iterative gains. This confirms that our method does not merely teach domain-specific knowledge, but enhances the model's underlying capabilities in a way that successfully generalizes across domains.

5 Analysis

In this section, we conduct a series of in-depth analyses to better understand the behavior and effectiveness of our *R-Zero* framework. To ensure consistency, all analytical experiments presented here were conducted on the Qwen3-4B-Base model, unless explicitly stated otherwise.

5.1 Ablation Study

To isolate the contribution of each key component within our *R-Zero* framework, we conduct a comprehensive ablation study on the `Qwen3-4B-Base` model. We specifically investigate the importance of three critical modules by disabling them one at a time and observing the impact on performance. The results are summarized in Table 3.

As shown in the table, removing any core components leads to a significant degradation in performance. The largest drop occurs when we disable the Challenger's reinforcement learning (**w/o RL-Challenger**), with the Math and General average scores decreasing by 3.7 and 4.1 points, respectively. This result highlighting the importance of our co-evolutionary curriculum generation process. Similarly, removing the **Repetition Penalty** also harms performance, indicating that generating a diverse set of questions is crucial for effective Solver training.

Table 3: Ablation study results on the Qwen3-4B-Base model. **w/o RL-Challenger**: Disables GRPO training for the Challenger. **w/o Filtering**: Disables the difficulty-based curriculum filtering. **w/o Rep. Penalty**: Removes the repetition penalty from the Challenger's reward.

Method	Math AVG	General AVG
R-Zero (full)	48.06	30.41
Ablations		
⊢ w/o RL-Challenger	44.36	26.32
⊢ w/o Rep. Penalty	45.76	27.56
⊢ w/o Filtering	47.35	24.26

Finally, disabling the **Task Filtering** module results in a notable performance drop, particularly on the general-domain average, which falls by over 6 points. As discussed in Section 3.3, this filtering serves a dual purpose: it calibrates the curriculum's difficulty and acts as an implicit quality control mechanism by removing questions with low answer consistency. Without this filter, the Solver is trained on a noisy and poorly curated dataset that likely includes ambiguous or ill-posed questions, which harms its ability to learn robustly.

5.2 Evolution of Question Difficulty and Data Accuracy

Table 4: Performance and data accuracy analysis. The highlighted column represents the *true accuracy* of the self-generated pseudo-labels for each question set.

	Performance of Evaluated Model (vs. Ground Truth)				
	Base Model	Solver (Iter 1)	Solver (Iter 2)	Solver (Iter 3)	Pseudo-Label Acc.
$\mathcal{D}_{\text{Iter 1}}$	48.0	59.0	57.0	61.0	79.0%
$\mathcal{D}_{\text{Iter 2}}$	52.5	53.0	51.5	53.5	69.0%
$\mathcal{D}_{\text{Iter 3}}$	44.0	47.0	45.0	50.5	63.0%

To understand the co-evolutionary dynamic, we analyzed how the Challenger's generated questions and their corresponding pseudo-labels change across iterations. We sampled 200 questions from the Challenger's policy after each of the first three training iterations, creating three distinct test sets: $\mathcal{D}_{\text{Iter 1}}$, $\mathcal{D}_{\text{Iter 2}}$, and $\mathcal{D}_{\text{Iter 3}}$. For this analysis, we assumed the external oracle model, GPT-4o, to be a perfect annotator, providing the ground truth answers for all generated questions.

The evaluation was conducted as follows: the performance of our internal models was measured against these GPT-4o ground truth answers. The score reported for GPT-4o itself, however, reflects the **true accuracy of our self-generated pseudo-labels** by comparing the pseudo label against the ground truth from the oracle (GPT-4o). The results on the filtered dataset are summarized in Table 4.

This analysis reveals a multi-faceted dynamic. The first finding is that the questions generated by the Challenger become **progressively more difficult**. This is directly evidenced by evaluating a fixed model against the evolving question sets. For instance, the performance of the static Solver (Iter 1), when measured against the consistent GPT-4o ground truth, drops from 59.0% on Iteration 1 questions to 47.0% on Iteration 3 questions. This confirms that the Challenger is successfully increasing the intrinsic difficulty of its curriculum. The second finding, revealed by the highlighted column, pertains to the **true accuracy of the self-generated dataset**. Unfortunately, while the accuracy of the pseudo-labels is initially high at 79.0%, it systematically drops to 63.0% by the third iteration. This trend indicates that as the system generates more difficult problems, the Solver's majority vote becomes a less reliable source for ground truth. This decline in data quality is a critical trade-off and a potential bottleneck for the framework's ultimate performance.

Finally, despite this drop in absolute label accuracy, the framework's internal reward mechanism functions precisely as designed. The scores on the table's diagonal show how each Solver performs on questions from its contemporary Challenger. The Solver (Iter 2) achieves 51.5% and the Solver (Iter 3) achieves 50.5% on their respective question sets. This demonstrates that the Challenger successfully calibrates the question difficulty to match the Solver's evolving capabilities, consistently targeting the 50% success rate that our reward function incentivizes.

5.3 Synergy with Supervised Data

To analyze the utility of our framework in scenarios where a labeled dataset is available, we measure the synergy between *R-Zero* and traditional supervised fine-tuning using labeled datasets[1]. The GRPO settings for this experiment were kept identical to our main experiments.

We first establish a supervised baseline by fine-tuning the base model directly on the labeled data. For this process, we employ GRPO, an approach similar to Zero-RL (Zeng et al., 2025).

We then apply our *R-Zero* framework, where at the end of each co-evolutionary iteration, the resulting checkpoint is also fine-tuned on the same labeled dataset. The results show that our method provides significant additional gains. As highlighted in Figure 3, this represents a gain of **+2.35 points** over the direct training baseline.

This finding confirms that *R-Zero* is not redundant with labeled data; instead, it acts as a powerful performance amplifier. The co-evolutionary process enables the model to better leverage the supervised information and achieve performance levels unattainable by standard fine-tuning alone.

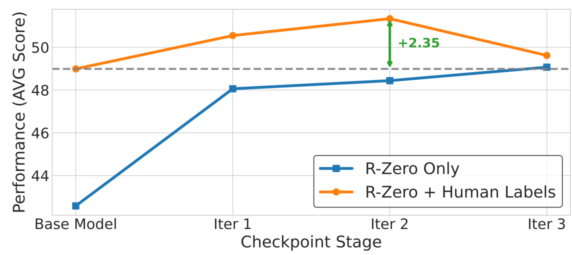

Figure 3: Performance of *R-Zero* when combined with supervised fine-tuning. The dashed line represents the baseline of fine-tuning the base model on labelled data alone, showing that our iterative method provides a better initialization.

5.4 Iteration Scaling

Previous results demonstrate that *R-Zero* generally enhance the Solver's capabilities across iterations. A closer inspection, however, reveals that the improvement is not consistent, with performance on certain challenging benchmarks degrading in later iterations. This raises a critical question about the long-term stability of our self-improvement loop: *what are the limits of this process, and what causes this eventual performance degradation?* In this section, we conduct a dedicated analysis to investigate these iteration scaling dynamics, aiming to diagnose the underlying cause of this instability.

5.4.1 The Inevitability of Collapse: An Empirical Analysis

As illustrated in Figure 4, our framework initially delivers on its promise, with models of all sizes showing significant performance improvements in the early stages of co-evolution. Unfortunately, this virtuous cycle does not continue indefinitely. After multiple iterations, we observe a consistent and concerning trend of performance degradation across all models. Intriguingly, we found a direct correlation between model scale and resilience to this collapse: the larger the model, the later the onset of performance degradation.

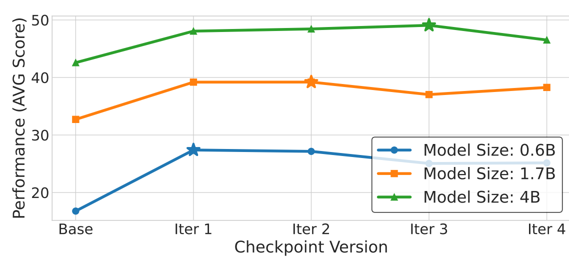

Figure 4: Math performance across different iteration times and model scales. The star markers indicate the peak performance for each model size.

[1] https://huggingface.co/datasets/hiyouga/math12k

For instance, the smallest 0.6B model reaches its peak performance as early as the first iteration (Iter 1), after which its capabilities begin to decline. In contrast, the largest 4B model sustains its upward trajectory for three full iterations, only experiencing a sharp drop at Iter 4. This pattern strongly suggests that while larger model capacity can delay the negative effects, it does not prevent them. This eventual collapse points to an inherent instability or limitation within our current self-improvement framework, highlighting a critical area for future investigation.

5.4.2 Beyond Label Noise: Unpacking the Roots of Instability

The most immediate hypothesis for this performance collapse is the degradation of pseudo-label quality, a potential failure mode of the self-correction mechanism we discussed in Section 5.2. As the Challenger generates increasingly difficult problems, it is plausible that the Solver's majority vote becomes a less reliable source for ground truth, resulting in a noisy training signal that could ultimately harm performance. To empirically test the extent to which this is the primary cause, we sampled 500 questions from a later training iteration to conduct a more granular investigation into the relationship between pseudo-label fidelity and the observed performance drop.

Table 5: Accuracy of self-generated pseudo-labels (%), labeled by Gemini. Shaded and bolded values indicate the best checkpoint for each model size.

Iteration	Model Size		
	0.6B	**1.7B**	**4B**
Iter 1	**70.6**	69.4	71.0
Iter 2	53.4	**55.2**	56.2
Iter 3	50.8	52.2	**48.8**
Iter 4	44.0	45.2	42.2

Although the degradation of pseudo-label accuracy is a consistent trend across iterations, our analysis suggests this is not the primary, nor even the sole, driver of the eventual performance collapse. Table 5 presents the pseudo-label data quality for each model at the onset of its performance collapse. Intriguingly, there appears to be no universal accuracy threshold that triggers this degradation. For instance, the 0.6B model begins to decline when data accuracy is still as high as 70.6% (Iter 1), whereas the 4B model tolerates an accuracy as low as 48.8% (Iter 3) before its performance drops.

This suggests that the absolute percentage of label noise is not the sole determinant of instability. Another potential, and perhaps more fundamental, reason is a form of model collapse that can be introduced when training exclusively on self-synthesized data (Tan et al., 2024b; Shumailov et al., 2024; Dohmatob et al., 2024b; Zhou et al., 2025b; Seddik et al., 2024; Dohmatob et al., 2024a; Briesch et al., 2023; Zheng et al., 2025a). A model can enter a degenerative feedback loop, suffering from a loss of diversity or an amplification of its own biases, which presents a significant challenge.

5.5 Parameter Sharing Between Challenger and Solver

Table 6: Comparison of math performance and pseudo-label accuracy between the standard R-Zero (two-model) and Single-R-Zero (unified model, shared parameters) frameworks across iterations.

	R-Zero (ours)		Single-R-Zero	
Iteration	**Performance**	**Pseudo-label Acc (%)**	**Performance**	**Pseudo-label Acc (%)**
Iter 1	48.06	71.0	**47.31**	63.4
Iter 2	48.44	56.2	46.95	46.6
Iter 3	**49.07**	48.8	45.57	32.6
Iter 4	46.52	42.2	43.89	33.8

To investigate whether the separation of the Challenger and Solver into two independent models is a necessary component for the success of *R-Zero*, we conduct an ablation study using a unified model with shared parameters. In this configuration (Single-R-Zero), a single model is tasked with performing both roles, i.e., generating a challenging curriculum and subsequently learning from it.

The results, presented in Table 6, clearly indicate that separating the Challenger and Solver into two independent models is crucial for both performance and stability. We observe two key findings. First, our standard two-model R-Zero framework not only achieves a higher peak performance (49.07) but also sustains improvement for more iterations, with its collapse occurring after the third iteration. In contrast, the unified Single-R-Zero model's performance peaks after the very first iteration and degrades immediately thereafter. Second, the Single-R-Zero model, where the agent must generate and solve its own problems, produces pseudo-labels of significantly lower accuracy at every stage. For example, in the first iteration, its pseudo-label accuracy is already substantially lower than the R-Zero's (63.4% vs. 71.0%). We hypothesize that this is because having the problem-setter and solver originate from the same model leads to a form of overconfidence that comes from internal bias.

6 Related Work

6.1 Label-Free Reinforcement Learning

A significant trend in recent research is Label-Free Reinforcement Learning, which aims to improve LLM reasoning without human-annotated data. Many such methods use the model's own outputs as a reward signal. This includes leveraging sequence-level confidence (Li et al., 2025a; Prabhudesai et al., 2025), the consistency of answers derived from varied reasoning paths (Zhang et al., 2025a; Zuo et al., 2025; Zhang et al., 2025b), minimizing the output entropy (Agarwal et al., 2025; Cheng et al., 2025), or even random (Shao et al., 2025) or negative reward (Zhu et al., 2025). These signals are often used within self-training loops where models fine-tune on their own most plausible solutions (Shafayat et al., 2025; Zhao et al., 2025b). While these methods all rely on a pre-existing set of unlabeled problems, *R-Zero* removes the need for any seed dataset.

6.2 Self-Play in Large Language Models

The paradigm of self-play, where models take on dual roles to create a self-improvement loop, has recently been adapted to improve language models without human data. This approach has been particularly fruitful in verifiable domains like code generation, where a "Coder" agent's program is verified by a "Tester" agent's unit tests (Lin et al., 2025; Wang et al., 2025a; Pourcel et al., 2025). More advanced frameworks push autonomy further by learning to generate the problems themselves, creating an adaptive curriculum from a small seed of examples or from scratch (Zhao et al., 2025a; Li et al., 2025c; Zhou et al., 2025a; Fang et al., 2025). Our work distinguishes itself by extending this paradigm to general reasoning domains that lack such verifiable environments, instead learning from a reward signal derived from the model's own internal consistency.

6.3 Reinforcement Learning with Verifiable Rewards (RLVR)

Reinforcement Learning with Verifiable Rewards (RLVR) has been widely adopted as a versatile paradigm for enhancing LLMs across a multitude of tasks. Its effectiveness is demonstrated in diverse applications such as relation extraction (Dai et al., 2025), interactive GUI navigation (Shi et al., 2025b) and search-engine utilization (Jin et al., 2025). While early implementations relied on rule-based verifiers, recent work has begun to explore more sophisticated, model-based verifiers (Ma et al., 2025; Li et al., 2025b; 2024).

7 Conclusion and Future Work

In this paper, we introduced *R-Zero*, a fully autonomous self-evolving framework that overcomes data dependency by having a Challenger and Solver co-evolve to create a self-generating curriculum. Our experiments demonstrate that *R-Zero* significantly improves LLM's reasoning capability on multiple domains. Future work could further focus on improving efficiency, exploring more robust labeling techniques, and expanding *R-Zero* to new domains. It is crucial to note, however, that the core mechanism of *R-Zero* is currently suited for domains where correctness can be objectively determined. Extending this self-evolutionary paradigm to open-ended generative tasks, such as creative writing or dialogue, where evaluation is subjective, remains a significant hurdle for future research. We believe *R-Zero* is a significant step towards creating truly self-evolving LLMs.

References

Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. *ArXiv preprint*, abs/2505.15134, 2025.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, et al. Online difficulty filtering for reasoning oriented reinforcement learning. *ArXiv preprint*, abs/2504.03380, 2025.

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, et al. Llama-nemotron: Efficient reasoning models. *ArXiv preprint*, abs/2505.00949, 2025.

Martin Briesch, Dominik Sobania, and Franz Rothlauf. Large language models suffer from their own output: An analysis of the self-consuming training loop. *ArXiv preprint*, abs/2311.16822, 2023.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, et al. Reasoning with exploration: An entropy perspective. *ArXiv preprint*, abs/2506.14758, 2025.

Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, et al. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168, 2021.

Runpeng Dai, Tong Zheng, Run Yang, and Hongtu Zhu. R1-re: Cross-domain relationship extraction with rlvr. *ArXiv preprint*, abs/2507.04642, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, abs/2501.12948, 2025.

Elvis Dohmatob, Yunzhen Feng, Arjun Subramonian, and Julia Kempe. Strong model collapse. *ArXiv preprint*, abs/2410.04840, 2024a.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, François Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024b.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *ArXiv preprint*, abs/2502.14739, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783, 2024.

Wenkai Fang, Shunyu Liu, Yang Zhou, Kongcheng Zhang, Tongya Zheng, et al. Serl: Self-play reinforcement learning for large language models with limited data. *ArXiv preprint*, abs/2505.20347, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proc. of ICLR*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, et al. Measuring mathematical problem solving with the math dataset. *ArXiv preprint*, abs/2103.03874, 2021b.

Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, et al. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. 2025.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. Efficient test-time scaling via self-calibration. *ArXiv preprint*, abs/2503.00031, 2025.

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

Nathan Lambert, Jacob Daniel Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, et al. Tülu 3: Pushing frontiers in open language model post-training. *ArXiv preprint*, abs/2411.15124, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Pengyi Li, Matvey Skripkin, Alexander Zubrey, Andrey Kuznetsov, and Ivan V. Oseledets. Confidence is all you need: Few-shot rl fine-tuning of language models. *ArXiv preprint*, abs/2506.06395, 2025a.

Ruosen Li, Ziming Luo, and Xinya Du. Fg-prm: Fine-grained hallucination detection and mitigation in language model mathematical reasoning. *ArXiv preprint*, abs/2410.06304, 2024.

Zongxia Li, Yapei Chang, Yuhang Zhou, Xiyang Wu, Zichao Liang, Yoo Yeon Sung, and Jordan Lee Boyd-Graber. Semantically-aware rewards for open-ended r1 training in free-form generation. *ArXiv preprint*, abs/2506.15068, 2025b.

Zongxia Li, Xiyang Wu, Guangyao Shi, Yubin Qin, Hongyang Du, Tianyi Zhou, Dinesh Manocha, and Jordan Lee Boyd-Graber. Videohallu: Evaluating and mitigating multi-modal hallucinations on synthetic video understanding. *ArXiv preprint*, abs/2505.01481, 2025c.

Zi Lin, Sheng Shen, Jingbo Shang, Jason Weston, and Yixin Nie. Learning to solve and verify: A self-play framework for code and test generation. *ArXiv preprint*, abs/2502.14948, 2025.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, et al. General-reasoner: Advancing llm reasoning across all domains. *ArXiv preprint*, abs/2505.14652, 2025.

Julien Pourcel, Cédric Colas, and Pierre-Yves Oudeyer. Self-improving language models for evolutionary program synthesis: A case study on arc-agi. 2025.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, et al. Maximizing confidence alone improves reasoning. *ArXiv preprint*, abs/2505.22660, 2025.

Mohamed El Amine Seddik, Suei-Wen Chen, Soufiane Hayou, Pierre Youssef, and Mérouane Debbah. How bad is training on synthetic data? a statistical analysis of language model collapse. *ArXiv preprint*, abs/2404.05090, 2024.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. Can large reasoning models self-train? *ArXiv preprint*, abs/2505.21444, 2025.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, et al. Spurious rewards: Rethinking training signals in rlvr. *ArXiv preprint*, abs/2506.10947, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300, 2024.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning. *ArXiv preprint*, abs/2504.05520, 2025a.

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, et al. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. 2025b.

Mehrangiz shoaa kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, et al. Big-bench extra hard. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. Ai models collapse when trained on recursively generated data. *Nature*, 631, 2024.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, et al. Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains. *ArXiv preprint*, abs/2503.23829, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In *Conference on Empirical Methods in Natural Language Processing*, 2024a.

Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. In *Proc. of EMNLP*, 2024b.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, et al. A survey on self-evolution of large language models. *ArXiv preprint*, abs/2404.14387, 2024.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving llm coder and unit tester via reinforcement learning. *ArXiv preprint*, abs/2506.03136, 2025a.

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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *ArXiv preprint*, abs/2506.20512, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388, 2025.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, et al. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *ArXiv preprint*, abs/2503.18892, 2025.

Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. *ArXiv preprint*, abs/2504.09696, 2025.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, et al. Consistent paths lead to truth: Self-rewarding reinforcement learning for llm reasoning. *ArXiv preprint*, abs/2506.08745, 2025a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *ArXiv preprint*, abs/2504.05812, 2025b.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, et al. Absolute zero: Reinforced self-play reasoning with zero data. *ArXiv preprint*, abs/2505.03335, 2025a.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Xiaodong Song. Learning to reason without external rewards. *ArXiv preprint*, abs/2505.19590, 2025b.

Yulai Zhao, Haolin Liu, Dian Yu, S. Y. Kung, Haitao Mi, and Dong Yu. One token to fool llm-as-a-judge. volume abs/2507.08794, 2025c.

Tong Zheng, Lichang Chen, Simeng Han, R Thomas McCoy, and Heng Huang. Learning to reason via mixture-of-thought for logical reasoning. *arXiv preprint arXiv:2505.15817*, 2025a.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, et al. Easyr1: An efficient, scalable, multi-modality rl training framework. 2025b.

Yifei Zhou, Sergey Levine, Jason E. Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *ArXiv preprint*, abs/2506.01716, 2025a.

Yujun Zhou, Jiayi Ye, Zipeng Ling, Yufei Han, Yue Huang, Haomin Zhuang, Zhenwen Liang, Kehan Guo, Taicheng Guo, Xiangqi Wang, et al. Dissecting logical reasoning in llms: A fine-grained evaluation and supervision study. *arXiv preprint arXiv:2506.04810*, 2025b.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, et al. The surprising effectiveness of negative reinforcement in llm reasoning. *ArXiv preprint*, abs/2506.01347, 2025.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, et al. Ttrl: Test-time reinforcement learning. *ArXiv preprint*, abs/2504.16084, 2025.

A Experiment Details

A.1 Training Hyperparameter

This section summarizes the most critical algorithmic hyperparameters for the Solver and Challenger training stages. All experiments were conducted using BFloat16 (BF16) mixed precision and FlashAttention 2.

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}
- **Max Steps**: 15
- **Number of Rollouts**: 5
- **Rollout Temperature**: 1.0
- **Rollout Top-p**: 0.99

A.1.2 Challenger Training

- **Global Batch Size**: 128
- **Learning Rate**: 1×10^{-6}
- **Weight Decay**: 1×10^{-2}
- **KL Penalty Coefficient** (λ_{KL}): 1×10^{-2}
- **Max Steps**: 5
- **Number of Rollouts**: 4
- **Rollout Temperature**: 1.0
- **Rollout Top-p**: 0.99

A.2 Prompt Templates

This section presents the exact prompt templates used for the solver and challenger models.

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.*

Challenger 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 re-using 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.

A.3 GPT-4o Judge Prompt

To programmatically evaluate the correctness of answers on mathematical benchmarks where the final answer can be complex (e.g., simplified expressions), we use GPT-4o as a judge. The exact prompt and configuration used for this evaluation are detailed below.

Configuration for GPT-4o as Judge

- **Model**: `gpt-4o`
- **Temperature**: 0.1

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 the **only**
Yes or No.

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

A.4 Repetition Penalty Implementation

To encourage the Challenger to generate a diverse set of questions within each batch, we apply a repetition penalty, r_{rep}. This penalty is designed to disincentivize the model from producing semantically similar questions in the same batch. The implementation is a multi-step process based on clustering questions by their BLEU score similarity.

1. Pairwise Distance Calculation via BLEU Score First, we compute a pairwise distance matrix for all questions in a batch. The distance d_{ij} between any two questions, x_i and x_j, is defined as one minus their BLEU score:

$$d_{ij} = 1 - \text{BLEU}(x_i, x_j)$$

For this calculation, we specifically use the `sentence_bleu` function from the NLTK library (`nltk.translate.bleu_score`). To ensure numerical stability, especially for shorter questions with limited n-gram overlap, we employ its first smoothing function, `SmoothingFunction().method1`. The questions are tokenized for the BLEU calculation by splitting on whitespace; no further text normalization, such as lowercasing or punctuation removal, is performed.

2. Agglomerative Clustering With the pairwise distance matrix computed, we then group similar questions using agglomerative hierarchical clustering. This step is performed using the `Clustering` implementation from the `scikit-learn` library. The clustering algorithm is configured with the following key parameters:

- **Metric**: Set to `'precomputed'`, indicating that we provide our custom BLEU-based distance matrix instead of having the algorithm compute distances.
- **Linkage**: Set to `'average'`. This method defines the distance between two clusters as the average of the distances between all pairs of questions across the two clusters.

3. Final Penalty Calculation Once each question in the batch is assigned to a cluster, the repetition penalty $r_{\text{rep}}(x_i)$ for a given question x_i is determined by the relative size of the cluster C_k to which it belongs. The penalty is calculated as:

$$r_{\text{rep}}(x_i) = \frac{|C_k|}{B}$$

Here, $|C_k|$ represents the number of questions in cluster C_k, and B is the total number of questions in the batch (i.e., the batch size).