Think Clearly: Improving Reasoning via Redundant Token Pruning

Anonymous ACL submission

Abstract

Recent large language models have shown promising capabilities in long-form reasoning, following structured chains of thought before arriving at a final answer. However, we observe that these reasoning paths tend to include substantial redundancy; analyzing attention patterns reveals that attention scores are widely scattered, particularly incorrect answers exhibit greater attention sparsity. In this paper, we demonstrate that deliberately removing this redundancy in the reasoning process significantly improves performance through clear thinking, i.e., removing distraction. Specifically, we systematically identify reasoning redundancy by measuring token-level attention scores to a special end-of-thinking token, which is appended to an explicit instruction inserted to conclude each intermediate reasoning step. Furthermore, we propose structure-aware pruning that prioritizes removing tokens in low-contributing reasoning chunks over individual tokens. After evicting redundant tokens, we remove the injected end-of-thinking instruction, then resume the reasoning generation. We demonstrate that our method significantly improves overall accuracy across reasoning-intensive benchmarks without any training involved. In particular, our method shows strong performance on challenging mathematical competition benchmarks such as AIME and AMC, where reasoning redundancy is more prevalent.

1 Introduction

Large language models (LLMs) have demonstrated remarkable progress in complex reasoning tasks (Wei et al., 2022; Zelikman et al., 2022), including mathematical problem solving (Shao et al., 2024), multi-hop question answering (Chen et al., 2019), and long-form instruction following (Bai et al., 2024). This success is often attributed to the emergence of structured reasoning chains, generating sequences of intermediate thoughts that gradually lead to a final answer (Kojima et al., 2022; Hao et al., 2024). These reasoning chains allow models to break down complex problems into smaller, more manageable subproblems, mimicking the step-by-step cognitive strategies humans employ when reasoning under uncertainty (Prystawski et al., 2023).

In particular, recent reasoning models are trained to verbalize their internal thoughts, effectively leveraging the language abilities of pre-trained models (Guo et al., 2025; Jaech et al., 2024). Additionally, this verbalization offers a key advantage: it allows users to monitor and analyze—or even intervene in—the model's thought process during generation (Baker et al., 2025; Wu et al., 2025).

In this paper, we found a somewhat interesting observation by monitoring this internal thought process of reasoning LLMs: *reasoning chains often consist of significant redundancy*. Specifically, the model generates intermediate reasoning steps that tend to be repetitive, verbose, or include speculative detours that do not ultimately contribute to the final answer. Such redundancies are also observed by analyzing the attention patterns, where attention distributions during reasoning typically consist of sparse patterns. This is especially problematic as LLM can be easily distracted by irrelevant or redundant context (Shi et al., 2023), and this tendency is particularly evident when incorrect answers are generated (see Fig. 1). More intriguingly, we observe reasoning chunks that receive consistently low attention from subsequent tokens, suggesting that the model briefly explores these misleading paths but eventually abandons them, leaving behind redundant traces in the generated sequence. This raises a key question:

Can we improve the performance by identifying and removing redundant tokens on-the-fly during the reasoning process?

To this end, we propose a simple yet effective test-time token pruning method that removes redundant reasoning tokens. The core idea is to dynamically eliminate redundant tokens during generation, thereby enabling the model to preserve only the most critical reasoning steps necessary for reaching the correct answer. Specifically, we propose two components: (i) identifying redundant tokens by measuring their contribution to a summarization-inducing end-of-thinking token, and (ii) structure-aware pruning that prioritizes removing low-contributing reasoning chunks rather than individual tokens. Motivated by the observation that redundant tokens tend to receive low attention from the token that concludes the reasoning step, we inject an explicit instruction that prompts the model to summarize and terminate the current thought process, enabling redundancy measurement at intermediate stages.[1] Rather than removing tokens in isolation, we first detect reasoning chunks that are unlikely to contribute to the final answer (i.e., misleading paths), and prune tokens within those chunks. Once pruning is completed, we resume the generation by removing the injected end-of-thinking instruction.

We conduct a comprehensive set of experiments to evaluate our token pruning scheme, focusing on reasoning-heavy scenarios across a wide range of tasks and models. Our results demonstrate that this simple and lightweight inference-time approach can significantly improve reasoning performance. In particular, our method yields significant gains on challenging mathematics competition benchmarks such as AIME and AMC, where the model tends to generate more redundant reasoning steps, making our pruning approach especially effective. For instance, our method improves the original model's accuracy from 55.0% to 82.5% on AMC2023 (AI-MO, 2023), while reducing KV cache memory usage by 10.3% on Qwen2.5-7B (Yang et al., 2024b).

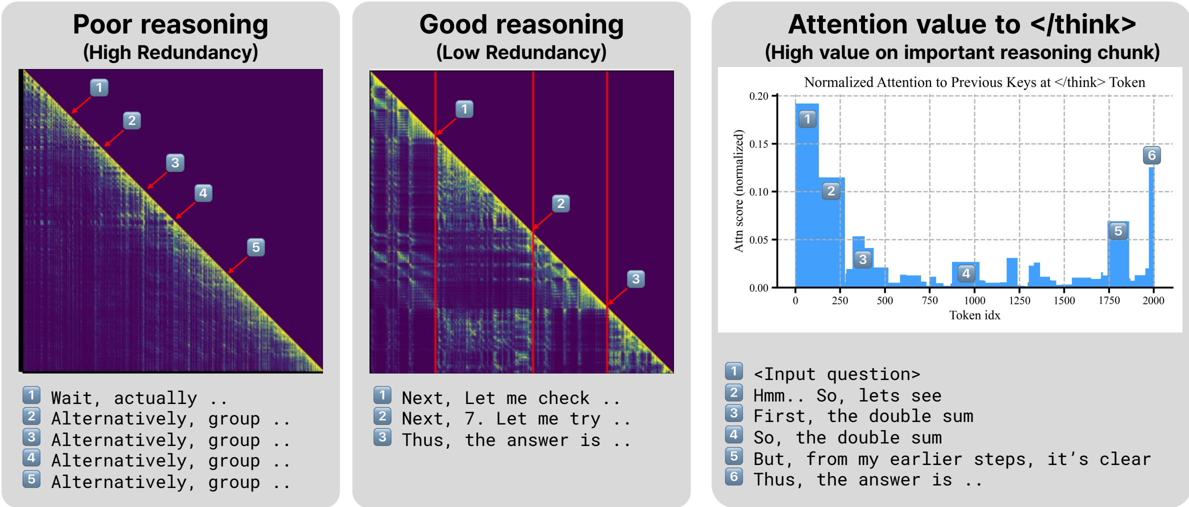
2 Related Work

We provide a comprehensive review of related works on reasoning (focusing on the long chain-of-thought literature) and token pruning and compression frameworks.

Long chain-of-thought. Recent Large Reasoning Models (LRMs) (Guo et al., 2025; Jaech et al., 2024) have increasingly adopted explicit chain-of-thought (CoT) reasoning to enhance performance on complex tasks such as math (Shao et al., 2024), science (Qwen, 2024), and symbolic reasoning (Xu et al., 2024). Instead of producing direct answers, these models generate multi-step intermediate reasoning traces, allowing them to break down complex problems into smaller, more manageable steps (Kojima et al., 2022; Prystawski et al., 2023). This long-form reasoning improves both accuracy and robustness (Wang et al., 2024; Guan et al., 2024), as it provides opportunities for self-correction (Kumar et al., 2025), verification (Lee et al., 2025), and intermediate supervision during training or inference (Wu et al., 2025). A common pattern involves separating the reasoning phase from the answer phase, either through special tokens or by internal abstraction, where the reasoning is hidden and only summarized (Hammoud et al., 2025a). Some models expose the entire reasoning trace to the user to increase interpretability and transparency (Baker et al., 2025), while others keep it latent to reduce vulnerability to prompt manipulation or over-reliance (Hao et al., 2024). This shift toward structured, multi-step reasoning has been central to the recent progress of reasoning-focused LLMs, enabling strong generalization to diverse domains, from mathematics to program synthesis (Gao et al., 2023). In this paper, we propose an efficient yet effective test-time reasoning method by pruning redundant tokens, enabling the model to focus on critical points during long thinking.

Token pruning and compression. As the context length and generated sequences of large language models (LLMs) increase, the memory cost of self-attention becomes a significant bottleneck (Dao et al., 2022). In particular, the key-value (KV) cache, which stores past activations for each generated token, grows linearly with the sequence length. To address this, several existing works have explored strategies to identify and evict redundant tokens in KV cache (Li et al., 2024). Xiao et al. (2024) observe that attention distributions often exhibit strong focus on initial tokens, and show that retaining only the initial and most recent tokens is sufficient to preserve performance. Other approaches leverage attention-based metrics to guide KV eviction (Chen et al., 2024). For example, Zhang et al. (2023) proposes accumulating attention scores over decoding steps and using them as token importance indicators. Oren et al. (2024) evict the token with the lowest attention score at each decoding step. Yang et al. (2024a)

[1]We perform such an additional (summarization) prompting every 200 token-generation and observe that it introduces marginal overhead in inference speed.



(a) Attention map and corresponding text

(b) Attention score on important chunks

Figure 1: **Not all tokens are created equal for reasoning.** We visualize and analyze the attention map of the output sequence. (a) Attention maps when the model fails to produce the correct answer (i.e., poor reasoning) and when it succeeds (i.e., good reasoning). Poor reasoning leads to highly redundant attention patterns. (b) Attention scores associated with the end-of-thinking token </think>. The histogram shows that </think> attends to key reasoning chunks that contain crucial information for deriving the final answer. We use Qwen2.5-7B distilled from DeepSeek-R1 on a subset of the MATH-500 dataset.

observe that the number of crucial tokens varies across layers, motivating a layer-wise compression strategy. While these methods primarily aim to improve inference efficiency, our approach focuses on eviting redundant reasoning tokens to improve the performance (compared to the full KV cache). Nevertheless, we demonstrate the potential of using our method as a reasoning compression scheme, achieving competitive performance under memory constraints and often exceeding recent token eviction and compression baselines.

3 Not All Tokens Are Created Equal for Reasoning

In this section, we investigate whether reasoning models truly require all previously generated tokens to reach a correct final answer. To this end, we focus on recent reasoning LLMs output $\mathcal{T}_{\text{output}}$, which consists of multi-step intermediate reasoning traces $\mathcal{T}_{\text{reason}}$ followed by final answer $\mathcal{T}_{\text{answer}}$ with special delimiters <think>... </think>.

Existence of redundant reasoning tokens. To identify which reasoning tokens are important for reaching the answer, we visualize the attention maps of two samples: (i) a sample that fails to answer the given question, and (ii) a sample that successfully reaches the end-of-thinking token to produce the correct answer. As shown in Fig. 1a, the first case includes redundant text (e.g., repeated attempts to rethink the process using phrases like "Alternatively"), whereas the second case exhibits a clear reasoning trajectory. Interestingly, the attention maps reflect this behavior; in the first case, the attention is highly sparse due to redundancy in reasoning, while in the second case, the model attends more frequently to previously generated tokens and demonstrates a more global structure. These results highlight the potential of using attention scores to identify redundant reasoning steps.

Attention score to </think>. To quantify token importance more systematically, we analyze the attention scores directed to the special token </think>, which marks the end of the reasoning process. As shown in Fig. 1b, reasoning tokens that contribute to the final answer tend to have high attention scores to </think>. For example, the </think> token frequently attends to sentences or chunks that initiate the reasoning process or summarize key conclusions. Interestingly, redundant tokens often appear in contiguous chunks rather than in isolation, suggesting that the </think> token selectively attends to informative reasoning segments while ignoring irrelevant parts.

Based on this observation, we design a systematic token pruning strategy that leverages the end-of-thinking token </think> and the chunked structure of the reasoning process to identify and remove redundant reasoning tokens.

Algorithm 1 Redundant Token Eviction via Self-summarization

Require: Reasoning tokens $T = \{t_1, \ldots, t_L\}$, eviction budget k, layers $\ell \in [1, L]$, heads $h \in [1, H]$

Ensure: Set of k tokens to evict from KV cache

Inject summarization prompt $\mathcal{T}_{\text{summ}}$ into the input
Generate response including summarization trigger token </think>
for each layer ℓ and head h **do**
  for each token $t \in T$ **do**
    $s_t^{(\ell,h)} \leftarrow \alpha^{(\ell,h)}_{\text{</think>} \to t}$
  end for
  Segment $\mathcal{T}_{\text{reason}}$ into steps $\{r_1, \ldots, r_N\}$
  for each step r_i **do**
    $c_{r_i}^{(\ell)} \leftarrow \frac{1}{|H \cdot r_i|} \sum_{h \in H} \sum_{t \in r_i} s_t^{(\ell,h)}$
  end for
  Sort steps $\tilde{r}_1^{(\ell)}, \ldots, \tilde{r}_N^{(\ell)}$ in ascending order of $c_{r_i}^{(\ell)}$
  $k_{\text{rem}} \leftarrow k$
  for each sorted reasoning step $\tilde{r}_i^{(\ell)}$ **do**
    $e_{\tilde{r}_i}^{(\ell)} \leftarrow \min\left(|\tilde{r}_i^{(\ell)}|,\ k_{\text{rem}}\right)$
    $k_{\text{rem}} \leftarrow k_{\text{rem}} - e_{\tilde{r}_i}^{(\ell)}$
    Evict $e_{\tilde{r}_i}^{(\ell)}$ tokens with lowest $s_t^{(\ell,h)}$ in $\tilde{r}_i$ per head h in layer ℓ
    if $k_{\text{rem}} = 0$ **then**
      break
    end if
  end for
end for

4 Improving Reasoning with Redundant Token Pruning

In this section, we propose a novel KV cache eviction policy that can improve reasoning models' effectiveness and efficiency by removing redundant tokens. Specifically, we suggest a novel scoring function driven by self-summarization to identify redundant tokens (in Section 4.1), and introduce a stepwise eviction policy that aggressively removes KV cache in the redundant reasoning step (in Section 4.2). The overall procedure is illustrated in Algorithm 1.

4.1 Identifying redundant tokens via self-summarization

As shown in Fig. 1, the end-of-thinking token </think> serves as a crucial cue for identifying important reasoning tokens. Based on this, we propose a novel scoring function for identifying redundant tokens in the reasoning trace during the intermediate decoding so that one can only preserve important tokens. Specifically, we leverage a short summarization prompt ending with </think>, prompting the model to briefly summarize its own reasoning. This forces the LLM to end the thinking process, thereby effectively localizing the essential part inside the reasoning trace.

Use of summarization prompts. During the intermediate step of the decoding, we periodically trigger the model to summarize and answer the question at every fixed interval. Here, to evaluate the redundancy of tokens during reasoning, our key idea is to forward the reasoning model with a short summarization prompt $\mathcal{T}_{\text{summ.}}$ which is constructed as follows:

```
"Time is up. Given the time I've
spent and the approaches I've
tried, I should stop thinking and
now write summarization in one
sentence.</think>"
```

Especially, the prompt is designed to explicitly shift the model from reasoning to summarization, making the </think> token to capture informative tokens in the reasoning trace without generating explicit summarization.

Token importance score. To quantify the importance of each token, we accumulate attention weights assigned to previous tokens given the summarization prompt $\mathcal{T}_{\text{summ.}}$. Specifically, for each token t in the current reasoning trace, we define its importance score $s_t^{(\ell,h)}$ at layer ℓ and head h by aggregating attention values by injecting the summarization prompt with </think>:

$$s_t^{(\ell,h)} = \alpha^{(\ell,h)}_{\text{</think>} \to t}, \tag{1}$$

where $\alpha^{(\ell,h)}_{\text{</think>} \to t}$ denotes the attention weight from the </think> in summarization tokens $\mathcal{T}_{\text{summ.}}$ at layer ℓ and head h. This score reflects how much each token contributes to the final summarization, as perceived by the model. Since the scores are computed separately for each layer and attention head, pruning decisions are made independently at each level, enabling fine-grained control over which tokens are retained.

4.2 Step-aware eviction with hierarchical budget allocation.

We now present our eviction policy under a fixed token eviction budget k. Motivated by our observation that reasoning traces often contain redundant steps, we aim to remove tokens from such steps while preserving essential ones. To this end, we first segment the reasoning trace into semantically coherent steps and then allocate the eviction budget hierarchically across these steps based on the importance score.

Aggregating importance score per reasoning step. Following a previous work (Hammoud et al., 2025b), we first divide the reasoning trace into intermediate steps, and each steps consists of a consecutive set of tokens with logical continuity in reasoning (See Appendix A.2 for detail). Given importance score $s_t^{(\ell,h)}$, we compute step score $c_r^{(\ell)}$ by taking the mean over all tokens t within the same reasoning step r across head h:

$$c_r^{(\ell)} = \frac{1}{|H \cdot r|} \sum_{h \in H} \sum_{t \in r} s_t^{(\ell,h)}, \tag{2}$$

where $|H|$ is the number of heads and $|r|$ is length of reasoning step.

Hierarchical eviction. Given a token eviction budget k, we aim to evict tokens primarily from redundant reasoning steps, while preserving informative ones. To achieve this, we suggest a hierarchical eviction policy that allocates the eviction budget k in a step-aware manner, considering the reasoning structure. Formally, given an importance score of reasoning step $c_r^{(\ell)}$, we first sort all reasoning steps $\tilde{r}_1^{(\ell)}, \tilde{r}_2^{(\ell)}, \ldots, \tilde{r}_N^{(\ell)}$ in ascending order of $c_{\tilde{r}_i}^{(\ell)}$ (i.e., $\tilde{r}_1^{(\ell)}$ is the most redundant step at layer ℓ). Then, following this order, we greedily allocate a step-level eviction budget $e_{\tilde{r}_i}^{(\ell)}$ to each reasoning step $\tilde{r}_i^{(\ell)}$ as:

$$e_{\tilde{r}_i}^{(\ell)} = \min \left(|\tilde{r}_i^{(\ell)}|, \; k - \sum_{j=1}^{i-1} e_{\tilde{r}_j^{(\ell)}} \right), \tag{3}$$

where $|\tilde{r}_i^{(\ell)}|$ is the number of tokens in step $\tilde{r}_i^{(\ell)}$, and k is the total token eviction budget. This allocation ensures that the total number of evicted tokens does not exceed k, while prioritizing the more redundant steps. After allocation, we evict tokens with the lowest token-level redundancy scores $s_t^{(\ell,h)}$ within each step $r_i^{(\ell)}$. This strategy naturally favors highly redundant reasoning steps while avoiding premature removal from important ones.

5 Experiments

In this section, we present a thorough evaluation of our proposed framework, with the goal of verifying its ability to improve both reasoning accuracy and inference efficiency. Our evaluation is divided into three parts: (1) effectiveness, which examines how well our method improves final answer correctness across a range of mathematical reasoning tasks (Table 1); (2) efficiency, which measures the memory savings achieved by our token pruning strategy (Table 2); and (3) ablation and analysis, which assesses the contribution of each individual component of our framework, and explores how well it generalizes to other domains and tasks (Table 3, Table 4, and Table 5).

We empirically demonstrate that our method not only reduces the computational overhead typically associated with long-form reasoning, but also improves accuracy by filtering out redundant and misleading intermediate steps. Across all tested datasets and model sizes, our approach outperforms existing KV cache compression baselines, often by a significant margin. Importantly, the gains are achieved without retraining or additional supervision, indicating that our method is broadly applicable as a plug-and-play enhancement for any autoregressive reasoning model.

5.1 Experimental setup

We provide a detailed description of the experimental setup, covering the datasets, models, baselines, and evaluation protocol.

Datasets. We evaluate our method on a diverse suite of publicly available benchmarks that span a wide range of mathematical reasoning tasks, difficulty levels, and linguistic diversity. Our core evaluation includes three widely used English benchmarks: GSM8K (Cobbe et al., 2021), which focuses on grade-school level arithmetic and requires step-by-step calculations; MATH-500 (Hendrycks et al., 2021), a dataset of competition-level problems across algebra, geometry, and combinatorics; and Minerva Math (Lewkowycz et al., 2022), which consists of high-school and advanced math questions sourced from web documents.

To further assess the robustness of our method on real-world and harder problems, we include recent

Table 1: **Effectiveness of the redundant token pruning.** We compare the proposed method (Ours) with the standard decoding (FullKV) under six reasoning-intensive mathematical benchmarks, including MATH-500 (MATH), Minerva, GaoKao, AIME2024, AIME2025, AMC2023. All models, including Qwen2.5-7B and Llama3.1-8B, are reasoning LLMs distilled from DeepSeek-R1. We report accuracy (%) with the corresponding average KV cache length shown below in parentheses. Average accuracy and KV cache are presented in the final column. The bold indicates the best accuracy within the group.

Model	Method	Dataset						
		MATH	Minerva	GaoKao	AIME2024	AIME2025	AMC2023	Avgerage
Qwen2.5-7B	FullKV	87.0 (3397)	59.9 (3391)	65.8 (3845)	36.7 (7060)	23.3 (7133)	55.0 (5004)	54.6 (4972)
	Ours	**87.2** (2926)	**60.5** (3471)	**67.1** (4219)	**46.7** (6841)	**36.7** (6905)	**82.5** (4488)	**63.5** (4808)
Llama3.1-8B	FullKV	81.0 (3389)	45.9 (4060)	**67.1** (4689)	33.3 (7067)	13.3 (7088)	80.0 (4949)	53.4 (5207)
	Ours	**82.2** (3365)	**47.4** (4186)	65.7 (4507)	**46.7** (7163)	**36.7** (7057)	**82.5** (4759)	**60.2** (5172)

evaluation sets from mathematical competitions: AIME 2024 (AIME, 2024), AIME 2025 (AIME, 2025) and AMC 2023 (AI-MO, 2023), all of which contain challenging, high-school level problems that demand precise logical deductions. Additionally, we include GaoKaoMath (Zhong et al., 2023), to assess the generality of our method on questions originating from the Chinese college entrance exam. Finally, we further validate the general applicability of our method by evaluating it on a non-mathematical reasoning dataset, namely the GPQA (Rein et al., 2024). Note that GQPA is a graduate-level question that consists of science fields, requiring intensive reasoning capability to reach the answer.

Models. Our experiments are primarily based on the DeepSeek-R1-Distill family of models, which are designed to emulate the reasoning behavior of DeepSeek-R1 (Guo et al., 2025) using distilled versions of popular backbones. We mainly consider three backbone models: Qwen2.5-1.5B, Qwen2.5-7B (Yang et al., 2024b), and Llama3.1-8B (Grattafiori et al., 2024), all trained with visible chain-of-thought reasoning traces. Across all models, we observe consistent benefits of our method, suggesting that our framework is architecture-agnostic and works well across a wide range of capacities.

Baselines. We mainly compare our method with the standard decoding strategy that uses the full KV cache (FullKV). Additionally, we compare our method against a range of recent decoding-time KV compression methods that can be applied without retraining, including StreamingLLM (Xiao et al., 2024), which retains only the first and most recent tokens; H2O (Zhang et al., 2023), which uses accumulated attention scores to determine token importance; Pyramid-Infer (Yang et al., 2024a), which performs layer-wise importance-based pruning.

Evaluation protocol. We adopt a standardized evaluation protocol across all methods and datasets to ensure fair comparisons. All models are evaluated using deterministic generation with a temperature of 0.6 and top-p of 0.9 under fixed seed to maximize replicability. The maximum generation length is capped at 8192 tokens, sufficient for nearly all long-form reasoning examples. To extract the final answer from the generated output, we introduce a designated token such as </think> to mark the end of the reasoning phase. Accuracy is measured as the fraction of correctly answered questions, based on an exact match with the ground truth. For efficiency, we measure the average number of KV tokens stored during generation, normalized by the total number of generated tokens, as well as the total memory consumption when storing the KV cache.

5.2 Redundant reasoning token pruning improves the performance

We present that removing redundant tokens can improve the accuracy. To this end, we compare our method with the full KV (FullKV) cache method on reasoning-intensive mathematical benchmarks. As

Table 2: **KV cache efficiency of the redundant token pruning.** We compare the proposed method (Ours) with KV compression frameworks on MATH-500. We use Qwen2.5-1.5B reasoning LLM distilled from DeepSeek-R1. For reference, we report the standard decoding without KV compression (FullKV) results with the accuracy (%). We evaluate accuracy under two KV compression ratios (25% and 50%). The bold indicates the best results within the group.

	Compression ratio	
Method	25%	50%
FullKV	42.6	
Streaming-LLM	34.8	38.6
H2O	34.6	39.2
Pyramid-Infer	29.8	38.7
Ours	**36.2**	**40.4**

Table 3: **Component analysis.** We ablate the two main components of our method: self-summarization (Summ) and step-aware token eviction (Step). We report the accuracy (%) on the AIME2024 and AMC2023 datasets. Here, we use Qwen2.5-7B distilled from DeepSeek-R1, where all decoding uses a temperature of 0.6. The bold indicates the best results.

Summ	Step	AIME2024	AMC2023
✗	✗	40.0	70.0
✓	✗	36.7	77.5
✓	✓	**46.7**	**82.5**

shown in Table 1, our method significantly and consistently outperforms FullKV in accuracy. For instance, our method improves the average accuracy of FullKV from 54.6% to 63.5% under Qwen2.5-7B. It is worth noting that our method only involves changing the inference strategy, thus showing wide applicability.

Notably, our approach consistently outperforms FullKV decoding despite using significantly fewer tokens in the KV cache. This suggests that our pruning mechanism not only reduces memory usage but also acts as a form of implicit regularization that helps the model focus on essential reasoning steps by making LLM less distracted by unnecessary text (Shi et al., 2023). The forced summarization phase typically encourages the model to internally consolidate reasoning before producing an answer, leading to more coherent and accurate generations.

Furthermore, more interestingly, our method yields greater improvements on more challenging benchmarks such as AMC2023 or AIME datasets (i.e., mathematical competition problems). For example, the performance on the AMC2023 dataset significantly improves from 55.0→82.5 on Qwen2.5-7B, and even uses 10% less KV cache. We conjecture that the LLM tends to struggle more and produce a more redundant reasoning path in challenging setups, thus pruning such redundant tokens is effective. These results validate our core hypothesis: not all tokens are necessary for reasoning, and selectively pruning unhelpful tokens can enhance the final outcome.

5.3 Redundant token pruning efficiently and effectively reduces KV cache budget

To evaluate the memory efficiency of our method, we vary the token pruning budget and compare the resulting KV cache size and model accuracy. Table 2 shows the performance of all methods at various relative cache budgets (e.g., 25% and 50% compared to full KV cache size). Our method achieves superior compression ratios without compromising accuracy, while other methods suffer significant accuracy degradation under aggressive pruning.

Unlike previous approaches that are not specialized to the reasoning process compression, our method dynamically adapts to the reasoning process by allocating eviction budgets fairly across reasoning chunks. This chunk-aware strategy ensures that each reasoning step retains its most important context, enabling robust final answers even under tight memory constraints. In particular, our method maintains over 95% of the full-KV accuracy at just 50% memory usage, making it an attractive solution for deployment in resource-constrained environments such as mobile devices, embedded systems, or large-batch inference setups.

5.4 Ablation and analysis

We further analyze the contributions of individual components in our framework and investigate its generalizability across domains and complementary methods. Throughout this section, unless otherwise specified, we consider the Qwen2.5-7B reasoning model that is distilled from DeepSeek-R1.

Component analysis. To understand which parts of our method drive the observed gains, we perform an ablation study where we remove key components one at a time. As shown in Table 3, both the self-summarization phase and the stepwise eviction

Table 4: **Importance of deliberate token pruning.** Random or structure-agnostic pruning degrades accuracy (%), while our method improves performance by pruning at semantically meaningful reasoning boundaries. We evaluate on Qwen2.5-7B distilled from DeepSeek-R1 across mathematical reasoning benchmarks, including AIME2024 and AIME2025. The bold indicates the best results.

Score	AIME2024	AIME2025
FullKV	36.7	23.3
Random	36.7	33.3
H2O	40.0	26.7
Ours	**46.7**	**36.7**

Table 5: **Effectiveness on non-mathematical reasoning benchmarks.** We compare the proposed method (Ours) with the standard greedy decoding (FullKV) under a non-mathematical reasoning benchmark, GPQA (science). We use Qwen2.5-7B reasoning LLM distilled from DeepSeek-R1. We report the accuracy (%) and the corresponding average KV cache length in the parentheses. The bold indicates the best results.

Method	GPQA
FullKV	32.0 (6418)
Ours	**36.4** (6277)

budget contribute meaningfully to performance. Removing the summarization step by just inserting the end-of-thinking </think> token leads to inaccurate importance scores, resulting in the removal of critical tokens. Conversely, removing the balanced budget allocation step results in over-pruning from certain chunks, reducing coherence. The full model, with both components, consistently yields the best results. This supports our design intuition that token importance is context-dependent and that a structured pruning policy is necessary to avoid harming the model's reasoning ability.

Does eviction alone improve performance? A natural question is whether token eviction itself—regardless of how the evicted tokens are selected—can lead to improved performance. To verify this, we compare three strategies under our framework: (1) Random, which evicts tokens uniformly at random at each step, (2) H2O, which prunes tokens with the lowest accumulated attention scores, and (3) Ours, which step-aware evicts tokens based on a score triggered by an end-of-thinking token </think>.

As shown in Table 4, both Random and H2O yield marginal performance improvements, indicating that even naive or structure-agnostic pruning can occasionally help by reducing redundancy. However, such approaches risk removing semantically important context, leading to unstable gains. In contrast, our method significantly outperforms others by aligning token eviction with the semantic structure of the reasoning trace.

We observe that using reflection markers as terminators enables more frequent pruning, and degradation in performance is due to the premature removal of context that is still relevant for the final answer. Fixed-interval pruning also performs poorly, as it fails to align with the logical structure of the reasoning path and often removes informative tokens arbitrarily. Overall, our results confirm that pruning at semantically meaningful boundaries—particularly after conclusive reasoning segments—is critical for effectiveness.

Effectiveness on a non-mathematical reasoning benchmark. Finally, we test whether our method generalizes beyond mathematical reasoning. In Table 5, we report results on GPQA (Rein et al., 2024), a dataset of expert-written multiple-choice science questions. Despite the distinct nature of these tasks, our method consistently improves the performance over Full-KV, demonstrating its robustness across reasoning styles. We find that the original model iteratively generates intermediate justifications when tackling GQPA, which hurts answer quality. Here, our method effectively suppresses such distractions, thus improving the performance.

6 Conclusion

In this paper, we propose a test-time scaling method for enhancing reasoning in large language models by identifying and pruning redundant tokens during the generation of reasoning traces. By introducing a forced summarization phase at intermediate reasoning steps, we estimate the contribution of each token to the ongoing reasoning process. Combined with a stepwise budget allocation strategy, our method demonstrates that targeted KV cache pruning can not only serve as a compression mechanism but also improve reasoning performance. Moreover, our eviction algorithm is plug-and-play, making it suitable for memory-constrained settings, where it outperforms existing baselines under high compression scenarios.

Limitations

While our proposed framework demonstrates strong performance in both accuracy and memory efficiency across a range of reasoning tasks, it also comes with several limitations that open avenues for future work. First, our method relies on the presence of explicit reasoning traces in model outputs (e.g., chain-of-thought reasoning or intermediate steps marked by special tokens such as <think>). This restricts applicability to models trained with visible thoughts or intermediate supervision. For models that operate in an end-to-end manner without exposing their internal reasoning, our summarization-based token importance estimation may not generalize directly. Second, our pruning strategy is applied only at test time and does not benefit from end-to-end training with pruning in the loop. While this makes our method widely compatible with existing models, it also limits its adaptiveness. Learning token importance jointly with model weights—potentially via reinforcement learning or differentiable attention masking—could further improve performance. Third, while our experiments span a diverse set of math and code reasoning datasets, the majority of our evaluation focuses on factual or symbolic reasoning tasks. Extending our method to open-domain question answering, commonsense inference, or multimodal reasoning (e.g., visual QA) remains an open challenge. These settings may require different forms of redundancy detection or finer-grained reasoning segmentation.

References

AI-MO. 2023. AIMO Validation AIME Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime.

AIME. 2024. American invitational mathematics examination (aime). https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

AIME. 2025. American invitational mathematics examination (aime). https://huggingface.co/datasets/opencompass/AIME2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models. In *Annual Conference of the Association for Computational Linguistics*.

Bowen Baker, Joost Huizinga, Leo Gao, Zehao Dou, Melody Y Guan, Aleksander Madry, Wojciech Zaremba, Jakub Pachocki, and David Farhi. 2025. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation. *arXiv preprint arXiv:2503.11926*.

Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019. Multi-hop question answering via reasoning chains. *arXiv preprint arXiv:1910.02610*.

Renze Chen, Zhuofeng Wang, and 1 others. 2024. Arkvale: Efficient generative llm inference with recallable key-value eviction. In *Advances in Neural Information Processing Systems*.

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*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Helyar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, and 1 others. 2024. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*.

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*.

Hasan Abed Al Kader Hammoud, Hani Itani, and Bernard Ghanem. 2025a. Beyond the last answer: Your reasoning trace uncovers more than you think. *arXiv preprint arXiv:2504.20708*.

Hasan Abed Al Kader Hammoud, Hani Itani, and Bernard Ghanem. 2025b. Beyond the last answer: Your reasoning trace uncovers more than you think. *arXiv preprint arXiv:2504.20708*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2025. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. 2025. Revise: Learning to refine at test-time via intrinsic self-verification. In *International Conference on Machine Learning*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Yuhong Li and 1 others. 2024. Snapkv: Llm knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*.

Ben Prystawski, Michael Li, and Noah Goodman. 2023. Why think step by step? reasoning emerges from the locality of experience. In *Advances in Neural Information Processing Systems*.

Qwen. 2024. QwQ: Reflect deeply on the boundaries of the unknown. `https://qwenlm.github.io/blog/qwq-32b-preview/`. Accessed: 2025-05-13.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

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

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*.

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024. Offline reinforcement learning for llm multi-step reasoning. *arXiv preprint arXiv:2412.16145*.

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. In *International Conference on Machine Learning*.

Tong Wu, Chong Xiang, Jiachen T Wang, and Prateek Mittal. 2025. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Annual Conference of the Association for Computational Linguistics*.

Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024a. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *Preprint*, arXiv:2405.12532.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, and 25 others. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

A Experimental Details

A.1 Model details

In our proposed framework, we use the DeepSeek-R1-Distill family of models, namely the Qwen2.5-1.5B[2], Qwen2.5-7B[3], and Llama3.1-8B[4]. All checkpoints are downloaded from Huggingface.

A.2 Implementation

Segment of reasoning steps. At the core of our method is segmenting the initial raw reasoning trace $\mathcal{T}_{\text{Reason}}$ into a sequence of meaningful intermediate steps. This segmentation aims to capture points where the model might pause, reflect, change direction, or move to a distinct next step in its reasoning. Following prior work (Hammoud et al., 2025b), we perform segmentation based on occurrences of words or phrases from a predefined set W. These markers often signal reflection, correction, sequencing, or the exploration of alternatives. The set W used in our experiments is:

> "Wait" "Alternatively" "Another angle" "Another approach" "But wait" "Hold on" "Hmm" "Maybe" "Looking back" "Okay" "Let me" "First" "Then" "Alright" "Compute" "Correct" "Good" "Got it" "I don't see any errors" "I think" "Let me double-check" "Let's see" "Now" "Remember" "Seems solid" "Similarly" "So" "Starting" "That's correct" "That seems right" "Therefore" "Thus"

Eviction budget. For the effectiveness setting, we consider a token eviction budget k, which denotes the number of tokens to be removed from the KV cache across all heads and layers at every predefined generation step p. In all experiments, we fix $k = 5$ and $p = 200$ for mathematical reasoning datasets and $p = 300$ for non-mathematical datasets.

KV cache budget. For the efficiency setting, we define a maximum KV cache budget during decoding, computed based on the average KV length L_{Full} of the Full KV baseline. Specifically, we compute compressed budgets by multiplying L_{Full} with target compression ratios of 25%, 50%. Following prior works (Zhang et al., 2023; Li et al., 2024), we also preserve a recent window of KV entries by keeping the most recent tokens, and set this recent size to half of the allocated cache. For comparison solely focused on reasoning compression, all methods retain the full problem prompt in the KV cache throughout generation. This portion is excluded when measuring the fixed cache budget.

[2] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
[3] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
[4] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Llama-8B

Table 6: **Effectiveness of the redundant token pruning on a small-sized reasoning LLM.** We compare the proposed method (Ours) with standard decoding (FullKV) under six reasoning-intensive mathematical benchmarks, including MATH-500 (MATH), Minerva, GaoKao, AIME2024, AIME2025, AMC2023. We use Qwen2.5-1.5B reasoning LLM distilled from DeepSeek-R1. We report accuracy (%) with the corresponding average KV cache length shown below in parentheses. Average accuracy and KV cache are presented in the final column. The bold indicates the best accuracy within the group.

	Dataset						
Method	MATH	Minerva	GaoKao	AIME2024	AIME2025	AMC2023	Avgerage
FullKV	**42.6** (6125)	21.7 (5663)	34.2 (5825)	0.0 (8192)	0.0 (8192)	10.0 (7398)	18.1 (6899)
Ours	41.2 (6166)	**25.8** (5690)	**36.9** (6071)	**3.3** (8071)	0.0 (8192)	**20.0** (7477)	**21.2** (6945)

B More Experimental Results

B.1 Additional results on a small reasoning model

We also demonstrate the effectiveness of our method on a small reasoning LLM, namely the Qwen2.5-1.5B. As shown in Table 6, our method significantly improves the overall reasoning accuracy across multiple reasoning-intensive mathematical benchmarks even on small reasoning LLMs. Here, we also notice that our method is effective on challenging benchmarks such as AMC2023, indicating the importance of the deliberate token pruning based on redundancy.