Are "Aha moments" Always Beneficial? Measuring and Optimizing the Efficiency of Chain-of-Thought in Reasoning Models

Anonymous ACL submission

Abstract

Recent reasoning models often improve performance by generating longer chain-of-thought traces with reflection, verification, and "Aha moments". However, recent studies show that such traces can also exhibit suboptimal reasoning behaviors, such as overthinking. This raises a central question: does deeper reasoning ability always require longer thinking? To answer this question, we introduce the Redundant Derivation Score (RDS), a direct measure of repeated derivation in reasoning traces. Using RDS as a lens, we benchmark reasoning efficiency across a range of model families on challenging reasoning tasks. Our analysis reveals that deeper reasoning does not necessarily require longer reasoning: Newer models such as DeepSeek-V3.2 achieve higher accuracy with shorter traces and sharply lower RDS than their predecessors. This contrasts with previous model updates such as DeepSeek-R1-0528, where accuracy gains are accompanied by longer traces while RDS remains at similar levels. Motivated by this finding, we incorporate RDS as an auxiliary reward in RLVR. RDS-aware training preserves the accuracy gains of standard RL while reducing both redundant derivation and response length by over 20%, outperforming length-based rewards.

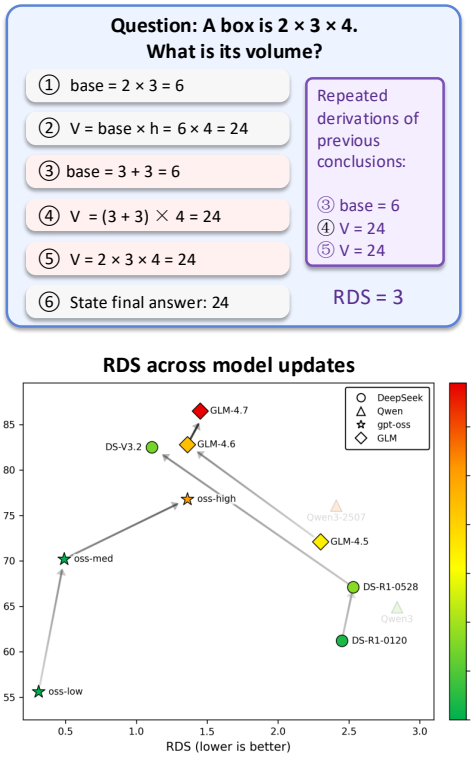

Figure 1: **Top:** An illustrative example of how the Redundant Derivation Score (RDS) is measured. **Bottom:** RDS vs. accuracy across recent model generations, with reasoning length encoded by color. The transitions from DeepSeek-R1-0528 to DeepSeek-V3.2 and from GLM-4.5 to GLM-4.6 exhibit a distinct efficiency-driven pattern, achieving higher accuracy with lower RDS.

1 Introduction

Recent reasoning models have achieved substantial improvements on complex reasoning tasks by generating long chain-of-thought traces. Through reinforcement learning with verifier-based rewards (RLVR), models learn to explore alternative solution paths, verify intermediate results, and revise earlier mistakes through reflection and apparent "Aha moments" (Shao et al., 2024). These behaviors are often treated as signs of deeper reasoning: the model thinks longer, checks more possibilities, and eventually arrives at a better answer.

However, longer reasoning is not always better reasoning. Recent studies show that long reasoning traces can also exhibit suboptimal behaviors, including repeated checking, circular exploration, and overthinking (Zhang et al., 2025; Chen et al., 2025; Wang et al., 2025a). In many cases, a model continues to verify or re-derive conclusions that have already been established, increasing response length without making meaningful progress. This lead to the key question: *does deeper reasoning ability always require longer thinking traces?* To answer this question, we introduce the Redundant Derivation Score (RDS), which captures re-

derivations of conclusions that have already been established earlier in the reasoning trace. This allows us to measure reasoning efficiency objectively without judging the usefulness of each step or relying on length as a proxy. Using RDS, we benchmark a range of reasoning model families and versions on challenging math and coding tasks.

Our benchmark reveals an emerging efficiency-driven improvement pattern. Previous model updates often improve accuracy by generating longer traces while maintaining similar levels of RDS. In contrast, newer models such as DeepSeek-V3.2 (DeepSeek-AI, 2025b) and GLM-4.6 (Z.ai, 2025a) achieve higher accuracy with comparable or even shorter traces and sharply lower RDS than their predecessors as shown in Figure 1. This suggests that deeper reasoning can decouple from longer reasoning: performance gains can come from making each reasoning step more informative, rather than extending the chain of thought.

We further analyze this shift through reasoning traces from successive model versions that correctly solve the same problems. The comparison shows that efficient models do not merely compress their outputs. Instead, they preserve useful verification while avoiding repeated derivations of already established conclusions. For example, DeepSeek-V3.2 performs concise end-stage checks over key assumptions and possible omissions, whereas earlier models often recompute the same conclusions through similar or alternative paths.

Motivated by this finding, we incorporate RDS as an auxiliary reward in RLVR. On DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI, 2025a), RDS-aware training reproduces the efficiency-driven pattern observed in newer models: it preserves the accuracy gains of standard RL while reducing both redundant derivation and response length by over 20%. By contrast, length-based rewards shorten responses but substantially weaken standard RL gains. These results reveal that models can gain reasoning ability without longer traces when optimization reduces redundant derivation rather than length itself.

2 Related Work

Case-by-Case Analysis of Problematic Reasoning Several recent works analyze problematic reasoning through close examination of individual chain-of-thought traces. Zhang et al. (2025) map reasoning traces into directed acyclic graphs (DAGs) to identify structural patterns such as loops, revisits, and branching, showing that models frequently revisit intermediate states and exhibit non-linear reasoning behavior. Focusing on answer generation, Kang et al. (2025) segment traces at each generated answer and measure when the first correct answer appears. They find that models often reach correct solutions early but continue generating reflection steps, and that training on reflection-rich rollouts improves first-answer correctness. Similarly, Chen et al. (2025) document cases where models continue elaborating after reaching a plausible or correct solution and show that removing later reasoning steps can degrade final accuracy. In contrast, Wang et al. (2025a) investigate underthinking by identifying traces in which models prematurely terminate reasoning, demonstrating that insufficient exploration can lead to shallow or incorrect solutions. Together, these studies reveal that extended reasoning is neither uniformly beneficial nor uniformly redundant, highlighting the need for a principled definition that isolates genuinely redundant derivations.

Reasoning Length as a Measure of Problematic Thinking Beyond case-by-case trace analysis, another line of research evaluates problematic thinking behavior through aggregate measures of reasoning length.Aggarwal et al. (2025) construct separate overthinking and underthinking benchmarks based on easier and harder tasks, respectively, to test whether models adjust reasoning length according to task difficulty. They show that *think models* tend to generate more tokens than necessary on simpler tasks, while *instruct models* under-generate on more complex tasks. Similarly, Singh et al. (2025) use the joint behavior of accuracy and reasoning length as an indicator of appropriate thinking depth. In a different direction, Yong et al. (2025) analyze the relationship between reasoning length and model confidence using token-level predictive entropy. They track how entropy evolves as additional reasoning tokens are generated across correct and incorrect solutions, datasets, and model variants. Their analysis shows that entropy generally decreases as reasoning progresses, and that longer reasoning traces tend to concentrate probability mass toward the eventual answer, often increasing model confidence regardless of correctness. While these approaches enable comparative evaluation of reasoning efficiency and optimality across models and datasets, they rely on

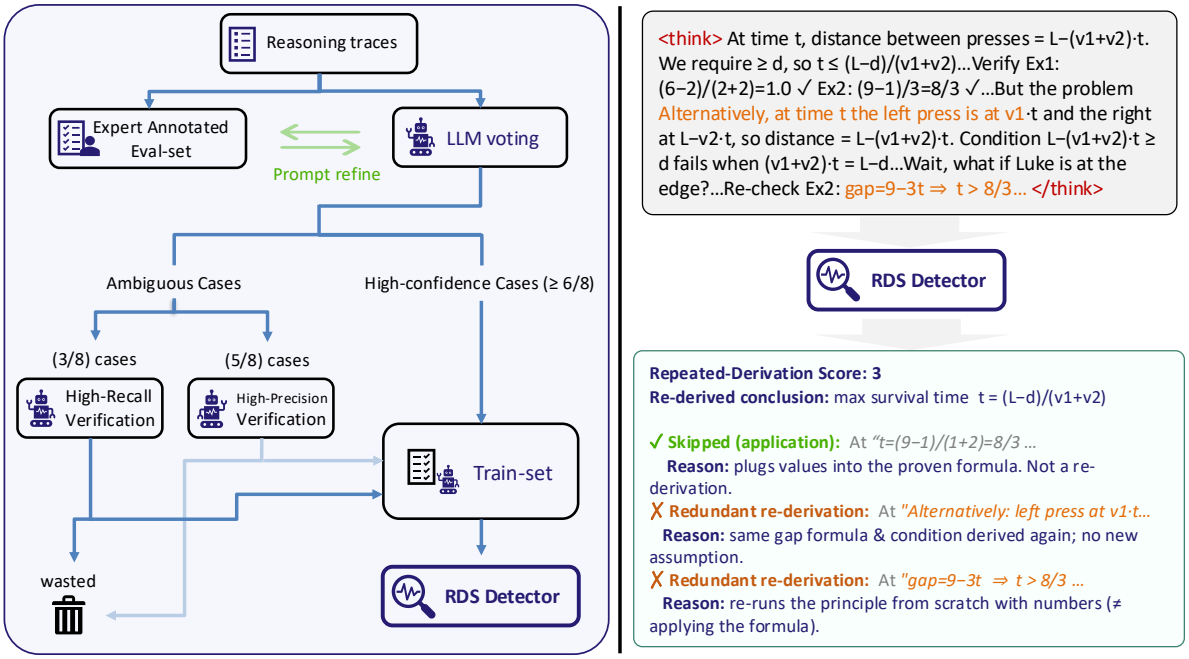

Figure 2: Pipeline for constructing training data for the RDS detector (left), and example reasoning traces with corresponding RDS annotations (right).

benchmark design and aggregate statistical patterns. This highlights the need for a benchmark-agnostic, instance-level measure of redundancy within individual reasoning traces.

Improving Reasoning Efficiency in RLVR. Beyond analyzing reasoning behavior, another line of research seeks to directly improve reasoning efficiency during reinforcement learning. A common approach modifies the RL objective by introducing length-based penalties as an auxiliary term alongside correctness rewards, encouraging shorter reasoning while maintaining accuracy (Arora and Zanette, 2025; Yi et al., 2025; Team et al., 2025b). Other methods adjust the training dynamics rather than the reward function itself. For example, Shrivastava et al. (2025) increase rollout group size and selectively update on shorter samples to mitigate length inflation during GRPO training. Aggarwal and Welleck (2025) introduce prompt-conditioned length constraints into the RL objective, enabling controllable trade-offs between reasoning length and performance. Yao et al. (2025) encourage concise reasoning by prompting self-rewriting when half of rollout samples are already correct. Meanwhile, Wang et al. (2025b); Wu et al. (2025) train auxiliary judging models to explicitly evaluate and reward concise reasoning. Collectively, these approaches demonstrate that reasoning length can be reduced during RL training alongside performance gains. However, in several works, we notice comparisons against strong vanilla RL baselines are limited or absent. As a result, it is difficult to fully disentangle efficiency improvements from potential trade-offs relative to standard RL training.

3 Benchmarking Reasoning Efficiency

3.1 Redundant Derivation Score

To study whether stronger reasoning necessarily requires longer reasoning, we need a measure of reasoning efficiency that is independent of final-answer correctness and response length. We therefore introduce the **Redundant Derivation Score (RDS)**, a direct measure of repeated derivation in reasoning traces. RDS counts cases where a trace re-derives an already established conclusion under unchanged premises and constraints. The conclusion may be either intermediate or final, and the repeated step must involve a substantive derivation rather than a simple restatement.

Formally, let a reasoning trace be represented as an ordered sequence of steps $\{s_1, \ldots, s_T\}$. At each step t, the trace induces an inferential state $\mathcal{I}_t$, consisting of the conclusions and operative constraints established so far. Let c denote an intermediate or final conclusion derived under premises and constraints $\mathcal{P} \subseteq \mathcal{I}_{t-1}$. A later step consti-

Model	AIME 2025			HMMT			LiveCodeBench			Avg.		
	RDS/Ratio	Len	Acc	RDS/Ratio	Len	Acc	RDS/Ratio	Len	Acc	RDS/Ratio	Len	Acc
deepseek-ai												
DeepSeek-R1-0120	2.42/58.8	10.8k	69.2	2.00/52.5	12.3k	44.6	2.54/58.8	10.4k	64.0	2.45/57.8	10.7k	61.2
DeepSeek-R1-0528	2.29/57.1	21.2k	86.7	2.15/50.0	25.9k	74.2	2.62/59.3	10.3k	64.4	2.53/57.7	13.3k	67.1
DeepSeek-V3.2	1.04/25.0	15.1k	91.5	0.99/24.0	18.1k	86.7	1.14/26.1	11.7k	81.1	1.11/25.7	12.9k	82.5
Distilled-Qwen32B	2.18/50.8	12.2k	57.6	2.34/52.5	13.8k	37.1	3.00/65.6	12.1k	55.9	2.85/62.7	12.4k	53.1
zai-org												
GLM-4.5-FP8	2.55/67.0	23.2k	83.3	2.35/53.5	27.0k	76.3	2.27/46.0	14.4k	70.5	2.30/48.3	16.8k	72.1
GLM-4.6-FP8	1.35/31.2	18.6k	94.2	1.58/31.7	27.6k	84.2	1.31/26.0	17.1k	81.8	1.36/27.1	18.8k	82.8
GLM-4.7-FP8	1.32/30.4	22.5k	94.2	1.23/29.2	29.6k	95.4	1.50/33.2	22.7k	84.2	1.45/32.4	23.8k	86.5
openai												
gpt-oss-120b-high	1.09/20.9	18.1k	91.7	1.52/28.8	24.7k	82.0	1.34/26.0	19.3k	74.8	1.36/26.2	20.1k	76.8
gpt-oss-120b-med	0.41/8.6	5.1k	80.0	0.28/8.7	6.2k	53.8	0.54/14.4	2.8k	72.8	0.49/13.2	3.4k	70.2
gpt-oss-120b-low	0.24/5.0	1.4k	53.3	0.26/6.7	1.2k	27.1	0.32/9.7	0.6k	61.4	0.31/9.0	0.8k	55.6
Qwen												
Qwen3-235B	2.86/63.9	12.3k	76.7	1.90/51.7	13.1k	60.0	3.02/66.1	12.3k	65.1	2.84/63.7	12.4k	64.9
Qwen3-235B-2507	3.21/64.2	22.3k	89.2	2.53/59.1	30.6k	78.3	2.33/48.7	19.2k	74.8	2.41/51.1	21.2k	76.1
Qwen3-235B-2507*	3.26/73.9	23.6k	90.0	2.43/64.2	31.3k	78.2	2.22/45.8	19.9k	74.6	2.31/50.1	21.9k	76.0
QwQ-32B	2.68/66.7	15.3k	70.8	2.18/55.8	18.0k	46.9	2.62/61.4	13.2k	60.4	2.56/60.8	14.1k	58.8

Table 1: RDS Benchmark across model families. For each benchmark we report RDS and high RDS ratio (RDS/Ratio), reasoning length in thousands of tokens (Len, k), and accuracy (Acc). Models marked with * denote evaluation with temperature 1.0.

tutes a *re-derivation* of c if it again establishes the same semantic conclusion under the same operative premises and constraints, regardless of differences in reasoning path or surface form. To be counted as a re-derivation, the step must involve substantive inferential work toward deriving c; merely restating, recalling, or referring to a previously established conclusion does not count.

To accommodate reasonable verification, a single re-derivation of a conclusion is treated as a permissible double-check and receives an RDS of zero. Once any conclusion is re-derived two or more times under unchanged premises and constraints, the trace is considered to exhibit redundant derivation, and all such re-derivations are counted toward RDS.

This design has two important consequences. First, because RDS is defined independently of both final-answer correctness and reasoning length, it allows us to analyze the relationship among reasoning efficiency, task accuracy, and trace length without conflating these quantities. Second, it avoids judging whether each step is beneficial to the final answer, which is difficult to assess reliably: reasoning behaviors such as verification, exploration, and self-correction may appear unnecessary in isolation yet still affect the final outcome. For example, Kang et al. (2025) show that removing reasoning after the first correct answer in training rollouts can hurt performance.

To apply RDS at scale, we train an automatic RDS detector over long reasoning traces. We first construct a corpus of more than 30,000 traces generated by multiple reasoning models across math, coding, logic, and instruction-following tasks. From this corpus, we manually annotate 200 traces with RDS labels and justifications, and use them to calibrate an annotation pipeline based on repeated long-context judgments. As shown in Figure 2, the pipeline combines majority voting with targeted filtering of ambiguous cases, yielding over 20,000 filtered training examples. We then fine-tune Qwen3-14B as the detector, which achieves a weighted F1 of 0.84 on the held-out manual evaluation set. Full annotation prompts, filtering rules, and training details are provided in Appendix A.

3.2 Benchmark Setup

Using RDS, we benchmark reasoning efficiency across a range of recent reasoning model families, versions, and updates. Our benchmark includes successive releases from the DeepSeek

HMMT: Let S be the set of all nonconstant monic polynomials P with integer coefficients satisfying $P(\sqrt{3}+\sqrt{2}) = P(\sqrt{3}-\sqrt{2})$. If $Q \in S$ has minimal degree, compute the only possible value of $Q(10) - Q(0)$.

DeepSeek-R1 0120

- Define $\alpha = \sqrt{3}+\sqrt{2}$, $\beta = \sqrt{3}-\sqrt{2}$.
- Recognize reciprocal structure.
- Compute $\alpha\beta = 1$, hence $\beta = 1/\alpha$.
- Derive degree 4 minimal polynomial $x^4 - 10x^2 + 1$.
- Derive degree 2 fails coefficient constraint.
- Derive cubic family $x^3 - 11x + c$.
- Observe that c remains free.
- Realize $P(\alpha)$ can be non-zero.
- Conclude that degree 3 is minimal.
- Derive degree 1 fails since $\alpha \neq \beta$.
- **Derive degree 2 fails coefficient constraint.**
- Conclude degree 3 constraints force $a = 0$, $b = -11$.
- Note c is free, min-degree solutions are $x^3 - 11x + c$.
- Compute $Q(10) - Q(0) = 890$.
- **Substitute numbers, show the diff is always 890.**
- Conclude the degree 4 difference is 9000.
- **Revisit $P(x) = x^3 - 11x + c$ satisfies $P(\alpha) = P(1/\alpha)$.**
- **Revisit $P(x) = x^3 - 11x + c$ satisfies again.**

Token:6539

DeepSeek-R1 0528

- Define $\alpha = \sqrt{3}+\sqrt{2}$, $\beta = \sqrt{3}-\sqrt{2}$.
- Recognize reciprocal structure.
- Compute $\alpha\beta = 1$, hence $\beta = 1/\alpha$.
- Derive degree 4 minimal polynomial $x^4 - 10x^2 + 1$.
- Note Galois/automorphism (τ: $\sqrt{2} \mapsto -\sqrt{2}$, $\sqrt{3}$ fixed)
- Infer $P(\alpha)$ lies in the τ-fixed field $Q(\sqrt{3})$.
- Reformulate condition as $P(\alpha)$ fixed under τ.
- **Derive minimal polynomial $x^4 - 10x^2 + 1$ again.**
- Derive min degree is 4 for form $P(x) = m(x)C(x) + c$.
- Derive degree 1 fails since $\alpha \neq \beta$.
- Derive degree 2 fails coefficient constraint.
- Derive cubic family $x^3 - 11x + c$.
- **Test cubic solution with numerical sanity check**
- Resolve rounding error in the sanity check
- Conclude degree 3 constraints force $a = 0$, $b = -11$.
- Note c is free, min-degree solutions are $x^3 - 11x + c$.
- Compute $Q(10) - Q(0) = 890$.
- **Investigate why min degree 4 is wrong.**

Token:12759

DeepSeek-V3.2

- Define $\alpha = \sqrt{3}+\sqrt{2}$, $\beta = \sqrt{3}-\sqrt{2}$.
- Recognize reciprocal structure.
- Identify α, β as conjugates under automorphism
- Note $\sqrt{2}$ and $\sqrt{6}$ coefficients need vanish
- Derive minimal polynomial $x^4 - 10x^2 + 1$.
- Note Galois/automorphism (τ: $\sqrt{2} \mapsto -\sqrt{2}$, $\sqrt{3}$ fixed)
- Infer $P(\alpha)$ lies in the τ-fixed field $Q(\sqrt{3})$.
- Compute $P(\alpha) - P(1/\alpha)$.
- **Note $\sqrt{2}$ and $\sqrt{6}$ coefficients need vanish again**
- Briefly mention degree 4 works.
- Derive degree 1 fails since $\alpha \neq \beta$.
- Derive degree 2 fails coefficient constraint.
- Derive cubic family $x^3 - 11x + c$.
- Conclude degree 3 constraints force $a = 0$, $b = -11$.
- Note c is free, min-degree solutions are $x^3 - 11x + c$.
- **Rapid checks on possible missing cases.**
- Compute $P(10) - P(0) = 890$.

Token:4989

DeepSeek-V3.2 Rapid checks:
"…Could there be degree 2? …no, because the condition forces $b = -2\sqrt{3}$, not an integer. So the minimal degree is 3? …Could **there be a degree 3 polynomial that is not of that form** if we consider …But is it possible that **there is a degree 1 or 2 monic polynomial if we allow some other interpretation**?... Could there be **a degree 3 polynomial that is not monic?**... What about **degree 0**? Constant polynomials are not nonconstant…"

Figure 3: Normalized bullet-point reasoning traces of DeepSeek-R1-0120, 0528, and DeepSeek-V3.2 on a sampled HMMT question. Steps marked in orange denote additional reasoning beyond the most direct solution path. Steps formatted as **bold and underlined** indicate verification. At the bottom of the figure, we further extract representative checking behaviors exhibited by DeepSeek-V3.2.

family, including DeepSeek-R1-0120, DeepSeek-R1-0528 (DeepSeek-AI, 2025a), DeepSeek-V3.2 (DeepSeek-AI, 2025b), and DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025a); the GLM family, including GLM-4.5 (Team et al., 2025a), GLM-4.6 (Z.ai, 2025a), and GLM-4.7 (Z.ai, 2025b); the gpt-oss-120b model (OpenAI, 2025) under different reasoning-efforts; and the Qwen family, including Qwen3-235B, Qwen3-235B-2507 (Team, 2025a), and QwQ-32B (Team, 2025b).

We evaluate these models on challenging math and coding tasks, including AIME25, HMMT, and LiveCodeBench2025 (Jain et al., 2024). For each model and dataset, we report four complementary metrics: average RDS, high-RDS ratio, reasoning length, and task accuracy. The high-RDS ratio is defined as the proportion of samples whose RDS is at least 3. Unless otherwise specified, all models are evaluated under their recommended temperature and top-p settings, and each query is sampled four times.

3.3 Efficiency-Driven Reasoning Gains

Our benchmark reveals an emerging efficiency-driven improvement pattern, in which models achieve stronger reasoning without producing longer traces. As shown in Figure 1, this pattern is most visible in the transitions from DeepSeek-R1-0528 to DeepSeek-V3.2 and from GLM-4.5 to GLM-4.6. In both cases, newer models achieve substantially higher performance with comparable or shorter reasoning length and markedly lower RDS.

Table 1 provides a more detailed comparison. From DeepSeek-R1-0120 to DeepSeek-R1-0528, accuracy improves substantially, but this improvement is accompanied by much longer reasoning traces while RDS remains nearly unchanged. On average, reasoning length increases from 10.7k to 13.3k tokens, while the high-RDS ratio remains above 57%. By contrast, DeepSeek-V3.2 further improves accuracy while reducing both reasoning length and redundancy: it achieves the highest average accuracy in the DeepSeek family, lowers average reasoning length to 12.9k tokens, and reduces the high-RDS ratio from over 57% to about 26%.

A similar pattern appears in the GLM family. GLM-4.6 improves over GLM-4.5 while substantially reducing RDS and the high-RDS ratio. Although its average reasoning length increases slightly, this increase is small relative to its performance gain and is accompanied by a large reduction in redundant derivation.

Other model updates exhibit a more length-driven improvement pattern. For example, Qwen3-235B-2507 improves accuracy over Qwen3-235B, but does so with substantially longer traces and high redundancy. Similarly, increasing the

reasoning-effort setting of gpt-oss-120b improves accuracy while also increasing reasoning length. This is especially clear from medium to high effort, where reasoning length increases sharply but the corresponding performance gain is more limited. The transition from GLM-4.6 to GLM-4.7 also partially follows this pattern. While the math benchmarks show little change in reasoning length, LiveCodeBench shows a noticeable increase in length, RDS, and accuracy.

We include several auxiliary settings to provide additional context. The distilled DeepSeek-R1 model exhibits higher RDS than its teacher model, suggesting that distillation does not necessarily preserve reasoning efficiency. We also evaluate Qwen3-235B-2507 under a different sampling temperature and observe similar RDS and accuracy trends, indicating that the main pattern is not driven solely by a single decoding configuration.

Overall, the benchmark suggests that reasoning gains are beginning to decouple from reasoning length. Newer models can improve not only by spending more tokens, but also by reducing redundant derivation and making each reasoning step contribute more useful inferential progress.

3.4 Case Analysis

To further examine how gains in reasoning efficiency emerge across model updates, we conduct a qualitative comparison within the DeepSeek family by analyzing reasoning traces from DeepSeek-R1-0120, DeepSeek-R1-0528, and DeepSeek-V3.2 that correctly solve the same problem. As illustrated by a representative example in Figure 3, DeepSeek-R1-0120 follows a largely single-track trajectory but exhibits limited upfront planning relative to its later variants. It first checks the degree-4 and degree-2 cases before concluding that degree 3 is minimal, yet later revisits lower-degree cases again. Near the end of the trace, it further reveals residual uncertainty by repeatedly recomputing the same core conclusion using essentially the same method.

DeepSeek-R1-0528 adopts a more exploratory style. After parsing the question, it begins with an automorphism-based approach and derives a degree-4 polynomial before eventually arriving at the minimal-degree solution. The trace also drifts away from the original goal: even after reaching the correct answer, it continues by analyzing why the earlier degree-4 derivation was incorrect. Verification remains prominent, but unlike R1-0120, it is carried out through alternative checking strategies, in this case numerical substitution. This continued exploration and post-solution analysis likely explain why R1-0528 produces substantially longer traces despite having an RDS comparable to that of R1-0120.

Putnam-style: Find the locus of centers (a, b) of circles passing through (1, 2) that intersect the parabola $y = x^2$ in exactly two points.

- Set up the circle–parabola intersection as a quartic equation.
- Interpret the boundary case as tangency between the circle and parabola.
- Derive a parametric locus from the tangency condition.
- Attempt algebraic elimination; abandon it as too complex.
- Try conic fitting from sample points; the conic hypothesis fails.
- Return to algebra using F(x) and F'(x); derive Eq C.
- Attempt elimination through substitution using Eq C.
- The derivation collapses back to Eq C.
- Recognize: "This is exactly Eq C earlier. So we are back where we started."
- Redirect: **eliminate a different variable between the core equations.**
- Derive a cleaner condition and continue.
- Test candidate closed forms; reject those inconsistent with samples.
- Report the final locus in implicit/parametric form.

Figure 4: Example gpt-oss-120b reasoning trace illustrating reasoning-state awareness.

DeepSeek-V3.2, by contrast, follows a similarly exploratory trajectory to R1-0528 but exhibits a markedly different verification structure. Instead of repeatedly recomputing the same result or extending verification through alternative checks such as numerical substitution, it performs a brief end-stage review. It revisits only key intermediate steps, checks for possible omissions, and proceeds once the remaining uncertainty is resolved (see the bottom of Figure 3).

We further examine reasoning traces from gpt-oss-120b, which exhibits particularly low RDS in our benchmark. Unlike DeepSeek-V3.2, whose efficiency mainly appears in compact and targeted verification, gpt-oss-120b often reduces redundancy through explicit reasoning-state awareness: when it detects that a derivation has returned to an already explored state, it redirects rather than continuing the same path. As shown in Figure 4, one rollout notes: *"Wait! This is exactly Eq C earlier. So we are back where we started. So nothing new,"* before switching to a different algebraic elimination strategy.

We provide the full rollouts for each representative case discussed here, along with additional comparisons within the DeepSeek family, in the Appendix B.

4 Optimizing Reasoning Efficiency

The patterns observed in our RDS Benchmark motivate a natural question: *can we explicitly*

Setting	Metric	Avg	AIME24	AIME25	CMIMC	HMMT	GPQA	Math500	AMC23
				DeepSeek-R1-Distill-Qwen-1.5B					
SFT	Acc	51.6	20.0	22.5	14.1	9.6	7.1	79.9	62.8
	Len	6.9k	15.5k	13.4k	15.1k	15.3k	6.7k	4.4k	8.1k
	OT	54.4%	56.0%	49.5%	52.8%	55.5%	60.0%	26.3%	46.9%
Vanilla GRPO	Acc	56.9	29.4	27.1	19.4	16.3	9.3	85.8	71.6
	Len	6.9k	14.0k	12.4k	13.8k	13.7k	6.0k	5.0k	8.8k
	OT	58.5%	50.3%	54.7%	59.6%	62.2%	70.0%	50.7%	47.8%
RDS Penalty	Acc	56.3	30.8	25.6	18.8	15.1	9.5	84.8	73.4
	Len	5.5k	12.5k	12.0k	12.8k	12.4k	4.5k	3.6k	7.0k
	OT	36.7%	32.0%	50.7%	48.4%	48.7%	70.9%	23.2%	44.4%
Len Penalty	Acc	52.3	30.4	22.1	18.8	14.6	6.9	78.9	73.8
	Len	5.3k	14.6k	14.0k	14.2k	14.7k	3.1k	3.2k	6.9k
	OT	35.1%	47.6%	49.6%	52.9%	58.3%	51.0%	22.2%	40.1%

Table 2: RLVR results. Metrics: accuracy (Acc), reasoning length in tokens (Len), and RDS ratio (RDS).

steer models toward the efficiency-driven trajectory of improvement, in which accuracy increases without lengthening reasoning traces? To this end, we adopt a Reinforcement Learning with Verifier-based Rewards (RLVR) framework based on GRPO, and incorporate our RDS as an auxiliary signal to discourage redundant re-derivations while preserving task performance.

4.1 RDS Auxiliary Reward

To incorporate the RDS as an auxiliary signal while avoiding reward hacking, we adopt the length-penalty formulation introduced in the Kimi-1.5 report (Team et al., 2025b), replacing the length term with the RDS. Specifically, given k sampled responses $(y_1, z_1), \ldots, (y_k, z_k)$ for a problem x with ground-truth answer y^*, let $\mathrm{RDS}(i)$ denote the RDS of response (y_i, z_i). We further denote by $\min_j \mathrm{RDS}(j)$ and $\max_j \mathrm{RDS}(j)$ the minimum and maximum RDS within the sampled group, respectively. We define the auxiliary RDS reward as

$$\mathrm{reward}_{\mathrm{RDS}}(i) = \begin{cases} \lambda_i, & \text{if } r(x, y_i, y^*) = 1, \\ \min(0, \lambda_i), & \text{if } r(x, y_i, y^*) = 0, \end{cases}$$

where

$$\lambda_i = 0.5 - \frac{\mathrm{RDS}(i) - \min_j \mathrm{RDS}(j)}{\max_j \mathrm{RDS}(j) - \min_j \mathrm{RDS}(j)}.$$

We combine the RDS reward with the correctness reward using a weight of 0.01, and introduce an additional handling of all-negative groups to account for differences in the RLVR algorithm. Unlike Kimi, which filters out groups where all sampled responses are incorrect, GRPO retains such groups. We therefore disable the auxiliary term when a group contains no correct responses. This design ensures that no incorrect answer can obtain a positive advantage after group-wise normalization. The final reward for response i is defined as

$$R_i = r(x, y_i, y^*) + 0.01 \cdot \mathbb{1}_{\text{pos}} \cdot \mathrm{reward}_{\mathrm{RDS}}(i).$$

where $\mathbb{1}_{\text{pos}}$ equals to 1 if the group contains at least one correct response.

4.2 Experimental Setup

We conduct experiments on supervised fine-tuned (SFT) models: DeepSeek-R1-Distill-Qwen-1.5B. models are further optimized on the DAPO-Math-17k dataset using the Group Relative Policy Optimization (GRPO) algorithm (Shao et al., 2024).

To assess the effect of RDS-aware supervision, we compare three training configurations: (1) vanilla GRPO with a binary correctness reward; (2) GRPO augmented with the proposed RDS auxiliary reward; and (3) GRPO augmented with a length-based auxiliary reward following the Kimi-1.5 formulation. The auxiliary weight and the handling of all-negative groups are kept identical across (2) and (3), allowing a direct comparison between RDS and length signals.

Across all configurations, we use a batch size of 64 and a rollout group size of 8. Sampling is performed with temperature 0.7 and top-p 1.0, and the maximum decoding length is set to 32k tokens. Additional implementation details are provided in the Appendix C. Each model is trained for 250 optimization steps. We then evaluate the resulting checkpoints on a diverse suite of mathematical reasoning benchmarks spanning multiple difficulty levels, including AIME 2024, AIME 2025,

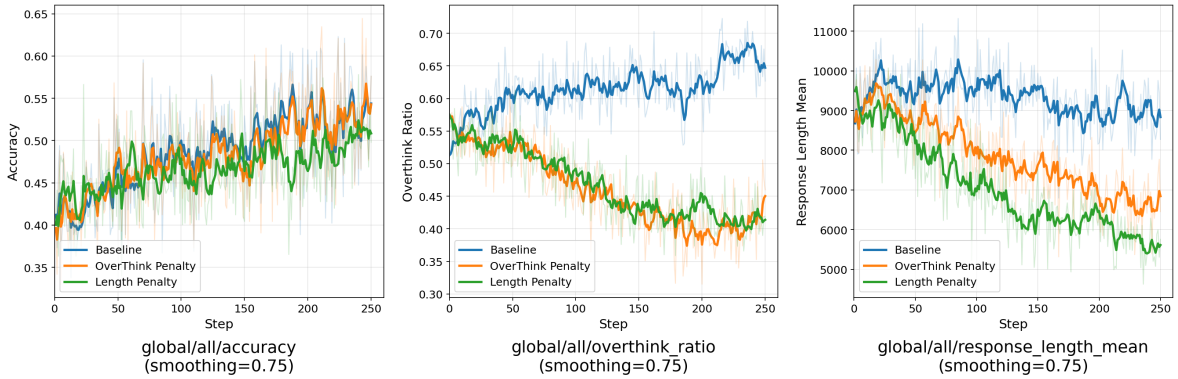

Figure 5: Training dynamics of DeepSeek-R1-Distill-Qwen-1.5B under vanilla RL, RL with overthinking penalty, and RL with length penalty.

AMC 2023, CMIMC, GPQA-Diamond (Rein et al., 2023), HMMT, and MATH500 (Hendrycks et al., 2021). During evaluation, each problem is sampled 8 times using the same temperature and top-p settings as in training to ensure consistency.

4.3 Main Results

Our training dynamics exhibit the efficiency-driven improvement pattern identified in the RDS Benchmark: accuracy increases while both reasoning length and RDS are reduced (Figure 5). RL with RDS penalty reward achieves an accuracy trajectory nearly identical to vanilla GRPO, yet induces a markedly different evolution in reasoning behavior. Whereas vanilla GRPO improves correctness while maintaining similar reasoning length and increasing the RDS rate by 7 percentage points, RL with RDS penalty reward progressively shortens responses and reduces RDS by 10 percentage points.

In contrast, RL with length penalty reward also reduces response length, but its accuracy growth begins to diverge from both vanilla GRPO and RL with RDS penalty reward after approximately 175 training steps, ultimately converging to a slightly lower performance level.

This divergence persists under held-out evaluation shown in Table 2. RL with RDS penalty reward improves accuracy by approximately 5 percentage points over the SFT baseline, matching the gain achieved by vanilla GRPO. Crucially, this improvement is accompanied by a reduction of 1.4k tokens in average reasoning length and a decrease in RDS rate from 54.4% to 36.7%. By comparison, RL with a length-penalty reward achieves similar reductions in length and RDS rate, but delivers smaller performance gains, improving by only 1 percentage point over SFT—about 4 percentage points below both vanilla RL and RL with an RDS-penalty reward. This contrast suggests that directly penalizing length can suppress useful reasoning along with redundancy, whereas RDS provides a more targeted efficiency signal: it reduces redundant derivation and shortens responses without sacrificing the performance gains of standard RL.

5 Conclusion

In this work, we study whether stronger reasoning necessarily requires longer chain-of-thought traces. We introduce the Redundant Derivation Score (RDS), an instance-level measure that captures repeated derivations of already established conclusions under unchanged premises.

Using RDS, we benchmark reasoning efficiency across recent reasoning model families. Our benchmark shows that newer models can achieve higher accuracy with comparable or shorter traces and substantially lower RDS, suggesting that reasoning gains can arise from more efficient use of reasoning steps rather than simply longer chains of thought. Our qualitative analysis further shows that efficient reasoning is not simply shorter reasoning: models can preserve useful exploration and verification while avoiding repeated derivations, either through more targeted checking or explicit awareness of previously explored reasoning states.

Finally, we show that this efficiency-driven pattern can be encouraged during RL training. Incorporating RDS as an auxiliary reward preserves the accuracy gains of standard RL while reducing both redundant derivation and response length. Overall, our work suggests that reasoning efficiency should be measured not only by the length of the trace, but by whether each reasoning step continues to add new information.

Limitations

This work focuses on one specific form of reasoning inefficiency: repeated derivation of conclusions that have already been established under unchanged premises. While this captures an important and observable form of overthinking, it does not cover all possible inefficient reasoning behaviors. For example, a trace may be unnecessarily verbose, pursue irrelevant branches, or contain overly long but non-repetitive exploration without triggering a high RDS. Thus, RDS should be viewed as a targeted measure of redundant derivation rather than a complete measure of all reasoning inefficiency.

The benchmark is limited to a selected set of recent reasoning model families and challenging math and coding tasks. These settings are useful for studying long chain-of-thought reasoning, but they may not fully represent other domains such as open-ended writing, planning, tool use, multi-turn interaction, or domain-specific expert reasoning. Future work should examine whether the same efficiency-driven patterns appear in broader task settings and under different prompting or decoding configurations.

Finally, our RL experiments are conducted on a relatively small model and a mathematical reasoning training setup. The results show that RDS can serve as an effective auxiliary signal for reducing redundant derivation without sacrificing standard RL gains, but they do not establish that the same recipe will transfer unchanged to larger models, other domains, or different RL algorithms. Scaling RDS-aware optimization and studying its interaction with exploration, verification, and self-correction remain important directions for future work.

References

Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. Optimalthinkingbench: Evaluating over and underthinking in llms. *Preprint*, arXiv:2508.13141.

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

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

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.

DeepSeek-AI. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI. 2025b. Deepseek-v3.2: Pushing the frontier of open large language models.

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.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2024. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint*.

Liwei Kang, Yue Deng, Yao Xiao, Zhanfeng Mo, Wee Sun Lee, and Lidong Bing. 2025. First try matters: Revisiting the role of reflection in reasoning models. *Preprint*, arXiv:2510.08308.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

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.

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.

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. *Preprint*, arXiv:2508.09726.

Joykirat Singh, Justin Chih-Yao Chen, Archiki Prasad, Elias Stengel-Eskin, Akshay Nambi, and Mohit Bansal. 2025. Think right: Learning to mitigate under-over thinking via adaptive, attentive compression. *Preprint*, arXiv:2510.01581.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025a. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

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. 2025b. Kimi k1.5: Scaling reinforcement learning with llms. *Preprint*, arXiv:2501.12599.

Qwen Team. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025a. Thoughts are all over the place: On the underthinking of o1-like llms. *Preprint*, arXiv:2501.18585.

Yuhao Wang, Xiaopeng Li, Cheng Gong, Ziru Liu, Suiyun Zhang, Rui Liu, and Xiangyu Zhao. 2025b. Efficient reasoning via reward model. *Preprint*, arXiv:2511.09158.

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.

Jiashu Yao, Heyan Huang, Shuang Zeng, Chuwei Luo, WangJie You, Jie Tang, Qingsong Liu, Yuhang Guo, and Yangyang Kang. 2025. Incorporating self-rewriting into large language model reasoning reinforcement. *Preprint*, arXiv:2511.16331.

Jingyang Yi, Jiazheng Wang, and Sida Li. 2025. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. *Preprint*, arXiv:2504.21370.

Xixian Yong, Xiao Zhou, Yingying Zhang, Jinlin Li, Yefeng Zheng, and Xian Wu. 2025. Think or not? exploring thinking efficiency in large reasoning models via an information-theoretic lens. *Preprint*, arXiv:2505.18237.

Z.ai. 2025a. Glm-4.6. `https://z.ai/blog/glm-4.6`. Accessed: 2026-02-20.

Z.ai. 2025b. Glm-4.7. `https://z.ai/blog/glm-4.7`. Accessed: 2026-02-20.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. 2025. Do llms really need 10+ thoughts for "find the time 1000 days later"? towards structural understanding of llm overthinking. *Preprint*, arXiv:2510.07880.

A RDS Detector Construction

To operationalize our RDS, we construct a corpus of more than 30,000 reasoning traces generated by four open-source models—gpt-oss-120b, GLM-4.6, DeepSeek-R1, and Qwen3-32B-A3B (Team, 2025a)—across challenging benchmarks including AIME24, AIME25, LCB, and Zebralogic, together with a small set of instruction-following tasks. All traces are generated with a decoding limit of 64k tokens so that long reasoning trajectories are preserved in their original form.

From this corpus, we randomly sample 200 traces to form a manually annotated evaluation set. Each trace is annotated with an overthinking score, defined as the number of redundant re-derivations it contains. For each such instance, we additionally annotate its location and provide a brief justification verifying that the step re-derives an already established result under unchanged premises, introduces no new evidence, refinement, or edge-case correction, and reflects a genuine re-derivation rather than a brief restatement.

We then use this evaluation set to develop annotation prompts for the remaining corpus, adopting GLM-4.6 as the annotator due to its long-context capabilities. However, preliminary experiments reveal that single prompt-based judgments yield unstable predictions, likely because the task requires comparing prior and result states across long contexts while making fine-grained distinctions between inference and repetition. To mitigate this, we sample the model eight times per reasoning trace using a standard detection prompt and take the majority outcome as the initial label. This majority-vote procedure achieves a weighted F1 score of nearly 90% on our manual evaluation set. To further improve quality, we introduce a secondary filtering stage for cases near the decision boundary: traces with a clear consensus are accepted directly, while those with intermediate distributions (i.e., majority votes between 3/8 and 5/8) are re-evaluated using complementary high-recall and high-precision prompts. We retain only cases where these two prompts agree, discarding the rest. This procedure yields a filtered set of over 20,000 automatically annotated traces, as illustrated in Figure 2.

Using these filtered annotations, we then fine-tune Qwen3-14B as an automatic RDS detector. On the held-out manually annotated evaluation set, the resulting model achieves a weighted F1 score

of 0.84.

A.1 Prompt Templates

We use three prompt templates in the annotation pipeline. The *general detector* prompt identifies repeated derivations in a reasoning trace and produces candidate RDS annotations. For ambiguous cases, we further apply two complementary prompts: a *high-precision verifier*, which removes cases that may be justified restatements, applications, or condition changes, and a *high-recall scanner*, which re-examines the trace to detect missed re-derivations. All prompts require JSON-formatted outputs to support automatic parsing.

General RDS Detector Prompt. This prompt is used to identify candidate redundant derivations from a full reasoning trace.

> **Task.** You are given a passage containing a reasoning process that attempts to answer a question. Your task is to identify all derived conclusions in the passage that are overthought, regardless of correctness or validity.
>
> **Derived conclusion.** A derived conclusion is a concrete result obtained through explicitly written reasoning effort, such as deduction, calculation, verification, systematic search, code construction, or evidence extraction through analysis. Statements that are merely assumed, stated without reasoning effort, summarized, or used as structural transitions do not count as derived conclusions.
>
> **Redundant re-derivation.** A conclusion is counted as redundantly re-derived if the same derived conclusion is produced again under the same operative premises and constraints, without a meaningful refinement in scope, boundary, assumptions, or result. Re-derivation must involve comparable reasoning effort. Merely restating, recalling, or applying a previous conclusion does not count.
>
> **Exclusion rules.** Do not count a later step as redundant if it introduces a meaningful change in assumptions, adds or removes edge cases, corrects an earlier misconception, turns a hypothesis into a proven claim, or uses the previous conclusion as a premise for new reasoning.
>
> **Procedure.** Read the passage line by line. Each time the reasoning appears to reproduce a previously derived conclusion, record the earlier derivation, the later re-derivation, whether the premises and conclusion are unchanged, and whether the later instance is a restatement, application, or genuine re-derivation. If any condition is ambiguous, omit the case.
>
> **Output format.** If no cases are found, return:

```
{"bad_thinkings": []}
```

> Otherwise, return:

```
{
  "bad_thinkings": [
    {
      "re-derived conclusion": "...",
      "unjustified_instance1": "...",
      "unjustified_instance2": "...",
      "repeated_times": 3
    }
  ]
}
```

> Only include conclusions with more than one unjustified re-derivation, following the RDS scoring rule.

High-Precision Verification Prompt. This prompt is applied to candidate annotations produced by the detector. Its goal is to remove false positives.

> **Task.** You are given a reasoning passage and candidate evidence of re-derived conclusions. Your task is to verify whether any candidate instance is actually justified and should be removed.
>
> **Verification criteria.** A candidate should be removed if the later instance does not involve comparable reasoning effort, if it is only a reminder, summary, narrative restatement, or application of a prior result, or if the later derivation changes the assumptions, conditions, variables, edge cases, or proof status of the conclusion.
>
> **Procedure.** For each candidate conclusion, recheck every cited instance. Remove any instance that is justified under the criteria above. If a conclusion has no remaining unjustified re-derivations after verification, remove the entire entry.
>
> **Output format.** Return the corrected result in the same JSON format:

```
{
  "verified_result": [
    {
    "Unjustified Re-derivation conclusion": "...",
    "record of re-derived instance 1": "...",
    "record of re-derived instance 2": "...",
      "repeated_times": 3
    }
  ]
}
```

High-Recall Re-Scanning Prompt. This prompt is used to re-scan ambiguous traces with maximal sensitivity.

> **Task.** You are given a reasoning trace. Your task is to identify any overthinking where a derived conclusion is re-derived without a meaningful change in premises or result. Be maximally sensitive and do not miss subtle or paraphrased re-derivations.
>
> **Criteria.** An unjustified re-derivation must involve comparable reasoning effort, such as deduction, verification, systematic exploration, or coding. A later derivation counts if it repeats the same

HMMT: Four unit squares form a 2x2 grid. Each of the 12 unit line segments is colored red or blue so each square has 2 red and 2 blue sides. Count colorings.		
DeepSeek-R1 0120 • Parse; visualize 2x2 grid; 12 edges, 4 internal shared • Label all 12 edges by coordinate (Hl-H6, V1-V6); map squares • Write square constraints: sum of incident edges = 2 • Reformulate: boundary pair-sum = 2 - internals; 0/2->1, 1->2 • Enumerate 16 internal configs; multiply per-square factors • Tabulate: 2x1 + 12x4 + 2x16 = 82 • **Recheck arithmetic by cumulative re-summation over 16 cases** • **Sanity-check extremes: all-blue -> all-red; all-red -> all-blue** • Conclude answer = 82	DeepSeek-R1 0528 • Parse; 12 edges -> 4 internal shared + 8 boundary • Fix internals first (Vt,Vb,Hl,Hr); 16 configs; count = 2^f • f = # squares whose two interior edges differ in color • Split sum by Hl/Hr: 2x25 + 2x16 = 82 • Pause -- question whether sum covers all valid colorings • Justify: each full coloring <-> internal + boundary, 1-to-1 • **Spot-check extremes (all-red/all-blue internals)** • **Verify 2^f rule by explicit boundary enumeration** • Reorganize count via R_int red-internals as second approach • Sum R_int=0..4 contributions -> 82 again • Compare with probabilistic expectation • Conclude answer = 82	DeepSeek-V3.2 • Parse; enumerate 12 edges via 3x3 vertex coordinate model • List 4 squares and incident edges; note "exactly 2 reds" • Try parity (GF(2)) approach • Abandon parity: "even reds" weaker than "exactly two" • Split: 4 interior {M1,M2,C1,C2} shared; 8 boundary unique • Iterate 16 interior assignments; per-square count via C(2,k) • Introduce g(k); total = sum g(a+c)g(a+d)g(b+c)g(b+d) • Factorize: sum_{c,d}(sum_a g(a+c)g(a+d))^2 -> S=5,5,4,4 -> 82 • **Full 16-case enumeration table as second verification -> 82** • **Map given example into one interior case to confirm** • Conclude answer = 82

Figure 6: Additional DeepSeek-family comparison on a second query.

HMMT: On triangle ABC, A,D,E,B on AB (AD=4,DE=16,EB=8); A,F,G,C on AC (AF=13,FG=52,GC=26). M=reflection of D through F, N=reflection of G through E. area(DEGF)=288. area(heptagon AFNBCEM)=?		
DeepSeek-R1 0120 • Parse; AB=28, AC=91; reflections -> midpoint formulas • Try right-angle placement (AB on x, AC on y) • Compute area(DEGF)=624 -> contradicts 288 -> reject right angle • Notice 1:4:2 ratio; re-parameterize C=(91cos(t), 91sin(t)) • Shoelace area(DEGF) ~ sin(t); solve sin(t) from 288 • Shoelace heptagon AFNBCEM -> 588 • **Recheck each shoelace term -> 588 again** • Suspect self-intersection; area(triangle ABC)=588 -> paradox • **Recompute with decimal approximations -> 588 again** • Try alternate decomposition (AFN+FNBC+CEM) -> double-counting • **Verify area(DEGF)=288 holds; accept 588 despite concerns**	DeepSeek-R1 0528 • Parse; AB=28, AC=91; reflections -> M=2F-D, N=2E-G • Choose coordinate route; A=(0,0), B=(28,0), C=(xc,yc), AC=91 • Place D,E,F,G; anchor scale via area(DEGF)=288 (shoelace) • Shoelace heptagon AFNBCEM; xc cancels -> tilt-independent -> 588 • **Sanity-check: compute area(triangle ABC) -> 588, triggers doubt** • Explore why equal: net cancellation of extra/missing pieces • Add local area checks (triangle AFN, triangle EMA) for intuition • **Reaffirm via coordinate+shoelace robustness; area(DEGF)=288 holds** • Conclude 588	DeepSeek-V3.2 • Restate; AB=28, AC=91; reflections -> midpoint relationships • Set coords: A=(0,0), B=(28,0), C at angle t; M,N from midpoints • Shoelace area(DEGF) ~ sin(t); solve sin(t) from 288 • Shoelace heptagon AFNBCEM; cos(t) terms cancel -> 588 • Compare to (1/2)*AB*AC*sin(t); expressions match • **Recompute area(triangle ABC) -> 588 confirmed** • Question self-intersecting polygon / shoelace interpretation • Note answer independent of cos(t) sign choice • Conclude 588

Figure 7: Additional DeepSeek-family comparison on a third query.

conclusion under essentially identical premises with no meaningful change in assumptions, scope, or result. Restatements or applications without new reasoning do not count.

Procedure. Scan the passage carefully and identify every point where a previously established result is actively reconstructed or re-proven. For each detected case, explain why it meets the criteria for unjustified re-derivation.

Output format. If no cases are found, return:

```
{"bad_thinkings": []}
```

Otherwise, return:

```
{
  "bad_thinkings": [
    {
      "re-derived conclusion": "...",
      "reason": "...",
      "repeated_times": 2
    }
  ]
}
```

B Additional Qualitative Examples

In addition to the representative case discussed in Section 3, we provide two additional DeepSeek-family comparisons in Figures 6 and 7. Each figure presents normalized reasoning traces from DeepSeek-R1-0120, DeepSeek-R1-0528, and DeepSeek-V3.2 on the same query, highlighting how reasoning behavior changes across model updates. These examples further illustrate that DeepSeek-V3.2 tends to preserve useful exploration and verification while reducing repeated derivations of already established conclusions.

Full raw rollouts will be released together with

the detector model weights upon publication.

C RL Training Details

Table 3 summarizes the main RL training configuration used in our experiments.

Item	Value
Global batch size	64
Rollout group size	8
Mini-batch size	16
Training steps	250
Learning rate	1×10^{-6}
LR scheduler	Constant
Weight decay	1×10^{-2}
Adam β_1, β_2	0.9, 0.999
Adam ϵ	1×10^{-8}
Maximum response length	32,768 tokens
KL coefficient	0.0
Entropy coefficient	0.0

Table 3: Additional RL training configuration.

D Miscellaneous Details

D.1 Artifacts and Licenses

We use publicly available model checkpoints and benchmark datasets. The model artifacts used in this work include DeepSeek-R1-0120, DeepSeek-R1-0528, DeepSeek-V3.2, DeepSeek-R1-Distill-Qwen-32B, DeepSeek-R1-Distill-Qwen-1.5B, GLM-4.5, GLM-4.6, GLM-4.7, gpt-oss-120b, Qwen3-235B, Qwen3-235B-2507, QwQ-32B, and Qwen3-14B. All model checkpoints are accessed from Hugging Face or the corresponding official release pages, and are used consistently with their stated licenses and intended research usage.

The datasets used in this work are public benchmark or training datasets, including AIME, HMMT, LiveCodeBench, GPQA-Diamond, MATH500, AMC, CMIMC, Zebralogic, and DAPO-Math-17k. Our detector training data consist of model-generated reasoning rollouts over these public tasks and a small set of instruction-following prompts. We do not use private user data. For artifacts created by this work, including the RDS detector and the derived RDS annotations, we plan to release them for research use upon publication, together with a model card, intended-use statement, and license.

D.2 Data Privacy and Content

The data used in this work consist primarily of mathematical, coding, logic, and instruction-following problems, together with model-generated reasoning traces. We do not intentionally collect personal information, user identifiers, or private user-generated data. The manually annotated set is sampled from model-generated traces and is annotated only for reasoning structure, specifically repeated derivations under unchanged premises and constraints. We also do not infer demographic attributes or protected characteristics.

D.3 Computational Resources

The experiments were conducted on H100 GPUs. RL experiments use DeepSeek-R1-Distill-Qwen-1.5B and are trained with 32 H100 GPUs. A single RL run takes approximately five days. In addition to the final reported runs, we conducted multiple RL variants, detector training, benchmark generation, and detector-based evaluation. Because these experiments were run across several stages and configurations, we report the main reproducible training settings in Appendix C, including model, data, rollout group size, batch size, decoding length, sampling temperature, top-p, and number of optimization steps.

D.4 Human Annotation

The manual RDS evaluation set contains 200 reasoning traces annotated by the authors. No external crowdworkers or paid annotators were used. The annotation task involved model-generated reasoning traces rather than human-subject data. Annotators labeled repeated derivations, their locations, and brief justifications according to the RDS definition. The annotation instructions and prompt templates are provided in Appendix A.

D.5 Use of AI Assistants

AI assistants were used for language polishing and coding assistance during paper preparation. All scientific claims, experimental designs, annotations, results, and final text were reviewed and verified by the authors. AI assistants were not used as authors and do not replace author responsibility for the content of the paper.