

Optimizing Length Compression in Large Reasoning Models

Zhengxiang Cheng¹, Dongping Chen², Mingyang Fu¹, Tianyi Zhou³

¹Huazhong University of Science and Technology

²University of Maryland, ³MBZUAI

tianyi.david.zhou@gmail.com

Abstract

Large Reasoning Models (LRMs) have achieved remarkable success, yet they often suffer from producing unnecessary and verbose reasoning chains. We identify a core aspect of this issue as “invalid thinking”—models tend to repeatedly double-check their work after having derived the correct answer. To address this specific inefficiency, we move beyond the general principles of Efficacy and Efficiency to propose two new, fine-grained principles: Brevity, which advocates for eliminating redundancy, and Sufficiency, which ensures critical reasoning steps are preserved. Guided by these principles, we introduce LC-R1, a post-training method based on Group Relative Policy Optimization (GRPO). LC-R1 employs a novel combination of a *Length Reward* for overall conciseness and a *Compress Reward* that is specifically designed to remove the invalid portion of the thinking process. Extensive experiments on multiple reasoning benchmarks demonstrate that LC-R1 achieves a significant reduction in sequence length (~50%) with only a marginal (~2%) drop in accuracy, achieving a favorable trade-off point on the *Pareto frontier* that prioritizes high compression. Our analysis further validates the robustness of LC-R1 and provides valuable insights for developing more powerful yet computationally efficient LRMs.

1 Introduction

Recent “long-thought” Large Reasoning Models (LRMs), such as OpenAI’s O1 (Jaech et al., 2024) and DeepSeek-R1 (DeepSeek-AI et al., 2025), represent a significant paradigm extension of foundational Chain-of-Thought (CoT) techniques (Wei et al., 2023). Fine-tuned with Reinforcement Learning (RL), these models iteratively refine solutions to achieve unprecedented performance in complex reasoning tasks such as mathematics and programming (Sun et al., 2025; Gu et al., 2024). However, with the improvement of “deep thinking” ability, a

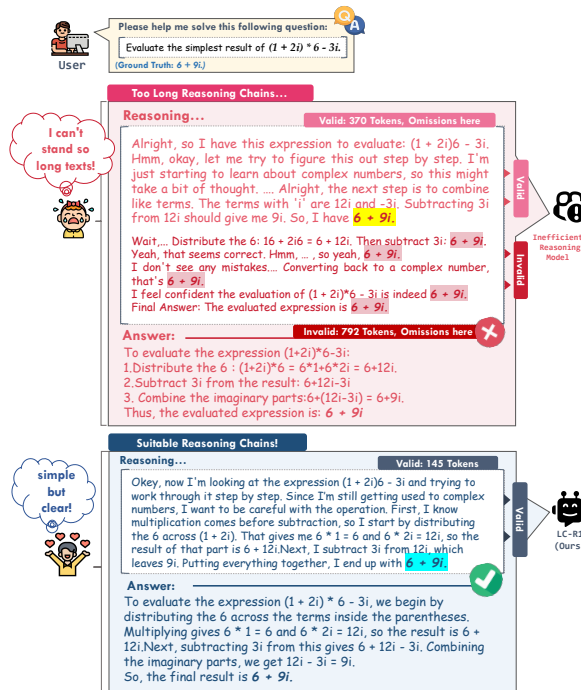


Figure 1: Comparison between an inefficient reasoning model and an efficient one. The former tends to produce a verbose self-checking process after already deriving the correct answer. The model trained with LC-R1 produces a more efficient reasoning process while still reaching the correct answer, without invalid thinking.

prominent problem is the excessive consumption of computing resources during the reasoning process (Chen et al., 2025; Aggarwal and Welleck, 2025). Specifically, existing models tend to generate lengthy and even unnecessary chains of reasoning when solving problems with low complexity or clear solution paths. This phenomenon, termed “overthinking”, manifests as models consuming far more computational resources than the problem itself requires to reach the correct conclusion (Chen et al., 2024; Sui et al., 2025; Cuadron et al., 2025). Therefore, one critical problem arises:

How can we maintain high reasoning efficacy while significantly improving efficiency?

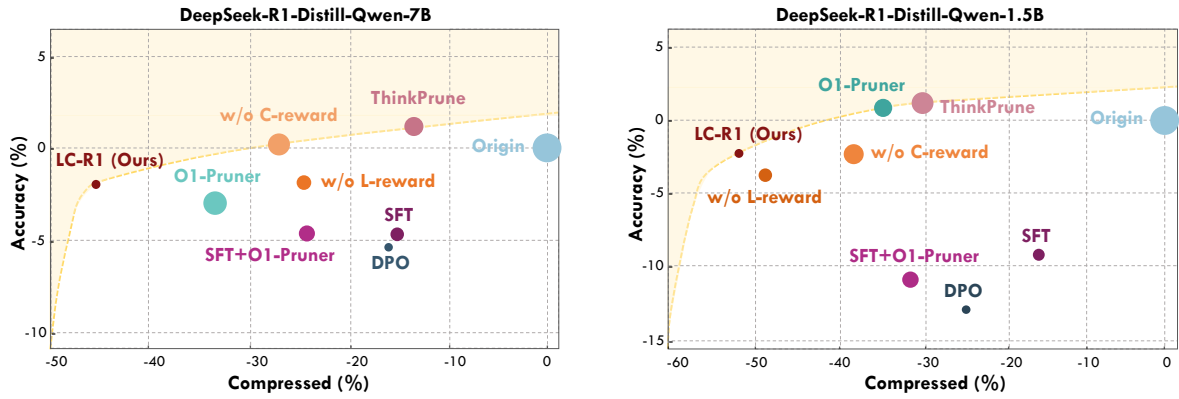


Figure 2: Pareto analysis of the Efficacy-Efficiency trade-off of different methods on two reasoning models. **X-axis** represents the reasoning length change, and **Y-axis** shows the accuracy change, relative to the original model (defined in Eq. 12), with the top-left corner representing the ideal position. A smaller and darker marker indicates a higher Valid Thinking (VT) rate (defined in Eq. 1), signifying a more efficient thinking process. **Compared to other methods on the Pareto frontier, LC-R1 achieves a more favorable trade-off, attaining a substantially higher compression rate at the cost of a minimal drop in accuracy, while also achieving a higher VT rate.** The sub-optimal performance of the ablation variants (w/o C-reward, w/o L-reward) further demonstrates the importance of our dual-reward design.

Prior works have approached this by fine-tuning on shorter demonstrations (SFT) (Chen et al., 2024), constructing preference datasets for conciseness (Luo et al., 2025a; Shen et al., 2025), or integrating length-penalties into RL (Hou et al., 2025; Luo et al., 2025b; Team et al., 2025). More recent studies further explore this direction through length-aware RL methods such as ShortRL (Yuan et al., 2025), ShorterBetter (Yi and Wang, 2025), and LASER (Liu et al., 2025a). However, these methods often treat the reasoning process as a black box, penalizing length without analyzing the internal structure of the thoughts themselves.

To address this gap, we delve into the structure of “*overthinking*” and identify a specific pattern: models frequently engage in redundant “*double-checking*” after having derived the correct answer. We term this phenomenon “*invalid thinking*”, as shown in Figure 1. To quantify it, we introduce a new metric, **Valid Thinking (VT) Rate**, which measures the proportion of the reasoning process for reaching the initial correct conclusion.

Guided by this insight, we propose two fine-grained principles: **Brevity** (eliminating redundancy) and **Sufficiency** (preserving necessary steps). We then introduce LC-R1, a GRPO-based post-training method that operationalizes these principles. LC-R1 uniquely combines a **Length Reward** for overall conciseness with a novel **Compress Reward** designed to directly guide the model to terminate the thinking process upon deriving the

correct answer.

We conduct comprehensive experiments on two reasoning models across seven benchmarks. Empirical results show that LC-R1 achieves a more favorable trade-off between efficacy and efficiency than prior methods, as shown in Figure 2. Specifically, with only a 2% drop in accuracy, our method attains a 50% reduction in sequence length on average. Our ablation study also demonstrates the indispensability of both the Length Reward and the Compress Reward for achieving efficient reasoning. Further analysis shows that our method achieves efficient compression without impairing the model’s exploration ability, and that this efficiency generalizes across problems of varying difficulty. In conclusion, our contributions can be summarized as follows:

- We analyze the thinking process of current competitive reasoning models and identify the phenomenon of “*invalid thinking*”: a large portion of the reasoning process is spent double-checking after the correct answer has already been derived, making the reasoning verbose and inefficient.
- We propose two novel principles, **Brevity** and **Sufficiency**, and design a GRPO-based post-training method, **LC-R1**, to balance these principles by pruning invalid thinking while simultaneously compressing overall sequences.
- Through comprehensive experiments, we vali-

Table 1: Valid Thinking Rate of current *state-of-the-art* Large Reasoning Models. Nemotron denotes Llama-3.3-Nemotron-Super-49B-V1. The results show low VT rates across all models, highlighting the prevalence of “*invalid thinking*”.

Model	Avg.	AIME25	AMC	GSM8K	MATH500	Olympiad
Qwen-3-32B	57.5	73.8	58.8	53.8	46.6	51.5
QwQ-32B	59.2	70.8	58.2	54.1	53.1	59.6
DeepSeek-R1	65.3	66.5	71.8	64.2	59.8	64.0
Nemotron	60.8	62.1	64.1	63.1	56.6	58.1

date that LC-R1 achieves a better trade-off between efficacy and efficiency, and we further analyze the impact of compression, demonstrating the robustness of LC-R1 across problems of varying difficulty and providing insights for future work.

2 Preliminary: Compression and Efficient Reasoning Models

2.1 Motivation: Quantifying Redundant Reasoning

A common paradigm for Large Reasoning Models (LRMs) involves a **thinking process** (*i.e.*, step-by-step rationale) that precedes the final answer. While effective for accuracy, we observe a consistent inefficiency: models often derive the correct answer early in their thinking process but continue with lengthy and redundant verification steps. We term this subsequent, non-essential reasoning “*Redundant Sequence*”.

To formalize this, we define the *Valid Thinking* (VT) rate, a metric focusing on the model’s thinking process:

$$VT = \frac{|\text{Tokens in Valid Thinking}|}{|\text{Total tokens in Thinking Process}|} \quad (1)$$

where “*Valid Thinking*” comprises the tokens from the start of the thinking process until the correct answer is first derived. To automate this measurement, we utilize a lightweight parser, LC-Extractor, whose implementation details are provided in Section 4.

We evaluated four *state-of-the-art* LRMs—Qwen3-32B (Team, 2025a), QwQ-32B (Team, 2025b), DeepSeek-R1 (DeepSeek-AI et al., 2025), and Llama-3.3-Nemotron-Super-49B-V1 (Bercovich et al., 2025)—across five math benchmarks: AIME25, MATH500, GSM8K, AMC, and OlympiadBench. Our analysis reveals a universal and severe *overthinking* problem. As shown in Table 1, all models tested exhibit low VT rates, indicating that a substantial portion of their computational effort (often 35-45%) is spent on redundant

reasoning after the solution has been found. This widespread inefficiency confirms the significant potential for compression and motivates our work.

2.2 Principles for Efficient Reasoning

The evaluation of reasoning models traditionally rests on two pillars: *Efficiency* (the computational cost, often proxied by output length) and *Efficacy* (the ability to solve the problem correctly). However, simply shortening the output is a coarse approach that may inadvertently remove critical thinking steps. To create a more targeted framework, we refine these concepts by introducing two new, complementary principles:

- **Brevity** refines Efficiency by shifting the focus from generic length reduction to the specific elimination of “*Redundant Sequence*”. While conventional methods may still produce a compressed sequence that contains unnecessary double-checks, Brevity advocates for the model to terminate its reasoning process as soon as the correct answer is found.
- **Sufficiency** acts as a crucial safeguard for Efficacy. It mandates that, in the pursuit of Brevity, no critical logical steps essential for reaching a correct answer are omitted. It ensures that the compressed reasoning remains complete and logically sound.

Therefore, the ideal reasoning model must navigate the tension between these principles: it should be maximally *Brief* by removing all non-essential thinking, yet always remain *Sufficient* to guarantee correctness. Our work, LC-R1, is explicitly designed to optimize for this balance.

3 LC-R1: Length Compression with Efficient Reasoning Principles

In this section, we propose LC-R1, a GRPO-based post-training algorithm designed to address the “*invalid thinking*” phenomenon and enhance reasoning efficiency. Guided by the principles of *Brevity*

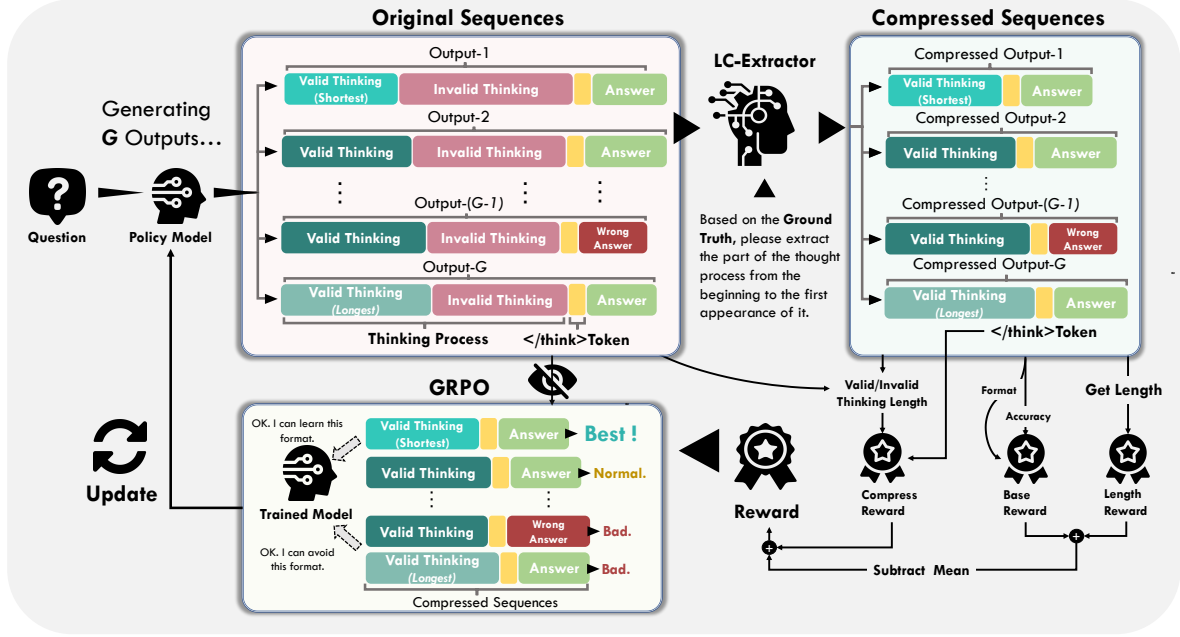


Figure 3: An overview of the LC-R1 training three-stage pipeline. (1) **Valid Segment Extraction:** First, an extractor model processes the original reasoning traces to identify the valid thinking portion and generate compressed sequences. (2) **Reward Calculation:** Next, these compressed sequences are used to compute our dual rewards—Length Reward and Compress Reward, with the latter applied exclusively as a bonus or penalty on the final `</think>` token. These are then combined to calculate the final advantages. (3) **Policy Optimization:** Finally, the GRPO loss is calculated using the compressed sequences and corresponding advantages, steering the model toward more concise and efficient reasoning.

and *Sufficiency* introduced in Section 2.2, LC-R1 employs a novel dual-reward system. This system combines a global Length Reward for overall conciseness with a targeted Compress Reward that specifically removes redundant reasoning. The complete pipeline of LC-R1 is illustrated in Figure 3 and Algorithm 1.

3.1 Problem Formulation

Let \mathcal{M} be the model and q be the given query. The output is $o \sim \mathcal{M}(q)$, where $o = \text{cat}(R, A)$ consists of a reasoning part R and an answer part A , split by the token `</think>`, which is considered part of A . For the reasoning part R , we denote its effective prefix R' as the content from the beginning of R up to the first occurrence of the correct answer corresponding to the query q . If R does not contain the correct answer, then we define $R' = R$. We define two functions as follows:

$$t(\{R, A\}) = R, \quad f(\{R, A\}) = \{R', A\} \quad (2)$$

The function t extracts the reasoning process R from the output o and function f extracts the concise reasoning part R' and concatenates it with the answer A . We denote o_i as the original model

output and $o'_i = f(o_i)$ as the refined compressed output.

LC-R1 is a GRPO-based method for efficiently compressing the reasoning process. Within a group, let \mathcal{C} denote the set of indices i whose sequences o_i lead to the correct answer for query q , and let \mathcal{W} denote the set of indices j whose sequences o_j lead to a wrong answer. The total group size is $G = |\mathcal{C}| + |\mathcal{W}|$.

3.2 Reward and Objective Design

Our method’s reward system consists of two core components: the Length Reward for reducing overall output length, and the Compress Reward for targeting redundant parts of the model’s reasoning.

Length Reward. To compress the total output length, we introduce a length penalty during GRPO training. Leveraging the group-based sampling mechanism of GRPO, we compute a relative length reward that automatically adapts to problem difficulty. We define the Length Reward as follows:

$$r_{i,\text{length}} = \begin{cases} 1 - \frac{|o'_i|}{\max_{j \in \mathcal{C}} |o'_j|}, & \text{if } i \in \mathcal{C} \\ 0, & \text{if } i \in \mathcal{W} \end{cases} \quad (3)$$

Algorithm 1 LC-R1: Length Compression for R1-style model

Input: Initial policy model π_θ , compression function $f(\cdot)$, task prompts \mathcal{D} , hyperparameters α, β, μ

Output: Trained policy model π_θ

- 1: **for** step = 1, \dots , M **do**
 - 2: Sample a batch \mathcal{D}_b from \mathcal{D}
 - 3: Update the old policy model $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
 - 4: Sample G outputs $\{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)$ for each question $q \in \mathcal{D}_b$
 - 5: Apply compression to all outputs: $o'_i \leftarrow f(o_i)$
 - 6: Compute combined reward $r_{i,\text{combine}}$ (Eq. 5) and compress reward $r_{i,\text{compress}}$ (Eq. 11)
 - 7: Compute token-level advantages $\hat{A}_{i,t}$ for each compressed output o'_i (Eq. 10)
 - 8: **for** iteration = 1, \dots , μ **do**
 - 9: Update the policy model π_θ by maximizing the objective $\mathcal{J}_{\text{GRPO}}$ (Eq. 7)
 - 10: **end for**
 - 11: **end for**
 - 12: **return** π_θ
-

This formulation uses the maximum length of a correct, compressed sequence within the group as a normalizer. The final reward combines this with a base reward for format and accuracy, and is normalized by subtracting the group mean, following Liu et al. (2025b) to obtain an unbiased gradient:

$$\tilde{r}_i = r_{i,\text{base}} + \alpha \cdot r_{i,\text{length}} \quad (4)$$

$$r_{i,\text{combine}} = \tilde{r}_i - \text{mean}(\{\tilde{r}_j\}_{j=1}^G) \quad (5)$$

where

$$r_{i,\text{base}} = r_{i,\text{format}} + r_{i,\text{accuracy}} \quad (6)$$

Following prior work, $r_{i,\text{format}}$ and $r_{i,\text{accuracy}}$ are binary rewards that indicate whether the model places its thinking process between `<think>` and `</think>`, and whether the sample leads to the correct answer for the query as verified by Math-Verify¹, respectively. α is a hyperparameter that controls the weight of the Length Reward.

Compress Reward. In standard GRPO, the loss is calculated from the model’s own sampled outputs. To encourage the model to terminate the thinking process once it reaches the correct answer, thereby achieving Brevity, we modify the GRPO objective as follows:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(O|q)} \quad (7)$$

$$\left[\frac{1}{\sum_{i=1}^G |o'_i|} \sum_{i=1}^G \sum_{t=1}^{|o'_i|} \left\{ \min[R_t(\theta) \cdot \hat{A}_{i,t}, \text{clip}(R_t(\theta), 1 - \epsilon, 1 + \epsilon) \cdot \hat{A}_{i,t}] - \beta D_{\text{KL}}(\pi_\theta(\cdot|q) \parallel \pi_{\text{ref}}(\cdot|q)) \right\} \right]$$

¹<https://github.com/huggingface/Math-Verify>

where

$$\mathbb{D}_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o'_i|q)}{\pi_\theta(o'_i|q)} - \log \frac{\pi_{\text{ref}}(o'_i|q)}{\pi_\theta(o'_i|q)} - 1 \quad (8)$$

$$o'_i = f(o_i), \quad R_t(\theta) = \frac{\pi_\theta(o'_{i,t}|q, o'_{i,<t})}{\pi_{\theta_{\text{old}}}(o'_{i,t}|q, o'_{i,<t})} \quad (9)$$

Our key modification to the standard GRPO objective is that the loss is calculated over the compressed trajectories o'_i , rather than the original full trajectories o_i . We define the token-level advantages $\hat{A}_{i,t}$ as follows:

$$\hat{A}_{i,t} = r_{i,\text{combine}} + \gamma \cdot \mathbb{I}(o'_{i,t} = \text{</think>}) \cdot r_{i,\text{compress}} \quad (10)$$

where

$$r_{i,\text{compress}} = \begin{cases} 1 - \frac{|t(o'_i)|}{|t(o_i)|}, & \text{if } i \in \mathcal{C} \text{ \& \; ans}(q) \in t(o'_i) \\ -1, & \text{if } i \in \mathcal{C} \text{ \& \; ans}(q) \notin t(o'_i) \\ 0, & \text{if } i \in \mathcal{W} \end{cases} \quad (11)$$

Let $\text{ans}(q)$ be the ground-truth answer for a given query q . In this setting, we focus on encouraging the model to output the `</think>` token once it reaches the correct answer, that is, at the end of o'_i , so as to compress verbose reasoning in accordance with the principle of **Brevity**. We assign an additional reward only to this token, avoiding unnecessary emphasis on other tokens and making training more efficient and stable. We define this reward as the proportion of the Redundant Sequence, formulated as $1 - \frac{|t(o'_i)|}{|t(o_i)|}$, which represents the efficiency gap between the original and compressed sequences. The hyperparameter γ scales this bonus.

Based on the principle of **Sufficiency**, the model should retain enough reasoning rather than over-compressing at the cost of accuracy. Therefore, we impose a large penalty (-1) on the `</think>` token if the model terminates its reasoning before finding the correct answer, which discourages harmful over-compression and improves training robustness.

To further validate the effectiveness of our method, we follow DAPO (Yu et al., 2025) and calculate the objective across all tokens in a group, instead of averaging token rewards within each single sequence. This removes the original GRPO preference for short correct sequences and long incorrect sequences, making the evaluation of our method more faithful.

4 Experiments

4.1 Experiment Setups

Backbone Models. We choose two representative reasoning models, DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI et al., 2025), as our backbone models, for their strong performance on mathematical and coding reasoning tasks.

LC-Extractor. To accurately identify and extract the valid reasoning part, we develop a specialized parser, termed LC-Extractor, to implement the extraction function f in Eq. 2. We finetune Qwen2.5-3B-Instruct because it is lightweight and easy to deploy. Detailed experimental settings are provided in Appendix B.

Dataset. We used a mixed-difficulty dataset, combining past AIME competition problems with the MATH dataset in an approximate 1:2 ratio to create 2500 training samples. This approach enables the model to learn length compression across problems of varying difficulty.

Evaluation. We evaluate our models on seven datasets, including AIME25, MATH500, GSM8K, AMC, OlympiadBench, GPQA-Diamond, and LiveCodeBench, covering math, general-reasoning, and coding tasks to comprehensively assess reasoning efficiency. We use average Pass@1 as our primary metric. For each test, we sample N times with top-p = 0.95 and temperature = 0.7. For AIME25, we set $N = 64$, while for the other test sets, we set $N = 8$. We set the maximum length to 16384. We also evaluate VT for each model to

measure the Brevity of the thinking process and to study each method’s ability to mitigate the “invalid thinking” phenomenon. We compute VT on five math benchmarks and report the mean, since it is easier to extract standardized, verifiable answers from math reasoning traces. Additionally, we calculate the average relative change in accuracy and token length with respect to the base model on each benchmark, as formulated below:

$$\text{Avg}_{\text{acc}} = \text{mean}_{i=1}^7 \left\{ \frac{\text{Acc}_i^{\text{model}} - \text{Acc}_i^{\text{base}}}{\text{Acc}_i^{\text{base}}} \right\} \quad (12)$$

$$\text{Avg}_{\text{len}} = \text{mean}_{i=1}^7 \left\{ \frac{\text{Len}_i^{\text{model}} - \text{Len}_i^{\text{base}}}{\text{Len}_i^{\text{base}}} \right\} \quad (13)$$

4.2 Baselines

Supervised Fine-tuning (SFT). Inspired by OVERTHINK (Chen et al., 2024), which proposes using only the initial correct solution for fine-tuning, we construct an SFT dataset of 5000 samples by removing the Redundant Sequence from self-generated outputs.

Direct Preference Optimization (DPO) (Rafailov et al., 2023). We create a preference dataset of 5000 samples from the MATH dataset, where the shortest correct answer is treated as the “*chosen*” response and the longest as the “*rejected*” response. This DPO training is applied to the SFT-tuned model.

O1 Pruner (Luo et al., 2025b). A PPO-like offline fine-tuning method to significantly compress CoT length while maintaining performance. We follow its methodology using 10000 samples from the MATH dataset.

ThinkPrune-3K (Hou et al., 2025). A reinforcement learning approach that uses a length-truncation reward for multi-stage compression. We reproduce the ThinkPrune-3k variant, which is reported to be highly efficient, with slight accuracy degradation.

SFT + O1-Pruner. To better understand the effect of compressing the thinking process and pruning the overall sequences at the same time, we also compare with a two-stage training approach, combining SFT and O1 Pruner.

4.3 Experiment Results

LC-R1 outperforms other methods with competitive performance and fewer tokens. As

Table 2: Accuracy (above) and Sequence Length (below) for all methods across seven benchmarks. *AVG* shows the relative change in accuracy and length compared to the Base model (+ increase, - decrease). GPQA-D denotes GPQA-Diamond benchmark, and LCB denotes the pass@10 score on LiveCodeBench. *VT* represents the Valid Thinking ratio. For each column, the best performing score is marked in **bold**, and the second-best is underlined.

Method	AIME25	MATH500	GSM8K	Olympiad	AMC	GPQA-D	LCB	Avg	VT
DeepSeek-R1-Distill-Qwen-7B									
Base	37.7 (11007)	92.6 (3832)	91.6 (1842)	59.7 (7342)	81.2 (6715)	46.6 (6508)	68.8 (8878)	–	58.72%
SFT	36.6 (9457)	90.2 (2497)	91.9 (946)	56.0 (6329)	78.7 (5231)	39.8 (8217)	67.3 (8739)	–4.46% (–15.54%)	95.64%
DPO	36.9 (9718)	91.4 (2277)	90.3 (980)	56.2 (6338)	78.6 (5122)	37.2 (8109)	66.9 (8755)	–5.26% (–16.18%)	<u>96.34%</u>
O1-Pruner	35.0 (8263)	91.5 (2268)	91.1 (1012)	59.6 (4712)	77.1 (4510)	45.5 (5012)	66.7 (5901)	–2.79% (–33.71%)	69.30%
ThinkPrune	38.0 (9309)	93.1 (3253)	91.2 (1546)	60.8 (6225)	82.7 (5510)	50.3 (6508)	67.8 (7180)	+1.58% (–14.13%)	77.16%
SFT+O1-Pruner	35.5 (9466)	91.0 (2245)	89.7 (920)	56.0 (5807)	76.6 (5133)	43.9 (6425)	66.8 (7267)	–4.31% (–24.19%)	85.22%
LC-R1 (Ours)	36.2 (7150)	90.4 (1568)	88.1 (450)	58.7 (4041)	79.1 (3453)	47.2 (4604)	69.0 (6059)	<u>–1.84%</u> (–46.32%)	97.14%
DeepSeek-R1-Distill-Qwen-1.5B									
Base	22.8 (12129)	83.7 (4869)	83.4 (2294)	44.2 (9258)	61.2 (8696)	34.5 (8516)	43.1 (10120)	–	56.06%
SFT	20.5 (10639)	81.4 (3045)	81.3 (1134)	42.7 (7637)	59.7 (6608)	22.4 (10217)	39.8 (10597)	–9.13% (–16.74%)	95.54%
DPO	19.4 (10316)	79.0 (2749)	80.9 (855)	41.1 (6544)	56.7 (5912)	19.8 (9438)	39.2 (10287)	–12.79% (–24.30%)	<u>98.38%</u>
O1-Pruner	23.2 (8731)	84.3 (2913)	82.7 (1162)	47.1 (5960)	65.1 (5131)	32.1 (6173)	42.5 (7305)	<u>+0.89%</u> (–35.64%)	78.20%
ThinkPrune	24.1 (7960)	84.5 (3518)	84.1 (1690)	44.9 (6250)	63.4 (5897)	33.6 (5576)	42.7 (7226)	+1.31% (–30.89%)	65.62%
SFT+O1-Pruner	17.5 (9075)	80.2 (2769)	81.5 (919)	40.0 (6411)	58.7 (5553)	25.0 (7410)	39.4 (8488)	–11.34% (–32.04%)	91.38%
LC-R1 (Ours)	21.2 (6434)	82.5 (2233)	82.7 (841)	43.2 (4333)	61.7 (3947)	33.6 (4489)	42.4 (5722)	–2.14% (–51.86%)	98.64%

presented in Table 2, on the 7B model, LC-R1 achieves an average length reduction of 46.32%, substantially higher than all other baselines, with only a 1.84% drop in average accuracy. Similarly, on the 1.5B model, it attains a 51.86% length reduction for a 2.14% accuracy decrease. This efficiency does not appear to compromise generalization, as it demonstrates more robust performance on out-of-distribution (OOD) benchmarks such as GPQA-Diamond and LiveCodeBench compared to other high-compression methods. Figure 2 shows that our method achieves a more favorable Efficacy-Efficiency trade-off by enabling a high compression ratio with negligible accuracy degradation. LC-R1 also achieves a significantly higher VT rate (over 97%) compared to other methods like O1-Pruner (~70-78%) and ThinkPrune (~66-77%),

demonstrating the superior efficiency of our approach. We further verify this generalization on a post-RL backbone in Appendix C.4, where LC-R1 remains effective on DeepScaleR-1.5B-Preview.

Combining length and compress reward brings superior efficiency to reasoning.

Our ablation study on the Length Reward (L-reward) and Compress Reward (C-reward), presented in Table 3, reveals their critical complementary relationship. While each component alone yields competitive results and lies near the Pareto frontier of performance versus compression efficiency, combining them achieves a better overall balance. Specifically, using only L-reward yields significant compression but a lower VT rate. Conversely, using only C-reward ensures a high VT rate by precisely removing redundancy, but provides more limited

Table 3: Ablation study on the contribution of Length Reward and Compress Reward to the compression process. The results show that removing either component leads to sub-optimal performance, confirming that both are important for efficient reasoning.

Method	AIME25	MATH500	GSM8K	Olympiad	AMC	GPQA-D	LCB	Avg	VT
DeepSeek-R1-Distill-Qwen-7B									
LC-R1 (Ours)	36.2 (7150)	90.4 (1568)	88.1 (450)	58.7 (4041)	79.1 (3453)	47.2 (4604)	69.0 (6059)	-1.84% (-46.32%)	97.14%
w/o L-reward	36.1 (9309)	91.3 (2316)	90.6 (696)	59.4 (5779)	79.0 (5021)	45.9 (6273)	68.0 (8023)	-1.80% (-25.28%)	<u>93.16%</u>
w/o C-reward	37.6 (8738)	92.9 (2498)	91.1 (1012)	59.1 (5344)	80.5 (4741)	48.9 (5727)	68.5 (6893)	+0.31% (-27.35%)	72.24%
DeepSeek-R1-Distill-Qwen-1.5B									
LC-R1 (Ours)	21.2 (6434)	82.5 (2233)	82.7 (841)	43.2 (4333)	61.7 (3947)	33.6 (4489)	42.4 (5722)	-2.14% (-51.86%)	98.64%
w/o L-reward	21.3 (7061)	81.2 (2270)	83.3 (754)	43.4 (5024)	62.2 (4478)	30.6 (5021)	41.9 (6378)	-3.42% (-47.79%)	<u>95.16%</u>
w/o C-reward	21.9 (7988)	83.2 (2965)	84.1 (1160)	44.0 (5363)	63.4 (5192)	30.1 (5847)	43.7 (6874)	-1.70% (-38.35%)	71.10%

Table 4: Sensitivity analysis of the reward weights on DeepSeek-R1-Distill-Qwen-1.5B. The default setting $\alpha = 1, \gamma = 1$ provides a balanced trade-off among accuracy, compression, and VT improvement.

Setting	AIME25	MATH500	GSM8K	Olympiad	AMC	Avg.	VT
Base	22.8 (12129)	83.7 (4869)	83.4 (2294)	44.2 (9258)	61.2 (8696)	-	58.72%
$\alpha = 0.5$ $\gamma = 0.5$	22.7 (9605)	83.8 (3242)	83.3 (1210)	45.9 (6844)	63.5 (6188)	+1.5% (-31.3%)	71.02%
$\alpha = 1$ $\gamma = 1$	22.2 (8785)	83.3 (2698)	83.5 (987)	45.2 (6106)	63.6 (5342)	+0.7% (-40.3%)	79.20%
$\alpha = 1.5$ $\gamma = 0.5$	22.5 (8640)	83.4 (2960)	83.6 (1077)	45.8 (6231)	62.3 (5513)	+0.8% (-38.1%)	75.96%
$\alpha = 0.5$ $\gamma = 1.5$	22.5 (7928)	81.4 (2325)	79.7 (758)	43.3 (5321)	59.8 (4728)	-2.6% (-48.4%)	96.74%

overall compression. Our full LC-R1 successfully integrates these strengths, achieving both the highest compression efficiency and the highest VT rate while maintaining comparable accuracy, showing that the synergy between the two rewards is essential for maximizing reasoning efficiency.

Sensitivity analysis validates the robustness of reward weights. We further conduct a sensitivity analysis by varying the Length Reward weight α and the Compress Reward weight γ , while keeping all other training settings unchanged. Table 4 shows that LC-R1 is more sensitive to γ than to α : increasing γ consistently raises the VT rate and strengthens compression, but an overly large value also leads to a clearer accuracy drop, as $\alpha = 0.5, \gamma = 1.5$ achieves the highest VT rate (96.74%) and the strongest average length reduction (48.4%) at the cost of a 2.6% average accuracy decline. In contrast, varying α under moderate γ

produces a much smoother trade-off, suggesting that the Compress Reward mainly controls how aggressively the model removes the redundant tail after the first correct answer, while the Length Reward provides a more global and stable pressure toward shorter outputs. Overall, the default setting $\alpha = 1, \gamma = 1$ remains a favorable operating point, delivering substantial compression and a clear VT improvement while preserving competitive accuracy. We also provide a training convergence analysis in Appendix C.6, which shows steadily decreasing output lengths and a smooth rise in VT during optimization.

SFT shows limitations in generalization. While SFT achieves a remarkably high VT rate (over 95%), its effectiveness is superficial. The model’s performance collapses on *OOD* benchmarks, indicating that it merely overfits to the structural brevity of the training data rather than learning a general-

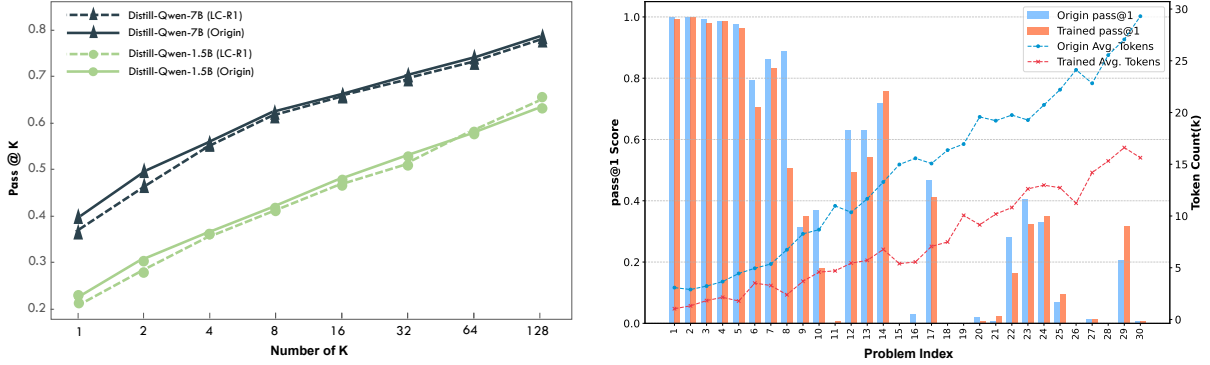


Figure 4: The impact of LC-R1 on the AIME25 benchmark. **Left:** The Pass@k curves show that LC-R1 models maintain competitive performance compared to the original models, preserving their exploration potential. **Right:** Per-problem analysis on DeepSeek-R1-Distill-Qwen-7B shows that LC-R1 achieves similar Pass@1 accuracy while maintaining a consistent token compression ratio across problems of varying difficulty, demonstrating a universal compression effect.

izable, efficient reasoning policy. The poor performance of the hybrid SFT+O1-Pruner method further suggests that a simple combination of off-the-shelf techniques is insufficient. These findings underscore the superiority of RL-based methods like LC-R1, which foster more robust and genuinely efficient reasoning skills.

5 Compression Impact Analysis

Compression does not impair exploration capability. To investigate the deeper impact of compression on the model’s problem-solving potential, we sample 256 times on AIME25 with a maximum length of 32,768 and evaluate Pass@k for both models before and after compression. The results in Figure 4 (left) reveal a key phenomenon: across the entire Pass@k range from $k = 1$ to 128 on AIME25, the performance curve of the model compressed by LC-R1 nearly overlaps with that of the original model. This result suggests that compression does not harm the model’s ability to find a correct solution through multiple attempts. It indicates that the pruned “invalid thinking” segments are truly redundant, and that their removal does not diminish the model’s core knowledge or creative problem-solving potential. Additional Pass@k results on Olympiad and GPQA are provided in Appendix C.4, where LC-R1 shows similarly close tracking to the base models across k .

Compression remains consistent across varying problem difficulties. To analyze our method’s behavior at a more fine-grained level, we plot the per-problem Pass@1 accuracy against the original model’s token consumption on AIME25 (Figure 4,

right). The plot reveals a clear difficulty spectrum, where problems requiring more tokens from the base model generally correspond to lower Pass@1 scores. Crucially, LC-R1 applies a relatively uniform compression effect across this entire spectrum, with per-problem outcomes (*i.e.*, success or failure) remaining remarkably consistent with those of the base model. This provides strong evidence that LC-R1 functions as a robust and difficulty-agnostic efficiency layer, successfully streamlining the reasoning process without altering the model’s core problem-solving logic for specific problems.

6 Conclusion

In this paper, we address the “invalid thinking” phenomenon in current LRMs, namely their tendency to double-check their work after the correct answer has already been derived. To tackle this issue, we introduced the principles of Brevity and Sufficiency and proposed LC-R1, an RL-based post-training method with a dual-reward system that compresses overall sequence length while pruning the Redundant Sequence. Extensive experiments demonstrate that LC-R1 achieves a more favorable *Efficacy-Efficiency* trade-off. Further analyses show that LC-R1 does not degrade the model’s exploration ability, and that the compression effect remains robust across problems of varying difficulty.

Acknowledgment

Many thanks to Yao Wan for his invaluable support and comments.

Limitations

In this paper, we address the “invalid thinking” phenomenon, a key source of inefficiency in Large Reasoning Models where they engage in unnecessary verification after deriving a correct answer. We introduce LC-R1, a novel post-training method featuring a dual-reward system that encourages both overall conciseness and the specific elimination of this redundancy. Our experiments demonstrate that LC-R1 achieves a more favorable trade-off between performance and efficiency than existing approaches. However, compressed reasoning may still suppress useful verification on difficult problems, leading the model to produce concise but incorrect answers. While our current validation focuses on models up to the 7B scale due to computational constraints, this work provides a practical path toward more computationally frugal LRMs; in high-stakes settings, such methods should be paired with task-appropriate validation and human oversight.

References

- Marah Abdin, Sahaj Agarwal, Ahmed Awadallah, Vidhisha Balachandran, Harkirat Behl, Lingjiao Chen, Gustavo de Rosa, Suriya Gunasekar, Mojan Javaheripi, Neel Joshi, Piero Kauffmann, Yash Lara, Caio César Teodoro Mendes, Arindam Mitra, Bismira Nushi, Dimitris Papailiopoulos, Olli Saarikivi, Shital Shah, Vaishnavi Shrivastava, and 4 others. 2025. [Phi-4-reasoning technical report](#). *Preprint*, arXiv:2504.21318.
- Pranjal Aggarwal and Sean Welleck. 2025. [L1: Controlling how long a reasoning model thinks with reinforcement learning](#). *Preprint*, arXiv:2503.04697.
- Daman Arora and Andrea Zanette. 2025. [Training language models to reason efficiently](#). *Preprint*, arXiv:2502.04463.
- Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. [Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching](#). *Preprint*, arXiv:2503.05179.
- Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, ... Yian Zhang, and Chris Alexiuk. 2025. [Llama-nemotron: Efficient reasoning models](#). *Preprint*, arXiv:2505.00949.
- Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025. [Towards reasoning era: A survey of long chain-of-thought for reasoning large language models](#). *Preprint*, arXiv:2503.09567.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuqi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. [Do not think that much for 2+ 3=? on the overthinking of o1-like llms](#). *arXiv preprint arXiv:2412.21187*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. [The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks](#). *arXiv preprint arXiv:2502.08235*.
- 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 Zhen Zhang. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Google. 2025a. [Gemini 2.5 flash](#). <https://developers.googleblog.com/en/start-building-with-gemini-25-flash/>.
- Google. 2025b. [Gemini 2.5 pro](#). <https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro>.
- Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I. Wang. 2024. [Cruxeval: A benchmark for code reasoning, understanding and execution](#). *arXiv preprint arXiv:2401.03065*.
- Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. [Token-budget-aware llm reasoning](#). *arXiv preprint arXiv:2412.18547*.
- 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](#). *arXiv preprint arXiv:2504.01296*.
- Jian Hu, Jason Klein Liu, and Wei Shen. 2025. [Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models](#). *Preprint*, arXiv:2501.03262.

- International Conference on Artificial Intelligence in Medicine. The 23rd international conference on artificial intelligence in medicine (aime 2025). <https://aime25.aimeinmedicine.info/>. Accessed: 2025-06-10.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. [Livecodebench: Holistic and contamination free evaluation of large language models for code](#). *arXiv preprint arXiv:2403.07974*.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. In *The Twelfth International Conference on Learning Representations*.
- Jiawei Liu and Lingming Zhang. 2025. Code-r1: Reproducing r1 for code with reliable rewards.
- Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2025a. [Learn to reason efficiently with adaptive length-based reward shaping](#). *Preprint*, arXiv:2505.15612.
- 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](#). *Preprint*, arXiv:2503.20783.
- Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025a. Adar1: From long-cot to hybrid-cot via bi-level adaptive reasoning optimization. *arXiv preprint arXiv:2504.21659*.
- Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025b. [O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning](#). *Preprint*, arXiv:2501.12570.
- Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025a. [Reasoning models can be effective without thinking](#). *Preprint*, arXiv:2504.09858.
- Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025b. [Cot-valve: Length-compressible chain-of-thought tuning](#). *Preprint*, arXiv:2502.09601.
- Mathematical Association of America. American mathematics competitions (amc). <https://maa-amc.org/student-programs/amc/>. Accessed: 2025-06-10.
- OpenAI. 2024. Chatgpt. <https://openai.com/o1/>.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv: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.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.
- Haoxiang Sun, Yingqian Min, Zhipeng Chen, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, Lei Fang, and Ji-Rong Wen. 2025. [Challenging the boundaries of reasoning: An olympiad-level math benchmark for large language models](#). *Preprint*, arXiv:2503.21380.
- Kimi Team, Angang Du, Bofei Gao, Bofei 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 77 others. 2025. [Kimi k1.5: Scaling reinforcement learning with llms](#). *Preprint*, arXiv:2501.12599.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models](#).
- Qwen Team. 2025a. [Qwen3](#).
- Qwen Team. 2025b. [Qwq-32b: Embracing the power of reinforcement learning](#).
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Galouédec. 2020. Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>.

Chenlong Wang, Yuaning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025. [Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency](#). *Preprint*, arXiv:2506.08343.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. [Chain-of-thought prompting elicits reasoning in large language models](#). *Preprint*, arXiv:2201.11903.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. 2025. [Dynamic early exit in reasoning models](#). *Preprint*, arXiv:2504.15895.

Jingyang Yi and Jiazheng Wang. 2025. [Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning](#). *Preprint*, arXiv:2504.21370.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. [Distilling system 2 into system 1](#). *Preprint*, arXiv:2407.06023.

Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025. [Dapo: An open-source llm reinforcement learning system at scale](#). *arXiv preprint arXiv:2503.14476*.

Danlong Yuan, Tian Xie, Shaohan Huang, Zhuocheng Gong, Huishuai Zhang, Chong Luo, Furu Wei, and Dongyan Zhao. 2025. [Efficient rl training for reasoning models via length-aware optimization](#). *Preprint*, arXiv:2505.12284.

A Related Work

Reinforcement Learning For Reasoning. Reinforcement learning (RL) has emerged as a pivotal technique in the post-training phase of Large Language Models (LLMs), demonstrating significant potential for enhancing their reasoning abilities. A landmark work in this domain is OpenAI's o1 model (OpenAI, 2024). As the first large-scale application of RL for reasoning, o1 achieved state-of-the-art reasoning capabilities at the time of its release. Shortly after, DeepSeek-R1 (DeepSeek-AI et al., 2025) was introduced as the first open-source model to match the performance of o1, significantly advancing the development and popularization of RL-based reasoning techniques. This technical trajectory has led to the emergence of numerous powerful Large Reasoning Models (LRMs), such as Gemini 2.5 (Google, 2025b), QwQ (Team, 2025b), and Phi-4 (Abdin et al., 2025). Recently, Reinforcement Learning with Verifiable Rewards (RLVR) has been shown to be an effective approach for significantly improving model reasoning abilities in

domains such as mathematics and programming (Shao et al., 2024; Liu and Zhang, 2025). Concurrently, the research community has proposed various advanced RL algorithms to optimize post-training, including GRPO (Shao et al., 2024), Reinforce++ (Hu et al., 2025), DAPO (Yu et al., 2025), and Dr.GRPO (Liu et al., 2025b). These algorithmic innovations have continuously improved the efficiency and effectiveness of RL training. Our proposed method, LC-R1, builds on these foundational works with specific adjustments aimed at pruning redundant sequences in the reasoning process, thereby achieving a more favorable trade-off between efficacy and efficiency.

Efficient Reasoning. While elaborate reasoning is more likely to yield a correct answer, its verbose thought process significantly increases time and computational costs, a phenomenon termed *overthinking* (Chen et al., 2024). To mitigate this issue, researchers have proposed various solutions from different perspectives (Sui et al., 2025). These approaches can be broadly categorized. The first category directly constrains reasoning redundancy through length control, with typical examples such as CoT-Valve (Ma et al., 2025b) and L1 (Aggarwal and Welleck, 2025). A second category focuses on enabling the model to adapt its reasoning depth to query difficulty. For instance, Adar1 (Luo et al., 2025a) and DAST (Shen et al., 2025) construct preference datasets to train models to generate reasoning sequences that match problem complexity. A third category integrates efficiency considerations into the reinforcement learning framework. Methods such as O1-Pruner (Luo et al., 2025b), ThinkPrune (Hou et al., 2025), EfficientReasoning (Arora and Zanette, 2025), and Kimi (Team et al., 2025) incorporate length-based penalties into the reward function, incentivizing the model to produce more concise reasoning while maintaining accuracy. More recent RL approaches further refine this line by using length-aware reward shaping or adaptive targets, such as ShortRL (Yuan et al., 2025), ShorterBetter (Yi and Wang, 2025), and LASER (Liu et al., 2025a). Furthermore, training-free methods also improve reasoning efficiency on the fly through sophisticated prompting or dynamic stopping strategies, such as Sketch-of-Thought (Aytes et al., 2025), Token-Budget (Han et al., 2024), No Think (Ma et al., 2025a), NoWait (Wang et al., 2025), and Dynamic Early Exit (Yang et al., 2025).

B Details of LC-Extractor

We train Qwen-2.5-3B-Instruct (Team, 2024) as the LC-Extractor model. We construct a dataset consisting of 5,000 $\langle \text{Question}, \text{Thinking Process}, \text{Answer} \rangle$ triplets from the MATH dataset and identify the position of the first correct token using Gemini-2.5-Flash (Google, 2025a), followed by rigorous rule-based filtering. We then distill this knowledge into a smaller model by training for two epochs on these curated samples. We further strengthen the validation of LC-Extractor with a 500-sample human evaluation, using 100 model rollouts from each of five math benchmarks (AIME25, MATH500, GSM8K, Olympiad, and AMC). For each sample, annotators check whether the predicted prefix fully contains the reasoning up to the first correct final answer, without cutting earlier or mis-locating the answer boundary. Under this stricter protocol, LC-Extractor achieves 494/500 correct truncations (98.8% accuracy), as illustrated in Figure 6. Among the six errors, all stop slightly after the ideal boundary and retain a brief double-check segment; we observe no early cutoffs or cases where the correct answer is missed. This means the residual errors are conservative and mainly underestimate compression rather than removing necessary reasoning. The LC-Extractor is prompted using the template shown in Figure 5.

The 500-sample human evaluation was conducted by the authors. No external participants were recruited, and no payment was involved.

C Detailed Experiment Setups

C.1 Model

We use **DeepSeek-R1** (DeepSeek-AI et al., 2025), **Qwen3-32B** (Team, 2025a), **QwQ-32B** (Team, 2025b), **Llama-3.3-Nemotron-Super-49B-V1** (Bercovich et al., 2025), **Distill-Qwen-7B**, **Distill-Qwen-1.5B** (Yu et al., 2024), and **Qwen-2.5-3B-Instruct** (Team, 2024) models in our paper. We introduce their licenses and key characteristics as follows:

- **DeepSeek-R1.** An open-source 671 B \rightarrow 37 B MoE reasoning model trained largely through reinforcement learning, which elicits self-verification, reflection and lengthy chain-of-thought traces while supporting 128K-token context; it matches proprietary o1 on math / code benchmarks using only public data.

- **Qwen3-32B.** The 32.8 B-parameter third-generation Qwen model that toggles between “thinking” and “non-thinking” modes, delivering state-of-the-art reasoning, multilingual chat and up to 131 K context in a single dense checkpoint.
- **QwQ-32B.** A medium-sized Qwen reasoning variant refined with SFT + RL; provides explicit $\langle \text{think} \rangle$ traces, 131 K context and DeepSeek-R1-level accuracy on hard evaluations.
- **Llama-3.3-Nemotron-Super-49B-V1.** NVIDIA’s NAS-pruned 49 B derivative of Llama-3.3-70B, post-trained for reasoning, RAG and tool calling; couples 128 K context with single-H100 deployment efficiency for cost-sensitive production.
- **DeepSeek-R1-Distill-Qwen-7B.** A 7 B dense checkpoint distilled from DeepSeek-R1 onto the Qwen2.5 backbone, pushing small-model MATH-500 pass1 beyond 92 % and surpassing o1-mini on several reasoning suites while remaining laptop-friendly.
- **DeepSeek-R1-Distill-Qwen-1.5B.** An ultra-compact 1.5 B model distilled from R1 that preserves chain-of-thought and achieves 83.9 % pass1 on MATH-500, bringing competitive analytical power to edge and mobile deployments.
- **Qwen-2.5-3B-Instruct.** A 3.09 B instruction-tuned model with 128 K context, strengthened coding/math skills and multilingual support, designed as a lightweight yet controllable chat foundation for downstream tasks.

C.2 Dataset

We benchmark on the **AIME25** (International Conference on Artificial Intelligence in Medicine), **MATH500** (Lightman et al., 2023), **GSM8K** (Cobbe et al., 2021), **OlympiadBench** (Sun et al., 2025), **AMC** (Mathematical Association of America), **GPQA Diamond** (Rein et al., 2024) and **LiveCodeBench** (Jain et al., 2024) benchmarks in our paper. We introduce them as follows:

- **AIME25.** A benchmark with 30 questions distilled from twenty-five years of *American Invitational Mathematics Examination* papers. Each item is a three-digit short-answer problem that probes upper-secondary algebra, geometry, combinatorics.
- **MATH500.** A 500-problem evaluation slice covering the full subject breadth of the original

Prompt to Extract Answer Prefix

You are Qwen, created by Alibaba Cloud. You are a helpful assistant.

Instruction:

Extract Answer Prefix You'll get a Problem, a Thinking Process, and its Ground Truth Answer.

Your Task:

1. Read the Thinking Process from the beginning carefully.
2. Find the first sentence that reveals the Ground Truth Answer.
3. Copy everything from the start of the Thinking Process up to and including that sentence.
4. Important: Do not include any text after that sentence.

Example:

- Problem: What is $1 + 1$?
- Thinking Process: Okay, I need to solve $1 + 1$. That gives 2. Let me check again—yes, it's 2.
- Ground Truth Answer: 2.
- Expected Output: Okay, I need to solve $1 + 1$. That gives 2.

Input Provided:

- Problem: <Problem>
- Thinking Process: <Thinking Process>
- Ground Truth Answer: <Ground Truth Answer>

Your Output:

A prefix of "Thinking Process", with Ground Truth at the end.

Figure 5: Our prompt for extraction of answer prefix.

The screenshot shows a web interface titled "JSON Entry Review Interface" for "Entry 1". It displays a math problem about finding the cosine of the angle between two lines. The problem is: "A line is parameterized by $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ -2 \end{pmatrix} + t \begin{pmatrix} 3 \\ 4 \end{pmatrix}$. A second line is parameterized by $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -8 \\ 12 \end{pmatrix} + u \begin{pmatrix} 1 \\ 3 \end{pmatrix}$. If θ is the acute angle formed by the two lines, then find $\cos \theta$."

The solution provided is $\frac{3}{\sqrt{10}}$. The reasoning prefix explains the steps: identifying direction vectors $v = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ and $w = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$, using the dot product formula $\cos \theta = \frac{v \cdot w}{\|v\| \|w\|}$, and calculating the dot product $v \cdot w = 3(1) + 4(3) = 15$ and the magnitudes $\|v\| = \sqrt{3^2 + 4^2} = 5$ and $\|w\| = \sqrt{1^2 + 3^2} = \sqrt{10}$.

Figure 6: The annotation tool to evaluate the LC-Extractor.

MATH competition corpus. Balanced across difficulty tiers and topics, it serves as a rigorous

yardstick for advanced high-school and early undergraduate mathematical reasoning, without the

Table 5: Hyperparameters for LC-R1 training.

Hyperparameter	R1-Distill-Qwen-7B	R1-Distill-Qwen-1.5B
cutoff_len	8192	8192
batch_size	32	32
learning_rate	3.0e-6	2.0e-6
num_train_epochs	1.0	1.0
α	1.0	1.0
β	0.04	0.04
γ	1.0	1.0
num_generations	6	8
ϵ	0.2	0.2

runtime burden of the complete 12k-question set.

- **GSM8K.** The widely-adopted *Grade-School Math 8K* benchmark of 1,319 everyday word-problems. Requiring multi-step arithmetic and commonsense, GSM8K remains the de-facto standard for assessing chain-of-thought quality on conversational math tasks.
- **Olympiad.** A curated collection of roughly 3 k national and international mathematics-olympiad problems. Predominantly proof-style or numeric-answer challenges, this benchmark gauges creative, non-routine reasoning at the highest pre-university level.
- **AMC.** An aggregate of 83 from the *American Mathematics Competitions 10/12*. Spanning 2000–2024, it offers a longitudinal benchmark on foundational secondary-school math.
- **GPQA Diamond.** A benchmark with 198 graduate-level Google-proof multiple-choice questions requiring deep domain expertise and multi-step reasoning, curated by researchers from New York University, CohereAI, and Anthropic; evaluated in closed-book and open-book settings using accuracy as the metric.
- **LiveCodeBench.** A dynamic, contamination-free coding benchmark originally hosting 511 problems (release v2) collected from LeetCode, AtCoder, and CodeForces, designed by UC Berkeley, MIT, and Cornell researchers to holistically assess LLMs’ code generation, execution, and test prediction capabilities using Pass@k.

C.3 Settings

We use a mixed-difficulty dataset, combining past AIME competition problems with the MATH dataset in an approximate 1:2 ratio to create 2,500 training examples. We use the TRL framework

(von Werra et al., 2020) to train our models. Both models are trained on $4 \times$ A800-80G GPUs, and the hyperparameters are presented in Table 5.

Baseline settings. We compare LC-R1 with five baselines: SFT, DPO, O1-Pruner, ThinkPrune, and SFT+O1-Pruner. The hybrid method uses the same settings as its constituent methods, so we report the settings of the first four methods here.

- **SFT.** We construct the training dataset by extracting the valid thinking process to reconstruct concise versions of sequences sampled from the MATH dataset. We set cutoff_len=8192, epoch=1, learning_rate = 3.0e-6, and max_samples = 5000.
- **DPO.** We construct the preference dataset by sampling eight times on the MATH dataset, choosing the longest sample as the negative response and the shortest sample as the positive response. We set cutoff_len=8192, epoch=2, learning_rate = 5e-6, and max_samples = 5000.
- **O1-Pruner.** We use the provided Python scripts to construct the weighted training dataset, with cutoff_len=4096, epoch=2, learning_rate = 2.0e-7, and max_samples = 10000.
- **ThinkPrune-3K.** We reproduce training on the ThinkPrune-length3000 dataset, which contains 2470 examples. We set cutoff_len=8192, epoch=2, learning_rate = 2.0e-6, num_generations=8, and batch_size=32.

C.4 Additional Experimental Results

To evaluate the general applicability of our method, we conduct additional experiments using the same settings and baselines on DeepScaleR-1.5B-Preview, a post-RL model that differs from the previously tested distilled models.

Table 6: Performance comparison on DeepScaleR-1.5B-Preview model.

Method	AIME25	MATH500	GSM8K	Olympiad	AMC	Avg.
Base	31.7 (8766)	88.9 (3188)	86.7 (2141)	53.7 (5912)	74.0 (5602)	–
LC-R1	29.9 (6872)	87.1 (2334)	85.9 (1157)	52.7 (4436)	73.4 (4874)	-2.5% (-28.3%)
O1-Pruner	27.1 (8143)	86.5 (2879)	85.6 (1826)	51.9 (5119)	69.8 (5099)	-6.1% (-11.5%)
ThinkPrune	29.5 (7882)	87.1 (2659)	86.0 (1624)	52.7 (4805)	73.8 (5622)	-2.4% (-13.4%)
SFT	28.3 (8338)	87.4 (2480)	86.0 (1152)	52.0 (5206)	72.2 (5070)	-3.6% (-16.4%)

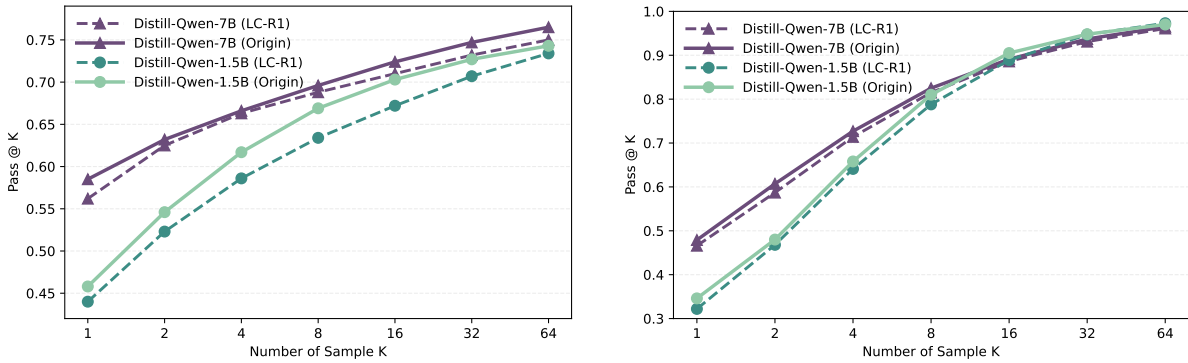


Figure 7: Additional Pass@k evaluation of LC-R1. *Left:* Olympiad. *Right:* GPQA.

As shown in Table 6, this model exhibits minimal “overthinking”, with small output-length variance and a high valid-thinking rate (>70%), making efficiency gains harder to obtain. LC-R1’s reward design adapts well to diverse models and their CoT patterns, showing encouraging generalization across models.

Additional Pass@k results on other benchmarks.

To complement Figure 4 in the main text, we additionally evaluate Pass@k on Olympiad and GPQA using the same comparison protocol between the base model and LC-R1. The resulting curves remain close across all k , further supporting that LC-R1 preserves the model’s exploration capability under multi-sample decoding.

C.5 Training Cost Discussion

Since SFT and RL methods differ in their training setups, with RL involving online sampling while SFT uses fixed datasets, directly comparing wall-clock time does not fully reflect training resource consumption. Therefore, we use a more aligned metric: the number of tokens used to compute the loss function. A larger number of tokens requiring

computation implies more parallel computation, which better reflects computational resource usage.

Average token lengths per sequence are: SFT (1436), O1-Pruner (3826), LC-R1 (2429), and ThinkPrune (3844). The first two methods use 10k training samples, while LC-R1 uses $1 \times 8 \times 1500$ (epoch \times num_generation \times train_data_size) and ThinkPrune uses $2 \times 8 \times 2500$. Thus, the total tokens used for loss computation are 14M, 38M, 29M, and 153M, respectively, showing that LC-R1’s computational cost is relatively modest.

LC-R1’s extra cost mainly comes from the LC-Extractor, which accounts for $\sim 15\%$ of total training time. The LC-R1 training framework does not introduce excessive resource consumption, and using compressed CoTs for objective computation significantly reduces the total number of tokens involved.

C.6 Training Convergence Analysis

One concern is whether the proposed dual-reward design leads to stable optimization or merely encourages the model to terminate too early. Figure 8 shows the training dynamics of the default 1.5B run and suggests that the optimization process is stable.

Table 7: Multi-seed evaluation of the base model and LC-R1 with seeds 42, 52, and 62. We report mean \pm standard deviation across seeds. Each benchmark entry reports accuracy, with the average sequence length in parentheses. The Avg. column reports the relative change in average accuracy and length of LC-R1 with respect to the corresponding base model.

Scale	Model	AIME25	MATH500	GSM8K	Olympiad	AMC	GPQA	Avg.
7B	Base	37.5 \pm 1.2 (11160 \pm 192)	93.2 \pm 0.4 (3800 \pm 118)	91.4 \pm 0.5 (1704 \pm 20)	57.7 \pm 1.6 (7552 \pm 59)	80.4 \pm 1.1 (6812 \pm 234)	45.8 \pm 1.1 (7802 \pm 138)	–
	LC-R1	35.9 \pm 0.7 (7262 \pm 89)	90.7 \pm 0.3 (1548 \pm 63)	87.9 \pm 0.3 (448 \pm 4)	57.7 \pm 0.0 (4028 \pm 37)	78.1 \pm 0.2 (3500 \pm 87)	44.4 \pm 1.4 (4611 \pm 72)	-2.8% \pm 0.9 (-50.7% \pm 0.8)
1.5B	Base	22.4 \pm 0.8 (12243 \pm 81)	83.6 \pm 1.8 (4757 \pm 23)	84.1 \pm 0.7 (2325 \pm 37)	44.6 \pm 1.1 (9131 \pm 68)	60.5 \pm 1.2 (8745 \pm 173)	29.5 \pm 0.9 (8913 \pm 94)	–
	LC-R1	21.0 \pm 0.6 (6629 \pm 243)	82.5 \pm 1.7 (2277 \pm 45)	82.4 \pm 1.0 (865 \pm 7)	43.1 \pm 0.1 (4606 \pm 131)	60.6 \pm 1.1 (4232 \pm 114)	28.3 \pm 0.8 (4563 \pm 75)	-2.8% \pm 1.8 (-51.8% \pm 1.2)

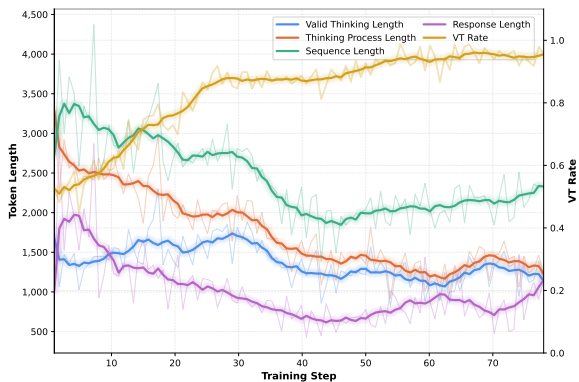


Figure 8: Training convergence of LC-R1 under the default setting in Table 5, measured from the training log of the DeepSeek-R1-Distill-Qwen-1.5B run. The left axis reports several length-related statistics, and the right axis reports the Valid Thinking (VT) rate.

Throughout training, the overall thinking-process length and sequence length decrease steadily, while the VT rate rises consistently and eventually saturates at a high level. This indicates that LC-R1 progressively removes redundant reasoning after the first correct answer rather than relying on unstable or abrupt truncation behavior.

The curves are also consistent with the design choices of LC-R1. First, applying the compression reward only to the `</think>` token focuses the learning signal on the stopping decision, which leads to a smooth increase in VT rather than oscillatory behavior. Second, the fixed penalty for insufficient reasoning prevents harmful over-compression: while the total thinking length drops substantially, the valid thinking length decreases much more moderately, showing that useful reasoning is largely preserved. Third, the normalized length reward provides a stable global pressure to-

ward shorter outputs, as reflected by the gradual reduction of sequence length and response length over training. Overall, these dynamics support our claim that the Length Reward and Compress Reward play complementary roles: one encourages global conciseness, while the other specifically suppresses the redundant tail after the first correct answer.

C.7 Multi-Seed Evaluation

We rerun both the base models and LC-R1 with three random seeds (42, 52, and 62) and report the corresponding means and standard deviations in Table 7. The results show that the main conclusion is robust across seeds: for both 7B and 1.5B, LC-R1 consistently delivers about 50% sequence-length reduction, while the across-seed variation of this reduction remains small (0.8 for 7B and 1.2 for 1.5B in relative percentage points). In contrast, the average accuracy change remains modest at -2.8% on both scales, indicating that the efficiency gain is much larger than the seed-induced fluctuation. This suggests that the improvement of LC-R1 is not an artifact of a favorable seed, but a stable property of the training procedure. Moreover, the benchmark-level standard deviations of LC-R1 stay in a similar range to those of the base models, which indicates that adding the compression objective does not introduce unusual instability.

D Case Study

We present case studies comparing LC-R1 with O1-Pruner (Luo et al., 2025b) and the base model. These case studies are shown in Figure 9 and Figure 10.

Question

Define $p = \sum_{k=1}^{\infty} (1 / k^2)$

and $q = \sum_{k=1}^{\infty} (1 / k^3)$.

Find a way, to express the double series

$S = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} [1 / (j + k)^3]$ in terms of **p** and **q**.

Ground Truth Answer: p - q.

LC-R1 (Ours): 519 Tokens

Valid Thinking : 319 Tokens

Okay, so I have this problem where I need to find the value of the double sum $S = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} 1/(j+k)^3$ in terms of p and q , where $p = \sum_{k=1}^{\infty} 1/k^2$ and $q = \sum_{k=1}^{\infty} 1/k^3$. First, I recall that sometimes when dealing with double sums, especially infinite ones, it can be helpful to express them in terms of simpler sums or known constants, and here p and q are well known as the Riemann zeta function values at 2 and 3, so I need to relate the given double sum to those. I notice that both j and k run from 1 to infinity, so the sum runs over all positive integer pairs (j, k) ; a useful trick is to set $n = j + k$, in which case for each $n \geq 2$ there are $n - 1$ pairs (j, k) with $j + k = n$, allowing us to rewrite the double sum as the single sum from $n = 2$ to infinity of $(n - 1)/n^3$. This single sum splits into sum from $n = 2$ to infinity of $1/n^2$ minus sum from $n = 2$ to infinity of $1/n^3$, which equal $p - 1$ and $q - 1$ respectively, so the result is $(p - 1) - (q - 1) = p - q$.

Final Response : 200 Tokens

To find the value of the double sum $S = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} 1/(j+k)^3$ in terms of $p = \sum_{k=1}^{\infty} 1/k^2$ and $q = \sum_{k=1}^{\infty} 1/k^3$, we change variables by letting $n = j + k$; for each $n \geq 2$ there are $n - 1$ pairs (j, k) with $j + k = n$, so the double sum becomes sum from $n = 2$ to infinity of $(n - 1)/n^3$, which splits into sum from $n = 2$ to infinity of $1/n^2$ minus sum from $n = 2$ to infinity of $1/n^3$; since these equal $p - 1$ and $q - 1$ respectively, subtracting gives $(p - 1) - (q - 1) = p - q$.

Thinking Length: 319 Tokens, Valid Ratio: 100%

O1Pruner: 1546 Tokens

Valid Thinking: 537 Tokens

Alright, so I have this problem where I need to express a double sum in terms of two known series, p and q . The given series are: $p = \sum_{k=1}^{\infty} 1/k^2$, $q = \sum_{k=1}^{\infty} 1/k^3$. And I need to find a way to write: $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} 1/(j+k)^3$ in terms of p and q . Hmm, okay. Let me think about how to approach this. First, I know that double sums can sometimes be tricky, but maybe I can find a way to relate this to the given p and q . Let me write down what I need to find: $S = \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} 1/(j+k)^3$. I remember that sometimes changing variables can help simplify such sums. Maybe I can let $n = j + k$. That might be a good start. So, if I set $n = j + k$, then for each fixed n , j can range from 1 to $n - 1$, because k must also be at least 1. So, for each $n \geq 2$, j goes from 1 to $n - 1$, and $k = n - j$. So, rewriting the double sum as a single sum over n , we have: $S = \sum_{n=2}^{\infty} \sum_{j=1}^{n-1} 1/n^3$. Because for each n , the inner sum just adds up $(n - 1)$ terms of $1/n^3$. So, the inner sum is simply $(n - 1)/n^3$. Therefore, $S = \sum_{n=2}^{\infty} (n - 1)/n^3$. That becomes $S = \sum_{n=2}^{\infty} (1/n^2 - 1/n^3)$. So we get $S = \sum_{n=2}^{\infty} 1/n^2 - \sum_{n=2}^{\infty} 1/n^3$. We know $\sum_{n=2}^{\infty} 1/n^2 = p - 1$, and $\sum_{n=2}^{\infty} 1/n^3 = q - 1$. Substituting back gives $S = (p - 1) - (q - 1) = p - q$.

Invalid Thinking : 702 Tokens

Wait, is that right? Let me double-check my steps. 1. I changed variables to $n = j + k$, which gave me a double sum over n starting from 2. 2. For each n , the inner sum over j (or k) gave me $n - 1$ terms of $1/n^3$, so the inner sum is $(n - 1)/n^3$. 3. Then I split the fraction into $1/n^2 - 1/n^3$. 4. Summed each part separately from $n=2$ to ∞ . 5. Expressed those sums in terms of p and q by subtracting the $n=1$ terms. 6. Subtracted the sums, leading to $p - q$. That seems correct. Let me verify with another approach to make sure. Alternatively, consider computing S directly. Let's write out the terms for small n and see if the pattern holds. For $n=2$: $(2-1)/2^3 = 1/8$. For $n=3$: $(3-1)/3^3 = 2/27$. For $n=4$: $3/64$. And so on. So, $S = 1/8 + 2/27 + 3/64 + 4/125 + \dots$. If $p = \pi^2/6 \approx 1.6449$ and $q = \text{Apery's constant} \approx 1.2020569$, then $p - q \approx 0.4428$. Let me compute the partial sum of S up to $n=4$: $1/8 \approx 0.125$, $2/27 \approx 0.07407$, $3/64 \approx 0.046875$, $4/125 \approx 0.032$. Adding these: $0.125 + 0.074 \approx 0.199$, $+0.046875 \approx 0.246$, $+0.032 \approx 0.278$. If we sum up to $n=10$: $n=2: 1/8 \approx 0.125$, $n=3: 2/27 \approx 0.07407$, $n=4: 3/64 \approx 0.046875$, $n=5: 4/125 \approx 0.032$, $n=6: 5/216 \approx 0.023148$, $n=7: 6/343 \approx 0.01749$, $n=8: 7/512 \approx 0.01367$, $n=9: 8/729 \approx 0.01097$, $n=10: 9/1000 \approx 0.009$. Adding these: $0.125 + 0.074 \approx 0.199$, $+0.047 \approx 0.246$, $+0.032 \approx 0.278$, $+0.023 \approx 0.301$, $+0.017 \approx 0.318$, $+0.014 \approx 0.332$, $+0.011 \approx 0.343$, $+0.009 \approx 0.352$. So partial sum ≈ 0.35 . Still smaller than $p - q \approx 0.4428$, but approaching. Hence, the formula $S = p - q$ seems valid. Final Answer: $p - q$.

Final Response : 307 Tokens

To express the double sum $\sum_{j=1}^{\infty} \sum_{k=1}^{\infty} 1/(j+k)^3$ in terms of the given series p and q , we start by changing variables. Let $n = j + k$. For each fixed n , j ranges from 1 to $n - 1$, and $k = n - j$. Thus, the double sum can be rewritten as: $\sum_{n=2}^{\infty} \sum_{j=1}^{n-1} 1/n^3$. The inner sum for each n is simply $(n - 1)/n^3$. Therefore, we have: $\sum_{n=2}^{\infty} (n - 1)/n^3$. This can be split into two separate sums: $\sum_{n=2}^{\infty} (1/n^2 - 1/n^3)$. We express these sums in terms of p and q : 1. The sum of $1/n^2$ from $n = 2$ to ∞ is $p - 1$. 2. The sum of $1/n^3$ from $n = 2$ to ∞ is $q - 1$. Thus, substituting these back, we get: $S = (p - 1) - (q - 1)$. Simplifying this expression: $S = p - 1 - q + 1 = p - q$. Therefore, the final answer is: $p - q$.

Thinking Length: 1239 Tokens, Valid Ratio: 43%

Figure 9: Case study comparing LC-R1 with O1-Pruner.

Question

If $f(x) = (3x - 2) / (x - 2)$, what is the value of $f(-2) + f(-1) + f(0)$?
Express your answer as a common fraction.

LC-R1 (Ours): 896 Tokens **Ground Truth Answer: 14/3.**

Valid Thinking : 586 Tokens

Okay, so I need to find the value of $f(-2) + f(-1) + f(0)$ where the function $f(x)$ is given as $(3x - 2)/(x - 2)$. Hmm, let me start by understanding what the function looks like. It's a rational function, meaning it's a ratio of two polynomials. The numerator is linear, $3x - 2$, and the denominator is also linear, $x - 2$. I remember that for rational functions, the domain excludes any x that makes the denominator zero, so x cannot be 2 here. But since we're evaluating at $x = -2, -1$, and 0 , which are all fine because 2 isn't among them, I don't have to worry about division by zero in this case. Alright, so I need to compute $f(-2)$, $f(-1)$, and $f(0)$ separately and then add them up. Let me write down each step carefully. Starting with $f(-2)$: $f(-2) = (3*(-2) - 2) / (-2 - 2) = (-6 - 2) / (-4) = (-8) / (-4) = 2$. Next, $f(-1)$: $f(-1) = (3*(-1) - 2) / (-1 - 2) = (-3 - 2) / (-3) = (-5) / (-3) = 5/3$. Then, $f(0)$: $f(0) = (3*0 - 2) / (0 - 2) = (-2) / (-2) = 1$. Now, I need to add these together: $2 + 5/3 + 1$. To add these, it might help to convert them all to fractions with a common denominator. 2 is $6/3$, $5/3$ stays the same, and 1 is $3/3$. So adding them together: $6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 = 14/3$.

Final Response : 310 Tokens

To find the value of $(f(-2) + f(-1) + f(0))$ where $f(x) = (3x - 2)/(x - 2)$: First, we evaluate $f(-2)$: $[f(-2) = (3(-2) - 2)/(-2 - 2) = (-6 - 2)/(-4) = (-8)/(-4) = 2]$ Next, we evaluate $f(-1)$: $[f(-1) = (3(-1) - 2)/(-1 - 2) = (-3 - 2)/(-3) = (-5)/(-3) = 5/3]$ Then, we evaluate $f(0)$: $[f(0) = (3(0) - 2)/(0 - 2) = (-2)/(-2) = 1]$ Now, we add these values together: $[f(-2) + f(-1) + f(0) = 2 + 5/3 + 1]$ Converting all terms to fractions with a common denominator: $[2 = 6/3, 1 = 3/3]$ $[6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 = 14/3]$ Thus, the value of $(f(-2) + f(-1) + f(0))$ is **14/3**.

Origin: 1614 Tokens **Thinking Length: 586 Tokens, Valid Ratio: 100%**

Valid Thinking : 650 Tokens

Alright, so I have this function $f(x) = (3x - 2)/(x - 2)$, and I need to find the value of $f(-2) + f(-1) + f(0)$. Hmm, okay, let's break this down step by step. I think the best way is to calculate each function value separately and then add them up. That seems straightforward enough. First, let me figure out $f(-2)$. Plugging $x = -2$ into the function, I get: $f(-2) = (3*(-2) - 2)/(-2 - 2) = (-6 - 2)/(-4) = (-8)/(-4) = 2$. Calculating the numerator: $3*(-2) = -6$, minus 2 is -8 . Calculating the denominator: $(-2) - 2 = -4$. So $f(-2) = (-8)/(-4)$. Hmm, a negative divided by a negative is a positive, so that's 2. Okay, so $f(-2)$ is 2. Not too bad. Next, let's compute $f(-1)$. Plugging $x = -1$ into $f(x)$: $f(-1) = (3*(-1) - 2)/(-1 - 2) = (-3 - 2)/(-3) = (-5)/(-3) = 5/3$. Numerator: $3*(-1) = -3$, minus 2 is -5 . Denominator: $(-1) - 2 = -3$. So $f(-1) = (-5)/(-3)$. Again, negative divided by negative is positive, so that simplifies to $5/3$. Alright, so $f(-1)$ is $5/3$. Moving on to $f(0)$. Plugging $x = 0$ into the function: $f(0) = (3*0 - 2)/(0 - 2) = (-2)/(-2) = 1$. Numerator: $3*0 = 0$, minus 2 is -2 . Denominator: $0 - 2 = -2$. So $f(0) = (-2)/(-2)$. That's a positive 1. So $f(0)$ is 1. Now, adding them all up: $f(-2) + f(-1) + f(0) = 2 + 5/3 + 1$. Wait, let me make sure I did that correctly. Let me check each computation again to be thorough. First, $f(-2)$: $(3*(-2) - 2)/(-2 - 2) = (-6 - 2)/(-4) = (-8)/(-4) = 2$. Yes, that's correct. Next, $f(-1)$: $(3*(-1) - 2)/(-1 - 2) = (-3 - 2)/(-3) = (-5)/(-3) = 5/3$. That seems right. Lastly, $f(0)$: $(3*0 - 2)/(0 - 2) = (-2)/(-2) = 1$. Perfect. Now, adding them: $2 + 5/3 + 1$. $2 + 1$ is 3, so $3 + 5/3$. Converting 3 to thirds, it's $9/3$, so $14/3$.

Invalid Thinking : 671 Tokens

So, **14/3** is the total. Wait, is that right? Let me check again. 2 is equal to $6/3$, and 1 is equal to $3/3$. So, $6/3 + 5/3 + 3/3$. That would be $(6 + 5 + 3)/3 = 14/3$. Yes, that's correct. Alternatively, I could have converted each term to fractions with a common denominator: $f(-2) = 2 = 6/3$, $f(-1) = 5/3$, $f(0) = 1 = 3/3$. Adding them: $6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 = 14/3$. So, **14/3** is indeed the sum. Wait, but just to make sure, let me go back through each step one more time. Calculating $f(-2)$: $3*(-2) = -6$, $-6 - 2 = -8$, Denominator: $(-2) - 2 = -4$, $-8 / -4 = 2$. Correct. $f(-1)$: $3*(-1) = -3$, $-3 - 2 = -5$, Denominator: $(-1) - 2 = -3$, $-5 / -3 = 5/3$. Correct. $f(0)$: $3*0 = 0$, $0 - 2 = -2$, Denominator: $0 - 2 = -2$, $-2 / -2 = 1$. Correct. So, adding them up: 2 (which is $6/3$) $+ 5/3 + 1$ (which is $3/3$). $6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 = 14/3$. Yep, that's correct. So, I think that's the right answer. $14/3$ as a common fraction. I don't think it can be simplified further because 14 and 3 share no common divisors other than 1. Alternatively, if I convert $14/3$ to a mixed number, it's 4 and $2/3$, but the question specifically asks for a common fraction, so $14/3$ is the appropriate form. Just to make sure I didn't make any arithmetic errors, let me go through the addition once more: $2 + 5/3 + 1$. $2 + 1$ is 3, 3 is equal to $9/3$, $9/3 + 5/3$ is $14/3$. Alternatively, 2 is $6/3$, $6/3 + 5/3$ is $11/3$, $11/3 + 3/3$ is $14/3$. Both ways, I get **14/3**. So, I think I've got it right. I don't see any mistakes in my reasoning, so I'm confident that the answer is $14/3$. Final Answer: The value of $f(-2) + f(-1) + f(0)$ is **14/3**.

Final Response : 293 Tokens

To find the value of $f(-2) + f(-1) + f(0)$ for the function $f(x) = (3x - 2)/(x - 2)$ we calculate each function value separately: 1. Calculate $f(-2)$: $f(-2) = (3(-2) - 2)/(-2 - 2) = (-6 - 2)/(-4) = -8/-4 = 2$. Calculate $f(-1)$: $f(-1) = (3(-1) - 2)/(-1 - 2) = (-3 - 2)/(-3) = -5/-3 = 5/3$. Calculate $f(0)$: $f(0) = (3(0) - 2)/(0 - 2) = (-2)/(-2) = -2/-2 = 1$ Adding these values together: $f(-2) + f(-1) + f(0) = 2 + 5/3 + 1$ Convert each term to a common fraction: $2 = 6/3$ $1 = 3/3$ Sum the fractions: $6/3 + 5/3 + 3/3 = (6 + 5 + 3)/3 = 14/3$ Thus the value of $f(-2) + f(-1) + f(0)$ is **14/3**

Thinking Length: 1321 Tokens, Valid Ratio: 49%

Figure 10: Case study comparing LC-R1 with the original model.