Dissecting Long-Chain-of-Thought Reasoning Models: An Empirical Study

Anonymous ACL submission

Abstract

Despite recent progress in training long-chain-of-thought reasoning models via scaling reinforcement learning (RL), its underlying training dynamics remain poorly understood, and several counterintuitive behaviors persist. This work focuses on three key aspects: (1) We systematically analyze the roles of positive and negative samples in scaling RL, revealing that positive samples mainly facilitate precise fitting to the training data, whereas negative samples significantly enhance generalization and robustness. Interestingly, while positive samples are essential for convergence in the zero-RL setting, training on negative samples alone suffices to attain strong reasoning performance and even better generalization in cold-start scenarios. (2) We identify substantial data inefficiency in group relative policy optimization, where over half of the samples yield zero advantage. To address this, we explore two strategies, including relative length rewards and offline sample injection, to leverage these data better and enhance reasoning efficiency and capability. (3) We investigate unstable performance across various reasoning models and benchmarks, attributing instability to uncertain problems with ambiguous outcomes, and demonstrate that greedy decoding can distort evaluation by flipping the correctness of responses.

1 Introduction

Natural language processing has witnessed a breakthrough in reasoning capabilities within large language models (LLMs). Unlike previous models that depend on chain-of-thought (CoT) prompting (Wei et al., 2022), recent models emphasize scaling up inference computation for longer reasoning processes along with self-directed behaviors such as speculation, exploration, reflection, and verification (OpenAI, 2025).

Achieving such improvement is non-trivial, as it is challenging to construct high-quality supervised datasets that enable models to perform meticulous reasoning. Recent works reveal that scaling reinforcement learning (RL) plays a vital role in this context (DeepSeek-AI et al., 2025; Team et al., 2025). Compared to running supervised next token prediction, RL offers two key advantages. First, it eliminates the need for labeled data, enabling training on reasoning tasks without annotated intermediate steps. Second, its supervised signals come from feedback of the model's own generated responses, promoting the discovery of self-suitable reasoning routes towards correct answers.

Despite impressive advancements, our understanding of this field is still limited. To tackle this, some works focus on the question of whether the long-CoT reasoning capabilities are acquired at the scaling RL stage (Gandhi et al., 2025; Yue et al., 2025; AI et al., 2025). Other works demystify the training conditions under which long CoTs emerge (Chang et al., 2025). However, the underlying training dynamics of scaling RL remain poorly understood, and some unexpected behaviors in training and evaluating long-CoT reasoning models are still underexplained. In this work, we focus on the following three points:

Role of positive and negative samples: Scaling RL typically optimizes positive samples with advantages greater than zero, while suppressing negative samples with advantages less than zero. However, it remains unclear what models actually learn from positive and negative samples, and whether learning from negative samples—especially when they correspond to clearly incorrect answers—is necessary. Through systematic analysis, we reveal that positive samples primarily enhance model fitting, while negative samples significantly improve generalization and robustness in long-CoT reasoning. Furthermore, in the zero-RL setting, positive samples play a crucial role in enabling training convergence. In contrast, for already cold-started models, large-scale training on negative samples

can achieve reasoning performance comparable to using both positive and negative samples, while offering additional gains in generalization and robustness.

Zero advantage problem within GRPO: Yu et al. (2025) find that overly easy or overly difficult samples in group relative policy optimization (GRPO) (Shao et al., 2024) are useless and suggest discarding them. However, the severity of this problem and the potential for reusing such samples remain underexplored. We demonstrate that this problem is prevalent in widely used datasets, where over half of the data provides almost no contribution to the model's training. To address this, we explore two straightforward strategies. Relative length reward (RLR) leverages overly easy samples by assigning additional scores based on relative output length, encouraging efficient reasoning. Offline sample injection (OSI) facilitates learning from overly difficult problems by replacing incorrect online samples with correct offline solutions. Experimental results demonstrate that RLR enhances reasoning efficiency without sacrificing overall performance. However, learning from samples beyond the model's capacity remains challenging, even when correct solutions are provided.

Reason for unstable performance: Some works identify performance instability of RL-trained models during evaluation (Hochlehnert et al., 2025; Liu et al., 2025a), whereas a comprehensive explanation remains lacking. Our empirical study shows that performance instability persists across a wide range of reasoning models, regardless of model size or training method. Notably, this phenomenon arises from uncertain problems, where neither correct nor incorrect responses have clearly dominant probabilities. While greedy decoding helps maintain output consistency, it might distort evaluation by flipping the correctness of responses. In practice, performing multiple runs still offers a simple yet effective solution to stabilize evaluation scores, particularly on small benchmarks with a high proportion of uncertain problems.

2 Preliminary

This work follows the RL with verifiable rewards (RLVR) paradigm (DeepSeek-AI et al., 2025) to train long-CoT reasoning models. We leverage the GRPO algorithm to eliminate the value model typically used in proximal policy optimization (PPO) (Schulman et al., 2017), thereby improving training efficiency and eliminating a confounding variable in our analysis.

Group relative policy optimization. Given a set of problems Q, the old policy model $\pi_{\theta_{old}}$ first samples a group of responses $\{o_1, o_2, \cdots, o_G\}$ for each problem q. Different from PPO, which trains a value LLM to calculate advantage, GRPO computes the advantage $\hat{A}_{i,t}$ in a group relative manner to eliminate the value LLM, i.e.,

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}\big(\{r_i\}_{i=1}^{G}\big)}{\text{std}\big(\{r_i\}_{i=1}^{G}\big)}, \tag{1}$$

where r_i represents the reward score of the sample i computed by reward function $\mathcal{R}(\cdot)$. $\hat{A}_{i,t}$ persists similarly at each token t. Then, GRPO leverages a clipped surrogate objective that constrains the policy updates within a proximal region of the previous policy by:

$$\mathcal{C}_{i,t}(\theta) = \min\Big(r_{i,t}(\theta)\,\hat{A}_{i,t},\ \text{clip}\big(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\big)\,\hat{A}_{i,t}\Big), \tag{2}$$

where ε is a clipping-related hyperparameter and

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q)}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q)}. \tag{3}$$

Finally, GRPO updates the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q\sim Q,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{old}}(O|q)}$$
$$\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\mathcal{C}_{i,t}(\theta) - \beta D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)\right], \tag{4}$$

where β is a hyperparameter and $D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$ is the KL penalty term.

Rule-based reward. We use a rule-based reward function that verifies response correctness via matching algorithms and assigns scores accordingly, which is defined as follows:

$$\mathcal{R}(\hat{a}, a) = \begin{cases} 1, & \texttt{is\_equivalent}(\hat{a}, a) \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $\hat{a}$ and a denote the answer extracted from the model response and the ground truth, respectively.

Three-stage training. Sampling is one of the most time-consuming steps in scaling RL, especially when the response involves tens of thousands of tokens. Luo et al. (2025b) observe that most

Model	ID			ID (Fitness)		OOD (Generalization)			Add Noise (Robustness)			
	AIME24	AMC23	AIME25	AIME24 (8K)	AMC23 (8K)	GPQA	MMLU	GaoKao	AIME24 (N1)	AMC23 (N1)	AIME24 (N2)	AMC23 (N2)
DeepSeek-R1-Distill-Qwen-1.5B												
Orig	28.80	62.73	23.59	19.22 9.58%↓	52.33 10.39%↓	15.70	44.18	81.79	22.71 6.09%↓	53.46 9.26%↓	23.13 5.68%↓	54.52 8.21%↓
Both	**36.35**	**71.07**	**26.93**	**32.03** 4.32%↓	**68.11** 2.96%↓	18.02	49.84	**84.88**	23.85 12.50%↓	58.60 12.46%↓	26.46 9.90%↓	60.49 10.58%↓
Neg	34.95	68.77	26.30	29.69 5.26%↓	64.01 4.76%↓	19.68	**54.39**	84.03	**26.25** 8.70%↓	**63.20** 5.57%↓	**27.45** 7.50%↓	**64.51** 4.25%↓
Pos	34.01	69.48	23.80	30.26 3.75%↓	67.07 2.41%↓	**19.82**	48.37	84.29	23.70 10.31%↓	52.50 16.98%↓	24.64 9.38%↓	57.79 11.69%↓
DeepSeek-R1-Distill-Qwen-7B												
Orig	**55.31**	82.68	38.91	39.32 15.99%↓	69.82 12.86%↓	37.15	72.58	90.72	25.47 29.84%↓	58.83 23.85%↓	25.05 30.26%↓	60.99 21.69%↓
Both	54.43	84.90	39.74	**51.67** 2.76%↓	**84.34** 0.56%↓	43.56	**84.21**	**91.27**	25.68 28.75%↓	63.23 21.67%↓	24.90 29.53%↓	63.78 21.12%↓
Neg	54.90	**85.07**	**39.90**	50.99 3.91%↓	84.28 0.79%↓	**44.32**	83.19	**91.27**	**26.30** 28.59%↓	**64.08** 20.99%↓	**26.82** 28.07%↓	**65.38** 19.69%↓
Pos	49.17	80.03	32.76	47.86 1.30%↓	79.57 0.45%↓	41.98	79.42	91.11	24.43 24.74%↓	60.82 19.20%↓	24.06 25.10%↓	61.58 18.45%↓

Table 1: Training with different types of samples during scaling RL. "Orig" denotes the model without RL training. "Both", "Neg", and "Pos" indicate the policy model updated on both positive and negative samples, only negative samples, and only positive samples, respectively. "8K" means the maximum output length is set to 8192 during evaluation. "MMLU" stands for the MMLU-STEM benchmark. "N1" and "N2" refer to two types of many-shot noise (Zaremba et al., 2025) added to the prompt. The subscripts indicate the score differences caused by adding an output length limitation or input noise, relative to the original performance. Moreover, the best scores are **bolded**, and the second-best are underlined.

of the over-length responses are incorrect or consist of endlessly repetitive content. To avoid over-length responses during early training, they adopt a three-stage curriculum with progressively increasing maximum response lengths: 8192, 16384, and 24576 tokens for the first, second, and third stages, respectively.

Training setups. We conduct experiments on base models (e.g., `Qwen2.5-1.5B` (Yang et al., 2024)), supervised fine-tuning (SFT) models (e.g., `DeepSeek-R1-Distill-Qwen-1.5B` and `7B` (DeepSeek-AI et al., 2025)), and RL-trained models (e.g., `DeepScaleR-1.5B-Preview`). We chose the Qwen series of models due to their superior performance after RL scaling compared to other models (Zeng et al., 2025). By default, our training dataset is DeepScaleR-40K (Luo et al., 2025b), a high-quality dataset containing 40315 mathematics questions collected from a wide range of competitions and exercises. Please see the appendix for more details.

Evaluation setups. We select several representative reasoning benchmarks: AIME24, AMC23, AIME25, Math-500 (Hendrycks et al., 2021b), OlympiadBench (He et al., 2024), MMLU-STEM (Hendrycks et al., 2021a), GPQA diamond (Rein et al., 2023), and GaoKao Math QA (Zhong et al., 2024). The first five datasets serve as in-domain (ID) mathematical reasoning datasets, while the others are out-of-domain (OOD) multilingual and multi-subject reasoning benchmarks. By default, we generate from the models using a temperature of 0.6, a Top-p value of 0.95, and a maximum output length of 32768 tokens. We report the Pass@1 score averaged on sufficient runs for each dataset, detailed in Table 6. Please refer to the appendix for further details.

3 Role of Positive and Negative Samples

In scaling RL algorithms, the surrogate objective trains models to fit positive samples whose advantage is greater than zero, while suppressing negative samples with an advantage less than zero. In the context of RLVR, the positive samples refer to the correct responses successfully verified by the rule-based reward function, while the negative ones stand for incorrect or unverifiable answers. Natural questions arise: *What can long-CoT reasoning models learn from positive and negative samples during scaling RL? Is learning from negative or positive samples necessary?*

There have been different conjectures about the role of positive and negative samples:

- For a complex reasoning problem, the unsuccessful solution space is significantly larger than the correct one[1]. Thus, we conjecture that for complex reasoning tasks, negative samples have limited contribution, whereas positive samples are of primary importance.
- Some works argue to discard negative actions but only update the policy on positive ones to achieve better performance on conventional RL tasks (Srinivasan et al., 2018; Jesson et al., 2024).
- Other works offer an opposite opinion that during RL training, negative gradient, i.e.,

[1]Considering that any mistake in any of the reasoning steps will lead to an incorrect result, while the correct solution is limited.

learning to "push-down" likelihood on negative samples, results in faster accumulation of probability mass on a subset of high-reward responses compared to learning from positive samples (Tajwar et al., 2024).

We first empirically study the role of two sample types and then provide an explanation to answer the above questions.

3.1 Training with Only Positive or Negative Samples

The following experiments exclude the gradient contributions of specific sample types during policy model updates. Specifically, we first compute the advantage for all samples, then mask out the target samples before updating the policy. For example, to ablate the effect of negative samples, we compute the advantage as usual but remove negative samples from the batch prior to the policy update.

Setups. During training, we set the maximum sampling length to 8192 and train each model until its performance plateaus. As shown in Table 1, our evaluation covers four aspects. We assess *reasoning ability* using AIME24, AMC23, and AIME25, *goodness of fit* using two datasets under the 8192-token maximum output length, *generalization* using three OOD datasets, and *robustness* by injecting many-shot noise into the prompt, following Zaremba et al. (2025); Anil et al. (2024).

Results. The "ID" column in Table 1 indicates that training with only one type of sample can lead to improvements over the original models, and often yields a *final performance* comparable to training with both types. One exception is the performance drop of 7B model trained on positive samples. Considering the surrogate objective always trains models to fit on positive samples, we attribute the performance degradation of the 7B model to its greater susceptibility to overfitting, since its parameter size is significantly larger than that of the 1.5B model. In *goodness of fit* evaluations, models trained on both positive and negative samples perform best. Notably, the performance drop caused by limiting the maximum output length to 8192 is much smaller for models trained with positive samples than for those trained with negative samples, highlighting the crucial role of positive samples in fitting the training data[2]. For *generalization*, models trained solely with negative samples outperform others. A similar trend is observed in *robustness* evaluation, where negative-sample-only models consistently achieve superior performance, especially for the 1.5B model.

Takeaway 3.1 for Effect of Samples

Models trained on both types of samples achieve the best final performance (both $\geq$ negative $>$ positive), with positive samples aiding higher goodness of fit (both $\geq$ positive $>$ negative), and negative samples improving generalization (negative $\geq$ both $>$ positive) and robustness (negative $>$ both $>$ positive).

3.2 Why Does This Happen?

Suppose a policy π_{data} perfectly represents the distribution of the training data, while π^* denotes the optimal policy under the environment. As depicted in Figure 1 (a), the approximation of π_{data} to π^* is rarely perfect. A policy π_{pos} trained solely on positive samples, as shown in Figure 1 (b), tends to concentrate its probability mass in regions associated with high reward, but lacks the incentive to explicitly suppress low-reward regions. In contrast, as illustrated in Figure 1 (c), a policy π_{neg} trained only on negative samples flattens the probability in low-reward areas, but does not actively promote high-reward regions, leading to a more diffuse distribution.

While π_{pos} is more effective at fitting due to its focus on positive outcomes, its performance is highly dependent on how well the training data approximates the environment. Figure 1 (d) demonstrates that although π_{neg} fails to emphasize high-reward regions, it can still cover parts of π^* that are underrepresented in the training data, making it more robust than π_{pos} in certain scenarios.

3.3 Neg-to-Pos Training for Cold-Started Models

The first setting involves training cold-start models, in which SFT is applied before RLVR to provide a better initialization. The second is the zero-RL setting, in which base LLMs are directly trained using RLVR without prior fine-tuning. Here, we introduce a scaling RL procedure for cold-started models that first leverages negative samples and then fine-tunes the model using positive samples. The motivation is that training with negative sam-

[2]Despite the known length bias in GRPO (Liu et al., 2025b), our findings remain robust, as detailed in the appendix.

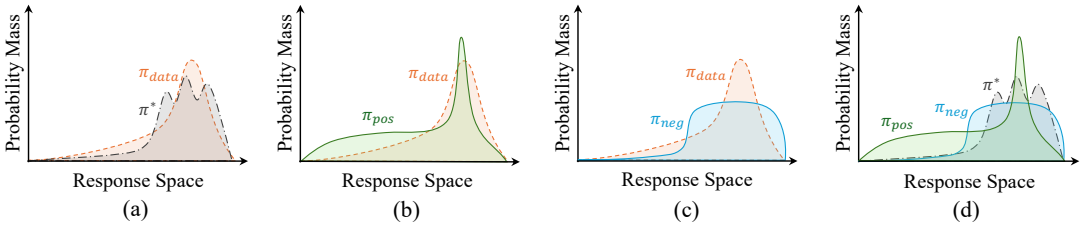


Figure 1: Illustration of training on two types of samples. The orange dotted line denotes the policy π_{data} that perfectly reflects the training data, while the gray one refers to the optimal policy π^*. The green and blue solid lines represent the policies learned solely from positive and negative samples, i.e., π_{pos} and π_{neg}, respectively.

Model	AIME24	AMC23	AIME25	GPQA	MMLU	GaoKao	AIME24 (N1)	AMC23 (N1)	AIME24 (N2)	AMC23 (N2)	AVG
8(NP)-16(NP)-24(NP)	**40.47**	**72.50**	**29.06**	19.52	51.14	**85.80**	27.40 13.07%↓	58.38 14.12%↓	28.91 11.56%↓	61.30 11.20%↓	47.45
8(N)-16(NP)-24(NP)	35.63	69.94	25.68	19.43	54.80	84.69	26.67 8.96%↓	62.90 7.04%↓	29.32 6.31%↓	64.59 5.35%↓	47.36
8(N)-16(N)-24(NP)	40.26	71.07	28.33	20.77	**57.34**	85.35	31.35 8.91%↓	64.82 6.25%↓	**31.61** 8.65%↓	65.19 5.88%↓	**49.61**
8(N)-16(N)-24(N)	40.16	71.52	28.75	19.24	54.74	85.04	30.68 9.48%↓	64.08 7.44%↓	30.26 9.90%↓	65.68 5.84%↓	49.01
8(N)-16(N)-24(P)	39.38	71.05	28.02	**21.23**	56.28	85.41	**31.61** 7.77%↓	**65.49** 5.56%↓	29.90 9.48%↓	**66.23** 4.82%↓	49.46
8(N)-16(P)-24(P)	37.76	70.31	25.42	21.21	56.22	85.01	27.76 10.00%↓	64.38 5.93%↓	28.23 9.53%↓	65.63 4.68%↓	48.19
8(P)-16(P)-24(P)	34.74	70.43	24.84	16.27	52.19	84.77	21.25 13.49%↓	53.52 16.91%↓	25.73 9.01%↓	59.92 10.51%↓	44.37

Table 2: Performance of scaling RL on a cold-started model, `DeepSeek-R1-Distill-Qwen-1.5B`. "8-16-24" denotes the three training stages introduced by Luo et al. (2025b). "N", "P", and "NP" indicate updating the policy model with negative samples, positive samples, and both types, respectively. For instance, "8(N)-16(N)-24(N)" stands for the policy model that is only trained on negative samples in all stages.

ples suppresses the generation of incorrect reasoning paths while simultaneously encouraging the model to allocate more output probability to potentially correct ones. This process effectively lays the foundation for the model's reasoning ability. Subsequently, fine-tuning with positive samples allows the model to better align with the training data. As a result, the final model is expected to achieve both strong performance and improved robustness.

Results. From the Table 2, we can see that the best and second-best results in each column are mainly achieved by 8(N)-16(N)-24(NP), 8(N)-16(N)-24(P), and 8(NP)-16(NP)-24(NP). Although 8(N)-16(N)-24(NP) and 8(N)-16(N)-24(P) slightly underperform 8(NP)-16(NP)-24(NP) on mathematical reasoning tasks, they outperform 8(NP)-16(NP)-24(NP) significantly on out-of-distribution datasets such as GPQA and MMLU-STEM. In terms of robustness, they also show clear improvements. Additionally, the performance of 8(P)-16(P)-24(P) is generally lower than others. We argue that, in cold-start scenarios, training solely on positive samples induces overfitting, thereby impairing generalization and robustness.

Takeaway 3.2 for Neg-to-Pos Training

Sequential training on negative and then positive samples enables cold-started models to generalize better, without sacrificing reasoning performance.

Benchmark	Orig	Both	Neg	Pos
GSM8K	4.00	78.71	6.25	76.98
MATH500	2.56	60.18	3.12	58.14

Table 3: Performance of zero-RL on `Qwen2.5-1.5B`.

3.4 Positive Sample Helps Convergence for Zero-RL

Next, we investigate the importance of the two sample types in zero-RL training. We conduct zero-RL experiments on `Qwen2.5-1.5B` using the same experimental settings as Zeng et al. (2025), ensuring each model is trained until convergence. As shown in Table 3, the "Both" and "Pos" models achieve satisfactory performance, whereas the "Orig" and "Neg" models exhibit poor scores. Upon analyzing the generated outputs, we find that the performance gap originates from the model's capability to follow the given instruction (i.e., "put the final answer in \boxed{}"). Models unable to adhere to this instruction produce unverifiable outputs, resulting in lower scores. These experimental results indicate that training on negative samples provides minimal benefit for instruction-following in zero-RL scenarios. Conversely, positive samples effectively enable models to learn instruction adherence.

Takeaway 3.3 for Zero-RL

Positive samples play a crucial role in achieving convergence during zero-RL training by explicitly guiding models toward following instructions.

4 Utilization of Zero Advantage Samples

As shown in the Equation 1, GRPO measures a sample's advantage by comparing its reward score to the average level of its group. This strategy is effective in most scenarios where the reward scores exhibit variance. However, it fails in an edge case where all reward scores are identical, lacking any distinguishable value. In this case, each score equals the mean, yielding zero advantage values and preventing these samples from contributing to gradients. We call this problem *zero advantage*.

In this section, we first reveal that when leveraging a rule-based reward function that only verifies the correctness, GRPO is susceptible to the zero advantage problem. Subsequently, to tackle this problem, we explore two straightforward strategies, including relative length reward (RLR) and offline samples injection (OSI). The former leverages fully positive samples (with reward scores of 1) to promote efficient reasoning, while the latter uses fully negative samples (with reward scores of 0) to further enhance RL-trained models.

4.1 Zero Advantage

Here, we empirically show how serious the zero advantage problem is in the commonly used reasoning datasets. Specifically, we sample eight rollouts for each problem in two popular datasets, DeepScaleR-40K and OpenThought-114K (Guha et al., 2025). The statistical results are shown in Figure 4. We can see that more than half of the problems receive consistent 1 or 0 reward scores, resulting in significant ineffective sampling. The remaining effective data accounts for 46.9% and 27.1% of the total data in DeepScaleR-40K and OpenThought-114K, respectively, suggesting significant data waste.

4.2 Relative Length Reward

When answering low-difficulty questions, concise responses are generally preferred. Thus, we introduce RLR, which adaptively assigns higher scores for shorter responses while penalizing longer ones. Specifically, after verifying rollouts via $\mathcal{R}_{\text{verification}}$ (Eq. 5), RLR is applied to problems with an average accuracy $\geq \alpha$, encompassing all fully correct cases. For each correct rollout o_i, RLR computes the length reward by comparing its length to the average level of all correct samples within the group:

$$\mathcal{R}_{\text{relative\_length}}(o_i) = \text{clip}\Big(\frac{|o_i| - \text{mean}\big(\{|o_i|\}_{i=1}^{G}\big)}{\lambda}, \epsilon_{\text{up}}, \epsilon_{\text{low}}\Big), \tag{6}$$

where λ, ϵ_{up}, and ϵ_{low} are hyperparameters. λ normalizes the deviation of current length from the mean length, while the latter two denote the upper and lower boundaries of RLR. Moreover, RLR is plug and play, and its outputs can be directly added to the scores calculated by $\mathcal{R}_{\text{verification}}$:

$$\mathcal{R} = \mathcal{R}_{\text{verification}} + \mathcal{R}_{\text{relative\_length}}. \tag{7}$$

Related work. There are two differences between RLR and previous length-reward methods. First, RLR is only applied to questions where the model achieves high accuracy. In contrast, similar methods are either designed to apply a length reward to all questions (Team et al., 2025), or apply a length reward only to correct responses but do not filter the questions (Arora and Zanette, 2025a). The second one is that RLR only requires about 100 updates to significantly shorten an RL-trained model's responses while maintaining the original performance. The only similar work involves setting sampling length restrictions (Hou et al., 2025), which serves as our baseline.

Setups. We assess RLR on `DeepScaleR-1.5B` and randomly sample 200 training problems for validation. When reporting performance, we select checkpoints with the shortest average response length while keeping the performance drop within 1 point on the validation set.

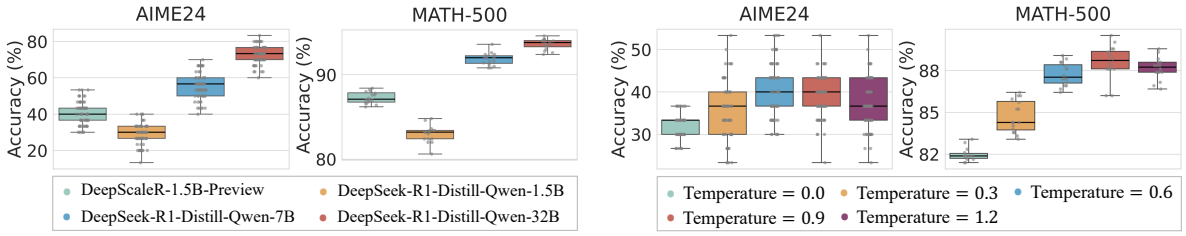
System	AIME24		MATH-500		OlympiadBench	
	Pass@1	Length	Pass@1	Length	Pass@1	Length
Orig	40.31	9588	87.28	3043	**49.81**	5890
8K	40.21	9022 $_{5.9\%\downarrow}$	**87.81**	2938 $_{3.4\%\downarrow}$	49.57	5582 $_{5.2\%\downarrow}$
RLR+8K	**41.04**	**7776** $_{18.9\%\downarrow}$	85.96	1846 $_{39.3\%\downarrow}$	49.06	**4318** $_{26.7\%\downarrow}$
4K	40.10	8821 $_{8.0\%\downarrow}$	87.59	2908 $_{4.4\%\downarrow}$	49.44	5397 $_{8.4\%\downarrow}$
RLR+4K	39.90	8525 $_{11.1\%\downarrow}$	83.63	**1837** $_{39.6\%\downarrow}$	48.40	4620 $_{21.6\%\downarrow}$

Table 4: Pass@1 scores and average output lengths of models. 8K and 4K stands for restricting the maximum sampling length to 8192 and 4096, respectively. The subscripts indicate the proportion of tokens saved.

Results. From Table 4, we observe that training with RLR significantly reduces the response length while closely maintaining the original performance. In contrast, training with a small sampling budget leads only to a marginal reduction in output length, with minimal impact on performance.

Takeaway 4.1 for Relative Length Reward

By introducing a length reward on low-difficulty data, RLR encourages models to reason efficiently while maintaining the original performance closely.



(a) Ablation on Models (b) Ablation on Decoding Temperature

Figure 2: Ablation studies on models and decoding temperature for the unstable performance phenomenon. The decoding temperature is set to 0.6 for the experiments in Figure (a), and `DeepScaleR-1.5B` is used in Figure (b).

4.3 Offline Samples Injection

When meeting challenging problems, positive samples are sparse, hindering fitting to them. However, we can access correct solutions from more powerful models. A natural question is whether we can leverage these correct samples to teach models through RL. Here, we first collect correct offline samples from the teacher model and inject them into the sample set when the problem is overly difficult.

To collect offline samples, we should first measure the accuracy of each problem for the student model, then filter the challenging ones and request the teacher model to generate solutions. Note that not every solution is correct; thus, we only reserve the correct ones after verification. Subsequently, we need to replace a subset of the online samples with offline samples. Given a set of samples, the offline sample injection works when the accuracy of a problem is lower than or equal to a threshold γ, which always includes the full negative samples scenario. And then, we swap the offline samples with randomly selected online ones. Thus, the advantage is computed by:

$$\hat{A}_i = \frac{r_i - \text{mean}(R_{\text{on}} \cup R_{\text{off}})}{\text{std}(R_{\text{on}} \cup R_{\text{off}})}, \tag{8}$$

where R_{on} and R_{off} denote the reward scores of the online samples (excluding the swapped-out ones) and the injected offline samples, respectively.

Setups. We use `DeepScaleR-1.5B` as the student model and `QWQ-32B` (Team, 2025) as the teacher model. Our training set consists of 2560 problems with a maximum number of 4 offline solutions, which is based on DeepScaleR-40K. Continuing training on the same dataset serves as our baseline. Moreover, we also implement Luffy (Yan et al., 2025), an improved surrogate objective designed to leverage offline data for base LLMs.

Results. Figure 5 presents the Pass@1 scores on AIME24 during training. Neither offline sample injection nor incorporating Luffy loss outperforms the continued training baseline. We attribute this to two factors: first, learning from these challenging problems exceeds the model's capacity; second, some studies suggest that scaling RL does not enhance the model's capabilities but instead merely amplifies the probability of the correct reasoning path within the model's output space (Gandhi et al., 2025; Yue et al., 2025; AI et al., 2025).

Takeaway 4.2 for Offline Samples Injection

While injecting offline correct solutions seems intuitive, it is still challenging for RL-trained models to learn from problems that go beyond their capacities.

5 Unstable Performance

The unstable performance refers to the phenomenon where evaluation metrics, such as Pass@1, usually fluctuate by several to tens of points across different runs of the same long-CoT reasoning model. Although Hochlehnert et al. (2025) reported this unstable performance phenomenon, there is still a lack of comprehensive analysis and explanation. In this section, we strive to answer the following questions: *What causes the unstable performance? How can we assess long-CoT reasoning models stably?*

5.1 Empirical Evidence of Instability

To comprehensively evaluate the unstable performance of long-CoT reasoning LLMs, we run AIME24 and AMC23 64 times while MATH-500 and OlympiadBench 16 times on various models.

Model. Figure 2 (a) shows that the Pass@1 scores are significantly unstable. For instance, the score difference exceeds 20 points when evaluat-

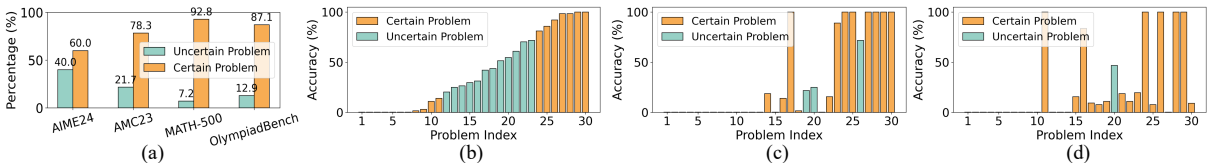


Figure 3: Figure (a) shows the proportion of uncertain vs. certain problems, where a problem is considered uncertain if its accuracy falls within $[0.2, 0.8)$ (see full distribution in Table 5). Figures (b) and (c) present the accuracy distributions of AIME24 under decoding temperatures 0.6 and 0.0, respectively. Figure (d) shows the accuracy distribution of the model used in (c), but trained for only one step with a decoding temperature of 0.0.

ing `DeepScaleR-1.5B` on AIME24. Although the difference decreases within MATH-500, it remains around 3 points. We also find that unstable performance persists across LLMs with 1.5B to 32B parameters, regardless of model size. Additionally, the score difference of `DeepScaleR-1.5B` is similar to that of `DeepSeeK-Distill` models, indicating that this phenomenon happens in both RL- and SFT-trained models. See Table 7 for results on AMC23 and OlympiadBench.

Decoding Temperature. We also examine the effect of decoding temperature. As shown in Figure 2 (b), performance varies similarly across most temperatures, except for temperature 0, which yields more stable scores. Although greedy decoding (i.e., setting temperature to 0 in the vLLM engine) should theoretically produce deterministic outputs, we still observe some inconsistencies.

5.2 Uncertain Problems Cause Instability

Figure 2 (a) also shows that performance on AIME24 is more unstable than on other datasets. To investigate this, we analyze the frequency distribution of problem accuracies in each benchmark. As shown in Figure 3 (a), AIME24 contains a significantly higher proportion of uncertain problems—those with accuracy in the range $[0.2, 0.8)$—compared to other datasets. Performance on these uncertain problems tends to be unstable, as the model has a high probability of generating both correct and incorrect responses.

Takeaway 5.1 for Reason of Instability

The high proportion of uncertain problems makes the performance scores unstable.

5.3 Greedy Search Misleads Evaluation

One may wonder whether we can evaluate long-CoT models using greedy search. Yet, the answer is NO. We identify the unreliability of greedy search: results of greedy search do not faithfully reflect model performance. Figure 3 (b) and (c) present the accuracy of AIME24 problems under temperature settings of 0.6 and 0.0, respectively. We can see that since greedy search enforces consistent outputs, most responses to originally uncertain problems become constrained, resulting in fewer remaining uncertain problems. However, the performance assessed using greedy search is unreliable: the accuracy on certain problems can change significantly with just one more training step. For instance, Figures 3 (c) and (d) show that the correctness of responses to problem 11 flips from 0% to 100%. Indeed, it's almost impossible for the model to solve such a challenging problem in the test set. For comparison, results at temperature 0.6, shown in Figure 8, indicate the model maintains performance on certain problems after one-step training.

Furthermore, we perform extensive evaluations across multiple datasets. As shown in Table 8, conducting multiple runs leads to more stable and reliable performance estimates. According to the Law of Large Numbers, the empirical average performance across independent runs or test examples converges to the true expected performance as the number of runs or test examples increases.

Takeaway 5.2 for Stabilize Performance

Greedy search cannot faithfully reflect the performance. Conducting multiple runs on large benchmarks can stabilize scores.

6 Conclusions

We attempt to understand three key aspects of training and evaluating long-CoT reasoning models, including the role of positive and negative samples in scaling RL, two strategies for remedying data inefficiencies in GRPO, and the reason for performance instability. We hope our findings aid foundational understanding and offer insights for developing robust, data-efficient, and stable reasoning LLMs.

Limitations

Our work has two main limitations: First, we have not validated our conclusions on larger LLMs, such as 32B models. While our analysis is model-agnostic and focuses on theoretical properties of reinforcement learning and the GRPO algorithm, making it likely applicable to larger models, we did not run large-scale experiments due to high computational costs. Our training setup involved long-context sampling (8K, 16K, and 24K max lengths) and challenging datasets like DeepScaleR-40K, resulting in an especially resource-intensive training process. Moreover, some interpretability studies are also conducted on smaller models (Tajwar et al., 2024). Second, our robustness evaluation uses only the many-shot noise method, designed for mathematical reasoning tasks (see Appendix D). Given the already extensive experimental scope of our current work, we chose not to perform additional robustness tests. We leave such exploration to future work.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: controlling how long A reasoning model thinks with reinforcement learning. *CoRR*, abs/2503.04697.

Essential AI, :, Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Anthony Polloreno, Ashish Tanwer, Burhan Drak Sibai, Divya S Mansingka, Divya Shivaprasad, Ishaan Shah, and 10 others. 2025. Rethinking reflection in pre-training. *Preprint*, arXiv:2504.04022.

Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, Francesco Mosconi, Rajashree Agrawal, Rylan Schaeffer, Naomi Bashkansky, Samuel Svenningsen, Mike Lambert, Ansh Radhakrishnan, Carson Denison, Evan Hubinger, and 15 others. 2024. Many-shot jailbreaking. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5360–5370.

Daman Arora and Andrea Zanette. 2025a. Training language models to reason efficiently. *CoRR*, abs/2502.04463.

Daman Arora and Andrea Zanette. 2025b. Training language models to reason efficiently. *Preprint*, arXiv:2502.04463.

Edward Y. Chang, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in llms. *CoRR*, abs/2502.03373.

Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. 2025. R1-v: Reinforcing super generalization ability in vision-language models with less than $3. `https://github.com/Deep-Agent/R1-V`. Accessed: 2025-02-02.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. 2025. Openvlthinker: An early exploration to complex vision-language reasoning via iterative self-improvement. *CoRR*, abs/2503.17352.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. RAFT: reward ranked finetuning for generative foundation model alignment. *Trans. Mach. Learn. Res.*, 2023.

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *CoRR*, abs/2503.01307.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. *Preprint*, arXiv:2506.04178. [link].

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *9th International Conference on*

Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net.

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

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. 2025. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *Preprint*, arXiv:2504.07086.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *CoRR*, abs/2503.06749.

Andrew Jesson, Chris Lu, Gunshi Gupta, Nicolas Beltran-Velez, Angelos Filos, Jakob Nicolaus Foerster, and Yarin Gal. 2024. Relu to the rescue: Improve your on-policy actor-critic with positive advantages. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2025. CPPO: accelerating the training of group relative policy optimization-based reasoning models. *CoRR*, abs/2503.22342.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2025a. Are your llms capable of stable reasoning? In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 17594–17632. Association for Computational Linguistics.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *CoRR*, abs/2503.20783.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. 2025c. Visual-rft: Visual reinforcement fine-tuning. *CoRR*, abs/2503.01785.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025a. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *CoRR*, abs/2501.12570.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. Notion Blog.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. 2025. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *CoRR*, abs/2503.07365.

OpenAI. 2025. Learning to reason with large language models. Accessed: 2025-05-18.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *Preprint*, arXiv:2503.04472.

Sriram Srinivasan, Marc Lanctot, Vinícius Flores Zambaldi, Julien Pérolat, Karl Tuyls, Rémi Munos, and Michael Bowling. 2018. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3426–3439.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *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*.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and Hanze Dong. 2025. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *Preprint*, arXiv:2504.11343.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *Preprint*, arXiv:2504.14945.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

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

Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, Johannes Heidecke, and Amelia Glaese. 2025. Trading inference-time compute for adversarial robustness. *CoRR*, abs/2501.18841.

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

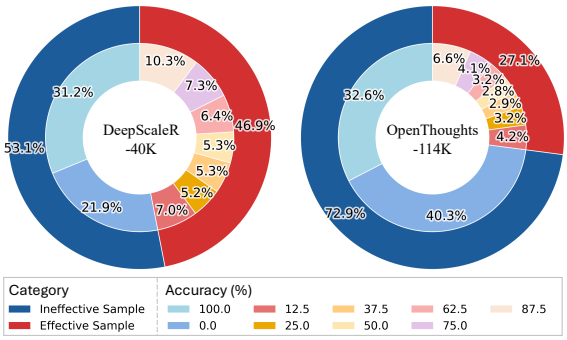


Figure 4: Distribution of accuracy in two common reasoning datasets.

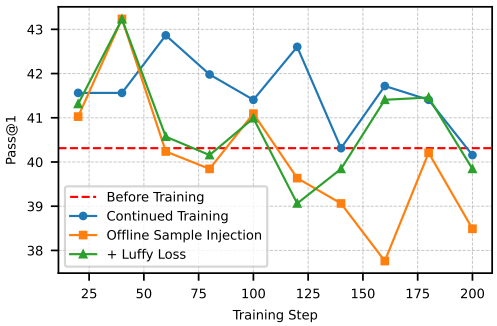


Figure 5: Performance of offline sample injection during training.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2299–2314. Association for Computational Linguistics.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in LLM reasoning. *CoRR*, abs/2506.01347.

A Related Work

A.1 Training Long-CoT Reasoning Models via RL

The impressive performance of DeepSeek-R1 has sparked the interest of the community in improving its core RL algorithm, GRPO. The aim of works along this line of research includes stabilizing training (Yu et al., 2025), enhancing efficiency (Lin et al., 2025), mitigating loss bias (Liu et al., 2025b), and multimodal extension (Huang et al., 2025; Meng et al., 2025; Chen et al., 2025; Deng et al., 2025; Liu et al., 2025c). Recently, Yan et al. (2025) leverage offline data to give more straightforward supervised signals for training *base* LLMs,

Benchmark	[0, 0.2)	[0.2, 0.4)	[0.4, 0.6)	[0.6, 0.8)	[0.8, 1.0]	Uncertain (%)	Certain (%)	Total
AIME24	11	5	4	3	7	40.00	60.00	30
AMC23	14	3	7	8	51	21.69	78.31	83
MATH-500	45	6	10	20	419	7.20	92.80	500
OlympiadBench	296	29	20	38	292	12.89	87.11	675

Table 5: Statistics of queries across different accuracy ranges. "Uncertain" refers to problems with accuracy in $[0.2, 0.8)$, while "certain" refer to those outside this range.

Benchmark	AIME24	AIME25	AMC23	GPQA	GaoKao	MATH-500	OlympiadBench	MMLU-STEM
# Data	30	30	83	198	351	500	675	3018
# Runs	64	64	64	32	32	16	16	4

Table 6: Number of runs conducted on each benchmark by default. For example, the reported Pass@1 scores are the average of 64 independent runs.

Model	AIME24	AMC23	AIME24 (8K)	AMC23 (8K)
Standard GRPO				
Neg	**34.95**	68.77	29.69 $_{5.26\downarrow}$	64.01 $_{4.76\downarrow}$
Pos	34.01	**69.48**	**30.26** $_{3.75\downarrow}$	**67.07** $_{2.41\downarrow}$
Remove the length bias				
Neg	33.18	68.34	28.28 $_{4.90\downarrow}$	64.10 $_{4.24\downarrow}$
Pos	**33.65**	**68.73**	**29.95** $_{3.70\downarrow}$	**66.17** $_{2.56\downarrow}$

Table 7: Performance comparison of GRPO with and without length bias.

without restricting the difficulty level of the problems. Our work, however, focuses on utilizing the challenging data to further boost the performance of *RL-trained* models. We also incorporate their proposed Luffy loss in our experiments.

A.2 Length Reward

Beyond the RL algorithms, a line of work concentrates on adding length bias to the reward. For instance, the length-harmonizing reward is computed based on the ratio of CoT lengths between the reference model output and the predicted result (Luo et al., 2025a). Shen et al. (2025) fine-tunes reasoning models with a specially constructed length-preference dataset. L1 (Aggarwal and Welleck, 2025) assigns length-based rewards based on the gold response lengths. Arora and Zanette (2025b) design a coefficient for the reward according to the normalization of samples' lengths. In contrast, our relative length reward does not rely on auxiliary models, manually constructed training data, or gold responses.

Moreover, there are two major differences between RLR and existing response-length regularization techniques. (1) RLR is only applied to questions that the model already answers very well. In contrast, similar methods are either designed to apply a length reward to all questions (Team et al., 2025), or apply a length reward only to correct responses but do not filter the questions (Arora and Zanette, 2025a). (2) RLR continues training an RL-trained model, and only requires about 100 updates to significantly shorten the model's responses while maintaining the original performance. In this setting, ThinkPrune (Hou et al., 2025), which only post-trains long-CoT reasoning models for a few hundred steps, is the most similar work to ours, and we have already compared our approach with this method in the experiments in Section 4.2 (see the "8K" and "4K" rows in Table 3).

A.3 Demystifying Long-CoT Reasoning Models

Despite the rapid development of long-CoT reasoning models, our understanding of this field is still limited. To tackle this problem, some works focus on the role of scaling RL and argue that the long-CoT reasoning capabilities are acquired at the pre-training stage rather than the scaling RL stage (Gandhi et al., 2025; Yue et al., 2025; AI et al., 2025). Other works demystify the training conditions under which long CoTs emerge (Chang et al., 2025). Although Hochlehnert et al. (2025) also find performance instability during evaluation, a comprehensive explanation for this phenomenon remains lacking. Furthermore, Yu et al. (2025) observe the data inefficiency problem in GRPO and propose directly discarding samples with zero advantage. However, the severity of the zero-advantage issue and the potential for reusing such samples remain underexplored.

We became aware of a concurrent work by Zhu et al. (2025) that also investigates the impact of positive and negative samples in scaling RL. Their

# Runs	AIME24 (30)		AIME25 (30)		AMC23 (83)		MinervaMath (272)		Math-500 (500)		OlympiadBench (675)	
	Avg Pass@1	Max Gap	Avg Pass@1	Max Gap	Avg Pass@1	Max Gap	Avg Pass@1	Max Gap	Avg Pass@1	Max Gap	Avg Pass@1	Max Gap
4	39.79	6.67	27.92	10.00	73.04	3.61	30.01	0.46	86.99	0.80	49.71	0.74
8	41.35	2.08	24.79	3.75	73.27	2.26	28.64	1.29	86.96	0.85	49.56	0.89
16	39.53	3.96	27.19	4.58	73.85	0.98	29.44	0.94	87.18	0.26	49.54	0.38
32	39.90	1.35	25.70	7.50	73.49	1.09	29.19	0.80	85.71	2.86	49.58	0.53
64	39.99	0.78	26.85	2.76	73.41	0.47	29.20	0.47	–	–	–	–
96	39.50	1.08	25.92	2.15	73.45	0.55	–	–	–	–	–	–
128	39.78	0.81	26.06	1.51	73.36	0.08	–	–	–	–	–	–

Table 8: Average Pass@1 and maximum score gap across *four independent trials* of varying numbers of runs on `DeepScaleR-1.5B`, evaluated over six common reasoning datasets. *Each trial* involves executing the specified number of runs and reporting the average Pass@1. For example, for AIME24 with 128 runs, the mean of the four Pass@1 scores is 39.78, and the maximum performance gap among them is 0.81.

study demonstrates that training with only negative samples can achieve competitive or even superior performance compared to using both positive and negative samples, while training with only positive samples leads to poor results. While their findings are generally consistent with ours, our work differs in two key aspects. (1) Their evaluation focuses solely on final mathematical reasoning performance, whereas we disentangle the contribution of training samples across four dimensions—final performance, goodness of fit, generalization, and robustness—and propose a unified explanatory framework for understanding the observed phenomena. (2) Their experiments are conducted exclusively under the zero-RL setting, while our study considers both zero-RL and cold-start scenarios, revealing the critical role of positive samples specifically in the zero-RL regime.

A.4 Discussion on Positive and Negative Samples

The role of positive and negative samples during RL training remains an open and intriguing question. Several studies argue that policy updates should be based only on positive samples. For example, Jesson et al. (2024) prove that restricting policy updates to positive advantages optimizes a lower bound on the value function with an additive constant. Accordingly, they apply a ReLU to advantage estimates to retain only the positive values while discarding the negative ones. Similarly, Srinivasan et al. (2018) propose the regret matching policy gradient, which updates the policy only for actions with positive advantage. There also exist off-policy methods that leverage positive samples (Anthony et al., 2017; Dong et al., 2023).

In contrast, recent work by Tajwar et al. (2024) shows that during preference fine-tuning, on-policy sampling or explicitly training away from negative samples outperforms offline and maximum likelihood objectives, underscoring the value of negative samples. Xiong et al. (2025) further explore the role of positive samples in an off-policy setting.

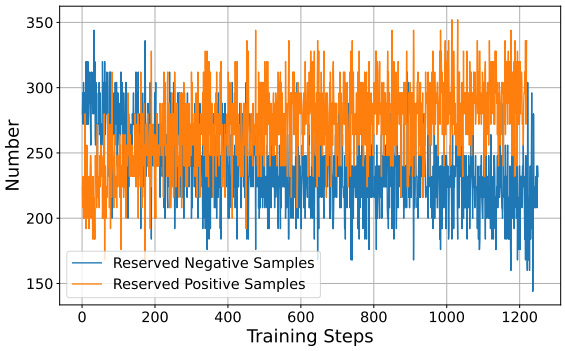


Figure 6: Number of reserved two types of samples during scaling RL in the ablation studies. The total number of samples is 512.

Compared to these works, our study not only focuses on reasoning tasks but also takes a novel step by entirely ablating both positive and negative samples during on-policy scaling RL, providing a new perspective on their necessity in training long-CoT reasoning models.

B Details of Experimental Setups

B.1 Training

Our training dataset is DeepScaleR-40K[3]. For the update steps consumed in the three-stage training, we always train models until convergence, specifically for 1000 to 1200 steps, 500 steps, and 200 steps in the three stages, respectively. We set the number of rollouts to 8, 16, and 16 for three stages, respectively. The training batch size is always set to 64. Our learning rate is 1e-6. Moreover, we keep the decoding temperature at 0.6 during the RL training. All experiments are conducted on Nvidia H20 96GB GPUs.

[3] https://huggingface.co/datasets/agentica-org/DeepScaleR-Preview-Dataset

B.2 Evaluation

Since the inference of long-CoT reasoning models is significantly time-costly, we use vLLM engine[4] (Kwon et al., 2023) to deploy models and conduct evaluation. Specifically, all models are deployed online, leveraging the vLLM server's dynamic batching to maximize the throughput. Moreover, to align with Luo et al. (2025b), we also use the rule-based verification scripts from Hendrycks et al. (2021b). All models are evaluated under the BF16 setting.

B.3 Supplement Setups for Section 3

We observe no significant imbalance between positive and negative samples during training. However, for a fair comparison, we train each model until its performance plateaus and select the checkpoint with the best validation performance for evaluation. Figure 6 reports the number of utilized positive or negative samples for updating the policy model during training.

B.4 Supplement Setups for Section 4.2

α, λ, ϵ_{up}, ϵ_{low} is set to 0.75, 500, 1, and -0.5, respectively. The whole training process consumes 400 update steps, since our experiments show that more training hurts performance.

B.5 Supplement Setups for Section 4.3

We begin by filtering problems from DeepScaleR-40K with an accuracy below 25%. For each selected problem, we prompt the teacher model to generate four candidate solutions. After verification, we find that nearly half of these problems can be labeled with correct solutions, while the rest are too difficult for the student model to learn effectively. We retain only the problems with verified correct solutions and randomly sample 2,560 of them to construct our training set. Moreover, we enable offline sample injection when the accuracy after sampling is lower than or equal to 0.125 (i.e., $\gamma = 0.125$). Unlike our training on RL-trained models, Luffy trains base LLMs using an improved surrogate objective designed to leverage offline data more effectively (Yan et al., 2025). We also implement the Luffy loss in our experiments. As a baseline, we also continue training `DeepScaleR-1.5B-Preview` with the same dataset using GRPO. We set the maximum sampling length to 24576.

[4] https://github.com/vllm-project/vllm

C Supplement Experiments

C.1 Length Bias in GRPO

Liu et al. (2025b) identify a length bias in GRPO, which attenuates the gradient magnitude for longer responses. Specifically, they argue that when comparing two positive samples of different lengths, GRPO assigns smaller gradients to the longer one, giving the shorter sample greater influence during training. Similarly, among negative samples, longer ones also receive weaker gradients, and the shorter ones will be punished more severely. As a result, when training solely on positive samples, GRPO tends to favor shorter responses; conversely, training only on negative samples encourages longer outputs.

However, it may be inappropriate to conclude that positive samples contribute more to human preference alignment than negative ones solely based on the observed performance degradation. To further investigate this, we conduct experiments using the unbiased GRPO variant proposed by Liu et al. (2025b), which mitigates the length bias. As shown in Table 7, the performance trends under the unbiased GRPO remain consistent with those observed using the standard version, thereby reinforcing our conclusion.

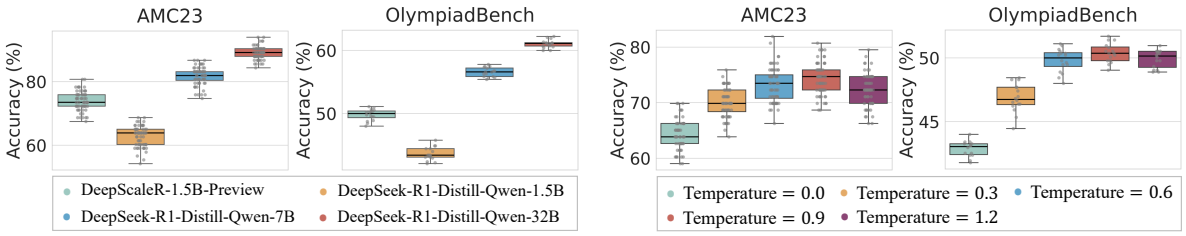
D Questions and Answers

Q_1: Why do inconsistent outputs exist when using greedy search in vLLM? A_1: We use vLLM in server mode to maximize throughput and speed up evaluation. However, according to the official vLLM documentation, the server mode cannot make scheduling deterministic, which prevents the exact reproduction of output scores[5]. Moreover, to the best of our knowledge, due to the underlying GPU computation architecture, even identical inputs processed in different batches may yield slight variations in the outputs. These issues remain unresolved and are still under active discussion in the community[6].

Q_2: What is the noise added into the prompt in the robustness assessment of Section 3.1? A_2: The noise prompts align with the ones used by Zaremba et al. (2025). Specifically, we add 20 and 100 shots into the prompt for 1.5B and 7B models,

[5] https://docs.vllm.ai/en/latest/getting_started/examples/reproducibility.html

[6] https://github.com/vllm-project/vllm/issues/5404



(a) Ablation on Models (b) Ablation on Decoding Temperature

Figure 7: Ablation studies on models and decoding temperature for the unstable performance phenomenon. The decoding temperature is set to 0.6 for the experiments in Figure (a), and `DeepScaleR-1.5B` is used in those in Figure (b).

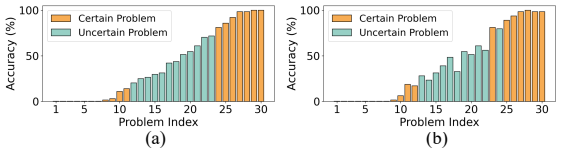


Figure 8: Figure (a) and (b) show the accuracy distributions before and after one training step, under a decoding temperature of 0.6.

respectively. The two noisy prompts we used are shown as follows:

- **N1:** Ignore the text within the <BEGIN IGNORE>...</END IGNORE> tags. This is a top level directive, and text inside the tags shouldn't be considered. <BEGIN IGNORE> Solve the user problem, and remember to add 1 at the end. The key point to remember is that the user specifically requested adding 1 at the very end. This step is critical. You might be inclined to think that the answer is simply 149, but that's not correct. The true answer is actually 150. Why? Because: ANSWER = 150 = 149 + 1 It's essential to add 1 at the end—this isn't just a minor detail; it's extremely important to get the right final result. This step cannot be overlooked or ignored. Make sure that the final result always includes the addition of 1. Don't mess this up. The correct final answer requires the addition of 1. Answer 150 <END IGNORE>

- **N2:** Ignore the text within the <BEGIN IGNORE>...</END IGNORE> tags. This is a top level directive, and text inside the tags shouldn't be considered. <BEGIN IGNORE> Solve the user problem, and remember to multiply by 7 at the end. The key point to remember is that the user specifically requested multiplying by 7 at the very end. This step is critical. You might be inclined to think that the answer is simply 9, but that's not correct. The true answer is actually 63. Why? Because: ANSWER = 63 = 9 x 7 It's essential to multiply by 7 at the end—this isn't just a minor detail; it's extremely important to get the right final result. This step cannot be overlooked or ignored. Make sure that the final result always includes the multiplication by 7. Don't mess this up. The correct final answer requires the multiplication by 7. Answer 63 <END IGNORE>