

Bridging the Memorization-Utilization Gap: Near-Lossless Context Compression via Reinforcement Learning

Yujan Ting, Xu Tang, Terrence Chen, Weijing Huang*

United Imaging Intelligence, Boston, MA, USA

{yujan.ting, xu.tang, terrence.chen, weijing.huang}@uii-ai.com

Abstract

Despite recent progress in context compression, we identify a fundamental memorization-utilization gap where models can compress context with near-perfect fidelity yet fail to effectively utilize these compressed representations for downstream tasks. We address this with a holistic training paradigm spanning pre-training, instruction tuning, and reinforcement learning, built upon an average pooling compression. Our key innovation uses outcome-based RL to enable implicit expansion: the model learns to adaptively unfold task-relevant details during generation, interleaving reconstruction with reasoning. We achieve near-lossless $16\times$ context compression ($\approx 5.3\times$ decoder sequence-length reduction in our current implementation) across 7B and 32B models, recovering over 98% of full-context QA performance and outperforming prior methods by 11 points. Our 32B model demonstrates strong out-of-distribution and length generalization, robustly scaling to 120k-token contexts despite training on no more than 4k tokens, matching full-context performance on NIAH, LongBench v2, and multi-hop reasoning. We verify the implicit expansion behavior in experiments.

1 Introduction

Consider an AI assistant processing hundreds of documents. Compressing input tokens at this scale directly translates to substantial savings in cost and latency. In fact, early work on soft prompt compression, which encodes text into continuous embeddings rather than discrete tokens, has shown that LLMs can achieve near-perfect memorization by reconstructing almost all information when sufficiently pretrained. ICAE (Ge et al., 2024) was among the first to demonstrate this capability and drew an analogy to how humans enhance memory encoding through training. Recent methods like GMSA (Tang et al., 2026), DeepSeek-OCR

(Wei et al., 2025), and its text-based counterpart C3 (Liu and Qiu, 2025) further improve reconstruction from compressed representations. Despite these advances, a persistent performance gap remains between compression approaches and full-context LLMs on downstream tasks. As noted by Lee et al. (2026), high-fidelity reconstruction does not necessarily translate into downstream utility, a gap evidenced in actual task performance (Liu et al., 2026). This fundamental gap between memorization and utilization raises a critical question: *If LLMs can already preserve information near losslessly, what prevents them from effectively leveraging that information for downstream tasks?*

Existing soft prompt compression methods (Ge et al., 2024; Louis et al., 2026) primarily rely on autoencoding pretraining and finetuning to align compression with downstream utilization, yet fail to fully bridge this gap. We push further by applying reinforcement learning (RL) to soft prompt compression, directly optimizing decoder generation for downstream performance. Surprisingly, we find that RL training incentivizes an implicit expansion behavior during decoding. As Figure 1 shows, the RL-trained model learns to selectively reconstruct short relevant spans when reasoning unfolds. By leveraging the model’s near-perfect reconstruction ability to local spans, this adaptively unpacks compressed information and enables more faithful utilization than supervised finetuned models, which are prone to hallucinations as they conflate entangled details from memory embeddings. Because reconstruction occurs on demand during generation, it adds minimal output overhead while achieving substantial input savings.

This RL-elicited decoding behavior parallels how humans recall memories. Cognitive research has established that humans recall through reconstruction rather than replay, with information actively rebuilt from distributed fragments instead of retrieved as exact copies (Bartlett, 1932; Schacter

*Corresponding author.

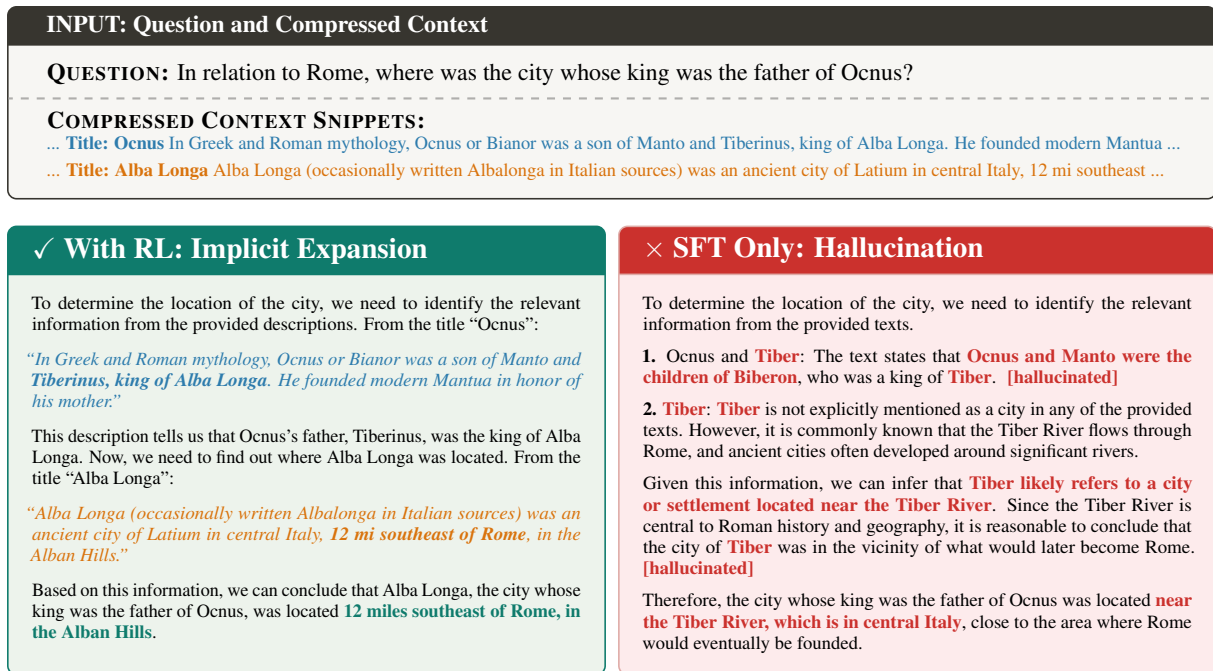


Figure 1: **Implicit expansion via RL bridges memorization and utilization.** Given compressed context (16 \times), our 32B RL-trained model (left) reconstructs essential details during reasoning: first expanding the passage linking Ocnus to Tiberinus, then Alba Longa’s location relative to Rome. The SFT model (right), despite access to the same compressed memory, hallucinates facts (“Biberon,” “Tiber”) and fails to retrieve stored information, illustrating the utilization gap that RL training addresses.

and Addis, 2007; Nader et al., 2000). This analogy offers a cognitive lens for understanding how LLMs utilize compressed memories, and may help explain why direct utilization via finetuning, which accesses compressed information without reconstruction, is less effective. Just as human memory benefits from reconstructive processes, LLM utilization of compressed information may require dynamic reconstruction rather than static access.

In this work, we address the memorization-utilization gap through a unified framework. Our main contributions are as follows:

- We propose a holistic training paradigm spanning pretraining, finetuning, and RL, built on a simple compression mechanism using average pooling and linear projection.
- We are the first to apply outcome-based RL to soft prompt compression. This elicits LLM’s implicit expansion behavior during reasoning, enabling more faithful memory utilization beyond what finetuning alone can achieve. We further empirically verify this behavior.
- We demonstrate strong length generalization. Despite training only on 4K-token sequences,

our model scales effectively to original contexts of 120K tokens, dramatically reducing training costs for long-context applications.

Our approach achieves near-lossless 16 \times context compression at both 7B and 32B scales. Specifically, 16 input tokens are encoded into one memory embedding, which corresponds to 5.3 \times reduction in decoder sequence length after accounting for the boundary tokens used for decoding. Both models recover 98% of full-context QA performance and outperform prior methods such as PISCO (Louis et al., 2025) and OSCAR (Louis et al., 2026) by over 11 points. The 32B model further retains 98.6% accuracy across five out-of-distribution benchmarks. On long-context tasks, it remains competitive with the full-context baseline on Needle In A Haystack (NIAH) (Kamradt, 2023), LongBench v2 (Bai et al., 2025), and multi-hop reasoning, while achieving up to 11.4 \times TTFT speedup at 120K tokens. We present the main results in Section 5, with latency analysis in Appendix J. We further show in Section 6 that implicit expansion is an empirically verified behavior, characterized by highly selective reconstruction and cross-passage chaining for multi-hop reasoning, without inflating output length.

2 Related Work

2.1 Soft Prompt Compression

Learned Memory Token Approaches A common approach in soft prompt compression is to append learnable memory tokens to the input and extract their hidden states as compressed representations. AutoCompressors (Chevalier et al., 2023) pioneered this with recursive compression using the same model as both encoder and decoder, while ICAE (Ge et al., 2024) employs a LoRA-adapted encoder with a frozen decoder. COCOM (Rau et al., 2025) demonstrated the importance of joint encoder-decoder training. PISCO (Louis et al., 2025) simplifies training via sequence-level distillation without pretraining, and OSCAR (Louis et al., 2026) introduces query-dependent compression. However, these methods suffer from semantic dilution: information is distributed unevenly across memory tokens (Tang et al., 2026).

Average Pooling Approaches To address semantic dilution, recent work directly pools encoder hidden states. GMSA (Tang et al., 2026) employs group average pooling with layer alignment, and Feldman and Artzi (2025) shows simple mean pooling is surprisingly effective. Our approach shares architectural similarities with these pooling methods but differs in training. Whereas prior work mainly employs autoencoding objectives or finetuning, we introduce a holistic paradigm that extends from pretraining through outcome-based RL. We are the first to apply RL to soft prompt compression, directly optimizing decoder performance on downstream tasks.

2.2 Reinforcement Learning for Compression

Prior RL work typically focuses on hard compression or input-side expansion decisions. CORE (Cui et al., 2025) applies end-to-end RL to optimize a compressor that generates textual summaries for RAG, but operates in a discrete space and thus risks information loss. More relevant to our work, REFRAG (Lin et al., 2025) trains an RL policy to selectively expand compressed soft chunks back to full tokens using perplexity reward. DAST (Chen et al., 2025) similarly allocates more soft tokens to dense regions based on perplexity, though without RL. However, both methods rely on heuristic signals and make expansion decisions prior to generation. In contrast, we optimize decoder utilization directly via outcome-based RL, enabling

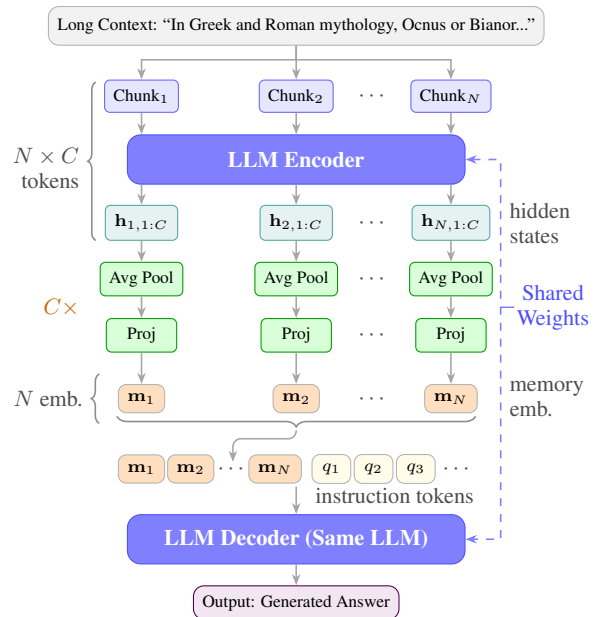


Figure 2: **Context compression architecture.** Input is chunked into C -token segments and encoded by the LLM. Hidden states are average-pooled and projected into memory embeddings, achieving $C \times$ compression. The same LLM decodes outputs from memory embeddings prepended to instructions.

on-demand reconstruction as generation unfolds.

3 Methodology

3.1 Model Architecture

Figure 2 illustrates our compression model architecture. We employ the same LLM as both encoder and decoder with an average pooling module. For encoding, we divide the input context into fixed-size chunks of C tokens, where C controls the compression rate (e.g., $C = 16$ yields a $16 \times$ compression). Each chunk is forwarded through the LLM, and the resulting final-layer hidden states are average-pooled across tokens and projected to produce a single memory embedding:

$$\mathbf{m}_i = \text{Proj} \left(\frac{1}{C} \sum_{j=1}^C \mathbf{h}_{i,j} \right) \quad (1)$$

where d is the model’s hidden dimension, $\mathbf{h}_{i,j} \in \mathbb{R}^d$ denotes the final-layer hidden state for the j -th token in chunk i , and $\text{Proj} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a learnable linear transformation. The process compresses N chunks into N memory embeddings $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_N]$.

This design is motivated by two considerations. First, LLM hidden states inherently capture contextual information through attention, so we

can directly pool them rather than learning compressed embeddings from scratch. Second, encoding chunks independently enables efficient batched processing. In line with these considerations, we adopt average pooling as our primary pooling strategy. While we also explored attention-based pooling, we found that simple averaging converges faster and requires less memory (Appendix B). This aligns with concurrent work (Tang et al., 2026; Feldman and Artzi, 2025), which shows that average pooling mitigates semantic dilution and yields more uniform information retention.

For decoding, the memory embeddings are prepended to the instruction tokens and fed to the same LLM for autoregressive generation. In our current implementation, each memory embedding is wrapped by boundary tokens (<MEM_START>, <MEM_END>). Since the encoder and decoder share the same LLM, the compressed representations naturally lie close to the decoder’s latent space, making them easier to interpret without complex alignment.

Compression-Rate Definitions We report two distinct quantities throughout the paper. (i) **Context compression rate (16×)**: the ratio of input context tokens to memory embeddings; with $C = 16$, every 16 input tokens are encoded into one memory embedding. This is a representation-level quantity and follows the convention of prior soft-prompt compression work. (ii) **Decoder sequence-length reduction ($\approx 5.3\times$)**: the ratio of input context tokens to decoder positions occupied by the compressed context, accounting for the two boundary tokens that wrap each memory embedding in our current implementation. These boundary tokens do not carry contextual information and serve only as decoding delimiters.

3.2 Training Objectives

We formulate context compression as a unified language modeling objective where the model learns to generate target sequences conditioned on memory embeddings. For continual pretraining and instruction tuning, all compression tasks share a common loss:

$$\mathcal{L} = - \sum_{t=1}^{|y|} \log P_{\theta}(y_t \mid \mathbf{M}, \mathbf{x}_{\text{inst}}, y_{<t}) \quad (2)$$

where \mathbf{M} denotes the memory embeddings, \mathbf{x}_{inst} the task instructions (Figure 11), and y the target sequence. We define four tasks (Table 1) that form

a curriculum of increasing abstraction: Reconstruction trains the model to retain complete information. Continuation generates text from a compressed prefix. Summarization produces coarse-grained reconstructions. QA requires reasoning over compressed memory to answer questions.

Task	Memory	Instruction	Target
Reconstruction	Full doc.	Prompt	Original
Continuation	First half	Prompt + bridges	Second half
Summarization	Full doc.	Prompt	Summary
QA	Context	Question	Answer

Table 1: **Compression training task formulation.** Bridge refers to uncompressed tokens that stabilize the transition from compressed memory to generation.

3.3 Training Recipe

We adopt a three-stage curriculum that progressively builds compression capability and effective downstream utilization.

3.3.1 Continual Pretraining

Design Principles This stage builds the foundations for model compression while preserving its base language modeling ability. We adopt two design choices. First, following multimodal training paradigms, we treat compressed memory embeddings as a distinct modality and mix compression tasks with standard full-context language modeling within each training batch to prevent catastrophic forgetting. Second, we introduce bridge chunks: small, uncompressed spans (typically 16 tokens) retained in the input that bridge compressed embeddings and the generation target. We find that continuation loss is inherently difficult to optimize and decreases only marginally. Bridge chunks serve as scaffolding that eases this transition and helps reduce continuation loss (Appendix C).

Phase 1 Pretraining proceeds in two phases. The first phase establishes foundations by training on a balanced mixture of reconstruction and continuation tasks, following prior work that found this split effective (Rau et al., 2025). We interleave these with standard pretraining at approximately 70:30, and use a single bridge chunk for all continuation tasks to stabilize the transition.

Phase 2 Since reconstruction converges quickly to a loss below 0.1, we reallocate capacity to tasks that emphasize language modeling and utilization. The task mixture shifts to continuation (40%), QA

(27.5%), summarization (17.5%), and reconstruction (15%). Continuation receives the highest weight as it converges most slowly. We adjust the compression-to-standard ratio to 60:40, and vary the number of bridge chunks from 0 to 4 to improve robustness. Throughout both phases, the encoder, projection layer, and decoder are trained jointly.

3.3.2 Instruction Tuning

We then adapt the model to downstream QA tasks with the ChatML format (Appendix L). We freeze the projection layer and train only the encoder and decoder. Because the projection layer has already learned a robust encoder-decoder alignment through diverse compression tasks during pretraining, freezing it here preserves this mapping and prevents overfitting to narrow task distributions.

3.3.3 Reinforcement Learning

Motivation Supervised finetuning alone does not bridge the gap between memorizing compressed inputs and using them effectively. We attribute this gap to differences in computational demands. Reconstruction unfolds token by token, affording sequential computation that progressively disentangles information. Supervised finetuning instead learns a direct mapping from compressed representations to task outputs, forcing retrieval and reasoning to occur simultaneously without iterative refinement. Prior work shows that additional sequential computation increases transformer expressivity (London and Kanade, 2025) and improves reasoning on challenging representations (Kim et al., 2025). Compressed embeddings, which condense long contexts into a few dense vectors, are a particularly challenging case. Yet standard downstream supervision, typically direct answers or responses distilled from full-context models, provides no signal for how compressed information should be extracted or refined. As a result, reconstruction ability remains latent during generation. This suggests that effective utilization may require more flexible decoding strategies that afford sufficient computation access.

RL Formulation To this end, we adopt outcome-based reinforcement learning, which optimizes decoding behavior through task-level feedback rather than predefined utilization patterns. This allows the model to explore a broader space of decoding trajectories and discover strategies for utilizing compressed representations. As in finetuning, we use only downstream QA data. We apply DAPO

(Yu et al., 2025), which extends GRPO (Shao et al., 2024) with dynamic sampling to filter out prompt groups with zero reward variance, ensuring effective gradients for learning nuanced decoding strategies. We set a maximum response length of 600 tokens, well beyond typical QA outputs, giving the model sufficient room to refine information from compressed embeddings. To prevent degenerate exploitation of this headroom, we apply DAPO’s soft overlong punishment to penalize excessive length. Together, these choices give the model flexibility to allocate computation adaptively while avoiding verbosity. We use a binary reward based on correctness:

$$R(y, a^*) = \mathbb{I}[\text{Judge}(y, a^*) = \text{correct}] \quad (3)$$

where a generative judge evaluates whether the model’s response y is correct given ground truth a^* . We generate compressed embeddings offline from the SFT checkpoint, tuning only the decoder during RL. The decoding behaviors that emerge from this training are analyzed in Sections 5 and 6.

4 Experiments

4.1 Implementation Details

We use Qwen2.5-7B (base) and Qwen2.5-32B (base) (Yang et al., 2025b) as backbones, applying $16\times$ compression as our primary setting, with additional experiments at $64\times$. To balance training capacity with efficiency, we propose FQLoRA (Appendix O.1), a hybrid strategy that fully finetunes $\sim 36\%$ of transformer layers while applying QLoRA (Dettrmers et al., 2023) to the rest.

For pretraining and instruction tuning, we adopt DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) for distributed training. For reinforcement learning, we use VERL (Sheng et al., 2025), with Qwen3-32B (Yang et al., 2025a) and Qwen2.5-72B-Instruct serving as reward models for the 7B and 32B settings, respectively. Training details and compute costs are provided in Appendices L.1, O, and K.

4.2 Training Data

Our training data spans multiple domains: general pretraining data from DCLM (Li et al., 2024) and CCI2 (Beijing Academy of Artificial Intelligence, 2024); mathematical and reasoning content from NaturalReasoning (Yuan et al., 2025) and Proof-Pile-2 (Azerbaiyev et al., 2024); code from StarCoder (Li et al., 2023); general instructions from OpenHermes-2.5 (Teknum, 2023); and

Stage	Task	Samples	Data Sources
CPT-1	Compression (recon/cont) Standard CPT	220K 100K	DCLM, CCI2, NaturalReasoning, Proof-Pile-2, StarCoder Same + OpenHermes-2.5
CPT-2	Compression (+QA/summ) Standard CPT	400K 300K	CPT-1 sources + HotpotQA, TriviaQA, RAGBench Same as CPT-1
SFT	QA w/ compression	144K	HotpotQA, TriviaQA, Natural Questions, RAGBench
RL	QA w/ compression	60K	HotpotQA, TriviaQA, Natural Questions

Table 2: Training data composition for our three-stage recipe: continual pretraining (CPT-1, CPT-2), SFT, and RL.

QA data from HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), and RAGBench (Friel et al., 2025). Table 2 summarizes the task and data composition across all training stages.

4.3 Evaluation

We evaluate on both in-distribution and out-of-distribution QA benchmarks, and also employ long-context stress tests to assess retrieval and reasoning performance under compression. All evaluations use Qwen2.5-72B-Instruct as a judge, except for Long-Context Multi-Hop QA, which uses exact match following RULER (Hsieh et al., 2024). We also validate judge robustness using GPT-4o (Hurst et al., 2024) as an alternative judge on the main QA benchmarks (Appendix N). Details of the evaluation setup are provided in Appendix L.2 and M.

QA Benchmarks We sample 1,000 instances each from three in-distribution datasets (HotpotQA, TriviaQA, and Natural Questions) and five out-of-distribution datasets, including MuSiQue (Trivedi et al., 2022), 2WikiMultihopQA (Ho et al., 2020), BioASQ (Krithara et al., 2023), SQuAD (Rajpurkar et al., 2016), and MS MARCO (Nguyen et al., 2016). We use benchmarks with answer-containing contexts to avoid retrieval sensitivity and establish accurate upper bounds for compression.

Long-Context Benchmarks We evaluate on three benchmarks with contexts up to 120K tokens. NIAH (Kamradt, 2023) tests the retrieval of a target fact placed at various depths within long contexts (1K–120K tokens). For **Long-Context Multi-Hop QA**, we sample 100 questions each from HotpotQA, 2WikiMultihopQA, and MuSiQue, evaluating each across context lengths from 4K to 120K tokens by inserting distractor paragraphs following RULER (Hsieh et al., 2024). **LongBench v2** (Bai et al., 2025) is a challenging multiple-choice benchmark requiring reasoning over real-world long doc-

uments; we select 254 samples with contexts under 100K tokens.

4.4 Baselines

We compare against state-of-the-art $16\times$ compression methods: PISCO (Louis et al., 2025) and OSCAR (Louis et al., 2026). For fair comparison, we evaluate each compression method’s backbone model as the full-context upper bound: Llama-3.1-8B-Instruct (Grattafiori et al., 2024) for PISCO, Mistral-7B-Instruct-v0.2 and Mistral-Small-24B-Instruct-2501 (Jiang et al., 2023) for OSCAR, and Qwen2.5-7B/32B-Instruct (Yang et al., 2025b) for our models.

5 Results

Our experiments address two questions. First, can soft prompt compression achieve near-lossless downstream performance, and how does RL improve upon supervised finetuning? Second, to what extent can models trained on short contexts generalize to substantially longer ones?

We first verify that our pretrained models achieve near-perfect reconstruction, with ROUGE-1 F1 reaching 0.98 and ROUGE-L above 0.96 for contexts up to 2K tokens (Appendix A). This confirms that compressed representations faithfully preserve source information. The challenge, therefore, lies in utilizing that information for downstream tasks.

5.1 Main Results

Table 3 presents QA performance across model scales and compression ratios.

Near-lossless Compression At $16\times$ compression, our approach recovers nearly all baseline performance at both scales. The 7B RL model attains 82.10% in-distribution accuracy, recovering 98.8% of the full-context Qwen2.5-7B-Instruct baseline (83.07%). The 32B RL model reaches 86.20%, recovering 98.1% of its baseline (87.90%) while retaining 98.6% on out-of-distribution tasks.

Model	Comp.	In-Distribution				Out-of-Distribution					
		HotpotQA	NQ	TriviaQA	Avg.	2Wiki	BioASQ	SQuAD*	MuSiQue	MARCO*	Avg.
<i>7B Models — Full Context</i>											
Mistral-7B-Instruct-v0.2	—	74.00	79.90	91.80	81.90	55.30	95.40	89.40	38.70	65.50	68.86
Qwen2.5-7B-Instruct	—	76.70	82.30	90.20	83.07	65.60	93.90	92.30	43.40	64.50	71.94
Llama-3.1-8B-Instruct	—	82.40	83.30	91.60	85.77	69.00	94.10	90.40	47.90	64.80	73.24
<i>7B Models — 16× Compression</i>											
PISCO ^a	16×	53.50	68.10	86.30	69.30	33.40	86.60	79.90*	22.10	59.10*	56.22
OSCAR ^b	16×	58.50	66.20	87.60	70.77	38.10	82.60	84.70*	25.40	62.40*	58.64
Ours (SFT)^c	16×	71.00	70.50	85.20	75.57	65.10	71.30	66.60	35.40	53.90	58.46
Ours (RL)^c	16×	80.50	75.60	90.20	82.10	66.80	86.70	79.80	43.00	58.70	67.00
<i>7B Models — 64× Compression</i>											
Ours (SFT)^c	64×	51.20	55.80	81.20	62.73	41.90	56.90	40.40	17.50	45.50	40.44
Ours (RL)^c	64×	62.00	63.00	87.00	70.67	40.40	81.80	57.00	24.00	52.50	51.14
<i>32B Models — Full Context</i>											
Qwen2.5-32B-Instruct	—	84.70	85.50	93.50	87.90	75.90	95.30	93.30	52.30	67.60	76.88
Mistral-Small-24B-Instruct	—	89.20	81.40	94.50	88.37	78.50	96.40	92.10	67.60	58.60	78.64
<i>32B Models — 16× Compression</i>											
OSCAR-24B ^d	16×	63.10	72.30	88.90	74.77	45.20	88.60	85.50*	27.50	63.10*	61.98
Ours (SFT)^e	16×	77.90	76.50	90.10	81.50	71.40	80.70	76.40	45.40	54.30	65.64
Ours (RL)^e	16×	86.10	80.60	91.90	86.20	77.60	92.60	87.80	61.70	59.30	75.80

Table 3: **QA evaluation on in-distribution and out-of-distribution benchmarks.** At 16× compression, our 7B and 32B RL models retain 98.8% and 98.1% of full-context performance on in-distribution tasks, with 93.1% and 98.6% on out-of-distribution tasks, respectively. Backbones: ^aLlama-3.1-8B-Instruct, ^bMistral-7B-Instruct-v0.2, ^cQwen2.5-7B, ^dMistral-Small-24B-Instruct, ^eQwen2.5-32B. *These datasets were included in OSCAR/PISCO training, so results may not reflect fair comparison.

These results substantially exceed prior compression methods. Our 7B model outperforms PISCO (69.30%) and OSCAR (70.77%) by over 11 points in-distribution, while our 32B model surpasses OSCAR-24B by a similar margin despite comparable backbone capacity. Even before applying RL, our SFT models exceed these baselines (75.57% at 7B, 81.50% at 32B), indicating that our pretraining establishes stronger compression foundations. To further isolate the contribution of our method from backbone choice, we apply our full approach to Llama-3.1-8B, the same backbone used by PISCO, and observe 98.6% retention with a 15-point gain over PISCO (Appendix D). At 64× compression, our 7B RL model still achieves 70.67%, comparable to OSCAR at 16× and demonstrating graceful degradation at extreme ratios.

RL Enables Implicit Expansion RL training consistently improves over SFT at both model scales. At 7B, we observe gains of 6.5 points in-distribution and 8.5 points out-of-distribution. At 32B, gains reach 4.7 and 10.2 points, respectively. Analysis of model outputs reveals the mechanism behind these gains. RL-trained models learn to adaptively expand relevant details as reasoning unfolds (Figure 1 and Figures 17–20). These local reconstructions allocate more computation where

needed for more faithful utilization of compressed representations. This confirms our hypothesis that effective utilization requires sufficient computational depth during decoding. Without this behavior, SFT models tend to conflate entangled details and hallucinate more frequently. Appendix P illustrates both behaviors. We further show that supervised fine-tuning on distilled chain-of-thought traces (a quote-then-reason pattern) yields only partial gains, indicating that this behavior is not easily reproduced by imitation alone (Appendix E).

Compression Aids Multi-Hop Reasoning An interesting pattern emerges on multi-hop tasks, where our models consistently surpass their full-context counterparts. At 7B scale, the RL model outperforms the baseline on HotpotQA (+3.8) and 2WikiMultihopQA (+1.2). This pattern amplifies at 32B scale across all multi-hop benchmarks, most notably on MuSiQue (+9.4). We hypothesize that learned compression acts as beneficial information distillation, filtering noise and surfacing task-relevant content. Combined with implicit expansion acquired through RL, this allows the model to dynamically retrieve pertinent facts at each reasoning step rather than attending over lengthy contexts where relevant evidence is dispersed. Appendix P.2 provides example responses.

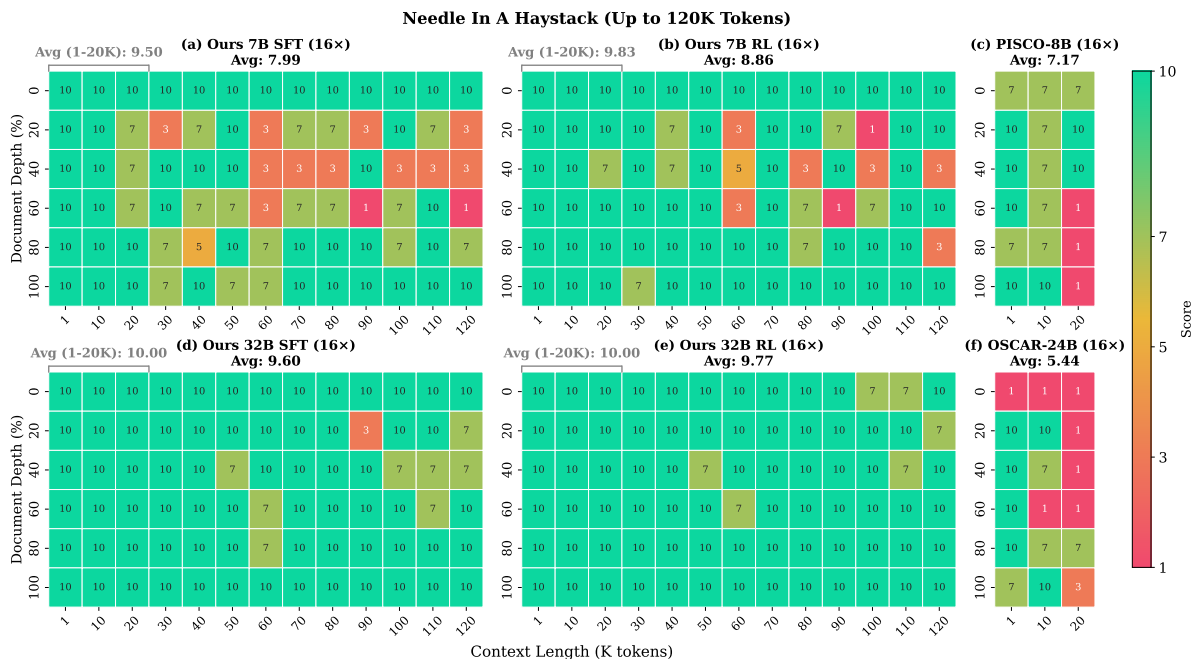


Figure 3: NIAH performance across context lengths up to 120K tokens. Our 32B RL models (16 \times) achieve near-perfect retrieval (9.77/10), while prior methods (PISCO, OSCAR) degrade after 10K and fail beyond 20K tokens due to memory or architectural constraints. Scores range from 1 (failure) to 10 (exact retrieval).

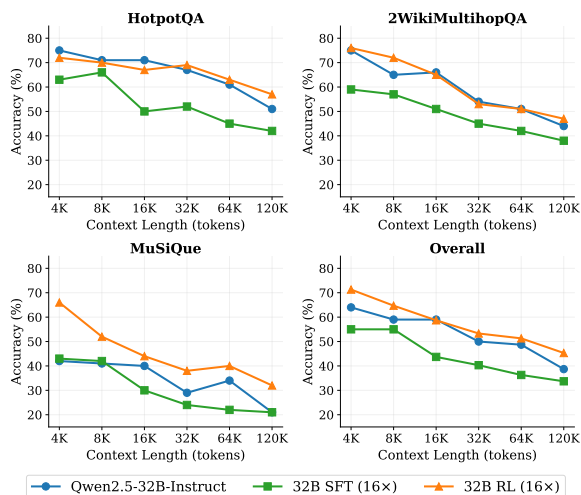


Figure 4: Multi-hop QA accuracy across 4K–120K contexts. The 32B RL model matches or outperforms the full-context baseline, particularly at longer contexts.

5.2 Long-Context Generalization

Beyond standard QA tasks, we test generalization far exceeding the 4K-token training length (1.3K memory embeddings plus boundary tokens). Our models scale robustly to 120K original tokens (7.5K memory embeddings) and generalize well beyond prior compression methods.

NIAH Figure 3 shows retrieval performance across contexts up to 120K tokens. This task is

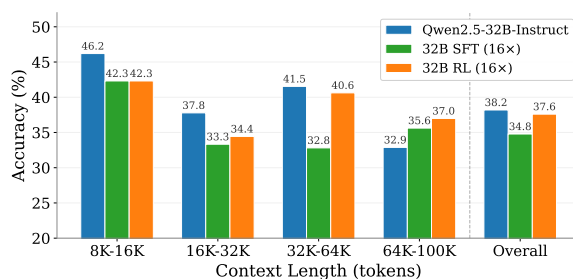


Figure 5: LongBench v2 accuracy across 8K–100K contexts. Our 32B RL model shows minimal degradation versus full-context on challenging reasoning tasks.

zero-shot as our post-training uses only QA data. Our 32B RL model achieves 9.77/10, matching the full-context baseline (9.96/10), while the 7B model reaches 8.86/10 with degradation primarily at mid-document positions, consistent with the “lost in the middle” phenomenon (Liu et al., 2024). In contrast, prior methods such as PISCO and OSCAR degrade after 10K tokens and fail beyond 20K due to memory or architectural constraints. Full results appear in Appendix F.

Long-Context Reasoning We evaluate whether the multi-hop advantage extends to longer contexts. On Long-Context Multi-Hop QA (Figure 4), our 32B RL model achieves 57.4% overall accuracy versus 53.2% for the full-context baseline (+4.2 points), while the SFT model reaches only 44.0%,

underscoring the importance of RL. On LongBench v2 (Figure 5), our model achieves comparable overall accuracy (37.6% vs. 38.2%) while outperforming the baseline by 4.1 points at 64K to 100K tokens. Across both benchmarks, our RL model consistently outperforms at longer contexts, aligning with Berton et al. (2025) who show that compression mitigates attention dilution by concentrating computation on fewer relevant tokens. Our method extends this benefit to more aggressive $16\times$ compression, where RL-elicited implicit expansion further enhances utilization.

6 Quantifying Implicit Expansion

To quantify the implicit expansion behavior elicited by RL, we measure **expansion rate**: the fraction of responses containing at least one exact contiguous n -word span from the context. We vary n from 8 to 20. Shorter thresholds are susceptible to coincidental phrase overlap, whereas longer matches provide stronger evidence that the model reconstructs specific details rather than sharing topical vocabulary. Table 4 shows expansion rates over all 8,000 QA samples used in Section 5.1, with a breakdown for the multi-hop QA subset. RL produces substantially more verbatim reconstruction than both SFT and the full-context baseline at every threshold, and the gap widens with n : at 32B, the RL/SFT ratio grows from $2.4\times$ at 8-gram to $11.3\times$ at 20-gram, with similar trends at 7B. The widening gap at stricter thresholds suggests targeted reconstruction rather than incidental lexical overlap.

Task-Adaptive Multi-Source Chaining Expansion is most pronounced on multi-hop tasks, where the model chains evidence across passages. At 32B, RL reaches 90.7% expansion rate at 8-gram on multi-hop, with 80.3% of responses drawing spans from two or more distinct context regions (vs. 12.6% for SFT; Table 11 in Appendix H). On single-hop tasks, the multi-region rate drops to 21.6%, showing that the model adapts its strategy to the task: it chains multiple passages when necessary, but focuses on a single passage when that is sufficient. Across all tasks, only 3.2% of available context words are reconstructed, suggesting that expansion remains highly selective.

Expansion Does Not Inflate Output Length Despite higher reconstruction volume, implicit expansion does not increase output length. On multi-hop tasks at 32B, the model reconstructs an average of

<i>n</i> -gram	32B			7B		
	Full	SFT	RL	Full	SFT	RL
<i>Overall</i>						
$n = 8$	33.1	30.6	73.5	30.7	27.7	50.1
$n = 15$	10.0	6.4	45.4	8.0	5.1	24.6
$n = 20$	5.4	2.8	31.5	4.3	2.0	12.8
<i>Multi-hop</i>						
$n = 8$	40.6	46.9	90.7	35.7	41.1	86.5
$n = 15$	14.6	10.7	76.8	9.5	8.4	55.2
$n = 20$	8.1	4.8	61.8	5.6	3.3	30.8

Table 4: Expansion rate at different n -gram thresholds under $16\times$ compression. RL elicits substantially more reconstruction than both SFT and the full-context baseline, with larger gaps at stricter thresholds, especially on multi-hop tasks.

64.2 verbatim words per response while generating only 190 tokens, comparable to the full-context baseline of 169 tokens (+12%). At 7B, RL reconstructs 38.2 verbatim words per multi-hop response while generating 151 tokens, slightly fewer than the full-context baseline of 155 tokens (−3%). Table 12 in Appendix I reports full results for reconstruction volume and response length. It suggests that expansion is integrated into the reasoning process rather than appended to the output. The output overhead is negligible relative to the input savings.

7 Conclusion

While soft prompt compression can preserve information with near-perfect fidelity, effective downstream utilization remains a challenge. We address this through outcome-based RL, which elicits an implicit expansion behavior where the model dynamically reconstructs relevant spans as reasoning unfolds. Our approach achieves near-lossless $16\times$ compression with over 98% performance retention on QA tasks, both in-distribution and out-of-distribution. Despite training on only 4K-token sequences, our 32B model scales to 120K-token contexts with capabilities comparable to full-context counterparts.

We link compression utilization to the model’s latent dynamic reconstruction during inference. Future work will extend this framework to function calling and agentic tasks, where compressed context could reduce inference costs while maintaining tool-use capabilities. A second direction is to refine the decoder format. Replacing per-embedding boundary tokens with a single pair around the full memory sequence would remove structural overhead. We leave this design to future work.

Limitations

While our framework achieves strong performance at $16\times$ compression, several limitations warrant discussion.

Decoding and Encoding Costs We report $16\times$ as the context compression rate (Section 3.1), consistent with prior work. Under our current per-embedding wrapping, the decoder sequence-length reduction is approximately $5.3\times$, due to the two boundary tokens introduced for each memory embedding. This structural overhead should be considered when estimating real-world inference savings. At the same time, while compression reduces decoder tokens, it also requires encoder computation. Our approach is therefore most beneficial when compressed representations are reused across multiple queries, amortizing the encoding cost.

Compression Rate vs. Performance Trade-off

At $64\times$ compression, our 7B model shows degradation (70.67% vs. 82.10% at $16\times$), suggesting limits to extreme compression for smaller models. At this rate, each memory embedding must encode 64 tokens of information, potentially exceeding the representational capacity of a single embedding to preserve fine-grained details. Exploring $64\times$ compression at 32B scale may reveal whether increased capacity can sustain performance at higher compression rates.

Answer Ambiguity in QA Datasets We observe performance gaps on NQ, SQuAD, and MS MARCO. These datasets exhibit inherent answer ambiguity, where multiple valid answer spans or free-form phrasings exist beyond the provided references. Since it is infeasible to enumerate all acceptable answers, LLM-as-a-judge evaluation may incorrectly reject valid responses. Binary rewards amplify this by providing no signal for such false negatives. Future work could explore soft reward signals to address this limitation.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. [GQA: Training generalized multi-query transformer models from multi-head checkpoints](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. [Llemma: An open language model for mathematics](#). In *The Twelfth International Conference on Learning Representations*.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. [LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.

Frederic C. Bartlett. 1932. [Remembering: A Study in Experimental and Social Psychology](#). Cambridge University Press.

Beijing Academy of Artificial Intelligence. 2024. [CCI2-Data](#). Dataset.

Gabriele Berton, Jayakrishnan Unnikrishnan, Son Tran, and Mubarak Shah. 2025. [CompLLM: Compression for long context q&a](#). *Preprint*, arXiv:2509.19228.

Shaoshen Chen, Yangning Li, Zishan Xu, Yongqin Zeng, Shunlong Wu, Xinshuo Hu, Zifei Shan, Xin Su, Jiwei Tang, Yinghui Li, and Hai-Tao Zheng. 2025. [DAST: Context-aware compression in LLMs via dynamic allocation of soft tokens](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20544–20552, Vienna, Austria. Association for Computational Linguistics.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. [Adapting language models to compress contexts](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

Ziqiang Cui, Yunpeng Weng, Xing Tang, Peiyang Liu, Shiwei Li, Bowei He, Jiamin Chen, Yansen Zhang, Xiuqiang He, and Chen Ma. 2025. [CORE-RAG: Lossless compression for retrieval-augmented llms via reinforcement learning](#). *Preprint*, arXiv:2508.19282.

Tri Dao. 2024. [Flashattention-2: Faster attention with better parallelism and work partitioning](#). In *The Twelfth International Conference on Learning Representations*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. [QLoRA: Efficient fine-tuning of quantized LLMs](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Yair Feldman and Yoav Artzi. 2025. [Simple context compression: Mean-pooling and multi-ratio training](#). *Preprint*, arXiv:2510.20797.

- Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2025. [RAGBench: Explainable benchmark for retrieval-augmented generation systems](#). *Preprint*, arXiv:2407.11005.
- Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. [In-context autoencoder for context compression in a large language model](#). In *The Twelfth International Conference on Learning Representations*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. [Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. [RULER: What’s the real context size of your long-context language models?](#) In *First Conference on Language Modeling*.
- Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 399 others. 2024. [GPT-4o System Card](#). *Preprint*, arXiv:2410.21276.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7B](#). *Preprint*, arXiv:2310.06825.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Greg Kamradt. 2023. [Needle in a haystack - pressure testing LLMs](#).
- Eunki Kim, Sangryul Kim, and James Thorne. 2025. [Learning to insert \[PAUSE\] tokens for better reasoning](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23760–23777, Vienna, Austria. Association for Computational Linguistics.
- Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. [BioASQ-QA: A manually curated corpus for biomedical question answering](#). *Scientific Data*, 10:170.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- 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, SOSP ’23*, pages 611–626, New York, NY, USA. Association for Computing Machinery.
- Ivan Yee Lee, Cheng Yang, and Taylor Berg-Kirkpatrick. 2026. [Optical context compression is just \(bad\) autoencoding](#). *Preprint*, arXiv:2512.03643.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, and 40 others. 2024. [DataComp-LM: in search of the next generation of training sets for language models](#). In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS ’24*, Red Hook, NY, USA. Curran Associates Inc.
- Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LL, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, and 39 others. 2023. [StarCoder: may the source be with you!](#) *Transactions on Machine Learning Research. Reproducibility Certification*.
- Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. 2025. [REFRAG: Rethinking rag based decoding](#). *Preprint*, arXiv:2509.01092.
- Fanfan Liu and Haibo Qiu. 2025. [Context cascade compression: Exploring the upper limits of text compression](#). *Preprint*, arXiv:2511.15244.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.

- Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and JingBo Zhu. 2026. [Autoencoding-free context compression for LLMs via contextual semantic anchors](#). In *The Fourteenth International Conference on Learning Representations*.
- Charles London and Varun Kanade. 2025. [Pause tokens strictly increase the expressivity of constant-depth transformers](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Maxime Louis, Hervé Déjean, and Stéphane Clinchant. 2025. [PISCO: Pretty simple compression for retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15506–15521, Vienna, Austria. Association for Computational Linguistics.
- Maxime Louis, Thibault Formal, Hervé Déjean, and Stéphane Clinchant. 2026. [OSCAR: Online soft compression for RAG](#). In *The Fourteenth International Conference on Learning Representations*.
- Karim Nader, Glenn E. Schafer, and Joseph E. Le Doux. 2000. [Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval](#). *Nature*, 406:722–726.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. [MS MARCO: A human generated machine reading comprehension dataset](#). In *CoCo@NIPS*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Samy Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. [ZeRO: Memory optimizations toward training trillion parameter models](#). In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- David Rau, Shuai Wang, Hervé Déjean, Stéphane Clinchant, and Jaap Kamps. 2025. [Context embeddings for efficient answer generation in retrieval-augmented generation](#). In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 493–502, New York, NY, USA. Association for Computing Machinery.
- Daniel L. Schacter and Donna Rose Addis. 2007. [The cognitive neuroscience of constructive memory: remembering the past and imagining the future](#). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):773–786.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. [Hybridflow: A flexible and efficient rlhf framework](#). In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pages 1279–1297, New York, NY, USA. Association for Computing Machinery.
- Jiwei Tang, Zhicheng Zhang, Shunlong Wu, Jingheng Ye, Lichen Bai, Zitai Wang, Tingwei Lu, Lin Hai, Yiming Zhao, Hai-Tao Zheng, and Hong-Ge Kim. 2026. [GMSA: Enhancing context compression via group merging and layer semantic alignment](#). *Preprint*, arXiv:2505.12215.
- Teknum. 2023. [OpenHermes 2.5: An open dataset of synthetic data for generalist llm assistants](#). Dataset.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. [MuSiQue: Multi-hop questions via single-hop question composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554.
- vLLM Project. 2025. [Prompt embedding inputs](#). Online documentation.
- Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. [DeepSeek-OCR: Contexts optical compression](#). *Preprint*, arXiv:2510.18234.
- 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, and 41 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025b. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin,

Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. [DAPO: an open-source LLM reinforcement learning system at scale](#). *Preprint*, arXiv:2503.14476.

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, Ilya Kulikov, Kyunghyun Cho, Yuandong Tian, Jason E Weston, and Xian Li. 2025. [NaturalReasoning: Reasoning in the wild with 2.8m challenging questions](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Reconstruction Performance

We evaluate reconstruction quality by measuring how well models can regenerate original text from compressed memory embeddings. We test our continual pretrained models on 100 samples from the DCLM (Li et al., 2024) holdout set across varying context lengths (256 to 4096 tokens). See Appendix M for evaluation prompts and generation parameters.

Figure 6 and Table 5 show ROUGE-1 and ROUGE-L F1 scores for both 7B and 32B models with 16x compression. Both pretrained models achieve near-perfect reconstruction ($F1 > 0.97$) for contexts up to 2048 tokens. Performance degrades at 4096 tokens, with F1 scores dropping to approximately 0.84 for ROUGE-1 and 0.75 to 0.82 for ROUGE-L. The 7B model performs comparably to the 32B model, demonstrating that smaller models can effectively learn the reconstruction.

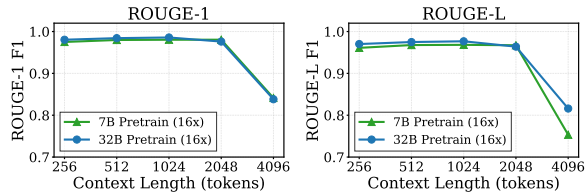


Figure 6: Reconstruction performance (ROUGE F1) across context lengths for continual pretrained 7B and 32B models with 16x compression. Both models achieve high fidelity reconstruction up to 2048 tokens.

Metric	Model	256	512	1024	2048	4096
ROUGE-1	7B (16x)	0.98	0.98	0.98	0.98	0.84
	32B (16x)	0.98	0.98	0.99	0.98	0.84
ROUGE-L	7B (16x)	0.96	0.97	0.97	0.97	0.75
	32B (16x)	0.97	0.97	0.98	0.96	0.82

Table 5: Reconstruction scores across context lengths for continual pretrained models.

Our downstream evaluations (Section 5) show that models generalize to 120K tokens with perfor-

mance comparable to full-context baselines, even as global reconstruction (evaluated here) degrades substantially beyond 4K tokens. While both global and local reconstruction likely degrade at longer contexts, the strong downstream results suggest that local reconstruction of task-relevant spans is more robust than exhaustive global reconstruction. This is what RL training encourages: rather than reconstructing the entire context, the model learns to selectively expand and retrieve relevant information at each reasoning step, making it more resilient to compression at long contexts.

B Pooling Strategy Comparison

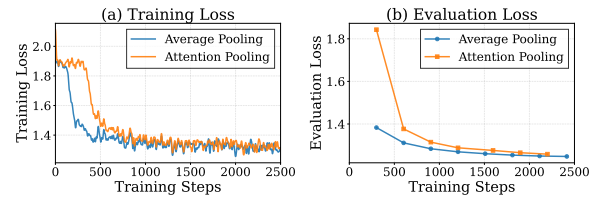


Figure 7: Training and evaluation loss comparison between average pooling and attention-based pooling (Grouped-Query Attention) during continual pretraining. Average pooling converges faster and achieves slightly lower evaluation loss.

Figure 7 compares the training and evaluation loss curves for average pooling and an attention-based pooling variant implemented with Grouped-Query Attention (GQA) (Ainslie et al., 2023) during continual pretraining. Average pooling demonstrates faster convergence, achieving lower training loss earlier in training. While attention-based pooling eventually reaches comparable training loss, average pooling maintains a slight advantage in evaluation loss, suggesting better generalization. Combined with its lower memory footprint and computational cost, we adopt average pooling as our default strategy.

C Bridge Chunk Ablation

To validate the effectiveness of bridge chunks, we conducted preliminary experiments during continual pretraining. We trained on 200K samples from DCLM (Li et al., 2024) with equal proportions of reconstruction and continuation tasks, comparing configurations with no bridge chunks versus a single bridge chunk.

Figure 8 shows that including a single bridge chunk consistently improves both reconstruction and continuation losses. While reconstruction

loss decreases substantially regardless of bridge chunk configuration, continuation loss decreases only marginally. Including a single bridge chunk enables further improvement on the continuation task.

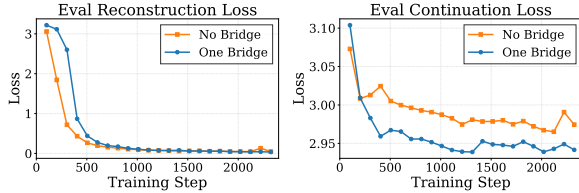


Figure 8: Evaluation loss comparison between configurations with no bridge chunks and one bridge chunk during continual pretraining.

Config	Train Loss		Eval Loss	
	Recon.	Cont.	Recon.	Cont.
No Bridge	0.067	2.991	0.047	2.974
One Bridge	0.037	2.911	0.036	2.942

Table 6: Final losses at training step 2300 for bridge chunk ablation.

D Cross-Backbone Comparison

To isolate the contribution of our method from backbone differences, we replicate our full training pipeline on Llama-3.1-8B (Base) (Grattafiori et al., 2024) and compare directly against PISCO (Louis et al., 2025), which uses the same backbone. Table 7 shows these results alongside our Qwen2.5-7B experiments.

On the Llama-3.1-8B backbone with $16\times$ compression, our RL model improves over PISCO by 15.3 points in-distribution (84.60% vs. 69.30%) and 14.0 points out-of-distribution (70.22% vs. 56.22%). It retains 98.6% of full-context Llama-3.1-8B-Instruct performance in-distribution and 95.9% out-of-distribution. These retention rates closely match those observed with our Qwen2.5-7B backbone (98.8% ID, 93.1% OOD). Consistent with our Qwen experiments, the SFT model alone surpasses PISCO on the same backbone (+10.4 ID, +3.8 OOD), despite training from the base checkpoint while PISCO fine-tunes from Llama-3.1-8B-Instruct. Since PISCO performs finetuning without continual pretraining, this gap highlights the importance of our pretraining stage in establishing stronger compression foundations. The subsequent RL stage provides further gains (+4.9 ID, +10.2 OOD over SFT), showing that each training stage

contributes meaningfully. These trends remain consistent across both backbones, even though we use only 8 rollouts for Llama RL training compared with 16 for Qwen, further supporting the robustness of our approach.

E Comparison with Chain-of-Thought Fine-tuning

We investigate whether the implicit expansion behavior induced by RL can be learned through supervised fine-tuning on chain-of-thought (CoT) traces. Using the same 60K QA training set as in our RL stage (Table 2), we distill CoT traces from Qwen2.5-72B-Instruct by prompting it to quote relevant context before answering. After filtering for correctness using the same judge criteria (Appendix M.3), we obtain 50.3K traces. Starting from our Qwen2.5-7B SFT checkpoint with $16\times$ compression, we compare our proposed CPT \rightarrow SFT \rightarrow RL pipeline with an alternative CPT \rightarrow SFT \rightarrow CoT-SFT variant, where the final RL stage is replaced by supervised fine-tuning on the 50.3K distilled traces using the same training configuration as the SFT stage (Appendix O). Figure 9 shows the distillation prompt and an example trace.

Prompt for CoT Trace Distillation

Context: {context}
 Question: {question}
 Think step by step to answer the question. As you reason, quote the relevant sentences from the context to support your reasoning. After your reasoning, provide the final answer in `\boxed{}`.

Example Distilled CoT Trace

Question: Crush 'n' Gusher is a water coaster to open at which theme park located in Lake Buena Vista in Orlando, Florida?
Response: To determine where Crush 'n' Gusher is located, we need to identify the theme park mentioned in the context. Let's break it down step by step:
 1. Identify the location of Crush 'n' Gusher:
 The context states: "Crush 'N' Gusher is a water coaster in Disney's Typhoon Lagoon on the Walt Disney World Resort property."
 2. Determine the location of Disney's Typhoon Lagoon:
 The context provides: "Disney's Typhoon Lagoon is a water theme park located at the Walt Disney World Resort in Lake Buena Vista in Orlando, Florida."
 By combining these two pieces of information, we can conclude that Crush 'n' Gusher is located in Disney's Typhoon Lagoon, which is in Lake Buena Vista, Orlando, Florida.
 Final answer: `\boxed{Disney's Typhoon Lagoon}`

Figure 9: CoT distillation setup. **Top:** Prompt used to collect quote-then-reason traces from Qwen2.5-72B-Instruct. **Bottom:** Example of a distilled trace. CoT-SFT then trains on these traces.

Table 8 shows that CoT-SFT provides modest gains over SFT (+1.85 overall), while remaining

Model	Comp.	In-Distribution				Out-of-Distribution					
		HotpotQA	NQ	TriviaQA	Avg.	2Wiki	BioASQ	SQuAD*	MuSiQue	MARCO*	Avg.
<i>Qwen2.5-7B Backbone</i>											
Qwen2.5-7B-Instruct [†]	—	76.70	82.30	90.20	83.07	65.60	93.90	92.30	43.40	64.50	71.94
Ours (SFT)^{a†}	16×	71.00	70.50	85.20	75.57	65.10	71.30	66.60	35.40	53.90	58.46
Ours (RL)^{a†}	16×	80.50	75.60	90.20	82.10	66.80	86.70	79.80	43.00	58.70	67.00
<i>Llama-3.1-8B Backbone</i>											
Llama-3.1-8B-Instruct [†]	—	82.40	83.30	91.60	85.77	69.00	94.10	90.40	47.90	64.80	73.24
PISCO ^{b†}	16×	53.50	68.10	86.30	69.30	33.40	86.60	79.90*	22.10	59.10*	56.22
Ours (SFT)^c	16×	74.80	74.30	90.10	79.73	68.50	73.70	67.10	37.80	53.00	60.02
Ours (RL)^c	16×	83.30	80.90	89.60	84.60	70.00	91.50	79.50	51.80	58.30	70.22

Table 7: Cross-backbone comparison at 16× compression on Qwen2.5-7B and Llama-3.1-8B. On the same Llama backbone used by PISCO, our method shows consistent improvements. Backbones: ^aQwen2.5-7B (base), ^bLlama-3.1-8B-Instruct, ^cLlama-3.1-8B (base). [†]Results from Table 3. *These datasets were included in PISCO training, so results may not reflect a fair comparison.

Model	Overall	In-Distribution				Out-of-Distribution					
		HotpotQA	NQ	TriviaQA	Avg.	2Wiki	BioASQ	SQuAD	MuSiQue	MARCO	Avg.
Qwen2.5-7B SFT (16×) [†]	64.88	71.00	70.50	85.20	75.57	65.10	71.30	66.60	35.40	53.90	58.46
Qwen2.5-7B CoT-SFT (16×)	66.73	74.70	68.80	82.60	75.37	67.90	80.40	77.50	35.60	46.30	61.54
Qwen2.5-7B RL (16×)[†]	72.66	80.50	75.60	90.20	82.10	66.80	86.70	79.80	43.00	58.70	67.00

Table 8: Comparison of RL and chain-of-thought fine-tuning at 16× compression. CoT-SFT trains on 50.3K reasoning traces distilled from Qwen2.5-72B-Instruct with explicit quote-then-reason prompting. RL outperforms CoT-SFT by 5.93 points overall. [†]Results from Table 3.

well below RL (66.73 vs. 72.66). CoT-SFT degrades on several benchmarks compared to SFT (e.g., NQ, TriviaQA, MS MARCO), suggesting that distilled reasoning traces alone can lead to overfitting to surface patterns.

We hypothesize that this gap arises from two factors. First, CoT-SFT learns to imitate a fixed quote-then-reason pattern rather than optimize answer correctness directly. As a result, it can produce expansions that appear plausible structurally but still contain subtle factual errors, such as incorrect names or numbers. RL, in contrast, optimizes directly for outcome correctness, allowing the model to adaptively determine what to reconstruct and how to reason over it. Second, the distilled traces are generated by a model with access to uncompressed context, where the information access patterns differ from those in memory embeddings. RL avoids this distribution mismatch by optimizing directly on the model’s own compressed representations.

F NIAH Results

Table 9 and Figure 3 report NIAH retrieval performance. Our 32B models achieve strong results, with both SFT (9.60) and RL (9.77) approaching the full-context baseline (9.96). For 7B, RL improves over SFT (8.86 vs. 7.99). Prior methods

(OSCAR, PISCO) do not generalize well, with OSCAR-Mistral-7B scoring only 1.50.

Model	Comp.	Max Ctx	Score
<i>7B Models</i>			
Qwen2.5-7B-Instruct	—	120K	10.00
OSCAR-Mistral-7B	16×	20K	1.50
PISCO-Llama-8B	16×	20K	7.17
Ours (SFT)	16×	120K	7.99
Ours (RL)	16×	120K	8.86
<i>32B Models</i>			
Qwen2.5-32B-Instruct	—	120K	9.96
OSCAR-Mistral-Small-24B	16×	20K	5.44
Ours (SFT)	16×	120K	9.60
Ours (RL)	16×	120K	9.77

Table 9: NIAH performance (score out of 10). Our models handle 120K contexts while prior compression methods fail beyond 20K due to embedding dimension mismatch (OSCAR) or OOM (PISCO, tested on 4×H100 80GB).

G Response Length Analysis

We analyze response length to verify that our implicit expansion strategy incurs only modest overhead. Table 10 shows average response tokens across all QA benchmarks. Our RL models produce responses of comparable length to their full-context baselines. At 7B scale, our model generates 151 tokens versus 155 for the baseline on average. At 32B scale, the increase is modest at 12% (190

Backbone	Model	Avg. Tokens
Qwen2.5-7B	Full Context	154.83
	Ours SFT (16×)	132.10
	Ours RL (16×)	151.16
Llama-3.1-8B	Full Context	80.72
	PISCO (16×)	51.60
Mistral-7B	Full Context	88.38
	OSCAR (16×)	52.53
Qwen2.5-32B	Full Context	168.54
	Ours SFT (16×)	134.98
	Ours RL (16×)	189.75
Mistral-Small-24B	Full Context	78.94
	OSCAR (16×)	52.63

Table 10: Average response length (in tokens) across all QA benchmarks (8,000 samples per model).

<i>n</i> -gram	32B			7B		
	Full	SFT	RL	Full	SFT	RL
<i>Multi-hop</i>						
<i>n</i> = 8	8.1	12.6	80.3	5.8	9.5	64.4
<i>n</i> = 15	1.7	1.2	49.4	1.3	0.6	20.6
<i>n</i> = 20	1.1	0.3	29.5	0.6	0.3	6.3
<i>Single-hop</i>						
<i>n</i> = 8	6.1	3.8	21.6	6.4	2.7	3.6
<i>n</i> = 15	1.2	0.3	4.8	1.4	0.3	0.3
<i>n</i> = 20	0.4	0.1	1.7	0.6	0.1	0.1

Table 11: Multi-region rate (%) at key *n*-gram thresholds, broken down by task type. RL’s multi-region rate is dramatically higher on multi-hop tasks than on single-hop tasks, showing the model chains multiple passages when the task demands it.

vs 169 tokens). PISCO and OSCAR generate more concise responses (~52 tokens). We also note that their Llama and Mistral backbones inherently produce shorter outputs than Qwen (80–88 vs 155–169 tokens).

H Multi-Region Rate

We define the **multi-region rate** as the fraction of responses whose matched verbatim spans originate from two or more distinct context regions (separated by >200 characters). Table 11 reports this rate broken down by task type, revealing that the model adapts its expansion strategy to task structure.

I Reconstruction Volume vs. Response Length

Table 12 reports mean reconstructed words per response on multi-hop tasks alongside mean total response length (tokens) and the **reconstruction**

ratio: the fraction of all available context words that appear verbatim in model outputs, aggregated across responses (total reconstructed words ÷ total context words). Despite reconstructing tens of verbatim words per response, RL response lengths remain comparable to full-context baselines, confirming that expansion is integrative rather than additive.

J Latency Analysis

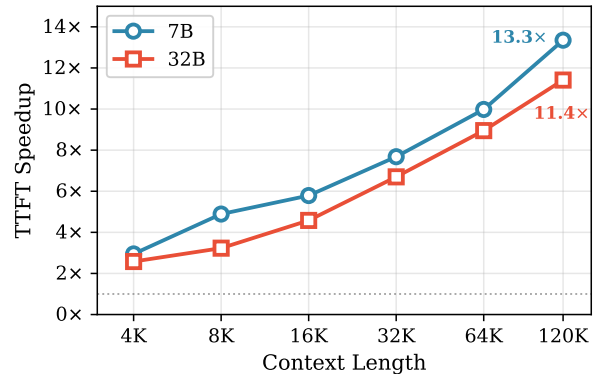


Figure 10: TTFT speedup vs. context length. With pre-computed memory embeddings, our 16× compression achieves up to 13.3× (7B) and 11.4× (32B) TTFT reduction at 120K tokens.

We analyze inference efficiency by measuring Time To First Token (TTFT) and peak GPU memory for our RL models (7B RL and 32B RL) with 16× compression. Since SFT and RL models share identical architectures, differing only in decoder weights, their latency characteristics are equivalent. Following practical deployment scenarios where document embeddings are cached (e.g., in RAG systems), we assume memory embeddings are pre-computed offline by the encoder. All experiments use text samples from DCLM (Li et al., 2024), FlashAttention-2 (Dao, 2024), float16 precision, and greedy decoding on H100 80GB GPUs (1 GPU for 7B, 4 GPUs for 32B). We report means over 20 runs per configuration after 3 warmup iterations.

Table 13 and Figure 10 show that TTFT speedup scales with context length, reaching 13.3× for 7B and 11.4× for 32B at 120K tokens. Memory reduction follows a similar trend, with peak GPU memory reduced by 2.0× (7B) and 2.2× (32B) at 120K tokens. Combined with the near-lossless downstream performance shown in Section 5, these efficiency gains make our approach practical for long-context applications.

Metric	32B			7B		
	Full	SFT	RL	Full	SFT	RL
Average reconstructed words per response (multi-hop, $n=8$)	8.3	8.8	64.2	6.8	7.4	38.2
Average reconstructed words per response (multi-hop, $n=15$)	4.4	2.6	45.1	3.2	2.0	19.0
% of context words reconstructed (all tasks, $n=5$)	1.2	1.0	3.2	1.3	0.9	2.0
Average response length in tokens (all tasks)	169	135	190	155	132	151

Table 12: Reconstruction volume (mean verbatim words per response on multi-hop tasks) versus mean response length (tokens, all tasks). RL reconstructs substantially more verbatim content per response than SFT or the full-context baseline, yet total output length stays comparable to the full-context baseline.

Context	TTFT (ms)			Peak Memory (GB)		
	Full	Ours	Speedup	Full	Ours	Saving
<i>7B Models</i>						
4K	126	43	3.0×	15.3	14.7	1.04×
8K	261	53	4.9×	16.1	14.9	1.08×
16K	577	100	5.8×	17.7	15.1	1.17×
32K	1,450	189	7.7×	20.8	15.7	1.32×
64K	4,071	408	10.0×	27.1	16.9	1.60×
120K	11,241	842	13.3×	37.5	18.9	1.99×
<i>32B Models</i>						
4K	622	241	2.6×	15.6	14.7	1.06×
8K	1,136	352	3.2×	16.6	14.9	1.11×
16K	2,384	521	4.6×	18.7	15.3	1.22×
32K	5,662	845	6.7×	22.8	16.1	1.42×
64K	15,104	1,688	8.9×	31.1	17.6	1.77×
120K	39,900	3,497	11.4×	44.8	20.2	2.22×

Table 13: TTFT and peak memory comparison. **Full**: full-context Qwen2.5-Instruct baseline processing all input tokens. **Ours**: our 16× compression model with precomputed memory embeddings.

K Training Cost Analysis

Model	GPUs	CPT	SFT	RL	Total
<i>Wall-clock time</i>					
7B	4×H100	20h	7h	80h	107h
32B	8×H100	48h	17h	86h	151h
<i>GPU-hours</i>					
7B	–	80	28	320	428
32B	–	384	136	688	1,208

Table 14: Training cost breakdown by stage. All experiments use H100 80GB GPUs.

Table 14 summarizes the training cost of our three-stage pipeline. The total cost is 428 GPU-hours for 7B and 1,208 GPU-hours for 32B. Our continual pretraining and SFT stages remain moderate in scale (108 and 520 GPU-hours for 7B and 32B, respectively), and already outperform prior methods even without RL (Section 5). The RL stage accounts for the majority of compute (75% at 7B, 57% at 32B) due to multi-rollout genera-

tion (16 rollouts for 7B and 8 for 32B), but drives the largest gains in downstream performance and length generalization.

Training is incurred once, whereas inference cost scales with usage. This tradeoff is more favorable in long-context or high-throughput deployment, where compression yields up to 13.3× TTFT speedup and 2.0× peak memory reduction at 120K tokens (Appendix J).

L Prompt Templates

This appendix details the prompt templates used for training and evaluation.

L.1 Training Prompts

All compression tasks follow the unified formulation in Equation 2. For reconstruction, continuation, and summarization tasks, instructions are randomly sampled from pools of varying lengths (5–8 variants each) to improve robustness. Figure 11 shows representative examples. For QA tasks, the question itself serves as the instruction. Figure 12 illustrates the input formats used across training stages: Stage 1 uses the pretraining format (flat sequence), while Stages 2–3 use the ChatML format.

L.2 Evaluation Prompts

Reconstruction Evaluation We use the same prompt format as Stage 1 continual pretraining (Figure 12) without the target portion, with the instruction “Reconstruct the original text from the compressed memory provided above.”

PISCO/OSCAR Baseline Models For QA benchmarks and NIAH evaluation, PISCO and OSCAR use their built-in inference interface with the context passed as compressed documents and the question as input. See Generation Parameters (Appendix M) for details on their compression mechanism. For QA benchmarks, we pass the question

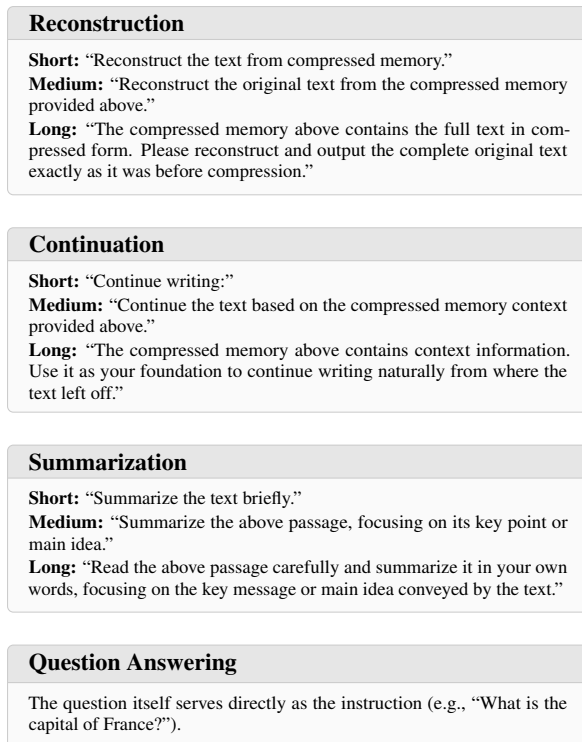


Figure 11: Task instruction examples. Each task samples from pools of 5–8 variants per length category during training to improve robustness.

directly; for NIAH, we use the specific question format shown in Figure 13.

QA Benchmarks & LongBench v2 For Qwen-based models, we use the same prompt format as training (Figure 12), without the target portion. Our SFT model uses the Stage 2 template, while our RL model uses the Stage 3 template with `\boxed{}` instruction.

NIAH Following the official implementation (Kamradt, 2023), we use the default needle statement: “The best thing to do in San Francisco is eat a sandwich and sit in Dolores Park on a sunny day” with the question: “What is the best thing to do in San Francisco?”. Figure 13 shows the prompts for all models.

Long-Context Multi-Hop QA Figure 14 shows the prompts for RULER-style evaluation. Our RL model uses a simpler prompt since it outputs answers in `\boxed{}` format from training.

M Evaluation Setup

This appendix details the benchmark configurations, generation parameters, and judgement criteria used for evaluation.

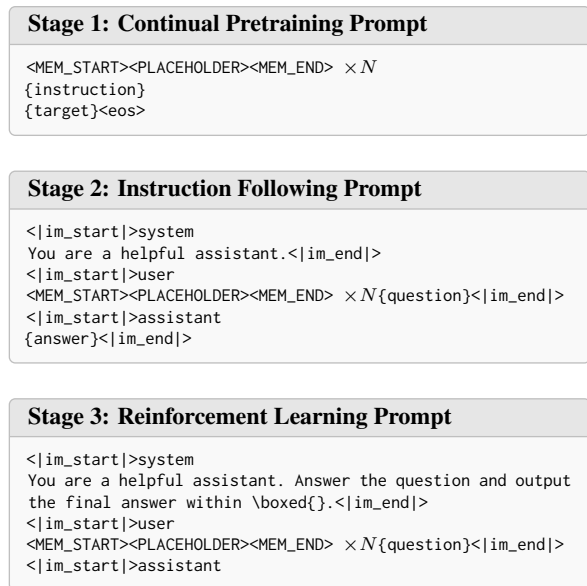


Figure 12: Training prompt templates across training stages. Memory placeholders are replaced with actual memory embeddings during forward pass. Stage 1 uses flat sequences; Stages 2–3 use ChatML with different system prompts. RL includes `\boxed{}` instruction for structured answer extraction.

M.1 Benchmark Configurations

QA Benchmarks We sample 1,000 instances from each benchmark. Specifically, we use the distractor setting for HotpotQA (Yang et al., 2018), the RC split for TriviaQA (Joshi et al., 2017), the v1.1 versions of SQuAD (Rajpurkar et al., 2016) and MS MARCO (Nguyen et al., 2016), and MuSiQue-Ans (Trivedi et al., 2022). We evaluate on the public development or validation splits, except for MS MARCO, which uses the public test split.

Long-Context Multi-Hop QA We sample 100 questions each from HotpotQA, 2WikiMultihopQA (Ho et al., 2020), and MuSiQue, using the same distributions as our QA benchmarks. Following RULER (Hsieh et al., 2024), we keep each question’s gold supporting passages and insert distractor paragraphs sampled from a global pool, shuffled to fill the target context length.

LongBench v2 We use LongBench v2 (Bai et al., 2025) as released, since it provides real-world long documents that already serve as authentic long-form contexts.

M.2 Generation Parameters

All Qwen-based models (Qwen2.5-Instruct and our SFT/RL) at both 7B and 32B scales

Qwen Baseline & Our SFT Model
System: You are a helpful assistant. User: {context with needle} User: Question: {question} Your final answer should be the complete sentence from the original document.
Our RL Model
System: You are a helpful assistant. Answer the question and output the final answer within \boxed{ }. User: {context with needle} User: Question: {question} Your final answer should be the complete sentence from the original document.
PISCO / OSCAR (Question Input)
Question: {question} Your final answer should be the complete sentence from the original document.

Figure 13: NIAH evaluation prompts. The needle is inserted at varying depths within Paul Graham essays, and models must retrieve the sentence.

Qwen Baseline & Our SFT Model
Answer the question based on the given documents. Please explicitly state your final answer in the last line as "Answer: [your final answer]". The following are given documents. {context} Answer the question based on the given documents. Please explicitly state your final answer in the last line as "Answer: [your final answer]". Question: {query}
Our RL Model
Answer the question based on the given documents. The following are given documents. {context} Answer the question based on the given documents. Question: {query}

Figure 14: Long-context multi-hop QA evaluation prompts.

share the same generation settings using official Qwen2.5 defaults: $\text{top}_k=20$, $\text{top}_p=0.8$, $\text{repetition_penalty}=1.05$. For QA benchmarks, LongBench v2, and multi-hop QA, we set $\text{temperature}=0.7$. For NIAH, we set $\text{temperature}=0.0$ following the benchmark’s default deterministic setting. For reconstruction evaluation, we use $\text{temperature}=0.1$ to encourage more deterministic outputs.

For baseline compression models (PISCO-Llama-8B, OSCAR-Mistral-7B, OSCAR-Mistral-Small-24B), we use their official model weights and built-in `generate_from_text()` method. These models chunk context into ~ 128 token segments, compressing each into 8 embeddings ($16\times$ compression). OSCAR uses query-dependent com-

pression where the question influences encoding. Following official defaults, all compression models use greedy decoding. For fair comparison, we also use greedy decoding for their backbone models (Llama-3.1-8B-Instruct for PISCO, Mistral-7B-Instruct-v0.2 and Mistral-Small-24B-Instruct for OSCAR).

M.3 Judgement Criteria

QA Benchmarks & LongBench v2 We employ Qwen2.5-72B-Instruct as a judge to assess answer correctness. Figure 15 shows the evaluation prompt.

QA Judgement Prompt
System: You are an expert evaluator for question-answering tasks. User: You are an expert evaluator. Your task is to determine if the generated answer is correct. === QUESTION === {question} === GROUND TRUTH ANSWER === {ground_truth} === GENERATED ANSWER === {generated_answer} === EVALUATION TASK === Compare the generated answer to the ground truth answer(s) and determine if the generated answer is semantically correct. If multiple ground truth answers are provided, the generated answer is correct if it matches ANY ONE of them. Answer with ONLY “yes” or “no”, followed by a brief explanation.

Figure 15: Judgement prompt for QA benchmarks and LongBench v2.

NIAH We use Qwen2.5-72B-Instruct as the evaluation judge with a score-based criteria (1–10 scale). Figure 16 shows the scoring rubric.

NIAH Scoring Rubric
Score 1: The answer is completely unrelated to the reference, the model refuses to answer, no actual answer is provided, or the answer is nonsensical. Score 3: The answer is marginally relevant but does not align with the reference. Score 5: The answer has moderate relevance but does not align with the reference. Score 7: The answer mostly aligns with the reference but contains clear inaccuracies or omissions. Score 10: The answer accurately aligns with the reference (reasonable wording variations are acceptable). <i>Important:</i> - Any answer claiming information is unavailable or not found gets Score 1. - Respond with only one of the following numerical scores: 1, 3, 5, 7, or 10.

Figure 16: NIAH scoring rubric (1–10 scale).

Long-Context Multi-Hop QA Following the setup of RULER (Hsieh et al., 2024), we use sub-span exact match for evaluation. For baseline and SFT models, we extract the final answer following

“Answer:” pattern; if extraction fails, we use the last 20 words. For our RL model, we extract from `\boxed{}`; if extraction fails, we use the last 20 words.

N Cross-Judge Evaluation

To verify that our results are not specific to the choice of evaluation judge, we re-evaluate all models using GPT-4o (Hurst et al., 2024) as an alternative judge with temperature 0.0 and the same judging prompt (Appendix M.3).

Table 15 shows that the results are consistent with our primary evaluation in Table 3. Relative rankings are preserved across all model scales and compression rates: RL consistently outperforms SFT, which in turn outperforms prior compression baselines. At $16\times$ compression, the 7B RL model retains 97.5% of full-context in-distribution performance and 94.6% out-of-distribution. At 32B scale, retention reaches 98.9% in-distribution and 99.8% out-of-distribution, closely matching the rates observed under the Qwen2.5-72B-Instruct judge. The performance margins between methods are also stable across judges: RL outperforms SFT by approximately 5–10 points under the Qwen2.5-72B-Instruct judge and 6–10 points under GPT-4o. Similarly, RL outperforms prior compression baselines by roughly 8–14 points under Qwen2.5-72B-Instruct and 9–15 points under GPT-4o. This consistency suggests that our findings are not sensitive to the choice of evaluation model.

O Training Hyperparameters

We provide detailed hyperparameters for all training stages in Tables 16 and 17. For CPT and SFT, we use FQLoRA (O.1) together with DeepSpeed ZeRO-3 (Rajbhandari et al., 2020). For RL, we use VERL (Sheng et al., 2025). All experiments are conducted on H100 80GB GPUs, using 4 GPUs for 7B models and 8 GPUs for 32B models.

O.1 Hybrid Fine-tuning Strategy (FQLoRA)

We propose FQLoRA, a hybrid strategy that fully fine-tunes selected transformer layers while applying QLoRA (Dettmers et al., 2023) to the rest. This combines the expressiveness of full fine-tuning with the efficiency of parameter-efficient methods.

We select layers at uniform intervals throughout the model. For Qwen2.5-7B with 28 layers, we fully fine-tune layers 0, 3, 6, 9, 12, 15, 18, 21, 24, and 27. For Qwen2.5-32B with 64 layers, we fully

fine-tune 23 layers at similar intervals. We always include the embedding layer and language model head.

As detailed in Table 16, we use different learning rates (LR) for different parameter groups. LoRA adapters on non-selected layers use $1e-4$. Fully fine-tuned layers use a lower rate of $5e-5$ for CPT and $2e-5$ for SFT, since these weights are initialized from the pretrained model rather than near-zero. The memory pooling projection, which is randomly initialized, uses a higher rate of $3e-4$ in CPT-1 and $2e-4$ in CPT-2 to accelerate early learning. During SFT, the projection layer is frozen.

O.2 Continual Pretraining and Supervised Fine-tuning

Table 16 shows the hyperparameters for continual pretraining and supervised fine-tuning. All stages use a cosine learning rate schedule with early stopping based on validation loss. In CPT-1, examples are truncated or padded to 2048 tokens without sequence grouping. In CPT-2 and SFT, training data are grouped into fixed-length sequences of 2048 and 4096 tokens, respectively.

O.3 Reinforcement Learning

Table 17 shows RL hyperparameters. We use VERL with FSDP backend for training and vLLM (Kwon et al., 2023) for rollout generation, leveraging vLLM’s prompt embeddings feature (vLLM Project, 2025) to directly pass memory embeddings during inference. We use DAPO (Yu et al., 2025), which builds on GRPO (Shao et al., 2024) with several improvements for stable training. Memory embeddings are pre-computed from the SFT checkpoint, and only the decoder is trained during RL. Additionally, we add a 25-step warmup before enabling the dynamic sampling to let the model first learn the output format. We use 16 rollouts per prompt for 7B and 8 for 32B due to memory constraints. For reward modeling, we use Qwen3-32B for the 7B model and Qwen2.5-72B-Instruct for the 32B model.

P Implicit Expansion Behavior Examples

We present examples of implicit expansion behavior learned through RL. All examples use 32B models with $16\times$ compression.

P.1 In-Distribution Examples

Figure 17 shows implicit expansion on a training example. The model first expands actor filmogra-

Model	Comp.	In-Distribution				Out-of-Distribution					
		HotpotQA	NQ	TriviaQA	Avg.	2Wiki	BioASQ	SQuAD*	MuSiQue	MARCO*	Avg.
<i>7B Models — Full Context</i>											
Mistral-7B-Instruct-v0.2	—	79.20	79.70	92.20	83.70	62.20	94.40	90.20	43.40	68.70	71.78
Qwen2.5-7B-Instruct	—	83.70	83.30	91.60	86.20	67.60	93.20	93.30	47.20	66.10	73.48
Llama-3.1-8B-Instruct	—	88.10	85.30	92.30	88.57	73.10	95.20	92.70	55.70	65.60	76.46
<i>7B Models — 16× Compression</i>											
PISCO ^a	16×	57.70	69.70	89.10	72.17	34.70	87.40	81.50*	24.20	61.80*	57.92
OSCAR ^b	16×	61.10	67.60	87.80	72.17	39.80	82.10	86.60*	27.00	64.70*	60.04
Ours (SFT) ^c	16×	74.00	69.60	86.90	76.83	65.50	70.90	71.90	36.20	57.00	60.30
Ours (RL) ^c	16×	85.10	76.40	90.70	84.07	69.40	86.70	81.80	48.10	61.50	69.50
<i>7B Models — 64× Compression</i>											
Ours (SFT) ^c	64×	50.40	51.10	79.00	60.17	39.40	57.20	43.80	17.90	48.20	41.30
Ours (RL) ^c	64×	64.20	60.90	87.30	70.80	39.20	81.30	59.60	25.60	52.70	51.68
<i>32B Models — Full Context</i>											
Qwen2.5-32B-Instruct	—	89.20	85.80	93.90	89.63	79.60	94.90	94.80	56.70	69.30	79.06
Mistral-Small-24B-Instruct	—	93.00	83.20	94.40	90.20	81.20	95.90	93.30	70.70	63.60	80.94
<i>32B Models — 16× Compression</i>											
OSCAR-24B ^d	16×	65.70	73.80	90.30	76.60	46.70	87.40	87.30*	29.60	66.90*	63.58
Ours (SFT) ^c	16×	81.60	77.10	89.80	82.83	73.40	81.60	80.90	49.50	58.10	68.70
Ours (RL) ^c	16×	90.40	82.50	93.10	88.67	81.80	93.10	90.80	65.20	63.50	78.88

Table 15: QA evaluation using GPT-4o as the judge model. Results are consistent with Table 3 (Qwen2.5-72B-Instruct judge), with consistent relative rankings and margins across all model scales and compression rates. Backbones: ^aLlama-3.1-8B-Instruct, ^bMistral-7B-Instruct-v0.2, ^cQwen2.5-7B, ^dMistral-Small-24B-Instruct, ^eQwen2.5-32B. *These datasets were included in OSCAR/PISCO training, so results may not reflect fair comparison.

Parameter	CPT-1	CPT-2	SFT
<i>Stage-specific</i>			
Full FT LR	5e-5	5e-5	2e-5
Memory Proj. LR	3e-4	2e-4	—
Global Batch Size	256	128	64
Warmup Ratio	4%	3%	3%
Min LR Ratio	10%	2%	10%
Sequence Length	2048	2048	4096
<i>Shared across stages</i>			
LoRA LR		1e-4	
Weight Decay (LoRA / Full FT)		0 / 0.1	
Max Grad Norm		1.0	
LoRA Rank / Alpha / Dropout	64 / 16 / 0.05		
Precision		BF16	

Table 16: Continual pretraining (CPT) and supervised fine-tuning (SFT) hyperparameters.

phy to identify the film, then expands film metadata to extract the director, forming a multi-hop reasoning chain by selectively unpacking information at each step. Figures 1 and 18 compare RL against SFT on in-distribution evaluation data.

These comparisons reveal a key limitation of SFT. While SFT models can retrieve relevant information, they are prone to hallucination due to detail confusion. When information is compressed, related concepts become entangled, causing SFT to conflate details during decoding. In Figure 1, SFT hallucinates “Biberon” as Ocnus’s father instead of the correct “Tiberinus, king of Alba Longa.” In

Figure 18, SFT identifies Rangers as the winner but hallucinates the statistics (152 and 59 instead of the correct 159 and 152).

RL addresses these issues by teaching the model to expand details selectively while maintaining coherent reasoning. Rather than conflating related concepts, the RL model unpacks specific facts at each reasoning step, yielding more precise retrieval. This expansion is targeted, reconstructing only the information necessary to answer the question while preserving the computational benefits of compression.

P.2 Out-of-Distribution Examples

Figures 19 and 20 show examples from MuSiQue, a challenging multi-hop benchmark unseen during training. Both examples show cases where our 32B RL model succeeds while the full-context Qwen2.5-32B-Instruct baseline fails. As discussed in Section 5, compression may act as information distillation that filters noise and surfaces task-relevant content, aiding multi-hop reasoning.

Question: Who directed the film which had Luke Goss playing the role of Jared Nomak?

Response: To determine who directed the film in which Luke Goss played the role of Jared Nomak, we need to identify the film where he took on this role. From the provided information:
“Luke Damon Goss (born 29 September 1968) is an English actor and former member of late 1980s band Bros. He has appeared in numerous films including “Blade II” (2002) as Jared Nomak, “One Night with the King” (2006) as King Xerxes, “Hellboy II: The Golden Army” (2008) as Prince Nuada, “Tekken” (2010) as Steve Fox and “Interview with a Hitman” (2012) as Viktor.”
 From this, we can see that Luke Goss played Jared Nomak in “Blade II” (2002). To find the director of “Blade II,” we refer to the information provided about “Blade II”:
“Blade II is a 2002 American superhero horror film based on the fictional Marvel Comics character Blade. It is the sequel of the first film and the second part of the “Blade” film series. It was written by David S. Goyer, who also wrote the previous film, directed by Guillermo del Toro, and had Wesley Snipes returning as the lead character and producer.”
 Therefore, the director of “Blade II” is Guillermo del Toro.

Answer: Guillermo del Toro

Figure 17: **Multi-Hop Reasoning Example.** The model first expands actor filmography from compressed context to identify the film, then expands film metadata to extract the director.

Question: Who has won the most Old Firm games?	
Ground Truth: Rangers have won 159 matches	
32B RL (16×) — Correct	32B SFT (16×) — Incorrect
<p>According to the information provided in the Wikipedia excerpt, Rangers have won the most Old Firm games. Specifically, the table states: <i>“Most wins: Rangers (159)”</i> Additionally, the text mentions that <i>“Rangers have won 159 matches, Celtic 152 matches and 98 ended in a draw.”</i> Therefore, the answer is:</p> <p style="border: 1px solid black; padding: 2px; display: inline-block;">Rangers (159 wins)</p>	<p>Based on the information provided in the passage, Rangers have won the most Old Firm games: Rangers and Celtic have played each other 409 times in major competitions: Rangers have won 152 matches, Celtic 59 matches and 192 ended as a draw. So Rangers have won 152 of the 409 Old Firm games played between the two clubs.</p> <p style="border: 1px solid black; padding: 2px; display: inline-block;">Rangers (152 wins)</p>

Figure 18: **Natural Questions Example.** The 32B RL model accurately extracts the correct statistic (159 wins) from the compressed context, citing both the summary table and detailed text. The SFT model fabricates incorrect statistics (Rangers 152 wins, Celtic 59 wins, and 192 draws) that do not match the source text, demonstrating imprecise information retrieval from compressed representations.

Question: In what region of the country where An Phu is located is John Phan’s birthplace?	
Ground Truth: South Central Coast	Qwen2.5-32B-Instruct — Incorrect
32B RL (16×) — Correct	
<p>To determine the region of the country where John Phan’s birthplace is located, we need to identify the relevant information from the provided texts.</p> <p>1. Identify An Phú’s location: “An Phú District is a district of An Giang Province in the Mekong Delta region of Vietnam on the border with Cambodia.”</p> <p>2. Identify John Phan’s birthplace: “John Phan (born October 10, 1974 in Da Nang, Vietnam) is a Vietnamese-American professional poker player based in Stockton, California.”</p> <p>From this information, we know that An Phú is located in the Mekong Delta region of Vietnam. However, we need to find out which region Da Nang, John Phan’s birthplace, is located in.</p> <p>3. Determine Da Nang’s region: “South Central Coast is one of the regions of Vietnam. It consists of the independent municipality of Da Nang and seven provinces.”</p> <p>Therefore, John Phan’s birthplace, Da Nang, is located in the South Central Coast region of Vietnam.</p> <p>South Central Coast region of Vietnam</p>	<p>Based on the information provided:</p> <ul style="list-style-type: none"> • An Phú District is located in the Mekong Delta region of Vietnam. • John Phan was born in Da Nang, Vietnam. <p>Da Nang is located in Central Vietnam, not in the Mekong Delta region where An Phú District is located. Therefore, John Phan’s birthplace (Da Nang) is in a different region of Vietnam compared to where An Phú District is located.</p>

Figure 19: **MuSiQue Example 1.** The 32B RL model correctly parses the question structure, recognizing that “the country where An Phu is located” establishes Vietnam as context, while the actual query asks for the region containing John Phan’s birthplace (Da Nang). The model traces: (1) An Phu → Vietnam, (2) John Phan → Da Nang, (3) Da Nang → South Central Coast. The baseline misinterprets the question as comparative, asking whether Da Nang and An Phu share the same region.

Question: In the state that includes Walnut Spring, what’s the typical winter daytime temperature range in the region of Richmond?	
Ground Truth: upper 40s–lower 50s °F	Qwen2.5-32B-Instruct — Incorrect
32B RL (16×) — Correct	
<p>To determine the typical winter daytime temperature range in the region of Richmond, we need to identify the state that includes Walnut Spring and then find the relevant information about Richmond’s climate.</p> <p>1. Identify Walnut Spring’s state: “Walnut Spring is a historic home located near Kanodes Mill, Montgomery County, Virginia.”</p> <p>This indicates that Walnut Spring is located in Virginia.</p> <p>2. Identify Richmond’s region: “Richmond is located in the Piedmont region of Virginia, at the highest navigable point of the James River.”</p> <p>3. Find winter temperature data: “In the winter, temperatures are typically in the upper 40s–lower 50s °F (8–12 °C) during the day and often dropping below the freezing point at night.”</p> <p>Therefore, the typical winter daytime temperature range in the region of Richmond is upper 40s–lower 50s °F.</p> <p>upper 40s–lower 50s °F (8–12 °C)</p>	<p>Based on the information provided, we don’t have specific winter temperature data for the region around Walnut Spring. However, we do have climate details for Richmond, Virginia, which is located in the same state (Virginia) and is also situated in the Piedmont region.</p> <p>From the Richmond, Virginia section: “Winters are cool, with January averaging 35.5 °F (2 °C).”</p> <p>While this gives us the average January temperature, it doesn’t specify a range. However, we can make an educated guess based on typical Piedmont winter temperatures. Generally, in the Piedmont region of Virginia during winter:</p> <ul style="list-style-type: none"> • Daytime highs: around 35–55°F • Nighttime lows: around 20–40°F

Figure 20: **MuSiQue Example 2:** The 32B RL model correctly extracts the explicit “upper 40s–lower 50s °F” daytime temperature from the compressed context. The baseline confuses mean daily temperature (35.5°F) with daytime highs, then extrapolates an incorrect range. This demonstrates the RL model’s superior ability to precisely extract and apply relevant information from compressed representations.

Parameter	Value
<i>Optimization</i>	
Learning Rate	1e-6
Warmup Steps	10
Batch Size	128
Mini Batch Size	64
<i>DAPO Algorithm</i>	
Advantage Estimator	GRPO
Loss Aggregation	token-mean
Clip Ratio Low	0.2
Clip Ratio High	0.28
KL Coefficient	0
Entropy Coefficient	0
<i>Dynamic Sampling</i>	
Filter Metric	accuracy
Warmup Steps	25
<i>Overlong Reward Shaping</i>	
Buffer Length	100
Penalty Factor	0.5
<i>Generation</i>	
Rollouts per Prompt	8/16
Temperature	1.0
Max Prompt Length	1024
Max Response Length	600

Table 17: Reinforcement learning hyperparameters using DAPO.