LEASH: Adaptive Length Penalty and Reward Shaping for Efficient Large Reasoning Model

Yanhao Li[1,2,*], Lu Ma[2,*], Jiaran Zhang[3], Lexiang Tang[1,2], Wentao Zhang[2], Guibo Luo[1,2,†],

[1]Guangdong Provincial Key Laboratory of Ultra High Definition Immersive Media Technology, Shenzhen Graduate School, Peking University, [2]Peking University, [3]Harbin Institute of Technology, Shenzhen, China

[*]Equal contribution, [†]Corresponding author

Correspondence: liyanhao@stu.pku.edu.cn, luoguibo@pku.edu.cn

Abstract

Large Language Models (LLMs) often produce unnecessarily lengthy reasoning traces, which significantly increase computational cost and latency. Existing approaches typically rely on fixed length penalties, but such penalties are hard to tune and fail to adapt to the evolving reasoning abilities of LLMs, leading to suboptimal trade-offs between accuracy and conciseness. To address this challenge, we propose LEASH *(adaptive LEngth penAlty and reward SHaping)*, a reinforcement learning framework for efficient reasoning in LLMs. We formulate length control as a constrained optimization problem and employ a Lagrangian primal–dual method to dynamically adjust the penalty coefficient. When generations exceed the target length, the penalty is intensified; when they are shorter, it is relaxed. This adaptive mechanism guides models toward producing concise reasoning without sacrificing task performance. Experiments on Deepseek-R1-Distill-Qwen-1.5B and Qwen3-4B-Thinking-2507 show that LEASH reduces the average reasoning length by 60% across diverse tasks—including in-distribution mathematical reasoning and out-of-distribution domains such as coding and instruction following—while maintaining competitive performance. Our work thus presents a practical and effective paradigm for developing controllable and efficient LLMs that balance reasoning capabilities with computational budgets. Our code is available at https://github.com/YanhaoLi-Cc/LEASH.

1 Introduction

Large Reasoning Models (LRMs), driven by reinforcement learning (RL), have substantially improved reasoning capabilities by generating longer chains of thought (CoT) to solve complex reasoning tasks (Guo et al., 2025; Jaech et al., 2024; OpenAI, 2025). However, this enhancement often comes with *token over-expansion* and *redundant self-reflection*, leading to excessively long outputs and high computational costs (Chen et al., 2025).

To mitigate the issue of over-thinking, recent research has focused on two directions: truncation-based CoT compression (Muennighoff et al., 2025; Yang et al., 2025b; Chen et al., 2025) and RL-based length-aware optimization (Team et al., 2025b; Aggarwal and Welleck, 2025; Hou et al., 2025; Liu et al., 2025; Arora and Zanette, 2025; Su and Cardie, 2025). The former directly limits the reasoning length by imposing a fixed generation budget or early termination, which effectively reduces computation but often cuts off essential intermediate reasoning steps, resulting in accuracy degradation. In contrast, RL-based methods explicitly incorporate length-aware objectives into the policy optimization process to dynamically balance accuracy and efficiency. For example, THINKPRUNE enforces a strict generation-length cap during RL training, where tokens exceeding the limit are discarded (Hou et al., 2025), while L-CPO formulates the task as a constrained optimization problem, introducing penalty terms to explicitly control the target generation length (Aggarwal and Welleck, 2025). Although these methods demonstrate effectiveness in reducing overlong reasoning, they typically rely on fixed constraints or manually tuned penalty strengths, making them unable to adapt to the evolving reward distributions and reasoning dynamics during RL training. Consequently, they often suffer from unstable trade-offs: over-penalization suppresses necessary reasoning steps and harms accuracy, whereas under-penalization fails to constrain verbosity. This highlights the need for an adaptive mechanism that can dynamically adjust penalty strength according to real-time constraint satisfaction, enabling the model to regulate reasoning length without external tuning.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2846–2856

To this end, we propose **LEASH** (LEngth penAlty and reward SHaping), an RL-based adaptive length control method. LEASH formulates reasoning-length control as a constrained optimization problem, aiming to maximize task reward while satisfying the expected length constraint. By introducing a dual variable λ that adaptively adjusts the penalty strength based on the degree of constraint violation, LEASH can flexibly "tighten" or "loosen" its "leash" on the output length—preserving critical reasoning steps while suppressing unnecessary verbosity.

To ensure training stability, we further design a one-sided penalized reward that penalizes only over-length generations while avoiding incentives for overly short outputs that could collapse model behavior. The dual variable λ is updated via a primal–dual optimization mechanism, forming a feedback loop that automatically balances task reward and constraint satisfaction throughout training. Unlike fixed-constraint RL methods, this adaptive process continuously adjusts the degree of length control according to the evolving reasoning behavior, effectively preventing both over-penalization and under-penalization, and improving training stability and convergence.

We conduct comprehensive experiments on both mathematical and general reasoning benchmarks, including AIME24, AIME25, HMMT25, AMC23, GPQA, and MMLU-Pro, covering reasoning models of 1.5B and 4B scales. Experimental results show that LEASH achieves a significantly better efficacy–efficiency trade-off than existing methods. On DeepSeek-R1-Distill-Qwen-1.5B, LEASH reduces the average generation length by 62.7% while improving accuracy by 0.8 points. Moreover, it maintains robust out-of-domain generalization on GPQA and MMLU-Pro. Training dynamics analysis reveals that LEASH rapidly satisfies the length constraint in the early training stage and progressively stabilizes the dual coefficient λ, leading to smoother convergence and higher reasoning efficiency. Further behavioral analysis shows that the generated CoTs become more concise and focused, with redundant "rethink" or self-reflective phrases substantially reduced, while preserving essential planning and summarization structures.

2 Method

Overview: We propose the adaptive **LE**ngth pen**A**lty and reward **SH**aping (**LEASH**) method for efficient reasoning model training. This algorithm extends the DAPO (Yu et al., 2025) framework by introducing a Lagrangian constraint mechanism to effectively control the length of generated sequences while optimizing task performance. Concise pseudocode is provided in Algorithm 1.

Problem Formulation: To control generation length while maximizing the task objective, we formulate the problem as a constrained optimization task. The primary goal is to train a policy π_θ that maximizes the expected task-specific reward, while ensuring that the generation length $L(y)$ does not exceed a predefined target length L_t. The objective can be formally expressed as:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[r(x,y)] \\ \text{s.t.} \quad & L(y) \le L_\mathrm{t}, \quad \forall y \sim \pi_\theta(\cdot|x), \end{aligned} \tag{1}$$

where $\mathcal{D}$ is a distribution of prompts, y is the response generated by the LLM π_θ. The task-specific reward function $r(x, y)$ provides a binary signal based on correctness, which utilizes a boolean function $\mathrm{is\_equivalent}(y, y^*)$ to determine if the generated response y is semantically or functionally equivalent to the reference answer y^*:

$$r(x,y) = \begin{cases} +1, & \text{if } \mathrm{is\_equivalent}(y,y^*) \text{ is true,} \\ -1, & \text{otherwise.} \end{cases} \tag{2}$$

However, the constraint in equation 1 entails the challenge of guaranteeing length satisfaction for all possible responses, which is not straightforward to enforce directly with RL methods. In light of this, we reformulate the strict length constraint into an expectation form. This modification introduces a more tractable objective that aims to control the average generation length. Our surrogate objective is presented as follows:

$$\underset{\theta}{\text{maximize}} \; J_R(\theta), \quad \text{s.t.} \quad J_P(\theta) \le 0, \tag{3}$$

where

$$J_R(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[r(x,y)\big], \tag{4}$$

$$J_P(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)}\left(\frac{L(y)}{L_\mathrm{t}} - 1\right). \tag{5}$$

To address this constrained problem, we leverage the Lagrangian method, a technique for finding the local maxima and minima of a function over a constraint set. This allows us to convert the constrained primal problem, as defined in equation 3,

Algorithm 1 LEASH: Adaptive Length Penalty and Reward Shaping

Input: dataset $\mathcal{D}$, target length L_t, policy π_θ
Hyper: $\alpha_\theta, \alpha_\lambda, [\lambda_{\min}, \lambda_{\max}], G$
Initialize θ, $\lambda \geq 0$
while not converged **do**
  Sample prompts $\{x_b\}_{b=1}^{B} \sim \mathcal{D}$; set $\pi_\theta^{\text{old}} \leftarrow \pi_\theta$
  for $b = 1 \to B$ **do**
    Sample G responses $y_{b,1:G} \sim \pi_\theta^{\text{old}}(\cdot|x_b)$ and lengths $L(y_{b,i})$
    Task reward $r_{b,i} \in \{-1, +1\}$ via is_equivalent
    One-sided penalty $\Delta_{b,i} = \max\big(0, \frac{L(y_{b,i})}{L_t} - 1\big)$
    Shaped reward $\tilde{r}_{b,i} = \text{clip}\big(r_{b,i} - \lambda\, \Delta_{b,i}, -1, 1\big)$
    Normalize advantages $\hat{A}_{b,i} = (\tilde{r}_{b,i} - \mu_b)/(\sigma_b + 10^{-8})$
  end for
  Policy update: maximize DAPO objective using $\hat{A}_{b,i}$; update $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta J(\theta)$
  Estimate constraint violation $\widehat{J_P} = \frac{1}{BG} \sum_{b,i} (\frac{L(y_{b,i})}{L_t} - 1)$
  Dual update: $\lambda \leftarrow \text{clip}\big(\lambda + \alpha_\lambda\, \widehat{J_P},\ \lambda_{\min},\ \lambda_{\max}\big)$
end while
return π_θ, λ

into its unconstrained saddle-point problem. We define the Lagrangian function $\mathcal{L}(\theta, \lambda)$ as:

$$\mathcal{L}(\theta, \lambda) = J_R(\theta) - \lambda \cdot J_P(\theta), \tag{6}$$

where $\lambda \geq 0$ serves as the Lagrange multiplier. The goal is to find a saddle point (θ^*, λ^*) that solves the max-min problem:

$$\max_{\theta} \min_{\lambda \geq 0} \mathcal{L}(\theta, \lambda). \tag{7}$$

This equation encapsulates our primary goal, maximizing the task reward $J_R(\theta)$ while minimizing the length penalty $J_P(\theta)$, thereby encouraging the model to generate high-quality outputs of a controlled length. We use a Primal-Dual algorithm to find this saddle point by alternately updating the primal variable θ and the dual variable λ.

Policy Update. In each iteration, we fix λ and maximize $\mathcal{L}(\theta, \lambda)$ by performing one step of gradient ascent on θ. Theoretically, the Lagrangian objective can be expressed as the expectation of an augmented reward signal:

$$\begin{aligned}\mathcal{L}(\theta, \lambda) &= J_R(\theta) - \lambda\, J_P(\theta) \\ &= \mathbb{E}_{x \sim \mathcal{D},\, y \sim \pi_\theta(\cdot|x)} \left[r(x, y) - \lambda \left(\frac{L(y)}{L_t} - 1 \right) \right].\end{aligned} \tag{8}$$

This formulation implies an augmented reward:

$$r'(x, y) = r(x, y) - \lambda \cdot \left(\frac{L(y)}{L_t} - 1 \right). \tag{9}$$

However, directly optimizing this signal can be problematic: if a sequence is much shorter than the target $L(y) \ll L_t$, the term $-\lambda \cdot ({}^{L(y)}/_{L_t} - 1)$ becomes a large positive bonus. This would incorrectly incentivize the policy to generate overly short responses, potentially harming task performance. We modify the reward signal for the policy update to be a one-sided penalized reward, which we denote as $r''(x, y)$. This practical adjustment ensures that we only penalize constraint violations without unintentionally rewarding conservative behavior, resulting in the modified reward $r''(x, y)$:

$$r''(x, y) = r(x, y) - \lambda \cdot \max\left(0, \frac{L(y)}{L_t} - 1\right). \tag{10}$$

Meanwhile, we also clip the reward signal to $[-1, 1]$ to prevent extreme rewards caused by fluctuations in the generated length $L(y)$ from triggering destructive gradient updates. The penalized reward is defined as:

$$r'''(x, y) = \text{clip}\big[r''(x, y), -1,\ 1\big]. \tag{11}$$

To maximize $r'''(x, y)$, we reframe this optimization problem as a standard RL task. We then apply the DAPO algorithm (Yu et al., 2025), using $r'''(x, y)$ as the reward signal to optimize policy loss $J(\theta)$. Notably, our approach deviates from the original DAPO implementation by omitting the dynamic sampling and overlong penalty techniques.

Dual Variable Update. After updating the policy parameters, we fix the policy parameters θ and up-

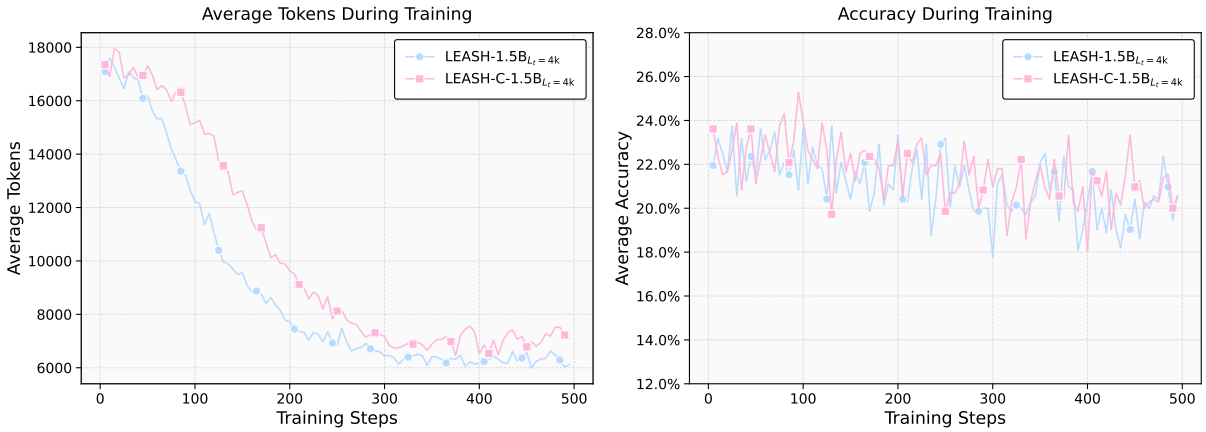

Figure 1: Training dynamics of LEASH and LEASH-C under the target length $L_t = 4$k on the 1.5B base. Left: average tokens per response vs. training steps—LEASH shortens trajectories faster and stabilizes at a lower length with smaller drift. Right: average accuracy vs. training steps—both variants remain comparable, indicating that length compression does not degrade task accuracy.

date the dual variable. The update is performed via a gradient ascent step on the dual objective, which corresponds to adjusting λ based on the constraint violation term $J_P(\theta)$. The gradient of the Lagrangian with respect to λ is:

$$\begin{aligned}\nabla_\lambda \mathcal{L}(\theta, \lambda) &= -J_P(\theta) \\ &= -\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)} \left(\frac{L(y)}{L_{\mathrm{t}}} - 1 \right). \end{aligned} \tag{12}$$

Accordingly, the update rule applies this gradient. For practical stability, we also clip the resulting value to a predefined range $[\lambda_{min}, \lambda_{max}]$:

$$\lambda_{k+1} = \mathrm{clip}[\lambda_k + \alpha_\lambda \cdot J_P(\theta_{k+1}),\ \lambda_{\min},\ \lambda_{\max}]. \tag{13}$$

This update rule creates an adaptive adjustment mechanism for the penalty coefficient λ. The direction of the adjustment is determined by the sign of the constraint violation term $J_P(\theta_{k+1})$. If the average length exceeds the target ($J_P(\theta_{k+1}) > 0$), λ increases to intensify the penalty on length in the next policy update step. Conversely, if the average length is below the target ($J_P(\theta_{k+1}) < 0$), λ decreases to relax the penalty, shifting the optimization focus back to maximizing the primary objective J_R.

By alternating between these primal and dual updates, the algorithm dynamically adjusts both the policy and the penalty strength to find an equilibrium that maximizes the task reward while satisfying the average length constraint.

Variant Label	λ Specification
LEASH-C-1.5B$_{L_t=4k}$	$\lambda = 0.1$
LEASH-1.5B$_{L_t=4k}$	$\lambda_{\text{init}} = 0.1, \lambda_{\text{lr}} = 0.005$
LEASH-C-4B$_{L_t=12k}$	$\lambda = 0.2$
LEASH-4B$_{L_t=12k}$	$\lambda_{\text{init}} = 0.2, \lambda_{\text{lr}} = 0.005$
LEASH-C-4B$_{L_t=8k}$	$\lambda = 0.2$
LEASH-4B$_{L_t=8k}$	$\lambda_{\text{init}} = 0.2, \lambda_{\text{lr}} = 0.05$

Table 1: Penalty settings for LEASH variants at target lengths L_t. The suffix "-C" indicates a *constant* penalty (fixed λ), while variants without "-C" use *adaptive* dual updates with initialization λ_{init} and step size λ_{lr}. Rows labeled *1.5B* are based on *DeepSeek-R1-Distill-Qwen-1.5B*, and rows labeled *4B* are based on *Qwen3-4B-Thinking-2507*. The table reports the exact λ or $(\lambda_{\text{init}}, \lambda_{\text{lr}})$ used for each configuration.

3 Experiments

3.1 Experimental Setup

Models and Datasets. Representative open-source long-reasoning LLMs include DeepSeek-R1 (Guo et al., 2025) and Qwen3-235B-A22B-Thinking (Yang et al., 2025a), along with their distilled variants. In our experiments, we adopt two representative models from these families: DeepSeek-R1-Distill-Qwen-1.5B[1] and Qwen3-4B-Thinking-2507[2]. For training, we construct a refined subset of the DAPO-MATH-17k dataset[3] (Yu

[1] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B

[2] https://huggingface.co/Qwen/Qwen3-4B-Thinking-2507

[3] https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

Model	AIME24		AIME25		HMMT25		AMC		Overall Performance			
	Avg. @32	Avg. Tokens	Avg. @32	Avg. Tokens	Avg. @32	Avg. Tokens	Avg. @32	Avg. Tokens	Avg. Acc.	Avg. Tokens	Δ. Acc.	Δ. Tokens(%)
Based on DeepSeek-R1-Distill-Qwen-1.5B												
Original Model	31.4	16,722	23.1	16,562	14.5	18,521	63.3	11,103	33.1	15,727	*Baseline*	
L1-Qwen-1.5B-Max	28.3	3,140	19.6	2,948	11.7	3,074	68.2	2,413	32.0	2,893	-1.1	-81.6%
L1-Qwen-1.5B-Extra	27.9	2,843	20.4	2,577	8.3	2,597	69.1	2,226	31.4	2,561	-1.6	-83.7%
LASER-DE$_{L_t=4096}$	32.8	8,115	22.8	7,690	14.9	8,808	67.7	4,849	34.6	7,366	+1.5	-53.2%
THINKPRUNE$_{4k\rightarrow 3k\rightarrow 2k}$	25.9	5,999	21.1	5,419	12.5	6,607	65.9	3,451	31.4	5,369	-1.7	-65.9%
LEASH-C-1.5B$_{L_t=4k}$	27.9	7,587	23.0	6,883	11.3	7,094	63.7	4,976	31.5	6,635	-1.6	-57.8%
LEASH-1.5B$_{L_t=4k}$	30.4	6,779	24.6	6,126	14.2	6,358	66.4	4,230	33.9	5,873	**+0.8**	**-62.7%**
Based on Qwen3-4B-Thinking-2507												
Original Model	80.8	19,193	74.6	21,414	52.9	24,645	93.8	12,961	75.5	19,553	*Baseline*	
LEASH-C-4B$_{L_t=12k}$	78.8	15,941	73.3	18,042	49.0	19,720	92.7	10,110	73.5	15,953	-2.1	-18.4%
LEASH-4B$_{L_t=12k}$	79.7	14,211	73.3	16,700	51.9	17,926	93.3	8,873	74.6	14,428	**-1.0**	**-26.2%**
LEASH-C-4B$_{L_t=8k}$	74.6	12,127	67.5	14,044	46.3	14,569	91.0	7,548	69.9	12,072	-5.7	-38.3%
LEASH-4B$_{L_t=8k}$	76.6	12,086	66.5	14,135	46.0	14,671	91.1	7,277	70.1	12,042	-5.5	-38.4%

Table 2: In-domain mathematical reasoning performance across four benchmarks (AIME24, AIME25, HMMT25, and AMC). Positive changes in accuracy and reductions in token usage are highlighted in green, while degradations are marked in red. LEASH consistently achieves the best efficacy–efficiency trade-off, substantially shortening reasoning trajectories while maintaining or improving accuracy compared with fixed-penalty (LEASH-C) and prior baselines such as L1-Qwen-1.5B-Max/Extra, THINKPRUNE and LASER-DE.

et al., 2025) by filtering out prompts where Qwen3-4B-Thinking-2507 consistently succeeds or consistently fails under pass@4 evaluation, yielding 3,939 instances used in our experiments.

Training configuration. The training is conducted using the VeRL (Sheng et al., 2024) framework's DAPO training loop, with overlong reward shaping and dynamic sampling disabled. We train on 32 NVIDIA H20 GPUs with a global batch size of 64, using Adam optimizer with a learning rate of 1×10^{-6} and a sampling temperature of 1.0. KL regularization is not applied, and the entropy coefficient is set to 0.0. Rewards are clipped to the range $[-1, 1]$, with DAPO clipping thresholds set asymmetrically at $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, while the dual variable is bounded by $\lambda_{\min} = 0.0$ and $\lambda_{\max} = 1.0$. The maximum context length during training is 32k tokens, with 1k tokens reserved for the prompt.

Baselines and Variants. We include the original base checkpoints as baselines for comparison. Additionally, we re-evaluate the open-sourced checkpoints of L1-Qwen-1.5B-Max/Extra (Aggarwal and Welleck, 2025), THINKPRUNE (Hou et al., 2025) and LASER-DE (Liu et al., 2025) within our evaluation protocol. To investigate the effect of the length constraint and dual updates, we also train variants under different target lengths L_t. For clarity, we refer to the runs with a constant penalty as LEASH-C, and those with the adaptive dual update as the LEASH variant. The specific schedules for the penalty (λ) applied to each target L_t are shown in Table 1. All other hyperparameters align with the training configuration outlined earlier.

Evaluation protocol. We evaluate our checkpoints on several widely used reasoning benchmarks, including AIME 24, AIME 25, HMMT 25 (Balunović et al., 2025) and AMC 23, with 32 samples per prompt for the first four datasets. We also include out-of-domain sets GPQA (Rein et al., 2023) and MMLU-Pro (Wang et al., 2024) in our evaluation. The maximum generation length is set to 32,768 tokens, which includes both the reasoning budget tokens and final answer tokens, aligning with the training phase. Unless specified otherwise, we use a sampling temperature of $T = 0.6$, top-$p = 0.95$, and top-k disabled.

4 Results and Analysis

4.1 Main Results

We summarize the main experimental results of our method in Table 2 and Table 3, which present the model performance on in-domain mathematical reasoning benchmarks and out-of-domain general reasoning tasks, respectively. Beyond the aggregate results, we also examine how LEASH evolves during training. As shown in Figure 1, the adaptive

variant reduces average tokens more quickly and settles at a lower plateau than the constant-penalty version, indicating faster and steadier length control.

Model	GPQA		MMLU-Pro		Overall Performance			
	Acc	Avg. Tokens	Acc	Avg. Tokens	Avg. Acc.	Avg. Tokens	Δ. Acc	Δ. Tokens(%)
Based on DeepSeek-R1-Distill-Qwen-1.5B								
Original Model	17.2	10,943	18.9	6,099	18.0	8,521	*Baseline*	
LEASH-C-1.5B$_{L_t=4k}$	22.3	4,379	19.4	2,619	20.8	3,499	+2.8	-58.9%
LEASH-1.5B$_{L_t=4k}$	22.7	3,908	19.7	2,222	21.2	3,064	+3.2	-64.0%
Based on Qwen3-4B-Thinking-2507								
Original Model	65.8	9,082	74.0	4,742	69.9	6,912	*Baseline*	
LEASH-C-4B$_{L_t=12k}$	68.2	7,435	74.1	4,049	71.2	5,742	+1.3	-16.9%
LEASH-4B$_{L_t=12k}$	66.7	7,182	74.1	3,900	70.4	5,541	+0.5	-19.8%
LEASH-C-4B$_{L_t=8k}$	61.1	6,311	73.8	3,510	67.5	4,910	-2.4	-29.0%
LEASH-4B$_{L_t=8k}$	64.6	6,302	73.9	3,406	69.3	4,854	-0.6	-29.8%

Table 3: Out-of-domain (OOD) general reasoning performance on GPQA and MMLU-Pro. Positive accuracy gains and reductions in token usage are highlighted in green, while degradations are shown in red. We report per-benchmark accuracy and average tokens, along with overall averages and deltas.

In-domain reasoning performance. As shown in Table 2, the proposed LEASH series achieves a favorable efficacy–efficiency trade-off, substantially shortening the generation length while maintaining or even improving reasoning accuracy. For the 1.5B model, LEASH-1.5B$_{L_t=4k}$ reduces the generation length by **62.7%** while improving accuracy by **+0.8** points over the original model, clearly outperforming THINKPRUNE and LASER-DE under the same length constraint. For the 4B model, the adaptive dual-update variants demonstrate the best overall balance. Specifically, LEASH-4B$_{L_t=12k}$ achieves a **26.2%** reduction in output length while maintaining **74.6%** average accuracy, only 1.0 points below the base model, showing stable performance even under aggressive constraints. Across all experiments, adaptive variants consistently outperform the constant-penalty ones, confirming that dynamically adjusting the dual variable λ effectively stabilizes training and prevents over-penalization.

Out-of-domain generalization. We further evaluate the models on GPQA and MMLU-Pro to assess generalization beyond the training distribution, as shown in Table 3. The results show that LEASH significantly improves efficiency while maintaining competitive accuracy. On Qwen3-4B-Thinking-2507, LEASH-4B$_{L_t=12k}$ improves the average accuracy by **+1.3 points** while reducing generation length by **16.9%**, suggesting that moderate length constraints can enhance focus and consistency in reasoning. When the constraint is further tightened to 8k, LEASH-4B$_{L_t=8k}$ still achieves about 29% token reduction with only minor performance degradation, demonstrating strong robustness across domains.

4.2 Behavior over Training

We further analyze the training behaviors of LEASH to better understand how the dual updates interact. Figure 2 presents the training dynamics of the proposed LEASH and its constant-penalty variant (LEASH-C) under the target length $L_t = 4$k. Four key metrics are tracked: the satisfaction rate of the length constraint, the adaptive penalty coefficient λ, the effective penalty value, and the average number of generated tokens during training.

In the early phase of training (approximately the first 100 steps), LEASH exhibits a sharp increase in satisfaction rate, indicating that the model quickly learns to generate responses within the target length range. During this period, since the average length initially exceeds the target, the penalty coefficient λ gradually increases to enforce stronger constraint pressure. Around step 150, λ reaches its peak and then steadily decreases as the model stabilizes near the target length. By approximately step 450, λ approaches zero, suggesting that the model has effectively satisfied the constraint and no longer requires further penalty adjustment. The penalty term follows a similar pattern: it rises early in training as the system strengthens length control, then progressively decays as the generated outputs converge toward the desired token budget. The average token count decreases sharply from around 9k tokens to below 3k and eventually stabilizes near 2k by the end of training, corresponding to a reduction of more than 75% relative to the initial generation length.

In contrast, the constant-penalty variant LEASH-C maintains a fixed λ, resulting in a smoother but less adaptive penalty schedule. While both methods ultimately achieve near-perfect satisfaction rates (close to 100%), the adaptive version converges faster and demonstrates more stable control. These observations highlight LEASH's ability to dynamically adjust its penalty coefficient in response to constraint violations, thereby achieving a stable balance between generation efficiency (shorter outputs) and constraint satisfaction.

4.3 Changes of Thinking Patterns

To better understand the mechanisms behind the changes in response length, we analyze the evolu-

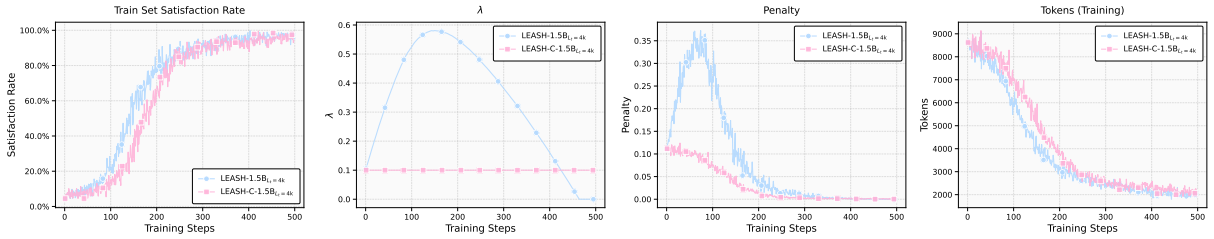

Figure 2: Training dynamics of LEASH and LEASH-C under $L_t = 4$k. The plots show (from left to right): (1) satisfaction rate on the training set, (2) adaptive penalty coefficient λ, (3) effective penalty value, and (4) average token length. LEASH dynamically adjusts λ to accelerate convergence and stabilize constraint satisfaction.

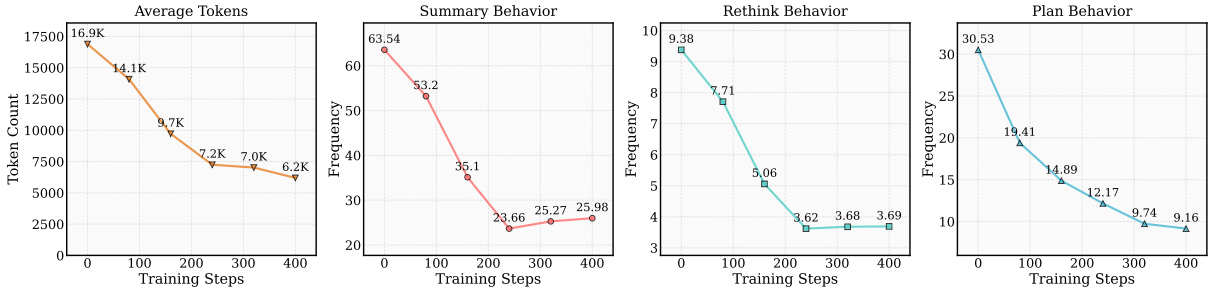

Figure 3: Evolution of the model's thinking patterns across RL iterations on AIME2024 and AIME2025, using DeepSeek-R1-Distill-Qwen-1.5B as the base model. The average response length (left) consistently decreases throughout training, indicating progressive compression of reasoning trajectories. Correspondingly, the frequencies of summary-, rethink-, and plan-related keywords (right) exhibit distinct dynamics: summary and rethink behaviors decline sharply before stabilizing with slight rebounds, while plan behaviors continuously diminish.

tion of the model's thinking patterns across RL iterations on the AIME2024 and AIME2025 datasets, as shown in Figure 3. For each question, 16 samples are generated, revealing substantial shifts in the model's reasoning dynamics during training. The average response length steadily decreases from 16.9K to 6.2K tokens, indicating that the model gradually learns to produce more concise reasoning trajectories while maintaining completeness. As the responses become shorter, different types of reasoning behaviors also exhibit systematic changes. *Summary-related* keywords (e.g., "So", "Therefore", "Thus") show an overall decreasing trend, dropping from 63.5 to about 24, but slightly rising and stabilizing around 26 in the later stages. This suggests that the model rapidly reduces redundant summarization in the early phase while later reintroducing moderate summarization behaviors to maintain the completeness of conclusions. Notably, despite the slight increase in keyword frequency, the overall responses remain shorter, indicating that the model retains essential structural conclusions while achieving a balance between brevity and completeness. *Rethink-related* expressions (e.g., "recheck", "re-evaluate") decrease from 9.38 to 3.62 and then slightly rebound to 3.69, showing that the model reduces excessive self-verification while preserving necessary reflective mechanisms. This aligns with our optimization objective, as the goal of the LEASH algorithm's RL training is to reduce redundant and inefficient reflection rather than suppress reflective reasoning entirely. The minor rebound indicates that the model reintroduces limited reflective behaviors after compression to maintain self-correction capability, achieving a balance between conciseness and robustness. Meanwhile, *plan-related* markers (e.g., "First", "Second", "Step") consistently decrease from 30.5 to 9.2, suggesting that explicit step-by-step planning is gradually replaced by more implicit and compact procedural reasoning. We provide the full list of tracked keywords for each reasoning behavior category in Appendix A.1.

5 Related Works

5.1 RL-Driven Emergence of Reasoning Abilities

Reinforcement learning has recently emerged as a promising approach for enhancing the reasoning capabilities of LLMs. Early breakthroughs, such as OpenAI o-series (Jaech et al., 2024; OpenAI, 2025), DeepSeek R1 (Guo et al., 2025), and Kimi

k-series (Team et al., 2025b,a, 2026), have demonstrated that applying Reinforcement Learning with Verifiable Rewards (RLVR) can lead to significant improvements in complex reasoning tasks.

Building on these foundations, Light-r1 (Wen et al., 2025) incorporates curriculum learning to refine reasoning skills, while Logic-rl (Xie et al., 2025) adopted logic-driven reward functions to improve general reasoning ability. Deepscaler (Luo et al., 2025) trains models with progressively longer contexts as their performance improves. Based on GRPO, DAPO (Yu et al., 2025) uses clip-higher to prevent entropy collapse, dynamic sampling for improved efficiency, and token-level policy gradient loss with overlong reward shaping to stabilize training. Yue et al. (2025); Yan et al. (2025); Ma et al. (2025) focus on introducing and addressing the concern that RLVR does not impart fundamentally new reasoning abilities to LLMs.

5.2 Efficient Large Reasoning Models

Recent large reasoning models (LRMs) have achieved remarkable performance on complex reasoning tasks by generating long CoTs, enabling effective problem-solving in domains such as mathematics and coding. However, while LRMs significantly improve performance on reasoning tasks, they also cause substantial overhead. Compared to LLMs, reasoning models lead to redundancy across multiple dimensions. Consequently, several methods have been proposed to mitigate this CoT length redundancy (Team et al., 2025b; Feng et al., 2025).

Several studies investigate techniques to control response length during inference time. For example, Hassid et al. (2025) indicates that shorter reasoning chains tend to be more accurate, and suggests voting over the shortest m of k samples. Similarly, Yang et al. (2025b) observes that monitoring model behavior and dynamically terminating reasoning chains enhances both accuracy and efficiency. Additionally, Muennighoff et al. (2025) introduces "budget forcing" to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times, without the need for retraining. These methodologies complement LEASH and can be combined to reduce inference time costs.

Numerous studies focus on training models with shorter correct responses. Several works adopt sample-filtering or preference-based fine-tuning strategies to encourage shorter correct responses (Shrivastava et al., 2025; Team et al., 2025b). Concurrently, other studies introduce explicit length-aware penalties within the reward of RLVR to discourage excessively long reasoning chains. For example, Yu et al. (2025) employs reward shaping to encourage shorter responses; Hou et al. (2025) imposes a token limit during RL, awarding zero rewards beyond this cap and progressively tightening it; Su and Cardie (2025) implements an adaptive direct length penalty that evolves over training to prevent over/or under-compression; Xiang et al. (2025) inversely scales the penalty with the prompt's solve rate; and Aggarwal and Welleck (2025) optimizes accuracy while adhering to a prompt-specified target length by penalizing deviations. Although these methods empirically adopt length penalties, the degree of the penalty does not dynamically adjust according to the model's capabilities. Our method, LEASH, proposes a more flexible penalty strategy from a mathematical perspective, offering advantages in theory and in experiments.

6 Discussion

Choosing the target length L_t. Although L_t is the only task-level hyperparameter of LEASH, a sensible choice matters in practice. Based on our experiments, we recommend one of two strategies. (i) *Anchor on the base model*: measure the average response length $\bar{L}_{\text{base}}$ of the unconstrained base on a small subset of training prompts and set L_t to a fraction of $\bar{L}_{\text{base}}$, trading off accuracy retention against token savings; in our setup this corresponds to L_t=4k for the 1.5B base and $L_t \in \{8\text{k}, 12\text{k}\}$ for the 4B base. (ii) *Anchor on the deployment budget*: when a token or latency budget is dictated by the serving environment, set L_t directly to that budget—LEASH will adaptively trade off verbosity against accuracy within it.

Reduced hyperparameter burden. Once L_t is chosen, the remaining coefficients $(\lambda_{\min}, \lambda_{\max}, \alpha_\lambda)$ require little tuning: the primal–dual update automatically tightens λ when $L(y) > L_t$ and relaxes it otherwise, so a single default setting transfers across model scales and tasks in our experiments. Empirically, L_t is the only knob a user needs to rethink when porting LEASH to a new model or domain.

7 Conclusion

We introduce LEASH, a reinforcement learning framework that frames length control as a con-

strained optimization problem and solves it with a Lagrangian primal–dual scheme. A one-sided length penalty, clipped reward shaping, and an adaptive update of the dual variable λ guide the policy toward concise and accurate reasoning without manual retuning. On the *DeepSeek-R1-Distill-Qwen-1.5B* and *Qwen3-4B-Thinking-2507* bases, LEASH delivers a favorable efficacy–efficiency balance on in-domain mathematical reasoning, as summarized in Table 2, and it generalizes well to GPQA and MMLU-Pro, as reported in Table 3. The adaptive variant cuts tokens by roughly sixty percent on average while preserving, and at times improving, accuracy. Training dynamics in Figure 2 show faster convergence than a constant-penalty counterpart, and the behavior study in Figure 3 indicates that LEASH suppresses redundant summarization and self-reflection yet preserves essential structure. Overall, LEASH offers a practical recipe for building controllable and compute-efficient reasoning models that respect token budgets.

8 Limitations and Future Work

LEASH is simple and broadly applicable. Our study uses two compact backbones and standard benchmarks to keep compute modest and results reproducible. The limitations are mainly about scope. We plan to scale to larger models and to multi-turn/tool-use settings to test generality, and to replace a fixed L_t with a learned or scheduled target that adapts to task difficulty and latency budgets, enabling automatic token allocation without sacrificing accuracy. We will also explore richer rewards and lightweight auto-tuning of λ for stability.

Acknowledgments

This work is supported by National Key Research and Development Program of China (2024YFE0203100), Guangdong Provincial Key Laboratory of Ultra High Definition Immersive Media Technology (Grant No. 2024B1212010006), and Shenzhen Science and Technology Program (JCYJ20230807120800001).

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

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

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms. *Preprint*, arXiv:2412.21187.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. 2025. Don't overthink it. preferring shorter thinking chains for improved llm reasoning. *arXiv preprint arXiv:2505.17813*.

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*.

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*.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. 2025. Learn to reason efficiently with adaptive length-based reward shaping. *Preprint*, arXiv:2505.15612.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, and 1 others. 2025. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Bin Cui, and 1 others. 2025. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *arXiv preprint arXiv:2506.07527*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

OpenAI. 2025. Openai o3 model. https://openai.com/index/introducing-o3-and-o4-mini/.

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. *Preprint*, arXiv:2311.12022.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. 2025. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*.

Jinyan Su and Claire Cardie. 2025. Thinking fast and right: Balancing accuracy and reasoning length with adaptive rewards. *arXiv preprint arXiv:2505.18298*.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, and 307 others. 2026. Kimi k2.5: Visual agentic intelligence. *Preprint*, arXiv:2602.02276.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025a. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025b. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, and 1 others. 2025. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *arXiv preprint arXiv:2506.05256*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

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. *arXiv preprint arXiv:2504.14945*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Zheng Lin, Li Cao, and Weiping Wang. 2025b. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

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

A Appendix

Behavior Category	Representative Keywords
Summary-related	So, Therefore, Thus, conclude, overall
Rethink-related	check again, double-check, re-evaluate, re-examine, reanalyze, reassess, recheck, reconsider, reevaluate, reevaluation, reexamine, rethink, think again, verify again, wait
Plan-related	first, First, Second, second, step, Step

Table 4: Keyword groups used for behavior analysis.

A.1 Keyword Groups for Thinking Behavior Analysis

To analyze the evolution of thinking behaviors, we categorize representative keywords into three groups: *Summary*, *Rethink*, and *Plan*, as shown in Table 4. These keywords are matched case-insensitively within generated responses to compute their frequency across RL iterations.

A.2 Use of LLM Assistants

We used large language models (e.g., ChatGPT, Gemini) solely for language editing (grammar/clarity), light LaTeX tweaks, and occasional debugging hints. All technical ideas, algorithms, experiments, and analyses were designed, implemented, and verified by the authors. No proprietary or personal data were shared with LLM services; only non-sensitive prompts about public benchmarks and settings were used. All results are reproducible from the released code and hyperparameters; LLM-assisted edits do not affect numerical outcomes.