Concise Math Reasoning via Difficulty-Aware Distillation

Anonymous ACL submission

Abstract

Human experts tackle difficult math problems by identifying and executing a few pivotal steps rather than listing every intermediate thought. In contrast, standard Chain-of-Thought (CoT) distillation trains small models on lengthy reasoning traces, encouraging a uniform overthinking style across easy and hard items alike. The result is rigid, slow solutions that sacrifice adaptivity. This approach stands in sharp contrast to human intuition. Humans naturally adapt their problem-solving strategy, dedicating significant effort to difficult problems while finding quick, simple solutions for easier ones. We argue that the root cause lies in the training data: it contains excess information and reasoning steps organized in ways misaligned with human practice. We address this with Difficulty-Aware Distillation(DAD), a procedure for producing training data that mirrors concise human reasoning. A large teacher model first assesses a problem's difficulty and then rewrites the solution to retain only the essential steps. Using this process, we constructed LiteCoT, a 100,000-example corpus of short, clear rationales, and used it to train our Liter models. With 100k LiteCoT, we outperform models trained on 800k long CoT and cut both training and inference costs. The advantage is consistent across standard math benchmarks, showing that concise, human-aligned data delivers equal or better accuracy with much less compute. For example, on the challenging AIME24 exam, our approach reaches 74.2% Pass@1 using only about 5K inference tokens, surpassing other methods that consume many more tokens.

1 Introduction

To improve the mathematical reasoning capabilities of large language models (LLMs), a common approach is to distill "Chain-of-Thought" (CoT) from powerful teacher models like DeepSeek-R1 (Guo et al., 2025) and OpenAI-o1 (OpenAI, 2025). However, this approach suffers from a fundamental flaw: the training data it relies on is inherently overly lengthy (Ye et al., 2025; Concise). These CoTs are filled with repetitive analytical steps, forcing the student model to learn a rigid pattern of "overthinking" in all mathematical problems (Yu et al., 2025a; Sui et al., 2025; Yin et al., 2025). This mechanical process is fundamentally misaligned with how humans approach mathematical problem solving: humans typically construct reasoning paths by focusing on a few key steps, without needing to record every intermediate thought. Notably, previous efforts aimed at improving data quality, such as data filtering (Muennighoff et al., 2025; Wen et al., 2025), synthesis (Yang et al., 2025) or selecting for difficulty (Ye et al., 2025) have largely ignored this core issue of verbosity. This neglect results in high training costs for current models and an inability to flexibly adjust their reasoning depth based on the actual difficulty of the mathematical problem.

Perhaps the key to improving the mathematical reasoning of LLMs is to have them reason more like humans—by focusing only on essential logical steps. This would require training data where solutions are concise and their length is adapted to the problem's difficulty. To explore this hypothesis, we introduce a simple method called Difficulty-Aware Distillation (DAD). DAD is designed to create this type of data: it uses a teacher model to assess a math problem's difficulty and then rewrites the lengthy original solution into a compact trace containing only the essential logical steps.

Using the DAD method, we built **LiteCoT**, a dataset of 100K concise samples averaging just 720 tokens—a stark contrast to the 5K-10K tokens in datasets like S1 (Muennighoff et al., 2025) and OpenThoughts (Team, 2025b). We used this data to train our **Liter** models. The results are compelling: across 9 benchmarks, models trained on our 100K LiteCoT samples consistently outperform those trained on 800K verbose examples. On the challenging AIME24 benchmark, Liter achieves 74.2%

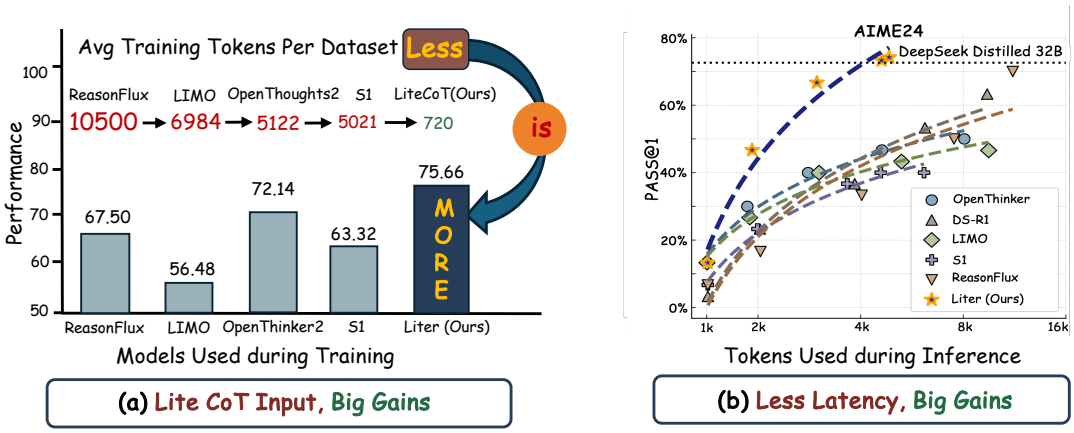

Figure 1: (a) Training tokens comparison of LiteCoT with prior distillation methods (32B model); (b) Inference efficiency and accuracy on the challenging AIME24 benchmark (32B model). Our Liter models trained with LiteCoT achieve higher accuracy (74.2% Pass@1) at substantially reduced inference token counts compared to prior methods.

Pass@1 with superior token efficiency, proving that for math reasoning, data quality and conciseness trump sheer quantity.

2 Related Work

2.1 LLM Distillation for Math Reasoning

Recent studies have shown that distilling the long Chain-of-Thought (CoT) reasoning from large teacher models to smaller student models is an effective method for enhancing their mathematical reasoning abilities (Guo et al., 2025; Wei et al., 2022). However, the training and inference costs of these methods remain high, with the root cause being that the CoT reasoning itself is overly verbose and consumes substantial computational resources. To alleviate this cost pressure, current mainstream research has focused on the data level, optimizing the training set by, for example, filtering for high-quality samples (Ye et al., 2025; Team, 2025b,a) or rewriting and synthesizing data (Yang et al., 2025; Guan et al., 2025). Nevertheless, these methods only address the issue of how much data to use, while overlooking a more central point: the selected reasoning chains themselves are still replete with unnecessary reasoning steps (Qu et al., 2025; Wu et al., 2025; Luo et al., 2025b). Little work has reflected on whether such lengthy reasoning chains are truly necessary for logically rigorous mathematical reasoning. We argue that the core of mathematical reasoning lies in logical clarity and procedural precision, not in the sheer volume of the process. Overly long reasoning chains not only introduce significant computational overhead, but their redundant information can even become "noise" that interferes with the model's learning performance (Lin et al., 2025; Hou et al.; Zhao et al., 2025b).

2.2 Reasoning Pruning via Budget Control

Beyond the aforementioned approaches that focus on data quantity, more advanced work has begun to directly address the cost issues caused by CoT length itself. However, most of these strategies concentrate on inference-time optimizations (Muennighoff et al., 2025; Teng et al., 2025), such as LightThinker (Zhang et al., 2025), Sketch-of-Thought (Aytes et al., 2025), and AdaR1 (Luo et al., 2025a), but they do not solve the efficiency problem during the training phase. In contrast, the exploration of training efficiency remains relatively underexplored. Current methods for this purpose fall into two main categories: (1) Learning-based approaches (Yu et al., 2025c; Zhao et al., 2025a) that train auxiliary models to produce shorter reasoning traces, which can adaptively switch between Long and Short CoTs but often incur substantial additional training costs. (2) Prompt engineering-based pipelines that heuristically rewrite CoTs without training extra models, such as DLCoT (Luo et al., 2025b) and Long-Short Chain-of-Thought Mixture (Yu et al., 2025b).

3 LiteCoT Dataset: Concise Reasoning with Big Gains

3.1 Difficulty-Aware Prompting for Curating LiteCoT

We now describe our difficulty-aware distillation (DAD) pipeline, designed to construct a concise

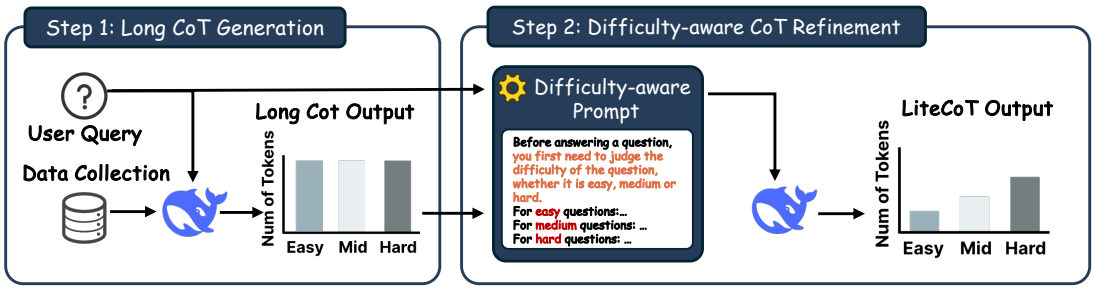

Figure 2: Overview of the DAD method. First, the DeepSeek R1 model generates initial Long CoT data from collected user queries. Then, a difficulty-aware prompt guides the model to rewrite concise CoT outputs, adapting reasoning length dynamically based on problem difficulty.

and difficulty-adaptive distilled reasoning dataset. The pipeline consists of two sequential steps: (1) generating initial long reasoning solutions, and (2) refining these chains using difficulty-aware prompts to produce shorter, complexity-aligned reasoning traces.

Step 1 (Long CoT Generation). In conventional distillation, a teacher LLM $M_{teacher}$ is used to generate a reasoning chain $CoT_{L,i}$ for each query Q_i in the training set D. Formally: $CoT_{L,i} = M_{teacher}(Q_i)$ for $i = 1, 2, \ldots, N$. The collected set of long chains is $C_L = (Q_i, CoT_{L,i})_{i=1}^{N}$. These Long CoTs often contain redundant content and do not adjust in length for query difficulty.

Step 2 (Difficulty-aware CoT Refinement). In this step, the teacher model takes each $(Q_i, CoT_{L,i})$ pair as input, along with a specially designed prompt P_{DA} that instructs it to produce a shorter solution appropriate for the difficulty of Q_i. This prompt P_{DA} provides different guidelines depending on whether a question is easy, medium, or hard. For example, it might include: "For easy questions: provide a brief solution. . . For hard questions: include all necessary detailed steps . . .". Guided by P_{DA}, the teacher first assesses the difficulty of the problem and then refines the solution accordingly:

1. **Difficulty Assessment**: Guided by P_{DA}, the model first assesses the difficulty of the input query Q_i, classifying it into a predefined set of difficulty levels $\mathcal{K} = \{k_{\text{easy}}, k_{\text{medium}}, k_{\text{hard}}\}$. This assessment process can be represented as:

$$d_i = M_{teacher}(Q_i, CoT_{L,i}, P_{DA}) \quad (1)$$

where $d_i \in \mathcal{K}$ is the assessed difficulty level. The prompt P_{DA} contains explicit instructions for different assessments.

2. **Difficulty-Adaptive CoT Refinement**: Based on the assessed difficulty level d_i, the model follows the sub-prompt or structured output template $P_{d_i} \subset P_{DA}$ within P_{DA} designed for that specific difficulty d_i. It then prunes, reorganizes, or guides the generation of a new, difficulty-matched short CoT, CoT_{S,d_i}. This process can be represented as:

$$CoT_{S,d_i} = M_{teacher}((Q_i, CoT_{L,i}), P_{DA}, d_i) \quad (2)$$

Or more specifically, if P_{d_i} is an explicit output template:

$$CoT_{S,d_i} = M_{teacher}(\text{input} = (Q_i, CoT_{L,i}), \text{template} = P_{d_i}(d_i)) \quad (3)$$

where $P_{d_i}(d_i)$ denotes the specific instruction part within P_{DA} for difficulty d_i (*e.g.,* "For easy questions: . . .", "For medium questions: . . .", "For hard questions: . . .").

Applying this process to all training samples yields a new distilled dataset $C_S = (Q_i, CoT_{S,i})i = 1^N$ comprised of shortened, difficulty-graded reasoning traces.

In addition, the refined CoTs are much shorter than the originals (for easy and medium queries, $Length(CoTS, i) \ll Length(CoT_{L,i})$). This adaptive reduction strategy is expected to significantly improve the efficiency of model training while preserving essential reasoning for challenging tasks. Ultimately, a student model trained on C_S should learn to solve problems with an appropriate amount of reasoning.

3.2 LiteCoT Overview

LiteCoT Overview. Using the DAD method, we created a distilled dataset, named LiteCoT, of 100K question–solution pairs with difficulty-adaptive CoTs. Table 1 highlights the token sav-

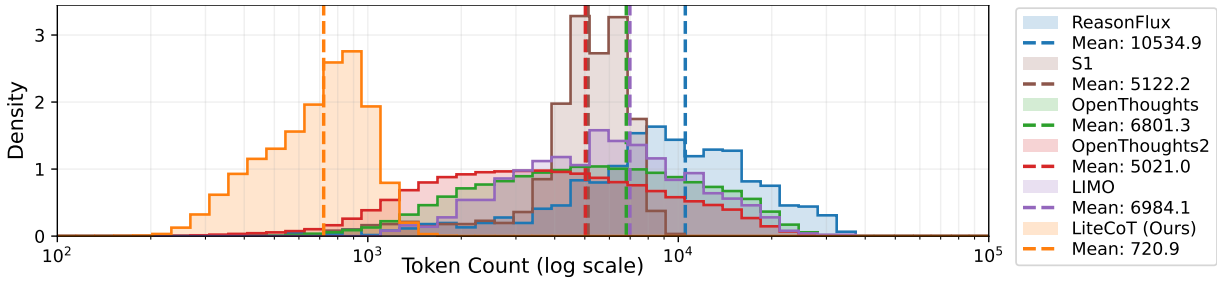

Figure 3: Token count distribution of LiteCoT compared with previous CoT distillation datasets.

Table 1: Performance comparison of different budget control methods. "Δ Tokens" denotes average tokens per CoT sample.

Datasets	#-Samples	Δ Tokens
Light-R1 (Wen et al., 2025)	75.9K	6382
OpenThoughts (Team, 2025b)	113.9K	6801
OpenThoughts2 (Team, 2025b)	1M	5021
S1 (Muennighoff et al., 2025)	1K	5122
LIMO (Ye et al., 2025)	817	6984
SKY-T1 (Team, 2025a)	16K	4095
ReasonFlux (Yang et al., 2025)	1K	10500
LiteCoT (ours)	100K	**720**

ings of our approach compared to prior distillation data. Other budget-control methods still produce CoT solutions averaging several thousand tokens each (*e.g.,* 5K–10K tokens for ReasonFlux (Yang et al., 2025) and OpenThoughts (Team, 2025b)), often necessitating tens or hundreds of thousands of training samples. In stark contrast, LiteCoT's CoT samples average only about 720 tokens, representing an order-of-magnitude improvement in token efficiency.

Figure 3 clearly shows this advantage: the token-count distribution for LiteCoT is markedly shifted toward shorter reasoning chains compared to other datasets.

Training the Liter with LiteCoT. Using LiteCoT, we distilled a new family of reasoning models, named `Liter` (1.5B, 7B, and 32B), based on the Qwen2.5. We followed the training strategy from Open-R1 (Face, 2025) and fine-tuned the Liter models for 3 epochs on the LiteCoT dataset. Specifically, we employed a learning rate of 5e-5 with a cosine learning rate scheduler, including a 10% warm-up phase. In addition, we increased the RoPE frequency to 100K, effectively extending the context length from 4K to 8K tokens.

ected to yield substantial benefits in terms of reduced training duration, lower computational resource demands, and potentially more focused learning for student models, the empirical validation of which will be discussed in the subsequent experimental sections.

4 Experiments

We conduct extensive experiments to evaluate the effectiveness of LiteCoT data. Our evaluations address three key questions: (**Exp-1**) Does training on short difficulty-aware CoTs, LiteCoT, actually improve model performance compared to long CoTs? (**Exp-2**) Can our DAD method rival or surpass models distilled with an order of magnitude more data from a powerful teacher? (**Exp-3**) How does our DAD approach compare to other recent CoT distillation and compression techniques? (**Exp-4**) Will DAD still be useful if we change another teacher model? (**Exp-5**) How many tokens can be saved using DAD method?

Due to space limitations, we have placed the details of Exp 4 and 5 in the appendix A.

4.1 Experimental Results and Analysis

4.1.1 Exp-1: LiteCoT vs. Long CoT Baselines

We evaluate the fine-tuned models trained on our LiteCoT and Long CoT data, comparing their overall performance and analyzing the accuracy-speed trade-off.

Overall Performance. As shown in Table 2, we comprehensively compare the average accuracy of Qwen2.5 series models and Llama3.1-8B trained with Short CoT and Long CoT on 11 different benchmarks. The average performances show that all models trained and distilled with short CoT data significantly outperform the corresponding models distilled with Long CoT data. For example, the overall average performance of Qwen2.5-1.5B-Instruct (short) reached 43.6%, while its Long CoT version was 35.7%; Qwen2.5-7B-Instruct (short) reached 57.3%, higher than the 53.3% of the Long CoT version. This trend is consistently reflected in

Table 2: Performance comparison of Long CoT distilled models and short CoT distilled models. Green means short CoT distilled models outperform the corresponding Long CoT distilled models. Pink means the opposite ("QW" and "LM" refer to Qwen2.5-Instruct and Llama3.1-Instruct).

Benchmark	QW-0.5B		QW-1.5B		QW-3B		QW-7B		QW-14B		LM-8B	
	short	long	short	long	short	long	short	long	short	long	short	long
MATH500	18.6	11.6	62.8	56.4	47.8	46.0	66.2	64.8	65.0	64.4	34.2	32.2
AIME24	0.0	0.0	3.3	0.0	3.3	0.0	23.3	3.3	13.3	10.0	3.3	0.0
GSM8K	33.6	20.5	85.1	84.4	77.3	61.4	95.5	95.0	90.8	85.1	73.7	26.8
Minerva Math	3.7	4.4	25.4	19.5	14.3	11.8	33.9	31.2	29.0	21.3	10.3	7.7
GaoKao2023	23.4	11.9	65.2	62.9	43.3	39.5	66.5	66.2	56.9	49.9	32.2	25.2
OlympiadBench	5.9	1.5	37.2	36.3	15.4	10.8	42.8	37.5	28.4	19.1	13.0	11.6
College Math	19.6	9.8	46.9	47.2	35.6	31.2	48.5	47.5	41.9	32.1	24.7	1.61
Sat Math	37.5	12.5	43.8	3.1	68.8	34.4	96.9	90.6	96.9	62.5	43.8	25.0
MMLU STEM	29.7	9.2	33.9	10.4	58.1	21.4	64.4	60.5	71.3	56.9	33.9	29.0
ACM23	2.5	0.0	47.5	45.0	30.0	17.5	62.5	57.5	42.5	27.5	5.0	25.0
Overall	17.5	8.1	45.1	36.5	39.4	27.4	60.1	55.4	53.6	42.9	27.4	18.4

models of different parameter sizes.

The relationship between inference speed and accuracy. Figure 4 clearly shows that for all models (Llama3.1-8B and Qwen2.5 series models), the use of Short CoT (filled points) generally achieves inference time that is much lower than Long CoT (hollow points). More importantly, this significant speed improvement does not come at the expense of accuracy; on the contrary, in most benchmarks, Short CoT is not only faster, but its accuracy can also reach or even exceed the level of Long CoT. For example, on multiple benchmarks such as GSM8K and MMLU STEM, Short CoT data points all show a better "speed-accuracy" balance. This finding spans models of different parameter sizes and a variety of task types, and strongly proves that the DAD Short CoT method can significantly improve inference efficiency while maintaining or even optimizing the model's inference accuracy.

4.1.2 Exp-2: Comparing our Liter Models with Mainstream Distilled LLMs

We first compared our Liter models against models distilled from DeepSeek-R1 (**Exp-2.1**). To further evaluate the effectiveness of our `Liter` models, we compared our Liter 7B and 32B variants against a broad range of mainstream distilled models across multiple reasoning benchmarks (**Exp-2.2**).

Exp-2.1: Comparison with DeepSeek-R1-Distilled Models. Figure 5 shows the performance of four models of different parameter sizes (1.5B, 7B, 14B and 32B) in the Qwen2.5 series on six key benchmarks after three training methods: original weights (Vanilla), DeepSeek-R1 distilled models (DeepSeek-R1-Distilled) and our Liter. It shows that the performance of the model representing Liter (Gray blue area) is consistent and significantly better than the other two methods at all parameter scales. Specifically, taking the Qwen2.5-7B model as an example, our method achieved an accuracy of 60.6% on Math500, while DeepSeek-R1 distilled and the original model achieved 55.8% and 42.6% respectively; on the more challenging OlympiadBench, our model scored 30.2%, also ahead of DeepSeek R1 distilled's 21.1% and the original model's 10.0%. This performance advantage is particularly evident in the Qwen2.5-32B model with a larger number of parameters. For example, on the AIME24 benchmark, the accuracy of the DAD distilled model is as high as 76.7%, far exceeding the 48.0% of DeepSeek R1 distillation and the 13.3% of the original model. These specific numerical comparisons strongly prove that the model trained on the LiteCoT can achieve comprehensive performance superiority over other training methods on multiple core benchmarks

Exp-2.2: Comparison with Mainstream RLMs of Similar Parameter Sizes. We conducted an extensive horizontal comparison involving our Liter models (1.5B, 7B, and 32B) against various publicly available mainstream reasoning models of comparable parameter scales. Evaluations were carried out on five representative benchmarks: AIME24, MATH500, GPQA, AMC23, and OlympiadBench, with results presented in Table 3.

Overall, whether at the parameter scale of 1.5B, 7B, or 32B, the models trained by the DAD distillation method show comparable or better performance than the current mainstream or even top reasoning models. At the 1.5B level, Ours-1.5B achieved an accuracy of 87.2% on MATH500,

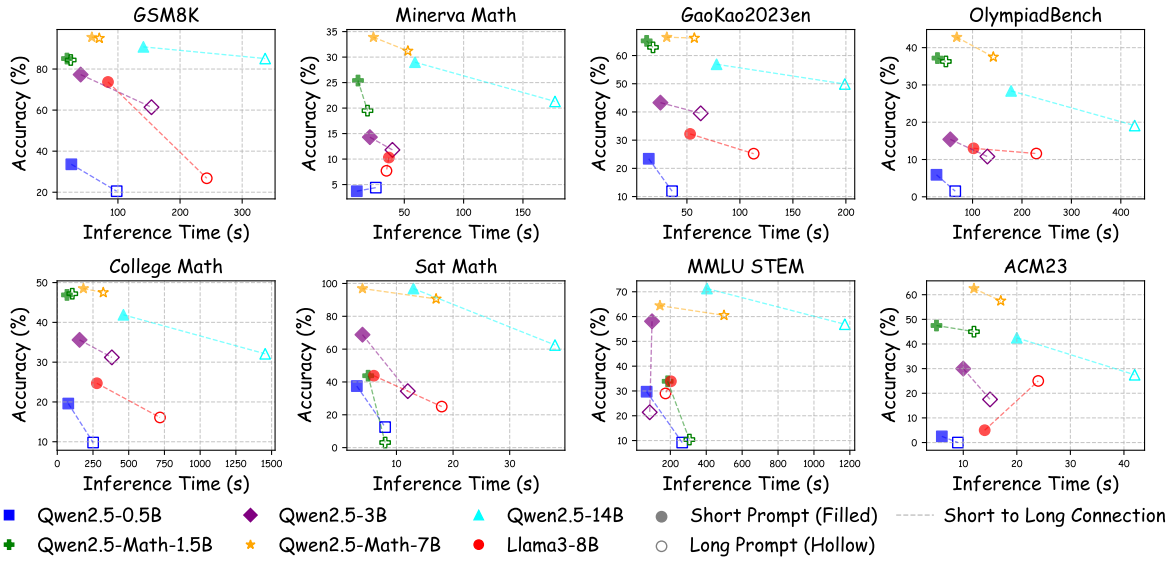

Figure 4: Inference time vs. accuracy across eight benchmarks. Models trained with short CoT (filled symbols) demonstrate significantly faster inference times compared to Long CoT (hollow symbols) while maintaining or exceeding accuracy levels.

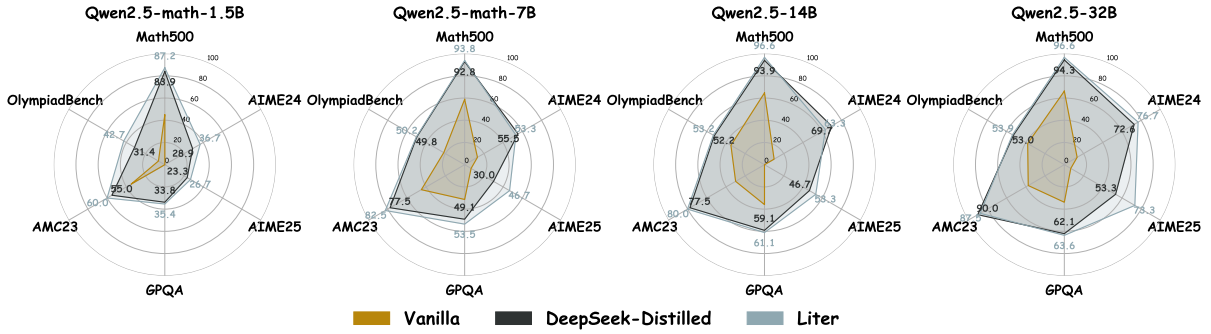

Figure 5: Radar chart show the performance comparison of the Vanilla, DeepSeek-R1-Distilled, and Liter versions of the Qwen2.5 model with different parameter sizes (1.5B, 7B, 14B, 32B) on multiple math and reasoning benchmarks (Math500, AIME24, OlympiadBench, AIME25, GPQA, AMC23).

significantly better than most of the same-level comparison models, and showed strong competitiveness on OlympiadBench (42.7%). Entering the 7B level, Ours-7B performs particularly well, achieving an accuracy of 93.8% on MATH500, and leading the other 7B parameter models listed in the table with 53.3% on AIME24 and GPQA respectively, while also being at the top level on OlympiadBench (50.2%). When the model scale is expanded to 32B, the performance advantage of Ours-32B is further consolidated, for example, it ranks among the top on multiple difficult benchmarks such as MATH500 (96.6%), AIME24 (76.7%), and GPQA (63.6%).

4.1.3 Exp-3: Comparison with other CoT Optimization methods.

We continue to compare models trained with DAD distilled data against those using other CoT decomposition techniques, focusing on overall performance and the speed-accuracy trade-off to analyze their key differences and advantages.

Overall Performance of Different CoT Decomposition Methods. Table 4 clearly shows the performance comparison of different CoT decomposition methods on multiple benchmarks, including the original baseline model (Vanilla), Chain-of-Draft (COD), two LLMLingua-2 with different compression rates (Lingua-2(0.9) and Lingua-2(0.5)), BudgetAware(2048), and our DAD method (Ours). From the accuracy data of each benchmark, our DAD method (Ours) shows the strongest overall performance. For example, on the challenging AIME24 benchmark, Ours achieved an accuracy of 23.3%, significantly higher than the other methods. On the important SAT Math benchmark, Ours also leads all other comparison methods with an accuracy of 96.9%, including COD (50.0%), LLMLingua-2(0.9) (93.8%), and BudgetAware(2048) (78.1%). It is worth noting that

Table 3: Performance Comparison of Language Models on Reasoning Benchmarks. Since there are too many models to be tested, we only select some of them for retesting, and we directly copy the results of all other models from their papers or blogs. † represents the copied results, ‡ represents the results retested by ourself. DS-R1-Distill represents the official Distilled Models in the DeepSeek-R1 report. OB represents the Olympiad Bench. **Bold** means the first performance and underline means the second performance.

Model	AIME24	MATH500	AMC23	OlympiaBench	OVERALL
Frontier Models					
DeepSeek-V3†	39.2	90.2	**85.0**	55.4	67.5
Mathstral-7B-V0.1†	0.0	57.8	37.5	21.5	29.2
DS-Coder-V2-Instruct†	13.3	75.3	57.5	37.6	45.9
Qwen2.5-Math-72B-Instruct†	30.0	85.6	70.0	49.0	58.6
NuminaMath-72B-CoT†	3.3	64.0	70.0	32.6	42.5
LLaMA3.1-405B-Instruct†	-	73.8	-	34.8	-
rStar-Math†	43.3	88.2	80.0	**63.1**	68.7
1.5B					
Qwen2.5-Math-1.5B-instruct†	10.0	60.0	60.0	38.1	42.0
ReasonFlux-1.5B†	20.0	70.4	**72.5**	**49.0**	53.0
DS-R1-Distilled-1.5B‡	28.9	83.9	55.0	31.4	49.8
Liter-1.5B‡	**36.7**	**87.2**	60.0	42.7	**56.7**
7B					
Qwen2.5-Math-7B-instruct†	13.3	82.6	62.6	39.0	49.4
Openthinker2-7B†	50.0	88.4	**89.5**	-	-
Bespoke-Stratos-7B†	20.0	82.0	-	-	-
ReasonFlux-7B†	36.7	88.6	80.0	**54.8**	65.0
DS-R1-Distilled-7B‡	**55.5**	92.8	77.5	49.8	68.9
Liter-7B‡	53.3	**93.8**	82.5	50.2	**70.0**
32B					
Qwen2.5-32B-instruct†	13.3	79.4	37.5	-	-
Openthinker2-32B‡	63.3	96.0	80.0	56.3	73.9
Light-R1-32B†	70.0	94.2	77.5	-	-
LIMO-32B‡	40.0	95.0	52.5	43.9	57.9
S1.1-32B‡	43.3	89.2	75.0	**56.6**	66.0
ReasonFlux-32B‡	73.3	96.4	70.0	34.7	68.6
Bespoke-Stratos-32B†	63.3	93.0	-	-	-
Sky-T1-32B-preview†	43.3	82.4	-	-	-
QwQ-32B-Preview‡	76.7	92.2	72.5	54.1	73.9
Qwen3-32B ‡	81.4	91.6	80.0	53.9	76.7
DS-R1-Distilled-32B‡	72.6	95.2	**90.0**	53.0	77.7
Liter-32B‡	**76.7**	**96.6**	87.5	53.9	**78.7**

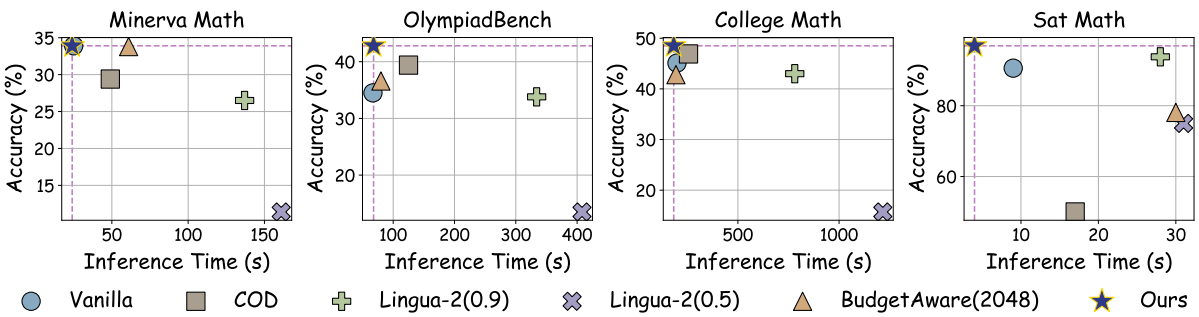

Figure 6: Scatter plots comparing inference time and accuracy of different methods across multiple benchmarks. Our method (Ours) consistently achieves optimal accuracy while significantly reducing inference time compared to existing methods.

Table 4: Performance comparison of different CoT decompose methods. COD represents Chain-of-Draft, Lingua-2 means LLMLingua-2.

Benchmark	Vanilla	COD	Lingua-2(0.9)	Lingua-2(0.5)	BudgetAware(2048)	**Ours**
AIME24	13.3	13.3	16.7	0.0	13.3	**23.3**
AIME25	10.0	13.3	23.3	0.0	3.3	**16.7**
GSM8K	94.8	94.4	94.8	72.4	94.4	**95.5**
Minerva Math	33.9	29.4	26.5	11.4	33.8	**33.9**
GaoKao2023en	64.2	62.3	64.4	34.5	65.2	**66.5**
OlympiadBench	34.5	39.4	33.8	13.5	36.6	**42.8**
College Math	45.1	46.9	43.0	15.7	42.8	**48.5**
Sat Math	90.6	50.0	93.8	75.0	78.1	**96.9**
ACM23	60.0	45.0	45.0	27.5	52.5	**62.5**

although some methods such as LLMLingua-2 may show good single-point performance on specific benchmarks (such as Lingua-2(0.9) 32.1% on GPQA), ours method can maintain or be close to the leading level on most benchmarks.

The relationship between inference speed and accuracy. Figure 6 fully demonstrates the superiority of the DAD distillation method (labeled as "Ours", purple star with gold rim), with additional results shown in Appendix **??**. In the tests across eight different benchmarks, the "Ours" method has the best trade-off in almost all speed-accuracy two-dimensional planes. Specifically, compared with other mainstream CoT decomposition techniques such as Chain-of-Draft, LLMLingua-2, and BudgetAware, our method not only achieves the highest accuracy in most cases (for example, 62.5% on ACM23 and 96.9% on Sat Math), but also achieves the shortest or very competitive inference times. In comparison, other methods either compromise on accuracy (such as LLMLingua-2 (0.5) or COD with high compression ratios) or far exceed "Ours" in reasoning time (such as LLMLingua-2 (0.9) with low compression ratios on some tasks). This better "price/performance ratio" consistently achieved on multiple benchmarks strongly demonstrates the unique advantages of the DAD distillation method.

5 Conclusion

In this paper, we introduced the Difficulty-Aware Distillation method, a new pipeline for CoT distillation. Leveraging DAP, we created LiteCoT, a difficulty-adaptive CoT dataset consisting of 100K distilled reasoning examples with an average length of only 720 tokens per sample, far shorter than existing alternatives. Utilizing LiteCoT, we distilled and trained a new family of reasoning models named Liter (1.5B, 7B, and 32B), based on the Qwen2.5 architecture. Extensive experiments across diverse reasoning benchmarks clearly demonstrate the effectiveness of the LiteCoT dataset and Liter models, consistently outperforming previous approaches that rely on substantially larger datasets with uniformly verbose reasoning chains.

6 Limitations

While our approach demonstrates promising results in generating concise mathematical reasoning, we acknowledge several limitations that offer avenues for future research.

Scope Limited to Text Modality First, our method's effectiveness has been validated on text-based tasks. Its applicability to multimodal reasoning, which involves diagrams or charts, remains unexplored. This is primarily because generating high-quality, concise CoT data for multimodal tasks is significantly more challenging, as it requires grounding logical steps in visual information.

Suboptimal Performance in Code Generation Second, our method is less effective for code generation. Unlike mathematical reasoning, where logical steps can often be condensed, code generation requires strict, step-by-step precision. Pruning even a minor detail in a code's logic can result in syntax errors or a non-functional program, making this approach less suitable for such tasks.

Dependency on the Teacher Model Finally, our approach is heavily dependent on the capabilities of the "teacher" model. The quality of the distilled dataset is fundamentally capped by the teacher's ability to accurately assess difficulty and generate logical, concise reasoning.

References

AI-MO. 2024. Aime 2024.

AI-MO. 2025. Aime 2025.

Simon A. Aytes, Jinheon Baek, and Sung Ju Hwang. 2025. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*.

Christopher A Choquette-Choo, Krishnamurthy Dvijotham, Krishna Pillutla, Arun Ganesh, Thomas Steinke, and Abhradeep Thakurta. 2023. Correlated noise provably beats independent noise for differentially private learning. *arXiv preprint arXiv:2310.06771*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

LLM Concise. Walk before you run! concise llm reasoning via reinforcement learning.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

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

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2024. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*.

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. *arXiv preprint arXiv:2103.03874*.

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

Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. 2025. Trimr: Verifier-based training-free thinking compression for efficient test-time scaling. *arXiv preprint arXiv:2505.17155*.

Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025a. Adar1: From long-cot to hybrid-cot via bi-level adaptive reasoning optimization. *Preprint*, arXiv:2504.21659.

Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and Bo Zheng. 2025b. Deconstructing long chain-of-thought: A structured reasoning optimization framework for long cot distillation. *arXiv preprint arXiv:2503.16385*.

math ai. 2025. Minervamath.

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

OpenAI. 2025. Introducing openai o1.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, and 1 others. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*.

Xiaoye Qu, Yafu Li, Zhaochen Su, Weigao Sun, Jianhao Yan, Dongrui Liu, Ganqu Cui, Daizong Liu, Shuxian Liang, Junxian He, and 1 others. 2025. A survey of efficient reasoning for large reasoning models: Language, multimodality, and beyond. *arXiv preprint arXiv:2503.21614*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

NovaSky Team. 2025a. Sky-t1: Train your own o1 preview model within $450. https://novasky-ai.github.io/posts/sky-t1. Accessed: 2025-01-09.

Open Thoughts Team. 2025b. Open Thoughts.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *arXiv preprint arXiv:2502.12018*.

TIGER-Lab. 2024. Mmlu-stem.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances*

in neural information processing systems, 35:24824–24837.

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

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. 2025. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Huifeng Yin, Yu Zhao, Minghao Wu, Xuanfan Ni, Bo Zeng, Hao Wang, Tianqi Shi, Liangying Shao, Chenyang Lyu, Longyue Wang, and 1 others. 2025. Towards widening the distillation bottleneck for reasoning models. *arXiv e-prints*, pages arXiv–2503.

Bin Yu, Hang Yuan, Haotian Li, Xueyin Xu, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025a. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *arXiv preprint arXiv:2505.03469*.

Bin Yu, Hang Yuan, Yuliang Wei, Bailing Wang, Weizhen Qi, and Kai Chen. 2025b. Long-short chain-of-thought mixture supervised fine-tuning eliciting efficient reasoning in large language models. *arXiv preprint arXiv:2505.03469*.

Qianjin Yu, Keyu Wu, Zihan Chen, Chushu Zhang, Manlin Mei, Lingjun Huang, Fang Tan, Yongsheng Du, Kunlin Liu, and Yurui Zhu. 2025c. Rethinking the generation of high-quality cot data from the perspective of llm-adaptive question difficulty grading. *arXiv preprint arXiv:2504.11919*.

Jintian Zhang, Yuqi Zhu, Mengshu Sun, Yujie Luo, Shuofei Qiao, Lun Du, Da Zheng, Huajun Chen, and Ningyu Zhang. 2025. Lightthinker: Thinking step-by-step compression. *CoRR*, abs/2502.15589.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2023. Evaluating the performance of large language models on gaokao benchmark. *arXiv preprint arXiv:2305.12474*.

Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 2025a. 1.4 million open-source distilled reasoning dataset to empower large language model training. *arXiv preprint arXiv:2503.19633*.

Shangziqi Zhao, Jiahao Yuan, Guisong Yang, and Usman Naseem. 2025b. Can pruning improve reasoning? revisiting long-cot compression with capability in mind for better reasoning. *arXiv preprint arXiv:2505.14582*.

A Complete Experiments

A.1 Exp-4: Generalization Analysis across Teacher Models

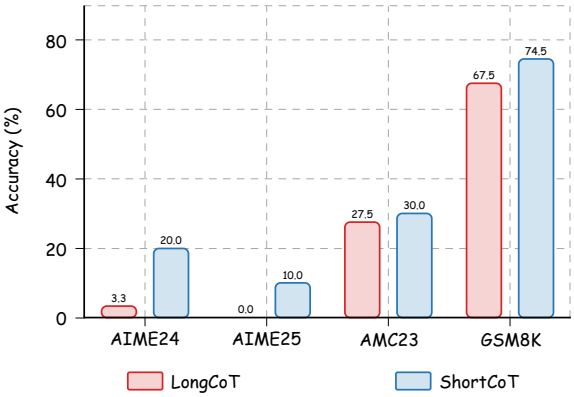

Figure 7: Performance comparison of LongCoT and ShortCoT distilled from the Qwen3-235B teacher model.

To verify that the effectiveness of our DAD method is not limited to a specific teacher model, we conducted a generalization experiment. In this experiment, we selected the most recent powerful Qwen3-235B-A22B-Thinking-2507 as the new teacher model. Using the same data sources as in **Exp-1,** we distilled ShortCoT by DAD to fine-tune the Qwen2.5-3B-Instruct base model. As illustrated in the figure 7, the results clearly indicate that the superiority of ShortCoT persists and remains significant even after changing the teacher model. Across all four mathematical benchmarks, the model trained on ShortCoT data achieved substantially higher accuracy than the one trained on LongCoT. This is especially pronounced on the challenging AIME24 task, where ShortCoT reached 20.0% accuracy compared to just 3.3% for LongCoT. Similarly, on the widely-used GSM8K dataset, ShortCoT demonstrated a commanding lead with 74% accuracy over LongCoT's 67.5%. This outcome provides strong evidence for two conclusions:

(1) Our DAD method possesses excellent generalization capabilities, as its effectiveness is not dependent on a specific teacher model.

(2) Regardless of the teacher model's strength, a verbose reasoning process itself is a suboptimal distillation strategy. Our proposed method of generating concise, key-step-focused reasoning chains

more efficiently transfers reasoning abilities to the student model.

A.2 Exp-5: Inference Efficiency Analysis: Token Savings Evaluation

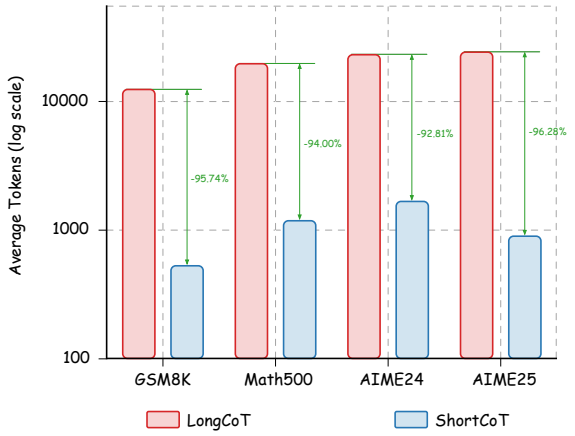

Figure 8: Comparison of Average Inference Tokens for LongCoT vs. ShortCoT across different benchmarks.

To quantify the improvements in inference efficiency offered by our DAP method, we analyzed the output token counts of the Qwen2.5-1.5B-Instruct model trained in **Exp-1**. This experiment was designed to measure the computational cost of the model during problem-solving. As illustrated in the figure above, the model trained on ShortCoT data, distilled via the DAD method, dramatically compresses inference costs.

Specifically, across all tested datasets, the average token output of the ShortCoT model was reduced by over 92% compared to the LongCoT model. For instance, on the GSM8K dataset, the token count plummeted from 12373 to 527, a saving of 95.74%. On AIME25, the token reduction was even more astounding, reaching 96.28%. This remarkable efficiency gain, combined with the findings from previous experiments where the ShortCoT model maintained or even surpassed the accuracy of the LongCoT model, proves the dual advantages of our approach. It not only guides the model to learn more precise and essential reasoning pathways but also drastically reduces the costs associated with model deployment and operation.

B Difficulty Aware Prompt

This section provides the complete "Difficulty Aware Prompt" used in the study. These prompts are designed to guide the model in assessing the difficulty of a problem and generating reasoning procedures of appropriate length accordingly.

Difficulty Aware Prompt

```
As an assistant, you need to first assess the complexity of the problem
and adopt an appropriate thinking framework before providing the final
solution. Structure your response into two main sections: Thought and
Solution.

First evaluate the complexity of the problem, then choose a suitable
thinking framework, and describe the thought process as detailed as
possible:

1. For simple problems:

**Analysis:**
[Understand the core elements and goals of the problem]

**Approach:**
[Propose direct solution methods]

**Summary:**
[Concisely summarize the solution approach and key points]


2. For moderately complex problems:

**Analysis:**
[Understand the problem and identify key points and challenges]

**Initial Approach:**
[Propose preliminary solutions]

**Reflection:**
[Evaluate the pros and cons of the initial approach]

**Improvement:**
[Refine the solution based on reflection]

**Summary:**
[Summarize the key points of the final solution]

3. For highly complex problems:

**Analysis:**
[Analyze various aspects of the problem and break down its structure]

**Problem Decomposition:**
[Break complex problem into manageable sub-problems]

**Sub-problem Processing:** (Repeat the following steps for each
sub-problem)
- Sub-problem 1:
  * Initial approach
  * Reflection
  * Improved solution
- Sub-problem 2:
  * Initial approach
  * Reflection
  * Improved solution
- ...(adjust according to the actual number of sub-problems)

**Integration:**
[Integrate sub-problem solutions into a complete solution]

**Overall Reflection:**
[Evaluate the integrated complete solution]

**Final Optimization:**
[Make final optimizations based on overall reflection]

**Summary:**
[Summarize key points of the final comprehensive solution]
```

```
The solution section should maintain logical,
↪  accurate, and concise
expression, detailing the steps needed to reach the
↪  conclusion,
formatted as:
**Solution:**
[Provide the final solution here]
```

C Extra Experiment Details

C.1 Training Details

Our training environment is equipped with 8 A800 GPUs with 80GB memory, and uses the DeepSpeed ZeRO-3 framework for distributed training. To further optimize GPU memory, we also introduced technologies such as Tensor Parallelism and liger kernel. Here, we share the YAML and DeepSpeed configuration files used for qwen2.5-Math-7B model training as an example.

Yaml config for running Qwen2.5-Math-7B on LiteCoT

```yaml
# Logging and Output arguments
log_level: info
logging_strategy: steps
logging_steps: 10
save_total_limit: 1
report_to:
# - tensorboard
- wandb
save_strategy: steps
save_steps: 100
output_dir: /output_path
overwrite_output_dir: true

# Model arguments
model_name_or_path: /model_path
model_revision: main
trust_remote_code: True
torch_dtype: bfloat16
attn_implementation: sdpa

# Data training arguments
dataset_name: /dataset_path
dataset_format: parquet
# dataset_configs:
# - all
max_seq_length: 8192
dataset_num_proc: 8
packing: true

# SFT trainer config
preprocessing_num_workers: 8
seed: 233
do_train: true
num_train_epochs: 3
per_device_train_batch_size: 4
do_eval: true
eval_strategy: steps
eval_steps: 10
per_device_eval_batch_size: 1
optim: adamw_torch
learning_rate: 5.0e-5
lr_scheduler_type: cosine_with_min_lr
lr_scheduler_kwargs:
  min_lr_rate: 0.1
warmup_ratio: 0.1
weight_decay: 0.0001
gradient_accumulation_steps: 4
gradient_checkpointing: true
gradient_checkpointing_kwargs:
  use_reentrant: false
max_grad_norm: 1.0
bf16: true
use_liger: true
```

DeepSpeed config for running Qwen2.5-Math-7B on LiteCoT

```yaml
compute_environment: LOCAL_MACHINE
debug: false
deepspeed_config:
  deepspeed_multinode_launcher: standard
  offload_optimizer_device: none
  offload_param_device: none
  zero3_init_flag: true
  zero3_save_16bit_model: true
  zero_stage: 3
  tensor_parallel:
    tp_size: 8
  offload_optimizer:
    device: none
    pin_memory: true
  offload_param:
    device: none
  train_batch_size: 16
  train_micro_batch_size_per_gpu: 4
  gradient_accumulation_steps: 4
  bf16:
    enabled: true
  optimizer:
    type: AdamW
    params:
      lr: 5.0e-5
      betas: [0.9, 0.999]
      eps: 1e-8
      weight_decay: 0.0001
      bias_correction: true
main_process_port: 29500
distributed_type: DEEPSPEED
downcast_bf16: 'no'
machine_rank: 0
main_training_function: main
mixed_precision: bf16
num_machines: 1
num_processes: 8
rdzv_backend: static
same_network: true
tpu_env: []
tpu_use_cluster: false
tpu_use_sudo: false
use_cpu: false
```

C.2 Evaluation Details

In our evaluation process, the evaluation scripts mainly come from two sources. For the four benchmarks AIME24, AIME25, GPQA, and MATH500, we use Hugging Face's LightEval library https://github.com/huggingface/lighteval/tree/main for model evaluation. For the remaining benchmarks focusing on mathematical reasoning ability, including GSM8K, Minerva Math, GaoKao2023en, OlympiadBench, College Math, Sat Math, MMLU STEM, and ACM23, we use the evaluation code provided in the Qwen2.5 code repository https://github.com/QwenLM/Qwen2.5-Math/tree/main/evaluation/data.

All evaluation experiments are performed uniformly under our specific GPU environment,

NVIDIA H20 with 96GB memories.

Next, we will share the shell (.sh) script files used to evaluate the AIME24, GPQA, MATH500, and AIME25 benchmarks based on Lighteval for reference.

Shell script for evaluating AIME24, AIME25, GPQA and MATH500

```
MODEL=/model_path
NUM_GPUS=n
export CUDA_VISIBLE_DEVICES="0,...,n-1"
MODEL_ARGS="pretrained=$MODEL,dtype=bfloat16,tensor_parallel_size=$NUM_GPUS,\
max_model_length=32768,gpu_memory_utilization=0.8,\
generation_parameters={max_new_tokens:32768,temperature:0.6,top_p:0.95}"
OUTPUT_DIR=data/evals/$MODEL

# AIME 2024
TASK=aime24
lighteval vllm $MODEL_ARGS "lighteval|$TASK|0|0" \
    --use-chat-template \
    --output-dir $OUTPUT_DIR
# AIME 2025
TASK=aime25
lighteval vllm $MODEL_ARGS "lighteval|$TASK|0|0" \
    --use-chat-template \
    --output-dir $OUTPUT_DIR
# MATH-500
TASK=math_500
lighteval vllm $MODEL_ARGS "lighteval|$TASK|0|0" \
    --use-chat-template \
    --output-dir $OUTPUT_DIR
# GPQA Diamond
TASK=gpqa:diamond
lighteval vllm $MODEL_ARGS "lighteval|$TASK|0|0" \
    --use-chat-template \
    --output-dir $OUTPUT_DIR
```

C.3 Benchmark Details

In our evaluation part, we totally evaluate our models on 11 benchmarks. They are:

- MATH500: Tests secondary school mathematics with open-ended problem-solving questions requiring step-by-step solutions, typically comprising 500 problems.
- AIME24: Assesses advanced high school mathematics (American Invitational Mathematics Examination level) through integer-answer competition-style problems, with a specific recent contest (e.g., AIME 2024) having 30 questions.
- GPQA: Evaluates graduate-level physics quantitative reasoning with challenging, multi-step, free-response questions written by physics experts, containing 448 questions.
- GSM8K: Focuses on grade school mathematics (grades 3-8) with multi-step arithmetic word problems requiring basic operations, featuring approximately 1,319 test questions.
- Minerva Math: Covers a broad range of quantitative reasoning from elementary mathematics to advanced levels using free-response math problems, often drawing from existing datasets and typically used with around 2,000 questions from its test set (MATH subset).
- GaoKao2023: Tests Chinese National College Entrance Examination (Gaokao) mathematics questions from 2023, featuring complex problems covering a wide high-school curriculum, with the math section typically having around 20-22 questions per exam paper version.
- OlympiadBench: Assesses mathematical Olympiad problem-solving abilities with highly challenging, proof-oriented or complex computational questions from various international and national Olympiads, containing 185 problems.
- College Math: Covers US college-level mathematics topics with multiple-choice and free-response questions spanning calculus, algebra, and statistics, with one common dataset having 1,760 questions.
- Sat Math: Evaluates high school mathematics relevant to college readiness (SAT exam) through multiple-choice and grid-in questions on algebra, problem-solving, and data analysis, with the math sections of an official SAT practice test typically containing 58 questions.
- MMLU STEM: A subset of the Massive Multitask Language Understanding (MMLU) benchmark, it tests expertise in STEM fields (Science, Technology, Engineering, and Mathematics) at high school and college levels via multiple-choice questions, with the STEM subset comprising several thousand questions across various specific STEM subjects (e.g., the "Mathematics" part alone has 250 dev and 1175 test questions).
- ACM23: Assesses competitive programming skills (similar to ACM International Collegiate Programming Contest problems) with algorithmic reasoning challenges requiring code solutions or detailed algorithmic steps, with specific contest sets varying but problem collections often containing hundreds of problems (the user might be referring to a specific year's problem set or a curated collection).

D Experiment Setup

We conduct extensive experiments to evaluate the effectiveness of LiteCoT data. Our evaluations address three key questions: (**Exp-1**) Does training on short difficulty-aware CoTs, LiteCoT, actually improve model performance compared to long CoTs? (**Exp-2**) Can our DAP method rival or surpass models distilled with an order of magnitude more data from a powerful teacher? (**Exp-3**) How does our DAP approach compare to other recent CoT distillation and compression techniques?

D.1 Experiment Setup

Training Data. Our experimental evaluation consists of three stages. All data comes from the OpenThoughts (Team, 2025b) dataset and is processed using our DAP distillation process. For **Exp-1**: Use $25K$ OpenThoughts samples. Their original Long CoT form is used as the baseline training data; at the same time, these samples are processed by the DAP process to generate $25K$ corresponding Short CoT data for comparison training. For **Exp-2**: Expand the amount of OpenThoughts data to $100K$. These samples are processed by the DAP process to generate $100K$ short CoT data(*i.e.,* LiteCoT), which constitute the training set of this experiment. For **Exp-3**: Directly use the $25K$ short CoT data generated in **Exp-1** as the training data for this stage.

Base Models. In **Exp-1** and **Exp-3**, due to the small amount of training data, the models we selected are all instruction versions in order to obtain better instruction following capabilities on small datasets. For **Exp-1**: Qwen2.5-0.5B-Instruct, Qwen2.5-Math-1.5B-Instruct, Qwen2.5-Math-7B-Instruct, Qwen2.5-14B-Instruct of the Qwen2.5 series, and Llama3.1-8B-instruct of the Llama3.1 series are included as base models. For **Exp-2**: Aiming to make a fair comparison with the results in the DeepSeek R1 report, we selected the same base version models used in the report: Qwen2.5-math-1.5B, Qwen2.5-math-7B, Qwen2.5-14B, and Qwen2.5-32B. For **Exp-3**: Similar to Exp-1, the model Qwen2.5-math-7B-Instruct is selected as the baseline model.

Benchmark Datasets. We evaluate model accuracy on a diverse suite of 11 benchmarks, covering math word problems (MATH500 (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021)), competition-level math (AIME24 (AI-MO, 2024)/AIME25 (AI-MO, 2025),OlympiadBench (Choquette-Choo et al., 2023), MinervaMath (math ai, 2025)), general QA (GPQA) (Rein et al.), and academic exams (GaoKao2023 (Zhang et al., 2023), MMLU-STEM (TIGER-Lab, 2024), etc.).

Baselines. For **Exp-1**, we mainly use the model directly distilled from the original Long CoT as the baseline to evaluate the relative improvement of our method. For **Exp-2**, we include a series of current mainstream reasoning models in the benchmark comparison range, including representative models such as S1 (Muennighoff et al., 2025) and OpenThoughts (Team, 2025b). For **Exp-3**, our benchmark comparison focuses on different Long CoT optimization or "decompose" methods. These methods include strategies that optimize at the prompt engineering level, such as Chain-of-Draft (Xu et al., 2025) and Budget Aware Reasoning (Han et al., 2024); in addition, we also include methods that require training additional models to simplify the output content, such as LLMLingua2 (Pan et al., 2024).

E Dataset Comparison

E.1 Cases of Samples

In this section, we will show the model distillation results for the same problem in different datasets. We will show the comparison of three different difficulty levels: easy, medium, and hard. Because the original answer for hard is too long, we only show the Litecot data and OpenThoughts data for easy and medium difficulty levels here.

Easy Question:

```
Return your final response within \boxed{}.
Find all natural numbers that,
when divided by 7, give the same quotient and remainder.
```

Solutions of easy problems in LiteCoT

```
<|begin_of_thought|>
**Analysis:**
The problem requires finding all natural numbers \( n
↪  \) such that
when divided by 7, the quotient \( q \) equals the
↪  remainder
\( r \). Using the division algorithm, \( n = 7q + r
↪  \),
with \( 0 \leq r < 7 \). The condition \( q = r \)
↪  must hold.

**Approach:**
Substitute \( q = r \) into the division equation to
↪  get \( n = 8r \).
Since \( r \) must be a valid remainder (1 \leq \( r
↪  \) \leq 6,
```

```
as \( r = 0 \) would make \( n = 0 \), not a natural
↪  number), compute
\( n \) for \( r = 1, 2, ..., 6 \).

**Summary:**
The natural numbers are \( 8r \) for \( r = 1 \) to \(
↪  6 \), yielding
\( 8, 16, 24, 32, 40, 48 \).
<|end_of_thought|>

<|begin_of_solution|>
**Solution:**
To find all natural numbers \( n \) such that when
↪  divided by 7,
the quotient equals the remainder:
1. Use the division formula: \( n = 7q + r \), where
↪  \( 0 \leq r < 7 \).
2. Set \( q = r \), leading to \( n = 7r + r = 8r \).
3. Since \( r \) must be a natural number (remainder
↪  \geq 1) and
\( r < 7 \), valid values for \( r \) are \( 1, 2, 3,
↪  4, 5, 6 \).
4. Calculate \( n \) for each \( r \):
   - \( r = 1 \Rightarrow n = 8 \)
   - \( r = 2 \Rightarrow n = 16 \)
   - \( r = 3 \Rightarrow n = 24 \)
   - \( r = 4 \Rightarrow n = 32 \)
   - \( r = 5 \Rightarrow n = 40 \)
   - \( r = 6 \Rightarrow n = 48 \)

Thus, the natural numbers are:
\[ \boxed{8, \ 16, \ 24, \ 32, \ 40, \ 48} \]
<|end_of_solution|>
```

Solutions of easy problems in OpenThoughts

```
<|begin_of_thought|>

Okay, so I need to find all natural numbers that,
↪  when divided by 7,
give the same quotient and remainder. Let me start by
↪  understanding
the problem properly.

First, let's recall that when we divide a natural
↪  number by 7,
we can express it in the form of the division
↪  algorithm: n = 7q + r,
where q is the quotient and r is the remainder. The
↪  remainder r must
satisfy 0 \len r < 7 because we're dividing by 7. So,
↪  the possible
values for r are 0, 1, 2, 3, 4, 5, 6.

Now, the problem states that the quotient and the
↪  remainder are the same.
That means q = r. So substituting that into the
↪  equation,
we get n = 7q + q = 8q. Therefore, n must be a
↪  multiple of 8. But wait,
we need to ensure that the remainder r is indeed
↪  equal to q and that it
satisfies the condition 0 \len r < 7.

Since r = q, and r must be less than 7,
q must also be less than 7. Because if q were 7 or
↪  more,
then the remainder r would be 7 or more, which isn't
↪  allowed in division
by 7. So q can be 0, 1, 2, 3, 4, 5, or 6.

However, n is a natural number. Depending on the
↪  definition,
natural numbers usually start from 1, so n must be at
↪  least 1. If q = 0,
then n = 8*0 = 0, which might not be considered a
↪  natural number.
Let me check if 0 is included here. If the problem
↪  defines natural
numbers as starting from 1, then q can only be from 1
↪  to 6. But if 0 is
included, then 0 would also be a solution.

But let me confirm the standard definition.
In mathematics, natural numbers typically start from
↪  1, so 0 is excluded.
Therefore, q must be at least 1. So q can be 1, 2, 3,
↪  4, 5, 6. Therefore,
the corresponding n values would be 8*1=8, 8*2=16,
↪  8*3=24, 8*4=32,
8*5=40, 8*6=48.

But wait, I need to verify if each of these numbers
↪  actually satisfies
the original condition. Let's take each n and divide
↪  by 7 to see if the
quotient and remainder are equal.

Starting with n=8: 8 divided by 7 is 1 with a
↪  remainder of 1.
Here, quotient q=1 and remainder r=1, so they are
↪  equal. That works.

Next, n=16: 16 divided by 7 is 2 with a remainder of
↪  2. So q=2 and r=2.
That's good.

n=24: 24 divided by 7 is 3 with a remainder of 3.
↪  Because 7*3=21,
24-21=3. So q=3, r=3. Correct.

n=32: 32 divided by 7 is 4 with remainder 4 (7*4=28,
↪  32-28=4).
So q=4, r=4.

n=40: 40 divided by 7 is 5 with remainder 5. 7*5=35,
↪  40-35=5.
So q=5, r=5.

n=48: 48 divided by 7 is 6 with remainder 6. 7*6=42,
↪  48-42=6.
So q=6, r=6.

Okay, all of these check out. Now, if we consider q=7,
↪  then n=8*7=56.
Let's divide 56 by 7: quotient is 8, remainder is 0.
↪  Here, q would have
been 8 (but according to our original substitution,
↪  q=r=7). Wait,
this is where the problem arises. If q=7, then r=7,
↪  but 7 is not a valid
remainder when dividing by 7. The remainder must be
↪  less than 7.
Therefore, q=7 is invalid because it would lead to a
↪  remainder of 7,
which isn't allowed. Hence, the maximum q can be is 6.

Therefore, the natural numbers are 8, 16, 24, 32, 40,
↪  48.

Wait a second, let me make sure there are no other
↪  numbers beyond these
that could satisfy the condition. Suppose someone
↪  claims that there is a
larger number where q=r. Let's say q=7, but as I saw,
↪  r would have to be
7, which is not allowed. Similarly, for q=8, r=8
↪  which is even more
invalid. So the only possible q values are 1 through
↪  6,
leading to n=8q where q=1 to 6.

But just to be thorough, let's check if there's a
↪  different approach.
Suppose we set up the equation:

When n is divided by 7, quotient q equals remainder r.
So, by division algorithm:

n = 7q + r

But since q = r, substitute:

n = 7q + q = 8q
```

Additionally, the remainder r must satisfy 0 \len r < 7. Since r = q,
this translates to 0 \len q < 7. However, since n is a natural number
(assuming starting at 1), q must be at least 1. Therefore, q
{1,2,3,4,5,6}, leading to n {8,16,24,32,40,48}.

Therefore, these are all the natural numbers that satisfy the condition.
Let me check if n=0 is allowed. If the problem includes 0 as a natural
number, which is sometimes the case in certain contexts, then 0 would be
a solution as well (0 divided by 7 is 0 with remainder 0). However,
since the problem specifies "natural numbers," and given that in most
mathematical contexts natural numbers start from 1, we can exclude 0.

Thus, the final answer should be the numbers 8, 16, 24, 32, 40, and 48.

<|end_of_thought|>

<|begin_of_solution|>

To find all natural numbers \(n \) that, when divided by 7, yield the
same quotient \(q \) and remainder \(r \), we start with the division
algorithm:
\[n = 7q + r \]
where \(0 \leq r < 7 \). The problem states \(q = r \),
so substituting \(r = q \) gives:
\[n = 7q + q = 8q. \]

Since \(r = q \) must satisfy \(0 \leq q < 7 \),
and \(n \) is a natural number (starting from 1),
\(q \) must be in \(\{1, 2, 3, 4, 5, 6\} \).
Substituting these values into \(n = 8q \):

- For \(q = 1 \): \(n = 8 \times 1 = 8 \).
- For \(q = 2 \): \(n = 8 \times 2 = 16 \).
- For \(q = 3 \): \(n = 8 \times 3 = 24 \).
- For \(q = 4 \): \(n = 8 \times 4 = 32 \).
- For \(q = 5 \): \(n = 8 \times 5 = 40 \).
- For \(q = 6 \): \(n = 8 \times 6 = 48 \).

Verification confirms each \(n \) satisfies \(\text{quotient} =
\text{remainder} \) when divided by 7. For example:
- \(8 \div 7 = 1 \) with remainder \(1 \),
- \(16 \div 7 = 2 \) with remainder \(2 \), etc.

Final Answer:
The natural numbers are \(\boxed{8, 16, 24, 32, 40, 48}\).

<|end_of_solution|>

Medium Question:

Return your final response within \boxed{}.
Let \(h_{1}, h_{2}, h_{3} \) be the heights of a triangle,
and \(r \) be the radius of the inscribed circle.
Prove that \(h_{1} + h_{2} + h_{3} \geq 9r \).

Solutions of medium problems in LiteCoT

<|begin_of_thought|>
Analysis:
The problem requires proving an inequality involving triangle heights
and inradius. Key elements are the relationships between heights (h_1,
h_2, h_3), inradius (r), area (S), and semiperimeter (p). The challenge
lies in connecting these quantities through known formulas and applying
an inequality.

Initial Approach:
Express heights in terms of area and sides (h_i = 2S/a_i). Use the
identity S = p \cdot r to substitute into the sum of heights. This leads to
h_1 + h_2 + h_3 = 2pr(1/a + 1/b + 1/c). To establish the inequality,
relate 1/a + 1/b + 1/c to semiperimeter via Cauchy-Schwarz.

Reflection:
The Cauchy-Schwarz inequality (for vectors (a, b, c) and (1/a, 1/b, 1/c))
provides (a + b + c)(1/a + 1/b + 1/c) \geq 9. Since a + b + c = 2p, this
bounds 1/a + 1/b + 1/c \geq 9/(2p). Substituting back confirms
the required inequality.

Improvement:
Verify the validity of Cauchy-Schwarz application and ensure all
substitutions align with triangle properties. No logical gaps detected.

Summary:
The proof hinges on expressing heights via area, applying Cauchy-Schwarz
to relate reciprocals of sides to semiperimeter, and substituting into
the sum of heights to derive the inequality.
<|end_of_thought|>

<|begin_of_solution|>
Solution:
1. **Express heights via area**: For a triangle with sides \(a, b, c\)
and corresponding heights \(h_1, h_2, h_3\), the area \(S\) is:
   \[
   h_1 = \frac{2S}{a},\quad h_2 = \frac{2S}{b},\quad h_3 = \frac{2S}{c}.
   \]

2. **Sum of heights**:
   \[
   h_1 + h_2 + h_3 = 2S\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right).
   \]

3. **Relate \(S\) to inradius \(r\)**:
Using \(S = p \cdot r\) (where \(p = \frac{a+b+c}{2}\) is semiperimeter):
   \[
   h_1 + h_2 + h_3 = 2pr\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right).
   \]

4. **Apply Cauchy-Schwarz inequality**:
   \[
   (a + b + c)\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq (1 + 1 + 1)^2 = 9.
   \]
   Since \(a + b + c = 2p\), this simplifies to:
   \[
   \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \geq \frac{9}{2p}.
   \]

5. **Substitute into height sum**:
   \[
   h_1 + h_2 + h_3 \geq 2pr \cdot \frac{9}{2p} = 9r.
   \]

```
Thus, the inequality \(h_1 + h_2 + h_3 \geq 9r\) is
↪  proven.

\[
\boxed{h_{1} + h_{2} + h_{3} \geq 9r}
\]
<|end_of_solution|>
```

Solutions of medium problems in OpenThoughts

```
<|begin_of_thought|>

Alright, so I need to prove that the sum of the
↪  heights of a triangle
is at least nine times the inradius. Let me start by
↪  recalling some
formulas related to triangles, heights, and the
↪  inradius.

First, the height (altitude) corresponding to a side
↪  of a triangle
is given by the formula \( h_a = \frac{2\Delta}{a} \),
↪  where
\( \Delta \) is the area of the triangle and \( a \)
↪  is the length
of
the side. Similarly, \( h_b = \frac{2\Delta}{b} \) and
\( h_c = \frac{2\Delta}{c} \). So, the sum of the
↪  heights
\( h_1 + h_2 + h_3 \) would be \( \frac{2\Delta}{a} +
\frac{2\Delta}{b}
+ \frac{2\Delta}{c} \).

On the other hand, the inradius \( r \) is related to
↪  the area and
the
semiperimeter \( s \) by the formula \( r =
↪  \frac{\Delta}{s} \),
where
\( s = \frac{a + b + c}{2} \). So, the inradius is the
↪  area divided
by
the semiperimeter.

Our goal is to show that \( h_1 + h_2 + h_3 \geq 9r
↪  \). Let's
substitute
the expressions for the heights and the inradius in
↪  terms of the
area
and the sides.

First, express \( h_1 + h_2 + h_3 \) as \( 2\Delta
↪  \left(
\frac{1}{a}
+ \frac{1}{b} + \frac{1}{c} \right) \). The inradius
↪  \( r \) is
\( \frac{\Delta}{s} \), so \( 9r = \frac{9\Delta}{s}
↪  \).
Therefore, the inequality we need to prove becomes:

\( 2\Delta \left( \frac{1}{a} + \frac{1}{b} +
↪  \frac{1}{c} \right)
\geq
\frac{9\Delta}{s} \)

Since \( \Delta \) is positive (it's the area of a
↪  triangle), we can
divide both sides by \( \Delta \) without changing
↪  the inequality
direction:

\( 2 \left( \frac{1}{a} + \frac{1}{b} + \frac{1}{c}
↪  \right) \geq
\frac{9}{s} \)

Simplifying further, multiply both sides by \( s \):

\( 2s \left( \frac{1}{a} + \frac{1}{b} + \frac{1}{c}
↪  \right) \geq 9
\)
```

```
So, the inequality reduces to proving that \( 2s
↪  \left( \frac{1}{a}
+ \frac{1}{b} + \frac{1}{c} \right) \geq 9 \), where
\( s = \frac{a + b + c}{2} \).

Let me write this more explicitly. Substitute \( s =
↪  \frac{a + b +
c}{2} \), so:

\( 2 \times \frac{a + b + c}{2} \times \left(
↪  \frac{1}{a} +
\frac{1}{b} + \frac{1}{c} \right) \geq 9 \)

The 2 and the denominator 2 cancel out, so:

\( (a + b + c) \left( \frac{1}{a} + \frac{1}{b} +
↪  \frac{1}{c}
\right)
\geq 9 \)

Ah, now this looks familiar. This is a classic
↪  inequality that might
be
related to the AM-HM inequality or Cauchy-Schwarz.
Let me recall the Cauchy-Schwarz inequality in the
↪  form:

\( (x_1y_1 + x_2y_2 + x_3y_3)^2 \leq (x_1^2 + x_2^2 +
↪  x_3^2)(y_1^2 +
y_2^2 + y_3^2) \)

But perhaps another form is more useful here. The
↪  Cauchy-Schwarz
inequality for sequences \( (a, b, c) \) and \(
↪  \left( \frac{1}{a},
\frac{1}{b}, \frac{1}{c} \right) \) gives:

\( \left( a \times \frac{1}{a} + b \times \frac{1}{b}
↪  + c \times
\frac{1}{c} \right)^2 \leq (a^2 + b^2 + c^2) \left(
↪  \frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2}
↪  \right) \)

But this simplifies to \( (1 + 1 + 1)^2 \leq (a^2 +
↪  b^2 + c^2)
\left( \frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2}
↪  \right) \), which is \( 9 \leq \) something. Not
↪  sure if that's directly helpful here.

Alternatively, consider the AM-HM inequality for
↪  three positive real
numbers:

\( \frac{a + b + c}{3} \geq \frac{3}{\frac{1}{a} +
↪  \frac{1}{b} +
\frac{1}{c}} \)

Rearranging this gives:

\( (a + b + c)\left( \frac{1}{a} + \frac{1}{b} +
↪  \frac{1}{c} \right)
\geq 9 \)

Which is exactly the inequality we need! So,
↪  according to the AM-HM
inequality,
the product of the sum of the sides and the sum of
↪  their reciprocals
is indeed greater than or equal to 9. Therefore, the
↪  original inequality \( h_1
+ h_2 + h_3 \geq 9r \) holds true.

But wait, let me double-check the steps to make sure
↪  I didn't skip
anything crucial. Starting from the heights:

\( h_1 = \frac{2\Delta}{a} \), similarly for \( h_2,
↪  h_3 \). Then sum is \( 2\Delta\left(\frac{1}{a} +
↪  \frac{1}{b} + \frac{1}{c}\right) \). Then, \( r \)
↪  is \( \frac{\Delta}{s} \), so \( 9r =
↪  \frac{9\Delta}{s} \). Therefore, the inequality
↪  becomes:
```

\(2\Delta\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq \frac{9\Delta}{s} \)

Cancel \(\Delta \):

\(2\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq \frac{9}{s} \)

Multiply both sides by \(s \):

\(2s\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq 9 \)

But since \(s = (a + b + c)/2 \), substituting:

\(2 \times (a + b + c)/2 \times (\text{sum reciprocals}) = (a + b + c)(\text{sum reciprocals}) \geq 9 \)

Which is exactly the AM-HM inequality. So, yes, the steps are correct.

But hold on, AM-HM gives \((a + b + c)/3 \geq 3/(1/a + 1/b + 1/c) \), which rearranges to \((a + b + c)(1/a + 1/b + 1/c) \geq 9 \). So that's exactly the inequality we have here. Therefore, equality holds when \(a = b = c \), i.e., in an equilateral triangle. So, this inequality is tight for equilateral triangles.

Therefore, the original inequality \(h_1 + h_2 + h_3 \geq 9r \) is proven by applying the AM-HM inequality to the sides of the triangle.

Alternatively, another approach might involve expressing the heights in terms of the inradius and other elements, but since this path worked out, maybe that's sufficient. However, to be thorough, let me consider if there are other ways or possible pitfalls.

Wait, another thought: The formula for the inradius is \(r = \Delta/s \), so \(s = \Delta/r \). Then, since \(h_1 + h_2 + h_3 = 2\Delta(1/a + 1/b + 1/c) \), we can write the inequality as:

\(2\Delta(1/a + 1/b + 1/c) \geq 9r = 9\Delta/s \)

Cancelling \(\Delta \), we get \(2(1/a + 1/b + 1/c) \geq 9/s \), which is equivalent to \(2s(1/a + 1/b + 1/c) \geq 9 \). Then substituting \(s = (a + b + c)/2 \), we have:

\(2 \times (a + b + c)/2 \times (1/a + 1/b + 1/c) = (a + b + c)(1/a + 1/b + 1/c) \geq 9 \)

Which again is the same inequality. So, regardless of the path, we end up with the AM-HM inequality.

Alternatively, if I didn't recall AM-HM, maybe using Cauchy-Schwarz:

\(\big((\sqrt{a})^2 + (\sqrt{b})^2 + (\sqrt{c})^2 \big) \times \big((1/\sqrt{a})^2 + (1/\sqrt{b})^2 + (1/\sqrt{c})^2 \big) \geq (1 + 1 + 1)^2 = 9 \)

But that's the same as \((a + b + c)(1/a + 1/b + 1/c) \geq 9 \). So, Cauchy-Schwarz also gives the same result. Therefore, the inequality holds by Cauchy-Schwarz as well.

Therefore, regardless of the method used, the key inequality needed is \((a + b + c)(1/a + 1/b + 1/c) \geq 9 \), which is a standard result. Therefore, the original inequality is proven.

Another angle: Let's consider specific cases. For an equilateral triangle, let's check if equality holds. Suppose all sides are equal, say length \(a \). Then, the semiperimeter \(s = 3a/2 \). The inradius \(r = \Delta/s \). The area \(\Delta = (\sqrt{3}/4)a^2 \). So \(r = \big((\sqrt{3}/4)a^2\big)/(3a/2) = (\sqrt{3}/6)a \). The heights \(h_1, h_2, h_3 \) are all equal to \((\sqrt{3}/2)a \). So sum of heights is \(3 \times (\sqrt{3}/2)a = (3\sqrt{3}/2)a \). Then \(9r = 9 \times (\sqrt{3}/6)a = (3\sqrt{3}/2)a \). Therefore, sum of heights equals \(9r \) in this case, so equality holds.

For a non-equilateral triangle, let's take a simple case. Suppose \(a = 3, b = 4, c = 5 \) (right-angled triangle). Then, semiperimeter \(s = (3 + 4 + 5)/2 = 6 \). Area \(\Delta = (3 \times 4)/2 = 6 \). Inradius \(r = \Delta/s = 6/6 = 1 \). Heights: \(h_1 = 2\Delta/a = 12/3 = 4 \), \(h_2 = 12/4 = 3 \), \(h_3 = 12/5 = 2.4 \). Sum of heights: \(4 + 3 + 2.4 = 9.4 \). \(9r = 9 \times 1 = 9 \). So \(9.4 \geq 9 \).

Another example: \(a = 2, b = 2, c = 3 \). Then \(s = (2 + 2 + 3)/2 = 3.5 \). Area using Heron's formula: \(\sqrt{3.5(3.5 - 2)(3.5 - 2)(3.5 - 3)} = \sqrt{3.5 \times 1.5 \times 1.5 \times 0.5} = \sqrt{3.5 \times 1.5 \times 0.75} = \sqrt{ (3.5 \times 0.75) \times 1.5 } \approx 1.984 \). So \(r = 1.984 / 3.5 \approx 0.567 \). Heights: \(h_1 \approx 2 \times 1.984 / 2 \approx 1.984 \), \(h_2 \approx 1.984 \), \(h_3 \approx 2 \times 1.984 / 3 \approx 1.323 \). Sum \(\approx 1.984 + 1.984 + 1.323 \approx 5.291 \). \(9r \approx 5.103 \). So \(5.291 \geq 5.103 \).

Wait, but according to the calculation, the sum of heights is approximately \(5.291 \), which is greater than \(9r \approx 5.103 \). So, even in this case, the inequality holds.

But let me check a more extreme triangle. Suppose a triangle with sides approaching degeneracy. Let’s take \(a = 1, b = 1, c \) approaching 2 (but still a valid triangle). Let’s compute when \(c \) approaches 2, say \(c = 2 - \epsilon \), where \(\epsilon \) is very small. Then semiperimeter \(s = (1 + 1 + 2 - \epsilon)/2 = (4 - \epsilon)/2 = 2 - \epsilon/2 \). Area \(\Delta \) using Heron's formula: \(\sqrt{ s(s-a)(s-b)(s-c) } = \sqrt{ (2 - \epsilon/2)(1 - \epsilon/2)(1 - \epsilon/2)(\epsilon/2) } \). As \(\epsilon \to 0 \), this becomes \(\sqrt{2 \times 1 \times 1 \times 0} = 0 \). So \(\Delta \to 0 \), and \(r = \Delta/s \to 0 \). The heights: \(h_1 = 2\Delta/a = 2\Delta \), \(h_2 = 2\Delta \), and \(h_3 = 2\Delta/c \approx \Delta \). So sum of heights \(\approx 2\Delta + 2\Delta + \Delta = 5\Delta \). \(9r \approx 9(\Delta/s) \approx 9\Delta/(2 - \epsilon/2) \approx (9/2)\Delta \). Since \(5 > 4.5 \), \(5\Delta \geq (9/2)\Delta \) when \(\Delta > 0 \). As \(\Delta \to 0 \), both sides \(\to 0\), inequality still holds.

Alternatively, consider a very skewed triangle where one side is much longer than the others. But perhaps in all cases, the sum of the heights is still greater than \(9r \).

Therefore, it seems that through multiple examples and applying the AM-HM inequality, the original statement is valid.

Another approach could be using trigonometric identities. Let's denote the triangle with sides \(a, b, c \) opposite to angles \(A, B, C \) respectively. The heights can be expressed as \(h_a = b \sin C = c \sin B \), similarly for others. But this might complicate things. Alternatively, since \(r = (a + b - c)/2 \times \tan(C/2) \) etc., but perhaps this is not useful here.

Alternatively, express everything in terms of \(r \) and the semiperimeter. Since \(r = \Delta/s \), and \(\Delta = r s \). Then, the heights can be written as \(h_a = 2\Delta/a = 2rs/a \). Similarly, \(h_b = 2rs/b \), \(h_c = 2rs/c \). Therefore, sum of heights is \(2rs (1/a + 1/b + 1/c) \). We need to show that this is \(\geq 9r \). Divide both sides by \(r \) (positive), so need to show \(2s(1/a + 1/b + 1/c) \geq 9 \), which is the same as before. So again, we come back to the same inequality.

Therefore, regardless of the approach, the crux is proving \((a + b + c)(1/a + 1/b + 1/c) \geq 9 \), which is a direct consequence of the AM-HM inequality. Hence, the original inequality holds.

In conclusion, the key steps are:

1. Express the heights in terms of the area and sides.
2. Express the inradius in terms of the area and semiperimeter.
3. Formulate the inequality in terms of the sides and semiperimeter.
4. Apply the AM-HM inequality to establish the required result.
5. Verify with specific examples to ensure validity.

Therefore, the inequality \(h_1 + h_2 + h_3 \geq 9r \) is proven.

<|end_of_thought|>

<|begin_of_solution|>

To prove that the sum of the heights of a triangle \(h_1 + h_2 + h_3 \) is at least nine times the inradius \(r \), we proceed as follows:

1. **Express Heights and Inradius**:
   - Heights: \(h_a = \frac{2\Delta}{a}, \, h_b = \frac{2\Delta}{b}, \, h_c = \frac{2\Delta}{c} \), where \(\Delta \) is the area and \(a, b, c \) are the sides.
   - Sum of heights: \(h_1 + h_2 + h_3 = 2\Delta\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \).
   - Inradius: \(r = \frac{\Delta}{s} \), where \(s = \frac{a + b + c}{2} \) is the semiperimeter.

2. **Formulate the Inequality**:
   Substitute \(h_1 + h_2 + h_3 \) and \(9r \) into the inequality:
   \[
   2\Delta\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq \frac{9\Delta}{s}.
   \]
   Cancel \(\Delta \) and multiply through by \(s \):
   \[
   2s\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq 9.
   \]

3. **Simplify Using Semiperimeter**:
   Substitute \(s = \frac{a + b + c}{2} \):
   \[
   (a + b + c)\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq 9.
   \]

4. **Apply the AM-HM Inequality**:
   The Arithmetic Mean-Harmonic Mean (AM-HM) inequality states:
   \[
   \frac{a + b + c}{3} \geq \frac{3}{\frac{1}{a} + \frac{1}{b} + \frac{1}{c}},
   \]
   which rearranges to:
   \[
   (a + b + c)\left(\frac{1}{a} + \frac{1}{b} + \frac{1}{c}\right) \geq 9.
   \]
   Equality holds if and only if \(a = b = c \) (equilateral triangle).

5. **Conclusion**:
   The inequality \(h_1 + h_2 + h_3 \geq 9r \) follows directly from the AM-HM inequality applied to the sides of the triangle. Examples and edge cases confirm the validity of the result.

Final Answer:
\boxed{h_{1} + h_{2} + h_{3} \geq 9r}

<|end_of_solution|>