LOGIC DENSITY PRUNING: MITIGATING OVERTHINKING VIA STEP-LEVEL CHAIN OF THOUGHT COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models often improve on complex tasks by generating longer Chains-of-Thought (CoT). However, prolonged reasoning trajectories do not strictly guarantee improved utility. The phenomenon of generating redundant paraphrasing, loose explanations, and unproductive branches, which substantially inflates token usage and latency, is often termed *overthinking*. Prior work attempts to control reasoning length via prompt engineering, teacher-driven rewriting and distillation, or training-based constraints, but it commonly lacks an interpretable metric over reasoning steps that attributes each reasoning step to the final answer, making it difficult to reliably balance compression and fidelity. Compared to these methods, this paper proposes Logic Density Pruning (LDP), a CoT compression and training framework. Rather than relying on token-level importance, LDP computes step importance based on a unified criterion, *logic density*, which is constructed from two complementary scores at the step level—Mutual Information (MI) and attention citation. Based on this logic density, LDP performs reasonable CoT compression. To reduce the risk of mistakenly removing truly critical steps, this paper introduces a step perturbation check that verifies whether removing a candidate step induces a significant shift in the model's subsequent generation distribution, and selectively recovers such steps. Notably, when the compressed CoTs are applied to Qwen3-8B, reasoning tokens are reduced by 80.2% (from 295.5 to 58.6) on CommonsenseQA, with only a 1.9% performance drop being observed.

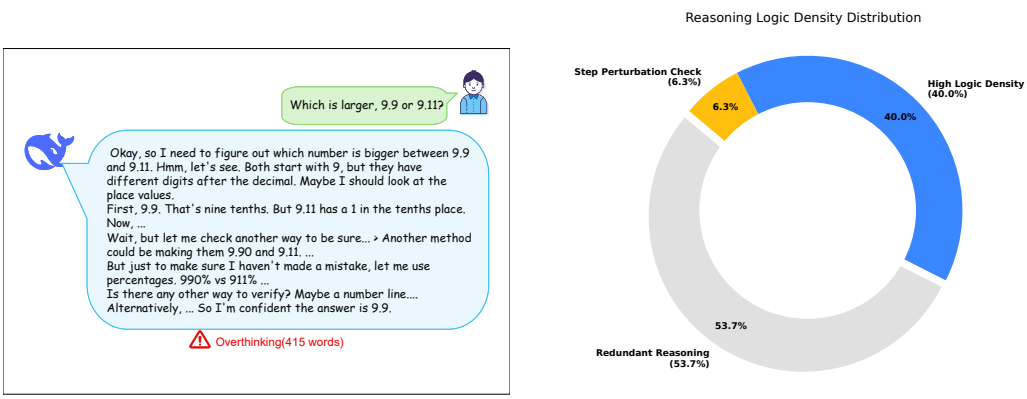
1 INTRODUCTION

Chain-of-Thought (CoT) has become a widely adopted paradigm to enhance the reasoning capabilities of large language models. Driven by instruction tuning and reinforcement learning, models have progressed from strong dialogue ability to increasingly capable reasoning behavior. By explicitly externalizing intermediate reasoning as a sequence of steps, models often achieve higher accuracy and better robustness on challenging problems involving multi-step inference, compositional decision-making, mathematics, and logic (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023; Wang & Zhou, 2024).

Despite these advancements in reasoning capabilities, an increasingly prominent phenomenon is *overthinking* (Yueh-Han et al., 2026; Sui et al., 2025; Yeo et al., 2025). During reasoning, models tend to generate excessively long CoTs containing repetition, loose explanations, or branches that are not directly useful for the final answer. Such redundancy may make the output appear more elaborate, but does not necessarily improve answer quality (Jiang et al., 2025; Boppana et al., 2026). More importantly, it is typically unclear which steps truly contribute to the final answer and which are merely apparently relevant. Without an observable and interpretable attribution score over reasoning steps, it becomes difficult to remove ineffective reasoning without harming correctness, leading to inflated token usage and latency—a direct mismatch with real-world deployment constraints on throughput, delay, and cost.

To vividly illustrate this issue, Figure 1(a) provides a typical overthinking example where the model generates excessive verification steps for a straightforward decimal comparison. Furthermore, when analyzing the reasoning trajectories generated by the Qwen3-8B model on the CommonsenseQA

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Overthinking Phenomenon

(b) Logic Density Distribution

Figure 1: *Overthinking in reasoning models and the proposed evaluation distribution.* (a) An example where the model generates excessive and redundant verification steps (e.g., percentages, number lines) for a simple decimal comparison, demonstrating typical overthinking behavior. (b) The composition of reasoning trajectories evaluated by the proposed framework. A significant portion of the steps (53.7%) are identified as redundant, while high logic density steps (40.0%) and critical steps recovered via the Step Perturbation Check (6.3%) are retained to ensure fidelity.

dataset, the statistical comparison between the original CoTs and the pruned results reveals a striking distribution. As shown in Figure 1(b), 53.7% of the generated steps are identified as redundant. Meanwhile, 40.0% are high logic density steps and 6.3% are critical steps recovered via the Step Perturbation Check. This quantitative evidence highlights the urgent need for a precise reasoning compression mechanism.

Toward the goal of reducing reasoning redundancy and mitigating overthinking, prior research can be summarized along two complementary directions—*Reasoning Process Optimization*, which constrains the generation process at inference time (e.g., via prompts) (Huang et al., 2026), and *CoT Compression and Efficiency*, which encourages models to produce shorter and more efficient CoT trajectories through training or post-processing (e.g., distillation and supervised fine-tuning) (Chen et al., 2025a; Shen et al., 2025b).

Despite their effectiveness in certain settings, these approaches often fall short on the core difficulty of *overthinking*—identifying which steps are explicitly essential. Prompt-based length control relies on implicit guidance and provides limited interpretability over reasoning steps and stability across tasks and models. Distillation typically learns surface-level sequence style rather than explicitly optimizing reasoning structure (Chen et al., 2025b), and may inherit *overthinking* patterns or oversimplification biases from the teacher. Training-based methods can shape reasoning style (Zhang et al., 2024), but commonly operate at token-level heuristics or coarse sequence-level constraints, lacking an explicit per-step criterion for pruning decisions.

To bridge this gap, this paper proposes Logic Density Pruning (LDP). Rather than relying on token-level importance, this method integrates an MI Score and attention citation to construct the *logic density* criterion. This provides an interpretable score to determine which unimportant steps to prune and which critical steps to retain. Furthermore, a Step Perturbation Check (SPC) compares the hidden-state representations of greedy continuations generated with and without a candidate step and selectively recovers steps whose representation change exceeds a threshold, reducing the risk of mistakenly deleting essential reasoning and achieving a more robust trade-off between compression and fidelity. By organizing the related work under the aforementioned taxonomy, this paper enables a direct and clear comparison between these existing macroscopic approaches and the proposed step-level, interpretable framework.

The contributions of this paper are summarized as follows. (1) *Logic density criterion.* This paper proposes an interpretable step-importance criterion that integrates an MI Score and attention citation, quantifying each step's contribution to the final answer and supporting principled pruning decisions. (2) *LDP framework.* This paper presents an interpretable procedure over reasoning steps. Intra-sample

adaptive thresholding is applied based on logic density, and SPC reduces the risk of deleting critical steps, achieving a robust trade-off between compression and fidelity. The resulting high-density CoTs are then used as improved supervision to mitigate overthinking. (3) *Empirical evaluation.* Experiments on CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021) across multiple model scales show that LDP substantially reduces reasoning tokens (about 22%–80%) while keeping accuracy changes small (about 0.9–2.3 points, with improvements in some settings).

2 Related Work

Prior work on reasoning efficiency follows two main directions: inference-time control of reasoning generation and training- or post-processing-based CoT compression. Inference-time methods use prompting, adaptive stopping, or structured search, whereas compression methods rely on distillation, token/sequence pruning, step-level scores, reinforcement learning, or latent reasoning. These approaches improve efficiency but generally lack an interpretable criterion that attributes each complete reasoning step to the final answer. A detailed discussion of these categories and representative studies is provided in Appendix A.

3 Motivation

Observation 1 (token-level deletion can break semantic integrity). Most existing CoT compression methods score and delete content at the token level, but this paradigm overlooks the basic structural units of natural-language reasoning. Reasoning is typically composed of semantically coherent sentences or clauses, often acting as shared decision pivots (Cho et al., 2025). Independent token operations can break syntactic dependencies and coreference, leading to semantic fragmentation. Although the resulting CoTs may be shorter, their readability and verifiability as reasoning evidence—as well as their effectiveness as supervision scores—can be weakened.

Observation 2 (MI-only scoring suffers from position bias). Recent studies analyzing mutual information in reasoning (Qian et al., 2026) provide a direct and general principle that if a step contributes to the final answer, the two should exhibit stronger statistical dependence. However, MI-based scoring can suffer from a notable position bias. Due to the accumulative nature of generation, steps near the final answer (often summarizing statements) may be systematically assigned higher importance, overestimating late steps and underestimating intermediate reasoning, thereby reducing the ability to detect redundancy.

These two observations motivate operations over reasoning steps and the need for complementary scores beyond MI alone, which this paper formalizes in the next section.

4 Method

This section presents the proposed LDP framework. As illustrated in Figure 2, the overall procedure consists of three main stages. The first stage is CoT Generation, which involves generating and collecting reasoning trajectories. The second stage is Logic Density Pruning, where the collected CoTs are compressed based on the logic density scores and SPC. The third stage is Supervised Fine-Tuning (SFT), which utilizes the pruned CoTs to fine-tune the model. The remainder of this section details these components. Section 4.1 describes CoT generation. Section 4.2 presents the logic density pruning module. Section 4.3 explains how the compressed CoTs are used for SFT.

4.1 CoT Generation

To make the subsequent pruning module compatible across models and datasets, this paper adopts a unified output template. The model generates step-structured reasoning within `<think>` and outputs the final answer with an explicit `Final Answer` marker. This formatting significantly reduces step-parsing failures and provides clear boundaries for operations over reasoning steps.

In addition, this paper uses self-consistency sampling (Wang et al., 2022) to generate multiple candidate reasoning traces per question, and selects a representative candidate with a length-regularized policy. If any candidate is correct, the shortest correct reasoning is selected to reduce redundancy.

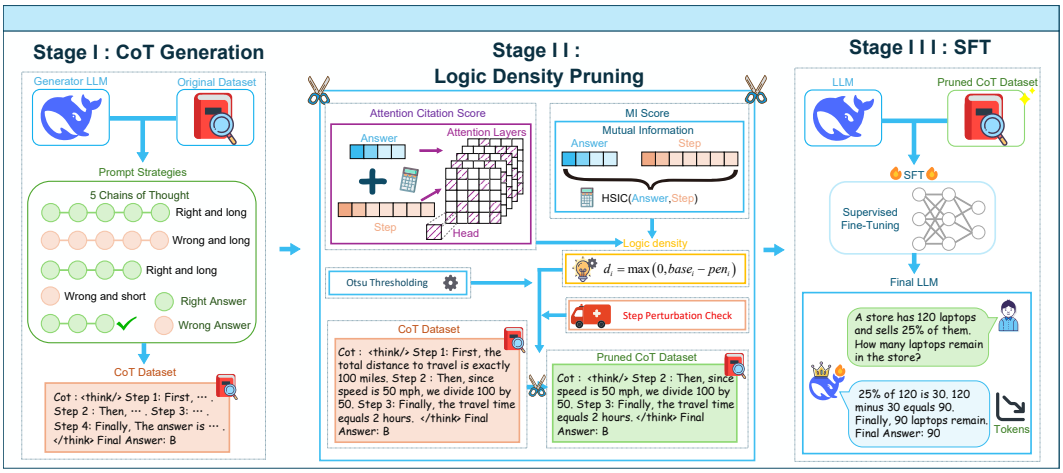


Figure 2: *LDP framework.* The overall procedure is divided into three stages. Stage I CoT Generation employs prompt strategies on a generator LLM to collect reasoning trajectories and construct an initial CoT dataset. Stage II Logic Density Pruning evaluates the steps using Attention Citation and an MI Score to compute a logic density score, applying Otsu thresholding and a Step Perturbation Check to compress the trajectories into a pruned CoT dataset. Stage III SFT utilizes this pruned dataset for supervised fine-tuning to train the final efficient LLM.

Otherwise, the shortest parseable candidate is selected to ensure the pipeline can proceed. The goal is not to maximize accuracy, but to provide a controlled starting point for evaluating whether compression can preserve answers under a smaller reasoning budget.

4.2 Logic Density Pruning

4.2.1 Step Extraction

The goal of step extraction is to normalize raw reasoning text into actionable step units. This paper prioritizes splitting by explicit numbering patterns (e.g., `Step k, 1), (k)`). When numbering is absent, a fallback sentence-level segmentation based on punctuation and discourse markers is used to approximate step boundaries. This design balances robustness and interpretability. Ideally, each s_i corresponds to a semantically coherent reasoning action. Under fallback segmentation, a stable step sequence is still obtained at low engineering cost for subsequent scoring and selection.

4.2.2 MI Score

The intuition behind the MI Score is that if a step makes a substantive contribution to the final answer (Qian et al., 2026), its representation should exhibit stronger statistical dependence with the answer representation. We emphasize that the implemented MI Score is a kernel-based dependence measure, not an exact mutual-information estimate or a standard biased or unbiased HSIC estimator.

For each reasoning step s_i, the text is independently passed through a scorer model. Token representations are extracted from the layer specified by `layer_idx`. The ground-truth answer is processed by an independent forward pass, and its token representations are taken from `output_hidden_states[-1]`. Thus, the step and answer representations are not guaranteed to come from exactly the same layer. The token hidden states on both sides are mean-pooled along the token dimension. If the hidden dimension exceeds 512, uniformly spaced hidden coordinates are retained so that at most 512 coordinates remain. Let $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}^d$ denote the resulting pooled step and answer vectors, respectively, where $d \leq 512$; in the implementation, each vector is reshaped to $[d, 1]$, and each hidden coordinate j defines a paired observation (x_{ij}, y_{ij}).

The implementation uses a Gaussian kernel with bandwidth $\sigma = 50$. Because each observation is scalar, the kernel is

$$K_{uv} = \exp\left(-\frac{(x_u - x_v)^2}{2\sigma^2}\right). \tag{1}$$

For each side, let K_x and K_y denote the corresponding kernel matrices and let

$$C = I_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top. \tag{2}$$

Here I_d is the $d \times d$ identity matrix and $\mathbf{1}_d \in \mathbb{R}^d$ is the all-ones vector. The code first computes $K_x^c = K_x C$ and $K_y^c = K_y C$, then forms

$$R_x = K_x^c \left(K_x^c + \epsilon d I\right)^{-1}, \qquad R_y = K_y^c \left(K_y^c + \epsilon d I\right)^{-1}, \tag{3}$$

where $\epsilon = 10^{-5}$. The resulting per-step score is

$$m_i = \mathrm{tr}(R_x R_y). \tag{4}$$

This statistic is not guaranteed to be nonnegative or normalized to $[0, 1]$. Before the subsequent score normalization, negative values of the raw proxy are clipped to zero. The revised description therefore specifies the observation axis, representation extraction, pooling, dimension reduction, kernel, bandwidth, and regularization explicitly; estimators based on unambiguous token-level paired observations are treated as a separate follow-up experiment rather than as a retrospective reinterpretation of the current results.

It is important to note that the MI Score m_i captures statistical relatedness rather than whether a step is truly used in the generation process. A step can be highly related to the answer yet play little causal role, especially when the reasoning contains paraphrases, rewrites, or late-stage summaries. Moreover, MI Scores may exhibit position bias, systematically overestimating steps near the end. These issues suggest that pruning based on m_i alone can mistakenly keep or delete steps, motivating an additional mechanistic score.

4.2.3 Attention Citation Score

Unlike the MI Score, the attention citation score aims to capture a mechanistic reference relationship. When generating final answer tokens, if the model's attention significantly points to tokens belonging to a step, that step is more likely to be explicitly used. This paper summarizes this phenomenon as a per-step citation rate $a_i \in [0, 1]$, estimated robustly via cross-layer and cross-head voting or proportions rather than directly summing attention magnitudes (which can be dominated by extreme heads).

When attention is unavailable (or degenerates to all zeros) in a given model or inference framework, the method naturally falls back to an MI-Score-only variant, preserving portability. The citation rate is defined by the following expression.

$$a_i \triangleq \mathbb{E}_{\ell,h}\big[\mathbf{1}\{\text{Answer attends to Step } i \text{ above } \tau\}\big]. \tag{5}$$

Here $a_i \in [0, 1]$ is the citation rate of step i, ℓ and h index Transformer layers and attention heads, respectively, and τ is the attention threshold for declaring a citation event.

4.2.4 Logic Density

Based on the two complementary scores above, this paper constructs the logic density score d_i as the final step-importance measure. Intuitively, the proxy m_i reflects representation dependence between a step and the answer, while attention citation rate a_i reflects whether the step is explicitly referenced in the generation mechanism. Before fusion, each raw score is clipped to be nonnegative, transformed with $\log(1 + 100x)$, clipped at its 95th percentile, and min–max normalized. We denote the resulting normalized scores by $\tilde{m}_i$ and $\tilde{a}_i$.

This paper introduces Vital Information Preservation Gating (VIP gating). When $\tilde{a}_i$ is high, the step is more strongly favored to be kept.

$$w_i = \sigma(\kappa(\tilde{a}_i - \tau_{\mathrm{vip}})). \tag{6}$$

This paper then defines a base score that mixes a VIP branch and a standard fusion branch.

$$b_i = w_i(1 + \tilde{a}_i) + (1 - w_i)\big(\alpha\tilde{m}_i + (1 - \alpha)\tilde{a}_i\big). \tag{7}$$

To more aggressively remove steps that are neither informative nor cited, this paper applies a temperature-controlled smooth penalty to scores below thresholds:

$$p_i = \lambda \left[\mathrm{sp}_\kappa(\tau_a - \tilde{a}_i) + \mathrm{sp}_\kappa(\tau_m - \tilde{m}_i) \right], \qquad d_i = \max(0, b_i - p_i), \tag{8}$$

where

$$\mathrm{sp}_\kappa(z) = \frac{1}{\kappa} \log(1 + \exp(\kappa z)) \,. \tag{9}$$

Here λ is the outer penalty weight, while κ controls both the softplus temperature and the VIP-gate steepness. The sigmoid is $\sigma(z) = 1/(1 + e^{-z})$.

The implementation uses PyTorch's numerically stable `F.softplus(z, beta=180, threshold=20)`, rather than an explicit ReLU. When $\kappa z > 20$, this implementation returns the linear value z; otherwise it evaluates the temperature-controlled softplus above. It is therefore a high-temperature smooth-ReLU with a linear asymptotic branch. With $\alpha = 0.6$, $\lambda = 15$, $\kappa = 180$, $\tau_a = 0.5$, $\tau_m = 0.45$, and $\tau_{\mathrm{vip}} = 0.9$, the penalty at $(\tilde{a}_i, \tilde{m}_i) = (1, 1)$ is approximately 6.8×10^{-41} and does not collapse all logic-density scores to zero.

Here $d_i \geq 0$ is the final logic density score for step i, $\alpha \in [0, 1]$ is the fusion weight on the MI Score m_i, $\lambda \geq 0$ controls the outer penalty strength, $\kappa > 0$ controls the softplus temperature and VIP-gate steepness, τ_a and τ_m are the attention and MI Score thresholds, and τ_{vip} is the high-citation threshold.

When attention is unavailable or all normalized attention scores are zero, the implementation uses the following MI-Score-only fallback:

$$d_i = \max\left(0, \tilde{m}_i - \lambda \, \mathrm{sp}_\kappa(\tau_m - \tilde{m}_i)\right). \tag{10}$$

4.2.5 Otsu Thresholding

After obtaining $\{d_i\}$, a log-robust normalization is applied before thresholding. A discrete keep/delete decision is then required. Instead of using a fixed threshold, this paper adopts Otsu adaptive thresholding to automatically select a threshold t^* from the score distribution within each CoT. Given t^*, the kept index set is determined by the following relation.

$$\widetilde{I} = \{ i \mid d_i \geq t^* \}. \tag{11}$$

The kept subsequence is then formed in the original order as $\widetilde{S} = (s_i)_{i \in \widetilde{I}}$. If all final scores on a trajectory are equal or zero, Otsu returns $t^* = 0$ and the rule $d_i \geq t^*$ retains all steps rather than deleting the entire trajectory. This design mitigates the instability of fixed thresholds that may over-delete or barely delete depending on the sample.

4.2.6 Step Perturbation Check (SPC)

Although the fused score and adaptive thresholding can remove substantial redundancy, mistakenly deleting critical steps remains possible. This paper therefore introduces *Step Perturbation Check (SPC)*. For each deleted candidate step, the implementation compares deterministic continuations generated from the original prefix with and without that step. Let $S_{<i} = (s_1, \ldots, s_{i-1})$ be the original prefix before step s_i, and let

$$c_i^+ = \mathrm{Prompt}(x, S_{<i} \oplus s_i), \qquad c_i^- = \mathrm{Prompt}(x, S_{<i}) \tag{12}$$

denote the two corresponding inputs. Greedy decoding, truncated to the first generated sentence, produces

$$y_i^+ = G_{\mathrm{greedy}}(c_i^+), \qquad y_i^- = G_{\mathrm{greedy}}(c_i^-). \tag{13}$$

Let $\phi(y)$ denote the final-layer hidden state of the last non-padding token of generated continuation y. The SPC score implemented in code is

$$D_i^{\mathrm{SPC}} = \begin{cases} 1 - \dfrac{\phi(y_i^+)^\top \phi(y_i^-)}{\|\phi(y_i^+)\|_2 \, \|\phi(y_i^-)\|_2}, & y_i^+ \neq \varnothing \text{ and } y_i^- \neq \varnothing, \\ 1, & \text{exactly one of } y_i^+, y_i^- \text{ is nonempty}, \\ 0, & y_i^+ = y_i^- = \varnothing. \end{cases} \tag{14}$$

When $D_i^{\mathrm{SPC}} > \delta$, the step is recovered and added back to the kept set.

4.3 SFT

After CoT compression, the kept subsequence $\widetilde{S}$ is treated as the compressed CoT and used to construct supervision for fine-tuning. Specifically, this paper keeps the input question x and the final answer y unchanged, and replaces only the reasoning sequence with $\widetilde{S}$ before applying a standard SFT objective. This directly encourages the model to produce shorter, higher-density intermediate reasoning while maintaining alignment to the final answer.

5 EXPERIMENTS

Experiments are conducted to evaluate the effectiveness of the Logic Density Pruning (LDP) framework across various model architectures and reasoning tasks. The evaluation aims to answer three key questions. (1) How effectively does LDP reduce token consumption while preserving or even enhancing task accuracy? (2) How does performance vary when compared to existing token- and sequence-level pruning methods? (3) How do the individual components (the MI Score and Attention) and linguistic patterns contribute to mitigating overthinking?

5.1 EXPERIMENTAL SETUP

Datasets and models. We evaluate LDP on commonsense, strategy, and mathematical reasoning benchmarks using four representative language models. Detailed dataset and model descriptions are provided in Appendix B.

Training and pruning details. All models are fine-tuned with LoRA-based SFT and evaluated with a consistent decoding protocol. Complete training and pruning hyperparameters, including the per-run settings, are reported in Appendix C.

Baselines and evaluation. Two categories of baselines are established. The first is the *Unpruned Baseline*, which represents the base model fine-tuned using the unmodified, full-length Chain-of-Thought (CoT) trajectories. The second compares LDP against TokenSkip, SBT, and StepEntropy (Li et al., 2026), representing token-, sequence-, and step-level compression, respectively. For inference, vLLM is used for high-throughput decoding, and greedy decoding is applied for final evaluation.

Table 1: *Main Results.* Accuracy (Acc) and average CoT tokens (#Tok) are reported across different reasoning datasets. Best results achieved by LDP are highlighted in **bold**.

Base Model	*Method*	*CommonsenseQA*		*StrategyQA*		*GSM8K*	
		Acc	#Tok	Acc	#Tok	Acc	#Tok
DeepscaleR-1.5B-Preview	Baseline	40.80	253.8	51.53	280.6	80.40	593.9
	LDP (Ours)	**44.30**	**129.0**	49.20	**183.4**	80.40	**422.9**
DeepSeek-R1-Distill-Llama-8B	Baseline	66.70	369.5	60.99	214.7	68.60	524.2
	LDP (Ours)	65.00	**159.9**	**62.15**	**150.9**	68.00	**444.7**
Qwen3-8B	Baseline	80.10	295.5	68.56	157.6	87.20	368.8
	LDP (Ours)	78.20	**58.6**	66.96	**86.9**	83.60	**287.5**
Qwen3-14B	Baseline	85.00	665.8	73.80	337.2	86.60	626.1
	LDP (Ours)	83.10	**183.6**	72.93	**131.2**	**87.20**	**308.9**

Comparison with existing methods. Table 2 compares LDP with token-level TokenSkip, sequence-level SBT, and the step-level StepEntropy baseline across all four models. Token-level deletion methods such as TokenSkip are agnostic to semantic boundaries, while sequence-level methods such as SBT do not provide fine-grained step attribution. StepEntropy offers a direct step-level comparison. LDP explicitly scores and selects intact, semantically coherent steps based on logic density.

5.2 ABLATION STUDY

To isolate the effect of the scoring rule from the amount of retained text, we replace the previous ablation with a matched token-budget experiment on Qwen3-8B. For each original reasoning trajectory

Table 2: *Methodological Comparison.* Accuracy and average CoT tokens for LDP, token-level TokenSkip, sequence-level SBT, and step-level StepEntropy across four models. Improvements or non-degradation of LDP (Ours) relative to the corresponding Baseline are highlighted in **bold**.

Base Model	*Method*	*CommonsenseQA*		*GSM8K*	
		Acc (%)	*#Tok*	*Acc (%)*	*#Tok*
DeepScaleR-1.5B-Preview	Baseline	40.80	253.80	80.40	593.90
	LDP (Ours)	**44.30**	**129.00**	**80.40**	**422.90**
	TokenSkip	23.20	207.41	76.60	554.61
	SBT	24.80	125.00	74.40	514.44
	StepEntropy	40.10	75.30	70.60	396.60
DeepSeek-R1-Distill-LLaMA-8B	Baseline	66.70	369.50	68.60	524.20
	LDP (Ours)	65.00	**159.90**	68.00	**444.70**
	TokenSkip	67.10	204.90	63.40	478.26
	SBT	56.00	432.18	70.00	327.06
	StepEntropy	64.30	74.66	59.40	284.60
Qwen3-8B	Baseline	80.10	295.50	87.20	368.80
	LDP (Ours)	78.20	**58.60**	83.60	**287.50**
	TokenSkip	78.90	294.92	79.60	348.80
	SBT	79.80	220.70	45.00	78.40
	StepEntropy	80.40	96.40	77.40	152.10
Qwen3-14B	Baseline	85.00	665.80	86.60	626.10
	LDP (Ours)	83.10	**183.60**	**87.20**	**308.90**
	TokenSkip	84.80	402.00	87.60	612.45
	SBT	47.00	754.70	95.00	1218.64
	StepEntropy	84.00	159.60	88.00	709.85

i, let L_i be its number of reasoning tokens under the same Qwen3 tokenizer and define the budget

$$B_i = \min(128, L_i). \tag{15}$$

We evaluate three scoring rules under the same recovery procedure: an MI-only scoring rule followed by SPC (MI + SPC), an Attention-only scoring rule followed by SPC (Attention + SPC), and the fused scoring rule followed by SPC (Fusion + SPC). Thus, SPC is applied to all three variants; it is not reserved for the fused model. All variants use exactly the same step boundaries and candidate steps. Because steps are kept as complete semantic units, the selected steps are ordered as in the original trajectory and their total token count is constrained not to exceed the same per-sample budget B_i.

On CommonsenseQA, all three variants retain exactly 24,561 reasoning tokens in total, and their final supervision targets each contain 26,690 tokens. On StrategyQA, step indivisibility leads to a maximum difference of 28 reasoning tokens across variants, which is approximately 0.13% of the roughly 21,700-token total; the final supervision-token totals differ by only the same small order of magnitude. Thus, the comparison is effectively matched in token budget rather than in a fixed number of steps.

Table 3 reports accuracy and the actual average output length for each variant. Fusion + SPC improves accuracy over both single-score + SPC variants under the same per-sample budget: by 1.20 and 1.90 percentage points over MI + SPC and Attention + SPC, respectively, on CommonsenseQA, and by 1.02 and 2.62 points over MI + SPC and Attention + SPC, respectively, on StrategyQA. The result separates the effect of the scoring rule from simply retaining more tokens; the small differences in realized average length arise from the indivisibility of complete semantic steps.

Supplementary analysis. Additional word-frequency analysis of overthinking markers is provided in Appendix D.

Table 3: *Matched token-budget ablation on Qwen3-8B.* MI + SPC, Attention + SPC, and Fusion + SPC use the same per-sample reasoning-token budget $B_i = \min(128, L_i)$, the same candidate steps, and the same SPC recovery procedure. MI + SPC and Attention + SPC use only the corresponding single scoring branch before SPC. Accuracy and actual average output tokens are reported. Best results are highlighted in **bold**.

Dataset	*Method*	*Accuracy*	*Avg. Output Tokens*
CommonsenseQA	MI + SPC	80.40	132.8
	Attention + SPC	79.70	137.5
	Fusion + SPC	**81.60**	**119.4**
StrategyQA	MI + SPC	69.58	**122.5**
	Attention + SPC	67.98	138.2
	Fusion + SPC	**70.60**	144.3

6 Conclusion

This paper presented Logic Density Pruning (LDP), a step-level Chain-of-Thought compression framework designed to mitigate the overthinking phenomenon in large reasoning models. By integrating the MI Score and attention citation into a unified logic density score, combined with Otsu adaptive thresholding and a Step Perturbation Check, LDP effectively identifies and preserves explicitly essential reasoning steps while pruning unproductive redundancies. Extensive experiments demonstrate that LDP can substantially reduce token consumption by up to 80% while maintaining highly competitive task accuracy.

Despite its effectiveness, the current framework has limitations that present opportunities for future work. Logic density is currently formulated as a relative metric within a single reasoning trajectory rather than an absolute measure. Consequently, a retained step or sentence might still exhibit intra-step variance, containing both high and low logic density components internally. Future research will explore more granular, sub-step boundary detection to address this variance and achieve even more precise reasoning compression.

AI USE STATEMENT

In preparing this manuscript, we used generative AI tools to assist with language editing, organization of the presentation, and LaTeX troubleshooting. Generative AI tools were not used to generate the experimental data, run experiments, determine evaluation results, or establish the scientific claims. All AI-assisted suggestions were reviewed and revised by the authors, and the authors take full responsibility for the final text, methodology, analysis, results, and any artifacts presented in this work.

ETHICS STATEMENT

This work does not involve human subjects, human-subject experiments, personally identifiable information, or the collection of new sensitive data. The experiments use publicly available benchmark datasets and pretrained models; we follow their stated terms of use and acknowledge their limitations, including possible dataset bias and uneven coverage of social and cultural perspectives. The proposed method is intended to reduce redundant reasoning and computational cost, and we do not identify an intended harmful application. We nevertheless recognize that compressed reasoning outputs may inherit biases or errors from the underlying models and datasets, so results should be interpreted with appropriate care.

REPRODUCIBILITY STATEMENT

The paper describes the Logic Density Pruning algorithm, its scoring and recovery procedures, and the experimental protocol in the main text. The datasets, pretrained models, evaluation tasks, and training

and pruning hyperparameters are specified in Appendix B and Appendix C; the supplementary analysis and additional experimental details are provided in the remaining appendix sections. These descriptions, together with the equations, tables, and cited benchmark definitions in the paper, are intended to make the reported experiments independently reproducible. Any released implementation and configuration files should be used together with these specifications to reproduce the results.

REFERENCES

Siddharth Boppana, Annabel Ma, Max Loeffler, Raphael Sarfati, Eric Bigelow, Atticus Geiger, Owen Lewis, and Jack Merullo. Reasoning theater: Disentangling model beliefs from chain-of-thought. *arXiv preprint arXiv:2603.05488*, 2026.

Xiaoshu Chen, Sihang Zhou, Ke Liang, Xiaoyu Sun, and Xinwang Liu. Skip-thinking: Chunk-wise chain-of-thought distillation enable smaller language models to reason better and faster. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12153–12168, 2025a.

Xinghao Chen, Zhijing Sun, Guo Wenjin, Miaoran Zhang, Yanjun Chen, Yirong Sun, Hui Su, Yijie Pan, Dietrich Klakow, Wenjie Li, et al. Unveiling the key factors for distilling chain-of-thought reasoning. In *Findings of the association for computational linguistics: ACL 2025*, pp. 15094–15119, 2025b.

Dongkyu Cho, Amy BZ Zhang, Bilel Fehri, Sheng Wang, Rumi Chunara, Hengrui Cai, and Rui Song. Correct reasoning paths visit shared decision pivots. *arXiv preprint arXiv:2509.21549*, 2025.

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

Tao Fan, Guoqiang Ma, Yuanfeng Song, Lixin Fan, and Qiang Yang. Ppc-gpt: federated task-specific compression of large language models via pruning and chain-of-thought distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14794–14805, 2025.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

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

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

Xingshuai Huang, Derek Li, Bahareh Nikpour, and Parsa Omidi. Hierarchical chain-of-thought prompting: Enhancing llm reasoning performance and efficiency. *arXiv e-prints*, pp. arXiv–2604, 2026.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, et al. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency. 2025.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think. *arXiv preprint arXiv:2504.16828*, 2025.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. Making slow thinking faster: Compressing llm chain-of-thought via step entropy. In *International Conference on Learning Representations*, volume 2026, pp. 38331–38349, 2026.

Ryan Lucas, Kayhan Behdin, Zhipeng Wang, Qingquan Song, Rahul Mazumder, et al. Reasoning models can be accurately pruned via chain-of-thought reconstruction. 2026:142840–142852, 2026.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 3(5), 2025.

Yansong Ning, Wei Li, Jun Fang, Naiqiang Tan, and Hao Liu. Not all thoughts are generated equal: Efficient llm reasoning via multi-turn reinforcement learning. *arXiv preprint arXiv:2505.11827*, 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 31932–31951, 2024.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *Advances in Neural Information Processing Systems*, 38:12533–12572, 2026.

Keshav Ramji, Tahira Naseem, and Ramón Fernandez Astudillo. Thinking without words: Efficient latent reasoning with abstract chain-of-thought. *arXiv preprint arXiv:2604.22709*, 2026.

Xuan Shen, Yizhou Wang, Yufa Zhou, Xiangxi Shi, Pu Zhao, Yanzhi Wang, and Jiuxiang Gu. Efficient reasoning with hidden thinking. *arXiv preprint arXiv:2501.19201*, 2025a.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025b.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, 2019.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Advances in Neural Information Processing Systems*, 38:4646–4668, 2026.

Jiecong Wang, Hao Peng, and Chunyang Liu. Latent chain-of-thought as planning: Decoupling reasoning from verbalization. *arXiv preprint arXiv:2601.21358*, 2026a.

Xuezhi Wang and Denny Zhou. Chain-of-thought reasoning without prompting. *Advances in Neural Information Processing Systems*, 37:66383–66409, 2024.

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

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

Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Zheng Wei, and Yang Tang. Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning. pp. 45236–45253, 2026b.

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

Junda Wu, Xintong Li, Ruoyu Wang, Yu Xia, Yuxin Xiong, Jianing Wang, Tong Yu, Xiang Chen, Branislav Kveton, Lina Yao, et al. Ocean: Offline chain-of-thought evaluation and alignment in large language models. 2025:100570–100589, 2025.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Chen Yueh-Han, Robert McCarthy, Bruce W Lee, He He, Ian Kivlichan, Bowen Baker, Micah Carroll, and Tomek Korbak. Reasoning models struggle to control their chains of thought. *arXiv preprint arXiv:2603.05706*, 2026.

Nan Zhang, Eugene Kwek, Yusen Zhang, Hieu Nguyen, Prasenjit Mitra, and Rui Zhang. When reasoning meets compression: Understanding the effects of llms compression on large reasoning models. 2026:62680–62706, 2026a.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 37:333–356, 2024.

Yuxiang Zhang, Zhengxu Yu, Weihang Pan, Zhongming Jin, Qiang Fu, Deng Cai, Binbin Lin, and Jieping Ye. Tokensqueeze: Performance-preserving compression for reasoning llms. *Advances in Neural Information Processing Systems*, 38:91516–91539, 2026b.

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

Yuqi Zhu, Jintian Zhang, Zhenjie Wan, Yujie Luo, Shuofei Qiao, Zhengke Gui, Da Zheng, Lei Liang, Huajun Chen, and Ningyu Zhang. Lightthinker++: From reasoning compression to memory management. *arXiv preprint arXiv:2604.03679*, 2026.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. Reasonflux-prm: Trajectory-aware prms for long chain-of-thought reasoning in llms. *Advances in Neural Information Processing Systems*, 38:26764–26802, 2026.

A Extended Related Work

This section reviews prior efforts on controlling reasoning length and improving reasoning efficiency. The discussion is organized into two parts aligned with the taxonomy introduced in the introduction. The first part covers inference-time control of the reasoning process. The second part discusses training- or post-processing-based compression and efficiency.

A.1 Reasoning Process Optimization

This direction intervenes in the generation process at inference time, and can be broadly grouped into three types. Prompt-based guidance (e.g., CoT prompting and self-consistency) (Wei et al., 2022; Wang et al., 2022; Huang et al., 2026) induces more structured reasoning. While simple and broadly applicable, it relies on implicit control and does not identify which steps are truly necessary, limiting interpretability over reasoning steps. Early stopping and adaptive computation methods attempt to dynamically control reasoning length using proxy scores such as confidence, but may omit key steps or retain redundancy, leading to unstable mitigation of overthinking. Structured search and pruning methods (e.g., Tree-of-Thought) (Yao et al., 2023) improve reasoning via multi-path exploration and heuristic pruning, but often incur high computational cost and provide limited stepwise explanations.

Overall, inference-time optimization always depends on heuristics or implicit control rather than explicit modeling of step contributions. In contrast, this paper models reasoning structure at the step level and evaluates steps via information contribution to enable more interpretable compression.

A.2 CoT Compression and Efficiency

This direction compresses reasoning primarily through training or sequence-level post-processing. Distillation and SFT encourage concise outputs by supervising models with short CoTs or final answers (Chen et al., 2025a; Shen et al., 2025b; Fan et al., 2025; Wu et al., 2025). These methods can be effective but typically depend on teacher models and operate at the sequence level, making it difficult to distinguish essential steps from redundant ones. Other approaches rely on auxiliary scores (e.g., confidence) or heuristic rules for token- or sequence-level deletion (Xia et al., 2025; Zhang et al., 2026b; Zhao et al., 2025; Wang et al., 2025; Lucas et al., 2026; Zhang et al., 2026a). Some recent works have also explored step-level metrics like step entropy to compress trajectories (Li et al., 2026). However, while sometimes controllable, such operations can disrupt semantic structure and offer limited mechanistic guarantees because the criterion is not always grounded in step-level answer attribution or evaluated for distributional shift.

To explicitly optimize reasoning trajectories and mitigate invalid branches, recent works have increasingly introduced multi-turn reinforcement learning and Process Reward Models (PRMs) (Zhang et al., 2024; Ning et al., 2025; Khalifa et al., 2025; Zou et al., 2026; Hou et al., 2025). Concurrently, some studies explore latent or abstract reasoning to bypass explicit text generation entirely, achieving efficiency natively within the latent space (Ramji et al., 2026; Tan et al., 2026; Wang et al., 2026a; Shen et al., 2025a; Wang et al., 2026b). While promising, this research maintains a focus on interpretable text-level CoT compression, seamlessly integrated with generation fidelity and memory management potentials (Zhu et al., 2026).

In summary, all compression criteria are driven by external scores or teacher biases rather than the reasoning structure itself. In contrast, this paper explicitly models step dependencies, selects steps via information contribution, and uses *SPC* to reduce the risk of deleting critical steps, improving both interpretability and reliability.

B Datasets and Models

We evaluate logical and commonsense reasoning on CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021), and mathematical reasoning on GSM8K (Cobbe et al., 2021). To assess generalization across model scales and architectures, experiments use DeepScaleR-1.5B-Preview (Luo et al., 2025), DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025), Qwen3-8B (Yang et al., 2025), and Qwen3-14B (Peng et al., 2024).

C Training and Pruning Hyperparameters

Training and pruning details. All models are fine-tuned using a standard Supervised Fine-Tuning (SFT) objective equipped with Low-Rank Adaptation (LoRA), with output directory overwriting enabled. The training hyperparameters are configured with 4-bit quantization, LoRA rank of 32, LoRA alpha of 64, LoRA dropout of 0.1, per-device training batch size of 1, gradient accumulation steps of 4, learning rate of 8×10^{-5} utilizing a cosine learning rate scheduler and 0.0 warmup ratio, 5 training epochs, logging every 5 steps, saving checkpoints every 20 steps, and a cutoff length of 512 using bfloat16 precision. For the CommonsenseQA (CQA) and StrategyQA full-fusion runs, the pruning hyperparameters are $\alpha = 0.6$, $\lambda = 15$, $\kappa = 180$, $\tau_a = 0.5$, $\tau_m = 0.45$, $\tau_{\text{vip}} = 0.9$, and SPC cosine-distance threshold $\delta = 0.75$. Here λ is the outer penalty weight, whereas κ controls both the temperature of the softplus and the steepness of the VIP gate. Because some runs use different default thresholds, the revision will report a complete per-run parameter table, score logs, and reproduction scripts rather than treating this configuration as universal across all experiments.

D Supplementary Experiment: Word Frequency Analysis on Overthinking

A core motivation of this work is mitigating the "overthinking" phenomenon, characterized by repetitive paraphrasing, unproductive branching, and excessive self-correction. To quantitatively understand how LDP addresses this, a word frequency analysis is conducted comparing the generated trajectories of Qwen3-8B on GSM8K before and after LDP application.

Linguistic markers of overthinking. By isolating the vocabulary distributions of the reasoning sequences, this paper identifies explicit lexical markers of hesitation versus direct derivation. As illustrated in Figure 3, the unpruned baseline model frequently generates transition phrases indicating uncertainty or unproductive backtracking (e.g., "Wait", "But", "Maybe", "Let me check"). After applying LDP, the frequency of these non-informative hesitation markers drops dramatically. Conversely, logical conclusion markers such as "Thus", "Overall", and "Finally" are either preserved or increased, indicating that LDP leads to more direct derivations. This lexical shift confirms that LDP does not indiscriminately truncate reasoning. Rather, it effectively targets and filters out low-density semantic patterns, cultivating the model's intrinsic ability to reason efficiently.

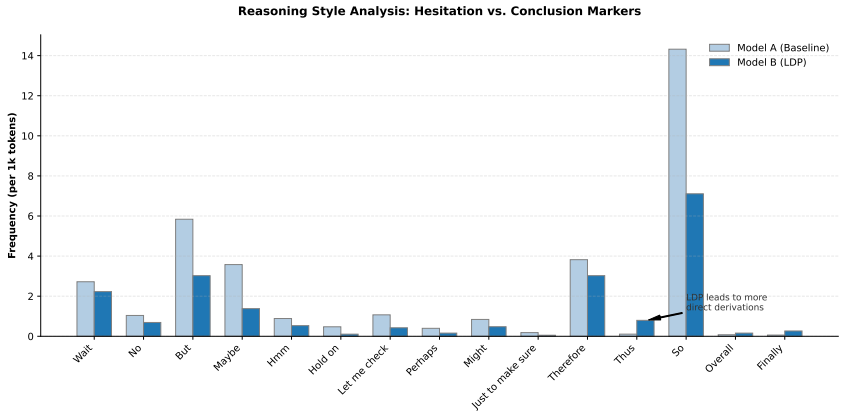


Figure 3: *Reasoning Style Analysis: Hesitation vs. Conclusion Markers.* The bar chart compares the word frequencies (per 1k tokens) between the baseline model and the LDP-compressed model. LDP significantly reduces hesitation markers (e.g., "Wait", "Maybe") while preserving or enhancing direct derivation markers (e.g., "Thus", "Finally").