

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

Anonymous ACL submission

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 pretraining, 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 $16\times$ compression 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, Long-Bench v2, and multi-hop reasoning.

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., 2025), 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. (2025), high-fidelity reconstruction does not necessarily translate into language modeling utility. A model that faithfully recovers compressed texts may still fail on effective next-token prediction, a gap evidenced in actual downstream performance (Liu et al., 2025). 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., 2025b) 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 tokens. Because reconstruction occurs on demand during generation rather than all at once, 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 and Addis, 2007; Nader et al., 2000). This analogy offers a cognitive lens for understanding how

INPUT: Question and Compressed Context

QUESTION: In relation to Rome, where was the city whose king was the father of Ocnus?

COMPRESSED CONTEXT SNIPPETS:
 ... **Title: Ocnus** In Greek and Roman mythology, Ocnus or Bianor was a son of Manto and Tiberinus, king of Alba Longa. He founded modern Mantua ...
 ... **Title: Alba Longa** Alba Longa (occasionally written Albalonga in Italian sources) was an ancient city of Latium in central Italy, 12 mi southeast ...

✓ With RL: Implicit Expansion	✗ SFT Only: Hallucination
<p>To determine the location of the city, we need to identify the relevant information from the provided descriptions. From the title "Ocnus":</p> <p><i>"In Greek and Roman mythology, Ocnus or Bianor was a son of Manto and Tiberinus, king of Alba Longa. He founded modern Mantua in honor of his mother."</i></p> <p>This description tells us that Ocnus's father, Tiberinus, was the king of Alba Longa. Now, we need to find out where Alba Longa was located. From the title "Alba Longa":</p> <p><i>"Alba Longa (occasionally written Albalonga in Italian sources) was an ancient city of Latium in central Italy, 12 mi southeast of Rome, in the Alban Hills."</i></p> <p>Based on this information, we can conclude that Alba Longa, the city whose king was the father of Ocnus, was located 12 miles southeast of Rome, in the Alban Hills.</p>	<p>To determine the location of the city, we need to identify the relevant information from the provided texts.</p> <ol style="list-style-type: none"> Ocnus and Tiber: The text states that Ocnus and Manto were the children of Biberon, who was a king of Tiber. [hallucinated] Tiber: Tiber is not explicitly mentioned as a city in any of the provided texts. However, it is commonly known that the Tiber River flows through Rome, and ancient cities often developed around significant rivers. <p>Given this information, we can infer that Tiber likely refers to a city or settlement located near the Tiber River. Since the Tiber River is central to Roman history and geography, it is reasonable to conclude that the city of Tiber was in the vicinity of what would later become Rome. [hallucinated]</p> <p>Therefore, the city whose king was the father of Ocnus was located near the Tiber River, which is in central Italy, close to the area where Rome would eventually be founded.</p>

Figure 1: **Implicit expansion via RL bridges memorization and utilization.** Given compressed context (16×), 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.

LLMs utilize compressed memories, and may explain why finetuning, which accesses compressed information without reconstruction, proves 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 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× compression at 7B and 32B scales, recovering 98% of full-context QA performance and outperforming prior methods like PISCO (Louis et al., 2025a) and OSCAR (Louis et al., 2025b) 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., 2024), and multi-hop reasoning, while achieving up to 11.4× speedup at extreme lengths. We note that the 16× compression rate refers to encoding-level compression, consistent with prior work; with special tokens required for decoding, the effective rate is approximately 5.3×. We present the main results in Section 5, with latency analysis detailed in Appendix E.

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., 2025a) simplifies training via sequence-level distillation without pretraining, and OSCAR (Louis et al., 2025b) introduces query-dependent compression. However, these methods suffer from semantic dilution: information is distributed unevenly across memory tokens (Tang et al., 2025).

Average Pooling Approaches To address semantic dilution, recent work directly pools encoder hidden states. GMSA (Tang et al., 2025) 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, risking 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 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., 2025; 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. Each memory embedding is wrapped by special 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.

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:

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

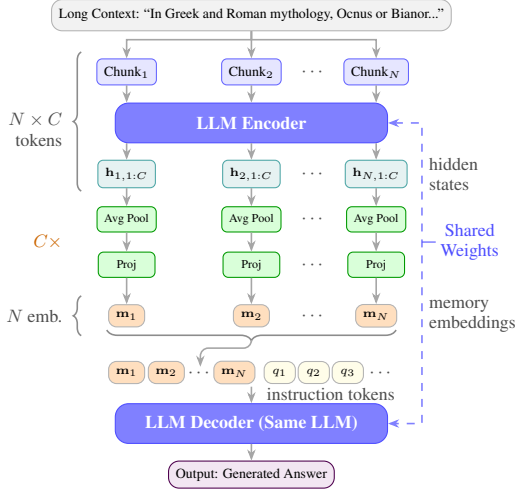


Figure 2: Context compression architecture. Input is chunked into C -token segments and encoded by the LLM. Hidden states are average-pooled and projected into memory embeddings, achieving $C\times$ compression. The same LLM decodes outputs from memory embeddings prepended to instructions.

where \mathbf{M} denotes the memory embeddings, \mathbf{x}_{inst} the task instructions (Figure 10), 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.

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

Task	Memory \mathbf{M}	Instruction \mathbf{x}_{inst}	Target y
Reconstruction	Full doc.	Prompt	Original
Continuation	First half	Prompt + bridges	Second half
Summarization	Full doc.	Prompt	Summary
QA	Context	Question	Answer

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 embed-

dings 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 G). 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 rea-

soning 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, condensing 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 Section 5.

4 Experiments

4.1 Implementation Details

We use Qwen2.5-7B (base) and Qwen2.5-32B (base) (Yang et al., 2024) as backbones, apply-

ing $16\times$ compression as our primary setting, with additional experiments at $64\times$. To balance training capacity with efficiency, we propose FQLoRA (Appendix I.1), a hybrid strategy that fully finetunes $\sim 36\%$ of transformer layers while applying QLoRA (Dettmers 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., 2024), with Qwen3-32B (Yang et al., 2025) and Qwen2.5-72B-Instruct serving as reward models for the 7B and 32B settings, respectively. Training configurations and hyperparameters are detailed in Appendix G.1 and I.

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 Natural Reasoning (Yuan et al., 2025) and Proof-Pile-2 (Azerbayev et al., 2024); code from StarCoder (Li et al., 2023); general instructions from OpenHermes-2.5 (Teknium, 2023); and QA data from HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), and RAGBench (Friel et al., 2024). 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). Details of the evaluation setup are provided in Appendix G.2 and H.

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 (Bajaj et al., 2016). We use benchmarks with answer-containing contexts to avoid retrieval sensitivity and establish accurate upper bounds for compression.

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

Stage	Task	Samples	Data Sources
CPT-1	Compression (recon/cont) Standard CPT	220K 100K	DCLM, CCI2, Natural Reasoning, 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

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., 2024) is a challenging multiple-choice benchmark requiring reasoning over real-world long documents; 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., 2025a) and OSCAR (Louis et al., 2025b). 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 (Jiang et al., 2023) for OSCAR, and Qwen2.5-7B/32B-Instruct (Yang et al., 2024) 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.

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. At $64\times$ compression, our 7B RL model still achieves 70.67%, comparable to OSCAR at $16\times$ 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 14–17). 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, and parallels the reconstructive nature of human memory. Without this behavior, SFT models tend to conflate entangled details and hallucinate more frequently. Appendix J

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.

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

Backbones: ^aLlama-3.1-8B-Instruct, ^bMistral-7B-Instruct-v0.2, ^cQwen2.5-7B (base), ^dMistral-Small-24B-Instruct, ^eQwen2.5-32B (base).

* These datasets were included in OSCAR/PISCO training, so results may not reflect fair comparison.

illustrates both behaviors.

This expansion incurs minimal overhead since it is reasoning-guided and localized. Response lengths remain comparable to full-context baselines (151 vs. 155 tokens at 7B, 190 vs. 169 at 32B), a negligible cost given 16× input reduction (see Appendix D for response analysis).

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 J.2 provides example responses.

5.2 Long-Context Generalization

Beyond standard QA tasks, we test generalization far exceeding the 4K-token training length (1.3K memory embeddings with wrapper 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 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 (Table 8).

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

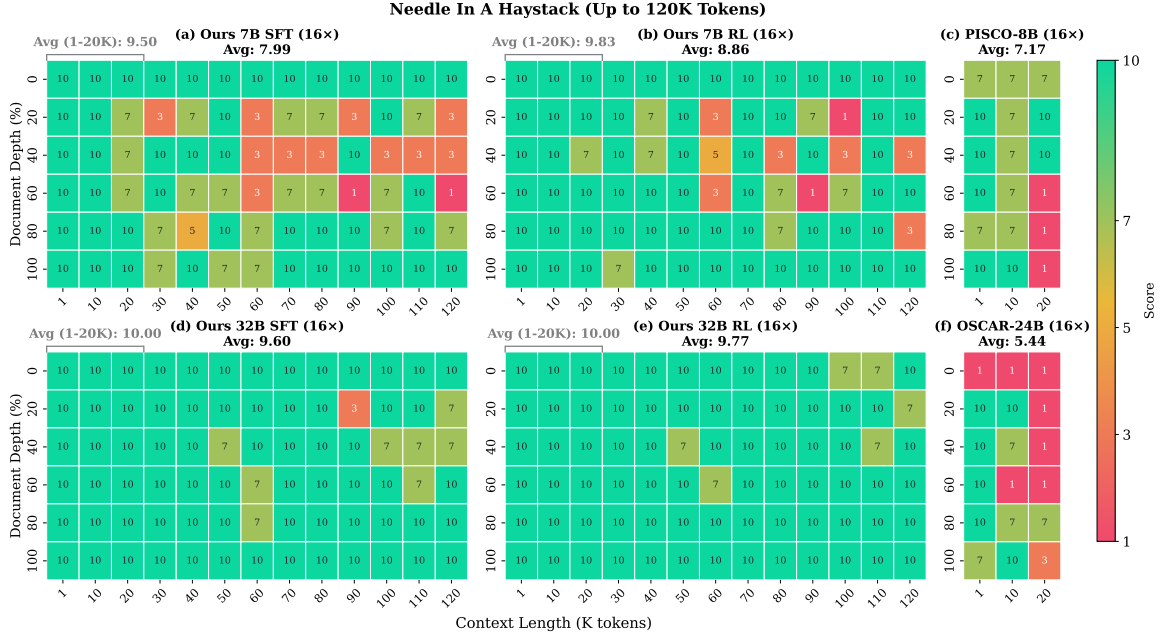


Figure 3: NIAH performance across context lengths up to 120K tokens. Our 32B RL models (16 \times) achieve near-perfect retrieval (9.77/10), while prior methods (PISCO, OSCAR) degrade after 10K and fail beyond 20K tokens due to memory or architectural constraints. Scores range from 1 (failure) to 10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

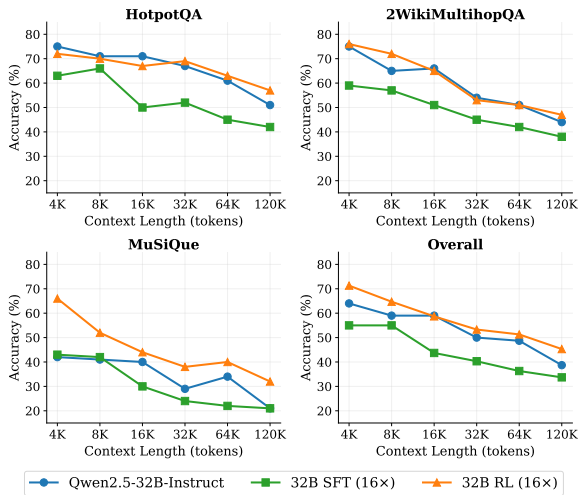


Figure 4: Multi-hop QA accuracy across 4K–120K contexts. The 32B RL model matches or outperforms the full-context baseline, particularly at longer contexts.

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 Conclusion

While soft prompt compression can preserve information with near-perfect fidelity, effective down-

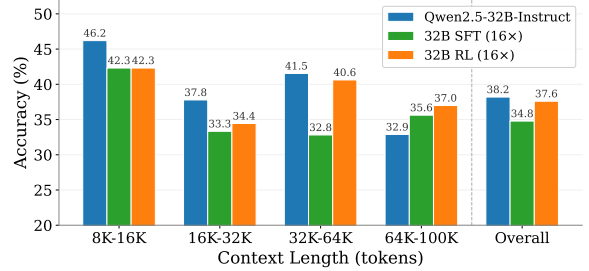


Figure 5: LongBench v2 accuracy across 8K–100K contexts. Our 32B RL model shows minimal degradation versus full-context on challenging reasoning tasks.

stream 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 tokens 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.

551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599

Limitations

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

Effective Compression Rate and Encoding Overhead

We report $16\times$ compression at the encoding level (16 tokens per memory embedding), consistent with prior work. In practice, each embedding requires two special tokens for decoding, yielding an effective rate of approximately $5.3\times$. This overhead should be considered when estimating real-world inference savings. And for the encoding overhead, compression reduces decoder tokens but requires encoder computation. Our approach is 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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen 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*. Introduces Proof-Pile-2 dataset.

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. 2024. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*. 600-603

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*. 604-610

Frederic C. Bartlett. 1932. *Remembering: A Study in Experimental and Social Psychology*. Cambridge University Press, Cambridge, UK. 611-613

Beijing Academy of Artificial Intelligence. 2024. Cci2-data. <https://huggingface.co/datasets/BAAI/CCI2-Data>. 614-616

Gabriele Berton, Jayakrishnan Unnikrishnan, Son Tran, and Mubarak Shah. 2025. CompLLM: Compression for long context q&a. *arXiv preprint arXiv:2509.19228*. 617-620

Shaoshen Chen, Yangning Li, Zishan Xu, Yinghui Li, Xin Su, Zifei Shan, and Hai-tao Zheng. 2025. DAST: Context-aware compression in llms via dynamic allocation of soft tokens. *arXiv preprint arXiv:2502.11493*. 621-625

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*. 626-630

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. *arXiv preprint arXiv:2508.19282*. 631-636

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*. 637-640

Yair Feldman and Yoav Artzi. 2025. Simple context compression: Mean-pooling and multi-ratio training. *arXiv preprint arXiv:2510.20797*. 641-643

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. RAGBench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*. 644-647

Tao Ge, Jing Hu, 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*. 648-651

653	Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and 1 others. 2024. The Llama 3 herd of models. <i>arXiv preprint arXiv:2407.21783</i> .	Ivan Yee Lee, Cheng Yang, and Taylor Berg-Kirkpatrick. 2025. Optical context compression is just (bad) autoencoding. <i>arXiv preprint arXiv:2512.03643</i> .	709
654			710
655			711
656	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 <i>Proceedings of the 28th International Conference on Computational Linguistics</i> .	Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, and 1 others. 2024. DataComp-LM: In search of the next generation of training sets for language models. In <i>Advances in Neural Information Processing Systems</i> , volume 37.	712
657			713
658			714
659			715
660			716
661	Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What’s the real context size of your long-context language models? In <i>Proceedings of the First Conference on Language Modeling (COLM)</i> .	Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023. StarCoder: may the source be with you! <i>arXiv preprint arXiv:2305.06161</i> .	717
662			718
663			719
664			720
665			721
666			722
667	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. <i>arXiv preprint arXiv:2310.06825</i> .	Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. 2025. REFRAG: Rethinking rag based decoding. <i>arXiv preprint arXiv:2509.01092</i> .	723
668			724
669			725
670			726
671			
672			
673			
674			
675	Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension . In <i>Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)</i> , pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.	Fanfan Liu and Haibo Qiu. 2025. Context cascade compression: Exploring the upper limits of text compression. <i>arXiv preprint arXiv:2511.15244</i> .	727
676			728
677			729
678			
679			
680			
681			
682	Greg Kamradt. 2023. Needle in a haystack - pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack .	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 . <i>Transactions of the Association for Computational Linguistics</i> , 12:157–173.	730
683			731
684			732
685	Eunki Kim, Sangryul Kim, and James Thorne. 2025. Learning to insert [PAUSE] tokens for better reasoning . In <i>Findings of the Association for Computational Linguistics: ACL 2025</i> , pages 23760–23777. Association for Computational Linguistics.	Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and Jingbo Zhu. 2025. Autoencoding-free context compression for LLMs via contextual semantic anchors. <i>arXiv preprint arXiv:2510.08907</i> .	733
686			734
687			735
688			736
689			737
690	Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. BioASQ-QA: A manually curated corpus for biomedical question answering . <i>Scientific Data</i> , 10(1):170.	Charles London and Varun Kanade. 2025. Pause tokens strictly increase the expressivity of constant-depth transformers. <i>arXiv preprint arXiv:2505.21024</i> .	741
691			742
692			743
693			744
694	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 . <i>Transactions of the Association for Computational Linguistics</i> , 7:452–466.	Maxime Louis, Herv�� D��jean, and St��phane Clinchant. 2025a. PISCO: Pretty simple compression for retrieval-augmented generation. <i>arXiv preprint arXiv:2501.16075</i> .	745
695			746
696			747
697			
698			
699			
700			
701			
702			
703	Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention . In <i>Proceedings of the 29th ACM Symposium on Operating Systems Principles</i> .	Maxime Louis, Thibault Formal, Herv�� D��jean, and St��phane Clinchant. 2025b. OSCAR: Online soft compression and reranking. <i>arXiv preprint arXiv:2504.07109</i> .	748
704			749
705			750
706			751
707			
708			
709			
710			
711			
712			
713			
714			
715			
716			
717			
718			
719			
720			
721			
722			
723			
724			
725			
726			
727			
728			
729			
730			
731			
732			
733			
734			
735			
736			
737			
738			
739			
740			
741			
742			
743			
744			
745			
746			
747			
748			
749			
750			
751			
752			
753			
754			
755			
756			
757			
758			
759			
760			
761			

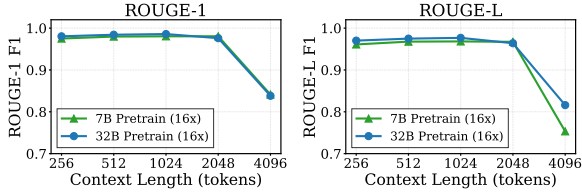


Figure 6: Reconstruction performance (ROUGE F1) across context lengths for continual pretrained 7B and 32B models with 16 \times compression. Both models achieve high fidelity reconstruction up to 2048 tokens.

Table 4: Reconstruction scores across context lengths for continual pretrained models.

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

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 compares the training and evaluation loss curves for average pooling versus attention-based pooling (Grouped-Query Attention) 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 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

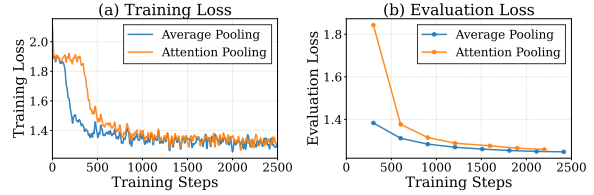


Figure 7: Training and evaluation loss comparison between average pooling and attention-based pooling (Grouped-Query Attention) during continual pretraining. Average pooling converges faster and achieves slightly lower evaluation loss.

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.

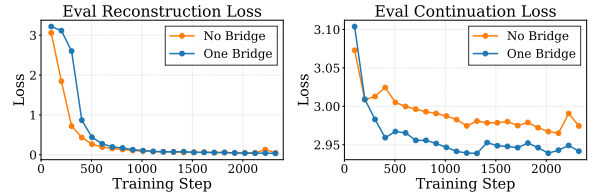


Figure 8: Evaluation loss comparison between configurations with no bridge chunks and one bridge chunk during continual pretraining.

Table 5: Final losses at training step 2300 for bridge chunk ablation.

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

D Response Length Analysis

We analyze response length to verify that our implicit expansion strategy incurs only modest overhead. Table 6 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 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).

Table 6: Average response length (in tokens) across all QA benchmarks (8,000 samples per model).

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

E Latency Analysis

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, Flash Attention 2, 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.

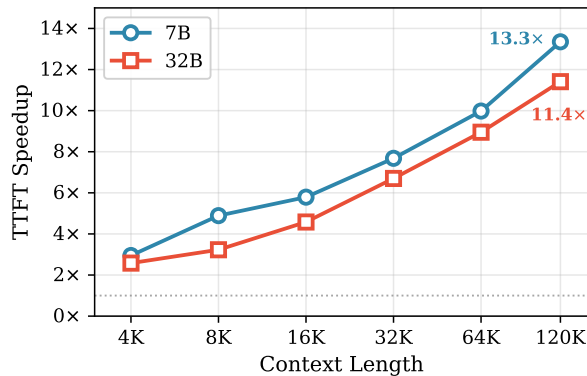


Figure 9: TTFT speedup vs. context length. With pre-computed memory embeddings, our 16× compression achieves up to 13.3× (7B) and 11.4× (32B) TTFT reduction at 120K tokens.

Table 7: 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.

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 7 and Figure 9 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.

F NIAH Results

Table 8 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.

G Prompt Templates

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

G.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 10 shows representative examples. For QA tasks, the question itself serves as the instruction.

Table 8: NIAH performance with score out of 10. Our models handle 120K contexts while prior compression methods fail beyond 20K.

Model	Comp.	Max Ctx	Score
<i>7B Models</i>			
Qwen2.5-7B-Instruct	–	120K	10.00
OSCAR-Mistral-7B ^a	16×	20K	1.50
PISCO-Llama-8B ^b	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 ^a	16×	20K	5.44
Ours (SFT)	16×	120K	9.60
Ours (RL)	16×	120K	9.77

^a Embedding dimension mismatch beyond 20K contexts.

^b OOM error beyond 20K contexts (tested on 4×H100 80GB).

Figure 11 illustrates the input formats used across training stages: Stage 1 uses the pretraining format (flat sequence), while Stages 2–3 use the ChatML format.

G.2 Evaluation Prompts

Reconstruction Evaluation We use the same prompt format as Stage 1 continual pretraining (Figure 11) 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 H) for details on their compression mechanism. For QA benchmarks, we pass the question directly; for NIAH, we use the specific question format shown in Figure 12.

QA Benchmarks & LongBench v2 For Qwen-based models, we use the same prompt format as training (Figure 11), 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 12 shows the prompts for all models.

Reconstruction

Short: “Reconstruct the text from compressed memory.”

Medium: “Reconstruct the original text from the compressed memory provided above.”

Long: “The compressed memory above contains the full text in compressed form. Please reconstruct and output the complete original text exactly as it was before compression.”

Continuation

Short: “Continue writing:”

Medium: “Continue the text based on the compressed memory context provided above.”

Long: “The compressed memory above contains context information. Use it as your foundation to continue writing naturally from where the text left off.”

Summarization

Short: “Summarize the text briefly.”

Medium: “Summarize the above passage, focusing on its key point or main idea.”

Long: “Read the above passage carefully and summarize it in your own words, focusing on the key message or main idea conveyed by the text.”

Question Answering

The question itself serves directly as the instruction (e.g., “What is the capital of France?”).

Figure 10: Task instruction examples. Each task samples from pools of 5–8 variants per length category during training to improve robustness.

Long-Context Multi-Hop QA Figure 13 shows the prompts for RULER-style evaluation. Our RL model uses a simpler prompt since it outputs answers in `\boxed{}` format from training.

H Evaluation Setup

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

H.1 Generation Parameters

All Qwen-based models (Qwen2.5-Instruct and our SFT/RL) at both 7B and 32B scales share the same generation settings using official Qwen2.5 defaults: `top_k=20`, `top_p=0.8`, `repetition_penalty=1.05`. For QA benchmarks, LongBench v2, and multi-hop QA, we set `temperature=0.7`. For NIAH, we set `temperature=0.0` following the benchmark’s default deterministic setting. For reconstruction evaluation, we use `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 seg-

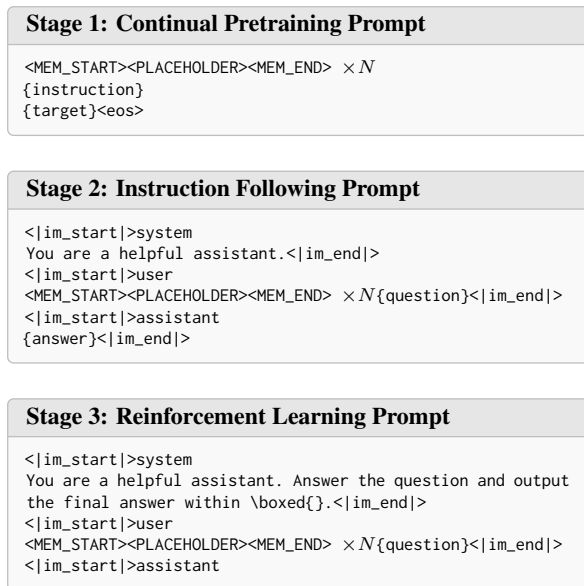
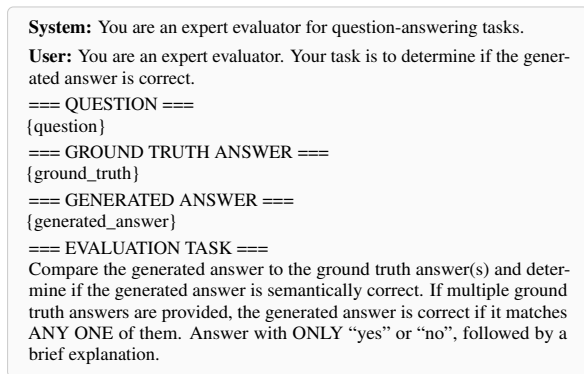


Figure 11: Training prompt templates across training stages. Memory placeholders are replaced with actual memory embeddings during forward pass. Stage 1 uses flat sequences; Stages 2–3 use ChatML with different system prompts. RL includes `\boxed{ }` instruction for structured answer extraction.

ments, compressing each into 8 embeddings (16× compression). OSCAR uses query-dependent compression 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).

H.2 Judgement Criteria

QA Benchmarks & LongBench v2 We employ Qwen2.5-72B-Instruct as a judge to assess answer correctness with the following evaluation criteria:



NIAH We use Qwen2.5-72B-Instruct as the evaluation judge with a score-based criteria (1–10

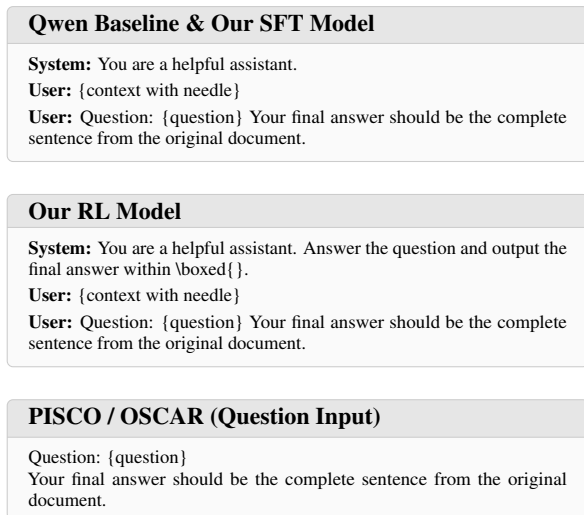


Figure 12: NIAH evaluation prompts. The needle is inserted at varying depths within Paul Graham essays, and models must retrieve the sentence.

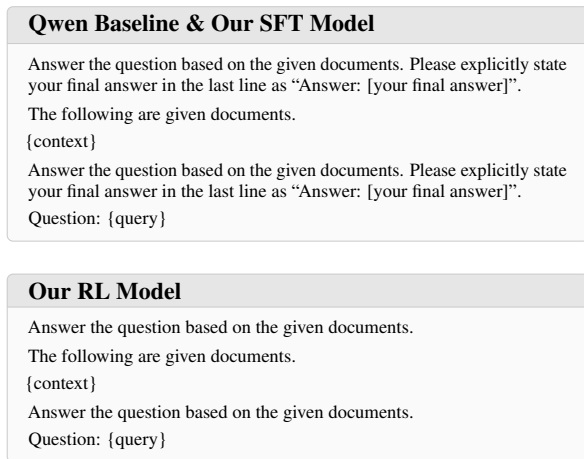
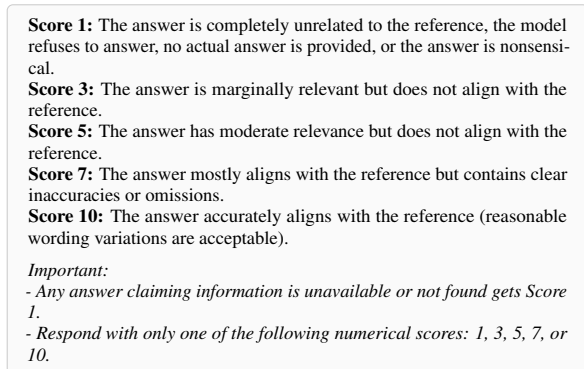


Figure 13: Long-context multi-hop QA evaluation prompts.

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.

I Training Hyperparameters

We provide detailed hyperparameters for all training stages in Tables 9 and 10. All experiments use H100 80GB GPUs, with 4 GPUs for 7B models and 8 GPUs for 32B models. We use DeepSpeed ZeRO-3 for CPT and SFT stages, and VERL with FSDP for RL training.

I.1 Hybrid Fine-tuning Strategy (FQLoRA)

We train with FQLoRA, a hybrid strategy that fully fine-tunes selected transformer layers while applying QLoRA 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 to ensure gradient flow across all depths. 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 9, 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.

I.2 Continual Pretraining and Supervised Fine-tuning

Table 9 shows hyperparameters for continual pretraining and supervised fine-tuning. All stages use a cosine learning rate schedule with early stopping based on validation loss.

I.3 Reinforcement Learning

Table 10 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 Team, 2024) to directly pass memory embeddings during inference. We use DAPO, which builds on

Table 9: Continual pretraining (CPT) and supervised fine-tuning hyperparameters (SFT).

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	256/512 ^a	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	

^a 256 for 7B with 4 GPUs, 512 for 32B with 8 GPUs.

GRPO 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. For reward modeling, we use Qwen3-32B for the 7B model and Qwen2.5-72B-Instruct for the 32B model.

Table 10: Reinforcement learning hyperparameters using DAPO.

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 to 16 ^a
Temperature	1.0
Max Prompt Length	1024
Max Response Length	600

^a 16 for 7B, 8 for 32B due to memory constraints.

J Implicit Expansion Behavior Examples

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

J.1 In-Distribution Examples

Figure 14 shows implicit expansion on a training example. The model first expands actor filmography 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 15 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 15, 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.

J.2 Out-of-Distribution Examples

Figures 16 and 17 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 14: 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; display: inline-block; padding: 2px;">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; display: inline-block; padding: 2px;">Rangers (152 wins)</p>

Figure 15: **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.

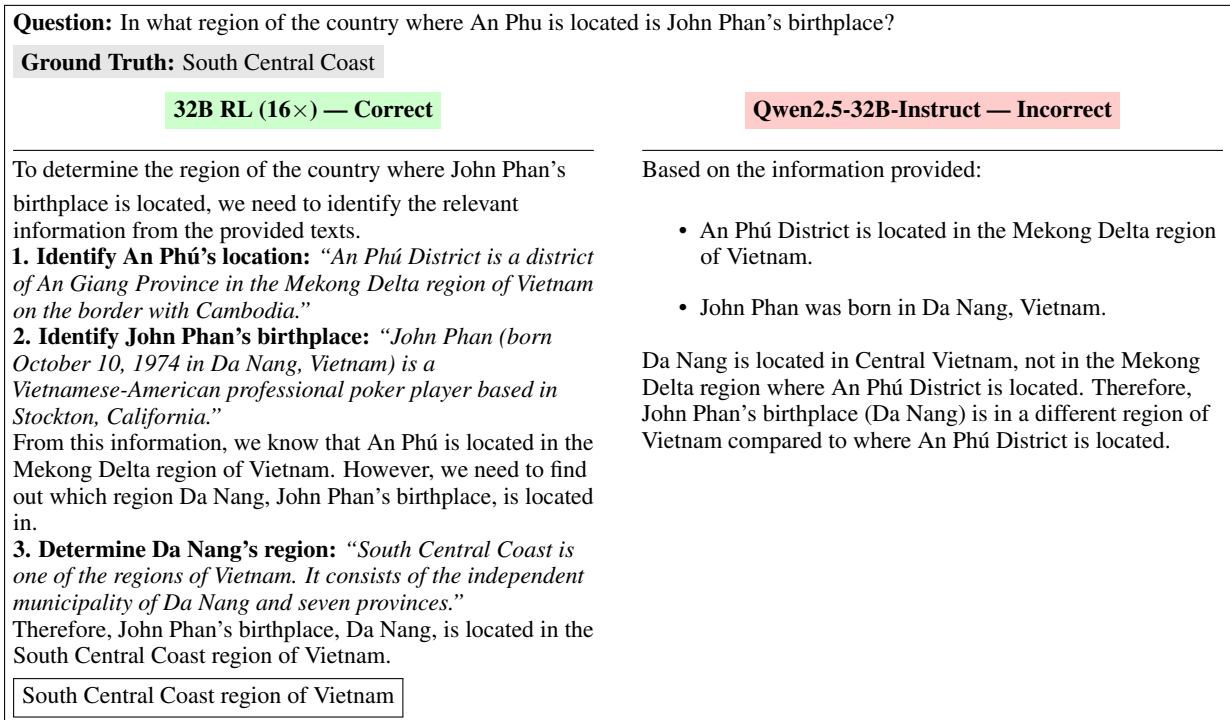


Figure 16: **MuSiQue Example 1.** The 32B RL model correctly parses the question structure, recognizing that “the country where An Phu is located” establishes Vietnam as context, while the actual query asks for the region containing John Phan’s birthplace (Da Nang). The model traces: (1) An Phu → Vietnam, (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.

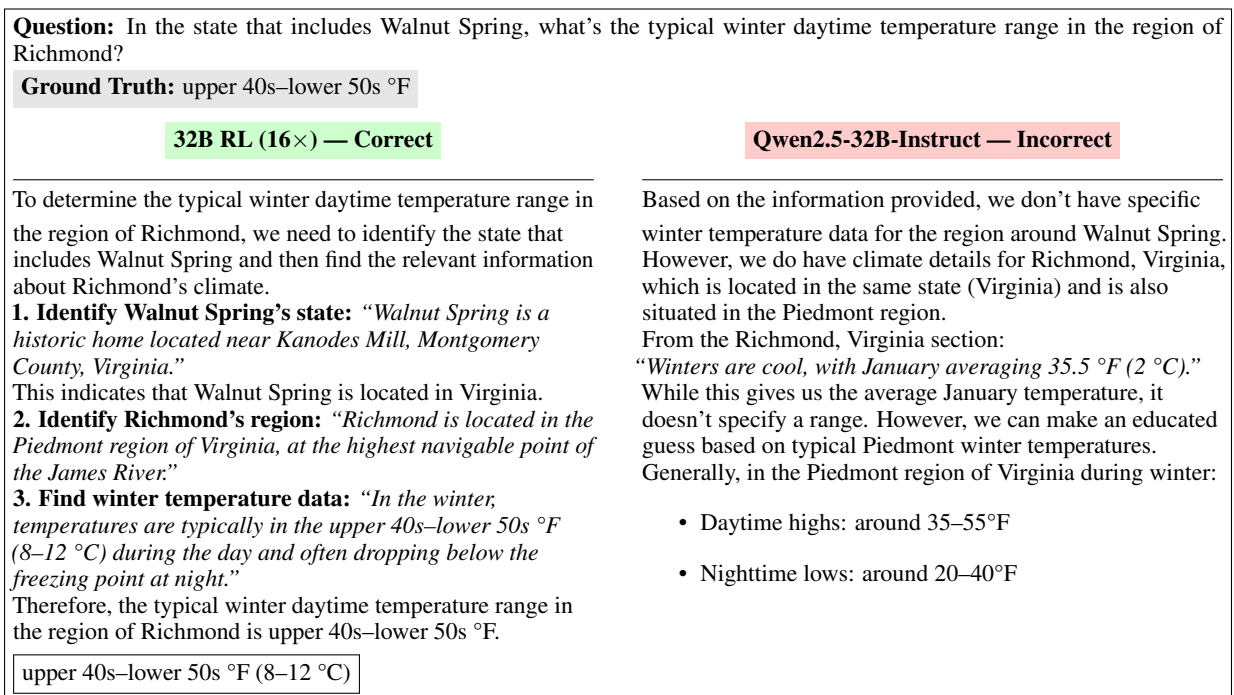


Figure 17: **MuSiQue Example 2:** The 32B RL model correctly extracts the explicit “upper 40s–lower 50s °F” daytime temperature from the compressed context. The baseline confuses mean daily temperature (35.5°F) with daytime highs, then extrapolates an incorrect range. This demonstrates the RL model’s superior ability to precisely extract and apply relevant information from compressed representations.