Pru-CoT: Towards Efficient Reasoning Distillation via Pruning Chain-of-Thought

Han Liu[1], Shuotian Ma[1], Hui Li[1], Xiaotong Zhang[1]*, Fenglong Ma[2], Hong Yu[1]
[1]Dalian University of Technology, Dalian, China,
[2]The Pennsylvania State University, Pennsylvania, USA
liu.han.dut@gmail.com, mashuotian@mail.dlut.edu.cn, lihui.200101@gmail.com,
zxt.dut@hotmail.com, fenglong@psu.edu, hongyu@dlut.edu.cn

Abstract

Knowledge distillation has emerged as a pivotal paradigm for transferring the superior reasoning capabilities of Large Reasoning Models (LRMs) to efficient student models. However, the raw Chain-of-Thought (CoT) trajectories are often verbose and redundant, which dilutes the underlying logic and hinders effective knowledge distillation for student models. Although recent work has focused on pruning CoT to streamline these reasoning paths, existing local heuristic methods often fail to capture global causal logic due to rigid rules and limited search spaces, while global heuristic approaches incur substantial computational costs. To address these issues, we propose Pru-CoT (Pruning Chain-of-Thought), a framework that aims to extract the essential logical structure from reasoning chains. Pru-CoT implements a step-level importance assessment via global optimization on a frozen student large language model (LLM), quantifying the gradient-based causal contribution of each component. Guided by these important signals, the framework performs fidelity-constrained pruning, utilizing an LLM-driven process to synthesize concise, logically coherent narratives. Extensive experiments on mathematical reasoning benchmarks demonstrate that models trained with Pru-CoT not only achieve superior accuracy but also generate significantly more compact reasoning paths compared to those trained on raw verbose data.

1 Introduction

Large Reasoning Models (LRMs), such as OpenAI's o1 (OpenAI, 2024) and DeepSeek-R1 (DeepSeek-AI, 2025a), have revolutionized complex problem-solving via Chain-of-Thought (CoT) (Wei et al., 2022). To cultivate these capabilities in cost-effective, small models, knowledge distillation has become the mainstream paradigm (DeepSeek-AI, 2025a). However, the raw inference chains

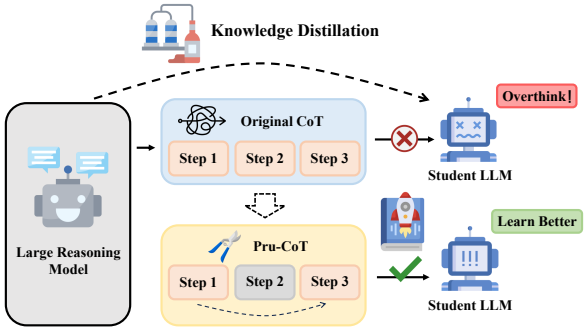


Figure 1: Pru-CoT addresses the "overthinking" issue in LRMs by pruning redundant steps to produce concise reasoning chains, thereby facilitating effective knowledge distillation to student models.

generated by teacher models often contain redundant steps such as repetitive verifications or trivial derivations, i.e., overthinking phenomena (Chen et al., 2025; Team, 2025a). The low signal-to-noise ratio reasoning data results in small models that tend to overfit the surface syntactic structure of "overthinking" rather than internalizing the core causal logic, making their reasoning computationally inefficient and susceptible to hallucinations.

To address this issue, recent studies explore pruning methods to streamline the redundant chain-of-thought generated by large reasoning models, aiming to mitigate the negative effects caused by overthinking on knowledge distillation. Specifically, local-based heuristic methods (Wang et al., 2025; Cui et al., 2025) conduct local spatial chain-of-thought pruning exploration through pre-defined simple rules, which are often constrained by simple rule biases and limited search spaces, failing to identify the global causal logic required for pruning. Conversely, global-based heuristic methods such as A*-Thought (Xu et al., 2025b) effectively model long-range dependencies via global tree search, but they are accompanied by substantial computational overhead.

To handle the aforementioned issues simultane-

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 33744–33756

ously, as shown in Figure 1, we propose **Pru-CoT** (**Pru**ning **C**hain-**o**f-**T**hought), a method based on global optimization and LLM-guided pruning. Diverging from heuristics-based methods that rely on rigid external rules, Pru-CoT performs step-level importance assessment via a learnable mechanism on a frozen student model. This mechanism assigns a learnable weight to each reasoning step, which is optimized via backpropagation derived from the final solution loss. Thereby, based on optimized weight, we accurately quantify the causal contribution of every step, which enables the precise pruning of "overthinking" noise while distinguishing essential logic. To address the potential semantic discontinuity caused by removal, we further execute LLM-guided pruning incorporating fidelity constraints. This phase utilizes key reasoning steps determined through global optimization as semantic anchors to synthesize a concise, logically coherent narrative, ensuring the distilled data remains highly effective for student training[1].

Our contributions are summarized as follows:

- We propose a novel method, termed Pru-CoT, to prune Chain-of-Thought, effectively removing structural redundancy to enhance both inference efficiency and data quality for student models.
- We develop a global optimization framework that quantifies step-wise causal contributions via learnable weights on a frozen model, identifying critical logic without relying on rigid heuristics.
- Extensive experiments demonstrate the effectiveness of Pru-CoT across several benchmarks, achieving significant improvements, with an average accuracy improvement of 2.4% on DeepSeek-R1-Distill-Qwen-7B.

2 Related Work

2.1 Generation-Guidance Compression

Reasoning chain compression approaches can be broadly categorized into generation-guidance methods and Chain-of-Thought pruning methods. Generation-guidance approaches aim to steer the model toward concise reasoning during the inference or training phase. Prompt-based methods employ carefully crafted prompts to guide the model directly toward a streamlined reasoning process (Ding et al., 2024; Xu et al., 2025a). Several studies adopt reinforcement learning frameworks (Hou et al., 2026; Luo et al., 2025; Singh et al., 2025) to reward concise outputs. Other lines of research introduce specialized control mechanisms. For instance, CoT-Valve (Ma et al., 2025) employs a parameter-space interpolation mechanism that steers LLMs to generate a chain-of-thought of controllable length and quality, while LightThinker (Zhang et al., 2025) trains models to dynamically compress ongoing thoughts. However, these approaches primarily act as remedies after distillation, failing to prevent students from being influenced by overthinking phenomena right from the beginning of the distillation process.

2.2 Chain-of-Thought Pruning

Another mainstream paradigm is Chain-of-Thought pruning, which refines high-quality traces from a teacher model, thereby avoiding the model from being affected by the overthinking phenomenon from the very beginning. These methods generally fall into local-based and global-based heuristic categories. ASAP (Zeng et al., 2025) uses "first-token surprisal" to retain logically critical steps. Tokenskip (Xia et al., 2025) prunes tokens according to importance scores from LLMLingua-2 (Pan et al., 2024) and constructs its training dataset based on data with different compression ratios. SPIRIT (Cui et al., 2025) iteratively prunes reasoning steps based on the changes in the target model's perplexity, and Binary Cutting (Wang et al., 2025) combines binary search pruning with policy verification to find reasoning prefixes that still yield correct answers. In contrast, A*-Thought (Xu et al., 2025b) introduces a global-based heuristic approach that determines reasoning paths with high information density and low cost via a global heuristic function. However, the heuristic-based methods are constrained by local biases and fail to capture global causal dependencies, while search-based strategies introduce prohibitive inference overhead, rendering them impractical for large-scale data processing.

3 The Proposed Method

3.1 Preliminary

Given a large reasoning model (LRM) $\mathcal{M}$ and an input question q, the model generates a reasoning chain comprising S sequential steps, denoted as

[1]The source code is publicly available at https://github.com/Co-Cl2/Pru-CoT.

$\mathcal{T} = (s_1, s_2, \ldots, s_S)$, which concludes with a final solution y. However, as the reasoning depth increases, LRMs are prone to the overthinking phenomenon, where the generated trajectories often contain redundant, repetitive, or irrelevant logical steps. This redundancy not only significantly escalates the inference computational cost but also introduces noise that can degrade the performance of student models when used as training targets.

Our objective is to identify a subset $\mathcal{T}' \subseteq \mathcal{T}$. This compact trajectory is expected to eliminate structural redundancy while guaranteeing the semantic integrity required to derive the correct solution y, thereby ensuring high knowledge density for efficient downstream applications.

3.2 Overview

We propose Pru-CoT (Pruning Chain-of-Thought), a framework designed to distill efficient reasoning paths by pruning overthinking from verbose CoT data. The overall architecture is illustrated in Figure 2. Premised on the insight that reasoning steps exhibit varying degrees of causal contribution, Pru-CoT implements a synergistic two-phase process: (1) Step-Level Importance Assessment via Global Optimization (Section 3.3), which rigorously quantifies the causal weight of each reasoning step through a differentiable probing mechanism on a frozen student LLM; and (2) LLM-Guided Pruning with Fidelity Constraints (Section 3.4), which synthesizes concise and coherent rationales by selectively retaining low-importance steps while ensuring the compressed chain remains faithful to the original logical validity.

3.3 Step-Level Importance Assessment via Global Optimization

To accurately assess the importance of each reasoning step within the Chain-of-Thought, we evaluate step necessity from a global optimization perspective, explicitly quantifying the causal contribution of individual steps to the final solution. Unlike heuristic methods that evaluate steps in isolation or based on local coherence, this approach captures the holistic dependency structure required for the correct answer.

Step-Wise Noise Soft Masking. Recent studies suggest that LLMs use filler tokens to sustain computation depth (Pfau et al., 2024). Motivated by this, we define a reasoning step as non-essential if the model maintains its predictive accuracy even when the step is substituted with non-informative noise. We formalize this by introducing a learnable masking mechanism. For the reasoning chain $\mathcal{T} = (s_1, \ldots, s_S)$, we assign a learnable importance vector $\mathbf{w} = (w_1, w_2, \ldots, w_S)$, where each scalar $w_t \in [0, 1]$ governs the semantic retention of step s_t. Let $\mathbf{e}_{t,k}$ denote the embedding of the k-th token within the t-th step, and $\mathbf{n}$ be a learnable noise embedding corresponding to a filler token (e.g., "."). The fused embedding $\tilde{\mathbf{e}}_{t,k}$ is computed using the shared step weight w_t:

$$\tilde{\mathbf{e}}_{t,k} = w_t \cdot \mathbf{e}_{t,k} + (1 - w_t) \cdot \mathbf{n}, \tag{1}$$

where w_t acts as a soft gate: as $w_t \to 1$, the original semantics are preserved; as $w_t \to 0$, the step is effectively replaced by noise.

Gradient-Based Optimization. In the probing stage, we freeze the parameters of the language model θ_{frozen} and optimize only the sample-specific importance vector $\mathbf{w}^{(n)} = (w_1, w_2, \ldots, w_S)$. For the n-th sample, the sequence of fused embeddings $\tilde{\mathbf{E}}^{(n)}$ (generated via Equation 1) is concatenated with the question embedding $\mathbf{E}_q^{(n)}$ to form the model input. We optimize $\mathbf{w}^{(n)}$ for a few epochs to minimize the cross-entropy loss on the solution sequence $y^{(n)}$:

$$\mathbf{w}^{(n)*} = \arg\min_{\mathbf{w}^{(n)}} \mathcal{L}(y^{(n)} \mid [\mathbf{E}_q^{(n)}; \tilde{\mathbf{E}}^{(n)}]; \theta_{\text{frozen}}). \tag{2}$$

This gradient-based update mechanism naturally captures the global dependencies between reasoning steps. When a critical step is masked, the resulting gradient perturbation propagates through the computational graph and influences the weight updates of other semantically or logically dependent steps, thereby reflecting the strength of inter-step reliance. In contrast, masking a redundant step does not induce significant gradient flow across steps, which helps to distinguish between structural criticality and isolated token-level salience.

We retain the weights achieving the lowest loss as the final importance weights, ensuring the assessment is tailored to the instance-specific reasoning structure. Mechanistically, masking critical steps disrupts the reasoning path and spikes the loss, generating strong gradients that push the corresponding weights toward 1 to restore semantics. Conversely, redundant or negative steps exhibit negligible impact on the prediction when replaced by filler tokens, resulting in weights that remain close to their initialization or converge to 0.

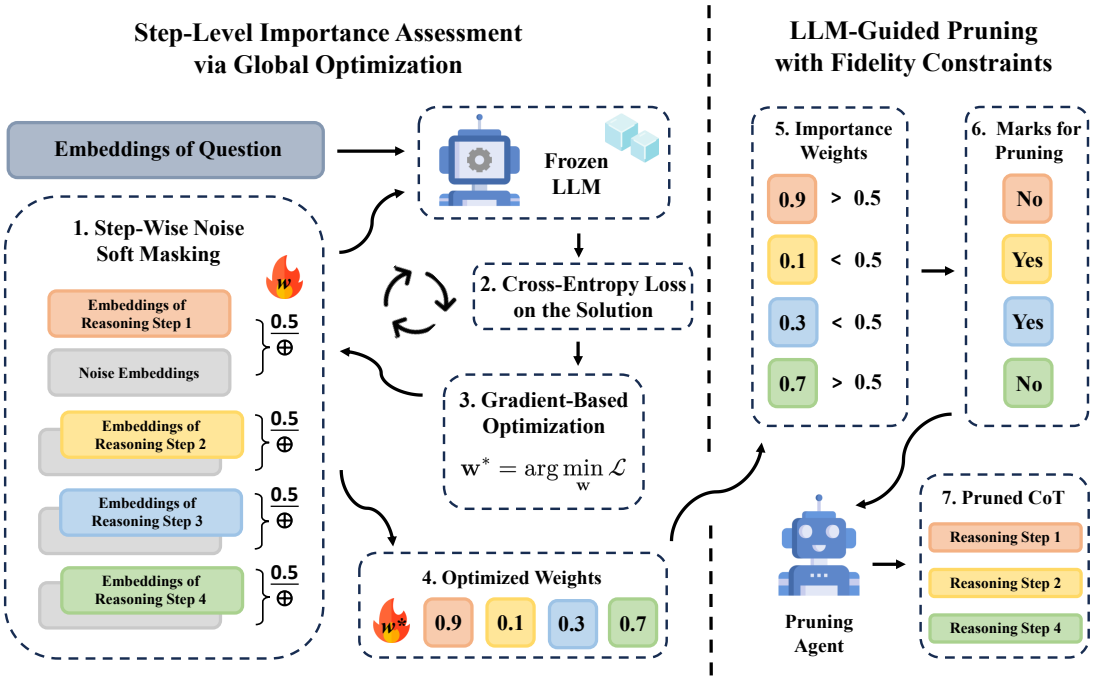


Figure 2: The framework of the Pru-CoT method. Step importance is quantified by optimizing learnable weights on a frozen LLM to minimize prediction loss. Guided by these weights, a Pruning Agent synthesizes concise rationales by anchoring on critical steps and selectively pruning low-importance candidates to ensure that the compressed chain remains logically faithful.

3.4 LLM-Guided Pruning with Fidelity Constraints

The sample-specific importance weights obtained through the global optimization establish a reliable foundation for pruning. From the perspective of reasoning validity, removing steps with low importance weights does not impair the model's ability to arrive at the correct solution from the condensed chain. However, overly aggressive pruning can disrupt the logical flow and linguistic coherence between steps, impeding the model's ability to learn from and reproduce such reasoning paths during training. Therefore, we introduce an LLM-guided pruning process that balances deductive efficiency with generative fluency.

Formally, we define the set of candidate steps for pruning $\mathcal{P}_{\text{cand}}$ based on the previously obtained importance weights:

$$\mathcal{P}_{\text{cand}} = \{s_t \mid w_t^* < \tau, \quad t = 1, 2, \ldots, S\}, \quad (3)$$

where τ is the importance threshold that determines whether an inference step can be pruned or not. Steps with importance weight w_t^* below the threshold τ are marked as candidates for pruning, while those with weights greater than or equal to τ are retained as essential and non-prunable.

Based on the candidate set, we employ an LLM-guided pruning process. Specifically, we provide the question, the original chain-of-thought, the candidate step annotations, and the final answer as input to a LLM-based pruning agent, which then outputs the final pruned set $\mathcal{P} \subseteq \mathcal{P}_{\text{cand}}$. Consequently, the final compressed reasoning chain is derived as $\mathcal{T}' = \mathcal{T} \setminus \mathcal{P}$.

During this phase, non-prunable critical reasoning steps serve as semantic anchors. The main reasoning flow constructed by these semantic anchors assists the pruning agent in better determining whether a prunable fragment contributes to the logical coherence of the core reasoning process, thereby enabling more precise and efficient pruning. Additionally, we carefully design the prompt for the pruning agent to mitigate hallucination during pruning tasks; the specific content of the prompt is provided in Appendix A.

4 Experiments

4.1 Experimental Setup

Models. To evaluate the effectiveness and scalability of our method for Chain-of-Thought pruning, we employ DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-1.5B (DeepSeek-AI, 2025a) as our student backbones.

Methods	AIME24/25		AMC23		MATH500		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
DeepSeek-R1-Distill-Qwen-7B									
SFT	30.8	6956	70.6	5035	80.2	3691	60.5	5227	1.16
BtC (Ding et al., 2024)	30.8	7100	71.3	5003	80.9	3591	61.0	5231	1.17
Tokenskip (Xia et al., 2025)	28.3	7084	73.1	4936	81.7	3314	61.0	5111	1.19
A*-Thought (Xu et al., 2025b)	32.1	**6867**	**75.0**	4764	81.7	**2972**	62.9	**4868**	1.29
Pru-CoT (Ours)	**38.3**	6870	**75.0**	**4674**	**82.6**	3070	**65.3**	4871	**1.34**
DeepSeek-R1-Distill-Qwen-1.5B									
SFT	**21.3**	7339	54.4	5262	70.6	3626	**48.8**	5409	0.90
BtC (Ding et al., 2024)	15.8	7412	48.8	5231	69.5	3429	44.7	5357	0.83
Tokenskip (Xia et al., 2025)	16.7	7411	53.8	5390	**74.3**	3794	48.3	5532	0.87
A*-Thought (Xu et al., 2025b)	18.8	**7050**	48.1	**3979**	69.2	**2843**	45.4	**4624**	**0.98**
Pru-CoT (Ours)	19.2	7220	**55.0**	4440	70.2	3023	48.1	4894	**0.98**

Table 1: Experimental results of different methods on several benchmarks with an 8K-token budget. The best result is in bold, and the second best is underlined. " **Avg.**" is the average across the three datasets, and **ACU** balances performance and efficiency.

Training Dataset. OpenThoughts-114k (Guha et al., 2025) is a dataset containing 114k complex problems and their reasoning chains, where the reasoning chains are derived from DeepSeek-R1. We randomly select 3,000 samples from OpenThoughts-114k and filter out samples with lengths longer than 8,192 tokens, resulting in a subset of 2,051 samples for training. We avoid additional filtering to preserve the original reasoning chains for evaluating the compression performance in a realistic way.

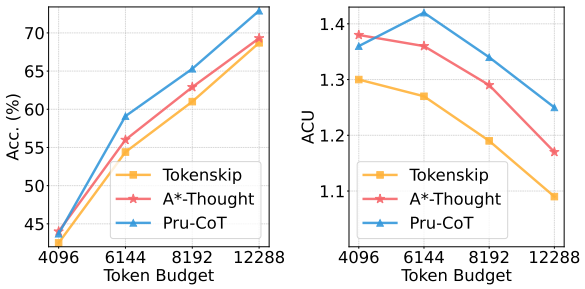
Evaluation Dataset. We benchmark our method on three representative mathematical reasoning benchmarks: AIME24/25 (Li et al., 2024; Zhang and Math-AI, 2025), AMC23 (Li et al., 2024), and MATH500 (Lightman et al., 2024).

Baselines. We compare **Pru-CoT** with the following baselines:

- **SFT**: The student model fine-tuned directly on the training dataset.
- **BtC** (Ding et al., 2024): A prompt-based method. It utilizes specialized prompting strategies to encourage the model to leverage shortcuts.
- **Tokenskip** (Xia et al., 2025): A token-level pruning method. It constructs its training dataset by compressing traces using importance scores from LLMLingua-2 (Pan et al., 2024), and is trained on data with mixed compression ratios.
- **A*-Thought** (Xu et al., 2025b): A search-based step-level framework that compresses reasoning paths via a heuristic function and bidirectional importance scoring.

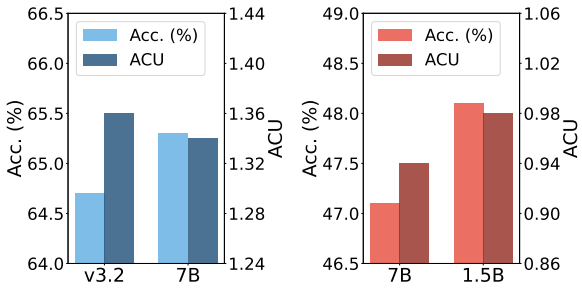
Implementation Details. In the step-level importance assessment phase, importance weight w_t is initialized to 0.5. We employ the token "." as the filler token. The importance weights are optimized for 3 epochs using the SGD optimizer with a learning rate of 10 to minimize the solution prediction loss (Equation 2). In the subsequent LLM-guided pruning phase, the threshold is set as $\tau = 0.5$, aligning with the initialization value. We employ Qwen2.5-7B-Instruct and Qwen2.5-1.5B-Instruct (Yang et al., 2024a,b) as pruning agents, ensuring the agent's capacity matches the scale of the corresponding student model as defined in Section 3.4. For additional experimental information, including hyperparameter settings for training are detailed in Appendix B.

Step Segmentation and Pruning Statistics. In our implementation, we use two consecutive newline characters ("\n\n") as the delimiter to split reasoning chains into discrete steps. This design choice is based on the observed generation patterns of the DeepSeek-R1 series, where the model naturally employs "\n\n" to divide relatively independent reasoning steps. On average, each training sample contains 76 steps. Using DeepSeek-R1-Distill-Qwen-7B and DeepSeek-R1-Distill-Qwen-1.5B as student models, Pru-CoT achieves step pruning ratios of 23.8% and 25.8%, respectively.



(a) Acc. (%) (b) ACU

Figure 3: Left: Accuracy (%) vs. maximum token budget. Right: ACU vs. maximum token budget. Pru-CoT consistently achieves higher accuracy and better performance-efficiency trade-off across all budgets.



(a) (b)

Figure 4: Comparison of student performance when pruned by same-scale vs. larger-scale models. Left: 7B student with DeepSeek-V3.2 vs. Qwen2.5-7B-Instruct pruning agent. Right: 1.5B student with Qwen2.5-7B-Instruct vs. Qwen2.5-1.5B-Instruct pruning agent.

Evaluation Metrics. We adopt three metrics for evaluation: Acc. (pass@1 accuracy), Tokens (total length of the generated reasoning chain and answer), and ACU (Ma et al., 2025) (Acc./Tokens), which assesses the performance-efficiency trade-off. To ensure statistical robustness, we conduct evaluations in a zero-shot setting by performing 4 independent inference runs for each instance (temperature $t = 0.6$, top-$p = 0.95$). The reported results for both Acc. and Tokens are the averages across these trials.

4.2 Main Results

Table 1 presents the main results on several benchmarks. Our method, Pru-CoT, demonstrates a clear and consistent advantage. Across both model sizes and most datasets, it achieves higher reasoning accuracy while simultaneously generating shorter reasoning chains, which collectively lead to the highest ACU scores. This validates our core hypothesis: pruning redundant content from verbose teacher-generated reasoning paths can improve both the quality and the efficiency of the knowledge distilled into smaller student models.

In terms of specific model performance, Pru-CoT shows significant gains. For the 7B student model, Pru-CoT substantially improves performance on the challenging AIME24/25 datasets. Specifically, accuracy on AIME24/25 improves by 7.5%, while maintaining strong performance on AMC23 and MATH500 compared with other baseline methods. For the 1.5B student model, Pru-CoT achieves the joint-best ACU score (0.98), tying with A*-Thought, while retaining a higher average accuracy (48.1% vs. 45.4%). Compared to the remaining baselines, we acknowledge a slight decline in absolute accuracy, likely due to the limited capacity of the 1.5B model to sustain performance trade-offs under compression. Nevertheless, Pru-CoT offers satisfactory efficiency, reducing the average reasoning length by approximately 9.5%.

4.3 Scalability and Robustness Analysis

To evaluate the robustness and scalability of Pru-CoT, we conduct a comparative analysis across varying inference token budgets ranging from 4,096 to 12,288 tokens. As shown in Figure 3a, while the reasoning performance of all models exhibits positive scaling with increased budgets, Pru-CoT consistently outperforms all baselines, across the entire spectrum of computational constraints. Even at the maximum budget of 12,288 tokens, Pru-CoT maintains its superiority and achieves the highest accuracy, demonstrating that its advantages are robust across different reasoning lengths rather than confined to a specific budget. This efficiency is further elucidated in Figure 3b, where Pru-CoT achieves the highest ACU scores at all evaluated points, indicating a superior performance-to-cost trade-off. Crucially, while all methods exhibit an expected downward trend in ACU as the budget expands, Pru-CoT exhibits the most gradual decay. This slower degradation suggests superior scalability, as Pru-CoT is able to utilize extended reasoning paths more effectively than competitors while maintaining higher marginal utility per token.

4.4 Impact of Pruning Agent Scale

To investigate the impact of pruning agent scale, i.e., the parameter size of the LLM-based pruning agent used in the LLM-guided pruning phase, we conduct controlled experiments across two distinct settings. Specifically, we introduce significant scale disparities between the pruning agent and

Methods	AIME24/25		AMC23		MATH500		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
DeepSeek-R1-Distill-Qwen-7B									
SFT	30.8	6956	70.6	5035	80.2	3691	60.5	5227	1.16
Opt. Only	33.8	**6766**	**75.6**	**4662**	**83.1**	3102	64.2	**4843**	1.33
LLM Only	28.8	7046	72.5	4898	80.4	3332	60.6	5092	1.19
Pru-CoT	**38.3**	6870	75.0	4764	82.6	**3070**	**65.3**	4871	**1.34**

Table 2: Ablation study of Pru-CoT on DeepSeek-R1-Distill-Qwen-7B. The best result is in bold, and the second best is underlined. The full Pru-CoT pipeline (both stages) achieves the highest accuracy and ACU, validating the necessity of its two-stage design.

Methods	Step 1	Step 2	Total
DeepSeek-R1-Distill-Qwen-1.5B			
A*-Thought (Xu et al., 2025b)	0.32	7.67	7.99
Pru-CoT (Ours)	1.03	0.79	**1.82 (4.4×)**

Table 3: Time consumption (GPU hours) of A*-Thought and Pru-CoT on the training dataset with the DeepSeek-R1-Distill-Qwen-1.5B model.

the student model: (1) employing the highly capable DeepSeek-V3.2 (DeepSeek-AI, 2025b) to prune traces for the 7B student model, and (2) utilizing Qwen2.5-7B-Instruct to prune traces for the 1.5B student model. In both cases, we compare the results against our proposed scale-aligned method. The results reveal a critical and consistent trade-off. As shown in Figure 4a and Figure 4b, students trained on traces pruned by significantly stronger agents exhibit markedly lower performance in average accuracy, while their ACU remains largely comparable to those pruned by scale-matched agents.

We hypothesize that this performance drop stems from a fundamental behavioral mismatch. A more powerful model often internalizes certain reasoning steps as "obvious" leading it to prune them during trace simplification. However, these steps represent critical scaffolding for the significantly less capable student. Moreover, as shown in Figure 4a, although using DeepSeek-V3.2 for trace pruning on the 7B student model yields a slight improvement in the ACU metric, this limited gain is completely offset by its substantial computational cost.

4.5 Ablation Study

To validate the design of Pru-CoT, we conduct an ablation study to analyze the contributions of its two core components: (1) Global Optimization and (2) LLM-Guided Pruning. Table 2 shows the influence of each component.

Impact of Global Optimization. To evaluate the contribution of our global optimization framework, we introduce a variant termed Opt. Only. This variant performs direct pruning by removing reasoning steps whose importance weights fall below a predefined threshold τ. To ensure a fair comparison, we set $\tau = 0.4$ to match the 0.2 pruning ratio achieved by the full Pru-CoT. As illustrated in Table 2, Opt. Only exhibits a slight performance decline compared to the full model, primarily because aggressive threshold-based pruning can occasionally disrupt the logical flow. Nevertheless, it still outperforms all other baseline methods. This resilience underscores that our global optimization effectively captures the underlying logical dependencies and dynamic interactions between steps, ensuring that even after coarse pruning, the remaining chain retains sufficient structural integrity to support sound reasoning.

Impact of LLM-Guided Pruning. We further investigate the necessity of LLM-guided pruning through a variant called LLM Only. In this setup, we directly prompt the LLM to compress original reasoning chains to the target pruning ratio of 0.2. The results in Table 2 reveal a significant deterioration in performance. This drop suggests that, absent the "semantic anchors" provided by global optimization, the LLM struggles to identify critical nodes and fails to maintain a logically coherent path. This contrast reinforces our two-stage design: global optimization identifies pruneable segments versus essential anchors, while the LLM leverages these anchors to execute precise pruning, thereby preserving the overall logical integrity.

4.6 Efficiency Analysis

Table 3 presents a detailed breakdown of the time consumption for A*-Thought (Xu et al., 2025b) versus our proposed Pru-CoT. To ensure a fair comparison, all experiments are conducted under iden-

Question: ... There are n players participating in a tennis tournament. Each player is eliminated from the tournament after their first loss. How many matches need to be played to determine the winner?

Chain-of-Thought

Weight	Prune	Content
0.0	True	Okay, let's try to figure out how many matches are needed... each match eliminates exactly one player... So every match removes exactly one player.
1.0	False	So if we start with n players, we need to eliminate $n-1$ players... Therefore, the number of matches required should be $n-1$... Let's take some small examples.
0.0	False	For example, if there are 2 players... That checks out. If there are 3 players... That works. What if there are 4 players?... Yeah, that also works...
0.02	True	But wait, maybe there's a different structure... Therefore, the total number of matches needed is always $n-1$...
0.0	True	Alternatively, maybe I can think of it as a binary tournament bracket... Therefore, the answer is $n-1$.
1.0	False	Wait, but let me test with another example. Let's take 5 players... So $5-1=4$. Correct...
1.0	False	Another way to think about it: each player except the winner loses exactly once... Therefore, the number of matches is equal to the number of losses, which is $n-1$.
0.63	False	So the key insight is that every match results in exactly one elimination... Therefore, the number of matches required is $n-1$. That seems solid.

Solution: ... 1. Key Insight: Every match eliminates exactly one player. 2. Total Eliminations Needed: To leave one winner, $n-1$ players must be eliminated. 3. Matches = Eliminations: Each match corresponds to one elimination. Final Answer: $\boxed{n-1}$

Table 4: A case study of Chain-of-Thought pruning using our method. The Weight column shows importance scores derived from global optimization (Sec. 3.3), while Prune marks steps ultimately discarded by the LLM-guided phase (Sec. 3.4). Ellipses (...) indicate content truncated for brevity.

tical GPU memory constraints. For A*-Thought, Step 1 and Step 2 represent the bidirectional importance score calculation and the A* search process. In contrast, for Pru-CoT, Step 1 corresponds to the global optimization phase, while Step 2 performs the LLM-guided pruning. The results demonstrate that Pru-CoT significantly reduces computational overhead, achieving a 4.4× speedup in total execution time compared to A*-Thought (1.82 vs. 7.99 GPU hours). This highlights that Pru-CoT offers a more computationally efficient and scalable solution for CoT pruning.

4.7 Case Study

Table 4 illustrates the pruning process of Pru-CoT on a specific chain-of-thought example. Taking Steps 1 and 3 as examples, both receive a weight of 0 from the global optimization. This is because Step 1 merely restates the problem, and Step 3 only reiterates the answer obtained from Step 2 through examples without introducing new insights. However, the LLM-guided pruning step retains Step 3 instead of pruning it. The reason lies in the contextual flow: Step 2 ends with "Let's consider some simple examples", while Step 6 begins with "Let me test another example", which presupposes that an example has already been discussed. Thus, preserving Step 3 helps maintain the coherence of the reasoning narrative. This case study further confirms that the two-stage pruning approach can accurately identify redundant reasoning steps while ensuring that the natural flow of the reasoning process is retained when such steps are removed.

5 Conclusion

In this paper, we introduce Pru-CoT (Pruning Chain-of-Thought), a novel framework for distilling efficient reasoning paths from large-scale reasoning models. To precisely quantify the causal contribution of each reasoning step and filter out

"overthinking" noise, we perform step-level importance estimation through a global optimization mechanism on a frozen model. To mitigate semantic discontinuity and ensure the logical validity of the compressed reasoning traces, we devise an LLM-guided pruning process with fidelity constraints to synthesize concise, coherent narratives using semantic anchors. Extensive experiments on multiple mathematical reasoning benchmarks with DeepSeek-R1-Distill-Qwen series models validate the superiority of the proposed approach. In future work, we plan to investigate its deployment in resource-constrained settings for greater generalizability and practicality.

Limitations

Our work is subject to several limitations, primarily stemming from computational constraints. First, our experimental validation on Pru-CoT was conducted on datasets and model scales that are relatively small in size. Consequently, we were unable to explore the potential and scalability of our method on larger architectures and more extensive benchmarks. Second, due to the resource demands of the global optimization procedure, we were unable to perform an exhaustive search over the hyperparameter space or systematically investigate the impact of using different filler tokens on optimization performance. These unexplored dimensions may hold the key to further efficiency gains or robustness improvements of Pru-CoT.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62206038, 62106035), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA0490301), Liaoning Binhai Laboratory Project (No. LBLF-2023-01), Xiaomi Young Talents Program, and the Interdisciplinary Institute of Smart Molecular Engineering, Dalian University of Technology.

References

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 long reasoning models. In *International Conference on Machine Learning (ICML)*, pages 9487–9499.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, Suhang Wang, Yue Xing, Jiliang Tang, and Qi He. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. In *Findings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 18581–18597.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

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

DeepSeek-AI. 2025b. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv*, abs/2512.02556.

Mengru Ding, Hanmeng Liu, Zhizhang Fu, Jian Song, Wenbo Xie, and Yue Zhang. 2024. Break the chain: Large language models can be shortcut reasoners. *arXiv*, abs/2406.06580.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Etash Kumar Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, Ashima Suvarna, Benjamin Feuer, Liangyu Chen, Zaid Khan, Eric Frankel, Sachin Grover, Caroline Choi, Niklas Muennighoff, Shiye Su, and 31 others. 2025. Openthoughts: Data recipes for reasoning models. *arXiv*, abs/2506.04178.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2026. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Trans. Mach. Learn. Res.*, 2026.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Symposium on Operating Systems Principles (SOSP)*, pages 611–626.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

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

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv*, abs/2501.12570.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2025. CoT-valve: Length-compressible chain-of-thought tuning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6025–6035.

OpenAI. 2024. Openai o1 system card. *arXiv*, abs/2412.16720.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 963–981.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *arXiv*, abs/2404.15758.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 3505–3506.

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. *arXiv*, abs/2510.01581.

Kimi Team. 2025a. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv*, abs/2501.12599.

Qwen Team. 2025b. Qwen3 technical report. *arXiv*, abs/2505.09388.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 95266–95290.

Zhaoyang Wang, Jinqi Jiang, Tian Qiu, Hui Liu, Xianfeng Tang, and Huaxiu Yao. 2025. Efficient long cot reasoning in small language models. *arXiv*, abs/2505.18440.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 24824–24837.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. TokenSkip: Controllable chain-of-thought compression in LLMs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3351–3363.

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

Xiaoang Xu, Shuo Wang, Xu Han, Zhenghao Liu, Huijia Wu, Pei Pei Li, Zhiyuan Liu, Maosong Sun, and Zhaofeng He. 2025b. A*-thought: Efficient reasoning via bidirectional compression for low-resource settings. In *Conference on Neural Information Processing Systems (NeurIPS)*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 technical report. *arXiv*, abs/2407.10671.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024b. Qwen2.5 technical report. *arXiv*, abs/2412.15115.

Wenhao Zeng, Yaoning Wang, Chao Hu, Yuling Shi, Chengcheng Wan, Hongyu Zhang, and Xiaodong Gu. 2025. Pruning the unsurprising: Efficient code reasoning via first-token surprisal. *arXiv*, abs/2508.05988.

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. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13307–13328.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 400–410.

A Prompt Details for LLM-Guided Pruning

Our LLM-guided pruning phase (Section 3.4) employs a two-part prompt: a system prompt that defines aggressive pruning rules and specifies a JSON output format, and a user prompt that provides the problem, reasoning steps, and the corresponding answer.

System Prompt

You are an **aggressive** CoT pruner. Your goal is to **remove all non-essential steps**.

Core Principle:

1. **Default:** `"prune": True`. Prune all redundant steps, clarifications, or checks.
2. **Exception:** `"prune": False`. ONLY keep steps that are **critically essential**. A step is essential *only if* its removal breaks the logical chain or makes the solution impossible.
3. **Rule: If in doubt, prune.**

Task & Format Rules:

1. You will see the thinking process in a table.
2. **ONLY focus on steps that have a Number ID** in the first column. Steps with an empty ID are protected context.
3. Respond with *only* a JSON object.
4. **Keys:** Must be the Number ID from the table (e.g., "1", "2").
5. **Values:** Must be {"reasoning": (string), "prune": (boolean)}.

Example Response:

```
{
   "1": {
     "reasoning": "Prune. Step 1 is a
redundant clarification.",
     "prune": True
   },
   "2": {
     "reasoning": "Keep. Step 2 defines
variable x...",
     "prune": False
   }
}
```

User Prompt Template

Here is the problem:
`{user_input}`

Here is the final solution:
`{solution}`

YOUR TASK:
Below is the **full thinking process**. Analyze **ONLY** the steps with a **Number ID** in the first column.

```
| ID | Thinking Step |
|-|-|
{markdown_table_rows}
```

For each step with an ID, decide if it can be pruned (`True`) or if it is logically essential (`False`).
Respond *only* with the JSON dictionary.

The reasoning steps are structured as a table with two columns: step index and step content. In this table, we reassign consecutive indices only to the steps that are prunable, while leaving the index column blank for previously defined non-prunable steps. This design prevents the pruning agent from hallucinating and pruning non-prunable reasoning steps. Finally, the JSON output is parsed to identify which specific steps should be pruned.

B Experimental Details

For the experiments in Section 3.4, we utilize the vLLM library (Kwon et al., 2023) to accelerate pruning. The generation process is configured with a temperature of 0.9, a top-p of 0.95, and a maximum sequence length of 8,192 tokens. To maximize throughput on our hardware, we set the GPU memory utilization to 0.9.

We implement the training pipeline described in Section 4 using the LLaMA-Factory framework (Zheng et al., 2024)[2], incorporating FlashAttention-2 (Dao, 2024) and DeepSpeed ZeRO Stage 2 (Rasley et al., 2020) with CPU offloading to optimize memory usage. Models are fine-tuned for 2 epochs using BFloat16 (BF16) precision and the AdamW optimizer with a cosine learning rate scheduler. We set the peak learning rate to 1×10^{-6} and the per-device batch size to 1

[2] https://github.com/hiyouga/LlamaFactory

with 16 gradient accumulation steps across 2 GPUs, resulting in an effective global batch size of 32.

Following the training phase, evaluations are conducted using the lm-evaluation-harness (Gao et al., 2024)[3], leveraging vLLM with a GPU memory utilization of 0.9 for acceleration. To ensure a fair comparison, we maintain strict consistency in the training environments across all baselines. The SFT baseline utilizes the hyperparameters identical to those described above. For BtC, we employ the specific prompt "Let's skip as much as possible." (Ding et al., 2024) based on SFT. For Token-skip and A*-Thought, we replicate the data processing and search algorithms detailed in their respective papers (Xia et al., 2025; Xu et al., 2025b), while keeping the base model and fine-tuning configurations identical to our method to isolate the algorithmic contributions. All experiments, including pruning, training, and evaluation, are performed on two NVIDIA L20 GPUs.

C Additional Experimental Results

C.1 Results on the Qwen3 Model

We evaluate the performance of the proposed framework on the Qwen3-8B (Team, 2025b). As shown in the Table 5, Pru-CoT achieves the highest average accuracy (61.8%) among all methods, while its ACU score (1.13) is on par with that of BtC (1.14) and significantly outperforms other baselines. These results demonstrate the strong generalization ability of our approach.

C.2 Results on a Scaled-Up Training Dataset

To assess scalability and generalization with more training data, we expand the training dataset to include 10k samples. We perform fine-tuning with 1 training epoch. As shown in the Table 6, Pru-CoT achieves the highest average accuracy (66.9%) and ACU score (1.44) among all methods. These results confirm the strong performance of our approach with a larger training dataset.

C.3 Results on Additional Benchmarks

To demonstrate that our method extends beyond math problems to broader general reasoning tasks, we conduct additional experiments on subsets of the MMLU-Pro (Wang et al., 2024). As shown in Table 7, the consistent performance gains across these tasks confirm that our method benefits other domains as well.

D Extending LLM Pruning with Rewriting

Building upon the standard pruning strategy described in Section 3.4, we further extend the LLM-guided pruning agent in this section to support an additional rewrite action alongside the original keep and prune operations. The rewrite action aims to preserve essential logical content while compressing verbose expressions into more concise forms, potentially offering a middle ground between completely retaining and discarding steps.

However, as shown in Table 8, the rewriting variant underperforms compared to the standard Pru-CoT. Specifically, on the challenging AIME24/25 dataset, the rewrite variant yields only 32.5% accuracy, a substantial 5.8 percentage points lower than standard Pru-CoT. This performance gap suggests that LLM-guided rewriting may introduce potential risks, such as inadvertently altering logical semantics or generating hallucinated content.

E Discussion of the License

All assets (including code frameworks, libraries, and baselines) used in this work are publicly available and utilized in strict accordance with their respective licenses.

[3] https://github.com/EleutherAI/lm-evaluation-harness

Methods	AIME24/25		AMC23		MATH500		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
Qwen3-8B									
SFT	29.6	7428	71.3	5493	81.4	3904	60.8	5608	1.08
BtC (Ding et al., 2024)	**30.4**	**7298**	69.4	**5065**	80.1	**3463**	60.0	**5275**	**1.14**
Tokenskip (Xia et al., 2025)	13.8	8000	43.8	6912	70.2	5300	42.6	6737	0.63
A*-Thought (Xu et al., 2025b)	24.2	7549	65.0	5649	82.0	3781	57.1	5660	1.01
Pru-CoT (Ours)	29.6	7377	**73.1**	5322	**82.6**	3674	**61.8**	5458	1.13

Table 5: Results of different methods on Qwen3-8B.

Methods	AIME24/25		AMC23		MATH500		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
DeepSeek-R1-Distill-Qwen-7B									
SFT	35.0	6925	65.0	4926	81.7	3586	60.6	5146	1.18
BtC (Ding et al., 2024)	35.0	6921	69.4	4970	79.7	3702	61.4	5198	1.18
Tokenskip (Xia et al., 2025)	32.9	6923	70.0	5030	81.3	3430	61.4	5128	1.20
A*-Thought (Xu et al., 2025b)	37.5	**6578**	**78.1**	**4384**	82.3	2871	66.0	**4611**	1.43
Pru-CoT (Ours)	**41.7**	6611	76.3	4478	**82.6**	**2840**	**66.9**	4643	**1.44**

Table 6: Results of different methods with a scaled-up training dataset.

Methods	Computer Science		Philosophy		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
DeepSeek-R1-Distill-Qwen-7B							
SFT	33.4	2552	15.2	1923	24.3	2238	1.09
BtC (Ding et al., 2024)	36.3	2536	13.8	1954	25.1	2245	1.12
Tokenskip (Xia et al., 2025)	33.2	**2094**	13.6	**1323**	23.4	**1709**	**1.37**
A*-Thought (Xu et al., 2025b)	**38.0**	2575	15.8	1599	26.9	2087	1.29
Pru-CoT (Ours)	37.3	2368	**18.0**	1685	**27.7**	2027	**1.37**

Table 7: Results of different methods on MMLU-Pro subsets (Computer Science and Philosophy).

Methods	AIME24/25		AMC23		MATH500		Avg.		ACU↑
	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	Acc.↑	Tokens↓	
DeepSeek-R1-Distill-Qwen-7B									
SFT	30.8	6956	70.6	5035	80.2	3691	60.5	5227	1.16
Pru-CoT(rewrite)	32.5	6958	**77.5**	**4642**	81.8	3167	63.9	4922	1.30
Pru-CoT	**38.3**	**6870**	75.0	4674	**82.6**	**3070**	**65.3**	**4871**	**1.34**

Table 8: Comparison of Pru-CoT with and without rewriting.