CRISP: Compressing Redundancy in Chain-of-Thought via Intrinsic Saliency Pruning

Yangsong Lan, Hongliang Dai, Piji Li*
[1] College of Artificial Intelligence,
Nanjing University of Aeronautics and Astronautics, Nanjing, China
[2] The Key Laboratory of Brain-Machine Intelligence Technology, Ministry of Education, Nanjing, China.
{lys2962331781, hongldai, pjli}@nuaa.edu.cn

Abstract

Long Chain-of-Thought (CoT) reasoning is pivotal for the success of recent reasoning models but suffers from high computational overhead and latency. While prior works attempt to compress CoT via external compressor, they often fail to align with the model's internal reasoning dynamics, resulting in the loss of critical logical steps. This paper presents **C**ompressing **R**edundancy in Chain-of-Thought via **I**ntrinsic **S**aliency **P**runing (**CRISP**), a framework that compresses CoT by exploiting the model's intrinsic saliency. Our analysis reveals a distinct phenomenon: the reasoning termination token `</think>` acts as an information anchor, where its attention pattern effectively demarcates essential reasoning from redundancy. Based on this finding, we design a policy that utilizes these intrinsic attention signals to guide atomic compression operations. In contrast to coarse-grained pruning strategies, CRISP strategically distills the reasoning chain to maximize information density while preserving logical coherence. Empirical results across various backbone models and mathematical datasets demonstrate that CRISP achieves a 50-60% reduction in token count without compromising accuracy, effectively mitigating the efficiency bottleneck of long-context reasoning. We open-source our implementation to facilitate further research in efficient reasoning[1].

1 Introduction

The emergence of reasoning-oriented LLMs, represented by OpenAI o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025), QwQ (Renze and Guven, 2024), and Kimi k1.5 (Team et al., 2025), marks a significant paradigm shift, demonstrating superior performance in complex reasoning domains. While these models achieve superior reasoning capabilities by decomposing complex problems into extensive Chains-of-Thought and employing iterative verification (Wei et al., 2022), this approach incurs substantial computational overhead (Feng et al., 2025; Sui et al., 2025). Such inefficiency renders deployment infeasible in resource-constrained environments, making it imperative to develop methods that distill compact reasoning paths without compromising model fidelity.

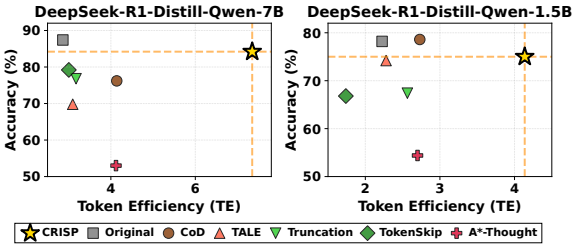

Figure 1: **Token Efficiency (TE.) vs. Accuracy on DeepSeek-R1-Distill-Qwen-7B/1.5B**. CRISP achieves the best trade-off, significantly outperforming baselines in efficiency while maintaining high accuracy.

To alleviate this computational burden, prevalent approaches (Xia et al., 2025; Xu et al., 2025b; Yan et al., 2025) resort to CoT compression, typically utilizing external proxy models to prune redundancy. However, this reliance on external compressors introduces a fundamental misalignment: these proxies are agnostic to the source model's intrinsic reasoning dynamics. External proxies often misclassify essential intermediate steps, particularly self-corrections, as redundancy, thereby ignoring their critical role in the source model's logical continuity. Consequently, fine-tuning on such mutilated sequences risks disrupting the coherence of the reasoning chain. This limitation motivates two pivotal research questions: (1) How can we identify pivotal reasoning steps solely based on the model's intrinsic signals? (2) How can we synthesize a concise yet coherent reasoning chain utilizing these identified pivots?

In this work, we address these challenges by in-

*Corresponding author.
[1]Our implementation is available at GitHub.

Findings of the Association for Computational Linguistics: ACL 2026, pages 39355–39373

vestigating the internal attention mechanisms of reasoning models. We observe that the </think> token, serving as the delimiter of the reasoning phase, acts as a critical information anchor. Our analysis reveals that during the generation of the final answer, the model attends minimally to intermediate reasoning steps but maintains high attention weights on the </think> token. We empirically demonstrate that the attention pattern at this position reliably indicates the saliency of preceding reasoning steps, effectively distinguishing essential logic from redundancy.

Building upon this observation, we propose **CRISP**, a framework that exploits the attention landscape at the </think> token to steer the compression process. We cast CoT compression as a search problem defined over a quadruplet of atomic operators: FUSE, PRUNE, REWRITE, and KEEP. By employing a reward function that harmonizes step-wise saliency with sequence length, our method efficiently navigates the reasoning space. To mitigate the logical discontinuities often introduced by discrete search operations, we further employ an advanced LLM-based refiner to refine the retrieved paths. This step effectively restores semantic coherence, yielding compressed chains that are both concise and logically fluid. Subsequently, we fine-tune the target model on these refined sequences via a multi-task learning objective.

As shown in Figure 1, CRISP achieves a superior efficiency-accuracy trade-off over strong baselines. Our framework effectively isolates essential reasoning pivots from redundancy, maintaining robust capabilities even under strict constraints.

To summarize, our main contributions are as follows:

- We identify that the attention weights at the </think> token serve as a reliable intrinsic indicator of reasoning step saliency, effectively distinguishing essential logic from redundancy without external proxies.
- We propose **CRISP**, a framework that optimizes CoT via a greedy search over four atomic operators (FUSE, PRUNE, REWRITE, KEEP) guided by intrinsic signals.
- We demonstrate that **CRISP** offers a superior trade-off between efficiency and accuracy, significantly reducing inference overhead without compromising the backbone model's performance.

2 Related Works

Reasoning in Large Language Models Chain-of-Thought has established itself as a cornerstone paradigm for Large Language Models (Wang et al.; Wei et al., 2022), enhancing interpretability and accuracy by decomposing complex tasks into intermediate reasoning steps (Wang et al., 2022; Zhou et al., 2022). Subsequent frameworks, such as Tree of Thoughts (ToT) (Yao et al., 2023) and Program of Thoughts (PoT) (Chen et al., 2022), have further expanded these capabilities. More recently, models like DeepSeek-R1 (Guo et al., 2025), Kimi k1.5 (Team et al., 2025) have demonstrated that reinforcement learning algorithms (Shao et al., 2024; Yu et al., 2025; Zheng et al., 2025), coupled with meticulously designed reward signals, can unlock even deeper reasoning potential. However, this enhanced reasoning capability typically comes at the cost of excessive token generation and significant computational overhead (Sui et al., 2025; Chiang and Lee, 2024). Consequently, a growing body of research is shifting focus toward efficient reasoning, aiming to compress the CoT process and achieve an optimal trade-off between model performance and inference latency.

Chain-of-Thought Compression To mitigate inference latency, recent research seeks to compress CoT outputs without compromising reasoning efficacy (Cui et al., 2025; Wang et al., 2025; Qiao et al., 2025). Initial efforts utilizing prompt engineering strategies (Xu et al., 2025a; Han et al., 2025) attempt to constrain sequence length via explicit instructions, yet often struggle with granular control and adherence to constraints, leading to potential degradation in generation quality. Reinforcement learning frameworks (Aggarwal and Welleck, 2025; Luo et al., 2025) explicitly incentivize brevity by incorporating length penalties into the reward function. However, this paradigm entails significant computational overhead and exhibits acute sensitivity to reward shaping, often leading to optimization instability. Concurrently, supervised fine-tuning paradigms (Xia et al., 2025; Yan et al., 2025; Xu et al., 2025b) aim to distill concise reasoning by pruning redundant steps. However, these methods typically utilize auxiliary models as external compressors to condense trajectories. This dependency creates a misalignment, as the external compression logic often diverges from the target model's intrinsic generation dynamics. Distinguishing itself from approaches that depend

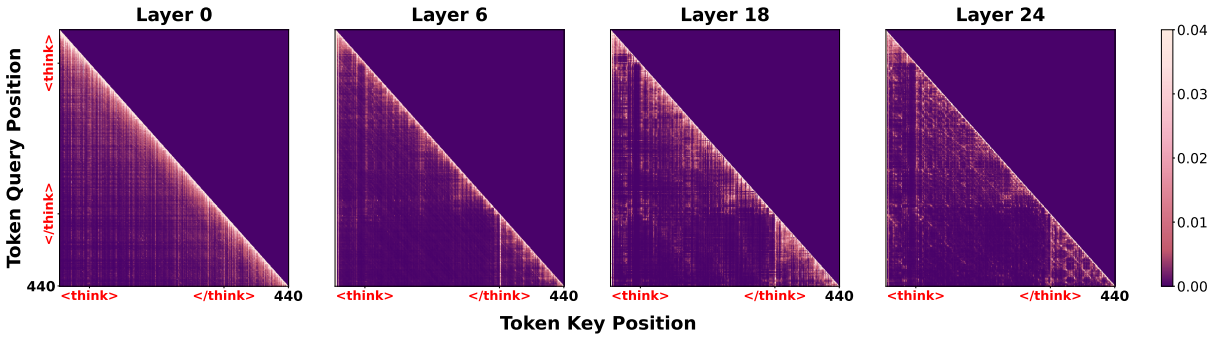

Figure 2: **Visualization of layer-wise attention dynamics in DeepSeek-R1-Distill-Qwen7B.**The heatmaps depict layer-wise attention distributions during inference. While shallow layers exhibit uniform attention across the context, deep layers reveal the </think> token functioning as a semantic anchor, progressively aggregating information from the reasoning chain to guide final answer generation.

on extrinsic constraints or offline heuristics, our method leverages the model's intrinsic attention mechanisms to guide adaptive CoT compression, ensuring that efficiency is derived directly from the instance-specific reasoning process.

3 </think> as the Information Anchor

Unlike prior approaches that rely on external compressors to condense CoT, our objective is to determine whether the model intrinsically distinguishes the contribution of specific steps toward the final answer. To investigate the inherent attention patterns governing this interaction, we visualize the layer-wise attention maps of DeepSeek-R1-Distill-Qwen7B on GSM8K samples, as illustrated in Figure 2. Motivated by the hypothesis that specific tokens function as informational anchors (Wang et al., 2023; Li et al., 2025), our analysis focuses on the temporal dynamics of the <think> and </think> tokens, which delimit the reasoning boundaries.

Our visualization reveals that while attention distribution remains relatively uniform in early layers, the </think> token progressively functions as a semantic anchor in deeper layers, aggregating information from the preceding reasoning chain. Crucially, during the generation of the final answer, the model predominantly attends to the representation at the </think> position, while direct attention to the raw reasoning chain diminishes. Consistent with findings in (Choi et al., 2025), steps that make high contributions to the solution retain high attention scores specifically within the </think> token's attention column, suggesting that the model compresses the reasoning history into this single token state. **Full attention profiles across all layers for both the 1.5B and 7B scales are detailed in Appendix D.4.**

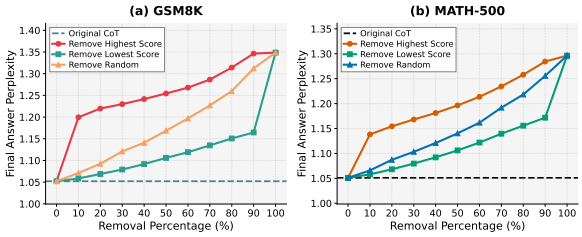

Figure 3: **Validation of anchor-guided redundancy identification.** Pruning reasoning steps with high attention to the </think> anchor precipitates a sharp PPL spike, whereas removing low-attention steps results in a significantly more gradual increase.

To empirically validate the role of </think> as a proxy for identifying information redundancy, we conducted a stepwise pruning experiment on the GSM8K and MATH-500 datasets, as detailed in Figure 3. We selectively pruned reasoning steps corresponding to the lowest, highest, and random attention scores registered at the </think> anchor, measuring the subsequent impact on the Perplexity (PPL) (Jelinek et al., 1977) of the final answer. Our empirical results demonstrate that pruning steps with high attention scores induces a sharp spike in PPL, suggesting these steps encode critical information. Conversely, pruning low-attention steps leads to only a marginal PPL increase, while random pruning results in an intermediate performance degradation. This empirical insight serves as the foundational premise for our proposed framework, **CRISP**, which leverages these intrinsic attention signals to guide efficient and adaptive chain-of-thought compression.

4 CRISP: Compressing Redundancy via Intrinsic Saliency Pruning

Building upon the identification of the </think> token as an informational anchor, we propose

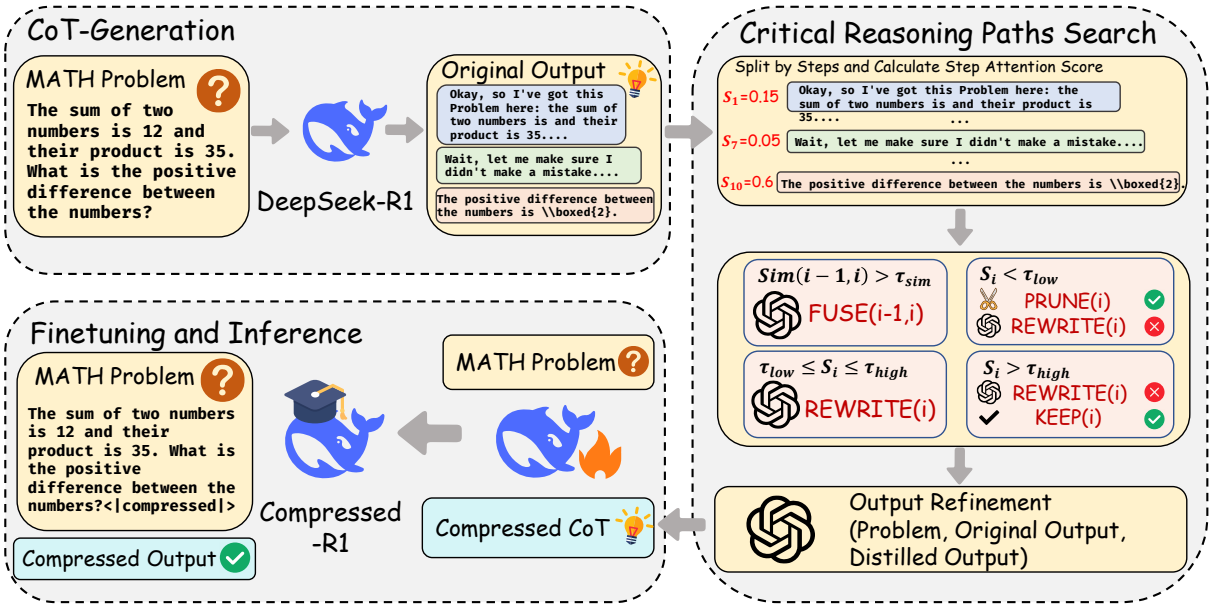

Figure 4: **Overview of the CRISP framework.** The process comprises: **(1) CoT-Generation**, eliciting raw reasoning trajectories from the source model; **(2) Critical Reasoning Paths Search**, which evaluates step salience via attention scores and distills chains using dynamic operators (KEEP, FUSE, PRUNE, REWRITE) followed by generative refinement; and **(3) Finetuning and Inference**, employing these refined, high-density trajectories as supervision for efficient reasoning generation.

CRISP (Figure 4), a framework designed to distill efficient reasoning paths from the model's internal attention dynamics.

Unlike prevailing compression strategies often rely on external language models for step evaluation, introducing the value misalignment highlighted in Section 2. In contrast, **CRISP** shifts from extrinsic supervision to endogenous self-selection. We formulate CoT compression as a structured search governed by intrinsic attention dynamics. By utilizing four atomic operators (KEEP, PRUNE, REWRITE, and FUSE), our framework iteratively constructs a critical reasoning skeleton. Subsequently, an auxiliary model serves as a linguistic refiner to restore syntactic coherence without compromising logical substance. This decoupled design ensures that the compressed CoT retains the target model's inherent reasoning fidelity while achieving textual fluency.

4.1 Original CoT Generation

Let $\mathcal{M}_\theta$ denote the target Large Language Model parameterized by θ. Given an input query x and a reasoning instruction $\mathcal{I}$, the model generates a response sequence comprising a reasoning chain $\mathcal{R}_{\text{orig}}$ and a subsequent final answer y. We formulate this joint generation process as sampling from the model's posterior distribution:

$$(\mathcal{R}_{\text{orig}}, y) \sim P_\theta(r, y \mid x, \mathcal{I}) \quad (1)$$

Here, $\mathcal{R}_{\text{orig}} = \{r_1, r_2, \ldots, r_L\}$ consists of L discrete reasoning steps. This sequence encapsulates the comprehensive yet potentially redundant inference process that precipitates the final answer y. $\mathcal{R}_{\text{orig}}$ serves as the foundational input for our framework, providing the raw trace from which the essential reasoning signal is distilled.

4.2 Critical Reasoning Step Search and Compression

Given the raw reasoning trajectory $\mathcal{R}_{\text{orig}}$ generated by the model, our objective is to distill a compact reasoning skeleton that preserves logical fidelity. We begin by decomposing $\mathcal{R}_{\text{orig}}$ into L discrete steps, $\mathcal{R}_{\text{orig}} = \{r_1, r_2, \ldots, r_L\}$, using the standard double newline `\n\n` delimiter. To evaluate the utility of each step without relying on external supervision, we leverage the intrinsic attention dynamics anchored to the `</think>` token (as detailed in Section 3). Since `</think>` marks the transition from reasoning to the final answer, its attention distribution naturally highlights the antecedent steps most critical to the prediction. Formally, we quantify the endogenous contribution S_i of step r_i by aggregating the attention weights from `</think>` to all tokens t_j within that step:

$$S_i = \frac{1}{|r_i|} \sum_{t_j \in r_i} \sum_{l=1}^{N_L} \sum_{h=1}^{N_H} \mathbf{A}_{h,l}(\texttt{</think>}, t_j) \quad (2)$$

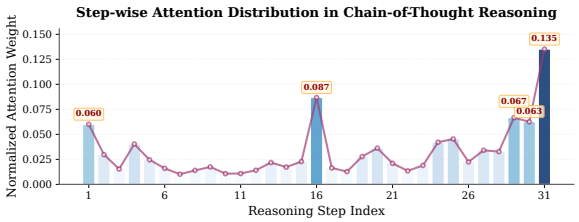

Figure 5: **Step-wise Attention Distribution in Chain-of-Thought.** The normalized scores S_i exhibit a non-uniform distribution, highlighting that critical information is localized in a few key steps.

Condition	**Allowed Actions ($\mathcal{A}_i$)**
$\text{Sim}(\mathcal{C}_{\text{last}}, r_i) \geq \tau_{\text{sim}}$	FUSE
$S_i < \tau_{\text{low}}$	PRUNE, REWRITE
$\tau_{\text{low}} \leq S_i \leq \tau_{\text{high}}$	REWRITE
$S_i > \tau_{\text{high}}$	KEEP, REWRITE

Table 1: Mapping of state conditions to the set of allowed actions $\mathcal{A}(\mathcal{S}_i)$. The conditions depend on the score S_i thresholds and semantic similarity.

Figure 5 visualizes the distribution of these attention scores. The plot demonstrates that not all steps hold high salience relative to the input query and the final solution; rather, only a limited subset of the trajectory offers significant contributions to the inference process. However, relying solely on static threshold-based filtration of S_i is insufficient, as it risks severing critical logical dependencies or retaining redundant, low-density segments. To navigate the trade-off between conciseness and information fidelity, we formulate CoT compression as a structured greedy search. We process the original reasoning trajectory $\mathcal{R}_{\text{orig}} = \{r_1, \ldots, r_L\}$ sequentially. At each step i, given the partial compressed chain constructed thus far, denoted as $\mathcal{C}$, we select the optimal operator to process the current step r_i.

To adaptively mitigate inter-step redundancy and intra-step verbosity, we introduce a dynamic action space $\mathcal{A}_i$. This space comprises non-parametric operators (PRUNE, KEEP), which either discard or preserve the step, and generative operators (REWRITE, FUSE), which leverage an LLM to synthesize concise or merged representations. The admissible actions are constrained by a sequential protocol detailed in Table 1. Initially, the mechanism assesses semantic redundancy; steps exhibiting high similarity to the context tail $\mathcal{C}_{\text{last}}$ are processed via FUSE to consolidate information. Subsequently, for non-redundant steps, the action space is stratified by the salience score S_i: low-salience steps are candidates for PRUNE or REWRITE, intermediate steps are exclusively refined via REWRITE, while high-salience steps are eligible for either KEEP or REWRITE. This structured filtering effectively narrows the search space before reward evaluation.

To select the optimal operator from the allowed set $\mathcal{A}_i$, we employ a reward function that balances the gain in answer likelihood against sequence length. For a candidate action a, the reward is calculated as:

$$R(a) = \log P_\theta(y \mid x, \mathcal{C} \oplus a(r_i)) - \log P_\theta(y \mid x, \mathcal{C}) - \beta \cdot \text{Len}(a(r_i)) \quad (3)$$

Here, the first term quantifies the marginal improvement in the model's likelihood of predicting the correct answer y given the action, while the second term imposes a length penalty controlled by the hyperparameter β. We apply a greedy strategy to iteratively select the action maximizing $R(a)$, appending the result to $\mathcal{C}$ to construct the final reasoning skeleton R'. The complete pseudocode for this procedure is detailed in Appendix B.1.

4.3 Distilled CoT Refinement

While the greedy search effectively distills a logical skeleton R', the discrete application of operators can induce syntactic fragmentation or minor logical gaps. To restore textual coherence, we employ a generative refiner $\mathcal{M}_{\text{refine}}$ to reconstruct the final trajectory. The refinement process is formulated as:

$$\mathcal{R}_{\text{CRISP}} = \mathcal{M}_{\text{refine}}(x, R_{orig}, R') \quad (4)$$

Conditioned on the input query x and the original CoT R_{orig}, the refiner restores semantic coherence to $\mathcal{R}_{\text{CRISP}}$, smoothing disjointed transitions while strictly preserving the underlying logical substance. This ensures that $\mathcal{R}_{\text{CRISP}}$ attains linguistic fluency without compromising the fidelity of the compressed reasoning path.

4.4 Multi-Task Fine-Tuning and Inference

Following (Yan et al., 2025), we employ a multi-task fine-tuning strategy governed by a special control token κ (denoted as `<|compressed|>`). To incorporate this control signal, we construct the input x_{comp} for compressed samples by appending κ to the query x, wrapped in `[EOS]` markers:

$$x_{\text{comp}} = x \oplus \texttt{[EOS]}\kappa\texttt{[EOS]} \quad (5)$$

Let the target sequence $\mathbf{y} = \mathcal{R}_{\text{CRISP}}$ denote the compressed reasoning path. We optimize the

Method	GSM8K			MATH-500			AMC23			Average		
	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑
Model: DeepSeek-R1-Distill-Qwen-1.5B												
Original	81.6	1669	4.89	78.2	3515	2.22	60.0	5265	1.14	73.3	3483	2.10
Truncation	73.4	1311	5.60	67.4	2629	2.56	50.0	3258	1.53	63.6	2399	2.65
CoD	70.7	677	10.44	**78.6**	2879	2.73	60.0	4462	1.34	69.8	2673	2.61
TALE	68.7	752	9.13	74.2	3257	2.28	**62.5**	4812	1.19	68.5	2940	2.31
TokenSkip	72.9	1682	4.33	66.8	3841	1.74	50.0	5202	0.96	63.2	4295	1.77
A*-Thought	67.9	978	6.95	54.4	2015	2.70	42.5	**2528**	1.68	55.0	1840	2.99
CRISP (Ours)	**80.6**	**587**	**13.73**	75.0	**1813**	**4.14**	60.0	2607	**2.30**	**71.9**	**1669**	**4.31**
Model: DeepSeek-R1-Distill-Qwen-7B												
Original	90.8	1376	6.60	87.4	3053	2.86	72.5	4483	1.62	83.6	2971	2.81
Truncation	84.5	1191	7.09	76.8	2419	3.17	65.0	3035	2.14	75.4	2215	3.40
CoD	71.2	279	25.52	76.2	1841	4.14	**82.5**	3217	2.56	76.6	1779	4.31
TALE	67.9	**169**	**40.20**	69.8	2254	3.10	77.5	4128	1.73	71.7	2253	3.15
TokenSkip	84.7	1212	6.99	79.2	2636	3.00	72.5	3676	1.97	74.9	2665	2.81
A*-Thought	75.5	663	11.39	53.0	1286	4.12	37.5	**2160**	1.74	55.3	1370	4.04
CRISP (Ours)	**90.1**	374	24.09	**84.2**	**1146**	**7.35**	77.5	2180	**3.56**	**83.9**	**1235**	**6.80**

Table 2: Main results on GSM8K, MATH-500, and AMC23. We report Accuracy (Acc.), Average Tokens (Tok.), and Token Efficiency (TE). **Bold** and underline denote the best and second-best results, respectively.

model parameters θ by minimizing the negative log-likelihood over the reasoning tokens:

$$\mathcal{L} = -\sum_{t=1}^{|\mathbf{y}|} \log P_\theta(y_t \mid x_{\text{comp}}, y_{<t}) \qquad (6)$$

To preserve general reasoning capabilities and mitigate catastrophic forgetting, we mix original trajectories $\mathcal{R}_{\text{orig}}$ into the training corpus, omitting κ for these instances. During inference, appending the control sequence `[EOS]`κ`[EOS]` to the query acts as a steering signal, prompting the model to generate concise, high-density reasoning paths.

5 Experiments

5.1 Experimental Setup

Models and Training Data. To evaluate the efficacy and scalability of CRISP, we conduct experiments on two distilled reasoning models of varying scales: **DeepSeek-R1-Distill-Qwen-1.5B** and **DeepSeek-R1-Distill-Qwen-7B**. For the training corpus, we construct a balanced subset from the MATH dataset (Hendrycks et al., 2024) via stratified sampling, selecting 500 instances uniformly from each of the five difficulty levels to yield a total of 2,500 samples. We strictly filter these instances to ensure that the ground-truth answer is derivable by the base models and that the maximum sequence length does not exceed 8,192 tokens. Crucially, we leverage the CRISP framework to generate model-specific compression targets rather than employing a unified dataset; this ensures that the supervision signal remains faithful to each model's intrinsic saliency, thereby minimizing distribution shifts.

Evaluation Benchmarks. We evaluate our method on three benchmarks spanning varying difficulty levels: **GSM8K** (Cobbe et al., 2021) for grade-school math, **MATH-500** (Hendrycks et al., 2024) for comprehensive problem solving, and **AMC23** (AMC, 2025) for competition-level reasoning (see Appendix C.2 for details). Following standard protocols (Guo et al., 2025), we adopt a consistent decoding strategy with temperature 0.6 and top-p 0.95. To accommodate extended reasoning trajectories, we set the maximum generation length to 4,096 tokens for GSM8K and expand it to 8,192 tokens for the more complex MATH-500 and AMC23 datasets. Further implementation details are provided in Appendix C.3.

Metrics. To comprehensively evaluate the trade-off between reasoning capability and computa-

tional overhead, we report three metrics: **Accuracy (Acc.)**, which validates the answer against the ground truth; **Tokens (Tok.)**, denoting the average length of the generated reasoning trajectories; and **Token Efficiency (TE.)**. Following (Yan et al., 2025), Token Efficiency is defined as:

$$\mathrm{TE} = \frac{\mathrm{Acc}}{\mathrm{Length}} \times 100 \tag{7}$$

This composite metric captures the accuracy-per-token utility, with higher values indicating more efficient reasoning.

Implementation Details. We adopt a unified optimization framework for the proposed model and most baselines, training for 3 epochs with a peak learning rate of 1×10^{-5} and a warm-up ratio of 0.1. For the hyperparameters specific to our method, we set the balance coefficient β to 0.005 (as in Eq. 4). The attention thresholds τ_{high} and τ_{low} are empirically determined based on the top 30% and bottom 20% quantiles of the attention distribution, respectively. Additionally, we leverage SimCSE (Gao et al., 2021) as the backbone for semantic similarity measurement, setting the similarity threshold τ_{sim} to 0.7. Further details regarding the hardware infrastructure and computational costs are provided in Appendix C.3.

5.2 Baselines

To validate the effectiveness of CRISP, we benchmark it against a diverse set of redundancy-reduction methods:

- **Truncation**: A baseline method that applies a hard cutoff at a fixed token length, forcing termination strictly based on the token count rather than semantic completion.
- **Chain-of-Draft (CoD)** (Xu et al., 2025a): A prompting strategy mimicking human drafting, which constrains the model to generate information-dense, minimalist reasoning phrases instead of verbose sentences.
- **TALE** (Han et al., 2025): A length-constrained prompting baseline that explicitly instructs the model to reason within a strict token budget via natural language system prompts.
- **TokenSkip** (Xia et al., 2025): A fine-tuning baseline that employs prompt compression techniques LLMLingua2(Pan et al., 2024) to distill supervision data, enabling the model to learn information-dense reasoning patterns from the pruned CoT trajectories.

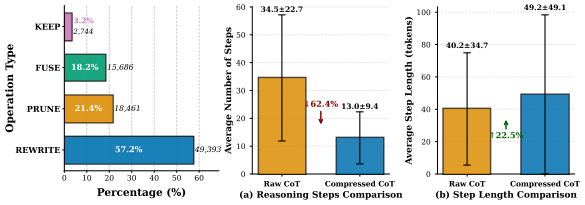

Figure 6: **Characterization of CRISP compression. Left**: Distribution of atomic operations favoring abstractive synthesis. **Right**: (a) Reasoning steps decrease by 62.4%, while (b) average step length increases by 22.5%, reflecting the consolidation of fragmented logical chains into information-dense units.

- **A*-Thought** (Xu et al., 2025b): A search-based compression framework that treats reasoning compression as a pathfinding problem. It employs an A* search algorithm guided by a bidirectional importance scoring mechanism to identify the optimal subset of reasoning tokens that preserves the deductive chain.

5.3 Results and Analysis

CRISP achieves a better trade-off between compression and performance. Table 2 compares CRISP with the backbone models and representative baselines. Overall, CRISP achieves a favorable balance between reasoning accuracy and inference efficiency. On the 7B benchmark, CRISP achieves an average accuracy of 83.9%, slightly exceeding the original model's 83.6%, while reducing token consumption by over 58% (1235 vs. 2971 tokens). This results in a Token Efficiency (TE) of 6.80, significantly outperforming both the original baseline and other compression methods. In comparison, prompt reduction methods such as TALE and CoD show a notable drop in accuracy, whereas finetuning approaches like TokenSkip yield limited efficiency gains. We also observe consistent performance on the 1.5B scale. Notably, on the AMC23 dataset with the 7B model, CRISP outperforms the original model by a clear margin (77.5% vs. 72.5%). This finding suggests that CRISP effectively filters out redundant reasoning steps that may introduce noise in language models, thereby serving as a mechanism for reasoning refinement.

How does CRISP optimize the trade-off between compression and performance? We dissect the efficiency gains by analyzing the atomic operations in Figure 6. The distribution characterizes an abstractive compression paradigm: REWRITE operations dominate (57.2%), while KEEP actions are negligible (3.2%). This underscores that CRISP

Method	Acc. ↑	Tok. ↓	TE. ↑
Model: DeepSeek-R1-Distill-Qwen-1.5B			
Original	78.2	3515	2.22
A*-Thought	54.4	2015	2.70
CRISP (w/o Refinement)	57.6	2265	2.54
CRISP (Full)	**75.0**	**1813**	**4.14**
Model: DeepSeek-R1-Distill-Qwen-7B			
Original	87.4	3053	2.86
A*-Thought	53.0	1286	4.12
CRISP (w/o Refinement)	70.6	1588	4.45
CRISP (Full)	**84.2**	**1146**	**7.35**

Table 3: **Ablation study on MATH-500.** "w/o Refinement" indicates the version with only Reasoning Search. **Bold** denotes the best results among inference-efficient methods (excluding Original).

actively synthesizes information to maximize semantic density rather than merely performing extractive pruning. Crucially, this synthesis facilitates a "step-wise consolidation": we observe a sharp reduction in reasoning steps (↓62.4%) alongside a moderate increase in average step length (↑22.5%). This suggests that CRISP leverages FUSE (18.2%) and REWRITE to collapse fragmented logical chains into fewer, high-information reasoning units, streamlining the logical topology while preserving critical evidence.

Ablation Study. We evaluate the isolated contributions of the core components within CRISP on the MATH-500 benchmark across both 1.5B and 7B parameter scales. As presented in Table 3, the heuristic pruning employed by A*-Thought results in severe performance degradation. In contrast, our reward-guided search preserves the logical backbone more effectively but still suffers from linguistic fragmentation. The Output Refinement module proves critical across both model sizes as it restores semantic coherence. This process recovers accuracy to 75.0% for the 1.5B model and 84.2% for the 7B model, retaining over 95% of the original performance in both cases. Additionally, the refinement stage functions as a secondary compressor by synthesizing disjointed steps. This mechanism reduces token counts to 1813 for the 1.5B model and 1146 for the 7B model, thereby significantly boosting Token Efficiency. These consistent results confirm that coupling topological search with semantic reconstruction is indispensable for maximizing efficiency regardless of model capacity.

Impact of CRISP on Reasoning Efficiency. To empirically validate the inference efficiency of

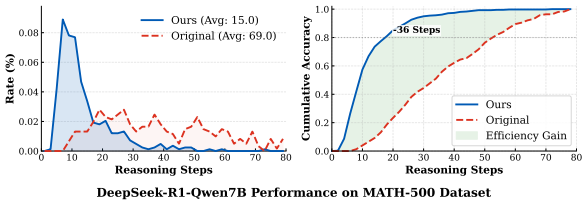

Figure 7: **Efficiency Analysis on MATH-500. (Left)** Distribution of reasoning steps for correct responses. CRISP significantly compresses the average trajectory length from 69.0 to 15.0. **(Right)** Cumulative accuracy as a function of steps. The shaded region illustrates the efficiency gain, indicating that CRISP achieves 80% accuracy using ∼36 fewer steps than the baseline.

CRISP, we analyze the reasoning trajectories of the CRISP-tuned DeepSeek-R1-Distill-Qwen7B compared to the original backbone on MATH-500. By isolating trajectories that lead to correct answers, we decouple efficiency from accuracy. As illustrated in Figure 7 (Left), our method induces a significant distributional shift toward shorter trajectories, demonstrating a tendency for more concise reasoning. Quantitatively, Figure 7 (Right) highlights that to maintain 80% accuracy, the CRISP-tuned model requires approximately 36 fewer steps than the baseline. This confirms that CRISP effectively mitigates the "overthinking" phenomenon without performance degradation. For a comprehensive analysis covering both 1.5B and 7B model scales across additional datasets, please refer to Appendix D.2.

6 Conclusion

In this work, we investigate the internal mechanics of R1-series reasoning models, identifying that the reasoning termination token (`</think>`) functions as a critical information anchor. The attention patterns at this specific position effectively delineate core reasoning trajectories from redundant cognitive computations. Leveraging this insight, we propose **CRISP**, a framework that steers thought compression via attention-guided search. By optimizing a reward objective that balances answer fidelity with token consumption, and subsequently applying generative refinement for semantic and logical repair, CRISP successfully distills high-density reasoning paths. Extensive experiments demonstrate that models fine-tuned on data synthesized using the CRISP method achieve significant reductions in response length while maintaining robust reasoning performance, offering a scalable solution for efficient LLM deployment.

Limitations

Despite the reasoning efficiency demonstrated by CRISP, we acknowledge three primary limitations in our current work. First, our evaluation focuses primarily on mathematical benchmarks characterized by structured logic. The extent to which this reward-guided topological optimization generalizes to open-ended tasks or domains with subjective criteria remains to be investigated. Second, despite the reduced inference cost, the data construction phase incurs significant computational overhead due to iterative search. This offline complexity may present a bottleneck for scaling to extensive datasets. Finally, the effectiveness of the reasoning search is contingent upon the fidelity of reward signals. In specialized or low-resource domains lacking reliable verifiers, the risk of suboptimal path selection could attenuate the quality of the compressed reasoning chains.

Acknowledgement

This research is supported by the National Natural Science Foundation of China (No.62476127), the Natural Science Foundation of Jiangsu Province (No.BK20242039), the Basic Research Program of the Bureau of Science and Technology (ILF24001), the Research Fund (No.PO250624101698), the Scientific Research Starting Foundation of Nanjing University of Aeronautics and Astronautics (No.YQR21022), and the High Performance Computing Platform of Nanjing University of Aeronautics and Astronautics.

References

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

AMC. 2025. American mathematics competitions (amc).

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. *arXiv preprint arXiv:2401.11467*.

Daewon Choi, Jimin Lee, Jihoon Tack, Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Bhavana Ganesh, Jinwoo Shin, Aram Galstyan, and Sravan Babu Bodapati. 2025. Think clearly: Improving reasoning via redundant token pruning. *arXiv preprint arXiv:2507.08806*, 4.

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*.

Yingqian Cui, Pengfei He, Jingying Zeng, Hui Liu, Xianfeng Tang, Zhenwei Dai, Yan Han, Chen Luo, Jing Huang, Zhen Li, and 1 others. 2025. Stepwise perplexity-guided refinement for efficient chain-of-thought reasoning in large language models. *arXiv preprint arXiv:2502.13260*.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

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. 2025. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24842–24855.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2024. Measuring mathematical problem solving with the math dataset, 2021. *URL https://arxiv. org/abs/2103.03874*, 2.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63.

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 *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Gengyang Li, Yifeng Gao, Yuming Li, and Yunfang Wu. 2025. Thinkless: A training-free inference-efficient method for reducing reasoning redundancy. *arXiv preprint arXiv:2505.15684*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

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 preprint arXiv:2501.12570*.

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*.

Ziqing Qiao, Yongheng Deng, Jiali Zeng, Dong Wang, Lai Wei, Guanbo Wang, Fandong Meng, Jie Zhou, Ju Ren, and Yaoxue Zhang. 2025. Concise: Confidence-guided compression in step-by-step efficient reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 8021–8040.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506.

Matthew Renze and Erhan Guven. 2024. The benefits of a concise chain of thought on problem-solving in large language models. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 476–483. IEEE.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

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*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, C Chen, C Li, C Xiao, C Du, C Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms, 2025. *URL https://arxiv.org/abs/2501.12599*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Renzhi Wang, Chongqiang Wei, Zhisheng Wang, and Piji Li. R4: Nested reasoning-retrieval for reward modeling in role-playing agents.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yibo Wang, Haotian Luo, Huanjin Yao, Tiansheng Huang, Haiying He, Rui Liu, Naiqiang Tan, Jiaxing Huang, Xiaochun Cao, Dacheng Tao, and 1 others. 2025. R1-compress: Long chain-of-thought compression via chunk compression and search. *arXiv preprint arXiv:2505.16838*.

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.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

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

Xiaoang Xu, Shuo Wang, Xu Han, Zhenghao Liu, Huijia Wu, Peipei Li, Zhiyuan Liu, Maosong Sun, and Zhaofeng He. 2025b. A*-thought: Efficient reasoning via bidirectional compression for low-resource settings. *arXiv preprint arXiv:2505.24550*.

Jianzhi Yan, Le Liu, Youcheng Pan, Shiwei Chen, Zike Yuan, Yang Xiang, and Buzhou Tang. 2025. From long to lean: Performance-aware and adaptive chain-of-thought compression via multi-round refinement. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12290–12306.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving

with large language models. *Advances in neural information processing systems*, 36:11809–11822.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, and 1 others. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Detailed Problem Setup

Given a dataset $\mathcal{D} = \{(x, y)\}$ consisting of input queries x and ground-truth answers y, a large language model $\mathcal{M}$ initially generates a verbose reasoning chain R to derive the answer. We denote the length of this reasoning chain as $|R|$.Our objective is to construct a compressed reasoning chain $\hat{R}$ derived from R, such that $|\hat{R}| < |R|$. We then fine-tune the model $\mathcal{M}$ on the compressed data $(x, \hat{R}, y)$ to internalize the efficient reasoning pattern. The optimization goal is to find the optimal parameters θ that minimize the prediction loss on the compressed chains:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(\mathcal{M}_\theta(x, \hat{R}), y)\right]$$

where $\mathcal{L}$ denotes the loss function. By solving this optimization problem, we aim to obtain a model that retains the high reasoning performance of the original verbose model while significantly reducing the computational cost in terms of token usage.

B More Implementation Details of CRISP

This appendix details the algorithmic implementation of **CRISP**. While the complete workflow encompasses the initial CoT generation and the subsequent fine-tuning phase, the core of our methodology lies in the intermediate compression algorithm. This central component operates via a two-stage pipeline. It begins with a greedy search for the critical reasoning path utilizing four atomic primitives (FUSE, PRUNE, REWRITE, and KEEP), followed by a refinement phase that synthesizes the final compressed trajectory. We present the formal algorithmic description and specific implementation nuances below.

B.1 Details of Critical Reasoning Paths Search and Compression

Formally, we model the compression process as a sequential decision-making problem. To avoid the computational intractability of an exhaustive search, we implement a Heuristic Gating Mechanism that dynamically prunes the action space $\mathcal{A}_t$ for each step r_t. This mechanism utilizes token-level attention distributions to identify low-contribution steps (triggering PRUNE/REWRITE) and employs a SimCSE (Gao et al., 2021) encoder to detect semantic redundancy (triggering FUSE). The optimal action is then determined by maximizing a joint scoring Reward:

$$\begin{aligned} R(a) = &\log P_\theta(y \mid x, \mathcal{C} \oplus a(r_i)) \\ &- \log P_\theta(y \mid x, \mathcal{C}) - \beta \cdot \text{Len}(a(r_i)) \end{aligned} \tag{8}$$

which explicitly balances the trade-off between prediction fidelity (log-likelihood gain) and efficiency (token reduction). The complete execution logic is formalized in Algorithm 1. To ensure reproducibility, we provide the specific instruction templates used for the REWRITE and FUSE operations. These prompts are designed to enforce strict conciseness constraints while preserving logical fidelity. The exact prompt specifications are detailed in Table 6.

B.2 Details of Distilled CoT Refinement

While the greedy search algorithm effectively extracts the critical reasoning backbone, the resulting compressed sequence R' frequently exhibits syntactic fragmentation and abrupt logical transitions. Direct training on such disjointed text may compromise the linguistic coherence of the reasoning model. To mitigate this, we introduce a Reference-Conditioned Restoration phase to recover logical fluency while maintaining the structural conciseness of the search result.

Formally, we employ DeepSeek-V3 (Liu et al., 2024) to synthesize a refined trajectory R_{CRISP}. We formulate this process as maximizing the conditional probability $P(R_{\text{CRISP}} \mid x, R', R)$, where the generation is conditioned on the joint context of

the query x, the distilled chain R', and the original chain R. This input configuration is critical: conditioning on R' imposes efficiency constraints, while the inclusion of R provides a semantic reference. This dual conditioning enables the model to bridge coherence gaps in R' by reconstructing necessary connective logic and arithmetic details without reverting to the original verbosity. Consequently, the restoration process ensures that R_{CRISP} maintains high reasoning fidelity with respect to the original logic while strictly adhering to the compression objectives. The specific prompt template used for this restoration process is detailed in Table 7.

B.3 Details of Multi-Task Finetuning and Inference

To achieve the multi-task fine-tuning objective, we construct training instances in two distinct formats, conditioned on a control signal κ (instantiated as `<|compressed|>`). This dual-task setup allows the model to learn the conditional distribution of compressed reasoning paths while retaining its original reasoning capabilities within a unified parameter space. Table 4 details the specific input templates used. Notably, for the compression task, the control signal is encapsulated by `[EOS]` delimiters to establish a clear boundary between the query context and the steering instruction. During inference, the model's generation behavior is modulated by the input structure: appending the control suffix triggers the compressed reasoning mode, whereas providing the raw query prompts the model to generate the original reasoning path.

Task Type	Input Prompt Construction

Standard	**Input:**
	`{Question}`
	`Please reason step by step, and put your final answer within \boxed{}.`
	↪ **Target:** `<think>` `{Original CoT}` `...`

Compressed	**Input:** `{Question}` `Please reason step by step, and put your final answer within \boxed{}`**`[EOS]<|compressed|>[EOS]`** ↪ **Target:** `<think>` `{CRISP CoT}` `...`

Table 4: **Input formatting protocols for multi-task fine-tuning.** We directly append the specific control suffix to the reasoning prompt for the compressed track, enabling the model to distinguish between the two generation modes.

C Details about Experiments

C.1 Training Datasets

To construct the training corpus, we curate a balanced subset from the training split of the MATH dataset. Adhering to the generation protocols of (Guo et al., 2025), we synthesize the initial reasoning chains using the base model with temperature $T = 0.6$ and sampling top-$p = 0.95$, utilizing the standard prompt template defined in Table 4. We enforce a correctness constraint by filtering out trajectories that fail to yield the ground-truth answer. From the valid pool, we apply stratified sampling to select 500 instances uniformly from each of the five difficulty levels, resulting in a total of 2,500 samples. These validated sequences constitute our original verbose reasoning chains, denoted as R_{orig}.

C.2 Evaluation Datasets

Our evaluation protocol encompasses three benchmarks representing a broad spectrum of reasoning complexity. We begin with GSM8K (Cobbe et al., 2021), utilizing the official test set to assess the model's proficiency in basic multi-step arithmetic and logical reasoning. To evaluate performance on more rigorous competition-level mathematics, we employ MATH-500, a representative test subset of the challenging MATH benchmark (Hendrycks et al., 2024) covering diverse subjects such as algebra, geometry, and calculus. Finally, to test the model's robustness to out-of-distribution tasks and unseen problems, we introduce AMC23 (AMC, 2025), a dataset comprised of questions from the 2023 American Mathematics Competitions (AMC 10/12). This diverse suite ensures a comprehensive assessment of the model's ability to maintain reasoning fidelity across varying difficulty levels under the compression constraints.

C.3 Implementation Details

Training Configuration. Following the data construction phase, we perform multi-task full-parameter fine-tuning on both the 1.5B and 7B variants of the DeepSeek-R1-Distill series. We utilize the **LLaMA-Factory** framework (Zheng et al., 2024) for efficient training. To accommodate memory constraints while maximizing throughput, we employ different DeepSpeed optimization strategies: **ZeRO-Stage 2** for the 1.5B model and **ZeRO-**

Stage 3 (Offload) for the 7B model (Rasley et al., 2020). We maintain a global effective batch size of 32 across all experiments using gradient accumulation. The models are trained for 3 epochs with the AdamW optimizer, a constant learning rate of 1×10^{-6}, and a maximum sequence length of 8,192 tokens. All training runs are conducted in `bfloat16` precision to ensure numerical stability.

Inference and Evaluation. For efficient evaluation, we deploy the fine-tuned models using the **vLLM** engine (Kwon et al., 2023). Following the recommended protocols from Guo et al. (2025), we adopt a sampling strategy with temperature $T = 0.6$ and nucleus sampling top-$p = 0.95$. To align with the varying complexity of the evaluation benchmarks, we set the maximum generation length to 4,096 tokens for GSM8K, and extend it to 8,192 tokens for more demanding datasets such as MATH-500, AMC23.

Hardware Infrastructure. Our experimental infrastructure comprises two distinct server nodes, equipped with 4 $\times$ NVIDIA L20 and 4 $\times$ NVIDIA RTX 3090 GPUs, respectively. All models are implemented in PyTorch.

C.4 Implementation Details of Baselines

To ensure reproducibility and fair comparison, we detail the specific input formulations and training configurations for the baseline methods. Table 5 provides a unified view of the exact input prompts used for CoD (Xu et al., 2025a), TALE (Han et al., 2025), A*Thought (Xu et al., 2025b) and TokenSkip (Xia et al., 2025). While CoD and TALE rely primarily on prompt engineering to induce brevity, TokenSkip and A*Thought involve specific training procedures, which we detail below.

Details for TokenSkip. Following the official implementation, we employ LLMLingua2 (Pan et al., 2024) to perform token-level compression on the source data. Crucially, to strictly align the experimental setting, we randomly subsample the compressed data to match the exact training set size of our method. In strict adherence to the original training protocols, we utilize Low-Rank Adaptation (Hu et al., 2022) for fine-tuning with a learning rate of 5×10^{-5}. During inference, we set the compression ratio control parameter $\gamma = 0.7$ to guide the generation length.

Details for A*Thought. For the A*Thought baseline, we adhere to the original algorithmic search framework. We utilize GPT-2 (Radford et al., 2019) as the scorer model and employ the backbone model itself as the validator. Both the training and search phases are conducted using the hyperparameters recommended in the official repository.

D Extended Experimental Results and Analysis

D.1 Performance Evaluation under Strict Resource Constraints

While previous experiments have established that the CRISP framework effectively reduces the length of Chain-of-Thought without significantly compromising model capabilities, it is crucial to investigate its viability in strictly resource-constrained scenarios. In this subsection, we focus on CRISP's performance and token efficiency under tighter computational budgets and severe generation limits.

Following the evaluation protocols established by A*Thought (Xu et al., 2025b), we assess our method on backbone models of varying scales (1.5B and 7B). We simulate low-resource environments by imposing strict maximum token generation limits of 1024, and 2048, respectively. The detailed results across various datasets are presented in 8 and 9.

The empirical results demonstrate CRISP's effectiveness across different Large Language Models. Notably, CRISP consistently achieves the highest Token Efficiency scores under all budget conditions compared to baselines. This consistent superiority strongly validates the robustness and efficacy of the CRISP method in the CoT compression process, confirming its practical utility even when deployment resources are severely limited.

D.2 Extended Analysis of Reasoning Trajectory

Complementing the discussion in Section 5.3, Figure 8 presents the comprehensive reasoning step distributions and cumulative accuracy curves across all evaluated configurations (Qwen-1.5B and Qwen-7B on both GSM8K and MATH-500).

These results corroborate the findings presented in the main text, demonstrating that the efficiency gains of CRISP are robust across different model scales and task complexities. As illustrated, the distributional shift towards shorter trajectories is consistent across all settings. Furthermore, the cumulative accuracy analysis confirms that to attain

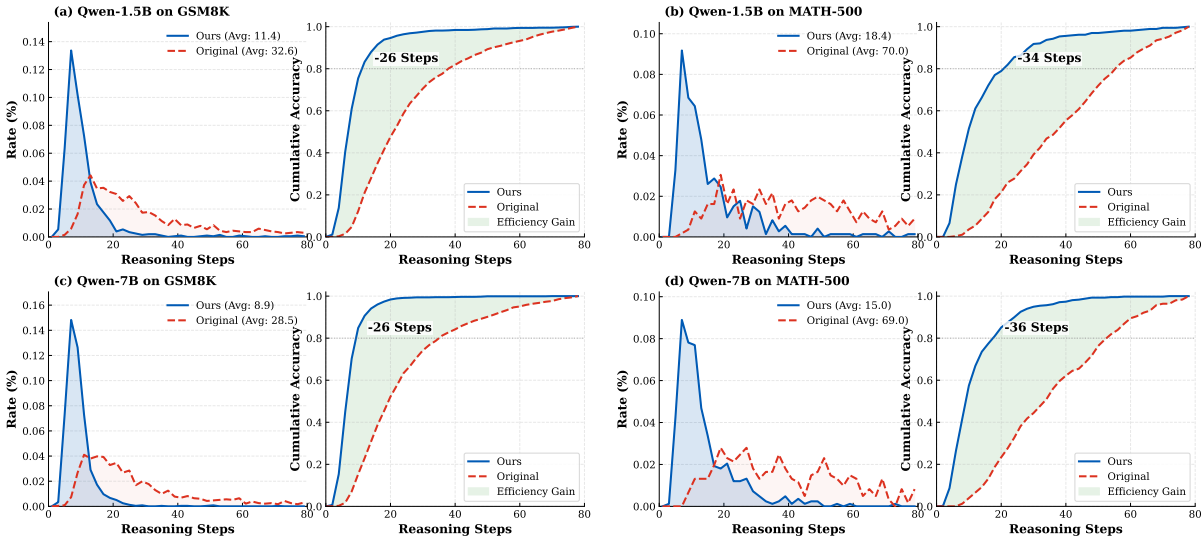

Figure 8: **Reasoning step distribution and cumulative accuracy analysis across different models and datasets.** (a)-(d) show results for Qwen-1.5B on GSM8K, Qwen-1.5B on MATH-500, Qwen-7B on GSM8K, and Qwen-7B on MATH-500, respectively. For each configuration, the left subplot displays the density distribution of reasoning steps for correct samples, while the right subplot shows the cumulative accuracy curves. The figures reveal significant distributional shifts in trajectory length, with the CRISP-tuned models requiring approximately 10-15 fewer steps to achieve 80% cumulative accuracy compared to the original models.

equivalent accuracy levels, the proposed method requires significantly fewer steps. Specifically, a reduction of approximately 25–40 steps is consistently observed across all experimental scenarios.

D.3 Case Study of CRISP

To intuitively illustrate the transformation process from the Original CoT to our Compressed CoT, we present a detailed case study in Table 10. Drawing from the GSM8K dataset, this example contrasts the raw, full-length CoT (Left) with our processed reasoning path (Right). Specifically, the left column visualizes how the four atomic operations (FUSE, PRUNE, REWRITE, KEEP) are employed to identify the key reasoning path. This is followed by a final optimization stage where an external LLM refines the semantic coherence and logical integrity of the condensed chain.

D.4 Comprehensive Visualization and Analysis of Attention Dynamics

To provide a comprehensive view of the model's internal mechanics, this section extends the analysis presented in the main text by visualizing the complete layer-wise attention maps for both the `DeepSeek-R1-Distill-Qwen-1.5B` (Figure 9) and the `7B` variant (Figure 10). These visualizations are obtained by averaging the attention scores across all heads within each layer, providing a representative view of how information is processed as it passes through the network.

Across both model scales, we observe a consistent transition in attention patterns as the depth increases. In the initial layers, attention is relatively diffuse, with the model attending broadly to the input tokens and the generated reasoning chain. This suggests that the early stages of processing are primarily concerned with local context and the immediate sequence structure.

As the hidden states progress into the middle and deeper layers, a distinct shift occurs: the attention becomes increasingly concentrated on the `</think>` token. This vertical concentration indicates that the model begins to treat the boundary token as a primary site for information aggregation. Specifically, during the generation of the final answer, the attention scores for the `</think>` position remain high, while direct attention to the preceding reasoning steps gradually diminishes. This observation holds for both the 1.5B and 7B models, effectively demonstrating that the use of the `</think>` token as a semantic anchor is a robust behavior across different parameter scales within the distilled R1 series.

Algorithm 1: Critical Reasoning Paths Search

Input : Query x, Verbose Chain $R = \{r_1, \ldots, r_N\}$, Model $\mathcal{M}$, Efficiency weight β, Thresholds τ
Output : Compressed Chain $\mathcal{C}$

// 1. Pre-computation Phase
Compute step-wise attention scores $\mathcal{S} = \{s_1, \ldots, s_N\}$ for the original chain using $\mathcal{M}$;
Initialize compressed chain $\mathcal{C} \leftarrow \emptyset$;
// 2. Sequential Greedy Search
for $t \leftarrow 1$ **to** N **do**
  Let r_t be the current step and $s_t \in \mathcal{S}$ its attention score;
  // Heuristic Gating: Determine Allowed Action Space $\mathcal{A}_t$
  if ***not*** *IsEmpty($\mathcal{C}$)* ***and*** $\texttt{Sim}(\mathcal{C}_{last}, r_t) \geq \tau_{sim}$ **then**
    $\mathcal{A}_t \leftarrow \{\textsc{Fuse}\}$; // Merge semantically similar steps
  else if $s_t < \tau_{low}$ **then**
    $\mathcal{A}_t \leftarrow \{\textsc{Prune}, \textsc{Rewrite}\}$; // Skip or Condense low-info steps
  else if $s_t < \tau_{high}$ **then**
    $\mathcal{A}_t \leftarrow \{\textsc{Rewrite}\}$; // Condense medium-info steps
  else
    $\mathcal{A}_t \leftarrow \{\textsc{Keep}, \textsc{Rewrite}\}$; // Preserve critical logic
  // Action Selection: Joint Optimization of Fidelity and Efficiency
  $a^* \leftarrow \text{argmax}_{a \in \mathcal{A}_t} \Big(\underbrace{\Delta \log P(y|x, \mathcal{C} \oplus a(r_t))}_{\text{Prediction Fidelity Gain}} - \beta \cdot \underbrace{\text{Len}(a(r_t))}_{\text{Token Cost}} \Big)$;
  // State Update
  $\mathcal{C} \leftarrow \texttt{Update}(\mathcal{C}, a^*, r_t)$;
return $\mathcal{C}$

Method	Input Prompt Construction
Standard / A*Thought	**Input:** {Question} Please reason step by step, and put your final answer within \boxed{}.
CoD (Xu et al., 2025b)	**Input:** {Question} Please reason step by step, and put your final answer within \boxed{}. Think step by step, but only keep a minimum draft for each thinking step, with 5 words at most.
TALE (Han et al., 2025)	**Input:** {Question} Let's think step by step and use less than 4096 tokens.
TokenSkip (Xia et al., 2025)	**Input:** {Question} Please reason step by step, and put your final answer within \boxed{}[EOS]{Ratio}[EOS]
CRISP (Ours)	**Input:** {Question} Please reason step by step, and put your final answer within \boxed{}**[EOS]<|compressed|>[EOS]**

Table 5: The exact input prompt templates used for the baselines and our method. Note that **TALE** replaces the standard instruction with a length-constrained prompt, whereas **CoD**, **TokenSkip**, and **CRISP** append specific constraints or control tokens to the standard instruction. The control suffix for our method is highlighted in bold.

Primitive	Prompt Template
REWRITE	**[System Message]** You are an expert at condensing reasoning steps. Your task is to rewrite the given reasoning step to be more concise while preserving all essential information and logical flow. *Rules:* 1. Keep all key facts, numbers, and logical connections. 2. Remove redundant phrases and verbose expressions. 3. Maintain the mathematical or logical correctness. 4. Output ONLY the condensed step, no explanations.
	[User Message] Compress this reasoning step as short as possible: `<step> {Original Step} </step>` Compressed:
FUSE	**[System Message]** You are an expert at merging reasoning steps. Your task is to combine two consecutive reasoning steps into a single, coherent step while preserving all essential information. *Rules:* 1. Preserve all key facts, numbers, and calculations. 2. Maintain logical flow and correctness. 3. Remove redundant information that appears in both steps. 4. The merged step should be shorter than the sum of both steps. 5. Output ONLY the merged step, no explanations.
	[User Message] Merge these two steps into one step as short as possible: Step 1: `{Step 1}` Step 2: `{Step 2}` Merged:

Table 6: **Prompt templates used for Atomic Operations.** We utilize structured system instructions to enforce strict constraints on information retention and output brevity. The `{...}` placeholders denote the dynamic input content from the reasoning chain.

Role	Content Specification
System	You are an expert mathematical editor. Your task is to refine a rough reasoning draft. Restore logical continuity and mathematical accuracy. Match the Original CoT's exact tone, formatting, and style.

User	**### Question**
	`{Input Query}`
	### Original CoT (ONLY for Reference)
	`{Original Verbose Chain}`
	### Rough Draft (To Refine)
	`{Compressed Output from Search}`
	### Instruction
	Refine the Rough Draft to ensure mathematical coherence and logical flow.
	1. Fill in missing algebraic manipulations and arithmetic calculations.
	2. Match the style and formatting of the Original CoT.
	3. Output ONLY the refined reasoning text.
	4. Ensure the calculations lead correctly to the final answer.
	### Refined Rough Solution:

Table 7: The full prompt template used in the **Reference-Conditioned Restoration** phase. The prompt is structured to condition the teacher model on three inputs: the query, the semantic reference (Original CoT), and the structural target (Distilled CoT), ensuring the output is both fluent and concise.

Method	GSM8K			MATH-500			AMC23			Average		
	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑
Model: DeepSeek-R1-Distill-Qwen-1.5B (Budget: 1024)												
Original	53.4	910	5.86	22.0	1007	2.18	10.0	1023	0.98	28.5	980	2.91
CoD	67.5	472	14.30	36.8	926	3.97	12.5	997	1.25	38.9	798	4.87
TALE	56.3	454	12.41	22.8	944	2.42	17.5	995	1.76	32.2	798	4.04
TokenSkip	54.6	822	6.64	29.6	958	3.09	10.0	1003	1.00	31.4	928	3.38
A*-Thought	63.1	638	9.89	36.2	**758**	4.78	15.0	960	1.56	38.1	785	4.85
CRISP (Ours)	**78.1**	**431**	**18.12**	**55.8**	765	**7.29**	**52.5**	**927**	**5.66**	**62.1**	**708**	**8.77**
Model: DeepSeek-R1-Distill-Qwen-7B (Budget: 1024)												
Original	58.4	897	6.51	22.8	1002	2.28	17.5	1020	1.72	32.9	973	3.38
CoD	72.8	262	27.78	46.6	775	6.01	22.5	925	2.43	47.3	654	7.23
TALE	70.7	**160**	**44.21**	34.6	684	5.06	12.5	962	1.30	39.3	**602**	6.53
TokenSkip	71.5	747	9.57	38.4	933	4.12	32.5	957	3.40	47.5	879	5.40
A*-Thought	71.6	564	12.69	41.6	**653**	6.37	17.5	**694**	2.52	43.6	637	6.84
CRISP (Ours)	**89.2**	369	24.18	**66.2**	715	**9.26**	**42.5**	873	**4.87**	**66.0**	652	**10.12**

Table 8: Performance comparison under a strict token budget of **1024 tokens**. We report Accuracy (Acc.), Average Tokens (Tok.), and Token Efficiency (TE). **Bold** and underline denote the best and second-best results, respectively.

Method	GSM8K			MATH-500			AMC23			Average		
	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑	Acc. ↑	Tok. ↓	TE. ↑
Model: DeepSeek-R1-Distill-Qwen-1.5B (Budget: 2048)												
Original	71.2	1310	5.43	45.4	1754	2.59	32.5	1913	1.70	49.7	1659	3.00
CoD	71.3	576	12.39	59.2	1468	4.03	35.0	1812	1.93	55.2	1285	4.30
TALE	64.3	645	9.97	47.4	1599	2.96	27.5	1806	1.52	46.4	1350	3.44
TokenSkip	67.4	1206	5.59	48.6	1633	2.98	27.5	1756	1.57	47.8	1532	3.12
A*-Thought	66.5	779	8.54	48.0	1102	4.36	22.5	**1309**	1.72	45.7	1063	4.30
CRISP (Ours)	**79.5**	**482**	**16.50**	**68.0**	**1057**	**6.43**	**50.0**	1311	**3.81**	**65.8**	**950**	**6.93**
Model: DeepSeek-R1-Distill-Qwen-7B (Budget: 2048)												
Original	83.6	1178	7.09	52.8	1713	3.08	40.0	1854	2.16	58.8	1582	3.72
CoD	72.0	275	26.16	61.2	1087	5.63	47.5	1575	3.02	60.2	979	6.15
TALE	69.0	**164**	**42.07**	49.8	1099	4.53	42.5	1654	2.57	53.8	972	5.53
TokenSkip	82.9	942	8.80	60.4	1437	4.20	47.5	1615	2.94	63.6	1331	4.78
A*-Thought	74.8	620	12.06	48.4	**804**	6.02	25.0	**1040**	2.40	49.4	**821**	6.02
CRISP (Ours)	**89.9**	369	24.37	**78.0**	895	**8.72**	**65.0**	1275	**5.10**	**77.6**	846	**9.17**

Table 9: Performance comparison under a token budget of **2048 tokens**. We report Accuracy (Acc.), Average Tokens (Tok.), and Token Efficiency (TE). **Bold** and underline denote the best and second-best results, respectively.

Original CoT (Annotated with Search Actions)

Question: What is the positive difference between the sum of $\frac{1}{2}$ and $\frac{1}{3}$ and the product of $\frac{1}{2}$ and $\frac{1}{3}$?

[REWRITE] Alright, so I need to find the positive difference between the sum of 1/2 and 1/3 and the product of 1/2 and 1/3. Hmm, okay, let's break this down step by step. I think I should first figure out what the sum is and then the product, and then subtract one from the other. But wait, the question says the positive difference, so maybe I need to make sure the result is positive, just in case the subtraction goes the other way. Let me write this down to visualize it better.

[REWRITE] First, let's calculate the sum of 1/2 and 1/3. I remember that to add fractions, they need to have the same denominator. The denominators here are 2 and 3. The least common denominator for 2 and 3 is 6. So I can convert both fractions to have 6 as the denominator.

[FUSE] 1/2 is equal to 3/6 because 1 times 3 is 3 and 2 times 3 is 6. Similarly, 1/3 is equal to 2/6 because 1 times 2 is 2 and 3 times 2 is 6. Now, adding these together: 3/6 + 2/6. Since the denominators are the same, I can just add the numerators. 3 plus 2 is 5, so the sum is 5/6. Got that part down.

[FUSE] Next, I need to find the product of 1/2 and 1/3. Multiplying fractions is straightforward, right? Multiply the numerators together and the denominators together. So, 1 times 1 is 1, and 2 times 3 is 6. Therefore, the product is 1/6. That seems simple enough.

[REWRITE] Now, the problem asks for the positive difference between the sum and the product. So, I think that means I need to subtract the product from the sum and then take the positive value of that result. Let me write that down: positive difference = |sum - product|.

[PRUNE] So, plugging in the values I found: sum is 5/6 and product is 1/6. Therefore, the difference is 5/6 - 1/6. Again, since the denominators are the same, I can subtract the numerators directly. 5 minus 1 is 4, so the difference is 4/6.

[PRUNE] Wait, 4/6 can be simplified, right? Both 4 and 6 are divisible by 2. So, dividing numerator and denominator by 2, we get 2/3. Therefore, the positive difference is 2/3.

[REWRITE] Let me just double-check my steps to make sure I didn't make a mistake. First, adding 1/2 and 1/3: converted to sixths, got 3/6 and 2/6, added to 5/6. Then, multiplying 1/2 and 1/3: 1/6. Then subtracting 1/6 from 5/6 gives 4/6, which simplifies to 2/3. Seems solid.

[REWRITE] Hmm, another way to think about it: maybe I should write it out as an equation. Let me define S as the sum and P as the product.

[REWRITE] S = 1/2 + 1/3
P = (1/2) * (1/3)
Difference = |S - P|

[REWRITE] Calculating S:
1/2 + 1/3 = (3/6) + (2/6) = 5/6

[FUSE] Calculating P:
(1/2)*(1/3) = 1/6

[REWRITE] Difference:
|5/6 - 1/6| = |4/6| = 4/6 = 2/3

[REWRITE] Yep, that's consistent with what I did before. So, I think 2/3 is the correct answer. Just to make sure, let me think about the operations again. Sum is bigger than the product because both fractions are less than 1, so their product will be smaller than either of them. So, subtracting the product from the sum should give a positive number, which it does, 2/3. So, that makes sense.

[FUSE] Alternatively, if I had subtracted the sum from the product, I would have gotten a negative number, but since the question asks for the positive difference, it's the absolute value, so it's 2/3 either way. But in this case, since sum is bigger, it's just 2/3.

[REWRITE] I don't think I made any calculation errors here. The steps all follow logically, and I've double-checked them. So, I'm confident that the positive difference is 2/3.

[KEEP] **Final Answer**
The positive difference is $\boxed{\frac{2}{3}}$.

Evolution of Reasoning (Compressed → Refined)

1. Compressed Output (Post-Search)
(Result of executing search actions)

Find the positive difference between (1/2 + 1/3) and (1/2 × 1/3).
To add 1/2 and 1/3, find a common denominator of 6, converting them to 3/6 and 2/6, then add the numerators to get 5/6; then find their product by multiplying the numerators (1×1=1) and denominators (2×3=6) to get 1/6.
Positive difference = |sum - product|.
Adding 1/2 and 1/3 gives 5/6. Multiplying them gives 1/6. Subtracting 1/6 from 5/6 yields 4/6, which simplifies to 2/3.
Define S as the sum and P as the product.
S = 1/2 + 1/3, P = (1/2)*(1/3), Difference = |S - P|
S = 1/2 + 1/3 = 3/6 + 2/6 = 5/6, and P = (1/2)*(1/3) = 1/6.
|5/6 - 1/6| = 4/6 = 2/3
Since the sum exceeds the product, the positive difference is 2/3.
The positive difference is 2/3.
Final Answer The positive difference is $\boxed{\frac{2}{3}}$.

↓ Refinement Step ↓

2. Final Refined CoT (Training Target)
Alright, so I need to find the positive difference between the sum of $\frac{1}{2}$ and $\frac{1}{3}$ and the product of $\frac{1}{2}$ and $\frac{1}{3}$. Let's break this down step by step.
First, I'll compute the sum:

$$\frac{1}{2} + \frac{1}{3}.$$

To add these fractions, I need a common denominator. The least common multiple of 2 and 3 is 6.

$$\frac{1}{2} = \frac{1 \times 3}{2 \times 3} = \frac{3}{6}, \quad \frac{1}{3} = \frac{1 \times 2}{3 \times 2} = \frac{2}{6}.$$

Adding them:

$$\frac{3}{6} + \frac{2}{6} = \frac{3+2}{6} = \frac{5}{6}.$$

Next, I'll compute the product:

$$\frac{1}{2} \times \frac{1}{3} = \frac{1 \times 1}{2 \times 3} = \frac{1}{6}.$$

Now, the problem asks for the positive difference between the sum and the product. That means I need

$$|\text{sum} - \text{product}|.$$

Substituting the values:

$$\left|\frac{5}{6} - \frac{1}{6}\right| = \left|\frac{4}{6}\right| = \frac{4}{6}.$$

This fraction can be simplified by dividing numerator and denominator by 2:

$$\frac{4}{6} = \frac{2}{3}.$$

Let me double-check: Since $\frac{5}{6} > \frac{1}{6}$, the difference is already positive, so the absolute value doesn't change the result. Thus, the positive difference is $\frac{2}{3}$.
Final Answer

$$\boxed{\frac{2}{3}}$$

Table 10: Case study illustrating the complete compression and refinement pipeline. The left column displays the original CoT annotated with our four discrete actions: REWRITE abstracts the core intent from conversational fillers (Step 0); FUSE synthesizes fragmented arithmetic details into a single dense step (Steps 2–3); PRUNE eliminates redundant intermediate derivations from the chain (Steps 5–6); and KEEP preserves valid original steps without modification. The right column shows the transition from the compressed (high-density but disjointed) state to the final refined CoT (coherent and structured).

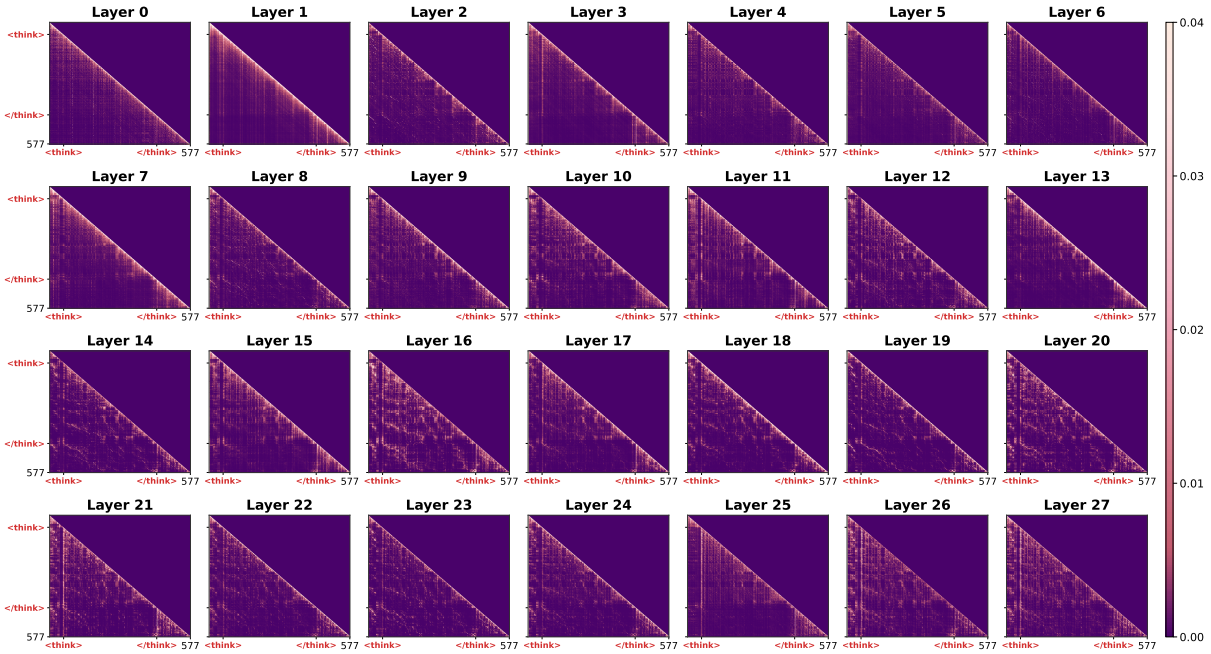

Figure 9: Full layer-wise attention maps for `DeepSeek-R1-Distill-Qwen-1.5B` on a representative GSM8K sample. Each subplot represents one layer (averaged across all heads). The prominent vertical column emerging in the middle and deep layers corresponds to the `</think>` token, acting as a site for information aggregation.

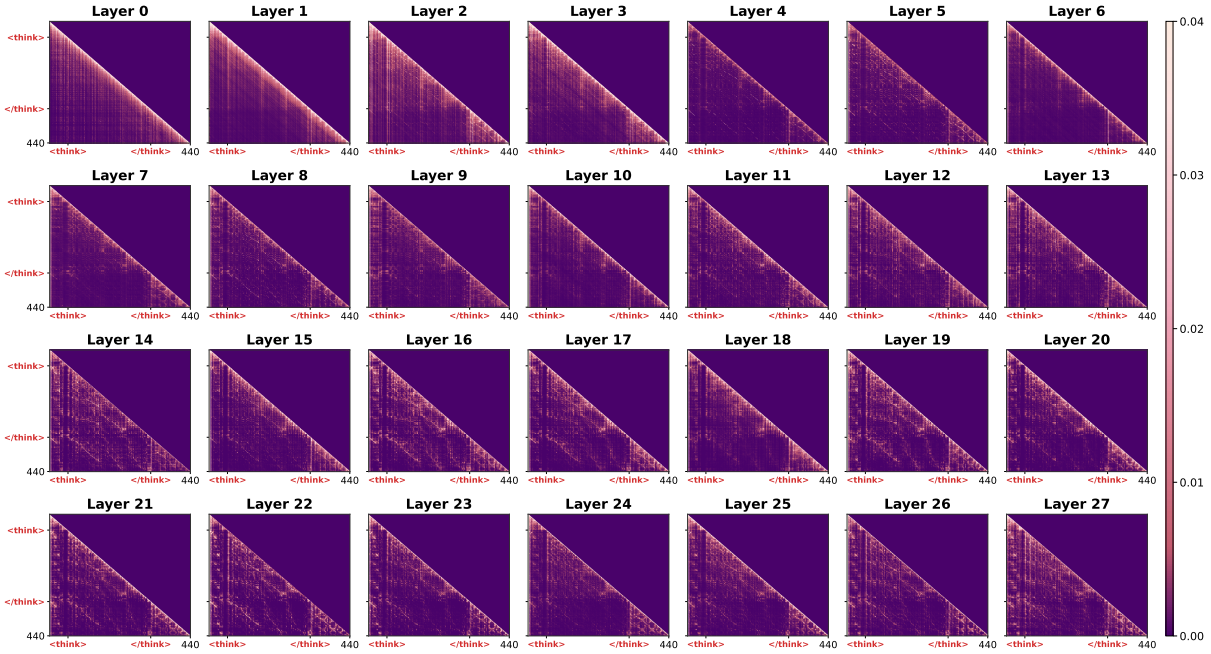

Figure 10: Full layer-wise attention maps for `DeepSeek-R1-Distill-Qwen-7B`. Similar to the 1.5B variant, the model exhibits a systematic transition from diffuse attention to concentrated attention at the `</think>` token position as depth increases, reinforcing its role as a semantic anchor for the preceding reasoning chain.