Scale-Aligned Reinforcement Learning for Adaptive Computation Allocation in Reasoning Language Models

Yilin Liu[1], Xuan Wang[1], Zheng Yang[1], Zhuoran Li[1], Fan Wan[2], Chen Wu[2], Xiaojie Wang[1]
[1]Beijing University of Posts and Telecommunications
[2]Central Research Institute, Tongfang Knowledge Network Digital Technology Co., Ltd.
Correspondence: xjwang@bupt.edu.cn

Abstract

Long chain-of-thought (CoT) reasoning in large language models (LLMs) improves performance on complex tasks but incurs substantial inference cost, although many queries do not require extended computation. Existing approaches typically reduce this cost via uniform length penalties or fixed computation budgets, but lack length constraints aligned with the model's own query-dependent distribution of successful reasoning lengths. We propose a reinforcement learning framework that aligns the generated reasoning length with query-dependent successful computation scales estimated from verified trajectories. By providing adaptive guidance derived from the distribution of successful reasoning lengths, our method avoids the limitations of global length regularization. Experiments on mathematical reasoning benchmarks demonstrate improved accuracy–efficiency trade-offs, reducing unnecessary reasoning while maintaining or improving task performance.

1 Introduction

Large language models (LLMs) (OpenAI et al., 2026; Comanici et al., 2025) have achieved strong performance on multi-step reasoning tasks through chain-of-thought (CoT) generation (GLM-5-Team et al., 2026; Yang et al., 2025a). However, reasoning behavior remains highly variable (Wei et al., 2022; Minaee et al., 2025; Chu et al., 2023): even for the same problem, repeated sampling can produce trajectories with dramatically different lengths and structures. This variability suggests that LLMs may adopt different amounts and patterns of reasoning across trajectories, rather than following a single fixed reasoning process.

Controlling computation allocation is challenging. Language models must allocate different amounts of reasoning computation across inputs according to problem complexity. Longer reasoning can improve performance through exploration, self-correction, and verification, but it also increases computational cost and inference latency, and may introduce substantial redundancy (Snell et al., 2024; Plaat et al., 2024). At the same time, not all problems require extended reasoning: simple problems can often be solved with concise computation, while complex problems may benefit from longer deliberation. The key challenge is therefore not simply to shorten reasoning, but to allocate computation appropriately for different problems.

Existing approaches provide only limited control over this behavior (Aggarwal and Welleck, 2025; Team et al., 2025). Most reasoning optimization methods supervise trajectories primarily through final outcomes, without explicitly modeling computation allocation. As a result, efficiency-oriented methods often rely on global length penalties or generic brevity preferences, which can suppress redundant reasoning but may also penalize computation that is genuinely necessary for difficult problems.

In this work, we study whether reasoning length can serve as a useful structural signal for adaptive computation allocation. Our starting point is a simple hypothesis: sampled trajectories for the same problem may correspond to different latent computation paths, and reasoning length provides a coarse observable projection of this structure. Although reasoning length alone is incomplete, we empirically find that trajectories with similar lengths also tend to be structurally similar, while correct trajectories exhibit substantially stronger concentration than incorrect ones. Moreover, successful reasoning often concentrates around one or a few characteristic computation scales for each problem.

Motivated by these observations, we propose a structure-aware reinforcement learning framework that estimates a problem-specific computation scale from correct trajectories and encourages scale-aligned reasoning during policy optimization.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 2323–2337
November 6-10, 2026 ©2026 Association for Computational Linguistics

Unlike global length penalties, our method does not uniformly favor shorter reasoning. Instead, it introduces a lightweight structural supervision signal that encourages the model to allocate computation in a manner consistent with successful reasoning behaviors for each problem.

Our method augments standard outcome-based reinforcement learning without requiring process-level annotations or additional verifier models. Experiments on multiple mathematical reasoning benchmarks show that the proposed approach consistently improves the accuracy-efficiency trade-off, reducing unnecessary computation while maintaining or improving reasoning accuracy.

Our contributions are summarized as follows:

- We propose a structural perspective on reasoning length, viewing sampled trajectories as mixtures of latent computation paths and reasoning length as a coarse projection of underlying computation structure.
- We introduce a structure-aware reinforcement learning framework that estimates problem-specific computation scales from correct trajectories and encourages scale-aligned reasoning during policy optimization.
- We demonstrate across multiple reasoning benchmarks that our method improves reasoning efficiency while maintaining or improving reasoning performance.

2 Related Work

Reasoning Trajectories and Test-Time Computation Chain-of-thought prompting improves complex reasoning by eliciting intermediate steps (Wei et al., 2022). Subsequent methods further enhance performance through self-consistency, tree- or graph-structured search, answer aggregation, and test-time scaling (Wang et al., 2023; Yao et al., 2023; Besta et al., 2024; Snell et al., 2024; Brown et al., 2024; Yang et al., 2026; Muennighoff et al., 2025). These works show that reasoning performance depends strongly on computation allocation. However, sampled trajectories are usually treated as independent candidates for selection or aggregation, rather than as evidence for problem-specific computation patterns. In contrast, we analyze the organization of sampled trajectories and use reasoning length as a projection of latent computation paths.

Supervision for Mathematical Reasoning Outcome-supervised optimization, including RLHF, DPO, and RL-based reasoning methods, improves final-answer accuracy by optimizing trajectory-level rewards (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025; Yu et al., 2025). Process supervision provides finer-grained feedback through step-level labels, verifiers, or process reward models (Cobbe et al., 2021; Lightman et al., 2024; Zhang et al., 2025b; Khalifa et al., 2025), but often requires extra annotations or auxiliary models. Our method instead derives a lightweight structural signal from correctness-conditioned sampled trajectories, without requiring process-level supervision.

Reasoning Efficiency and Length Control Recent studies on efficient reasoning aim to reduce redundant computation through early stopping (Yang et al., 2025b), adaptive thinking (Zhang et al., 2025a), explicit length control (Aggarwal and Welleck, 2025), and budget-aware optimization (Shen et al., 2025; Sui et al., 2025; Fan et al., 2025). These approaches regulate reasoning costs through various signals, including stopping decisions, target reasoning lengths, and difficulty-aware budgets. Beyond explicit computation constraints, several recent works explore learning efficient reasoning behaviors from successful trajectories. Length-Adaptive Policy Optimization (LAPO) (Wu et al., 2025) first identifies successful reasoning-length patterns and incorporates them as meta-cognitive guidance into the model context. AutoThink (Tu et al., 2026) learns when to invoke explicit reasoning through multi-stage reinforcement learning, while DECS (Jiang et al., 2026) reduces redundant reasoning through decoupled token-level rewards and curriculum scheduling. In contrast, our method estimates successful reasoning scales online from verified trajectories sampled by the current policy and uses this signal only as a reward during policy optimization, without introducing additional length guidance or inference-time control mechanisms.

3 Empirical Analysis of Computation Scales

3.1 Reasoning Length Variation

We first examine whether reasoning length reflects computation allocation. For each problem, we sample $K = 32$ trajectories under identical prompt-

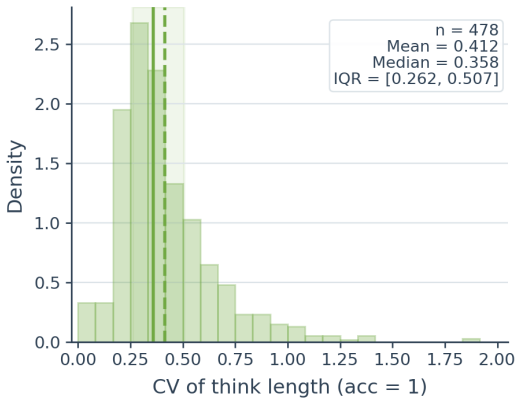


Figure 1: Variation of reasoning lengths across sampled trajectories for the same problem.

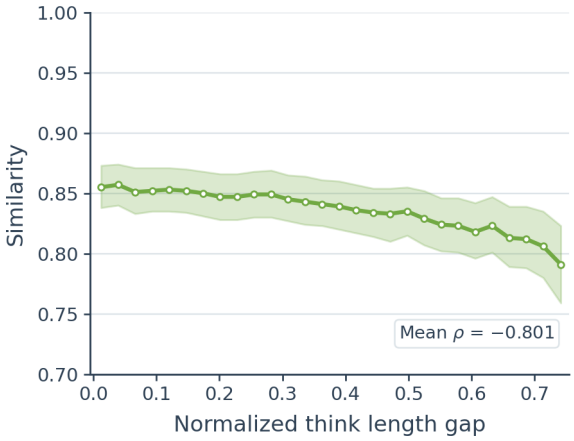


Figure 2: Trajectory similarity versus length difference.

ing and decoding settings, and measure reasoning length as the number of reasoning tokens generated before the final answer.

As shown in Figure 1, reasoning lengths vary substantially across trajectories for the same problem, suggesting heterogeneous computation behaviors rather than repeated executions of a fixed reasoning procedure. Moreover, we find that incorrect trajectories are often longer than verified correct ones, with the additional length frequently accompanied by redundant or unstable computation rather than improved reasoning.

This motivates a correctness-conditioned view of computation allocation. We therefore focus on verified correct trajectories and use their length distribution to estimate successful computation scales. These scales characterize the computation typically allocated to successful reasoning and form the basis of our scale-alignment objective.

3.2 Structure in Successful Reasoning Lengths

For reasoning length to serve as a proxy for computation allocation, it should reflect more than surface verbosity. We examine this through three structural properties of successful trajectories.

First, successful reasoning length increases monotonically with empirical problem difficulty, indicating that it reflects problem-dependent computation scale rather than a fixed or arbitrary generation tendency; see Appendix A for detailed statistics. Second, for the same problem, trajectories with similar lengths also exhibit greater structural similarity. To quantify this, we split each trajectory at natural paragraph boundaries, embed each step, and align the resulting step sequences using Soft Dynamic Time Warping (Soft-DTW) based on embedding-angle similarity. Figure 2 shows that trajectory similarity consistently declines as the relative length difference increases. This suggests that reasoning length captures coarse-grained structural variation rather than merely token count.

Table 1: Best-fitting distribution families for successful reasoning lengths under BIC.

Distribution family	BIC wins	Share
Log-normal	387	82.34%
Gamma	49	10.43%
Weibull	27	5.74%
Exponential	4	0.85%
Normal	3	0.64%

Successful lengths also exhibit stable distributional structure. Using Bayesian information criterion (BIC)-based selection over multiple candidate families, we find that most problems are best modeled by log-normal distributions (Table 1). This suggests that successful reasoning lengths concentrate around characteristic computation scales, which we use as lightweight estimates of problem-specific computation allocation. Detailed settings for each analysis experiment are provided in Appendix B.

3.3 Multiple Successful Scales

Structured successful lengths do not imply a unique optimal scale for each problem. The same problem may admit different valid computation paths, such as concise derivations or longer trajectories with additional verification. Thus, the key question is whether this variation remains organized enough to provide useful supervision.

Table 2: BIC-selected number of log-normal components fitted to successful reasoning lengths.

Scale structure	# Components	Proportion
Single-scale	1	58.5%
Multi-scale	2	30.9%
	3	6.4%
	4	4.3%

We model correctness-conditioned length distributions with mixtures of log-normal components and select the number of components using BIC.

Table 2 shows that successful reasoning is often concentrated around one scale, but a substantial fraction of problems exhibit multiple successful scales. These cases remain structured: the largest component retains substantial mass, with average dominant weights of 60.6%, 42.7%, and 33.8% for two-, three-, and four-component fits, respectively. Thus, multiple successful paths may coexist, yet the distribution often contains a dominant high-density region. This motivates scale-aware supervision that aligns trajectories with problem-specific successful regions rather than enforcing a fixed length or uniform brevity preference.

3.4 Purity Around Successful Scales

If successful scales capture meaningful computation regions, trajectories near these scales should be more likely to be correct. We examine this by measuring correctness purity in neighborhoods around the estimated dominant successful scale. For each problem q, let μ_q denote the dominant scale estimated from correct trajectories, and define

$$\mathcal{N}_q(\delta) = \left\{ y : \frac{|L(y) - \mu_q|}{\mu_q} < \delta \right\}, \qquad (1)$$

where $L(y)$ is the reasoning length. We define the purity of $\mathcal{N}_q(\delta)$ as the fraction of verified correct trajectories within this region, and compare it with two references: a scale estimated from all trajectories and a random-scale control.

Table 3 shows that neighborhoods centered at the successful scale consistently achieve higher correctness purity than both references, while maintaining comparable coverage. Thus, the gain comes from the location of the estimated scale rather than from selecting a smaller subset of trajectories. This connects successful length structure to correctness concentration: the estimated scale identifies a high-purity computation region, motivating scale alignment as a problem-specific alternative to global length preferences.

Table 3: Near-scale purity under different reference scales. Accuracy (Acc.) and coverage (Cov.) are reported as percentages.

δ	Acc. in near region			Cov.		
	Corr.	All	Rand.	Corr.	All	Rand.
0.05	**80.88**	71.35	69.17	10.10	10.39	10.73
0.10	**81.13**	71.12	71.35	20.08	20.66	20.67
0.20	**80.55**	73.35	73.75	39.83	39.92	39.74
0.30	**80.56**	74.17	74.49	56.76	56.82	56.56
0.50	**79.70**	74.47	74.82	78.52	80.45	79.97

Table 4: Finite-sample stability of dominant scale estimation relative to a 100-sample reference distribution.

Sample size	$\lvert\Delta \log \mu\rvert$	$\lvert\Delta \log \sigma\rvert$
8 trajectories	0.1185	0.3404
16 trajectories	0.0749	0.0632
32 trajectories	0.0515	0.1106

3.5 Recoverable Dominant Scale

Scale alignment is practical only if the successful computation scale can be estimated from limited online samples. Fortunately, our objective does not require reconstructing the full successful length distribution; it only needs a stable dominant region as the training anchor.

To test this, we use 100 correct trajectories per problem as a reference distribution, then estimate log-normal statistics from smaller successful subsets and compare them with the reference.

As shown in Table 4, the dominant-scale location is already stable with a small number of successful trajectories and improves with more samples. Although the variance estimate is less monotonic, it is secondary to our reward, which mainly depends on proximity to the dominant region. This supports estimating successful computation scales online with limited RL samples.

4 Method

Motivated by these observations, we propose a structure-aware RL framework that estimates successful computation scales from correct trajectories and encourages scale-aligned policy optimization, as shown in Figure 3.

4.1 Problem Formulation

Given a problem q with ground-truth answer y^*, the policy model π_θ samples a reasoning trajectory

$$\tau \sim \pi_\theta(\cdot \mid q), \tag{2}$$

from which an answer $\hat{y}$ is extracted. We denote the correctness of τ by

$$C(q, \tau) = \mathbb{I}(\hat{y} = y^*), \tag{3}$$

and use its reasoning length $L(\tau)$ as an observable proxy for computation allocation.

As observed in Sec. 3, successful trajectories for the same problem do not spread uniformly over all computation scales, but tend to concentrate around a problem-specific region. Let $\mathcal{A}_q$ denote the set of all trajectories for problem q, and let $\mathcal{S}_q \subseteq \mathcal{A}_q$ denote the successful trajectories. The lengths of these successful trajectories may reflect multiple distinct reasoning paths. We posit that these successful trajectories often contain a dominant high-density region, denoted by $\mathcal{D}_q \subseteq \mathcal{S}_q$.

Our goal is therefore not to minimize reasoning length globally, but to maximize the probability of generating trajectories that are both correct and aligned with the dominant successful region $\mathcal{D}_q$. Formally, we optimize

$$\max_\theta \mathbb{E}_{(q,y^*)\sim\mathcal{D}} \mathbb{E}_{\tau\sim\pi_\theta(\cdot|q)} \left[C(q,\tau)\, g\big(L(\tau), \mathcal{D}_q\big) \right], \tag{4}$$

where $g(\cdot,\cdot)$ measures the alignment between the trajectory length and the successful computation scale.

In the following section, we describe how $\mathcal{S}_q$ is estimated from sampled trajectories and incorporated into reinforcement learning.

4.2 Estimating Problem-Specific Successful Scales

For each problem q, we sample K reasoning trajectories $\{\tau_i\}_{i=1}^K$ from the current policy. Each trajectory produces a predicted answer $\hat{y}_i$ and a reasoning length $L_i = L(\tau_i)$.

Using answer verification, we construct the set of correct trajectories:

$$\mathcal{C}_q = \{\tau_i \mid \hat{y}_i = y^*\}. \tag{5}$$

When the number of correct trajectories is smaller than one quarter of all sampled trajectories, i.e.,

$$|\mathcal{C}_q| < \frac{1}{4}|\mathcal{A}_q|, \tag{6}$$

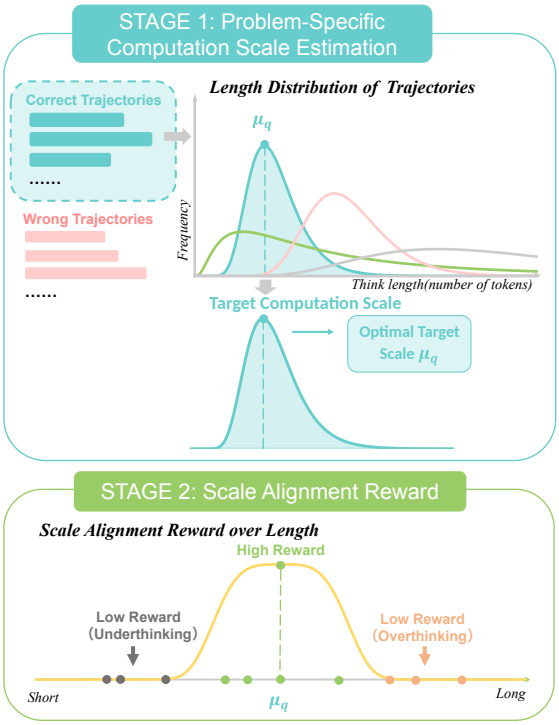


Figure 3: Overview of the proposed scale alignment framework.

we treat $\mathcal{C}_q$ as empty and do not apply any length constraint for problem q. This is because a small number of correct trajectories indicates a low-accuracy regime where correctness should take priority over length control, while also making successful scale estimation unreliable. Following the empirical analysis, we estimate successful computation scales from the verified correct trajectories. Since successful trajectories may contain multiple computation modes, we define the dominant successful region $\mathcal{D}_q \subseteq \mathcal{C}_q$ as the subset of correct trajectories closest to a problem-specific scale center μ_q:

$$\mathcal{D}_q = \underset{\mathcal{D}\subseteq\mathcal{C}_q,\, |\mathcal{D}|=\max(1,\lfloor|\mathcal{C}_q|/2\rfloor)}{\arg\min} \sum_{i\in\mathcal{D}} |L_i - \mu_q|. \tag{7}$$

That is, $\mathcal{D}_q$ contains the half of correct trajectories closest to the estimated scale center μ_q. We estimate μ_q from the correct trajectory lengths: it is given by the median successful length for coherent single-region distributions, and otherwise by the observed length nearest to the dominant peak of a one-dimensional KDE fitted to the correct lengths. Detailed criteria for distinguishing the two cases are deferred to Appendix C. We then compute the scale deviation statistic around the estimated successful scale region:

$$\sigma_q = \mathrm{MAD}\left(\{L_i\}\right), \tag{8}$$

where the MAD is computed either from all correct trajectories or from the dominant local region $\mathcal{D}_q$, depending on the estimated computation structure.

In practice, a small constant ϵ is added to σ_q to avoid numerical instability when successful trajectory lengths are highly concentrated. Importantly, the estimated scale is problem-specific rather than a global target length.

When $\mathcal{C}_q$ is empty, no successful computation pattern can be inferred from the current samples. In this case, the framework falls back to standard outcome-based optimization without additional scale supervision.

4.3 Computation-Scale Alignment Objective

Given the estimated successful scale (μ_q, σ_q), we define a scale-alignment score that measures how close a trajectory's computation is to the dominant successful region of problem q. Unlike monotonic length penalties, this objective favors trajectories near the estimated successful scale and penalizes deviations on both sides, combining outcome correctness with computation-scale alignment.

For each trajectory, we first normalize its length deviation as:

$$z_i = \frac{L_i - \mu_q}{\sigma_q + \epsilon}, \tag{9}$$

where ϵ is a small constant for numerical stability.

The scale-alignment reward is defined as:

$$r_i^{\text{scale}} = \begin{cases} \exp(-|z_i|^4), & i \in \mathcal{C}_q, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The scale-alignment reward is defined only over correct trajectories, while its magnitude is determined by the distance to the dominant successful scale estimated from $\mathcal{D}_q$. As a result, reasoning length acts as a conditional structural signal given correctness, rather than as an independent optimization target.

The quartic form provides a smooth and symmetric alignment signal around the estimated scale. It penalizes both under- and over-allocation of computation while decaying more sharply than a quadratic penalty outside the dominant successful region. Compared with higher-order exponents, it also avoids making the effective reward region overly narrow under limited online RL sampling. We study alternative exponents in Sec. 5.4.

4.4 Integration with Outcome-Based RL

The proposed scale-alignment objective can be directly integrated into outcome-based RL frameworks such as GRPO. For each trajectory, we combine the standard outcome reward with the scale-alignment reward:

$$R_i = r_i^{\text{acc}} + r_i^{\text{scale}}. \tag{11}$$

The combined reward is then group-normalized to compute the final advantage. Since the scale reward is activated only for correct trajectories, it serves as a secondary structural preference among successful samples rather than an independent length reward.

Both reward terms are bounded in $[0, 1]$. We add them directly and rely on group normalization, avoiding an additional coefficient to tune. Because the scale reward is zero for incorrect trajectories, it cannot reward a short answer solely for being short.

Scale statistics are estimated from current online successful trajectories during training, so the supervision signal tracks the policy's evolving computation behavior. If fewer than one quarter of a prompt group's samples are correct, we fall back to outcome-based optimization; this threshold is reached by about 6% of prompt groups in the 1.5B run. Scale estimation is training-time only: inference uses the trained policy without online scale estimation or a scale reward. The method therefore adds lightweight structural supervision using final-answer correctness and trajectory length, without process-level annotations or an additional verifier model.

5 Experiments

We evaluate the proposed method in terms of reasoning accuracy, computation efficiency, and computation-scale behavior. The experiments examine whether scale alignment improves the accuracy–efficiency trade-off beyond simply encouraging shorter generations.

5.1 Experimental Setup

We evaluate our method on GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), AMC23 (MAA, 2023), AIME24 (Mathematical Association of America, 2024), AIME25 (Mathematical Association of America, 2025), and OlympiadBench (He et al., 2024), covering mathematical reasoning tasks from grade-school arithmetic to competition-level problem solving. We retain GSM8K as a lower-difficulty benchmark to assess computation allocation across difficulty levels, and include AMC23 among the competition-level

benchmarks. We use DeepSeek-R1-Distill-Qwen-1.5B and 7B (DeepSeek-AI et al., 2025) as base models and additionally evaluate Qwen3-8B (Yang et al., 2025a). We train these models on Hendrycks MATH using a DAPO-style RL framework (Yu et al., 2025). Unless otherwise specified, training is conducted on 4 NVIDIA A800 GPUs with $K = 16$ sampled trajectories per input, a maximum response length of 8,192 tokens, and a learning rate of 1×10^{-6}. Training typically requires approximately 12 hours per run.

We compare against representative efficient reasoning methods, including Adaptthink (Zhang et al., 2025a), L1 (Aggarwal and Welleck, 2025), DEER (Yang et al., 2025b), and DAST (Shen et al., 2025), DECS, and AutoThink, which cover adaptive reasoning allocation, dynamic early exit, difficulty-aware slow thinking, decoupled token rewards, and multi-stage adaptive reasoning. We re-evaluate available public checkpoints with a unified evaluation pipeline. DEER is the sole exception: its results are taken from the original paper because a checkpoint suitable for our evaluation was unavailable. The source of each result is marked in Table 5.

We report **Pass@1** as the primary accuracy metric and **average reasoning length** as the computation cost. We use a unified evaluation protocol: for each item, Pass@1 estimates single-sample accuracy from 16 independent generations under fixed decoding settings, without majority voting or best-of-N selection. Scale estimation and the fallback to outcome-based RL are training-time operations; all six benchmarks are held out for evaluation, where the trained policy is used without online scale estimation or a scale reward. We further analyze computation-scale behavior using the normalized deviation $(L - \mu_q)/\mu_q$, where L is the generated reasoning length and μ_q is the estimated successful computation scale.

5.2 Main Results

Table 5 reports the unified re-evaluation and marks the DEER values taken from its original paper. For both DeepSeek model sizes, our method uses fewer reasoning tokens than the base model on all six benchmarks. On the 1.5B model, our method has the highest Pass@1 among the listed methods on MATH500, AIME24, AIME25, OlympiadBench, and AMC23; L1 has the highest GSM8K score. On the 7B model, our method has the highest score among unified re-evaluations on GSM8K, MATH500, and AMC23, ties Adaptthink on AIME25 and DAST on OlympiadBench, and reaches 49.1 on AIME24, matching Adaptthink and close to the DEER value reported in its original paper (49.2).

Importantly, the method is not designed to minimize reasoning length uniformly. Instead, it encourages problem-dependent computation allocation by aligning trajectories with dominant successful computation scales. The results therefore suggest that efficient reasoning depends on reallocating computation toward effective reasoning regimes rather than simply generating shorter responses.

On AMC23, our 1.5B and 7B models reach 72.4 and 90.0 Pass@1, respectively, compared with 58.2 and 82.5 for their base models. On Qwen3-8B, our method matches the base Pass@1 on GSM8K and MATH500 and improves it on the four competition-level benchmarks, while reducing average reasoning length on all six datasets (Appendix F).

5.3 Comparison with Length Regularization

Naive length regularization that ignores the original successful scale distribution may unintentionally under-allocate computation for difficult problems. In contrast, our method aligns trajectories with problem-specific successful scales. This difference is reflected in the results: on 7B GSM8K, Adaptthink reduces reasoning length to 144 tokens but achieves 89.8 Pass@1, while our method uses 367 tokens and reaches 93.3 Pass@1. On 7B MATH500, our method reaches 92.4 Pass@1 with 1,178 reasoning tokens; L1 is shorter at 1,078 tokens but reaches 90.1, illustrating the accuracy-efficiency trade-off. On harder benchmarks such as AIME24, AIME25, and OlympiadBench, our method improves or maintains accuracy over the base models while reducing reasoning length. Overall, these results suggest that scale alignment improves reasoning efficiency through problem-dependent computation allocation rather than uniformly shortening reasoning.

5.4 Ablation Study

Table 6 examines the main components of our method. The results support two claims: useful scale targets should be estimated from a more reliable subset of successful trajectories, and policy learning benefits from an explicit reward that encourages alignment with those targets.

Table 5: Main results across six reasoning benchmarks. Each dataset reports Pass@1 (%) and average reasoning length (tokens); a dash indicates an unavailable result. All rows except DEER (†) are re-evaluated with our unified pipeline; DEER values are from its original paper.

Method	GSM8K		MATH500		AIME24		AIME25		OlympiadBench		AMC23	
	P@1	Len	P@1	Len	P@1	Len	P@1	Len	P@1	Len	P@1	Len
DeepSeek-R1-Distill-Qwen-1.5B												
Base Model	73.0	1051	72.2	3233	22.2	10980	16.7	10976	37.2	7172	58.2	6819.0
Adaptthink	83.3	717	80.7	1683	20.9	6089	22.2	4663	44.2	4326	69.6	3532.4
L1	86.4	741	81.2	1820	23.3	2519	21.4	2426	43.4	2078	70.0	2121.9
DECS	83.2	771.3	81.5	1182.4	22.5	4851.6	23.5	4353.0	44.8	2485.0	65.0	2536.7
AutoThink	80.3	105.5	80.6	1227.1	25.2	7815.8	22.5	7031.7	43.4	3893.9	70.3	3345.4
DEER†	74.7	984	67.8	2497	23.3	9557	10.0	9281	32.0	5960	–	–
Ours	84.5	540	83.3	1660	28.1	6177	24.1	5936	47.0	3912	72.4	3839.4
DeepSeek-R1-Distill-Qwen-7B												
Base Model	92.2	684	89.4	2381	40.0	9148	35.0	10152	58.3	5658	82.5	4162.5
Adaptthink	89.8	144	91.0	1331	49.1	7113	36.7	7514	58.7	4348	87.5	2867.6
L1	92.8	567	90.1	1078	38.2	3172	28.8	2925	56.4	1855	85.3	1628.8
DECS	93.1	506.9	91.3	1209.9	43.3	4867.3	23.3	4759.5	57.8	2530.7	89.3	2299.0
AutoThink	89.5	77.1	90.8	1344.8	26.6	6798.7	36.25	8131.1	59.6	3876.2	87.5	2820.1
DAST	91.4	473	90.6	2095	45.8	10077	33.3	10115	59.3	5433	–	–
DEER†	90.6	917	89.8	2143	49.2	9839	36.7	7257	52.6	5420	–	–
Ours	93.3	367	92.4	1178	49.1	5552	36.7	5802	59.3	2953	90.0	2328.0

Table 6: Ablation results on MATH500 with the 1.5B model.

Variant	Pass@1	Avg Len
Full Method	83.3	1660
w/o Scale Reward	83.7	1939
w/o Dominant-Region Identification	82.5	1466
w/o Trajectory Selection	81.6	1548
w/ Linear Reward	82.8	1870
w/ Quadratic Reward	83.0	1625

Trajectory selection. The w/o Trajectory Selection variant estimates the target scale from all sampled trajectories, rather than filtering by correctness. Its Pass@1 drops from 83.3 to 81.6, suggesting that incorrect trajectories introduce noisy computation patterns and distort the estimated scale.

Dominant-region identification. The w/o Dominant-Region Identification variant estimates the scale from all verified correct trajectories, instead of selecting the dominant local region. Its Pass@1 drops from 83.3 to 82.5, indicating that even successful trajectories may contain heterogeneous computation modes and that identifying the dominant region provides a cleaner training signal.

Scale reward. The *w/o Scale Reward* variant removes the structural alignment signal while retaining correctness reward. Its accuracy remains competitive, but its average length increases from 1660 to 1939. This indicates that outcome supervision alone is insufficient to regulate computation allocation.

Reward sharpness. We compare the quartic alignment reward with linear and quadratic alternatives. Both alternatives underperform the full method. We attribute this to their sharper reward variation around the target scale: under high-variance reinforcement learning updates, overly sensitive reward shaping can make it difficult for the policy to converge toward the desired length region. Compared with linear and quadratic alternatives, the quartic form is flatter near the target scale but decays more strongly for large deviations. This provides tolerance to small sampling noise while still discouraging substantial departures from the dominant region.

Overall, the ablations show that the effectiveness of our method comes from correctness-conditioned,

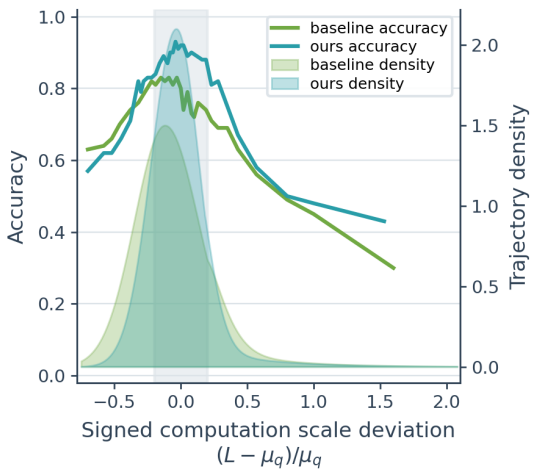


Figure 4: Reasoning accuracy and trajectory density under computation-scale deviation. The horizontal axis represents the signed relative deviation from the problem-specific computation scale, $(L - \mu_q)/\mu_q$.

problem-specific scale estimation together with a reward that explicitly encourages scale alignment.

5.5 Further Analysis

Probability Shifts toward Effective Computation. To understand why the method improves the accuracy-efficiency trade-off, we analyze reasoning performance as a function of deviation from the recovered computation scale. For each problem, we estimate a problem-specific scale μ_q from successful trajectories and normalize reasoning length by $(L - \mu_q)/\mu_q$, allowing computation behavior to be compared across problems with different absolute lengths.

Figure 4 shows that accuracy peaks near the recovered computation scale and declines as trajectories deviate from it. The decline is asymmetric: moderate under-computation has relatively limited impact, whereas large positive deviations consistently reduce accuracy, suggesting that excessively long reasoning often reflects unstable exploration or redundant computation rather than productive additional reasoning. Our method shifts generation probability toward these high-purity computation regions, benefiting from computation reallocation toward scales associated with successful reasoning.

Effects across Problem Difficulty. Effective computation allocation depends on problem difficulty. Easier problems often benefit from shorter reasoning, while harder problems may require longer derivations. Consistent with this, our method substantially improves the accuracy–efficiency trade-off on GSM8K and MATH500 by reducing redundant computation without harming performance. On harder benchmarks such as AIME and OlympiadBench, the method does not always produce the shortest trajectories, but instead preserves additional computation when it contributes to accuracy. These results suggest that reasoning efficiency should be defined relative to problem-specific successful computation scales rather than a global preference for shorter outputs.

Case Study. Appendix E compares matched samples. Our model uses less computation on problems with a stable short successful path, and more computation on a compound-interest problem where the base model's short trajectory is incorrect.

6 Conclusion

In this work, we show that excessive reasoning often reflects deviation from problem-appropriate successful computation scales, rather than merely producing too many tokens. By encouraging scale-aligned trajectories, our method improves the accuracy–efficiency trade-off without uniformly shortening reasoning. These results suggest that sampled trajectory structure can provide lightweight supervision for adaptive reasoning optimization without additional annotations or process-level labels.

7 Limitations

Our experiments focus on mathematical reasoning, so the effectiveness of the framework in coding, planning, and agentic reasoning remains untested. We measure inference cost by generated reasoning tokens; we do not measure wall-clock latency, energy use, or deployment cost, and token reductions should not be interpreted as direct measurements of those quantities. The method also provides no process-level guarantee that an individual reasoning trace is valid or faithful.

The framework depends on final-answer extraction and verification. Errors in either step can contaminate the correctness-conditioned trajectories and their estimated scales. When fewer than 25% of a prompt's sampled trajectories are correct, we fall back to outcome-based RL; this occurred for about 6% of prompt groups in the 1.5B run. Scale estimates may still be unreliable when correct trajectories are sparse or unstable. Finally, reasoning length is only a coarse proxy for computation structure and cannot distinguish a semantically sound

path from a concise but incorrect one, as illustrated by the failure case in Appendix D.

Societal Impact. Reducing generated reasoning tokens may lower inference resource use, although we do not directly measure latency or energy consumption. Since the method does not verify intermediate reasoning steps or guarantee correctness, its use does not remove the need to assess final answers in the intended application context.

Acknowledgments

The work is supported by National Key R&D Program of China (No 2024YFF0907003)

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling How Long A Reasoning Model Thinks With Reinforcement Learning. *Preprint*, arXiv:2503.04697.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, and Piotr Nyczyk. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *Preprint*, arXiv:2407.21787.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*, 1.

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. *Preprint*, arXiv:2110.14168.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *Preprint*, arXiv:2507.06261.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature*, 645(8081):633–638.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing Premise exacerbates Overthinking: Are Reasoning Models losing Critical Thinking Skill? *Preprint*, arXiv:2504.06514.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, and 167 others. 2026. GLM-5: From Vibe Coding to Agentic Engineering. *Preprint*, arXiv:2602.15763.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, and Yuxiang Zhang. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *Preprint*, arXiv:2103.03874.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2026. Overthinking reduction with decoupled rewards and curriculum data scheduling. In *International Conference on Learning Representations*, volume 2026, pages 114966–114995.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process Reward Models That Think. *Preprint*, arXiv:2504.16828.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601.

MAA. 2023. American mathematics competitions. In *American Mathematics Competitions*.

Mathematical Association of America. 2024. American invitational mathematics examination 2024.

Mathematical Association of America. 2025. American invitational mathematics examination 2025.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2025. Large Language Models: A Survey. *Preprint*, arXiv:2402.06196.

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. *Preprint*, arXiv:2501.19393.

OpenAI, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, and 245 others. 2026. OpenAI o1 System Card. *Preprint*, arXiv:2412.16720.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, and Alex Ray. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Back. 2024. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 1(2):3.

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.

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, Zhaoxiang Liu, and Shiguo Lian. 2025. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2322–2331.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters. *Preprint*, arXiv:2408.03314.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *Preprint*, arXiv:2503.16419.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 77 others. 2025. Kimi k1.5: Scaling Reinforcement Learning with LLMs. *Preprint*, arXiv:2501.12599.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. 2026. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. *Advances in Neural Information Processing Systems*, 38:15181–15207.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xingyu Wu, Yuchen Yan, Shangke Lyu, Linjuan Wu, Yiwen Qiu, Yongliang Shen, Weiming Lu, Jian Shao, Jun Xiao, and Yueting Zhuang. 2025. LAPO: Internalizing Reasoning Efficiency via Length-Adaptive Policy Optimization. *Preprint*, arXiv:2507.15758.

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, Minghui Chen, Zheng Lin, and Weiping Wang. 2025b. Dynamic Early Exit in Reasoning Models. *Preprint*, arXiv:2504.15895.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2026. Towards thinking-optimal scaling of test-time compute for llm reasoning. *Advances in Neural Information Processing Systems*, 38:43605–43631.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. *Preprint*, arXiv:2503.14476.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025a. AdaptThink: Reasoning Models Can Learn When to Think. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3716–3730, Suzhou, China. Association for Computational Linguistics.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516.

A Additional Analysis of Successful CoT Lengths

We analyze successful CoT lengths on MATH500 using its annotated difficulty levels. For each problem, we sample 100 trajectories from the baseline policy and compute statistics for trajectories with verified correct answers.

Table 7 summarizes all sampled trajectories, whereas Table 8 includes only trajectories with verified correct answers.

Mean successful CoT length increases from 822.2 tokens at Level 1 to 3,522.1 tokens at Level 5. Higher difficulty levels also show greater variation in length. These patterns suggest that more difficult problems require longer and more variable successful computation trajectories.

Table 7: CoT length by MATH500 difficulty level for all sampled trajectories.

Lvl	Count	Traj.	Acc	Mean	Med.	IQR
1	43	4.3K	85.7	1123	626	441–973
2	90	9.0K	80.0	1433	781	536–1387
3	105	10.5K	80.0	2243	1127	714–2143
4	128	12.8K	71.0	3294	1593	893–3894
5	134	13.4K	56.1	5859	3625	1445–9413

Table 8: CoT length of correct trajectories by MATH500 difficulty level.

Lvl	Count	Correct	Acc	Mean	Med.	IQR
1	43	3.7K	85.7	822	590	426–877
2	90	7.2K	80.0	1123	706	507–1120
3	105	8.4K	80.0	1677	1031	675–1768
4	128	9.1K	71.0	2212	1299	807–2474
5	134	7.5K	56.1	3522	2107	1138–4942

B Additional Details of the Empirical Analysis

B.1 Trajectory Similarity Analysis

To assess whether trajectories with similar CoT lengths follow similar semantic paths, we represent each trajectory as a sequence of steps and compare these sequences using embedding-based Soft-DTW alignment.

Step representation. We segment each trajectory into steps at blank lines:

$$\tau = (s_1, \ldots, s_m). \tag{12}$$

We map each step to an embedding vector $e_i \in \mathbb{R}^d$, yielding

$$E_\tau = [e_1; \ldots; e_m] \in \mathbb{R}^{m \times d}. \tag{13}$$

Given two trajectories τ^a and τ^b, we compute the pairwise cosine distance matrix

$$D_{ij} = 1 - \cos(e_i^a, e_j^b). \tag{14}$$

Soft-DTW trajectory alignment. We apply Soft-DTW to the step-level distance matrix to allow flexible alignment between trajectories. Let $\mathrm{sDTW}_\gamma(D)$ denote the Soft-DTW cost with smoothing parameter γ.

We compute the Soft-DTW divergence as

$$\Delta(\tau^a,\tau^b) = \mathrm{sDTW}_\gamma(D^{ab}) - \frac{1}{2}\left[\mathrm{sDTW}_\gamma(D^{aa}) + \mathrm{sDTW}_\gamma(D^{bb})\right]. \tag{15}$$

We define trajectory similarity as

$$\mathrm{Sim}(\tau^a, \tau^b) = \exp\left(-\frac{\max(\Delta(\tau^a, \tau^b), 0)}{m+n}\right). \tag{16}$$

Length-gap analysis. Similarity is computed only between trajectories sampled for the same problem. For each pair, we compute the relative CoT length difference:

$$G(\tau^a, \tau^b) = \frac{|L(\tau^a) - L(\tau^b)|}{\max(L(\tau^a), L(\tau^b))}. \tag{17}$$

We assess the association between $G(\tau^a, \tau^b)$ and $\mathrm{Sim}(\tau^a, \tau^b)$ using Spearman correlation and trends across length-difference bins.

B.2 Distribution and Mixture Fitting

This subsection details the distribution and mixture analyses in Sections 3.2 and 3.3.

Problem-level fitting setup. For each problem, we sample up to 100 trajectories using the same prompt. Let L_i denote the CoT length of trajectory i, and let $C_i \in \{0, 1\}$ indicate whether its final answer is verified as correct. We analyze the set of correct-trajectory lengths

$$\mathcal{L}_q^+ = \{L_i : C_i = 1\}. \tag{18}$$

We include a problem in the fitting analysis only if at least five correct trajectories are available.

Single-family fitting. For each eligible problem, we independently fit normal, log-normal, exponential, gamma, and Weibull distributions to $\mathcal{L}_q^+$. We select the best-fitting family using the Bayesian information criterion (BIC):

$$\mathrm{BIC} = k \log n - 2 \log \hat{\mathcal{L}}, \tag{19}$$

where n is the sample count, k is the number of free parameters, and $\hat{\mathcal{L}}$ is the maximized likelihood.

Log-normal mixture fitting. To analyze potential multi-scale structure, we further fit mixtures of log-normal components:

$$p(L) = \sum_{j=1}^{K} \pi_j \, \mathrm{LogNormal}(L; \mu_j, \sigma_j^2), \tag{20}$$

where $\sum_j \pi_j = 1$ and $\pi_j \geq 0$. The number of components is selected from $K \in \{1, 2, 3, 4\}$ using BIC.

We estimate mixture parameters by expectation-maximization in log-length space, using multiple random initializations and retaining the solution with the highest likelihood.

Dominant component statistics. For each selected mixture, we record the largest component weight

$$\max_j \pi_j, \tag{21}$$

which measures the probability mass assigned to the largest successful computation region. This statistic distinguishes mixtures with a dominant scale from those with more balanced components.

C Dominant Successful Region Estimation

This section describes how we estimate the problem-specific successful computation region. For each problem q, we verify the final answers of sampled trajectories and retain the indices of correct responses:

$$\mathcal{C}_q = \{i : C(q, \tau_i) = 1\}. \tag{22}$$

If too few correct trajectories are available, we omit the scale reward for that problem, as described in Sec. 4.2.

For eligible problems, we determine whether correct-trajectory lengths form one coherent region or multiple separated modes. Let $\ell_{(1)} \leq \ell_{(2)} \leq \cdots \leq \ell_{(m)}$ be the sorted correct lengths, and define adjacent gaps as $g_j = \ell_{(j+1)} - \ell_{(j)}$. We exclude zero gaps caused by duplicate lengths and compute the coefficient of variation over the positive gaps:

$$\mathrm{GapCV}_q = \frac{\mathrm{Std}(\{g_j : g_j > 0\})}{\mathrm{Mean}(\{g_j : g_j > 0\})}. \tag{23}$$

We use a threshold of $\gamma = 0.9$. If GapCV_q is non-finite or $\mathrm{GapCV}_q \leq \gamma$, we treat the correct lengths as one dominant region. Its center is the median correct length, and its dispersion is the median absolute deviation over all correct trajectories:

$$\mu_q = \mathrm{Median}\{L_i : i \in \mathcal{C}_q\}. \tag{24}$$

$$\sigma_q = \mathrm{Median}_{i \in \mathcal{C}_q} |L_i - \mu_q|. \tag{25}$$

This rule also covers cases with too few distinct lengths to establish a meaningful gap pattern.

If $\mathrm{GapCV}_q > \gamma$, we treat the correct lengths as containing separated computation modes. We fit a one-dimensional Gaussian kernel density estimate (KDE) using Scott's bandwidth rule. We evaluate the density on a 512-point grid, extending the observed range by 5% on each side with a minimum extension of one token, and identify its highest peak. Because the reward scores sampled trajectories, we set μ_q to the observed correct length nearest to this peak. For this case, we estimate dispersion from the dominant successful region defined in Sec. 4.2.

$$\sigma_q = \mathrm{Median}_{i \in \mathcal{D}_q} |L_i - \mu_q|. \tag{26}$$

In both branches, we apply the numerical floor

$$\sigma_q \leftarrow \max(\sigma_q, 10^{-6}). \tag{27}$$

D Qualitative Case Study

We present three cases illustrating reduced CoT length, improved answer accuracy, and a failure mode of scale alignment.

Case 1: Preserved Accuracy with Shorter CoT Trajectories. Problem. *The price of a laptop is $1000. If you get a 20% discount, how much do you have to pay?*

Both models answer correctly in all 32 samples. The baseline's correct trajectories have a mean CoT length of 445.1 tokens, compared with 114.0 tokens for our model. The following abbreviated excerpt illustrates a correct baseline response:

> *Baseline.* Compute the discount: $0.20 \times 1000 = 200$. Subtract from the original price: $1000 - 200 = 800$. Therefore, the final answer is $\boxed{800}$.

The scale-aligned model follows a shorter valid path:

> *Ours.* Discount amount: $\frac{20}{100} \times 1000 = 200$. Final price: $1000 - 200 = 800$. Answer: $\boxed{800}$.

This example shows that the model can preserve answer accuracy while reducing unnecessary elaboration on a simple problem.

Case 2: Higher Accuracy in Base-8 Arithmetic. **Problem.** *Find the product of* $6_8 \cdot 7_8$. *Express your answer in base* 8.

The baseline answers correctly in 1 of 32 samples, whereas our model does so in 31 of 32. Baseline errors often arise when the model computes the correct value but omits the required base-8 notation:

> *Baseline failure.* ... The product of 6_8 and 7_8 is 52_8. Therefore, the final answer is $\boxed{52}$.

The value is correct, but the boxed answer does not use the requested notation. The baseline's single correct trajectory uses 3,959 CoT tokens. Our model typically follows a complete, shorter conversion procedure:

> *Ours.* Convert 6_8 and 7_8 to decimal: 6_{10} and 7_{10}. Multiply: $6 \times 7 = 42_{10}$. Convert back to base 8: $42 \div 8 = 5$ remainder 2, so $42_{10} = 52_8$. Therefore, the product is $\boxed{52_8}$.

The representative trajectory from our model uses 562 CoT tokens. This example illustrates how alignment with a successful computation region can preserve all steps needed to meet the problem's answer-format requirement.

Case 3: A Failure of Semantic Interpretation. **Problem.** *There is space for 20 pencils in the box. If there are 4 pencils missing from the box, how many pairs of pencils are in the box?*

The baseline answers correctly in 28 of 32 samples, compared with 12 of 32 for our model. A correct trajectory computes the intended quantity directly:

> *Correct interpretation.* There are $20 - 4 = 16$ pencils in the box. Since each pair contains 2 pencils, the number of pairs is $16 \div 2 = 8$. Therefore, the final answer is $\boxed{8}$.

Several scale-aligned trajectories instead interpret "pairs" as unordered choices of two pencils, yielding concise but incorrect responses:

> *Ours failure.* ... A pair is a combination of 2 pencils. The number of ways to choose 2 pencils out of 16 is $\binom{16}{2} = 120$. Thus, the number of pairs is $\boxed{120}$.

This failure illustrates a limitation of length-based supervision: a concise trajectory can still reflect an incorrect interpretation of the problem. Answer correctness must therefore remain part of the training signal.

E Paired Case Studies

For each case, we compare the base and scale-aligned models using the same prompt and sampling index. CoT length counts tokens in the model's reasoning segment.

Signature-count word problem. Both trajectories yield correct answers. The base model uses 1,215 CoT tokens and repeatedly considers interpretations of the weekly rate that are irrelevant to the solution. Our model uses 367 tokens to compute the current total and remaining signatures directly.

Exponential equation $3^{2x} + 19 = 10^x$. Both trajectories obtain and verify $x = 2$. The base model uses 15,342 CoT tokens, including repeated exploration, whereas our model reaches the same answer in 1,304 tokens.

Compound-interest problem. The base model uses 143 CoT tokens and produces an incorrect answer after an incomplete treatment of compounding. Our model uses 6,221 tokens for a longer derivation and obtains the correct rounded value. This pair illustrates that scale alignment can retain additional computation when a problem requires it.

Together, these matched examples illustrate shorter trajectories when concise successful paths are available and longer trajectories when additional derivation supports a correct answer.

F Additional Results

F.1 Qwen3-8B

We evaluate Qwen3-8B on all six benchmarks (Table 9). Our method matches the base model's Pass@1 on GSM8K and MATH500, improves Pass@1 on AIME24, AIME25, OlympiadBench, and AMC23, and reduces average CoT length on every benchmark.

G Evaluation Protocol

For all methods evaluated with our pipeline, we use the same user-prompt prefix, decoding settings,

Table 9: Qwen3-8B results under the 16-sample evaluation protocol. Each dataset reports Pass@1 (%) and average CoT length (tokens).

Method	GSM8K		MATH500		AIME24		AIME25		OlympiadBench		AMC23	
	P@1	Len	P@1	Len	P@1	Len	P@1	Len	P@1	Len	P@1	Len
Base	95.8	1533.6	94.2	2366.5	53.9	10477.3	53.3	11571.0	60.9	8481.7	87.1	6764.3
Ours	95.8	878.9	94.2	2339.5	63.3	8525.9	56.6	9218.5	65.5	5597.4	90.3	4187.2

response-length limits, answer extractor, and verifier. The prompt prefix is:

```
Solve the following math problem efficiently and clearly. The last line of your response should be of the following format: 'Therefore, the final answer is: $\boxed{ANSWER}$' (without quotes) where ANSWER is just the final number or expression that solves the problem.
```

The prompt prefix ends with two newline characters.

We sample 16 independent responses per problem with temperature 0.6 and top-p 0.95. Pass@1 estimates single-sample answer accuracy from these responses; we do not use majority voting or best-of-16 selection. Maximum prompt and response lengths are 4,096 and 16,384 tokens, respectively, and the evaluation batch size is 512. We run inference with vLLM on four GPUs. We extract answers using `mathruler.grader.extract_boxed_content` and verify them using `mathruler.grader.grade_answer`.

H Training Configuration

The main 1.5B experiment uses the Hendrycks MATH training split without a system prompt (5,507 examples) and the corresponding combined validation set. Training follows a DAPO-style procedure with path-aware advantage estimation and the `adv_KDE_5` group reward. Table 10 lists the hyperparameters.

Code and Evaluation Assets. As stated in the rebuttal, we will release the training scripts, scale-reward implementation, evaluation prompt and decoding settings, verifier configuration, and result-source annotations with the camera-ready materials.

Table 10: Training hyperparameters for the 1.5B scale-aligned run.

Hyperparameter	Value
Training epochs	3
Total optimization steps	129
Rollout responses per prompt	16
Training prompt batch size	64
Generation prompt batch size	128 (`drop_last=True`)
PPO mini-batch size	16
Maximum prompt length	2,048
Maximum response length	8,192
Learning rate	1×10^{-6}
Warmup steps	10
Weight decay	0.1
Gradient clipping	1.0
Rollout temperature / top-p	1.0 / 1.0
PPO clip range	$[0.2, 0.28]$
Overlong buffer / penalty factor	2,048 / 1.0
KL reward / KL loss	Disabled / disabled
Loss aggregation	Token mean
GPUs	4 NVIDIA A800