Memory Augmentation Unlocks Efficient Chain-of-Thought Reasoning

Anonymous ACL submission

Abstract

Reasoning models achieve remarkable performance through Chain-of-Thought (CoT), yet the verbose reasoning process introduces significant inference latency and computational overhead. CoT compression aims to accelerate inference; however, naive compression approaches inevitably disrupt the coherence of reasoning logic, leading to severe performance collapse. To address this trade-off, we leverage the *Context-Generation Substitution Law*: shifting the computational burden from expensive serial generation to efficient parallel context processing. Guided by this insight, we propose *Memory-Augmented Compression*, a generalizable paradigm that utilizes an explicit memory of abstracted reasoning patterns as a cognitive scaffold. By injecting high-density reasoning patterns into the context, we incur only a marginal prefill cost to bypass redundant reasoning steps, achieving massive output compression. Extensive experiments demonstrate the superiority of this paradigm. Remarkably, as a *training-free, plug-and-play* solution, our method outperforms a fine-tuned baseline by over 22 percentage points on GSM8K. On the challenging MATH-500 benchmark, we achieve robust performance, surpassing the uncompressed Standard CoT by nearly 10%. Comprehensive evaluations validate that our approach effectively establishes a new efficiency frontier for reasoning models.

1 Introduction

While Large Language Models (LLMs) like GPT-4 excel in generation, their standard inference relies on rapid "System 1" pattern matching, which often falters in complex reasoning due to the lack of explicit intermediate steps (Vaswani et al., 2017; Brown et al., 2020; Booch et al., 2021; Bubeck et al., 2023; Wei et al., 2022). To address this, the field has shifted towards inference-time compute paradigms. By leveraging Chain-of-Thought (CoT) to simulate analytical "System 2" thinking (Yao et al., 2023; Wang et al., 2022), a new generation of reasoning models—exemplified by OpenAI o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025)—has emerged, marking a transition to deliberative cognitive planning.

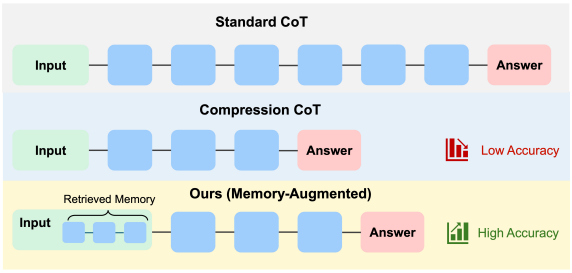

Figure 1: Visual comparison between Standard CoT, Compression CoT, and our Memory-Augmented approach. Our work achieve compact reasoning steps by augmenting input memory and mitigate the accuracy drop due to the disruption of reasoning logic.

Nevertheless, this explicit "long-thinking" paradigm imposes severe challenges on practical deployment. The inherently serial nature of autoregressive decoding means CoT generates reasoning chains spanning thousands of tokens—resulting in an order-of-magnitude increase in latency and memory consumption compared to standard direct responses (Kwon et al., 2023; Pope et al., 2023). To mitigate this, recent studies have explored two primary optimization avenues: dynamic pruning (e.g., TokenSkip (Xia et al., 2025), LightThinker (Zhang et al., 2025)), and reasoning compression (e.g., CoD (Xu et al., 2025), SoT (Aytes et al., 2025)). While these methods offer efficiency gains, they generally grapple with a severe accuracy-efficiency trade-off: aggressive pruning or compression often disrupts the integrity of logical chains. Critically, these approaches remain confined to optimizing the serial decoding process, failing to fundamentally overcome the latency accumulation of autoregressive generation. We argue for a paradigm shift: instead of incrementally improving the bottleneck,

we seek to bypass it entirely by reallocating the computational burden to the massively parallel Prefill Phase.

How can we leverage the parallelism of context scaling to substitute the serial overhead of lengthy generation?

To validate this, we conducted a preliminary study quantifying the interplay between the input context and the model's generated CoT. Our empirical observations reveal a distinct *Context-Generation Substitution Law*: Increasing the density of reasoning patterns within a prompt, achieved by embedding explicit logical cues, permits a corresponding reduction in generation length without compromising the integrity of the reasoning chain. Crucially, this strategy leverages the asymmetric computational characteristics of Transformer inference, where parallel context processing during the prefill phase is significantly cheaper than serial token generation during the decoding phase. We theoretically identify an efficiency frontier, demonstrating that a net reduction in latency is achievable as long as the computational savings from reduced generation outweigh the marginal overhead of context expansion. Crucially, unlike pruning-based methods that suffer severe performance degradation under aggressive compression, our approach utilizes high-quality explicit memory which acts as a "cognitive scaffold," effectively stabilizing the model's reasoning capabilities even when the generation steps are heavily condensed.

Guided by this, we propose *Memory-Augmented Compression*, a generalizable inference paradigm comprising two core phases: *Structured Memory Injection*, which retrieves problem-specific templates and populates the prompt with concise premises; and *Compression Execution*, which directs the model to generate condensed reasoning chains. Designed to be implementation-agnostic, the execution phase is compatible with diverse methodologies, encompassing both prompting strategies and token pruning techniques.

Our contributions are:

- First, we formalize the inference bottleneck with a latency model, establishing the theoretical foundation for substituting generation with context.
- Second, we propose the Memory-Augmented Compression framework, using high-density reasoning patterns as a cognitive scaffold to preserve logical coherence under extreme compression.
- Finally, we establish a context-generation substitution scaling law: replacing 76% of generation steps on GSM8K for only ˜1350 context tokens boosts accuracy by 22% over a fine-tuned baseline.

2 Preliminaries

2.1 Standard Chain-of-Thought Inference

We formalize the standard reasoning process of an LLM $\mathcal{M}_\theta$. Let $\mathbf{x} = (\mathcal{I}, x)$ denote the concatenation of the system instruction and user query. Instead of mapping directly to the answer y, the model generates an intermediate CoT sequence z (Wei et al., 2022). The joint probability decomposes as:

$$P_\theta(y, z \mid \mathbf{x}) = P_\theta(z \mid \mathbf{x}) \cdot P_\theta(y \mid \mathbf{x}, z). \quad (1)$$

The first term $P_\theta(z \mid \mathbf{x})$ corresponds to the *reasoning phase*, where explicit steps are derived. The second term $P_\theta(y \mid \mathbf{x}, z)$ represents the *answering phase*. While z is essential for accuracy, its generation comes at a high computational cost.

2.2 The Inference Latency Bottleneck

The computational cost of autoregressive generation is proportional to the sequence length. Let $\mathcal{L}_{gen} = |z|$ denote the length of the reasoning chain. In modern reasoning models (e.g., OpenAI o1, DeepSeek-R1), $\mathcal{L}_{gen}$ can extend to thousands of tokens, creating a severe latency bottleneck where $\mathcal{L}_{gen} \gg |y|$.

While techniques like Speculative Decoding (Leviathan et al., 2023) can accelerate generation, they do not reduce the fundamental FLOPs required for long z. Thus, compressing z is critical.

2.3 Context-Generation Substitution Law

To address the bottleneck, we introduce an external context variable $\mathcal{C}$ (i.e., explicit memory). Our core hypothesis is that relevant priors in $\mathcal{C}$ can substitute for verbose reasoning steps.

Substitution Mechanism. We propose a paradigm shift where the computational burden moves from serial generation to parallel context processing. Mathematically, we approximate the standard reasoning probability as:

$$P_\theta(y \mid \mathbf{x}, \mathcal{C}) \approx P_\theta(y \mid \mathbf{x}, z). \quad (2)$$

Specifically, we define a *retrieval function* ϕ that maps the input $\mathbf{x}$ and raw context $\mathcal{C}$ into a structured *Reasoning Memory* M:

$$M = \phi(\mathbf{x}, \mathcal{C}). \quad (3)$$

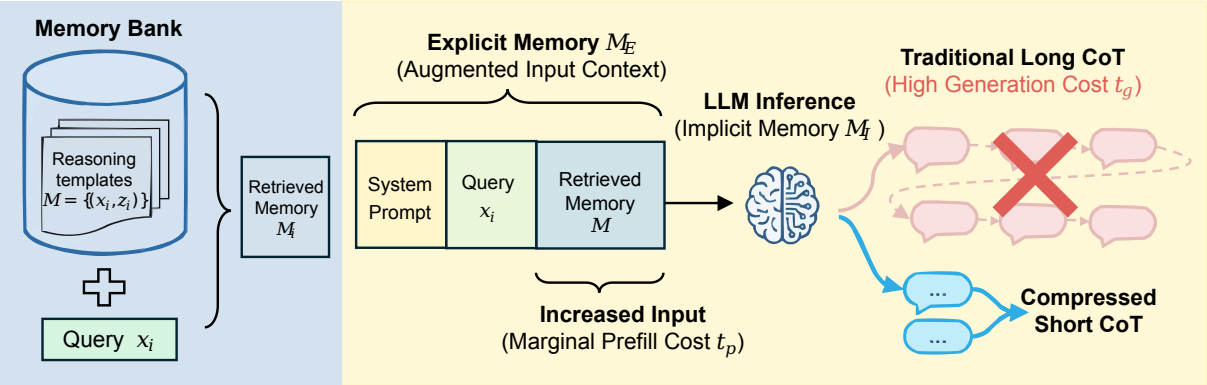

Figure 2: Overview of the proposed Memory-Augmented Inference framework. The system retrieves relevant reasoning patterns and injects them into the input as explicit memory, trading a small parallel prefill overhead for a significant reduction in serial generation steps.

By conditioning on M, the model generates a compressed reasoning chain z', where $|z'| \ll |z|$.

Optimization Objective. The goal is to maximize inference speedup without sacrificing accuracy. We formulate this as the problem of minimizing a compound loss function:

$$\min_{M,z'} \mathcal{J} = |z'| + \gamma \cdot |M| + \lambda \cdot \mathcal{L}_{\text{perf}}, \qquad (4)$$

where $|z'|$ and $|M|$ represent the generation length and context expansion length, respectively. The coefficient $\gamma = \tau_{pre}/\tau_{gen}$ denotes the *Hardware Trade-off Ratio*. Crucially, since parallel prefilling is significantly cheaper than serial decoding ($\tau_{pre} \ll \tau_{gen}$), we have $\gamma \ll 1$, which mathematically justifies expanding context $|M|$ to reduce generation $|z'|$. Finally, $\mathcal{L}_{\text{perf}}$ quantifies performance degradation (e.g., KL divergence), weighted by λ.

This formulation defines the *Context-Generation Substitution Law*, optimizing the balance between hardware efficiency and logical accuracy.

3 Memory-Augmented Compression

3.1 Cognitive Memory View of LLM Reasoning

We conceptualize LLM reasoning through a cognitive memory view, and distinguish three memory states where information flow dictates inference efficiency.

Implicit Memory ($\mathcal{M}_I$): Pre-trained weights (θ) embodying *latent* knowledge. Accessing this static repository entails deep computational traversal, analogous to recalling long-term memories.

Explicit Memory ($\mathcal{M}_E$): The input context ($\mathcal{L}_{ctx}$) serving as *manifest* memory. Whether populated via few-shot examples or retrieval, this information is directly accessible via parallel attention during the low-cost Prefill phase (t_p).

Working Memory ($\mathcal{M}_W$): The generated CoT sequence ($\mathcal{L}_{gen}$) and KV cache constitute *dynamic* working memory. Its sequential construction (t_g) represents the primary computational bottleneck.

The "Awakening" Cost. Traditional CoT builds Working Memory by computationally "awakening" Implicit Memory:

$$\mathcal{M}_I \xrightarrow[\text{High Cost } (t_g)]{\text{Generation}} \mathcal{M}_W \qquad (5)$$

This forces the model to re-derive known premises at a high computational price. Our objective is to circumvent this by shifting the information source to the cheaper Explicit Memory.

3.2 Empirical Observations and Analysis

To validate the feasibility of substituting generation with context, we conducted a preliminary study analyzing the interplay between Working Memory ($\mathcal{M}_W$) reduction and Explicit Memory ($\mathcal{M}_E$) expansion.

Observation 1: Cost of Compression. Experiments reveal that aggressively compressing CoT (reducing $\mathcal{L}_{gen}$) in isolation leads to severe degradation (Figure 3a), confirming that $\mathcal{M}_W$ acts as essential cognitive scaffolding.

Observation 2: Memory Compensation. Crucially, this loss is reversible. Injecting prior reasoning examples into the prompt (expanding $\mathcal{M}_E$) fully restores accuracy (Figure 3b). This demonstrates a *memory compensation effect*: the deficit from reduced $\mathcal{M}_W$ is offset by augmented $\mathcal{M}_E$.

Theoretical Analysis: Computational Trade-off We model inference latency ($\mathcal{T}$) based on hardware

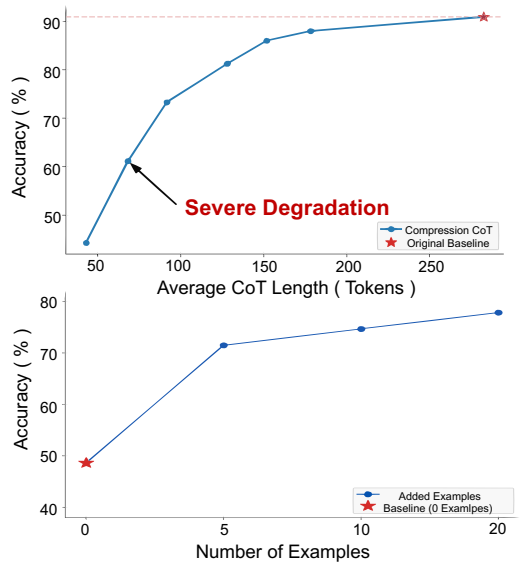

Figure 3: Explicit memory prevents the reasoning collapse caused by aggressive model compression.

asymmetry ($t_g \gg t_p$):

$$\mathcal{T} \approx |\mathbf{X}| \cdot t_p + |z| \cdot t_g \tag{6}$$

We operationalize the *Context-Generation Substitution Law* by injecting *relevant reasoning memory* into $\mathcal{M}_E$, incurring a marginal prefill overhead Δ_{in}. This injected serves acts as a scaffold, inducing the model to bypass redundant steps, reducing generation by Δ_{out}. The condition for net latency reduction ($\Delta\mathcal{T} < 0$) is:

$$\underbrace{\Delta_{out} \cdot t_g}_{\text{Generation Gain}} > \underbrace{\Delta_{in} \cdot t_p}_{\text{Prefill Cost}} \tag{7}$$

Rearranging yields the *Efficiency Frontier Condition*:

$$\frac{\Delta_{out}}{\Delta_{in}} > \frac{t_p}{t_g} \approx \epsilon \tag{8}$$

Here, ϵ (typically < 0.02) is the hardware cost ratio. This implies a favorable trade-off: effective compression (Δ_{out}) need only be a fraction of added context (Δ_{in}). Empirically, we achieve $\Delta_{out} \gg \Delta_{in}$, well above this bound.

3.3 The RIC Framework

Guided by the efficiency analysis above, we propose the **Retrieval-Injection-Compression (RIC)** framework. The core philosophy is to substitute the high-latency "Awakening" process (generation) with low-latency "Context" processing.

Algorithm 1 outlines the overall workflow. Unlike traditional pipelines that treat retrieval and generation as separate stages, RIC tightly couples them: the retrieved memory serves as a *necessary constraint* that forces the generation to be compressed.

Algorithm 1: RIC

Input : Query x, Memory Bank $\mathcal{B}$, Model Φ, Policy π
Output : Answer y, Compressed Chain z_{short}

Phase 1: Memory Retrieval ($\mathcal{R}$)
$M \leftarrow \text{Retrieve}(x, \mathcal{B}, k)$;
Phase 2: Context Injection ($\mathcal{I}$)
$\mathcal{I}_{sys} \leftarrow \text{FormatPrompt}(M)$;
$\mathcal{M}_E \leftarrow [\mathcal{I}_{sys}; x]$;
Phase 3: Compression Execution ($\mathcal{C}$)
$z_{short}, y \leftarrow \Phi(\mathcal{M}_E, \text{mode} = \pi)$;
return y, z_{short}

3.4 Implementation

Our framework comprises three key phases, designed to significantly enhance model inference efficiency without compromising accuracy through a "retrieve-inject-compress" paradigm.

Phase I: Memory Retrieval ($\mathcal{R}$) To precisely retrieve a "cognitive scaffold" that shares a logical structure with the query, we construct a memory bank $\mathcal{B}$ and employ a dual-filter strategy: first, *Label-Based Matching* quickly pinpoints the problem domain, followed by *Semantic Similarity* to select the most relevant solution templates M.

Phase II: Context Injection ($\mathcal{I}$) We leverage the fact that the Transformer architecture's parallel prefill is significantly faster than token-by-token generation ($t_p \ll t_g$). The retrieved templates M are formatted into a Scaffold(M) that explicitly demonstrates *how to skip steps*, which is then efficiently injected into the model's explicit memory $\mathcal{M}_E$. The computational overhead introduced by this process, Δ_{in}, is negligible.

Phase III: Compression Execution ($\mathcal{C}$) Guided by the scaffold, the model executes compression via two paradigms:

- **Instruction-Based (Training-Free):** Enforces concise output through system prompts.
- **Training-Based:** Employs methods like *TokenSkip*, where the model internalizes compression logic through fine-tuning.

In the second paradigm, the explicit memory $\mathcal{M}_E$ acts as a crucial "stabilizer," allowing the model to be compressed at a much higher pruning rate (γ) while effectively avoiding the semantic collapse

common in standard pruning methods.

4 Experiments

4.1 Experimental Setup

Datasets and Models We conduct experiments on two mathematical reasoning benchmarks: GSM8K (Cobbe et al., 2021) and MATH-500 (Hendrycks et al., 2021). As reasoning backbones, we employ Qwen2.5-7B-Instruct and LLaMA-3.1-8B-Instruct, selected for their strong instruction-following capabilities.

Baselines We compare our method against three distinct categories: (1) Standard CoT, the uncompressed upper bound; (2) TokenSkip, a training-based compression method evaluated at ratios of 0.2–0.4; and (3) Chain of Draft (CoD), a prompt-based baseline limiting steps to 5 words. We also include ablation variants of our method: QueryMatching (semantic retrieval) and TagsMatching (label-based retrieval).

Implementation and Metrics For memory retrieval, we utilize Qwen-Text-Embedding-v2 with ChromaDB. All inference runs use greedy decoding (temperature=0) with a maximum generation length of 512 tokens. Performance is evaluated via Accuracy, Average CoT Length, Compression Ratio ($1 - |z_{ours}|/|z_{base}|$), and Latency (measured on a single NVIDIA RTX 4090 GPU).

4.2 Main Results

The Context-Generation Trade-off Our results reveal the quantitative trade-offs of the Context-Generation Substitution Law. We consistently find that a small, fixed context overhead (e.g., *~1350 tokens* for our 10-shot memory) enables a large, relative reduction in generation length, typically *reducing required generation steps by 75% (a 4x factor)*. Crucially, this substitution is not a trade-off against accuracy; instead, it serves as a foundation for performance *gains*, as detailed below.

Performance This principle translates into superior performance in a fair comparison at the same compression level (~76%). Under this aggressive setting, the performance of the pruning-based TokenSkip ($\gamma = 0.2$) *degrades sharply*, dropping to 61.2% accuracy on GSM8K and *25.2%* on MATH-500. In sharp contrast, our *TagsMatching maintains high accuracy* at *83.2%* on GSM8K (a *22-point margin*) and demonstrates exceptional robustness on MATH-500 (58.0%). This empirically validates

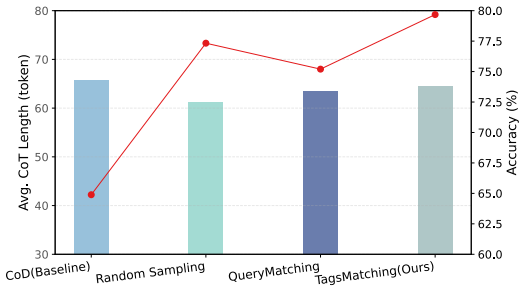

Figure 4: Ablation study on retrieval strategies. The figure compares accuracy (blue bars) against inference cost (red line) for different methods.

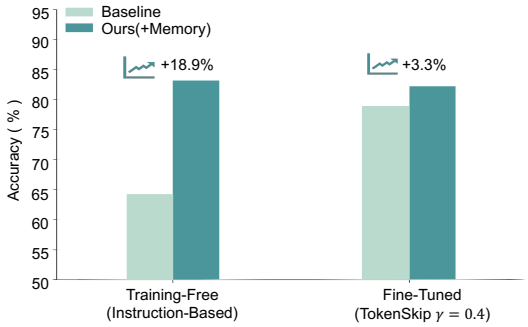

Figure 5: Our memory-augmented approach (Dark Teal) is compared against baselines (Light Mint) in both Training-Free and Fine-Tuned.

that while pruning tokens can irreparably damage the reasoning chain, our Explicit Memory acts as a vital cognitive scaffold.

Latency-Accuracy Profile The ~75% generation reduction translates to substantial KV-cache savings. Regarding latency, while our approach requires a longer prefill phase (1.57s) compared to TokenSkip-0.2 (0.48s), this upfront temporal investment yields a disproportionate return in accuracy. The result is a highly favorable point in the speed-accuracy space: a 22% absolute accuracy gain for an additional second of latency, establishing our framework as *a leading solution* for high-precision compressed reasoning.

4.3 Ablation Study

To dissect the contribution of each component in our Memory-Augmented paradigm, we conduct component-wise ablation studies.

Impact of Retrieval Strategy (Relevance) We evaluate memory selection strategies on GSM8K using Qwen2.5-7B with a fixed size of $k = 5$ (Figure 4). The *CoD* baseline's sharp decline to 64.29% confirms the necessity of memory augmen-

Method	GSM8K				MATH-500			
	Acc	Len	Lat(s)	Ratio	Acc	Len	Lat(s)	Ratio
Standard CoT	90.97	282.5	1.49	0.0%	48.40	438.9	2.66	0.0%
TokenSkip ($\gamma = 0.4/0.6$)	78.92	122.2	0.54	56.7%	37.40	272.3	2.44	37.8%
TokenSkip ($\gamma = 0.3/0.5$)	69.21	94.0	0.42	66.7%	31.00	224.8	1.98	48.6%
TokenSkip ($\gamma = 0.2/0.4$)	56.33	84.0	0.48	70.2%	25.20	183.9	1.56	42.0%
CoD (Zero-shot)	64.89	65.8	0.28	76.7%	29.80	57.7	1.06	58.0%
QueryMatching	77.90	66.4	1.32	76.5%	53.80	210.8	2.56	51.9%
TagsMatching (Ours)	**83.20**	**68.7**	1.57	**75.7%**	**58.00**	**224.6**	3.56	**48.8%**

Table 1: Main Results comparison. We observe significant accuracy gains using our method. Green numbers denote absolute percentage point improvements over QueryMatching.

tation. Counter-intuitively, standard *QueryMatching* (75.20%) underperforms *Random Sampling* (77.33%), suggesting that surface-level semantic similarity introduces distractors that mislead reasoning. In contrast, our *TagsMatching* filters this noise, achieving the highest accuracy of *79.68%*. Notably, while scaling to $k = 10$ yields our peak performance of *83.16%* (main results), the superiority of structural alignment remains evident even at this smaller retrieval size.

Method	Accuracy (%)	Avg. Length
Original Verbose CoT	90.97	282.45
Standard Compressed CoT†	61.18	68.08
Memory-Augmented CoT (Ours)	**83.16**	**68.70**

†Implemented using TokenSkip ($\gamma = 0.2$) without memory module.

Table 2: Ablation on Memory Format. Evaluating the impact of memory at similar compression levels.

Impact of Memory Representation (Format) We analyze the necessity of our compressed memory format by comparing it against the memory-free baseline. As shown in Table 2, simply enforcing brevity (Standard Compressed CoT) leads to a reasoning collapse, dropping accuracy from 90.97% to 61.18%. Crucially, our *Memory-Augmented* approach rescues this performance to 83.16% (+21.98%) while maintaining a nearly identical token budget (68.7 vs. 68.08 tokens). This result validates that our compressed memory serves as a critical *dual signal*: it provides the necessary logical guidance (content) while simultaneously demonstrating the desired concise format (style), thereby maximizing computational leverage without sacrificing reasoning capability.

Impact of Execution Strategy (Universality) To verify the universality of our paradigm, we evaluate it across two distinct execution modalities using Qwen2.5-7B (Figure 5). First, in the *Training-Free* setting, our method rescues the collapsed *CoD* baseline, delivering a substantial gain of *+18.9%* (64.29% $\rightarrow$ 83.16%). Second, in the *Fine-Tuned* setting (using *TokenSkip* with a strict compression ratio $\gamma = 0.4$), injecting our memory module yields a further improvement of **+3.26%** (78.92% $\rightarrow$ 82.18%). This demonstrates that our approach is implementation-agnostic and serves as a versatile optimization orthogonal to specific compression mechanisms.

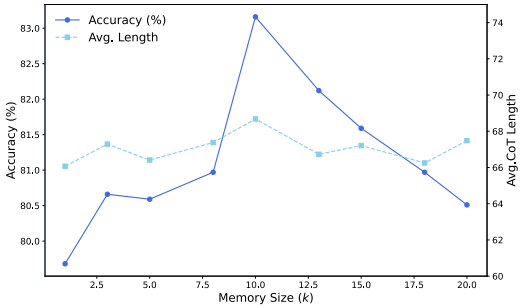

Figure 6: The figure shows the impact of memory size (k). Accuracy (solid line) peaks at $k = 10$, achieving a balance between context and noise. Meanwhile, the stable generation length (dashed line) confirms the method's robustness in efficiency regardless of context size.

4.4 Analysis

To provide a comprehensive understanding of the properties and boundaries of our framework, we performed a series of analytical experiments.

Sensitivity to Memory Size (k) We analyze the retrieval size $k \in [1, 20]$ in Figure 6. Accuracy exhibits an inverted U-shaped trend, peaking at $k = 10$ (83.16%), which represents the optimal trade-off between context sufficiency and noise interference. Crucially, the generation length (dashed

line) remains stable regardless of k. This indicates that while explicit memory aids reasoning, it does not inflate the generation cost. Furthermore, this retrieval mechanism is orthogonal to token-level compression. We posit that integrating techniques like *TokenDrop* or *KV Cache optimization* on the retrieved context could further reduce the memory footprint, offering a pathway to extremely efficient inference without compromising the structural guidance provided by the memory.

Scalability to Reasoning Models We extend our evaluation to the massive *Qwen3-235B-Thinking* model (Table 3). While the baseline achieves high accuracy (92%), its latency is prohibitive. Our method delivers a 2.2× speedup, slashing latency by 55% (262.2s → 117.8s) with a moderate accuracy trade-off (92% → 81%). This demonstrates our paradigm's potential to enable the practical deployment of 200B+ scale models in latency-sensitive applications.

Model	Baseline (CoD)			Ours		
	Acc.	Len.	Lat.	Acc.	Len.	Lat.
Qwen3-235B-Thinking	92	31.19	262.2	81	21.7	117.78

Table 3: Scalability Analysis. Performance of the Qwen3-235B-Thinking model on MATH.

Robustness to Decoding Strategies We evaluate the stability of our paradigm under stochastic decoding ($T \in [0, 1]$) in Figure 7. While *Standard CoT* suffers from "verbosity drift" at higher temperatures (length increasing to >300 tokens), our method demonstrates exceptional robustness. Explicit memory constraints effectively anchor the reasoning process, maintaining a stable length ($\approx$ 69 tokens) and competitive accuracy ($\sim$82%) even at $T = 1.0$. This confirms that our approach prevents hallucination sprawl, making it highly reliable for diverse deployment scenarios.

Synergy with Training-Based Compression To validate the extensibility of our framework, we integrated our Memory-Augmented paradigm directly into *TokenSkip* (a representative fine-tuning method). Figure 8 illustrates the performance comparison across varying retention rates (γ) on (a) GSM8K and (b) MATH-500. The results reveal a universal enhancement: our augmented version (Red) consistently outperforms the vanilla baseline (Blue) on both datasets. Crucially, the explicit memory effectively compensates for the semantic loss induced by aggressive pruning, acting not as a competitor but as a complementary booster that pushes the efficiency-accuracy Pareto frontier further.

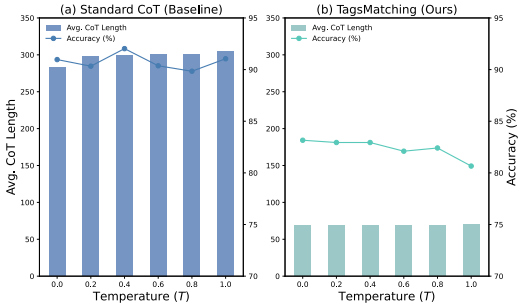

Figure 7: *Robustness to Temperature (T).* (a) *Standard CoT* becomes unstable and verbose as T increases. (b) *Ours* maintains consistent low latency and accuracy, demonstrating resilience to high-entropy sampling.

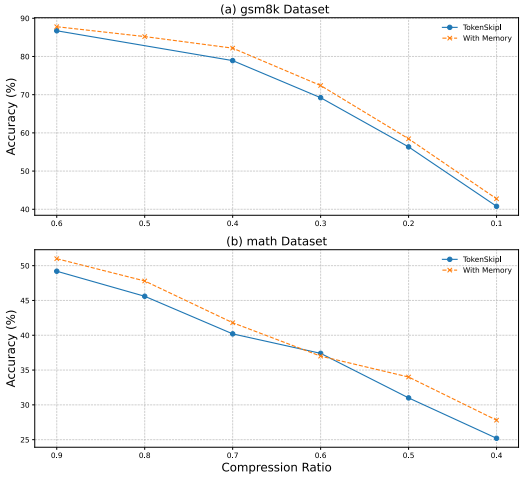

Figure 8: *Universal Enhancement on TokenSkip.* Performance comparison on (a) GSM8K and *(b) MATH-500* across varying retention rates (γ). Our method (Red) consistently lifts the accuracy curve above the vanilla TokenSkip (Blue), demonstrating that memory augmentation effectively compensates for the information loss induced by token skipping.

Case Study Table 4 illustrates the qualitative impact of our method. While the standard compressed baseline suffers from *logic collapse* (e.g., TokenSkip) due to aggressive token pruning, our method utilizes the retrieved template as a *logical scaffold*. This guidance ensures the model executes all necessary reasoning steps ("Left = Total - Part") even within a minimal token budget, effectively eliminating the hallucination seen in the baseline.

Table 4: **Case Study on GSM8K.** Comparison of reasoning behaviors. While the compressed baseline suffers from logic hallucination due to lack of context, our method successfully bridges the reasoning gap by retrieving a structurally similar template, achieving the correct answer with minimal tokens.

Input Question: Janet buys 3 packs of baseball cards. Each pack has 15 cards. She gives 1/3 of her cards to her brother. How many cards does she have left?
Standard CoT (Accurate but Costly, $\approx$ 60 tokens): First, we need to calculate the total number of cards Janet bought. She bought 3 packs, and each pack has 15 cards, so $3 \times 15 = 45$ cards in total. Next, she gives 1/3 to her brother. $45 \times (1/3) = 15$ cards given away. Finally, we subtract the cards given from the total: $45 - 15 = 30$. So Janet has 30 cards left.
Compressed Baseline (CoD) (Short but Wrong): Janet has $3 \times 15 = 45$ cards. She has 45 cards left. ***Analysis:*** *Without guidance, the model aggressively skips the "subtraction" step to satisfy the length constraint, leading to a wrong answer.*
Ours (Memory-Augmented) (Short & Correct): **[Retrieved Memory]:** *Template: Total = Unit $\times$ Num; Part = Total / Fraction; Left = Total - Part.* Total cards: $3 \times 15 = 45$. Given away: $45/3 = 15$. Left: $45 - 15 = 30$. ***Analysis:*** *Guided by the retrieved structural template, the model preserves the critical "subtraction" logic even within a tight token budget.*

5 Related Work

Context Optimization Research in this domain focuses on enhancing the utilization of input context. Techniques like *Self-Extend* (Jin et al., 2024) and *Activation Beacon* (Zhang et al., 2024) expand the effective context window, while optimized retrieval strategies (Liu et al., 2022; Ye et al., 2023) improve the relevance of in-context examples. Efficiency-oriented methods like *Recomp* (Xu et al., 2023) compress retrieved information to reduce prefill costs. However, these works primarily target the quality or capacity of the input. Unlike our approach, they do not explicitly model the *context-generation substitution* mechanism to proactively reduce the computational burden of subsequent reasoning.

Chain-of-Thought Compression Mitigating the cost of CoT is a rapidly evolving field. *Reasoning Step Offloading* (e.g., Toolformer (Schick et al., 2023)) outsources sub-tasks, while internal compression methods like *CoT Pruning* (Hou et al., 2025) and *TokenSkip* (Xia et al., 2025) eliminate redundant tokens during decoding. Post-hoc summarization techniques (Cheng and Van Durme, 2024; Zhang et al., 2025) also reduce storage overhead. In contrast to these reactive methods which often suffer from degradation under aggressive compression, our work introduces a *proactive* paradigm. By injecting high-density premises *before* generation, we utilize explicit memory as a cognitive scaffold, allowing the model to bypass redundant steps without compromising reasoning integrity.

Generation Process Acceleration This category focuses on reducing the unit cost of token generation (t_g). Foundational techniques like *Speculative Decoding* (Leviathan et al., 2023) leverage lightweight draft models for parallel verification, while model compression methods (e.g., SparseGPT (Frantar and Alistarh, 2023)) optimize architecture. Our work is orthogonal and complementary to these approaches. While they lower the *per-token latency*, our method reduces the *total token volume*, allowing for synergistic, multiplicative efficiency gains when combined.

6 Conclusion

This work introduces *Memory-Augmented Compression*, a generalizable inference paradigm that fundamentally rethinks the efficiency bottleneck of reasoning models. By substituting costly serial generation with parallel context processing, and utilizing explicit memory as a cognitive scaffold, our approach enables LLMs to bypass redundant reasoning steps without compromising logic integrity. Extensive experiments validate the effectiveness of our framework: at $\sim$76% compression, it outperforms state-of-the-art pruning methods by over 22% in accuracy on GSM8K, and surpasses the uncompressed Standard CoT by nearly 10% on MATH-500. We hope our investigations into the *context-generation substitution* mechanism will offer valuable insights for advancing efficient reasoning research and inspire future system-level optimizations.

Limitations

In this section, we discuss the potential limitations:our current investigation focuses on 7B/8B parameter models; while these are critical for efficient deployment, the scalability of the proposed *Context-Generation Substitution Law* to larger models (e.g., 70B+) remains to be empirically verified. Additionally, the fixed overhead of retrieval and prefill means our method is specifically optimized for complex reasoning tasks; for simple, short-horizon queries where generation cost is minimal, the latency benefits may diminish.

References

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

Grady Booch, Francesco Fabiano, Lior Horesh, Kiran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, et al. 2021. Thinking fast and slow in ai. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15042–15046.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

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

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pages 10323–10337. PMLR.

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

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

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

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

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd workshop on knowledge extraction and integration for deep learning architectures*, pages 100–114.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and

Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

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

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

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Veselin Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2023. Complementary explanations for effective in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4469–4484.

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

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024. Long context compression with activation beacon. *arXiv preprint arXiv:2401.03462*.

Appendix

A Experimental Setup Details

All experiments were conducted on a single NVIDIA RTX 4090 GPU (24GB) running Ubuntu 22.04 with Python 3.12 and CUDA 12.8. Our implementation is built upon key libraries including PyTorch (v2.5.1), Transformers (v4.52.4), and vLLM (v0.6.4.post1). A complete list of all package dependencies is available in the `requirements.txt` file included with our supplementary code. For decoding, we used a deterministic setting across all experiments with a `temperature` of 0.0 and `top_p` of 1.0, and set `max_new_tokens` to 512. Our Memory-Augmented Compression method introduces two key hyperparameters: the number of injected patterns, k, was set to 10, and the cosine similarity threshold for retrieving these patterns was set to 0.6.

B Prompt Templates

We present the full templates used for our baseline and proposed methods below.

Standard CoT Prompt

```
Please reason step by step, and put
your final answer within \boxed{}.
```

CoD Prompt

```
Think step by step, but only keep
a minimum draft for each thinking
step, with 5 words at most. Put
your final answer within \boxed{}.
```

C Statement on the Use of AI Assistants

During the preparation of this work, we utilized an AI assistant (e.g., ChatGPT) for tasks such as proofreading and improving language clarity. The core ideas, experimental design, and final text were authored by the human authors, who take full responsibility for the content of this paper.

D Data and Code Availability

All datasets used in this study, including GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), are publicly available academic benchmarks. To support reproducibility, our source code will be publicly released on GitHub under the **MIT License** upon publication. Anonymized code has been submitted as supplementary material.