COMBINING DOCUMENT LORA AND KV CACHE FOR LONG-CONTEXT QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Document LoRA encodes information from a source document through low-rank adaptation (LoRA), which trains small weight-update matrices while keeping the base model fixed. These document-specific parameters provide a form of memory alongside the key-value (KV) cache. We observe that the KV cache and document LoRA differ in the types of questions they answer correctly, motivating their combination for long-context question answering. Our approach first applies KV cache eviction, then fine-tunes the model on synthetic question-answer pairs from the source document using LoRA. Training updates only the low-rank matrices in the feed-forward projections while keeping the compressed KV cache in context. When answering a question, the model uses both the document LoRA and the compressed KV cache. On 2WikiMultihopQA, answering with both memories outperforms using the selected KV cache alone or the same learned adapter alone across all four models. The overall F1 gains reach 25.0 and 9.0 points, respectively. We expect this approach to improve efficiency for repeated QA over a shared document and, with chunked processing, for documents that exceed the model's context window.

1 INTRODUCTION

Long-context question answering (QA) requires language models to find and combine information across long documents (Bai et al., 2024). The key-value (KV) cache stores the document representations that the model attends to while answering. Its size grows with the input length, increasing memory use and attention computation during decoding (Li et al., 2024; Feng et al., 2025).

KV cache eviction reduces this cost by retaining a subset of the document's key–value pairs (Li et al., 2024; Feng et al., 2025). Eviction also removes information that a later question could require. Document low-rank adaptation (document LoRA) offers a second form of memory: small weight-update matrices that learn document information while the base model stays fixed (Hu et al., 2022; Caccia et al., 2025). The cache preserves token representations, whereas the adapter changes how the model processes a question. Understanding which questions each memory answers well helps determine how to use them together.

In *Rethinking LoRA Memory Through the Lens of KV Cache Compression*, Zuo et al. (2026) study how document LoRA's contribution changes with the KV eviction rate. They show that the adapter contributes more under aggressive eviction. Their study also compares adapter activation stages and supervision formats, but does not analyze which question types favor each memory. Our question-type analysis examines these relative strengths and motivates training the adapter to work with the retained cache. To encourage complementary use of both memories, we first apply KV cache eviction and then train document LoRA on synthetic QA with the selected cache in context.

1.1 RELATED WORK AND LIMITATIONS

KV cache eviction. SnapKV selects cache entries using attention from an observation window at the end of the prompt (Li et al., 2024). PyramidKV varies the cache budget across layers, retaining more entries in lower layers (Cai et al., 2024). R-KV combines attention importance and key redundancy for reasoning models (Cai et al., 2026). Query-aware selection depends on access

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to the question at compression time, which limits reuse of the compressed prefix across different questions (Chari & Van Durme, 2025).

Compactor combines approximate leverage scores with local attention statistics for query-agnostic selection (Chari & Van Durme, 2025). CapKV uses statistical leverage scores to approximate an information-capacity objective for KV retention (Yang et al., 2026). OBCache scores entries by estimating how their removal changes attention outputs (Gu et al., 2026). OBCache reports a bias toward earlier tokens, especially when it measures attention-output changes over larger windows (Gu et al., 2026). These selectors retain a subset of cached states; they do not train document-specific parameters to retain information from the discarded entries.

Document LoRA. Deep Context Distillation trains document-specific modules using a teacher that reads the document (Caccia et al., 2025). Parametric retrieval-augmented generation (PRAG) trains document adapters from augmented text and synthetic QA (Su et al., 2025). PERK meta-learns an adaptation procedure that updates LoRA on a new context through gradient descent (Chen et al., 2025). These approaches require a separate optimization run for each new document (Caccia et al., 2025; Su et al., 2025; Chen et al., 2025). Doc-to-LoRA generates document adapters with a hypernetwork, but adapting it to a new base model requires retraining the hypernetwork (Charakorn et al., 2026).

QA synthesis for document adaptation. The training examples determine how document information enters an adapter. PRAG augments a document with rewrites and generated QA, then trains its LoRA on the combined sequences (Su et al., 2025). Its objective covers document, question, and answer tokens; generated QA alone need not cover every fact in the document (Su et al., 2025). Zuo et al. (2026) generate short-answer QA from document chunks and compare answer-supervised training with raw-text and continuation objectives. Their QA supervision yields stronger document LoRAs under aggressive eviction or without document KV, showing that document exposure alone does not determine QA performance (Zuo et al., 2026). They also measure overlap between synthetic and evaluation questions, since generation from the same source can produce similar questions (Zuo et al., 2026). Chunk-local generation restricts each question to information in its target passage, even when the later task requires relations across passages (Zuo et al., 2026).

1.2 Contributions

Our question-type observations motivate combining the two memories: document LoRA handles some questions well even when it answers without document KV, while other types still benefit from cached context. We first apply KV cache eviction, then use the compressed cache as context to train document LoRA on synthetic QA from the source document. When answering a question, the model uses both the document LoRA and the compressed KV cache. The contributions are:

- **Question-type analysis** (§2): QA-supervised document LoRA improves comparison QA across four models, while Full KV remains stronger on compositional and inference questions.
- **Complementary document memory** (§3): The model learns document LoRA with the selected KV already in context and reuses both memories when answering.
- **Memory ablations** (§5): Removing either memory and varying rank and eviction rate reveal where each memory supports QA and where their combination still fails.

2 Why Combine KV Cache and Document LoRA?

QA-supervised document LoRA and the KV cache favor different question types in Figure 1. A single aggregate QA score hides this difference: an adapter can outperform the full cache on comparison questions and still lose on questions that require following relations.

2.1 Comparing cached and learned document memory

We use the 2WikiMultihopQA task from LongBench (Bai et al., 2023; Ho et al., 2020). Each example pairs a question with a context containing multiple Wikipedia passages, including supporting

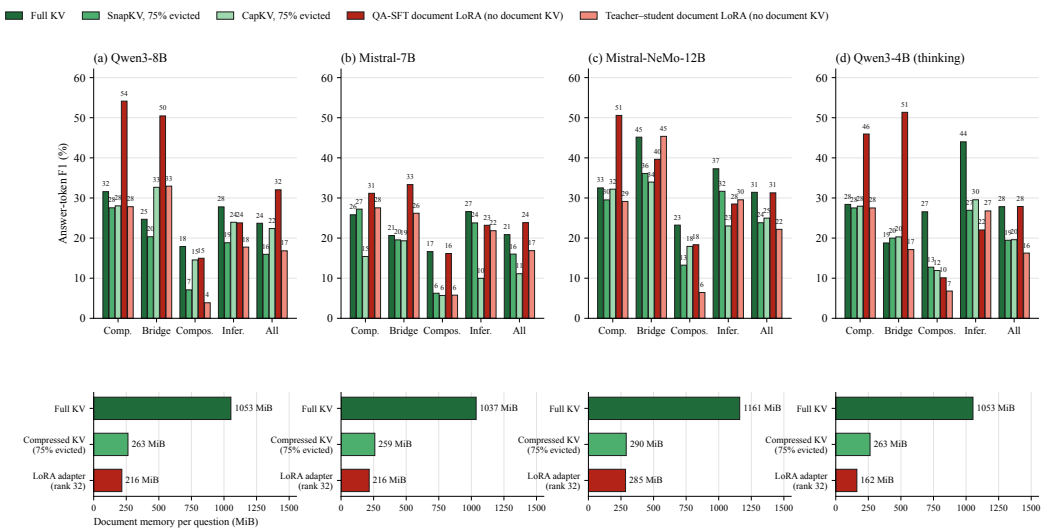


Figure 1: KV cache and document LoRA favor different question types across four models. Top: 2WikiMultihopQA answer-token F1. Bottom: document memory per question. Both rank-32 LoRA controls answer without document KV; SnapKV and CapKV retain 25% of the cache.

evidence and distractors. Answering requires combining information across passages; we treat the combined context as one document. After synthetic-QA filtering, the comparison uses 196 of the 200 documents (Appendix D). These comprise 46 comparison, 37 bridge-comparison, 87 compositional, and 26 inference questions.

Figure 1 compares five memory settings across four models on these 196 documents. Full KV retains the document cache, and SnapKV and CapKV each evict 75% of it. The two document-LoRA controls use the same rank and target modules but different supervision: synthetic QA with answer-token cross-entropy, or teacher–student distillation. Both adapters answer from the question alone, without any document KV. The two adapter controls show how supervision changes QA performance when answering uses only the learned parameters.

Scores average answer-token F1 within each type, and the overall score averages over all questions. The memory panels report the combined size of the stored KV and adapter parameters in MiB.

2.2 The advantage changes with the question type

Comparison questions ask the model to compare attributes of two named entities, such as two films' release years (Ho et al., 2020). QA-supervised LoRA exceeds Full KV on this type in all four models, by 5.3 to 22.6 F1 points. For example, Qwen3-8B scores 54.1 with QA-supervised LoRA and 31.6 with Full KV. Across the four models, the adapters store 162–285 MiB, compared with 1,037–1,161 MiB for Full KV.

Compositional questions follow an explicit relation chain, such as finding a person's spouse and then that spouse's birthplace (Ho et al., 2020). Full KV exceeds QA-supervised LoRA on compositional and inference questions in every model. For Qwen3-4B, the compositional scores are 26.6 and 10.1, and the inference scores are 44.0 and 22.0, respectively. The adapter's advantage on comparisons does not extend to every form of multi-hop reasoning. Appendix C gives a worked example of each question type.

Bridge-comparison questions combine relation lookup with attribute comparison, and their ranking depends on the model. QA-supervised LoRA leads Full KV on this type for Qwen3-8B, Mistral-7B, and Qwen3-4B, while NeMo favors Full KV. Changing the supervision also changes the pattern: QA-supervised LoRA has a higher overall score than the teacher–student control in all four models. This observation supports QA supervision for the adapter, without attributing its behavior to low-rank parameterization alone.

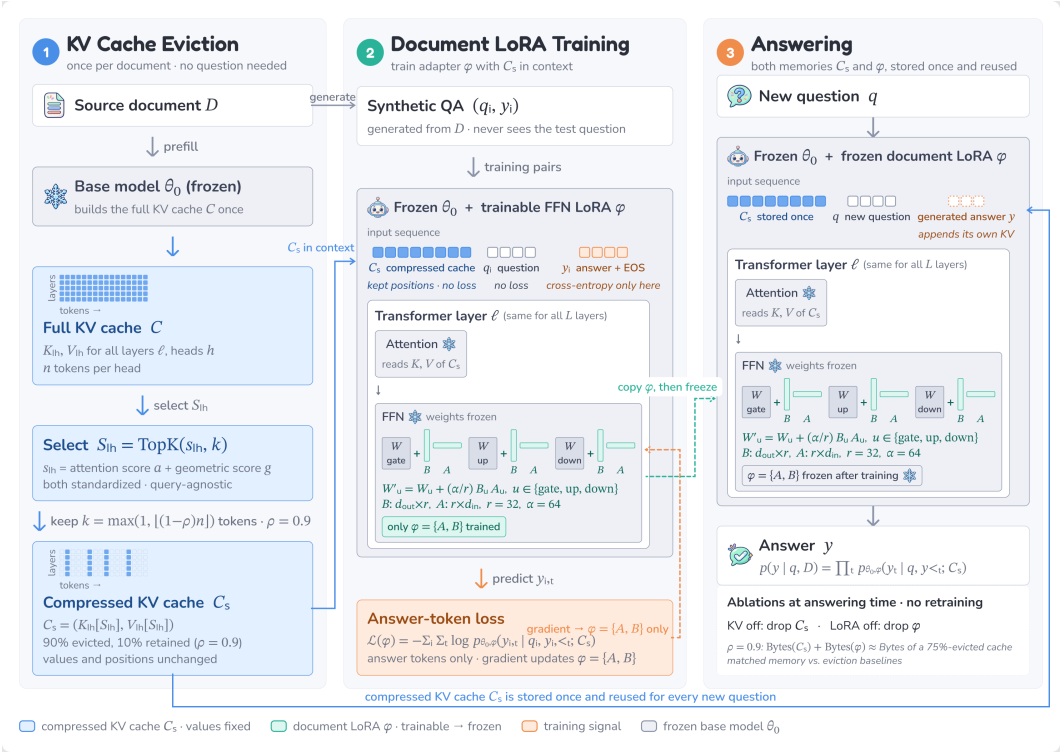


Figure 2: The model first applies KV cache eviction, then trains document LoRA with the selected cache in context, and uses both memories to answer a new question. The orange dashed arrow updates only the adapter.

2.3 Training with both memories available

These differences suggest retaining cached context alongside a QA-trained adapter. The adapter can supply learned responses to source-derived questions, while the cache preserves document states that later attention can read. We therefore apply eviction first and keep the selected cache in context throughout LoRA training, so the training objective evaluates answers with both memories available. The resulting model also uses both memories when answering.

The answering-stage ablations in Section 5 measure each memory's contribution using a fixed trained adapter and its selected cache. They test whether the combination helps question types on which the adapter alone performs poorly.

3 Method

The method represents a document with a compressed KV cache and a document LoRA adapter. Figure 2 shows the three stages: KV cache eviction, document LoRA training, and answering. Eviction fixes the context available during training; LoRA learns from synthetic QA with that context, and answering reuses both memories.

3.1 KV Cache Eviction

The frozen base model encodes source document D into a full prefix cache $\mathcal{C}$. For layer ℓ and KV head h, let $K_{\ell h}, V_{\ell h} \in \mathbb{R}^{n \times d_h}$ contain the cached keys and values for n prefix positions. Here, d_h is the head dimension, and the eviction rate ρ sets the retained count $k = \max(1, \lfloor(1-\rho)n\rfloor)$. Given a score vector $s_{\ell h} \in \mathbb{R}^n$, selection and packing produce

$$S_{\ell h} = \mathrm{TopK}(s_{\ell h}, k), \qquad \mathcal{C}_S = \left\{(K_{\ell h}[S_{\ell h}], V_{\ell h}[S_{\ell h}])\right\}_{\ell,h}. \tag{1}$$

Table 1: Notation for KV cache eviction, document LoRA training, and answering.

Symbol	Meaning
D, q, y	Source document, new question, and generated answer
$\mathcal{C}$, $\mathcal{C}_S$	Full prefix KV cache and retained KV cache
ℓ, h, u	Layer, KV head, and FFN projection indices
n, d_h, ρ, k	Prefix length, head width, eviction rate, and retained positions per head
$s_{\ell h}$, $S_{\ell h}$	Selection scores and retained position indices for layer ℓ, head h
$\widehat{a}$, $\widehat{g}$	Standardized attention and geometric scores
θ_0, ϕ	Frozen base weights and document LoRA parameters
$A_{\ell,u}$, $B_{\ell,u}$, r, α	Trainable low-rank matrices, adapter rank, and LoRA scale
d, d_{ff}, d_{in}, d_{out}	Model width, FFN width, and projection input and output widths
q_i, y_i, N	Training question, training answer, and number of synthetic QA pairs
$\mathcal{B}$, m_i	Sampled microbatch and answer length including the end-of-sequence token
i, t, T	Training-example index, answer-token index, and generated answer length
$\mathcal{L}_{\mathcal{B}}$	Mean answer-token cross-entropy in microbatch $\mathcal{B}$

The operator returns the indices of the k highest scores, and packing orders those indices by their original positions. The selected keys and values retain their original contents and positional encodings.

Selection uses document-derived scores so that cache construction requires no future question (Chari & Van Durme, 2025). For interior positions, the score is $s_{\ell h,j} = \widehat{a}_{\ell h,j} + \rho\,\widehat{g}_{\ell h,j}$, where the hats denote standardized scores. The attention score a measures local attention received by a key, weighted by its value norm; the geometric score g approximates statistical leverage of the pre-position-encoding keys (Chari & Van Durme, 2025). The first favors entries that local document tokens attend to; the second also accounts for distinctive key directions. Boundary positions receive the maximum selection score before the top-k operation. Appendix B identifies the implementation and scoring parameters.

Applying eviction before adaptation gives every training example the same document memory that the model will use when answering. Selection runs once per document, and the index sets $S_{\ell h}$ stay fixed throughout training. The main joint configuration evicts 90% of the cache, leaving room for the adapter in addition to the retained KV. The experiments report both memory components and compare them with eviction-only settings that retain 25% of the cache.

3.2 Document LoRA Training

Source-derived QA. The training set $\{(q_i, y_i)\}_{i=1}^{N}$ contains synthetic questions and answers generated from D, following document-derived QA supervision (Su et al., 2025; Zuo et al., 2026). The generator produces atomic-fact and typed QA from the source document. Atomic-fact QA requires a supporting quote from the document; typed QA also includes evidence quotes and passes a verification step based on those quotes. The main experiments add paraphrases of accepted typed questions while retaining their answers and evidence. The generator never sees the evaluation question; the joint-memory configurations share the augmented training rows for each document. Appendix D gives the acceptance rules and overlap audit.

Trainable parameters. LoRA updates the gate, up, and down projections of every feed-forward network (FFN), following the FFN adaptation used by PRAG (Hu et al., 2022; Su et al., 2025). For projection $u \in \{\text{gate}, \text{up}, \text{down}\}$ at layer ℓ, the adapted weight is

$$W'_{\ell,u} = W_{\ell,u} + \frac{\alpha}{r} B_{\ell,u} A_{\ell,u}, \qquad A_{\ell,u} \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B_{\ell,u} \in \mathbb{R}^{d_{\text{out}} \times r}. \tag{2}$$

Here, r is the rank and α is the scale; the main configuration uses $r = 32$ and $\alpha = 64$. The dimensions $(d_{\text{in}}, d_{\text{out}})$ are (d, d_{ff}) for gate and up projections and (d_{ff}, d) for down projections. The symbols d and d_{ff} denote the model and FFN widths. Only the low-rank parameters $\phi = \{A_{\ell,u}, B_{\ell,u}\}_{\ell,u}$ change; the base weights and stored document KV remain fixed.

Training objective. Each example uses the selected cache $\mathcal{C}_S$, followed by a training question q_i and its answer y_i. Let m_i be the answer length including its end-of-sequence token, and let $\mathcal{B}$ be a sampled microbatch. Supervised fine-tuning minimizes answer-token cross-entropy:

$$\mathcal{L}_{\mathcal{B}}(\phi) = -\frac{\sum_{i\in\mathcal{B}}\sum_{t=1}^{m_i}\log p_{\theta_0,\phi}(y_{i,t} \mid q_i, y_{i,<t}; \mathcal{C}_S)}{\sum_{i\in\mathcal{B}} m_i}. \tag{3}$$

The frozen base parameters are θ_0; question and document tokens carry no prediction loss. The sampler cycles through available question types and draws a QA pair uniformly within each type. Each optimizer step averages the losses of four two-example microbatches. The fixed cache lets each update learn with the retained evidence already available, without rebuilding document KV after changing the adapter.

3.3 Answering

After training, answering uses the selected cache and the frozen adapter. For a new question q and answer sequence $y = (y_1, \ldots, y_T)$, the model generates tokens according to

$$p(y \mid q, D) = \prod_{t=1}^{T} p_{\theta_0,\phi}(y_t \mid q, y_{<t}; \mathcal{C}_S). \tag{4}$$

The source document influences generation through the stored cache and the document-specific parameters ϕ. Question and generated-answer tokens append their own KV while the selected document entries remain unchanged. A subsequent question starts from the same document cache and adapter, without retaining the previous question's generated KV. Answering requires neither the full document cache nor the optimizer state.

4 Experimental Setup

4.1 Models, questions, and training data

The experiments use Qwen3-8B, Qwen3-4B with thinking enabled, Mistral-7B, and Mistral-NeMo-12B. The 2WikiMultihopQA evaluation separates comparison, bridge-comparison, compositional, and inference questions (Ho et al., 2020). Answer-token F1 measures agreement with the reference answer; averages within each type distinguish relation-following questions from attribute comparisons. Figure 1 compares all settings on the 196 documents described in Section 2. Table 2 uses the same 196 documents for every model and method.

The generator reads the source document without the evaluation question. The main experiments use accepted atomic-fact and typed QA plus two requested paraphrases of each accepted typed question. After filtering, this pool contains 8,554 rows across 196 documents, averaging 43.6 rows per document. The paraphrases retain the original answer and evidence; they change the question's wording. Appendix D reports the generation rules and overlap audit.

4.2 Baselines and memory settings

Full KV provides the uncompressed reference. Table 2 compares SnapKV, CapKV, and OBCache with the joint memory; Figure 4 also includes Compactor and PyramidKV. The selectors receive the document prefix before the evaluation question, and query-aware selectors use the end of that prefix as their observation window. Appendix B gives the implementation settings.

In Table 2, the eviction baselines retain 25% of the cache; the full method retains 10% and adds a rank-32 adapter. Document memory is the combined size of the stored KV and adapter parameters, so equal eviction rates would give the joint method a larger memory allocation. The two retention rates bring their memory use closer, and Table 2 reports the actual bytes as MiB. Training uses 160 AdamW steps, four two-example microbatches per step, and a peak learning rate of 2×10^{-4}. Appendix A gives the schedule and precision.

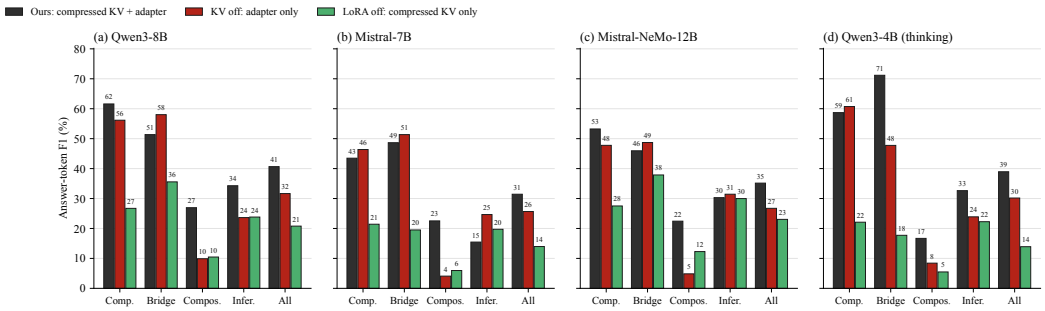


Figure 3: Removing a memory changes the question-type profile. Answer-token F1 for the full method, KV off, and LoRA off uses the same 196 documents for every model. The cache retains 10%.

Table 2: 2WikiMultihopQA answer-token F1 (%) and document memory (MiB). Eviction baselines retain 25% of the cache; the full method retains 10% and adds a rank-32 adapter. All settings use the same 196 documents. Bold marks the highest F1 in each column.

Method	Qwen3-8B	Qwen3-4B (Thinking)	Mistral-7B	NeMo-12B
2WikiMultihopQA, answer-token F1 (%) ↑				
Full KV	23.7	27.8	20.9	31.4
SnapKV (75% evicted)	16.0	19.5	16.0	23.8
CapKV (75% evicted)	22.4	19.6	11.1	25.0
OBCache (75% evicted)	16.9	18.9	17.0	24.7
Ours (90% evicted KV + adapter)	**40.7**	**38.9**	**31.4**	**35.1**
Document memory per question (MiB) ↓				
Full KV	1053	1053	1037	1161
SnapKV (75% evicted)	263	263	259	290
CapKV (75% evicted)	263	263	259	290
OBCache (75% evicted)	263	263	259	290
Ours (90% evicted KV + adapter)	321	267	320	401

4.3 Memory ablations

KV off removes document KV from the trained model; LoRA off disables its adapter while retaining the same selected cache. Neither condition retrains the model or changes which KV entries the full method selected. These comparisons measure each memory's contribution within a trained configuration. The rank sweep separately retrains adapters at each rank and eviction rate on a fixed 48-document subset.

5 Results

5.1 Accuracy and document memory

Table 2 pairs 2WikiMultihopQA answer-token F1 with the memory that each setting stores. The joint configuration uses 267–401 MiB per document across the four models, compared with 1,037–1,161 MiB for Full KV. Its adapter makes the total larger than a 10%-retained cache alone; comparing the stored totals captures that cost. The eviction-only baselines use 259–290 MiB, so their budgets are close to, but smaller than, the joint configuration's budgets.

The joint memory exceeds Full KV in all four models, with F1 gains from 3.7 to 17.0 points. Qwen3-8B shows the largest gain, improving from 23.7 to 40.7 while reducing document memory from 1,053 to 321 MiB.

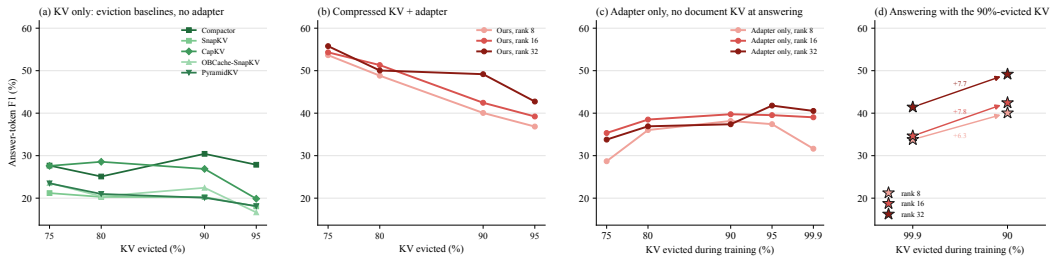


Figure 4: Rank and training context affect answer-token F1 on 48 Qwen3-8B documents. (a) KV eviction alone. (b) Compressed KV with an adapter. (c) Adapter-only answers after training at each eviction rate. (d) Both conditions answer with the same 90%-evicted cache; arrows show the gain from training with that cache instead of at 99.9% eviction.

5.2 The memories help different question types

Figure 3 disables one memory during answering while keeping the trained adapter and selected cache fixed. For Qwen3-8B, the full method scores 40.7 overall, compared with 31.7 for KV off and 20.8 for LoRA off. Removing the cache lowers its compositional score from 26.9 to 9.9, while comparison drops from 61.6 to 56.2. This contrast links the gain from retained context to questions that require following relations.

The compositional score falls after removing the cache in all four models, but cache access does not improve every category. Mistral-7B and Qwen3-4B score higher on comparison questions with KV off than with both memories. Qwen3-8B, Mistral-7B, and NeMo favor KV off on bridge-comparison questions. On inference questions, Mistral-7B and NeMo also favor KV off, scoring 24.7 and 31.4 against 15.5 and 30.3 with both memories. Combining the memories improves overall F1 in every model, while these categories leave room to improve how the model uses retained context.

5.3 Rank and eviction rate change the balance

Figure 4 compares KV eviction alone and ranks 8, 16, and 32 across 75–95% eviction on Qwen3-8B. For each rank and eviction rate, we train a separate adapter for each of 48 documents, with 12 documents per question type. At rank 32, raising eviction from 75% to 95% reduces F1 from 55.8 to 42.7 and stored memory from 495 to 272 MiB. At 95% eviction, rank 32 exceeds ranks 8 and 16, which score 36.9 and 39.2. A larger adapter partly offsets the loss of cached context in this sweep. Panel (c) also includes adapters trained at 99.9% eviction, which retain only a few document KV entries during training.

The adapter-only panel changes even though every point evaluates without document KV. For rank 32, KV-off F1 rises from 33.8 after training at 75% eviction to 41.8 after training at 95% eviction. The cache available during training therefore affects the adapter's later behavior, beyond the direct benefit of reading KV when answering. Panels (b) and (c) show why rank and eviction rate need joint evaluation: they change both the available context and what the adapter learns.

Training with the answering cache. Figure 4(d) compares adding the answering cache after training with keeping that cache in context throughout training. Both conditions answer the same 48 questions with identical KV entries from 90% eviction. The control trains at 99.9% eviction and switches to the larger cache only for answering; the other condition trains with the 90%-evicted cache. Training with the answering cache raises F1 from 33.8 to 40.1 at rank 8, from 34.6 to 42.5 at rank 16, and from 41.4 to 49.2 at rank 32. The gains of 6.3, 7.8, and 7.7 points support providing the selected cache during adaptation so the adapter learns to answer with the context it will later receive.

6 Discussion and Limitations

Training cost and reuse. Document LoRA incurs an upfront training cost in addition to QA synthesis and KV cache construction. Subsequent questions reuse the learned adapter and compressed cache. When the compressed cache improves answering throughput, cumulative time savings over repeated questions can offset this upfront cost.

Coverage and generalization. Synthetic QA covers selected facts and relations from a document. Evidence checks reject unsupported training rows, but they do not guarantee that the accepted rows cover every later question. The augmented-data audit flags 30 of 196 documents under the union of the ROUGE-L and shared-8-gram criteria. The reported results retain these flagged documents; a separate evaluation on the 166 unflagged documents would test sensitivity to question overlap. The current documents also fit within the evaluated models' context windows, so these experiments do not validate adaptation beyond those windows.

Limits of complementary memory. The ablations show that the two memories can help different question types, while some subsets favor the adapter alone. A fixed selection rule and training objective do not guarantee that adding a cache improves every answer. The evidence covers four models on one benchmark, and the rank sweep covers 48 documents from that benchmark. Evaluation on other tasks and longer contexts would test how broadly the observed complementarity holds.

AI use statement

We used generative AI for literature review, code development, experiment configuration, data analysis, and manuscript revision. Language models also generated the document QA used for adaptation. The authors take responsibility for the manuscript and accompanying artifacts.

REFERENCES

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. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024. URL https://arxiv.org/abs/2412.15204.

Yushi Bai et al. LongBench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023. URL https://arxiv.org/abs/2308.14508.

Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. Training plug-n-play knowledge modules with deep context distillation. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2503.08727v4.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. URL https://arxiv.org/abs/2406.02069.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-KV: Redundancy-aware KV cache compression for reasoning models. *arXiv preprint arXiv:2505.24133v4*, 2026. URL https://arxiv.org/abs/2505.24133v4.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts. *arXiv preprint arXiv:2602.15902v1*, 2026. URL https://arxiv.org/abs/2602.15902v1.

Vivek Chari and Benjamin Van Durme. Compactor: Calibrated query-agnostic kv cache compression with approximate leverage scores. *arXiv preprint arXiv:2507.08143*, 2025. URL https://arxiv.org/abs/2507.08143.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. *arXiv preprint arXiv:2507.06415v1*, 2025. URL https://arxiv.org/abs/2507.06415v1.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2407.11550.

Yuzhe Gu, Xiyu Liang, Jiaojiao Zhao, and Enmao Diao. OBCache: Optimal brain KV cache pruning for efficient long-context LLM inference. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2510.07651v4.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020. URL https://aclanthology.org/2020.coling-main.580/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Yuhong Li et al. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2404.14469.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. *arXiv preprint arXiv:2501.15915v1*, 2025. URL https://arxiv.org/abs/2501.15915v1.

Jiaming Yang, Chenwei Tang, Liangli Zhen, and Jiancheng Lv. Rethinking KV cache eviction via a unified information-theoretic objective. *arXiv preprint arXiv:2604.25975v2*, 2026. URL https://arxiv.org/abs/2604.25975v2.

Chunsheng Zuo, Liaoyaqi Wang, William Jurayj, William Fleshman, and Benjamin Van Durme. Rethinking lora memory through the lens of kv cache compression. *arXiv preprint arXiv:2606.05698*, 2026. URL https://arxiv.org/abs/2606.05698.

SUPPLEMENTARY MATERIAL

A TRAINING CONFIGURATION

Table 3: Training settings for the joint memory configuration.

Component	Setting
LoRA	Rank 32; $\alpha = 64$
Target modules	Gate, up, and down projections in every layer
Optimizer	AdamW; 160 steps; zero weight decay
Learning rate	Peak 2×10^{-4}; 8-step linear warmup, then cosine decay
Gradient accumulation	Four microbatches of two QA pairs per step
Precision	Float32 adapter; bfloat16 base weights and KV cache
Document KV	10% retained during training and answering

Compactor selects the retained KV; the trained adapter adds its own memory footprint. The eviction baselines in Table 2 and Figure 1 retain 25% of the cache; Figure 4 varies this rate. The two document-LoRA controls in Figure 1 use rank-32 FFN adapters and answer without document KV.

B KV EVICTION SETTINGS

All selectors run on the document prefix before the evaluation question arrives. Query-aware selectors use the final document tokens to score earlier entries.

Table 4: Baseline implementations and selection parameters.

Selector	Implementation and parameters
SnapKV	Pinned KVPress; 64-token observation window; pooling kernel 5 (Li et al., 2024)
Compactor	Published 256-token neighborhoods, global pooling, and normalization (Chari & Van Durme, 2025)
PyramidKV, CapKV, OBCache	Pinned author or KVPress implementations; default settings

The implementation preserves original positional encodings when copying selected KV into compact storage.

C QUESTION-TYPE ANALYSIS

The four types in 2WikiMultihopQA distinguish attribute comparison, relation chaining, and logical inference (Ho et al., 2020). The examples pair benchmark questions and reference answers with the reasoning needed to answer them.

C.1 COMPARISON

- **Question:** Which film came out earlier, Indradhanura Chhai or The Death Of Black King?
- **Answer:** *The Death Of Black King*.
- **Reasoning:** Retrieve the release years: 1993 for *Indradhanura Chhai* and 1971 for *The Death of Black King*. Compare the years and return the earlier film.

The question names both comparison targets directly.

C.2 BRIDGE-COMPARISON

- **Question:** Do both Beauty And The Bad Man and Wild Child (Film) films have the directors from the same country?
- **Answer:** *no*.
- **Reasoning:** Identify each film's director: William Worthington and Nick Moore, respectively. Retrieve their nationalities, American and British, then compare the countries.

The films lead to the directors through a relation; the comparison concerns these intermediate entities.

C.3 COMPOSITIONAL

- **Question:** Where was the wife of Francis I Rákóczi born?
- **Answer:** *Ozalj*.
- **Reasoning:** Identify his wife as Jelena Zrinska, also known as Ilona Zrínyi. Retrieve her birthplace, Ozalj.

The question explicitly specifies the two relations to follow: spouse and birthplace.

C.4 INFERENCE

- **Question:** Who is Sobe (Sister Of Saint Anne)'s grandchild?
- **Answer:** *John the Baptist*.
- **Reasoning:** The passages identify Sobe as Elizabeth's mother and Elizabeth as John the Baptist's mother. Combining these parent relations establishes the grandchild relation.

The question asks for the inferred grandchild relation without naming the intermediate parent.

D SYNTHETIC TRAINING DATA AND OVERLAP AUDIT

D.1 SOURCE DOCUMENTS AND ATOMIC-FACT QA

QA synthesis starts from the 200 document contexts in LongBench's 2WikiMultihopQA task (Bai et al., 2023; Ho et al., 2020). Training rows come from two source-only passes over each document. The generator never sees the benchmark question.

The first pass produces atomic-fact QA with a single supporting quote that contains the answer. The acceptance filter requires the quote to occur in the source and the answer to contain at most 16 words.

D.2 TYPED QA: ACCEPTANCE AND VERIFICATION

The second pass uses Qwen3-14B to read the whole document and generate comparison, bridge-comparison, compositional, and inference QA.

Acceptance rules. Each typed row must satisfy the following checks:

- **Evidence:** At least two quotes, each longer than ten characters and present verbatim in the normalized source.
- **Reasoning:** At least two reasoning steps; bridge-comparison requires at least four quotes and four steps.
- **Answer and fields:** At most 24 answer words and all required fields present.
- **Uniqueness:** The normalized question must differ from earlier questions.

Evidence-only verification. The generator judges each surviving candidate using only its quoted evidence. It checks the answer, every reasoning link, the declared type, and whether the question requires more than one fact. The filter rejects verdicts marked wrong, ambiguous, shortcut, or ungrounded, as well as candidates without a verdict.

D.3 ACCEPTED TRAINING DATA

We exclude documents with no accepted typed QA. Four of the 200 documents fail this condition, all of the compositional type, leaving 196 documents. The accepted dataset contains 7,239 rows, averaging 36.9 per document:

- **Atomic-fact QA:** 6,559 rows from the first pass.
- **Typed QA:** 680 rows: 336 comparison, 225 compositional, 116 inference, and 3 bridge-comparison.

All models and both objectives in the observation experiment train on the same accepted rows for a document. A final filter removes any synthetic question that exactly matches the normalized benchmark question.

D.4 PARAPHRASE AUGMENTATION FOR THE MAIN EXPERIMENTS

The main experiments in Section 4 add paraphrases to the accepted rows. The same generator receives a request for two paraphrases of every accepted typed question. It changes only the wording and retains the original answer, evidence, and verdict.

- **Acceptance:** Reject normalized text that duplicates the original or another question, or whose ROUGE-L against the original falls below 0.2.
- **Dataset size:** 8,554 rows across 196 documents, averaging 43.6 rows per document.
- **Average composition:** 3.5 typed originals, 6.7 paraphrases, and 33.4 atomic facts per document.

D.5 OVERLAP WITH BENCHMARK QUESTIONS

The audit compares each synthetic question with the benchmark question from the same document after text normalization. A document receives a flag if any synthetic question reaches ROUGE-L ≥ 0.7 or shares an eight-word sequence with the benchmark question. Table 5 compares the original and augmented training pools using these criteria.

Table 5: Question overlap before and after paraphrase augmentation, across 196 documents.

Criterion	Before	After
ROUGE-L ≥ 0.7	22	29
Any shared 8-gram	3	3
Either criterion	24	30
Flag with the same gold answer	2	2
Exact normalized question match	0	0
Neither criterion	172	166

Paraphrase augmentation increases the union of flagged documents from 24 to 30. The reported main results retain flagged documents; the augmented pool contains 166 documents without a flag under either criterion. An overlap flag measures question similarity and does not by itself establish that a synthetic row duplicates the benchmark question.