# MEMOSIGHT: IMPROVING THE FIDELITY OF CHAIN-OF-THOUGHT COMPRESSION WITH MEMORY AND FORESIGHT TOKENS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Long chain-of-thought (CoT) reasoning enables LLMs to solve increasingly challenging tasks, but incurs substantial inference costs as the KV cache grows with the reasoning length. Online CoT compression reduces this footprint by replacing completed reasoning steps with compact memory tokens. However, existing methods compress every step into a *fixed, length-agnostic* set of memory tokens that are *positioned far from the content they summarize* and trained with *short-horizon next-token supervision*, limiting the fidelity of the compressed representation and degrading reasoning accuracy. We propose **MemoSight**, a lightweight framework that improves the fidelity of compressed reasoning while accelerating inference, through three designs targeting *distinct sources* of information loss. 1) **Adaptive Memory Allocation** allocates memory tokens in proportion to the length of each reasoning step, providing longer steps with greater capacity to preserve their information. 2) **Uniform Positional Layout** assigns memory-token position IDs uniformly within the span of the step they summarize, shortening their positional distance to the content and encouraging them to cover different parts of the step. 3) **Foresight Multi-Token Prediction** introduces foresight tokens that predict reasoning tokens farther ahead rather than only the next one, providing longer-horizon supervision that encourages richer memory representations. Across four reasoning benchmarks and three model families, MemoSight improves average reasoning accuracy by 2.6 points over LightThinker at comparable peak KV-cache usage. Moreover, the same foresight tokens used for training supervision are also used to generate draft predictions for speculative decoding, accelerating inference by 24.4%–49.4% across model families. Together, these results establish a better accuracy–efficiency trade-off for CoT compression.

## 1 INTRODUCTION

Long chain-of-thought (CoT) reasoning has become a central mechanism for improving large language models (LLMs) on challenging tasks such as mathematics, science, and multi-step problem solving Zhao et al. (2023); Guo et al. (2025); Jaech et al. (2024). However, the benefits of long CoT come at a steep inference cost: in Transformer-based architectures Vaswani et al. (2017), the Key–Value (KV) cache grows linearly with the number of generated tokens, so the memory and latency of decoding scale with the length of the reasoning trace. As reasoning chains grow to thousands of tokens, this overhead becomes a primary obstacle to deploying long-CoT models in practice Arora & Zanette (2025).

CoT compression has emerged as a promising remedy. Rather than retaining the full reasoning trace, these methods condense completed thought steps into a small, fixed number of compact "memory tokens" and discard the original tokens' KV cache, so that subsequent decoding is conditioned only on the compressed representation Chang et al. (2024); Zhang et al. (2025a). This substantially reduces peak KV-cache usage. The central difficulty, however, is *fidelity*: whether the compressed memory preserves enough information from the original trace to support subsequent reasoning. In practice, insufficient fidelity translates into a noticeable drop in reasoning accuracy, precisely on the long, information-dense chains where compression matters most.



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.

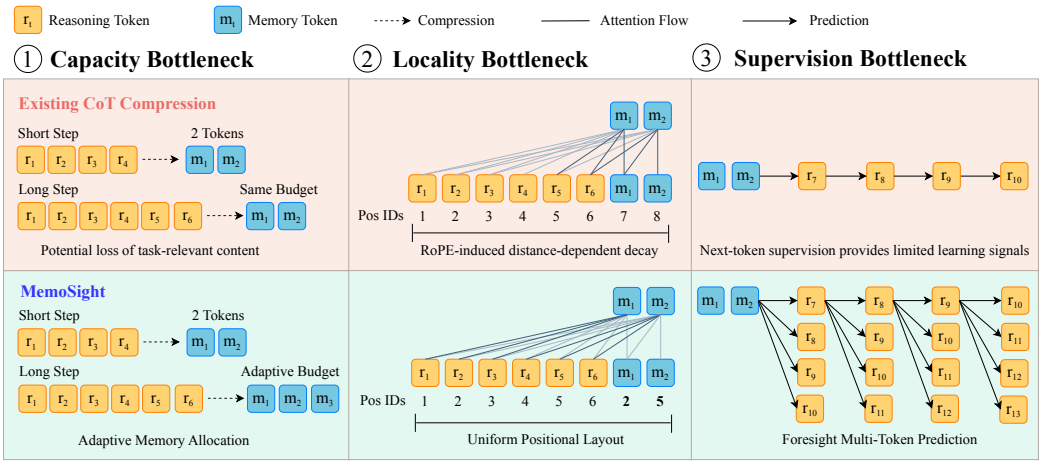



Figure 1: Motivation and overview of MemoSight. Existing CoT compression methods suffer from three key limitations: capacity, locality, and supervision bottlenecks (top). MemoSight addresses them through AMA, UPL, and MTP, respectively (bottom).

We argue that this fidelity gap arises from three distinct bottlenecks that existing methods leave largely unaddressed, as illustrated in the upper part of Figure 1. **(i) Capacity:** a fixed number of memory tokens is allocated to every reasoning step regardless of its length, so long steps are forced through the same bottleneck as short ones and lose disproportionately more information. **(ii) Locality:** memory tokens are assigned position IDs after the reasoning step, placing them far in positional space from the content they summarize and weakening their ability to capture it effectively. **(iii) Supervision:** standard next-token prediction provides only short-horizon supervision for learning memory representations that support subsequent reasoning. Each prediction target constrains only the immediate continuation, without explicitly encouraging the model to preserve information useful farther into the reasoning trajectory.

We introduce **MemoSight**, a lightweight CoT compression framework that targets all three bottlenecks without modifying the backbone LLM. MemoSight addresses these bottlenecks with three corresponding components, as illustrated in the lower part of Figure 1. **Adaptive Memory Allocation (AMA)** allocates memory tokens in proportion to the length of each reasoning step, so that longer steps receive greater capacity to preserve their content. **Uniform Positional Layout (UPL)** uniformly places memory-token position IDs within the reasoning step, reducing positional distance and encouraging each token to capture a distinct local region, collectively forming a more faithful representation of the step. **Foresight Multi-Token Prediction (MTP)** introduces *foresight tokens* that predict reasoning tokens farther ahead rather than only the next one, extending the supervision horizon during training. This longer-horizon signal encourages memory representations that better preserve information useful for subsequent reasoning, improving compression fidelity. As a by-product, these same foresight tokens act as draft predictions at inference time, enabling speculative decoding Cai et al. (2024a); Li et al. (2024b) that further accelerates generation.

We evaluate MemoSight on four reasoning benchmarks across three model families. MemoSight raises average reasoning accuracy by 2.6 points over LightThinker across three model families at comparable peak KV-cache usage, substantially narrowing the gap to the uncompressed model. Ablation studies confirm that all three components contribute to this improvement. Moreover, its speculative decoding accelerates generation by 24.4%–49.4% across model families without changing the generated outputs. Together, these results establish a better efficiency–performance trade-off for CoT compression.

## 2 Related Work

We review prior work related to MemoSight along four directions: prompt compression, KV-cache pruning and CoT compression, multi-token prediction, and speculative decoding.

**Prompt Compression.** Prompt compression methods (Jiang et al., 2023; Ge et al., 2023; Zhao et al., 2024; Chang et al., 2024; Li et al., 2024c; Mu et al., 2024; Li et al., 2024d; Wang et al., 2024; Tang et al., 2025; 2026a;b;c; Zhao et al., 2026a;b; Liu et al., 2025) condense the model *input* at the encoding stage, compressing context in a single pass before generation begins. Because CoT reasoning is generated incrementally during decoding and is not available ahead of time, these techniques are inapplicable to compressing the reasoning trace itself, which motivates methods that operate *during decoding*.

**KV-Cache Pruning and CoT Compression.** Methods for reducing the memory footprint of reasoning traces broadly fall into pruning and compression. *Pruning* methods selectively discard KV-cache entries using heuristics based on attention, recency, or structural tokens (Zhang et al., 2023; Xiao et al., 2024; Chen et al., 2025), but discarded information is no longer accessible to subsequent reasoning. *Compression* methods instead condense information before eviction, either by rewriting completed reasoning into shorter text (Yang et al., 2025b; Yan et al., 2026; Kontonis et al., 2026) or by encoding it into compact latent representations (Zhang et al., 2025a; Pang et al., 2024; Liu et al., 2026). Among the latter, LightThinker (Zhang et al., 2025a) compresses each reasoning step into a small set of memory-token hidden states and serves as our closest baseline. MemoSight builds on this representational compression paradigm while improving compression fidelity through adaptive memory allocation, positional layout, and longer-horizon supervision.

**Multi-Token Prediction and Speculative Decoding.** Multi-token prediction (MTP) trains models to predict multiple future tokens in parallel and can improve both representation learning and decoding efficiency (Gloeckle et al., 2024; Liu et al., 2024). Conventional MTP methods typically introduce dedicated prediction heads or Transformer blocks, while recent special-token approaches provide a lightweight alternative (Gerontopoulos et al., 2025). MemoSight adopts a lightweight special-token design but integrates MTP with CoT compression, using foresight tokens to provide longer-horizon supervision for memory representations. At inference time, these predictions further serve as self-drafts for speculative decoding (Cai et al., 2024a; Li et al., 2024b), reducing sequential decoding steps without an auxiliary draft model.

# 3 MEMOSIGHT

## 3.1 PRELIMINARY: COT COMPRESSION

**KV-Cache Cost.** Given a prompt $P = [p_1, \ldots, p_{|P|}]$, a reasoning LLM autoregressively generates a reasoning trace $R = [r_1, \ldots, r_{|R|}]$ according to $p_\theta(r_t \mid P, r_{<t})$. A Transformer caches the key–value (KV) states of preceding tokens during decoding, so the cache grows linearly with the reasoning length, making long CoT increasingly costly in both memory and decoding latency.

**Compression at inference time.** CoT compression partitions the reasoning trace into steps $R_1, \ldots, R_n$ and encodes each completed step into a small set of *memory tokens*. After step $R_i$ completes, a fixed number $l$ of memory tokens $M_i = [m_1, \ldots, m_l]$ and an output token $o_i$ are appended, after which the raw KV states of $R_i$ are evicted and subsequent decoding relies only on the surviving compressed context. With the first $i$ steps compressed, the cache contains the prompt $P$, the $l+1$ tokens retained for each compressed step, and the current uncompressed step $R_{\text{cur}}$, totaling approximately $|P| + i(l+1) + |R_{\text{cur}}|$ tokens. Thus, its size scales with the number of reasoning steps rather than the full trace length.

**Training by supervised fine-tuning.** CoT compression is learned through supervised fine-tuning on a constructed sequence with a customized attention mask (Figure 2). For a trajectory $R = [R_1, \ldots, R_n]$, the training sequence interleaves memory and output tokens after each reasoning step:

$$\mathcal{S} = [P, R_1, M_1, o_1, R_2, M_2, o_2, \ldots, R_n].$$

Once a step is compressed, subsequent reasoning tokens cannot attend to its raw tokens and can access earlier reasoning history only through the retained memory and output tokens, forcing historical information through the compressed representation. Supervision is provided by a next-token prediction loss $\mathcal{L}_{\text{NTP}}$ on the reasoning and output tokens; the memory tokens receive no direct target and are optimized only through gradients back-propagated from later tokens that attend to them.

### 3.2 IMPROVING COMPRESSION FIDELITY

The fidelity of CoT compression hinges on three design choices for memory tokens: *how many* to allocate, *where* to place them in position space, and *what signal* trains them, corresponding to the capacity, locality, and supervision bottlenecks identified in Section 1. MemoSight addresses these choices with AMA, UPL, and MTP, respectively, without modifying the backbone architecture. Throughout, we write $R_i = [r_1^i, \ldots, r_{|R_i|}^i]$ for the reasoning tokens of step $i$ and $\rho_t^i$ for the position ID of $r_t^i$.

**Adaptive Memory Allocation (AMA).** The *capacity* bottleneck arises from assigning the same memory budget to every step, causing longer steps to undergo more aggressive compression than shorter ones. Instead, AMA sizes each memory block in proportion to the length of the step it summarizes. Given a compression ratio $c$ shared across steps, step $R_i$ receives

$$l_i = \left\lceil |R_i|/c \right\rceil \tag{1}$$

memory tokens $M_i = [m_1, \ldots, m_{l_i}]$, in place of the constant $l$ used in standard CoT compression. By allocating greater memory capacity to longer steps, AMA alleviates the capacity bottleneck of fixed allocation while maintaining a consistent compression ratio across steps.

**Uniform Positional Layout (UPL).** The *locality* bottleneck arises because standard CoT compression appends memory tokens after each reasoning step and assigns them fresh, consecutive position IDs. Specifically, the $k$-th memory token of step $R_i$ receives

$$\rho(m_k) = \rho_{|R_i|}^i + k, \tag{2}$$

placing its position beyond the span of the step it summarizes. Under rotary position encoding (Su et al., 2024), this layout can weaken locality both when memory is *written*, as memory tokens are positionally distant from earlier parts of the step, and when it is later *read*, as their positions no longer reflect where the compressed content originally appeared.

UPL keeps the sequence order unchanged but reassigns memory-token position IDs to lie *within* the positional span of the step they summarize. Given the $l_i$ memory tokens allocated by AMA (Eq. 1), we divide $R_i$ uniformly into $l_i$ segments of width $s_i = |R_i|/l_i$ and assign each memory token the position ID at the discretized center of one segment:

$$\rho(m_k) = \rho_{1+\lfloor (k-\frac{1}{2})s_i \rfloor}^i, \qquad k = 1, \ldots, l_i. \tag{3}$$

Each memory token is therefore positionally close to a distinct region of the step, while the memory tokens collectively preserve a uniform positional coverage of the compressed content, alleviating the locality bottleneck during both memory writing and subsequent reading.

**Foresight Multi-Token Prediction (MTP).** The *supervision* bottleneck arises because memory tokens are shaped only through next-token prediction, which provides a short-horizon signal for learning the compressed representation. MTP extends this supervision horizon by introducing *foresight tokens* that predict reasoning tokens farther ahead.[1]

Specifically, we insert a foresight token $f$ immediately before each reasoning token $r_t^i$ and train it to predict $r_{t+\delta}^i$:

$$p_\theta\big(r_{t+\delta}^i \,\big|\, \mathcal{C}_{<t}^i,\ f\big), \qquad \delta \in \{0, \ldots, d\}, \tag{4}$$

where $d$ is the maximum foresight horizon and $\mathcal{C}_{<t}^i$ consists of the prompt $P$, the memory and output tokens of earlier steps $(M_{<i}, o_{<i})$, and the current reasoning prefix $r_1^i, \ldots, r_{t-1}^i$. Raw reasoning tokens from earlier steps $R_{<i}$ are excluded, so historical information is available only through their compressed memory.

When $\delta = 0$, the foresight token predicts $r_t^i$ from the prefix ending at $r_{t-1}^i$, recovering standard next-token prediction. Larger offsets require farther-ahead prediction from the same context, extending the supervision horizon. By providing longer-horizon supervision, MTP encourages memory tokens to preserve information useful for longer-range continuation, alleviating the supervision bottleneck and yielding a more informative compressed representation.

[1] We add a single foresight token $f$ to the vocabulary, shared across all positions, so MTP introduces only one additional embedding vector and leaves the backbone architecture unchanged.

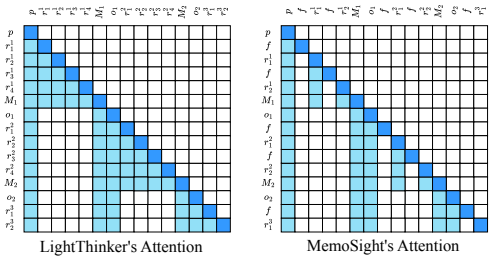



Figure 2: Attention patterns of LightThinker and MemoSight. MemoSight introduces isolated foresight-token branches for MTP.

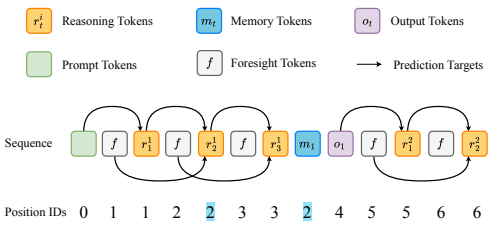



Figure 3: Simplified training sequence. Foresight tokens predict future reasoning tokens while memory tokens follow UPL position IDs.

### 3.3 Training

MemoSight is trained in a single supervised fine-tuning run by extending the standard CoT-compression setup in Section 3.1 with AMA, UPL, and foresight-token supervision.

**Training sequence.** As illustrated by the simplified training sequence in Figure 3, we augment each step with foresight tokens and size its memory block using AMA. Writing $\tilde{R}_i = [f, r_1^i, f, r_2^i, \ldots, f, r_{|R_i|}^i]$ for the foresight-augmented step, the training sequence is

$$\mathcal{S} = [\, P,\ \tilde{R}_1, M_1, o_1,\ \tilde{R}_2, M_2, o_2,\ \ldots,\ \tilde{R}_n \,], \qquad |M_i| = l_i.$$

**Supervision targets.** As shown in Figure 3, reasoning tokens retain standard next-token targets along the original reasoning chain, despite the inserted foresight tokens. The foresight token preceding $r_t^i$ predicts $r_{t+\delta}^i$, where $\delta$ is the offset sampled for that training example; its loss is masked when $t + \delta > |R_i|$. Each output token $o_i$ predicts the first token of the next step, $r_1^{i+1}$. Memory tokens receive no direct supervision and are optimized only through gradients from subsequent tokens that attend to them.

**Position-ID assignment.** Figure 3 also illustrates the position-ID assignment. Reasoning and output tokens retain their consecutive position IDs, with $o_i$ placed immediately after the last token of $R_i$ (i.e., $\rho_{|R_i|}^i + 1$). Memory tokens follow UPL (Eq. 3), with their position IDs distributed uniformly over the span of $R_i$. A foresight token targeting $r_{t+\delta}^i$ is assigned $\rho_{t+\delta-1}^i$, the position immediately preceding its target, so that it predicts from the same position as standard autoregressive prediction.

**Attention mask.** Figure 2 illustrates the unified attention pattern used during MemoSight training. Reasoning and output tokens attend causally to the prompt $P$, earlier memory and output tokens $(M_{<i}, o_{<i})$, and the preceding reasoning tokens of the current step, while raw reasoning tokens from previous steps are masked out. Memory tokens attend to the same compressed history together with the completed current step $R_i$ that they compress. Each foresight token attends to the compressed context and itself (Eq. 4), while remaining invisible to all other tokens. Thus, foresight tokens form isolated predictive branches without perturbing the main reasoning chain.

**Optimization objective.** A next-token loss $\mathcal{L}_{\text{NTP}}$ is applied to the reasoning and output tokens, and a multi-token loss $\mathcal{L}_{\text{MTP}}$ to the foresight tokens; memory tokens carry no loss term. The overall objective is

$$\mathcal{L}_{\text{MemoSight}} = \lambda\, \mathcal{L}_{\text{NTP}} + (1 - \lambda)\, \mathcal{L}_{\text{MTP}}, \tag{5}$$

where $\lambda$ balances next-token and foresight supervision.

### 3.4 Inference with Speculative Decoding

At inference time, MemoSight follows an accelerate-then-compress procedure. During each reasoning step, foresight tokens are reused for speculative decoding; once the step $R_i$ completes, memory

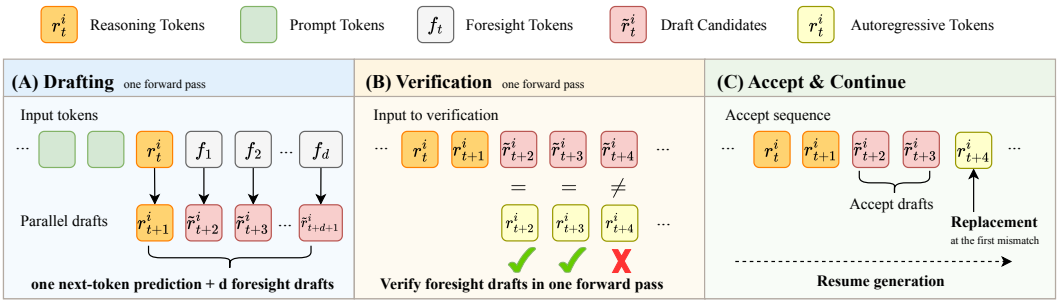



Figure 4: Inference with foresight-based speculative decoding: **(A)** parallel drafting, **(B)** parallel verification, and **(C)** prefix acceptance with autoregressive replacement at the first mismatch.

tokens $M_i$ are appended according to AMA and UPL, followed by an output token $o_i$. The raw KV states of $R_i$ are then evicted, and generation continues from the compressed cache.

Figure 4 illustrates the speculative decoding procedure. During drafting (A), the standard next-token prediction produces $r^i_{t+1}$, while $d$ foresight tokens produce farther-ahead drafts $\tilde{r}^i_{t+2}, \ldots, \tilde{r}^i_{t+d+1}$ in the same forward pass. During verification (B), the foresight drafts are compared against standard autoregressive predictions in a single forward pass, and the longest matching prefix is accepted. The first candidate $r^i_{t+1}$ requires no verification because it is generated directly from the actual context.

At the first mismatch (C), the corresponding autoregressive prediction from the verification pass replaces the rejected draft, and decoding resumes from the resulting prefix. This verification preserves exactly the output of standard greedy decoding while reducing the number of sequential decoding steps. No auxiliary draft model or inference-specific parameters are required, as the trained foresight mechanism is reused directly for drafting.

# 4 Experiments

## 4.1 Experimental Settings

**Baselines.** We evaluate MemoSight on three mainstream LLMs—Qwen2.5-7B Hui et al. (2024), Qwen3-8B Yang et al. (2025a), and Llama-3.1-8B Dubey et al. (2024)—against six representative baselines in three categories: (1) **Off-the-Shelf Models**: the base instruction-tuned model with CoT prompting (Standard) and the DeepSeek-R1 distilled model (Distill-R1) Guo et al. (2025). (2) **Trained Reasoning Models**: Vanilla, which directly fine-tunes Distill-R1 on Bespoke-Stratos-17k Labs (2025) without compression, and LightThinker Zhang et al. (2025a), which introduces CoT compression; MemoSight follows the same training setup. (3) **Training-Free Compression Methods**: H2O Zhang et al. (2023) and SepLLM Chen et al. (2025), which reduce KV-cache size via post-hoc pruning heuristics and are applied to the Vanilla model for fair comparison. Details of all baselines are provided in Appendix B.1.

**Datasets and Metrics.** We evaluate on four datasets covering diverse reasoning capabilities: mathematical reasoning (GSM8K Cobbe et al. (2021)), algorithmic and compositional reasoning (BBH Suzgun et al. (2023)), multi-domain knowledge and reasoning (MMLU Hendrycks et al. (2020)), and expert-level scientific reasoning (GPQA Rein et al. (2024)). We measure task performance by accuracy and computational efficiency by the peak number of retained KV-cache tokens (Peak) and generation throughput in tokens per second (Speed).

**Implementation Details.** All models are trained for 5 epochs with a batch size of 64. The constructed CoT-compression sequence, including reasoning, memory, and output tokens, is truncated to at most 4096 tokens before inserting foresight tokens for MemoSight. For MemoSight, we set the NTP loss weight $\lambda = 0.7$, compression ratio $c = 5$, and foresight horizon $d = 2$. During training, we sample one offset $\delta \in 0, 1, 2$ uniformly for each example. During inference, $d = 2$ yields one standard next-token prediction and two foresight drafts in parallel at each decoding step. Evaluation uses greedy decoding with a maximum output length of 10240 tokens. Full training details are provided in Appendix B.2.

Table 1: Main results on Qwen2.5-7B, Qwen3-8B, and Llama3.1-8B. Acceleration methods are highlighted in blue, with **bold** and underlined values denoting the best and second-best results among them. For MemoSight, Speed is reported as autoregressive → speculative decoding.

| Method | GSM8K | | | MMLU | | | GPQA | | | BBH | | | AVG. | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc ↑ | Speed ↑ | Peak ↓ | Acc ↑ | Speed ↑ | Peak ↓ | Acc ↑ | Speed ↑ | Peak ↓ | Acc ↑ | Speed ↑ | Peak ↓ | Acc ↑ | Speed ↑ | Peak ↓ |
| *Qwen2.5-7B Series* | | | | | | | | | | | | | | | |
| Standard | 88.32 | 24.63 | 519 | 70.01 | 23.78 | 676 | 28.28 | 25.50 | 998 | 71.31 | 27.47 | 582 | 64.48 | 25.35 | 694 |
| Distill-R1 | 60.58 | 24.82 | 522 | 31.26 | 21.49 | 1289 | 22.22 | 19.21 | 4383 | 51.31 | 29.35 | 905 | 41.34 | 23.72 | 1775 |
| Vanilla | 90.83 | 20.69 | 1559 | 65.92 | 18.60 | 2241 | 37.37 | 15.10 | 7184 | 81.01 | 24.32 | 2212 | 68.78 | 19.68 | 3299 |
| + H2O | 91.05 | 13.85 | 1024 | 62.32 | 10.36 | 1024 | 20.20 | 9.96 | **1024** | **75.76** | 14.14 | **1024** | 62.33 | 12.08 | **1024** |
| + SepLLM | **91.28** | 11.03 | 1024 | 59.20 | 10.97 | 1024 | 12.63 | 11.30 | **1024** | 69.70 | 14.27 | **1024** | 58.20 | 11.89 | **1024** |
| LightThinker | 87.26 | 22.56 | 791 | 60.66 | 22.58 | 956 | 37.37 | 22.38 | 2276 | 66.46 | 28.13 | 1133 | 62.94 | 23.91 | 1289 |
| MemoSight | 89.84 | 22.56→**30.94** | **745** | **63.49** | 22.45→**27.54** | **900** | **40.91** | 21.75→**27.64** | 2102 | 73.13 | 28.41→**32.26** | 1059 | **66.84** | 23.79→**29.60** | 1202 |
| *Qwen3-8B Series* | | | | | | | | | | | | | | | |
| Standard | 94.69 | 10.44 | 1207 | 88.22 | 9.02 | 2059 | 55.56 | 7.21 | 5736 | 91.31 | 10.45 | 1716 | 82.45 | 9.28 | 2680 |
| Vanilla | 93.71 | 10.38 | 1264 | 86.95 | 9.67 | 1909 | 54.55 | 7.75 | 5532 | 91.52 | 10.48 | 1708 | 81.68 | 9.57 | 2603 |
| + H2O | **93.32** | 9.56 | 1024 | 83.11 | 6.90 | 1024 | 33.06 | 7.13 | **1024** | **86.80** | 9.14 | 1024 | 74.07 | 8.18 | 1024 |
| + SepLLM | 92.87 | 17.15 | 1024 | 82.77 | **18.61** | 1024 | 30.81 | 15.43 | **1024** | 78.38 | 15.82 | 1024 | 71.21 | 16.75 | 1024 |
| LightThinker | 91.43 | 11.04 | **633** | 83.54 | 11.18 | **763** | **52.53** | 10.23 | 1443 | 82.42 | 11.21 | **801** | 77.48 | 10.92 | **910** |
| MemoSight | 93.18 | 11.73→**17.23** | 685 | **84.81** | 11.34→15.32 | 874 | **52.53** | 10.53→**16.01** | 1788 | 86.46 | 11.99→**19.57** | 838 | **79.25** | 11.40→**17.03** | 1046 |
| *Llama3.1-8B Series* | | | | | | | | | | | | | | | |
| Standard | 64.52 | 21.41 | 497 | 60.18 | 19.23 | 698 | 24.75 | 21.40 | 2230 | 53.13 | 26.28 | 596 | 50.65 | 22.08 | 1005 |
| Distill-R1 | 57.62 | 22.78 | 524 | 20.93 | 17.89 | 1611 | 30.81 | 16.40 | 5640 | 29.49 | 27.65 | 1146 | 34.71 | 21.18 | 2230 |
| Vanilla | 89.31 | 18.33 | 1702 | 71.08 | 16.26 | 2800 | 37.37 | 14.17 | 6824 | 78.18 | 21.63 | 2515 | 68.99 | 17.60 | 3460 |
| + H2O | **89.31** | 11.65 | 1024 | **69.52** | 9.03 | 1024 | 23.23 | 8.51 | **1024** | **81.62** | 13.04 | 1024 | 65.92 | 10.56 | **1024** |
| + SepLLM | 88.32 | 19.74 | 1024 | 64.85 | 19.02 | 1024 | 15.66 | 19.27 | **1024** | 72.32 | 24.12 | 1024 | 60.29 | 20.54 | **1024** |
| LightThinker | 85.82 | 21.38 | **664** | 60.86 | 20.53 | **932** | **37.37** | 19.89 | 1933 | 72.93 | 26.11 | 999 | 64.25 | 21.98 | 1132 |
| MemoSight | 87.19 | 20.91→**27.91** | 750 | 64.07 | 20.26→**25.89** | 933 | 35.86 | 20.44→**26.67** | 1942 | 78.38 | 25.92→**33.45** | **866** | **66.38** | 21.88→**28.48** | 1123 |

### 4.2 Main Results

Table 1 summarizes the main results across three model families. MemoSight consistently achieves higher reasoning accuracy than LightThinker, the most relevant CoT-compression baseline, with comparable peak KV-cache usage. For example, on Qwen2.5-7B, MemoSight improves average accuracy by 3.90 points (66.84 vs. 62.94) while using slightly less peak cache (1202 vs. 1289 tokens). Overall, MemoSight achieves a better accuracy–memory trade-off than prior CoT compression.

Compared with uncompressed Vanilla, MemoSight substantially reduces peak KV-cache usage while keeping the average accuracy degradation below 3 points across all three model families. It also consistently outperforms the training-free pruning methods H2O and SepLLM, indicating that learned step compression better preserves information for subsequent reasoning.

Finally, speculative decoding improves MemoSight's average throughput by 24.4%–49.4% across the three model families. Because foresight drafts are verified against standard autoregressive predictions, it preserves exactly the output of standard greedy decoding. Sections 5 and 6 further analyze the accuracy–memory trade-off and speculative decoding behavior, respectively.

### 4.3 Generalization to Challenging Reasoning

Longer and more complex reasoning provides a stricter test of CoT compression, as information loss can accumulate over extended chains. We evaluate MemoSight on MATH500 and AIME26 with Qwen3-8B. As shown in Table 2, with $d$=2, MemoSight improves accuracy over LightThinker from 64.47 to 71.80 on MATH500 and from 13.33 to 26.67 on AIME26. Increasing to $d$=4 further raises AIME26 to 33.33 while maintaining 70.20 on MATH500. Overall, MemoSight better preserves reasoning capability and substantially narrows the gap to uncompressed Vanilla.

## 5 Understanding Compression Fidelity

We next investigate why MemoSight better preserves reasoning fidelity under compression. Following the three bottlenecks identified in Section 1, we first ablate the corresponding components and then analyze how each design affects capacity, locality, and supervision.

Table 2: Challenging reasoning results on Qwen3-8B.

| Method | MATH500 | AIME26 |
|---|---|---|
| Vanilla | 74.89 | 46.67 |
| LightThinker | 64.47 | 13.33 |
| MemoSight $d = 2$ | **71.80** | 26.67 |
| MemoSight $d = 4$ | 70.20 | **33.33** |

Table 3: Component ablation on Qwen2.5-7B across the three fidelity bottlenecks.

| Method | GSM8K | MMLU | GPQA | BBH | Avg. |
|---|---|---|---|---|---|
| MemoSight | **89.84** | **63.49** | **40.91** | **73.13** | **66.84** |
| w/o AMA | 88.17 | 61.86 | 36.32 | 65.93 | 63.07 |
| w/o UPL | 89.31 | 62.55 | 37.37 | 65.73 | 63.74 |
| w/o MTP | 88.40 | 62.79 | 38.89 | 69.90 | 64.99 |

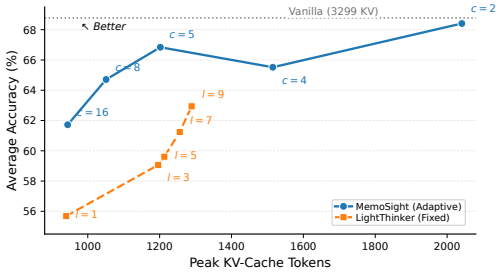



Figure 5: Accuracy-memory tradeoff.

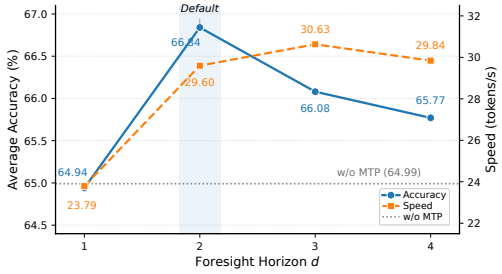



Figure 6: Foresight horizon analysis.

**Component ablation.** We first examine the contribution of the three designs. As shown in Table 3, removing AMA, UPL, or MTP reduces average accuracy by 3.77, 3.10, and 1.85 points, respectively. The degradation is consistent across the three components, with particularly large drops from removing AMA on GPQA ($-4.59$ points) and UPL on BBH ($-7.40$ points). These results show that all three designs contribute substantially to compression fidelity, motivating a closer examination of how they address capacity, locality, and supervision.

**Capacity: Accuracy–memory trade-off.** To study memory capacity, we vary the compression ratio $c$ for MemoSight and the fixed number of memory tokens $l$ per reasoning step for LightThinker. Figure 5 shows the accuracy–memory trade-off across these settings, with the uncompressed Vanilla model as a reference. MemoSight achieves a consistently better trade-off across the sweep. For example, at $c = 8$, MemoSight reaches 64.71% accuracy with only 1,051 peak KV-cache tokens, outperforming all LightThinker settings shown despite using less memory. Together with the AMA ablation in Table 3, this indicates that allocating memory according to reasoning-step length uses the available capacity more effectively than a fixed allocation.

**Locality: Memory–content alignment.** We next examine how positional layout affects the attention of memory tokens over the reasoning content they compress. Figure 7 visualizes memory-to-reasoning attention for a representative reasoning step. With UPL, successive memory tokens attend to progressively shifted regions of the reasoning span, forming a clear localized diagonal pattern. In contrast, conventional consecutive positioning yields more diffuse and irregular attention, with no clear progression across the reasoning span. We quantify this structure using the normalized offset between each memory token's most-attended reasoning position and the corresponding position under an ideal uniform coverage of the reasoning span. The offset decreases from 0.293 to 0.009, quantitatively confirming that UPL induces a much more orderly progression of attention across the reasoning content. Additional examples are provided in Appendix D.

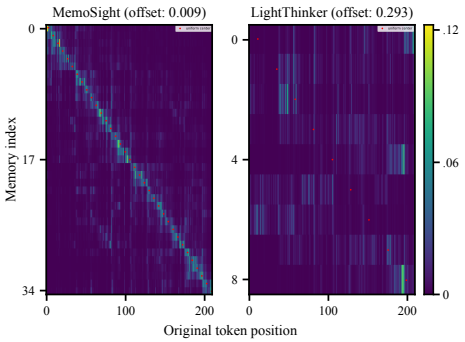



Figure 7: Attention patterns of memory tokens.

**Supervision: Foresight horizon.** We vary the maximum foresight offset $d$ to study how the supervision horizon affects compression fidelity. As shown in Figure 6, $d = 1$ achieves 64.94% average

Table 4: Foresight-token acceptance on Qwen3-8B. Each entry reports $d$=2 / $d$=4.

| Benchmark | Acceptance Rate (%) | | | | Tok./Step |
|---|---|---|---|---|---|
| | $\delta$=1 | $\delta$=2 | $\delta$=3 | $\delta$=4 | |
| GSM8K | 75.1 / 72.1 | 50.5 / 47.8 | – / 31.1 | – / 19.7 | 2.21 / 2.63 |
| MMLU | 74.6 / 73.6 | 49.3 / 48.7 | – / 30.4 | – / 18.1 | 2.20 / 2.63 |
| GPQA | 69.7 / 68.2 | 41.3 / 41.3 | – / 24.9 | – / 14.4 | 2.07 / 2.41 |
| BBH | 64.4 / 63.7 | 34.6 / 33.8 | – / 16.6 | – / 7.2 | 1.93 / 2.12 |

Table 5: Runtime breakdown on BBH with Qwen3-8B.

| Component | Time (s) | Fraction |
|---|---|---|
| Prediction | 107.9 | 48.0% |
| Verification | 109.7 | 48.8% |
| Compression | 0.9 | 0.4% |
| Other | 6.3 | 2.8% |
| Total | 224.8 | 100.0% |

accuracy, essentially matching the variant without MTP (64.99%). Extending the horizon to $d = 2$ improves accuracy to 66.84%, while further increasing it to $d = 3$ lowers accuracy to 66.08%, supporting our motivation that longer-horizon supervision is beneficial up to a moderate horizon. At $d = 4$, accuracy further decreases to 65.77%, indicating diminishing returns from extending the prediction horizon beyond $d = 2$. Meanwhile, the same foresight tokens also provide farther-ahead drafts for speculative decoding, increasing throughput from 23.79 tokens/s at $d = 1$ to around 30 tokens/s for $d = 2$–4. We therefore use $d = 2$ as the default, which provides strong compression fidelity while already delivering substantial decoding acceleration.

## 6 Analysis of Speculative Decoding

We next analyze how foresight tokens translate into lossless decoding acceleration, focusing on draft acceptance and runtime behavior.

**Foresight-token acceptance.** We first examine whether foresight tokens provide sufficiently accurate drafts to advance multiple tokens per decoding step. Table 4 reports the acceptance rate at each prediction offset $\delta$ for foresight horizons $d = 2$ and $d = 4$ on Qwen3-8B. Near-term drafts are accepted frequently, with $\delta = 1$ acceptance exceeding 63% across all benchmarks. Acceptance decreases at farther offsets because a draft can be committed only when all preceding drafts in the speculative prefix are also accepted.

Importantly, increasing the horizon from $d = 2$ to $d = 4$ largely preserves acceptance at the first two offsets while introducing additional farther-ahead drafts. As a result, the mean number of committed tokens per decoding step increases from 1.93–2.21 to 2.12–2.63 across the four benchmarks. Thus, although farther-ahead drafts are accepted less frequently, a longer horizon enables each decoding step to advance farther on average.

**Runtime breakdown.** We next examine how the increased number of committed tokens per decoding step translates into wall-clock acceleration. As shown in Table 5, prediction and verification dominate the runtime, accounting for 48.0% and 48.8% of the total decoding time, respectively, while context compression contributes only 0.4%. Thus, the compression procedure itself introduces negligible overhead. Each accepted foresight draft can be committed without an additional sequential autoregressive forward pass, while the first mismatch is replaced by the autoregressive token produced during verification. As a result, committing more drafts per step directly reduces the number of sequential forward passes required for generation.

## 7 Conclusion

We presented **MemoSight**, a lightweight framework for improving the fidelity and efficiency of CoT compression. By combining Adaptive Memory Allocation, Uniform Positional Layout, and Foresight Multi-Token Prediction, MemoSight achieves a better accuracy–memory trade-off than existing CoT-compression methods across multiple models and benchmarks. Its foresight tokens further enable lossless speculative decoding, improving generation efficiency without modifying the backbone architecture. Overall, MemoSight provides a simple and effective approach to more faithful and efficient long-CoT reasoning.

## AI USAGE DISCLOSURE

Generative AI tools were used to assist with language editing, drafting and refining portions of the manuscript, and identifying potentially relevant literature. All AI-assisted text and literature suggestions were reviewed and verified by the authors. Generative AI was not used to develop the proposed method, conduct experiments, generate experimental data, or formulate mathematical claims. The authors take full responsibility for the content of the paper.

## REFERENCES

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024a.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024b.

Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Xiaoqian Liu, Tong Xiao, and Jingbo Zhu. Efficient prompting methods for large language models: A survey. *arXiv preprint arXiv:2404.01077*, 2024.

Guoxuan Chen, Han Shi, Jiawei Li, Yihang Gao, Xiaozhe Ren, Yimeng Chen, Xin Jiang, Zhenguo Li, Weiyang Liu, and Chao Huang. SepLLM: Accelerate large language models by compressing one segment into one separator. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=MhVJCxYEEi.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

Peng Chunyi, Xu Zhipeng, Yan Yukun, Liu Zhenghao, Yu Shi, Mei Sen, Sun Yubo, Zhang Yongheng, Zhou Jie, Gu Yu, et al. Conceptformer: Learning adaptive latent concepts for query-document alignment in visual document retrieval. *arXiv preprint arXiv:2608.15698*, 2026.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Anastasios Gerontopoulos, Spyros Gidaris, and Nikos Komodakis. Multi-token prediction needs registers. *arXiv preprint arXiv:2505.10518*, 2025.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Song, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

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

Haoxiang Jia, Earl T Barr, and Sergey Mechtaev. Compressing code context for llm-based issue resolution. *arXiv preprint arXiv:2603.28119*, 2026.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Zhensheng Jin, Xinze Li, Yifan Ji, Chunyi Peng, Zhenghao Liu, Qi Shi, Yukun Yan, Shuo Wang, Furong Peng, and Ge Yu. Recut: Balancing reasoning length and accuracy in llms via stepwise trails and preference optimization. In *EMNLP (Findings)*, pp. 14269–14282, 2025.

Vasilis Kontonis, Yuchen Zeng, Shivam Garg, Lingjiao Chen, Hao Tang, Ziyan Wang, Ahmed Awadallah, Eric Horvitz, John Langford, and Dimitris Papailiopoulos. Memento: Teaching llms to manage their own context, 2026. URL https://arxiv.org/abs/2604.09852.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024b.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. Prompt compression for large language models: A survey, 2024c. URL https://arxiv.org/abs/2410.12388.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xcompressor: Generalized prompt compression for large language models, 2024d. URL https://arxiv.org/abs/2408.03094.

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

Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and Jingbo Zhu. Autoencoding-free context compression for llms via contextual semantic anchors. *arXiv preprint arXiv:2510.08907*, 2025.

Xin Liu, Runsong Zhao, Xinyu Liu, Junhao Ruan, Pengcheng Huang, Shichao Dong, Chunyang Xiao, Chenglong Wang, Changliang Li, Jingbo Zhu, and Tong Xiao. Hybridthinker: Efficient chain-of-thought reasoning via compressed memory and transient thought steps, 2026. URL https://arxiv.org/abs/2606.03768.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens, 2024. URL https://arxiv.org/abs/2304.08467.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*, 2024.

Jianhui Pang, Fanghua Ye, Derek Fai Wong, Xin He, Wanshun Chen, and Longyue Wang. Anchor-based large language models, 2024. URL https://arxiv.org/abs/2402.07616.

Chunyi Peng, Haidong Xin, Xuanshuo Sheng, Xin Dai, Zhenghao Liu, Shuo Wang, Yukun Yan, Zulong Chen, Yu Gu, and Ge Yu. Memory shot for long-term dialogue. *arXiv preprint arXiv:2606.28338*, 2026.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequencepre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2401–2410, 2020.

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

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Jiwei Tang, Jin Xu, Tingwei Lu, Zhicheng Zhang, Yiming Zhao, Lin Hai, and Hai-Tao Zheng. Perception compressor: A training-free prompt compression framework in long context scenarios, 2025. URL https://arxiv.org/abs/2409.19272.

Jiwei Tang, Shilei Liu, Zhicheng Zhang, Qingsong Lv, Runsong Zhao, Tingwei Lu, Langming Liu, Haibin Chen, Yujin Yuan, Hai-Tao Zheng, Wenbo Su, and Bo Zheng. Read as human: Compressing context via parallelizable close reading and skimming, 2026a. URL https://arxiv.org/abs/2602.01840.

Jiwei Tang, Shilei Liu, Zhicheng Zhang, Yujin Yuan, Libin Zheng, Wenbo Su, and Bo Zheng. Comi: Coarse-to-fine context compression via marginal information gain, 2026b. URL https://arxiv.org/abs/2602.01719.

Jiwei Tang, Zhicheng Zhang, Shunlong Wu, Jingheng Ye, Lichen Bai, Zitai Wang, Tingwei Lu, Lin Hai, Yiming Zhao, Hai-Tao Zheng, and Hong-Gee Kim. Gmsa: Enhancing context compression via group merging and layer semantic alignment, 2026c. URL https://arxiv.org/abs/2505.12215.

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

Jianjin Wang, Runsong Zhao, Xiaoqian Liu, Yuan Ge, Ziqiang Xu, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and Jingbo Zhu. Mtp-s2ut: Enhancing speech-to-speech translation quality with multi-token prediction. *arXiv preprint arXiv:2510.10003*, 2025.

Xiangfeng Wang, Zaiyi Chen, Zheyong Xie, Tong Xu, Yongyi He, and Enhong Chen. In-context former: Lightning-fast compressing context for large language model, 2024. URL https://arxiv.org/abs/2406.13618.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*, 2025.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, 2025.

Yuchen Yan, Yongliang Shen, Yang Liu, Jin Jiang, Mengdi Zhang, Jian Shao, and Yueting Zhuang. Inftythink: Breaking the length limits of long-context reasoning in large language models, 2026. URL https://arxiv.org/abs/2503.06692.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Chenxiao Yang, Nathan Srebro, David McAllester, and Zhiyuan Li. Pencil: Long thoughts with short memory. *arXiv preprint arXiv:2503.14337*, 2025b.

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

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

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *arXiv preprint arXiv:2512.21711*, 2025b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=RkRrPp7GKO.

Runsong Zhao, Xin Liu, Xinyu Liu, Pengcheng Huang, Chunyang Xiao, Tong Xiao, and Jingbo Zhu. Position ids matter: An enhanced position layout for efficient context compression in large language models. *arXiv preprint arXiv:2409.14364*, 2024.

Runsong Zhao, Shilei Liu, Jiwei Tang, Langming Liu, Haibin Chen, Weidong Zhang, Yujin Yuan, Tong Xiao, Jingbo Zhu, Wenbo Su, and Bo Zheng. Comet: Collaborative memory transformer for efficient long context modeling, 2026a. URL https://arxiv.org/abs/2602.01766.

Runsong Zhao, Shilei Liu, Jiwei Tang, Langming Liu, Haibin Chen, Weidong Zhang, Yujin Yuan, Tong Xiao, Jingbo Zhu, Wenbo Su, and Bo Zheng. Poc: Performance-oriented context compression for large language models via performance prediction, 2026b. URL https://arxiv.org/abs/2603.19733.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

Yiqing Zhou, Yu Lei, Shuzheng Si, Qingyan Sun, Wei Wang, Yifei Wu, Hao Wen, Gang Chen, Fanchao Qi, and Maosong Sun. From context to edus: Faithful and structured context compression via elementary discourse unit decomposition. *arXiv preprint arXiv:2512.14244*, 2025.

# A Inference Procedure

Algorithm 1 summarizes the iterative inference procedure of MemoSight. Each reasoning step is first generated with foresight-based speculative decoding and then compressed into memory tokens before generation continues.

**Algorithm 1** MemoSight Iterative Inference

**Require:** Prompt $P$, model $\mathcal{M}$, foresight horizon $d$, compression ratio $c$, step delimiter $\langle \text{e} \rangle$, and end-of-sequence token $\langle \text{eos} \rangle$

$S \leftarrow \text{Prefill}(\mathcal{M}, P)$
*finished* $\leftarrow$ **false**
**while not** *finished* **do**
  $R_{\text{current}} \leftarrow \emptyset$
  **// Phase 1: Foresight-based speculative decoding**
  **repeat**
    $A \leftarrow \text{SpeculativeStep}(\mathcal{M}, S, d)$
    $A \leftarrow \text{PrefixToBoundary}(A, \langle \text{e} \rangle, \langle \text{eos} \rangle)$
    $S \leftarrow S \oplus A$
    $R_{\text{current}} \leftarrow R_{\text{current}} \oplus A$
    **if** $\langle \text{eos} \rangle \in A$ **then**
      *finished* $\leftarrow$ **true**
    **end if**
  **until** $\langle \text{e} \rangle \in A$ **or** *finished*
  **// Phase 2: Step compression**
  **if not** *finished* **then**
    $l \leftarrow \lceil |R_{\text{current}}|/c \rceil$
    $(M, o) \leftarrow \text{CompressStep}(\mathcal{M}, S, R_{\text{current}}, l)$
    $S \leftarrow \text{ReplaceCache}(S, R_{\text{current}}, M, o)$
  **end if**
**end while**

In each SpeculativeStep, the standard next-token prediction produces the first token while $d$ foresight tokens generate farther-ahead drafts in parallel. The drafts are verified against standard autoregressive predictions, the longest matching prefix is accepted, and the verified autoregressive token replaces the first mismatched draft. If the accepted sequence crosses a step delimiter or the end-of-sequence token, only the prefix through the first such token is committed; any later tokens are discarded. This ensures that generation after a completed reasoning step resumes only after its KV states have been compressed. In CompressStep, AMA determines the number of memory tokens and UPL assigns their position IDs; the raw KV states of $R_{\text{current}}$ are then replaced by those of the resulting memory and output tokens.

# B Experimental Details

## B.1 Baseline Details

During evaluation, we employ greedy decoding with a maximum output length of 10,240 tokens for all models. We compare MemoSight against the following baselines:

- **Standard**: A baseline that applies few-shot Chain-of-Thought (CoT) prompting to the Qwen2.5-7B Hui et al. (2024), Qwen3-8B Yang et al. (2025a), and Llama-3.1-8B Dubey et al. (2024) models without additional training.
- **Distill-R1** Guo et al. (2025): A reasoning model distilled from DeepSeek-R1's response data, based on Qwen2.5-7B and Llama-3.1-8B backbones.
- **Vanilla**: A full-parameter fine-tuned model trained under the same setup as LightThinker and MemoSight, but without any compression or acceleration mechanisms. It serves as the uncompressed reference.

- **H2O** Zhang et al. (2023): A training-free KV cache eviction strategy that retains "Heavy Hitter" tokens and a local window. We apply H2O to the Vanilla model using a sliding window of 1024 and a heavy-hitter budget of 512 tokens.
- **SepLLM** Chen et al. (2025): A training-free framework that preserves KV caches for initial tokens, separators, and a local window. We configure SepLLM with an initial cache size of 384, a separator budget of 64, and a local window of 256, maintaining a total cache capacity of 1024.
- **LightThinker** Zhang et al. (2025a): A post-training method that compresses each reasoning step into a fixed number of memory tokens. Upon reaching a step boundary, the model generates memory tokens to summarize the context, after which the original reasoning tokens are evicted from the KV cache.

### B.2 Training Details

The Vanilla baseline, LightThinker, and MemoSight are trained on the Bespoke-Stratos-17k (BS17K) dataset for 5 epochs. Experiments are conducted on 8 H200 GPUs using DeepSpeed ZeRO-3 offload. We use a micro-batch size of 2 with 4 gradient accumulation steps, yielding a global batch size of 64. We employ a cosine learning rate schedule with a 0.05 warmup ratio. The peak learning rate is set to $1 \times 10^{-5}$ for Vanilla and $2 \times 10^{-5}$ for LightThinker and MemoSight.

For all methods, the constructed CoT-compression sequence, including reasoning, memory, and output tokens, is truncated to at most 4,096 tokens. For MemoSight, foresight tokens are inserted before the reasoning tokens only after this truncation, which can approximately double the physical training sequence length. The resulting sequences are then padded to the maximum length within each batch. For MemoSight, we set the compression ratio to $c = 5$ and sample the foresight offset $\delta \in 0, 1, 2$ uniformly for each training example. The NTP and MTP losses are weighted by $0.7$ and $0.3$, respectively (i.e., $\lambda = 0.7$).

### B.3 Ablation Details

We evaluate the contribution of Adaptive Memory Allocation (AMA), Uniform Positional Layout (UPL), and foresight-token-based Multi-Token Prediction (MTP) by removing each component individually while keeping the remaining training settings unchanged.

- **MemoSight w/o AMA:** We replace adaptive memory allocation with a fixed budget of 9 memory tokens for every reasoning step, following the LightThinker setting.
- **MemoSight w/o UPL:** Memory tokens use the conventional consecutive position assignment, $\rho(m_k) = \rho^i_{|R_i|} + k$, instead of being distributed within the positional span of the reasoning step. Foresight-token position IDs remain unchanged.
- **MemoSight w/o MTP:** We remove the foresight tokens and the MTP objective, while retaining AMA and UPL for memory compression.

All component ablations are conducted on Qwen2.5-7B using the same training configuration as the full MemoSight model.

### B.4 Comparison with Conventional MTP

We further compare our special-token-based MTP with conventional MTP architectures under the same CoT-compression setting. Following DeepSeek-V3 Liu et al. (2024), **MTP Last** appends two sequential Transformer prediction blocks to the final-layer hidden states to predict future tokens through the shared LM head. We also consider **MTP Middle**, which instead takes intermediate-layer representations as input to the MTP module Wang et al. (2025).

As shown in Table 6, both conventional variants introduce substantially more additional parameters but achieve lower reasoning accuracy than MemoSight. These results suggest that, in our supervised fine-tuning setting, the lightweight special-token formulation provides a more effective way to introduce longer-horizon supervision without adding auxiliary Transformer modules.

Table 6: Comparison with conventional MTP variants under the same CoT-compression setting. MemoSight uses far fewer additional parameters while achieving higher accuracy.

| Method | Added Param. | GSM8K | MMLU | GPQA | BBH | Avg. |
|---|---|---|---|---|---|---|
| MemoSight | **4K** | **89.84** | **63.49** | **40.91** | **73.13** | **66.84** |
| MTP Middle | 103M | 88.17 | 61.25 | 38.89 | 69.49 | 64.45 |
| MTP Last | 103M | 85.06 | 61.34 | 32.83 | 71.11 | 62.59 |

### B.5 Evaluation Details

We evaluate all models using deterministic greedy decoding with a repetition penalty of 1.1 and a maximum output length of 10,240 tokens. For Table 1, Vanilla, H2O, SepLLM, LightThinker, and MemoSight use the same system prompt (Figure 15) and task prompts (Figure 16). Distill-R1 uses the same task prompts without the additional system prompt, following its native prompting setup. The Standard baseline uses the few-shot system prompt in Figure 13 together with the corresponding task prompts in Figure 14. For MMLU Hendrycks et al. (2020) and GPQA Rein et al. (2024), the multiple-choice options are randomly shuffled to reduce positional bias.

All inference experiments are conducted on H200 GPUs in BF16 precision, with one input sequence processed per GPU. Generation throughput is reported in tokens per second and includes the full inference pipeline, including prompt prefilling, memory compression, and speculative verification. All methods are evaluated using the same implementation and inference backend.

## C More Analyses and Discussions

### C.1 Long-Context Efficiency Analysis

We further isolate the efficiency benefit of KV-cache compression by disabling speculative decoding and comparing MemoSight with the uncompressed Vanilla model under standard autoregressive generation. We partition the generated sequence into 56-token reasoning segments and apply MemoSight with a fixed compression ratio of $c = 7$. Figure 8 reports end-to-end inference time and peak retained KV-cache tokens as the generated context length increases.

MemoSight substantially reduces KV-cache growth, lowering the peak number of retained KV tokens by over 80% at long sequence lengths. Without speculative decoding, the runtime benefit comes solely from the reduced KV cache. In our setting, this advantage becomes noticeable beyond approximately 8K generated tokens, as the attention cost increasingly dominates decoding latency for longer contexts.

### C.2 Case Study

Figure 9 provides a qualitative example of how higher-fidelity compression can help preserve critical intermediate information during multi-step reasoning. In this nutritional calculation problem, LightThinker correctly computes the remaining calorie budget but omits the grams per serving. It subsequently divides the per-serving calories by the whole-bag weight ($250 \div 300$), leading to an incorrect energy density and final answer of 120 g. In contrast, MemoSight retains the critical intermediate quantity in its compressed representation, first computing the serving size ($300 \div 5 = 60$) and then applying the calorie constraint. This leads to the correct scaling step, $(200 \div 250) \times 60$, and the final answer of 48 g.

### C.3 Effect of NTP–MTP Loss Weighting

We study the effect of the loss weight $\lambda$ by comparing the default setting $\lambda = 0.7$, which assigns weights of $0.7/0.3$ to the NTP and MTP objectives, with equal weighting ($\lambda = 0.5$). As shown in Figure 10, $\lambda = 0.7$ achieves equal or higher average accuracy across all tested compression ratios $c$, with larger gains under stronger compression. Increasing the relative weight of MTP does not improve performance and leads to clear degradation at larger compression ratios, suggesting

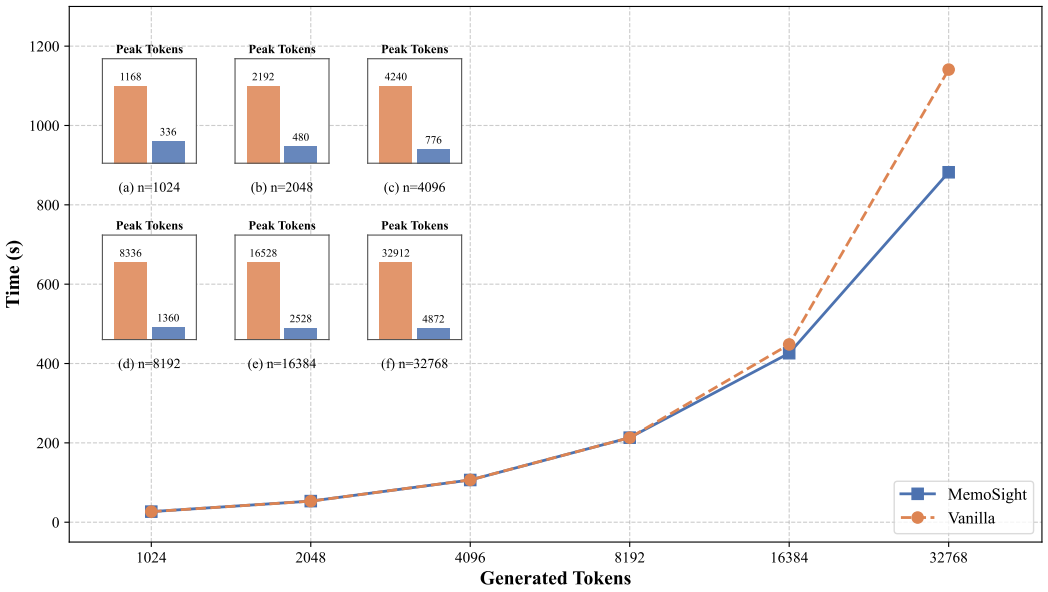



Figure 8: Long-context efficiency of KV-cache compression without speculative decoding. The main plot reports end-to-end inference time for MemoSight and Vanilla under autoregressive generation, while the inset plots compare peak retained KV-cache tokens at different sequence lengths.

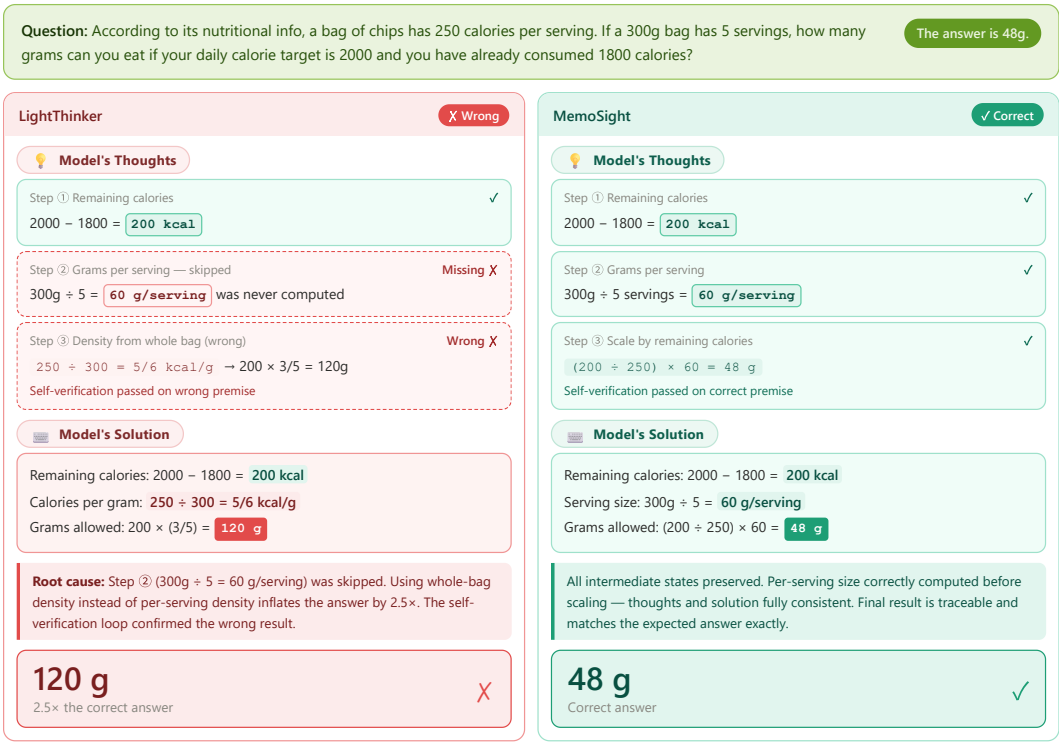



Figure 9: Case study comparing the reasoning trajectories of LightThinker and MemoSight. MemoSight preserves a critical intermediate quantity that is omitted by LightThinker, preventing the error from propagating to the final answer.

that a moderate MTP weight provides a better balance between standard next-token generation and longer-horizon supervision.

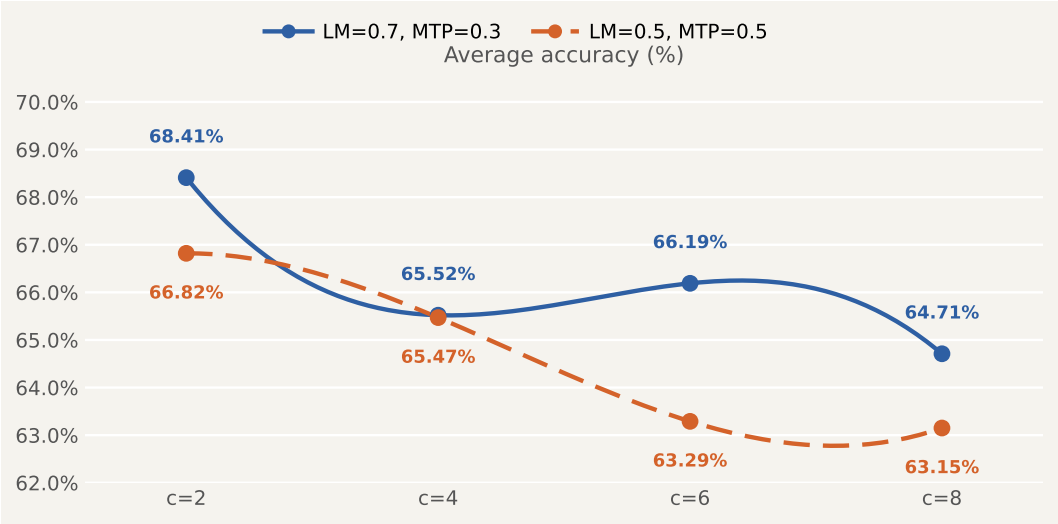



Figure 10: Sensitivity to the loss weight $\lambda$ across different compression ratios $c$. A larger weight on the NTP objective ($\lambda = 0.7$) yields equal or better accuracy than equal NTP–MTP weighting ($\lambda = 0.5$).

## D Additional Analysis of Memory–Content Locality

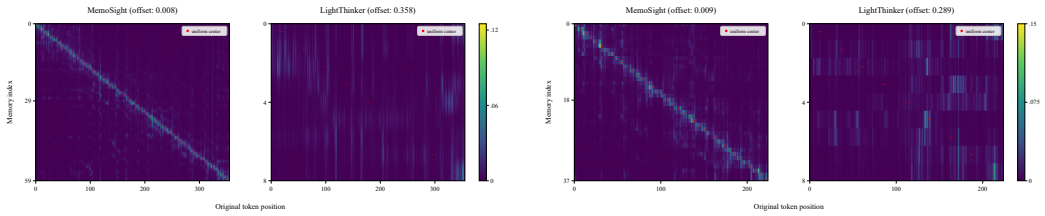
**Evaluation setup.** We provide additional visualizations of memory-to-reasoning attention using two independently trained checkpoints under their native positional layouts: MemoSight with UPL and LightThinker with conventional contiguous positioning. We follow the same attention aggregation procedure as in Figure 7. For memory token $j$, we identify its most-attended reasoning position $p_j$ and compare it with the center $c_j$ of the corresponding interval under an ideal uniform partition of a reasoning span of length $L$:

$$\text{Offset} = \frac{1}{M}\sum_{j=0}^{M-1}\frac{|p_j - c_j|}{L}, \qquad c_j = \left(j + \frac{1}{2}\right)\frac{L}{M}, \tag{6}$$

where $M$ is the number of memory tokens. A smaller value indicates a more orderly progression across the reasoning span.

**Additional examples.** Figure 11 presents two additional reasoning segments. For the 356-token segment, the normalized offset is $0.008$ with UPL, compared with $0.358$ under contiguous positioning. For the 225-token segment, the corresponding offsets are $0.009$ and $0.289$. Together with the example in Figure 7, UPL consistently maintains an offset below $0.01$ across all three segments, whereas contiguous positioning produces substantially larger offsets between $0.289$ and $0.358$. The visualizations likewise show a stable diagonal progression under UPL and diffuse, irregular attention under contiguous positioning.

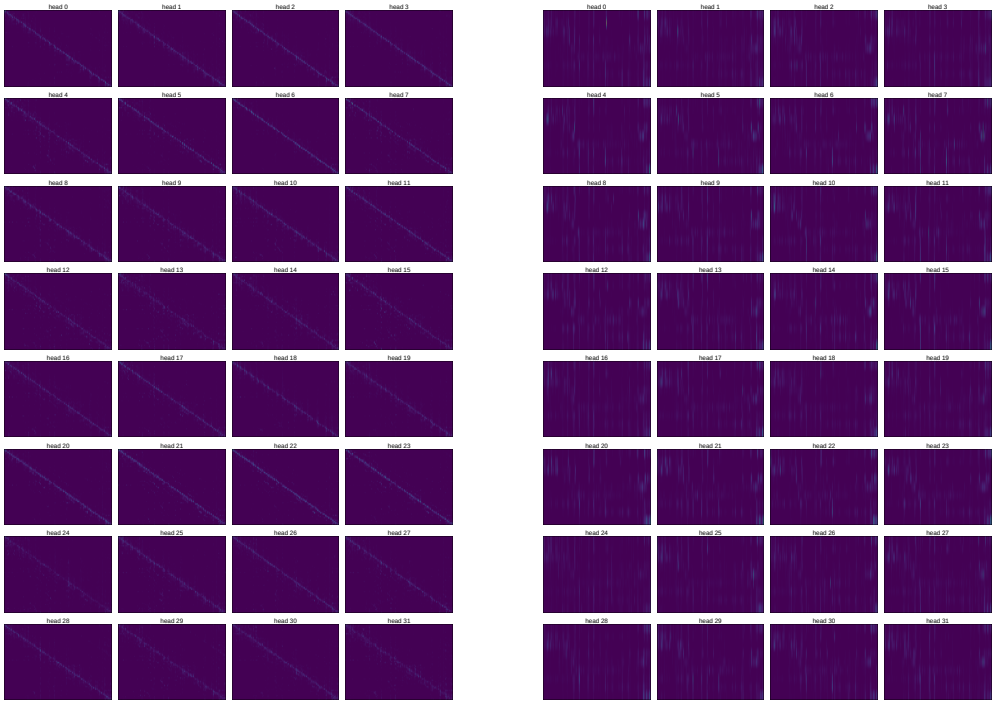
**Consistency across attention heads.** Figure 12 further decomposes the 356-token example by attention head. The localized diagonal pattern appears consistently across heads under UPL, while the contiguous baseline predominantly exhibits scattered vertical responses without a systematic progression. This confirms that the observed structure is not an artifact of averaging across attention heads.




(a) Memory-to-reasoning attention for a 356-token reasoning segment. UPL reduces the normalized offset from 0.358 to 0.008.

(b) Memory-to-reasoning attention for a 225-token reasoning segment. UPL reduces the normalized offset from 0.289 to 0.009.

Figure 11: Additional examples of memory-to-reasoning attention. Red markers indicate the ideal centers under uniform coverage. UPL produces a clear diagonal progression across reasoning positions, whereas contiguous positioning yields diffuse and irregular attention.



(a) MemoSight with UPL.

(b) LightThinker with contiguous positioning.

Figure 12: Per-head memory-to-reasoning attention maps for a representative compressed segment containing 356 reasoning tokens. UPL consistently produces a localized diagonal alignment pattern across attention heads, whereas contiguous positioning yields diffuse and irregular attention.

# E Additional Related Work

## E.1 Context Compression

Beyond the methods discussed in Section 2, a broad line of work studies compact representations for long contexts. For input-side compression, representative approaches include AutoCompressor Chevalier et al. (2023), ActivationBeacon Zhang et al. (2024), PyramidKV Cai et al. (2024b), SnapKV Li et al. (2024a), and SAC Liu et al. (2025). These methods primarily reduce the cost of processing or retaining the input context, whereas MemoSight focuses on reasoning states generated incrementally during decoding.

Related approaches also reduce reasoning-time context through latent reasoning, token selection, or latent condensation. **Latent reasoning** replaces explicit CoT tokens with continuous representations, as in Coconut Hao et al. (2025) and SoftCoT Xu et al. (2025). CODI Shen et al. (2025) aligns latent states with natural-language CoT trajectories, while SIM-CoT Wei et al. (2025) introduces auxiliary step-level supervision; recent work further examines whether current latent reasoning models consistently exhibit genuine latent reasoning behavior Zhang et al. (2025b).

**Token-selection** methods retain informative subsets of the context or reasoning trace. Examples include LLMLingua-2 Pan et al. (2024), Context Compressor Zhou et al. (2025), SWEzze Jia et al. (2026), and TokenSkip Xia et al. (2025).

Beyond direct context compression, ReCUT (Jin et al., 2025) reduces inference cost by shortening explicit reasoning trajectories, while ConceptFormer and Memory Shot (Chunyi et al., 2026; Peng et al., 2026) explore compact representations for retrieval and dialogue memory. These directions are complementary to MemoSight's focus on online compression of generated reasoning states.

### E.2 Multi-Token Prediction

Early work on multi-token prediction includes ProphetNet Qi et al. (2020), which predicts multiple future tokens through multi-stream attention for sequence-to-sequence learning. More recent MTP approaches and their relation to MemoSight are discussed in Section 2. Our additional comparison with conventional Transformer-based MTP variants is provided in Appendix B.4.

**System Prompt:**
Below is a question. Please think through it step by step, and then provide the final answer. If options are provided, please select the correct one.

## Output format:
Use "<THOUGHT>...</THOUGHT>" to outline your reasoning process, and enclose the final answer in '\boxed{}'.

## Example 1:
Question:
What is 2 + 3?
Output:
<THOUGHT>First, I recognize that this is a simple addition problem. Adding 2 and 3 together gives 5.</THOUGHT>
Therefore, the final answer is \boxed{5}.

## Example 2:
Question:
What is 2 + 3?
A. 4
B. 5
C. 10

Output:
<THOUGHT>First, I recognize that this is a simple addition problem. Adding 2 and 3 together gives 5.</THOUGHT>
Therefore, the final answer is \boxed{B}.

Figure 13: System prompt for Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct.

**GSM8K:**
Return your final response within \boxed{}. {Question}

**MMLU:**
Please select the option that best answers the question. Return your final response within \boxed{}. Here are the Question:
{Question}

**GPQA:**
Given a question, please select the option that best answers it. Return your final response within \boxed{}. {Question}

**BBH:**
Return your final response within \boxed{}. If options are provided, please select the correct one.
{Question}

Figure 14: Task prompt for Qwen2.5-7B-Instruct and Llama3.1-8B-Instruct.

**System Prompt:**
Your role as an assistant involves thoroughly exploring questions through a systematic long thinking process before providing the final precise and accurate solutions. This requires engaging in a comprehensive cycle of analysis, summarizing, exploration, reassessment, reflection, backtracking, and iteration to develop well-considered thinking process. Please structure your response into two main sections: Thought and Solution. In the Thought section, detail your reasoning process using the specified format: <—begin_of_thought—> {thought with steps separated with '\n\n'} <—end_of_thought—> Each step should include detailed considerations such as analyzing questions, summarizing relevant findings, brainstorming new ideas, verifying the accuracy of the current steps, refining any errors, and revisiting previous steps. In the Solution section, based on various attempts, explorations, and reflections from the Thought section, systematically present the final solution that you deem correct. The solution should remain a logical, accurate, concise expression style and detail necessary steps needed to reach the conclusion, formatted as follows: <—begin_of_solution—> {final formatted, precise, and clear solution} <—end_of_solution—> Now, try to solve the following question through the above guidelines:

Figure 15: The shared system prompt applied to Vanilla, H2O, SepLLM, LightThinker, and MemoSight across the Qwen and Llama series.

**GSM8K/MMLU/GPQA/BBH:**
Return your final response within \boxed{}. {Question}

Figure 16: The shared task prompt applied to Vanilla, H2O, SepLLM, LightThinker, and MemoSight across the Qwen and Llama series.