Rethinking LoRA Memory Through the Lens of KV Cache Compression

Chunsheng Zuo **Liaoyaqi Wang** **William Jurayj** **William Fleshman** **Benjamin Van Durme**
Johns Hopkins University
{czuo3,lwang240,wjurayj1,will.fleshman,bvandur1}@jhu.edu

Abstract

Parametric retrieval augmentation encodes document information into lightweight, document-specific modules such as LoRA adapters, reducing the need to include all evidence as input context. However, it remains unclear how this parameter-side memory interacts with context-side memory stored in the KV cache. We study this interaction in document-level question answering by progressively evicting document key-value states and measuring when a document LoRA contributes beyond the retained context. We find that document LoRA adds little when the KV cache is largely intact, but becomes increasingly useful under aggressive compression, recovering 13–21 ROUGE-L points when no document context remains. The gain is largest when the base model encodes the document, and the adapter is applied only during answer generation, suggesting that document LoRA is better understood as decoding-time parametric memory than as a document encoder. Finally, QA-style supervision produces substantially stronger adapters than raw-context next-token-prediction. These results position document LoRA as a complementary memory channel whose value emerges precisely when context-side evidence is scarce.

1 Introduction

Retrieval-augmented generation typically provides external knowledge by appending retrieved documents to the input context (Lewis et al., 2020). While this gives the model direct token-level access to evidence, each retrieved token incurs prefill computation and contributes key-value states that must be stored throughout decoding, making long or repeatedly queried documents expensive.

This has motivated two lines of work to reduce dependence on having full documents in the KV cache. The first is KV-cache compression, especially the KV-cache eviction method, which drops less important tokens while keeping the potentially more useful ones inside the KV cache to save memory while preserving most performance (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024; Chari and Van Durme, 2025). Another line of work instead explores parametric retrieval augmentation: encoding document information into lightweight document-specific parameter modules, often instantiated as LoRA adapters, which are loaded and used during inference (Hu et al., 2022; Su et al., 2025; Caccia et al., 2025; Back et al., 2026). This direction reframes external knowledge as memory carried by model parameters. These lines of work are complementary, but they are usually studied in isolation, making their interaction underexplored.

In this work, we study this interaction in document-level question answering. For each document, we train a document-specific LoRA adapter from supervision derived from the document, and evaluate it while progressively compressing the document portion of the KV cache. This protocol, illustrated in Figure 1, lets us compare two sources of document information: explicit evidence retained in the KV cache and the memory carried by the adapter. We further separate the stages at which LoRA is applied during inference: prefill, compression scoring, and decoding, and compare different ways of converting the document into adapter supervision.

Our experiments lead to three main findings:

- **Document LoRA is most useful in the high-compression regime.** Its benefit is limited when the KV cache retains sufficient evidence, but it increases as compression removes that evidence.
- **Document LoRA is more useful as a decoding-time memory than as a document-prefill module.** The base model is better suited to construct the compressed document KV cache, while document LoRA is more useful during answer generation.

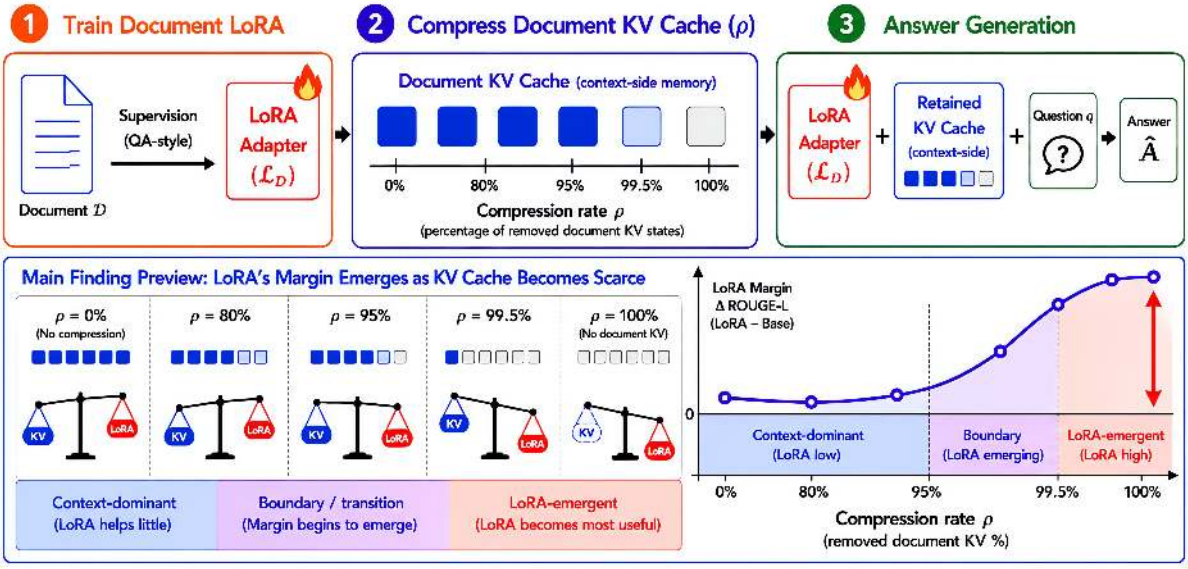

Figure 1: **Pipeline overview (Top).** We train a document-specific LoRA adapter with QA-style supervision (Step 1), compress the document KV cache by a factor ρ (Step 2), and generate the answer using the compressed cache together with the adapter (Step 3). **Main findings (Bottom).** Varying ρ traces out three regimes: at low compression the KV cache dominates and LoRA contributes little (context-dominant); under aggressive compression the LoRA margin (Δ ROUGE-L over base) begins to emerge (transition); and under extreme compression, the adapter becomes the dominant document-memory channel and yields the largest gain (LoRA-emergent).

- **QA-style supervision produces the strongest document LoRAs for QA.** It outperforms other document-derived training objectives in our current training-format comparison.

2 Related Work

2.1 KV Cache Compression

The KV cache grows linearly with sequence length and batch size, often surpassing the model parameters in memory footprint and becoming the dominant cost in long-context serving (Liu et al., 2024a). KV-cache compression methods include *eviction*, which keeps only KV entries selected by attention scores (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024), positional heuristics (Xiao et al., 2024), or learned budgets (Yang et al., 2024a; Feng et al., 2025); *merging*, which consolidates less important entries to mitigate eviction loss (Zhang et al., 2024; Wang et al., 2024; Wan et al., 2025); and *quantization*, which reduces the bit-width of stored states (Liu et al., 2024c; Hooper et al., 2024; Yang et al., 2024b). These categories are not mutually exclusive: eviction and merging change which states remain or how they are combined, while quantization changes the precision of retained states. In this work, we use KV-cache compression as a controlled way to vary how much document evidence remains available in context.

2.2 Parameter-Side Document Memory

Another line of work reduces reliance on long retrieved contexts by moving document information into trainable parameter modules, often implemented as LoRA adapters. These methods inject retrieved knowledge as low-rank weight updates rather than prompt tokens to avoid context bloat (Liu et al., 2024a) and knowledge conflicts (Xu et al., 2024). Per-document adapters can be obtained offline via QA-pair fine-tuning (Su et al., 2025) or context distillation (Caccia et al., 2025), or generated on the fly by a hypernetwork mapping document embeddings to LoRA weights (Tan et al., 2025). When a library of adapters already exists, LAG (Fleshman and Van Durme, 2025) routes among them per token and per layer without further training.

Closer to our setting, empirical analyses (Back et al., 2026; Zhao et al., 2025) characterize LoRA's capacity as parametric memory and show that parametric and in-context injection are largely complementary, while recent adapter-serving work studies delayed or selective LoRA activation to reuse base-model KV caches (Greenewald et al., 2025; Li et al.,

2025). These studies, however, either study LoRA under full-context or no-context settings or treat stage-wise LoRA activation as an efficiency mechanism. We instead use KV-cache compression as a controlled axis that varies how much context-side document memory is retained, and use inference-stage controls as a diagnostic to ask where along this axis document LoRA contributes and which stages should apply it.

3 Methodology

We study document-specialized LoRA as a form of parameter-side document memory under various KV cache compression settings. For each document, we train a separate LoRA using supervision derived from that document, then evaluate it on questions over the same document under different amounts of retained document KV cache.

This section describes the training procedure (subsection 3.1), the KV-cache compression protocol (subsection 3.2), the evaluation protocol (subsection 3.3), and the inference variants used to separate LoRA's roles across stages (subsection 3.4) and supervision formats (subsection 3.5).

3.1 Document-Specialized LoRA Training

To internalize a document into parametric memory, we follow a similar procedure as prior works (Su et al., 2025; Caccia et al., 2025; Back et al., 2026). For each document d, we split the text into chunks $\mathcal{C}_d = \{c_i\}_{i=1}^{n_d}$ and derive supervised training examples from those chunks. A supervision format f maps the chunks to a training set $\mathcal{D}_d^{(f)}$. Each example $(x, m) \in \mathcal{D}_d^{(f)}$ contains a token sequence x and a loss mask sequence $m \in \{0,1\}^{|x|}$ that has the same length as x to indicate which token the loss will be calculated on. Let θ_0 be the frozen base model. We train the document LoRA parameters $\Delta\theta_d^{(f)}$ by

$$\arg\min_{\Delta\theta} \sum_{(x,m)\in\mathcal{D}_d^{(f)}} \sum_{t=1}^{|x|} -m_t \log p_{\theta_0+\Delta\theta}(x_t \mid x_{<t}). \tag{1}$$

Only the LoRA parameters are updated. We refer to the resulting adapter as the document LoRA L_d. At evaluation time, questions from document d are answered with its document LoRA L_d.

3.2 KV-Cache Compression

Our compression method is based on Compactor (Chari and Van Durme, 2025), a recent KV eviction algorithm that is training-free and robust (see verification in subsection 6.1). We use it to compress the document portion of the KV cache at a fixed rate. For a given layer and attention head, let the document key-value states be $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N\times d}$, where N is the number of document tokens. Compactor assigns each token an importance score by combining an attention-based score a_i with an approximate leverage score $\widetilde{\ell}_i$:

$$s_i = \frac{a_i - \bar{a}}{\mathrm{std}(\mathbf{a})} + \lambda \frac{\widetilde{\ell}_i - \bar{\widetilde{\ell}}}{\mathrm{std}(\widetilde{\boldsymbol{\ell}})}. \tag{2}$$

Here a_i is computed from non-causal attention mass, while $\widetilde{\ell}_i$ approximates the statistical leverage of the token's key vector. Let ρ denote the compression rate, defined as the fraction of document key-value states removed, and let $r = 1 - \rho$ be the retention rate. We keep the indices

$$\mathcal{I}_\rho = \mathrm{TopK}\left(\mathbf{s}, \lceil rN \rceil\right), \tag{3}$$

and form the compressed document cache by retaining $\mathbf{K}_{\mathcal{I}_\rho}$ and $\mathbf{V}_{\mathcal{I}_\rho}$. In our experiemnts, we evaluate $\rho \in \{0, 80\%, 90\%, 95\%, 99\%, 99.5\%, 100\%\}$, where $\rho = 0$ keeps the full document KV cache, while $\rho = 100\%$ drops the document entirely. The extreme compression ratios are chosen because we found they can be reflective of LoRA's effect. Implementation details can be found in Appendix C.

3.3 Evaluation protocol

At evaluation time, each document comes with several questions. For each document, we first encode the user's conversational prefix followed by the document into the KV cache. For KV cache eviction, we only do it over the document portion of the KV cache, keeping the user's instruction prefix unchanged. After compression, we encode the question with the model and the compressed cache, then continue generating the answer by greedy decoding. Appendix A.1 details the prompt templates.

3.4 Inference-Stage Controls

Recent adapter-serving work uses selective LoRA activation to reuse base-model KV caches and reduce inference cost (Greenewald et al., 2025; Li et al., 2025). We repurpose stage-specific LoRA activation as a diagnostic: does document LoRA help more when constructing the document KV cache, or when decoding from a compressed one? To answer this, we separate inference into three stages (document prefill, compression scoring, and

answer decoding) and define four modes that vary LoRA usage across them. Let B denote running the frozen base model without the adapter, and let L_d denote running the same base model with the LoRA of document d enabled. A cell in the table indicates whether that stage uses the base-only computation or the document-LoRA-augmented computation. The four modes specify how LoRA usage is assigned across the 3 inference stages.

Mode	Prefill/KV	Score	Decode
Base	B	B	B
Adapter score + adapter prefill	L_d	L_d	L_d
Base KV + adapter decode	B	B	L_d
Base score + adapter prefill	L_d	B	L_d

Base. The base model performs document prefill, compression scoring, and answer decoding without any adapter.

Adapter score + adapter prefill (default). The document LoRA is active throughout inference: it produces the document KV states, provides the compression scores, and decodes the answer.

Base KV + adapter decode. The base model performs document prefill and compression scoring, producing the compressed document KV cache. The document LoRA is then loaded only for answer decoding.

Base score + adapter prefill. The document LoRA produces the document KV states and decodes the answer, while compression scores are computed without the adapter. The retained indices from base scoring are then applied to the adapter-produced document KV states.

3.5 Training-Format Controls

We compare four supervision formats derived from each document while keeping the document-specific training protocol fixed. Appendix A.1 gives the exact prompt and loss-mask format for QA and raw-context supervision.

Synthetic QA (default). We prompt the model to generate QA pairs from document chunks. Each training sequence contains the generated question and answer, and the loss is applied only to answer tokens. This is the default format used in the main experiments unless otherwise stated. More details for synthesis are in subsection A.2

Raw Context. For each chunk, the training sequence is the document itself, and the loss is applied to all tokens of the document.

Context Continuation. For adjacent chunks, we use the first chunk as context and the following chunk as the prediction target. The loss is applied only to the target chunk.

Context + QA. This format includes the source context together with the generated question, while still applying the loss to the answer.

4 Experiment Setup

Datasets. We use NarrativeQA (Kočiský et al., 2018) and LongHealth (Adams et al., 2024) as the main document-level QA testbeds. For NarrativeQA, each document corresponds to one story. For LongHealth, we treat one patient or case record as one document, so that the same per-document adapter protocol applies. Our evaluation subsets contain 20 documents each, with 596 QA examples for NarrativeQA and 400 QA examples for LongHealth. Using the Llama-3.1-8B-Instruct tokenizer, documents average 16K tokens for NarrativeQA and 11K tokens for LongHealth. Additional dataset statistics are given in Appendix B.

Models. We use `Qwen3-4B` (Yang et al., 2025) and `Llama-3.1-8B-Instruct` (Grattafiori et al., 2024) as base models, both of which are instruction-finetuned models that can follow a user-assitant conversational format for question answering. These models also generate the synthetic data to train their own LoRA's.

Default LoRA recipe. Unless otherwise specified, we use document chunks of word length 64 and train LoRAs with DoRA enabled (Liu et al., 2024b), $r = 16$, $\alpha = 32$, and dropout rate $= 0.1$ on the MLP projection modules. Adapters are trained for 4 epochs with learning rate 1×10^{-4}, batch size 4, and weight decay 0.001, using Synthetic QA supervision (§3.5). All trainings are conducted on 1 H100 GPU with 96GB of memory. Additional implementation details are in Appendix A.

Metrics. We report ROUGE-L as the main QA metric. Our QA setting uses generated phrase-level answers rather than fixed labels, so accuracy is not directly applicable, and exact match is too brittle to small wording or formatting differences. ROUGE-L gives a graded measure of answer overlap, which is better suited for comparing compression and adapter conditions.

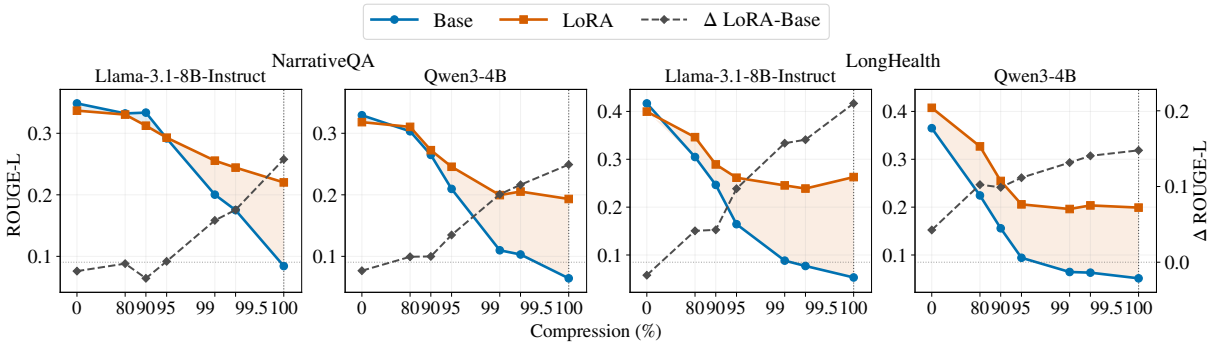

Figure 2: **Document LoRA complements aggressive KV-cache compression.** We compare QA performance with only the compressed document KV cache against performance with the corresponding document LoRA. Across **(a) NarrativeQA** and **(b) LongHealth**, the LoRA margin is small when much of the document KV cache remains, but grows under aggressive compression and at the no-context endpoint.

5 Main Findings

We treat document LoRA and the compressed KV cache as two complementary sources of document memory. In this section, we ask how they interact across compression regimes and organize the results around three diagnostic questions:

- when does document LoRA become useful as document KV states are removed (subsection 5.1);
- where should the adapter be active during inference: document prefill, compression scoring, or answer decoding (subsection 5.2);
- what supervision format makes adapter-side memory useful for QA (subsection 5.3).

5.1 Document LoRA complements aggressive KV-cache compression

LoRA adds little at low compression Figure 2 shows when document LoRA starts to matter. Across NarrativeQA and LongHealth, the same pattern emerges. When much of the document KV cache remains, the base model and the LoRA-augmented model remain relatively close because explicit document evidence is still available. On NarrativeQA, the LoRA margin is near zero or negative through most low-compression settings. LongHealth shows positive margins earlier, but these remain below the high-compression and no-context gains.

LoRA becomes the primary document memory under extreme compression. The contrast becomes clearest at the high-compression end of the curve. At the no-context endpoint, the LoRA–base gain is roughly 13 ROUGE-L points on NarrativeQA and about 15–21 points on LongHealth. The margin curves, therefore, show that the adapter's contribution is not constant; it becomes visible precisely when context-side document memory is removed.

These results suggest a division of labor between the two memory sources. The KV cache remains the stronger carrier of document information when it is available, while document LoRA acts as a complementary parametric memory that partially recovers information lost under extreme compression. At 99% compression, where only about 160 NarrativeQA tokens or 111 LongHealth tokens remain (Table 3), the adapter already recovers roughly half of its full no-context margin, suggesting that the transition from context-dominant to LoRA-emergent behavior occurs around this point.

Implication: Document LoRA should not be understood as a general replacement for long-context encoding. Instead, it is most useful as a memory augmentation mechanism under severe KV-cache budget constraints.

5.2 QA-trained LoRA Helps More During Generation Than Document Encoding

We next ask where the adapter should be active once the document KV cache is compressed. Since document LoRA can affect the document KV states, the compression scores used to select retained states, and the final answer decoding, Figure 3 compares four inference modes that separate these roles.

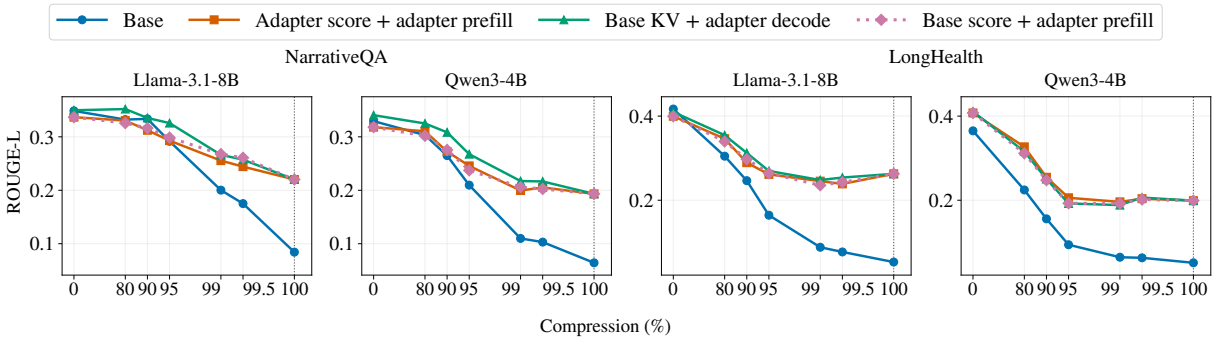

Figure 3: **Inference-stage controls.** We compare four ways of applying document LoRA across document prefill, compression scoring, and answer decoding on NarrativeQA and LongHealth. Base uses no adapter; Adapter score + adapter prefill enables LoRA throughout inference; Base KV + adapter decode uses the base model for document prefill and compression scoring, then enables LoRA for decoding; Base score + adapter prefill keeps LoRA-produced KV states but uses base-model compression scores.

Base-prefill with adapter decoding is strongest or competitive. At all compression points where some document KV remains ($\rho < 100\%$), using the base model to prefill and compress the document KV cache and then enabling the document LoRA for decoding is usually the strongest or competitive mode. The pattern is clearest on NarrativeQA, where Base KV + adapter decode gives the best high-compression average for both models. It also holds for LongHealth with Llama-3.1-8B, while LongHealth with Qwen3-4B is closer to a tie and sometimes slightly favors using the adapter throughout inference. At the most severe non-empty compression points (99–99.5%), Base KV + adapter decode improves over base-only inference by about 7–11 ROUGE-L points on NarrativeQA and 13–17 points on LongHealth. The question is therefore not whether the adapter helps, but where it should be applied: the strongest recipe is usually to let the base model build the document KV cache and reserve the adapter for generation.

The score-control condition does not make scoring the main explanation. The Base score + adapter prefill condition tests whether adapter-prefill is weaker simply because adapter-based compression scores select worse tokens. Changing only the scoring path has a small and inconsistent effect: it sometimes improves adapter-prefill, but it does not consistently match the base-prefill trajectory. This suggests that the central difference is not only which tokens are retained, but also which model produces the retained KV states. Base-produced document KV states provide stronger context-side memory, while the document LoRA is more useful as decoding-time parameter-side memory.

Implication: The stronger recipe is to use the base model for document prefill and apply the document LoRA primarily during answer generation. In this setting, LoRA is better understood as generation-time memory support rather than as a replacement document encoder.

5.3 QA-Style Supervision Produces the Strongest Document LoRAs

We next vary the supervision used to train document LoRA while keeping the evaluation task fixed to document QA. Figure 4 compares QA-only supervision with three alternatives: raw next-token prediction on document text, chunk-to-next-chunk continuation, and context-plus-QA training where the source context is included in the training prompt. This comparison asks whether useful adapter-side memory comes from document exposure alone, from continuation-style training, or specifically from answer-supervised QA training.

QA-only supervision is strongest when context evidence is scarce. Across both datasets and both backbones, QA-only supervision produces the strongest adapter in the high-compression and no-context regimes. At the no-context endpoint, QA-only reaches about 0.19–0.26 ROUGE-L, while context-plus-QA is around 0.11–0.13, chunk continuation around 0.08–0.13, and raw next-token prediction remains close to the base no-context level. The separation is largest when the model can no longer rely on retained document KV states, showing that the training format matters most when the adapter has to provide the missing document signal.

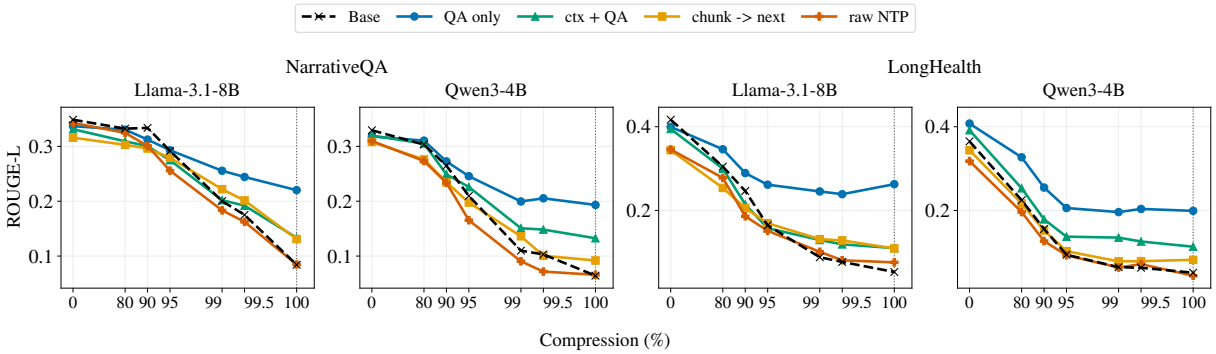

Figure 4: Training-format comparison on NarrativeQA and LongHealth with Llama-3.1-8B and Qwen3-4B. Each document LoRA is trained from the same documents using a different supervision format, then evaluated under KV-cache compression. QA-only supervision is strongest in the high-compression and no-context regimes across both datasets, while raw next-token prediction remains close to the base no-context behavior.

Document exposure and QA formatting are not sufficient. Raw next-token prediction directly exposes the adapter to document text, but it does not produce strong QA behavior under compression. Chunk continuation is sometimes better than raw NTP at the most compressed points, but remains well below QA-only supervision. The context-plus-QA control is especially informative: it uses a QA interface, but because the source context is present during training, the model can answer from the prompt rather than pressure the adapter to store answer-bearing information. Thus, the advantage of QA-only supervision is not just document fine-tuning or QA-like formatting; it better matches the way the adapter must be used at evaluation time, where it needs to recall and express document information in response to questions.

Implication: A useful document LoRA cannot be obtained by simply reciting the document text. The adapter needs to learn to map the question to the answer, such that important facts can be recalled when context is absent during question answering.

6 Ablation Studies

In this section, we consider two design choices that directly shape the interaction between compressed context and document LoRA: **compression algorithm** and **LoRA target module group**. The compression algorithm determines which document KV states remain available, while the LoRA target module group determines where document-specific information can be stored in the model. We therefore ablate both choices.

6.1 Compression Method Robustness

Different KV-cache eviction rules can retain very different document tokens, so the compression-regime effect we observe with Compactor could in principle hinge on its specific scoring rule rather than on KV-cache compression in general. Figure 5 tests this by replacing Compactor with Expected Attention (Devoto et al., 2025), which ranks document KV entries by their average attention weight. The two compressors produce different absolute degradation curves: at 10% retention on LongHealth/Llama, for instance, Expected Attention drops base ROUGE-L to 0.157 versus Compactor's 0.246. Crucially, however, the gap between LoRA and base follows the same pattern under both compressors—small margins when much of the document cache remains, and clear separation once compression becomes aggressive. This consistency across eviction rules supports the view that document LoRA compensates for missing document evidence rather than interacting with any particular compressor's selection bias.

6.2 Target-module ablation

Prior interpretability and model-editing work suggests that transformer feed-forward modules are natural sites for storing and modifying semantic or factual associations: FFN layers can behave like key-value memories, and MLP weights are effective targets for factual edits (Geva et al., 2021; Dai et al., 2022). This motivates our default use of MLP-targeted document LoRA, but it does not by itself establish that MLP targeting is optimal for our setting. We therefore vary the LoRA target modules while keeping the training signal and evaluation protocol fixed.

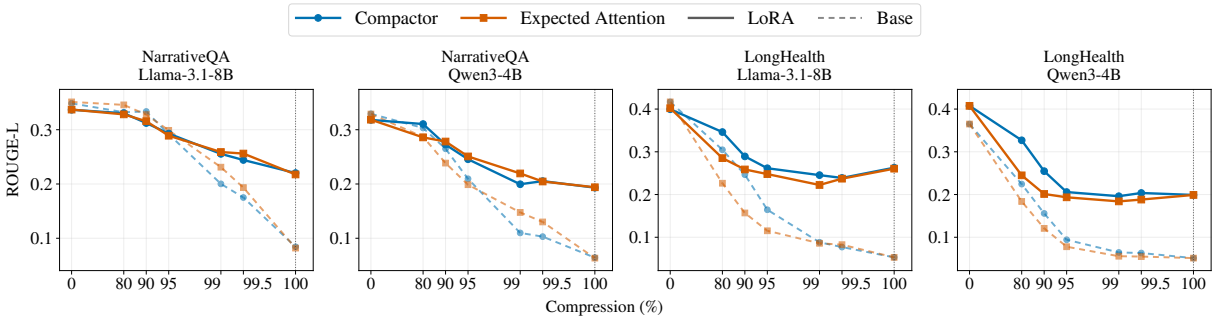

Figure 5: Compression method ablation. Each panel plots ROUGE-L against compression ratio for a given model, comparing base (no adapter) and document LoRA runs under two compression algorithms: Compactor and Expected Attention. **The margin between LoRA and base follows the same pattern for both algorithms: negligible at low compression ratios, and widening as the compression ratio increases.**

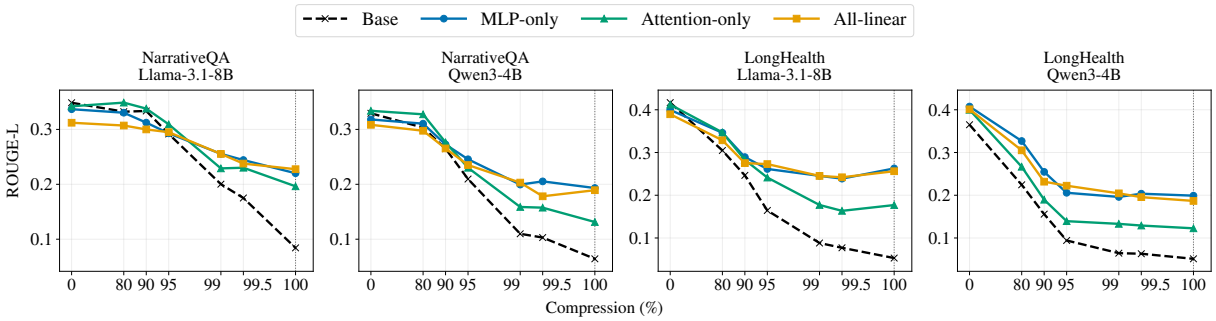

Figure 6: Target module ablation. Each panel plots ROUGE-L against compression ratio for three LoRA target configurations: attention-only, MLP-only, and all-linear, alongside the base model without any adapter. **MLP-targeted adapters are strongest or close to strongest at high compression ratios and in the no-context setting, while all-linear adaptation does not consistently improve over MLP-only.**

Figure 6 varies the LoRA target modules across MLP-only, all-linear, and attention-only while keeping the training signal and evaluation protocol fixed. The most informative comparison is between MLP-only and all-linear, since all-linear adapts the same MLP modules plus the attention projections. Surprisingly, MLP-only outperforms all-linear in most settings, even though all-linear adapts a broader set of modules and has larger capacity. For example, in the no-context setting on NarrativeQA, MLP-only reaches 0.207 ROUGE-L versus 0.208 for all-linear; on LongHealth the gap is similarly small (0.231 vs 0.221).

Attention-only adapters are consistently the weakest, trailing MLP-only by 0.02 to 0.09 ROUGE-L across both datasets. This is consistent with the view that document LoRA serves primarily as factual memory stored in the MLP rather than as a modified attention routing mechanism. Since all-linear adaptation adds attention modules on top of MLP without consistent gains, these results support the MLP-only recipe used in the main experiments.

7 Conclusion

We investigate how document-specialized LoRA adapters interact with KV-cache compression as two complementary sources of document memory for question answering. Through a controlled evaluation protocol that progressively evicts document KV states across two datasets and two models, we find three consistent patterns: (1) document LoRA is more helpful at intense KV compression; (2) the base model produces stronger document KV representations, while the adapter is more effective when applied only during answer generation; and (3) QA-style supervision produces substantially stronger adapters than raw-context, continuation, or context-plus-QA objectives. These findings imply that document LoRA should not be viewed as a replacement for long-context encoding, but rather as a complementary parametric memory channel whose value emerges precisely when context-side evidence becomes scarce, suggesting a practical recipe of base-model prefill combined with adapter-augmented decoding under tight KV-cache budgets.

8 Limitations

Our goal is diagnostic rather than to propose a complete LoRA-augmented serving system. We assume that the correct document adapter is available at inference time, whereas a deployed system would also need an adapter retrieval or routing layer, a policy for composing evidence from multiple documents, and mechanisms for managing adapter storage and loading latency. These requirements do not affect the central comparison in this paper, because all conditions are evaluated with the same document and differ only in whether document information is carried by retained KV states, by the document adapter, or by both. Rather, our results identify the regime in which such systems are most likely to be useful: repeated queries over the same document, private or fixed document collections, or settings where KV-cache budgets force severe context compression. In one-off query settings with ample context budget, standard long-context or retrieval-augmented inference may remain preferable.

References

Lisa Adams, Felix Busch, Tianyu Han, Jean-Baptiste Excoffier, Matthieu Ortala, Alexander Löser, Hugo JWL. Aerts, Jakob Nikolas Kather, Daniel Truhn, and Keno Bressem. 2024. Longhealth: A question answering benchmark with long clinical documents. *arXiv preprint arXiv:2401.14490*.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. 2026. Understanding LoRA as knowledge memory: An empirical analysis. *arXiv preprint arXiv:2603.01097*.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. 2025. Training plug-and-play knowledge modules with deep context distillation. In *Second Conference on Language Modeling*.

Vivek Chari and Benjamin Van Durme. 2025. Compactor: Calibrated query-agnostic kv cache compression with approximate leverage scores. *arXiv preprint arXiv:2507.08143*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. 2025. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

William Fleshman and Benjamin Van Durme. 2025. Lora-augmented generation (lag) for knowledge-intensive language tasks. *arXiv preprint arXiv:2507.05346*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Kristjan Greenewald, Luis A. Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Daniel Cox. 2025. Activated loRA: Fine-tuned LLMs for intrinsics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In *Advances in Neural Information Processing Systems*, volume 37, pages 1270–1303. Curran Associates, Inc.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Allison Li, Kristjan Greenewald, Thomas Parnell, and Navid Azizan. 2025. Efficient multi-adapter llm serving via cross-model kv-cache reuse with activated lora. *arXiv preprint arXiv:2512.17910*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024c. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32332–32344. PMLR.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. 2025. Parametric retrieval augmented generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 1240–1250, New York, NY, USA. Association for Computing Machinery.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and Mi Zhang. 2025. D2o: Dynamic discriminative operations for efficient long-context inference of large language models. In *The Thirteenth International Conference on Learning Representations*.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*.

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

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

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. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024a. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3258–3270, Bangkok, Thailand. Association for Computational Linguistics.

June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024b. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. 2024. CaM: Cache merging for memory-efficient LLMs inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 58840–58850. PMLR.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Jun Zhao, Yongzhuo Yang, Xiang Hu, Jingqi Tong, Yi Lu, Wei Wu, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Understanding parametric and contextual knowledge reconciliation within large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

A Method Details

A.1 QA Prompt Formats

Training. The default QA training example is a chat-formatted question-answer pair. It does not include the original document chunk. The user message contains the generated question and asks the model to place the answer in boxed notation:

```
<User_token>:
Question: {question}
Answer the question by putting the answer
    inside the brackets of the "\boxed{}"
    notation.

<Assistant_token>:
The answer is \boxed{{answer}}.
```

During training, the question and prompt prefix are masked, and loss is applied only to the answer continuation inside the boxed answer:

```
>>> loss_mask_start
Question: {question}
Answer the question by putting the answer
    inside the brackets of the "\boxed{}"
    notation.
The answer is \boxed{
>>> loss_mask_end
{answer}}.
```

Evaluation. Evaluation uses the same question-answer template and boxed-answer convention, but includes the document context before the question. The full evaluation prompt has the following form:

```
<User_token>:
Here is a context:

{document context}

Question: {question}
Answer the question by putting the answer
    inside
the brackets of the "\boxed{}" notation.

<Assistant_token>:
The answer is \boxed{
```

The model begins generation after the fixed answer prefix "`The answer is \boxed`", so the generated continuation should complete the boxed answer:

```
{answer}}.
```

For context-prefill evaluation, this same prompt is split into a document prefix and a question suffix. The document prefix contains the chat user prefix and document context, and the question suffix contains the question, answer instruction, assistant prefix, and fixed boxed-answer prefix. In the no-context setting, the document context is omitted, and the model receives only the question and fixed answer prefix.

Raw-context supervision. Raw-context training uses the document chunk itself as the training sequence, without a template:

```
{raw document chunk text}
```

A.2 Training Supervision Construction

Synthetic QA data are generated from document chunks before adapter training. In the main QA setting, each document is split into target chunks with a default target length of 64 words and no overlap. For each target chunk, we use fixed dispatch and request four short-answer QA pairs (`pairs_per_call=4`) with one generation call per chunk-task request. The synthesis model is the same model for training the adapter: Qwen3-4B generates the synthetic QA data for Qwen adapters, and Llama-3.1-8B-Instruct generates the synthetic QA data for Llama adapters. During synthesis, neighboring chunks may be supplied as background context, but the prompt instructs the generator to draw questions and answers only from the target passage. Generated outputs are parsed as JSON arrays with `question` and `answer` fields; outputs with more than the requested number of items are truncated to the requested count, and failed chunk-task requests are retried.

```
Below is a target passage, preceded by
    surrounding context for
background understanding only.

[Background context]
{surrounding_context}

[Target passage]
{chunk_text}

Generate {pairs_per_call} items whose
    questions and answers are drawn ONLY from
    facts stated in the [Target passage] above
    . Each answer must be a short factual
    phrase (one word to a few words), for
    example: a person's name, a number, a date
    , a location, or another specific fact.
    Prefer the exact wording used in the
    passage when possible. Return a JSON array
     where each element is an object with keys
     "question" and "answer". Do NOT include
    any explanation or text outside the JSON
    array.
```

When no surrounding context is used, the generator receives only the passage and the same JSON-format instruction. The background context is used

only to help generation; in the default SYNTHETIC QA training format, the original document chunk is not included in the training input.

Synthetic–evaluation QA overlap We also characterize overlap between synthetic training questions and evaluation questions within the same document or patient record. We compare questions under three increasingly permissive criteria: raw exact match, near-duplicate match, and a broader lexical-similarity match. Exact matches are extremely rare, ranging from 0 to 6 questions across the four source/model settings. Near-duplicate matches are also rare, with at most 16 matched synthetic questions. Even under the broader lexical-similarity criterion, only 13–88 synthetic questions are matched, corresponding to 0.18%–0.59% of generated synthetic questions. We do not filter these pairs in the main experiments because the synthetic QA data are generated from document chunks rather than from evaluation annotations; the statistic is reported to make the degree of overlap transparent.

B Dataset Statistics

Table 2 summarizes the document lengths used in our main NarrativeQA and LongHealth experiments. We count tokens with the Llama-3.1-8B-Instruct tokenizer, without adding special tokens, after deduplicating repeated QA rows by document identifier.

For NarrativeQA, we construct a 20-document subset from the official NarrativeQA documents and question-answer annotations. We use full story text rather than summaries or retrieval snippets. Candidate stories are randomly shuffled with a fixed seed, filtered to keep medium-length full-document examples, and the first 20 eligible documents are selected. The length filter only determines document eligibility; selected stories are not truncated. All available QA pairs for the selected documents are kept, yielding 596 evaluation examples. For LongHealth, each patient or case record is treated as one document, giving 20 documents and 400 evaluation examples.

Table 3 gives an approximate interpretation of the compression ratios used in our experiments. The values are computed by multiplying the average document length in each dataset by the fraction of context tokens retained after compression. For example, 99.5% compression keeps only 0.5% of the document context, corresponding to roughly 80 NarrativeQA tokens or 55 LongHealth tokens on average. These numbers are approximate because the implementation also preserves non-document prompt/template tokens, but they show the scale of the context bottleneck imposed by high compression.

C KV-Cache Compression Implementation

We implement KV-cache compression using `kvpress` (Devoto et al., 2025) CompactorPress, wrapped with local bookkeeping to support document-context-only compression. In the main generative QA setting, we use a `context_prefill` path: the prompt is split into a document prefill prefix and a question suffix. The model first runs a forward pass over the chat/template prefix and document context, producing a KV cache. Compression is applied during this prefill pass by forward hooks on the model's self-attention layers. The question and answer-formatting suffix is then evaluated against the compressed context cache using `past_key_values`. Thus, compression changes the document KV states available to the model, while the question tokens and answer prefix are not themselves compressed.

For each compressed layer, the press scores KV positions and keeps the top-scoring positions according to the requested retention rate. Our default scorer is Compactor, which combines a non-causal attention score with a leverage-score estimate of each key vector's statistical importance. The implementation applies this scoring only during prompt/context prefill; decode steps are skipped, so generated answer tokens are appended to the already-compressed cache rather than triggering new compression decisions. We use `chunk_size=256` for the non-causal attention scoring used by Compactor.

The compression ratio ρ is interpreted as the fraction of document-context KV states removed. In the `context_prefill` implementation, non-document prompt tokens before the document context are protected, and the requested compression rate is converted into an effective full-prefill rate so that approximately $(1-\rho)$ of the document context remains. For example, $\rho = 99.5\%$ keeps roughly 0.5% of the document context tokens, plus a small number of protected prompt/template tokens. The no-compression condition is represented separately and leaves the full document context cache intact.

Format	Source examples	Training input	Loss tokens
Synthetic QA	Generated QA pairs	Question + boxed answer	Answer only
Raw Context	Document text	Raw full document context	All context tokens
Context Continuation	Adjacent chunks	Current chunk → next chunk	Next chunk only
Context + QA	Generated QA pairs with context	Context + question + boxed answer	Answer only

Table 1: Training supervision formats used in the format ablation.

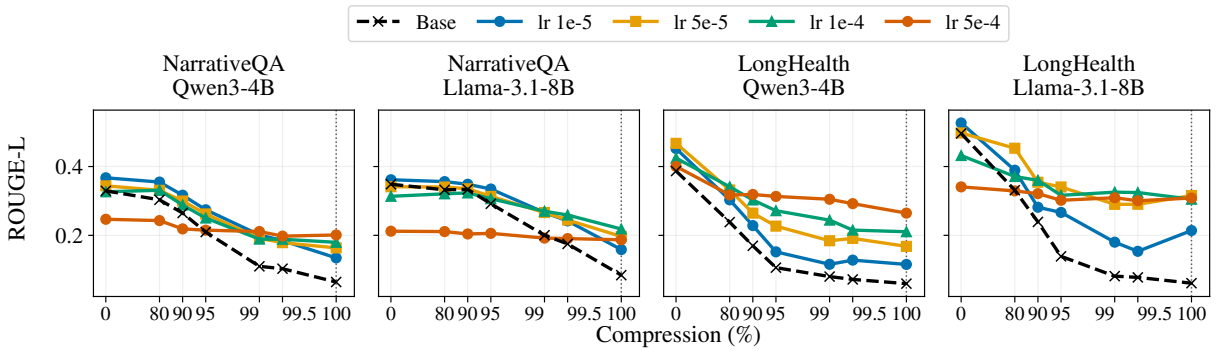

Figure 7: Controlled learning-rate ablation for compressed-generation performance. Each panel plots mean ROUGE-L across the same compression grid used in the main experiments, ending at the 100% no-context point. Learning-rate choices can change robustness under severe compression even when full-context performance is similar.

Dataset	Questions	Docs	Mean	Min	Max
NarrativeQA	596	20	16,022	11,436	22,974
LongHealth	400	20	11,081	9,418	12,332

Table 2: Document-level context length statistics for the main evaluation subsets, measured with the Llama-3.1-8B-Instruct tokenizer.

Dataset	80%	90%	95%	99%	99.5%
NarrativeQA	3,204	1,602	801	160	80
LongHealth	2,216	1,108	554	111	55

Table 3: Approximate remaining document-context tokens after KV-cache compression, computed from the average document lengths in Table 2.

D Additional Training Hyperparameter Controls

D.1 Learning-Rate Search

Figure 7 shows the learning-rate sweep used to choose the default adapter training recipe. Across the sweep, 1×10^{-4} provides the most stable trade-off between full-context performance and high-compression/no-context behavior, so we use it as the default learning rate in the main experiments. This search also shows why tuning only on the full-context point can be misleading: the learning rate that preserves the best uncompressed score is not always the one that gives the most useful adapter behavior when context-side document evidence is heavily compressed or removed.

D.2 Chunk-Size Search

Figure 8 shows the chunk-size sweep used to choose the default document chunking recipe. We compare chunk sizes 32, 64, and 128 under the same compression grid as the main experiments. On NarrativeQA, chunk sizes 64 and 128 behave similarly in the high-compression region, but chunk 64 is slightly more favorable at the no-context endpoint and avoids moving to a longer training context without clear gain. On LongHealth, the larger chunk-128 setting is noticeably weaker under severe compression, while chunk sizes 32 and 64 are much closer to each other. Since chunk 64 is competitive with chunk 32 on LongHealth and more favorable on NarrativeQA, we use chunk size 64 as the default in the main experiments. Overall, the sweep suggests that moderately sized local document chunks are sufficient for strong compressed-generation behavior, and simply increasing the chunk length does not reliably improve the adapter.

E AI Use Disclosure

We used Claude (Anthropic) and ChatGPT (OpenAI) for writing assistance and editing suggestions. All scientific content, experimental design, and analysis were conducted by the authors.

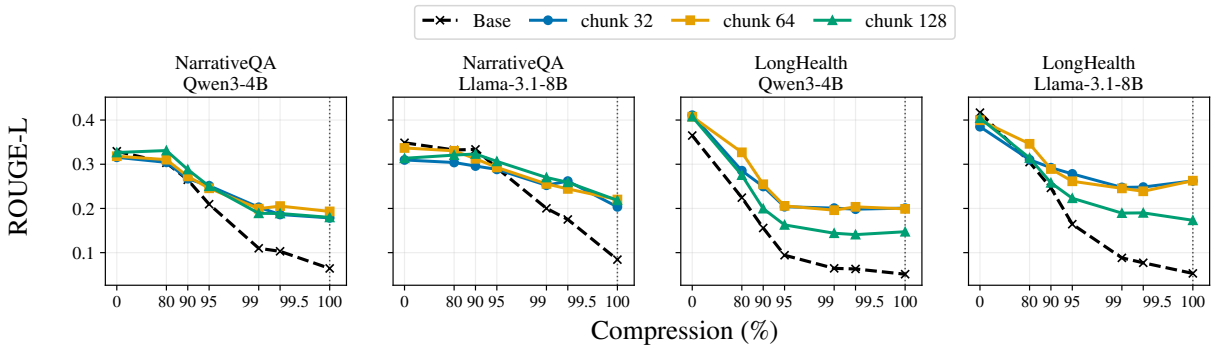

Figure 8: Chunk-size ablation on NarrativeQA and LongHealth. Moderate chunk sizes provide the most stable high-compression behavior: chunk 64 is strongest at the no-context endpoint on NarrativeQA, while chunk 32 and chunk 64 are generally more reliable than the larger-chunk reference on LongHealth.