DYNAMIC REINFORCEMENT LEARNING: SCALING LLM TRAINING WITH SELF-GENERATED DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

The scaling laws are a fundamental principle of large language models (LLMs), showing that model performance improves consistently as the amount of training data increases. In this paper, we propose Dynamic Reinforcement Learning (DynamicRL), a method that advances the scalability of RL for training LLMs autonomously. Unlike conventional approaches, DynamicRL does not require pre-collected datasets, human-labeled golden answers, or external verifiers for correctness; instead, it trains on self-generated data. DynamicRL operates by sampling data from the model as it evolves and using this self-generated data to optimize the model. Its dynamic characteristic allows the data distribution to continuously adapt to the evolving model, fostering better alignment between the training data and the model's capabilities. Experimental results demonstrate that DynamicRL can continuously improve model performance over more than a thousand training steps and achieve results comparable to models trained on large-scale external datasets.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable performance across a wide range of tasks (Jaech et al., 2024). A key factor driving this success is the principle of scaling laws (Kaplan et al., 2020), which shows that LLM performance improves with more training data. LLMs can be trained under different paradigms, such as supervised learning or reinforcement learning (RL), each with distinct scalability characteristics.

In supervised learning, models are trained on data sampled from a static distribution, requiring pre-collected datasets of questions paired with human-labeled solutions. However, these human-labeled solutions are costly and finite, which limits scalability. In contrast, RL can train models without relying on human-labeled solutions by sampling solutions from the dynamic LLM itself (Guo et al., 2025; Team et al., 2025), thereby achieving great scalability. Since questions are sampled from a static distribution while solutions are sampled from a dynamic distribution, we refer to this paradigm as *semi-dynamic reinforcement learning* (Semi-dynamicRL).

Despite its advantages, Semi-dynamicRL still depends on static datasets of human-created questions and human-labeled golden answers, which remain finite and costly. Moreover, since the questions are sampled from a static distribution that may not align with the evolving LLM, the effectiveness of the training data can gradually degrade over time (Yu et al., 2025; Zheng et al., 2025). This mismatch between static data and a continually evolving model ultimately constrains scalability.

In this paper, we propose *dynamic reinforcement learning* (DynamicRL), which takes a step toward scalable RL via the LLM itself. DynamicRL enables continuous learning from a data distribution that is dynamically generated by the LLM, allowing the model to progressively refine its capabilities without relying on external datasets, golden answers, or explicit correctness verifiers. Within this framework, both questions and solutions are generated from the LLM itself and are used to optimize the same LLM, forming a self-improving iterative training process. The dynamic nature of the training process allows the data distribution to co-evolve with the model, thereby maintaining alignment between the training data and the model's current capabilities.

Transitioning from Semi-dynamicRL to fully DynamicRL introduces additional challenges, most notably reward hacking and the risk of mode collapse. In DynamicRL, human-labeled golden answers are no longer available; instead, reward functions are defined based on estimated golden answers.

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.

Imperfect reward definitions may therefore induce reward hacking, whereby the model exploits weaknesses in the reward formulation to obtain artificially high rewards. To address these challenges, we first analyze the underlying failure modes and then propose mitigation strategies. Specifically, we design targeted prompts and filtering rules to exclude questions that are prone to exploitation, and we further regulate the training dynamics to encourage the generation of questions that are robust to reward hacking.

Another major challenge is mode collapse, a phenomenon in which generated data degenerates into a limited set of modes (Kossale et al., 2022). In DynamicRL, this manifests as question collapse, where the model repeatedly produces similar questions, and answer collapse, where the model defaults to identical answers. Such collapse ultimately leads to performance degradation. For example, the model may repeatedly generate nearly identical questions such as "Solve the equation $2x + 3 = 7$", or consistently output the same answer, such as "\\boxed{1}", across different questions. To mitigate this issue, we introduce a diversity reward function that prompts the model to generate new questions with diverse estimated golden answers. This mechanism effectively enhances both question and answer diversity, thereby alleviating mode collapse.

Finally, we experimentally validate the scalability of DynamicRL. The results show that DynamicRL scales effectively, sustaining training over more than a thousand steps without performance degradation, while achieving accuracy comparable to Semi-dynamicRL methods trained on large-scale, pre-collected static datasets.

2 METHOD

In this section, we first introduce our DynamicRL framework and discuss the reward hacking issues that arise in this setting, together with our solution.

2.1 DYNAMIC REINFORCEMENT LEARNING

Learning Scalability Scaling laws constitute a fundamental empirical principle in LLMs, indicating that model performance improves predictably as the model size, training data, and computational budget increase. This property underlies the success of LLMs, as it suggests that model capabilities can be enhanced by scaling training resources.

We analyze learning scalability from the perspective of training paradigms. LLMs are commonly trained using unsupervised learning, supervised learning, and RL. The supervised and unsupervised learning objectives can be unified as

$$\mathcal{J}_{sl}(\theta) = \mathbb{E}_{q\sim\phi,\, s\sim\mu(\cdot|q)}\big[\log \pi_\theta(s|q)\big],$$

where ϕ denotes the distribution over questions q, $\mu(\cdot|q)$ is the distribution of solutions s conditioned on q, and π_θ is the policy model parameterized by θ. Unsupervised learning can be viewed as a special case in which the conditioning variable q is absent.

In this formulation, both q and s are sampled from fixed data distributions. We therefore refer to this paradigm as *static learning*. Although the policy model π_θ evolves during training, the data distribution remains unchanged and cannot adapt to the model's evolving behavior. This mismatch inherently limits learning scalability, particularly as model capacity increases.

In contrast, RL optimizes a different objective:

$$\mathcal{J}_{rl}(\theta) = \mathbb{E}_{q\sim\phi,\, s\sim\pi_\theta(\cdot|q)}\big[R(q,s)\big],$$

where $R(q, s)$ is a reward function that evaluates the quality of a solution s given a question q. Recent work, such as DeepSeek-R1 (Guo et al., 2025), suggests that RL can improve scalability by enabling the model to learn from solutions sampled from its own evolving policy. Since solutions s are drawn from the dynamic distribution $\pi_\theta(\cdot|q)$ during training, this paradigm can be characterized as *semi-dynamic reinforcement learning*. Nevertheless, the questions q are still sampled from a fixed distribution ϕ, which continues to impose a scalability bottleneck.

Dynamic Reinforcement Learning To further improve the scalability of RL, we propose a framework termed *dynamic reinforcement learning*. In this framework, both questions and solutions are

drawn from distributions induced by the dynamic policy model π_θ, and training proceeds by iteratively optimizing the policy with respect to these self-generated samples. By continuously learning from dynamically evolving distributions, the policy model can progressively refine its capabilities.

To this end, we introduce two objective functions: one for improving solution quality and another for improving question quality. We first define the solution objective $\mathcal{J}_s(\theta)$:

$$\mathcal{J}_s(\theta) = \mathbb{E}_{q\sim\pi_\theta(\cdot|z),\, s\sim\pi_\theta(\cdot|q)}\big[R_s(q,s)\big], \tag{1}$$

where $\pi_\theta(\cdot|z)$ denotes the distribution over questions q conditioned on a prompt z, and $\pi_\theta(\cdot|q)$ denotes the distribution over solutions s conditioned on a question q. The solution reward function $R_s(q,s)$ evaluates the quality of a generated solution s for the corresponding question q.

We next define the question objective $\mathcal{J}_q(\theta)$:

$$\mathcal{J}_q(\theta) = \mathbb{E}_{z\sim p(z),\, q\sim\pi_\theta(\cdot|z)}\big[R_q(z,q)\big], \tag{2}$$

where $R_q(z,q)$ is the question reward function that assesses the quality of a generated question q given a prompt z. The distribution $p(z)$ specifies the sampling process for prompts.

In summary, DynamicRL jointly optimizes $\mathcal{J}_s(\theta)$ and $\mathcal{J}_q(\theta)$, enabling the policy model to simultaneously improve solution generation and question generation. In the following paragraphs, we detail the design of the prompt distribution $p(z)$, the solution reward function $R_s(q,s)$, and the question reward function $R_q(z,q)$.

Prompt As this work focuses on mathematical reasoning, we employ a single, fixed prompt, denoted by z_0, with $p(z_0) = 1$. That is, the prompt variable z is always instantiated as z_0. The prompt z_0 is explicitly designed to elicit mathematically well-posed questions, as shown in Prompt 1.

Specifically, z_0 consists of three components. The first instructs the model to generate exactly one mathematical question together with its step-by-step solution. Since the process of question generation may implicitly induce a solution, we allow the model to output both the question and its solution in a single response. The second component constrains the generated question to admit a unique, well-defined answer. The third component specifies a strict output format, from which we subsequently extract only the question portion.

Prompt z_0: Generate exactly one math question and its step-by-step solution. The answer to the question should exist and be unique.
Format the output as follows:
Question: <math question here>
Solution: <step-by-step solution here>

After obtaining the generated question q, we use a second prompt z_q, as shown in Prompt 2 to elicit the corresponding solution s. In this prompt, the placeholder {`question`} is replaced with the generated question q. The first instruction encourages step-by-step reasoning, while the second requires the model to abstain from answering when the question is ill-posed or admits multiple answers, as will be discussed in Section 2.2. For notational simplicity, although the policy should formally be written as $\pi_\theta(\cdot|q,z_q)$, we omit the explicit conditioning on z_q for simplicity of notation.

Prompt z_q: {`question`}
Please reason step by step, and put your final answer within \\boxed{}. If you cannot answer this question correctly, or if the question includes subquestions, or if the question lacks sufficient conditions to determine a unique answer, output "No Answer".

Solution Reward Function To define the solution reward function $R_s(q,s)$, we require a golden answer for each question q. In DynamicRL, however, such annotations are generally unavailable. We therefore estimate the golden answer using a majority-voting scheme (Wang et al.).

Concretely, for a given question q, we sample m candidate solutions $\{s_j\}_{j=1}^m$ from the current policy $\pi_\theta(\cdot|q)$. For each solution s_j, we extract a final answer $a_j = e(s_j)$ using an answer-extraction function $e(\cdot)$, which can be implemented via regular expressions (Hugging Face, 2025).

To aggregate these answers, we perform majority voting based on an answer-level similarity function $\rho_a(\cdot,\cdot)$ (Guo et al., 2025; Team et al., 2025). For any pair of extracted answers a_{j_1} and a_{j_2}, the similarity is defined as

$$\rho_a(a_{j_1}, a_{j_2}) = \begin{cases} 1, & \text{if } a_{j_1} \text{ and } a_{j_2} \text{ are equivalent,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Using this measure, we define the majority-voted golden answer $l(q)$ and its corresponding support size $r(q)$ as

$$l(q) = \arg \max_{a \in \{a_j\}_{j=1}^m} \sum_{k=1}^{m} \rho_a(a, a_k), \tag{4}$$

$$r(q) = \max_{a \in \{a_j\}_{j=1}^m} \sum_{k=1}^{m} \rho_a(a, a_k), \tag{5}$$

where $l(q)$ denotes the estimated golden answer for question q, and $r(q)$ measures the number of sampled answers that are equivalent to $l(q)$.

Finally, the solution reward function $R_s(q, s)$ assigns a binary reward based on whether the extracted answer $e(s)$ matches the majority-voted golden answer:

$$R_s(q, s) = \rho_a\big(l(q), e(s)\big). \tag{6}$$

Question Reward Function We define the question reward function in terms of the expected solution reward of a question q. For a given question q, the mean solution reward is defined as

$$p(q) = \mathbb{E}_{s \sim \pi_\theta(\cdot|q)}\big[R_s(q, s)\big] \approx \mathrm{Mean}\big(\{R_s(q, s_j)\}_{j=1}^m\big) = \frac{r(q)}{m}. \tag{7}$$

The quantity $p(q)$ serves as a proxy for question difficulty: larger values of $p(q)$ indicate that a higher fraction of sampled solutions are correct, corresponding to easier questions, whereas smaller values indicate harder questions.

Based on $p(q)$, we define the question reward function $R_q(z, q)$ as

$$R_q(z, q) = 1 - 2|p(q) - \frac{1}{2}| \tag{8}$$

As $p(q)$ increases from 0 to $\frac{1}{2}$, the reward increases; when $p(q)$ exceeds $\frac{1}{2}$, the reward decreases.

The question reward function $R_q(z, q)$ plays both a collaborative and an adversarial role relative to the solution reward $R_s(q, s)$. When $p(q) \leq \frac{1}{2}$, $R_q(z, q)$ favors easier questions, which in turn increases the expected solution reward and reinforces the optimization of $\mathcal{J}_s(\theta)$. Conversely, when $p(q) > \frac{1}{2}$, $R_q(z, q)$ encourages the generation of harder questions, counteracting further increases in $\mathcal{J}_s(\theta)$. This interplay steers the policy toward questions of moderate difficulty. The threshold $\frac{1}{2}$ can promote this balanced regime.

Mode Collapse Transitioning from Semi-dynamicRL to DynamicRL introduces additional challenges, particularly the mode collapse issues (Kossale et al., 2022). Mode collapse is a phenomenon in generative models in which the model produces a limited variety of outputs, ignoring many modes of the true data distribution. In DynamicRL, this occurs in two forms: question mode collapse and answer mode collapse.

Question mode collapse occurs when the questions generated by the policy model are highly similar. For example, the model repeatedly generates questions like "Solve the equation $2x + 3 = 7$.". This lack of diversity can lead to saturation or even collapse in model performance. Mitigating question mode collapse therefore requires mechanisms that encourage the generation of diverse questions.

Answer mode collapse arises when the extracted final answers from generated solutions become highly concentrated. Empirically, we observe that after several hundred training steps, the model often converges to producing the same final answer, such as "\\boxed{1}", across different questions, resulting in a collapse of training signals. This behavior is exacerbated by the use of majority voting to estimate golden answers, which systematically favors answers that are easy for the model to produce. As a result, preventing answer mode collapse necessitates encouraging the generation of questions whose majority-voted golden answers are themselves diverse.

Diversity Reward Function To mitigate mode collapse, we introduce diversity reward functions that incentivize greater variability in generated data. The core principle is that the more an object resembles others, the lower its diversity reward. We first present a general formulation, followed by specific forms for question and answer diversity rewards.

Let $\{x_k\}_{k=1}^K$ denote a set of K objects sampled from the policy model π_θ, where each object may represent a question or an answer. For any pair of objects in this set, we define a similarity metric $\rho(\cdot,\cdot) \in [0,1]$, where $\rho(x,x) = 1$ and larger values indicate greater similarity. The diversity reward for an object $x \in \{x_k\}_{k=1}^K$ is then defined as

$$R_d(x) = \frac{1}{\sum_{k=1}^K \rho(x, x_k)}. \tag{9}$$

Intuitively, if many objects in $\{x_k\}_{k=1}^K$ are similar to x, the sum $\sum_{k=1}^K \rho(x, x_k)$ will be large, resulting in a smaller diversity reward $R_d(x)$. Conversely, a larger $R_d(x)$ indicates that x is less similar to the other objects. Since $0 \le \rho(x, x_k) \le 1$, it follows that $0 \le R_d(x) \le 1$, with $R_d(x) = 1$ implying that x is only similar to itself.

The question diversity reward is then defined as

$$R_{dq}(q) = \frac{1}{\sum_{i=1}^n \rho_q(q, q_i)}, \tag{10}$$

where $q \in \{q_i\}_{i=1}^n$ and $\rho_q(\cdot,\cdot)$ measures the similarity between two questions. Each question q is first tokenized using a tokenizer $t(\cdot)$, and the similarity is computed via the sequence overlap ratio:

$$\begin{aligned}
\rho_q(q, q_i) =& \frac{\sum_{j=1}^{\min(|t(q)|,|t(q_i)|)} \mathbf{1}(t(q)_j = t(q_i)_j)}{\sqrt{|t(q)| \cdot |t(q_i)|}} \\
t(q) =& [t(q)_1, t(q)_2, t(q)_3, \cdots], t(q_i) = [t(q_i)_1, t(q_i)_2, t(q_i)_3, \cdots].
\end{aligned} \tag{11}$$

Other similarity metric, such as embedding-based similarity metric, achieves similar performance. The answer diversity reward is defined as

$$R_{da}(l(q)) = \frac{1}{\sum_{i=1}^n \rho_a(l(q), l(q_i))}, \tag{12}$$

where $l(\cdot)$ denotes the estimated golden answer from Eq. (4), and $\rho_a(\cdot,\cdot)$ is the answer similarity metric defined in Eq.(3).

By incorporating these diversity rewards into the training objective, the policy model is explicitly encouraged to generate questions and answers that cover a broader range, thereby mitigating both question and answer mode collapse.

Objective Function Combining the solution reward function, question reward function, question diversity reward function, and answer diversity reward function, we define the training objective as:

$$\begin{aligned}
\mathcal{L}(\theta) &= \mathcal{L}_q(\theta) + \mathcal{L}_s(\theta), \\
\mathcal{L}_q(\theta) &= \mathbb{E}_{z\sim p(z),\, q\sim\pi_\theta(\cdot|z)}\Big[\lambda_q R_q(z,q) + \lambda_{dq} R_{dq}(q) + \lambda_{da} R_{da}(l(q))\Big], \\
\mathcal{L}_s(\theta) &= \mathbb{E}_{q\sim\pi_\theta(\cdot|z),\, s\sim\pi_\theta(\cdot|q)}\Big[\lambda_s R_s(q,s)\Big].
\end{aligned} \tag{13}$$

where $\lambda_q, \lambda_{dq}, \lambda_{da}, \lambda_s \ge 0$ are coefficients to balance different reward functions.

To optimize $\mathcal{L}(\theta)$, we apply the policy gradient theorem (Sutton et al., 1999) to $\mathcal{L}_q(\theta)$ and $\mathcal{L}_s(\theta)$, yielding the surrogate objective

$$\begin{aligned}
\mathcal{J}(\theta) =& \mathbb{E}_{z\sim p(z),\, q\sim\pi_\theta(\cdot|z)}\Big[A_q \log \pi_\theta(q|z)\Big] + \mathbb{E}_{q\sim\pi_\theta(\cdot|z),\, s\sim\pi_\theta(\cdot|q)}\Big[A_s \log \pi_\theta(s|q)\Big], \\
A_q =& \lambda_q(R_q(z,q) - \text{Mean}(\{R_q(z,q_i)\}_{i=1}^n)) + \lambda_{dq}(R_{dq}(q) - \text{Mean}(\{R_{dq}(q_i)\}_{i=1}^n)) \\
&+ \lambda_{da}(R_{da}(l(q)) - \text{Mean}(\{R_{da}(l(q_i))\}_{i=1}^n)), \\
A_s =& \lambda_s(R_s(q,s) - \text{Mean}(\{R_s(q,s_j)\}_{j=1}^m)).
\end{aligned} \tag{14}$$

Notably, when applying the policy gradient to $\mathcal{L}_s(\theta)$, we treat $\pi_\theta(\cdot|z)$ as a fixed distribution, analogous to the Semi-dynamicRL setting. A complete implementation of the DynamicRL procedure is provided in Algorithm 1.

2.2 REWARD HACKING

Issues To enable scalable RL, the reliability of reward signals is crucial (Guo et al., 2025). However, imperfect reward definitions can lead to reward hacking. In DynamicRL, both the solution reward and the question reward are defined based on an estimated golden answer rather than the true golden answer. As this estimation may be unreliable, the learning process may suffer from reward hacking.

We first analyze reward hacking issues associated with the question reward. The question reward $R_q(z, q)$ is defined based on $p(q)$ (Eq.(7)), which is intended to serve as a proxy for question difficulty. However, $p(q)$ measures the consistency among the answers $\{a_j\}_{j=1}^m$ rather than the intrinsic difficulty of a question. As a result, the reward will encourage the policy to generate questions that induce low answer consistency.

Crucially, low answer consistency does not necessarily correspond to high question difficulty. Table 3 presents representative examples illustrating this mismatch. Two common failure modes arise: *Non-unique Answer* and *Sub-questions*. In both cases, the model can exploit the reward definition to obtain low consistency without genuinely increasing question difficulty.

The first three question in Table 3 fall into the category of *Non-unique Answer*, which admits no valid solution. Since the model may fail to detect the absence of a valid answer, it may still produce an arbitrary answer, resulting in inconsistent answers. This example demonstrates that questions without a unique well-defined answer can artificially lower answer consistency. Therefore, we require that every question admit a unique answer.

The fourth question illustrates the *Sub-questions* issue. This question consists of multiple sub-questions, yet the model may generate an answer for only one sub-question or place only one answer within "\\boxed{}". This behavior increases apparent answer diversity, thereby reducing consistency. Such questions are thus often misclassified as difficult under the current reward definition. To prevent this form of reward hacking, we require that each question contain no sub-questions.

We next analyze reward hacking issues associated with the solution reward. The solution reward $R_s(q, s)$ is defined based on an estimated golden answer $l(q)$ obtained via majority voting. However, when a question q exceeds the reasoning capability of the current model, the majority-voted answer may be incorrect.

The fifth question illustrates this *Overly difficult* issue. For this question, the true correct answer is $\frac{1}{8}$. However, among the sampled answers, we find that the frequency of "$\frac{1}{8}$" is 3, while that of "$\frac{1}{2}$" is 4. As a result, the majority-voted answer is incorrect and is mistakenly treated as the golden answer. This example demonstrates that, for overly difficult questions, majority voted answers fails to reliably approximate the golden answer.

Consequently, the solution reward may incentivize incorrect reasoning patterns, leading to reward hacking. To mitigate this issue, it is necessary to control question difficulty such that generated questions remain within the reasoning capacity of the current policy model.

Question Filtering To mitigate reward hacking for question reward, we further propose a question filtering mechanism. Specifically, we define a prompt to detect potentially exploitable questions and exclude them from training by setting both the question reward and the solution reward to zero, i.e., $R_q(z, q) = 0$ and $R_s(q, s) = 0$. Specifically, we include the instruction *"If you cannot answer this question correctly, or if the question includes subquestions, or if the question lacks sufficient conditions to determine a unique answer, output 'No Answer'."* in the Prompt 2. A question q is then filtered if any sampled answer a_j contains "No Answer". This rule can filter questions with subquestions or non-unique answers.

Difficulty Control To alleviate reward hacking from solution rewards, we control question difficulty so that it matches the capability of current policy by adjusting the hyperparameter λ_q in Eq.(14).

The question reward $R_q(z, q)$ encourages a gradual increase in question difficulty over the course of training. The rate of this progression is controlled by λ_q, which we update adaptively according to

$$\lambda_q^{(T+1)} = \max(0, \min(1, \lambda_q^{(T)} + (E(\{q_i\}_{i=1}^n)^{(T)} - \tau_e)) \tag{15}$$

where $\lambda_q^{(T)}$ denotes the value of λ_q at training step T, and the update is clipped to the interval $[0, 1]$. Here, $E(\{q_i\}_{i=1}^n)$ represents the proportion of questions with $r(q) = m$ (see Eq. (7)) in the current batch $\{q_i\}_{i=1}^n$. For the remaining λ_{dq}, λ_{da}, and λ_s, we fix their values to 1.

The statistic $E(\{q_i\}_{i=1}^n)$ serves as a signal for whether the evolution of question difficulty should be accelerated or decelerated. $E(\{q_i\}_{i=1}^n)$ serves as a metric to measure the difficulty of a batch questions. When $E(\{q_i\}_{i=1}^n) > \tau_z$, the proportion of questions with $r(q) = m$ is excessively high, indicating that question difficulty should increase more rapidly; accordingly, λ_q is increased to reduce $E(\{q_i\}_{i=1}^n)$. Conversely, when $E(\{q_i\}_{i=1}^n) < \tau_e$, λ_q is decreased to slow the progression of difficulty.

The hyperparameter τ_e governs the overall pace of difficulty evolution. Smaller values of τ_e result in more aggressive increases in question difficulty, whereas larger values promote a more conservative and gradual progression.

3 Related Work

Unsupervised RL DeepSeek-R1 (Guo et al., 2025) shows that RL can significantly enhance model performance without relying on human-labeled solutions. Nevertheless, it still depends on human-labeled golden answers to guide the learning process. In contrast, unsupervised RL seeks to train models entirely without human-labeled answers. Some approaches define objectives based on the consistency of model outputs, Zhang et al. (2025a) propose rewards derived from intermediate reasoning states, and Zuo et al. (2025) explore estimating golden answers through answer consistency. Meanwhile, Shao et al. (2025) demonstrate that even random or negative rewards can serve as effective training signals for RL. Other approaches, including Zhang et al. (2025b) and Agarwal et al. (2025), link RL with entropy minimization at either the sequence or token level, using answer entropy as a surrogate objective. These methods primarily focus on estimation strategies. In contrast, DynamicRL emphasizes generating questions that can be reliably assessed by estimation methods.

Self-play RL Self-play RL is a paradigm in which a model enhances its performance by iteratively interacting with versions of itself (Zhang et al., 2024). This approach typically relies on a verifiable environment, particularly in code-related tasks, where unit tests can provide efficient verification. Lin et al. (2025) propose a self-play solver-verifier framework that jointly improves a model's ability to generate both code and test units. Similarly, Wang et al. (2025) introduce a framework that co-evolves coding and unit test generation by leveraging feedback from their interactions. Other studies, such as Zhao et al. (2025) and Zhou et al. (2025), allow the LLM to generate code tasks and learn from them, provided these tasks are verifiable. Despite these advances, these methods still depend on an external verifier to check answer correctness, whereas DynamicRL estimates the golden answers directly from the model's outputs.

4 Experiments

4.1 Settings

Baselines We compare DynamicRL with Semi-dynamicRL (Guo et al., 2025). For Semi-dynamicRL, models are trained on datasets of varying scales, including the small-scale MATH-7.5K (Hendrycks et al., 2021), the medium-scale DAPO-Math-17K (Yu et al., 2025), and the large-scale DeepScaleR-Preview-40K (Luo et al., 2025). We consider two variants of Semi-dynamicRL: one in which the golden answers are estimated via majority voting, consistent with DynamicRL, and another in which the golden answers are human-labeled, serving as a strong baseline. We use Qwen2.5-Math-1.5B and Qwen2.5-Math-7B (Qwen et al., 2025) as the base models. We also compare with two self-play methods, Absolute-Zero (Zhao et al., 2025) and R-zero (Huang et al., 2025).

Evaluation We evaluate the models on four mathematical benchmarks: MATH500 (Hendrycks et al., 2021; Lightman et al., 2023), AMC23 (Art of Problem Solving, 2025b), AIME2024, and AIME2025 (Art of Problem Solving, 2025a). Performance is measured using the pass@1 metric, with results averaged over 16 evaluation runs.

Table 1: The performance on math datasets. Semi-dynamicRL (w/ ga 7.5K) and Semi-dynamicRL (w/o ga 7.5K) denote Semi-dynamicRL trained on a 7.5K math dataset with and without golden answers, respectively; other notations follow the same convention.

Methods	**MATH500**	**AMC23**	**AIME2024**	**AIME2025**	**Average**
			Qwen2.5-Math-1.5B		
Base Model	40.8	24.2	4.4	4.2	18.4
Semi-dynamicRL (w/ ga 7.5K)	68.2	50.0	11.7	6.3	34.0
Semi-dynamicRL (w/ ga 17K)	69.7	53.1	15.0	7.9	36.4
Semi-dynamicRL (w/ ga 40K)	73.9	**55.5**	**18.5**	9.4	**39.3**
Semi-dynamicRL (w/o ga 7.5K)	67.7	49.1	14.6	5.0	34.1
Semi-dynamicRL (w/o ga 17K)	67.9	49.5	14.2	5.0	34.1
Semi-dynamicRL (w/o ga 40K)	70.6	52.2	15.6	5.8	36.1
Absolute-Zero	63.8	47.3	11.0	6.0	32.1
R-zero	68.9	49.4	13.1	5.8	34.3
DynamicRL	**75.6**	54.1	12.5	**10.5**	38.2
			Qwen2.5-Math-7B		
Base Model	54.4	37.7	13.3	6.7	28.0
Semi-dynamicRL (w/ ga 7.5K)	75.6	60.0	27.7	10.8	43.5
Semi-dynamicRL (w/ ga 17K)	79.7	64.5	28.1	14.0	46.6
Semi-dynamicRL (w/ ga 40K)	81.4	65.9	**31.7**	16.9	**48.9**
Semi-dynamicRL (w/o ga 7.5K)	75.6	61.1	24.6	12.3	43.4
Semi-dynamicRL (w/o ga 17K)	76.5	60.2	21.3	11.7	42.4
Semi-dynamicRL (w/o ga 40K)	78.5	60.3	21.3	12.1	43.0
Absolute-Zero	72.6	55.1	16.0	8.3	38.0
R-zero	78.8	62.4	20.2	10.9	43.1
DynamicRL	**83.2**	**67.8**	21.0	**17.7**	47.4

Table 2: The average performance on four math datasets.

Methods	DynamicRL	w/o R_q	w/o R_{dq}	w/o R_{da}	w/o filtering	$\tau_e = 4/8$	$\tau_e = 5/8$	$\tau_e = 6/8$
Average	47.4	45.1	44.3	45.1	44.0	46.3	47.4	46.2

4.2 RESULTS

See Table 1 and Figure 1 for the experimental results. Please refer to Appendix A.2 for more results. Golden answers are essential for enabling scalable learning in Semi-dynamicRL. When trained on a small-scale dataset (MATH-7.5K), Semi-dynamicRL without golden answers attains performance comparable to that of Semi-DynamicRL with golden answers. However, as the training data scale increases to medium or large datasets, the performance of Semi-dynamicRL without golden answers deteriorates substantially compared to its counterpart trained with golden answers. This degradation is primarily due to reward hacking, as discussed in Section 2.2, which is consistent with the results in (He et al., 2026). During continual training, the impact of reward hacking becomes increasingly pronounced, ultimately leading to performance degradation, as illustrated in Figure 1.

By contrast, DynamicRL achieves performance comparable to Semi-dynamicRL trained on large-scale datasets without relying on external data and golden answers. Under identical conditions without golden answers, DynamicRL substantially outperforms Semi-dynamicRL, Absolute-Zero and R-zero, highlighting the effectiveness of our proposed strategy for mitigating reward hacking.

4.3 ABLATION STUDIES

We conduct ablation studies to assess the effectiveness of the question reward R_q, question diversity reward R_{dq}, answer diversity reward R_{da}, question filtering and difficulty control. As shown in Table 2, all components consistently improve performance, demonstrating their effectiveness.

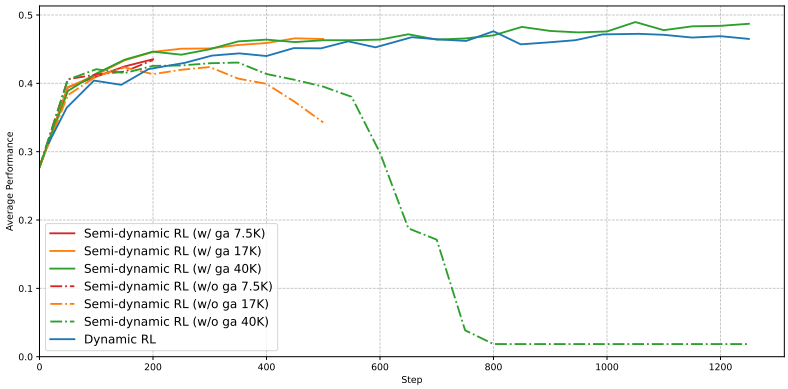

Figure 1: The average performance across different training steps.

Question reward. The question reward R_q encourages questions to become progressively harder during training. Without it, question difficulty fails to match the model's capability. As shown in Figure 2 (Appendix A.2), the proportion of questions with $r(q) = m$ (Eq. (7)), a proxy for easy questions, gradually rises and eventually dominates the batch. This imbalance produces too many easy questions and causes performance saturation. With R_q enabled, the ratio of questions with $r(q) = m$ remains low, supporting sustained learning progress.

Question diversity reward. The question diversity reward promotes diversity and prevents question mode collapse. Comparing training with $\lambda_{dq} = 1$ versus $\lambda_{dq} = 0$, Figure 3 shows that when $\lambda_{dq} = 0$, the reward drops near zero after 600 steps, with generated questions becoming highly similar, indicating severe mode collapse. In contrast, with $\lambda_{dq} = 1$, the reward rises early and stabilizes around 0.7, effectively preventing both collapse and excessive exploration.

Answer diversity reward. Similarly, the answer diversity reward encourages diverse answers and mitigates answer mode collapse. As illustrated in Figure 4, when $\lambda_{da} = 0$, the answer diversity reward decreases to nearly zero after Step 850. At Step 850, 97% of the sampled answers collapse to the same answer ("No Answer"). When $\lambda_{da} = 1$, the answer diversity reward steadily increases and eventually stabilizes near 1, successfully preventing answer mode collapse.

Question filtering. Question filtering eliminates questions prone to exploitation. Figure 5 shows the proportion of questions with non-unique answers during training. With filtering enabled, this proportion gradually decreases. We report non-unique answers only for experiments with filtering, as the metric depends on the filtering mechanism.

Difficulty control We study the effect of difficulty control by varying τ_e in Eq. (15). By tuning τ_e, we adjust the pace at which question difficulty evolves during training. The results in Table 2 indicate that the evolution of λ_q should be neither overly aggressive nor excessively conservative.

5 Conclusion

We propose DynamicRL, a framework that enhances RL scalability by leveraging the LLM itself to dynamically generate training data. To address reward hacking and mode collapse, we control training dynamics, filter questions, and introduce a diversity reward. Experiments show that DynamicRL can continuously improve performance without external training data or golden answers.

Despite its scalability, DynamicRL still performs below semi-dynamic RL trained with human-annotated golden answers, partly due to the absence of reliable ground-truth signals. DynamicRL also currently focuses primarily on mathematical reasoning tasks with relatively well-defined answer spaces. Future work will explore more accurate answer verification, difficulty calibration, and reward estimation, as well as extensions to broader domains with less structured answer spaces.

AI USE STATEMENT

In this work, generative AI tools were used solely for language editing and proofreading, such as improving grammar, clarity, readability, and the overall presentation of the manuscript. We did not use generative AI tools to develop research ideas, design the methodology, conduct experiments, analyze experimental results, or generate scientific claims, figures, or other research artifacts. All AI-assisted text was carefully reviewed and revised by the authors to ensure that it accurately represents the underlying research and does not introduce unsupported claims or errors. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

We have carefully reviewed the ICLR Code of Ethics and found no specific ethical concerns associated with this work. Our experiments rely exclusively on publicly available datasets and open-source models and do not involve human subjects, personally identifiable information, sensitive information, or private data. We adhere to standard practices for responsible research, with an emphasis on research integrity, transparency, and reproducibility.

REPRODUCIBILITY STATEMENT

To support the reproducibility of this work, we provide detailed descriptions of the DynamicRL framework and its training process in the main paper, covering the core methodology and algorithmic design. The implementation-related choices, hyperparameter configurations, and further experimental settings are documented in Appendix A.2. These details are intended to enable the research community to reproduce the training setup and experimental results of our study.

REFERENCES

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

Art of Problem Solving. Aime problems and solutions, 2025a. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions. Accessed: 2025-04-20.

Art of Problem Solving. Amc problems and solutions, 2025b. URL https://artofproblemsolving.com/wiki/index.php?title=AMC_Problems_and_Solutions. Accessed: 2025-04-20.

Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines. *arXiv preprint arXiv:2502.14739*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Bingxiang He, Yuxin Zuo, Zeyuan Liu, Shangziqi Zhao, Zixuan Fu, Junlin Yang, Cheng Qian, Kaiyan Zhang, Yuchen Fan, Ganqu Cui, et al. How far can unsupervised rlvr scale llm training? *arXiv preprint arXiv:2603.08660*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

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. *arXiv preprint arXiv:2508.05004*, 2025.

Hugging Face. Math-verify: A robust mathematical expression evaluation system. GitHub repository and Python package, 2025. https://github.com/huggingface/Math-Verify, version 0.8.0.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Youssef Kossale, Mohammed Airaj, and Aziz Darouichi. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, pp. 1–6. IEEE, 2022.

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 *The Twelfth International Conference on Learning Representations*, 2023.

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 arXiv:2502.14948*, 2025.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Kongcheng Zhang, Qi Yao, Shunyu Liu, Yingjie Wang, Baisheng Lai, Jieping Ye, Mingli Song, and Dacheng Tao. Consistent paths lead to truth: Self-rewarding reinforcement learning for llm reasoning. *arXiv preprint arXiv: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 arXiv:2504.05812*, 2025b.

Ruize Zhang, Zelai Xu, Chengdong Ma, Chao Yu, Wei-Wei Tu, Wenhao Tang, Shiyu Huang, Deheng Ye, Wenbo Ding, Yaodong Yang, et al. A survey on self-play methods in reinforcement learning. *arXiv preprint arXiv:2408.01072*, 2024.

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

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *arXiv preprint arXiv:2506.02177*, 2025.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *arXiv preprint arXiv:2506.01716*, 2025.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*, 2025.

A APPENDIX

A.1 EXAMPLES AND ALGORITHM

We show some examples of issues of generated questions in Table 3.

Table 3: Examples of issues in generated questions.

Issue	**Case**
Non-unique Answer (No Answer)	Find two integers x and y such that $x^2 + y^2 = 7$.
Non-unique Answer (Multiple Answers)	Find the unique positive integer x such that $\lfloor \frac{x}{5} \rfloor \lfloor \frac{x}{7} \rfloor = 15$.
Non-unique Answer (Insufficient Conditions)	What is the determinant of a 2×2 matrix?
Sub-questions	In triangle ABC, $\angle A = 163°$, $BC = 1.9$, and $\angle C = 7°$. Find: 1. The area of triangle ABC. 2. The circumradius of triangle ABC. 3. The inradius of triangle ABC.
Overly difficult	Let X and Y be independent random variables uniformly distributed on $[0, 1]$. Define $Z = \min(X, Y)$ and $W = \max(X, Y)$. Compute $P(Z + W \geq 1.5)$.

We show a complete implementation of the DynamicRL procedure in Algorithm 1.

Algorithm 1 Dynamic RL: Dynamic Reinforcement Learning

Input: Initial policy model π_{θ_0}, number of steps N, number of questions n, number of solutions m, prompt z_0, prompt z_q, coefficients $\{\lambda_q, \lambda_s, \lambda_{dq}, \lambda_{da}\}$, threshold τ_e.

1: Initialize policy model $\pi_\theta \leftarrow \pi_{\theta_0}$.
2: **for** step $T = 1$ **to** N **do**
3: Sample n questions q from $\pi_\theta(\cdot|z_0)$.
4: Sample m solutions s from $\pi_\theta(\cdot|q, z_q)$ for each question q.
5: Estimate the golden answer $l(q)$ by Eq. (4) for each question q.
6: Compute solution reward $R_s(q, s)$ by Eq. (6) for each question-solution pair (q, s).
7: Compute question reward $R_q(z, q)$ Eq. (8) for each question q.
8: Compute question diversity reward $R_{dq}(q)$ Eq. (10) for each question q.
9: Compute answer diversity reward $R_{da}(l(q))$ Eq. (12) for each question q.
10: Filter question by setting $R_q(z, q) = 0$ and $R_s(q, s) = 0$ according to Section 2.2.
11: Update λ_q by Eq.(15).
12: Optimize π_θ by maximizing objective function Eq. (14) via gradient ascent method.
13: **end for**

Output: Optimized policy model π_θ.

A.2 EXPERIMENTS

The additional experiments in this section serve the following purposes:

- **Majority Voting:** Evaluate the quality and reliability of majority-voted answer estimation across different training stages, and compare the training dynamics of DynamicRL and Semi-DynamicRL to demonstrate how adaptive difficulty control helps prevent reward collapse.
- **Validity of Generated Data:** Manually assess the correctness of generated reference answers at different training stages, identifying the main sources of invalid generated data and providing evidence for the reliability of the self-generated training data.

- **More Comprehensive Evaluation:** Evaluate DynamicRL on a larger Qwen3-8B-Base model and broader benchmark datasets, including general-domain tasks, to examine its effectiveness and generalizability across model scales and task domains.
- **Training Dynamics:** Analyze the evolution of question quality, question diversity, answer diversity, and filtering statistics throughout training, providing a detailed view of how DynamicRL controls and adapts the training data distribution.
- **Training Time:** Compare the training efficiency of DynamicRL and Semi-DynamicRL, and analyze how generated response length and training collapse affect the overall training cost.
- **Robustness:** Assess the stability of DynamicRL by measuring performance variation across three independent training runs with different random seeds, providing evidence that the observed performance is consistent across runs.
- **Examples of Generated Questions:** Provide representative generated questions at different training steps to illustrate how DynamicRL progressively increases question difficulty and complexity while maintaining valid and solvable question generation.
- **Hyperparameter Settings:** Provide detailed training and evaluation configurations, including batch size, rollout settings, sampling parameters, sequence lengths, difficulty-control parameters, training steps, and hardware, improving the transparency and reproducibility of our experiments.

Majority Voting To further demonstrate the quality of majority-voted estimation, we evaluate the performance of DynamicRL on external test datasets with known ground truth across multiple training checkpoints, as summarized in Table 4. For comparison, we also report the results of Semi-DynamicRL in Table 5.

We use *maj@16* (majority of 16 sampled answers) as the evaluation metric. For DynamicRL, the results show a steady increase in *maj@16* over time, with no signs of degradation. This consistent improvement indicates that majority-voted supervision, when integrated with the DynamicRL framework, effectively enhances performance.

In contrast, Semi-DynamicRL initially shows an increase in *maj@16*, which subsequently collapses to near zero as training progresses. This decline arises because Semi-DynamicRL often generates incorrect but consistent answers. Coupled with uncontrolled question difficulty, this behavior amplifies reward hacking, ultimately reducing overall performance.

Table 4: The *maj@16* metric of **DynamicRL** across different training steps.

Step	0	200	400	600	800	1000
MATH500	72.8	82.2	83.9	84.8	85.3	86.1
AMC23	50.2	69.3	69.1	71.0	69.9	70.9
AIME24	17.6	25.6	26.4	28.9	29.9	28.5
AIME25	8.9	14.7	18.1	17.8	19.2	21.8
Average	37.4	48.0	49.4	50.6	51.1	51.8

Table 5: The *maj@16* metric of **Semi-DynamicRL** across different training steps.

Step	0	200	400	600	800	1000
MATH500	72.8	83.2	81.4	63.9	5.0	5.0
AMC23	50.2	68.1	66.1	50.4	2.4	2.4
AIME24	17.6	28.4	19.5	8.1	0.0	0.0
AIME25	8.9	17.7	14.4	0.9	0.0	0.0
Average	37.4	49.4	45.4	30.8	1.9	1.9

Validity of Generated Data. To further evaluate the reliability of the generated questions and solutions, we manually inspect the correctness of the generated reference answers at different training steps. The results are shown in Table 6. The validity remains consistently high (81.25%–93.75%) throughout training, indicating that the proposed framework can reliably generate well-defined questions with correct solutions.

Table 6: Validity of generated reference answers at different training stages.

Step	200	400	600	800	1000
Validity	84.375%	87.500%	93.750%	87.500%	81.250%

The majority of invalid cases are caused by the model generating questions that are overly difficult and beyond the capability of the current model, rather than by ambiguity or missing conditions in the generated questions. This phenomenon is discussed in Section 2.2 and Table 3. Since both question difficulty and answer correctness are estimated automatically during training, estimation errors are inevitable and may occur when generated questions exceed the model's capability boundary.

These observations also motivate our adaptive difficulty control mechanism, which aims to maintain generated questions near the model's learning frontier rather than continuously increasing their difficulty. We include this manual evaluation and discussion in the revised manuscript to provide a more comprehensive assessment of the quality of the generated data.

More Comprehensive Evaluation We further report the performance of DynamicRL on the Qwen3-8B-Base model in Table 7. In addition, we include two general-domain benchmark datasets, SuperGPQA (Du et al., 2025) and MMLU-Pro (Wang et al., 2024), to provide a more comprehensive evaluation. The results show that DynamicRL consistently improves performance over the base model across different model scales and benchmark datasets, and significantly outperforms Semi-dynamicRL without golden answers. These results are consistent with Table 1 and further validate the effectiveness and generalizability of our DynamicRL approach.

Table 7: Performance comparison on Qwen3-8B-Base across six benchmark datasets.

Methods	MATH500	AMC23	AIME24	AIME25	MMLU-Pro	SuperGPQA	Average
Base	73.5	49.2	12.9	12.3	51.8	26.4	37.7
Semi-dynamicRL w/ golden answer	86.1	66.4	22.1	17.5	60.5	32.0	47.4
Semi-dynamicRL w/o golden answer	79.2	58.9	14.8	13.5	62.0	32.1	43.4
DynamicRL	83.5	63.4	20.4	16.2	62.1	32.6	46.4

Training Dynamics We present the training dynamics of our method. Figure 2 shows the proportion of questions satisfying $r(q) = m$. Figure 3 reports the evolution of the question diversity reward throughout training, while Figure 4 shows the corresponding dynamics of the answer diversity reward. Figure 5 illustrates the proportions of sub-questions and questions with the answer "No Answer" during training. Since the difficulty control mechanism in Eq. (15) depends on question filtering, we fix $\lambda_q = 1$ instead of applying Eq. (15) in experiments without question filtering.

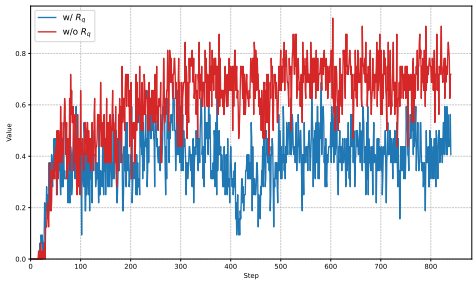

Figure 2: The ratio of questions with $r(q) = m$.

Training Time. The training time of each method is reported in Table 8. DynamicRL requires less training time than Semi-dynamicRL w/ golden answer because its generated responses are shorter. Since DynamicRL adaptively evolves question difficulty with model capability, the response length also adjusts accordingly. Semi-dynamicRL w/o golden answer has the lowest training time because its training collapses after certain steps, as shown in Figure 1.

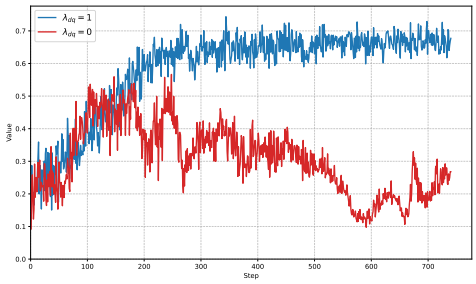

Figure 3: Question diversity reward.

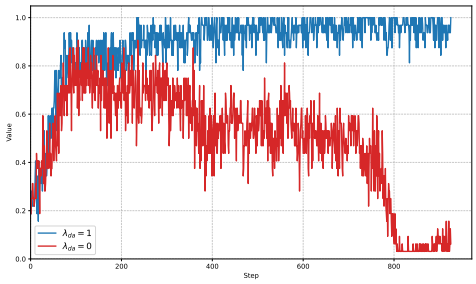

Figure 4: Answer diversity reward.

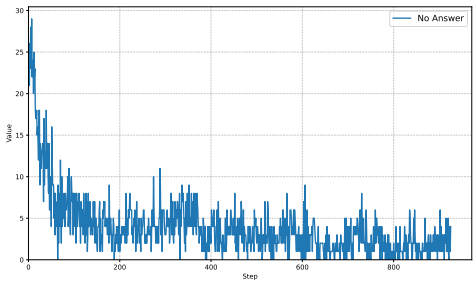

Figure 5: The ratio of questions with "No Answer".

Table 8: Training time of different methods.

Method	Semi-dynamicRL w/ golden answer	Semi-dynamicRL w/o golden answer	DynamicRL
Time	33.2h	12.4h	27.3h

Robustness. To assess robustness, we train each method with three independent random seeds and report the standard deviation of downstream performance in Table 9. The consistently low variance indicates that the observed performance is stable across different runs.

Examples of Generated Questions. We provide representative examples of generated questions at different training stages in Table 10. As training progresses, the generated questions gradually increase in difficulty and complexity, evolving from basic algebraic manipulation to more advanced topics such as set theory, multivariable calculus, and integral calculus. This demonstrates that DynamicRL can adaptively expand the difficulty frontier while maintaining valid and solvable question generation.

Table 9: Standard deviation over three independent training runs.

Method	MATH500	AMC23	AIME2024	AIME2025	Average
Semi-dynamicRL w/ golden answer	0.038	0.027	0.045	0.031	0.024
Semi-dynamicRL w/o golden answer	0.029	0.021	0.034	0.025	0.018
DynamicRL	0.047	0.035	0.052	0.041	0.031

Table 10: Representative generated questions at different training stages.

Step	Question
1	What is the value of x in the equation $2x + 3 = 7$?
50	Solve for x: $x^2 - 5x + 6 = 0$.
100	Consider three sets A, B, and C defined as follows: $A = \{x \in \mathbb{R} \mid x^2 - 3x + 2 \geq 0\}$, $B = \{x \in \mathbb{R} \mid x < 2\}$, $C = \{x \in \mathbb{R} \mid x^2 - 4 \leq 0\}$. Determine $(A \cup B) \cap C$.
500	Evaluate the line integral of the vector field $\vec{F}(x, y) = (y^2, x^2)$ along the curve defined by the parametric equations $x = \cos(t)$, $y = \sin(t)$, $0 \leq t \leq 2\pi$.
1000	Given the function $f(v) = v^5 + v^3$, find the integral of $\frac{f'(v)}{2f(v)}$ with respect to v.

Hyperparameter Settings We set the batch size to 32, number of rollouts to 16, and learning rate to 10^{-6}. For rollouts, we use temperature $= 1.0$ and top-$p = 1.0$, while for evaluation we use temperature $= 0.6$, top-$p = 0.95$ and top-$k = 20$. The maximum question length is set to 1024 tokens, and the maximum solution length is set to 3072 tokens. We search τ_e in $\{\frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}\}$ and finally set τ_e to $\frac{5}{8}$. We train semi-dynamic for 1 epoch and DynamicRL the same steps as semi-dynamic trained on DeepScaleR-Preview-40K dataset. All experiments conducted on four NVIDIA H100 GPUs.