RESTOREKV: RECOVERING FULL-CACHE BEHAVIOR UNDER AGGRESSIVE QUERY-AGNOSTIC KV CACHE EVICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Query-agnostic KV cache eviction compresses a context once and reuses the resulting cache for arbitrary future queries, but performance can collapse under tight budgets. Existing methods primarily improve which original KV pairs are retained. We introduce RestoreKV, which complements this selection-based formulation with learned restoration under the same total KV budget. The restore generator is trained to complement retained original KV states through predictions from their combined cache, rather than to replace the entire context. After context prefill, a few restore tokens attend to the full KV cache in a single LoRA-adapted pass, generating a compact, context-conditioned restore cache. The base importance scorer and eviction rule remain unchanged, and the adapters are disabled for all subsequent queries and decoding. RestoreKV is trained through parameter-efficient self-distillation from the frozen full-cache model, optimizing only 0.4% of the parameters and requiring no task-specific tuning. Across four backbones and four long-context benchmarks, RestoreKV substantially reduces compression-induced degradation. On Qwen3-4B, it improves 59 of 60 paired, budget-matched settings across five base eviction methods; at a 5% budget, it raises KVzip from 38.2 to 73.2 on RULER-4K. On Qwen3-8B, applying RestoreKV to KVzip+ reaches 86.4 RULER accuracy at $16\times$ compression on the KVPress Benchmark, while adding at most 0.5% one-time cache-construction overhead in a 32K-context evaluation.

1 INTRODUCTION

Large language models increasingly rely on long contexts, but their key–value (KV) caches grow linearly with sequence length. KV cache eviction reduces this cost by retaining only a subset of cached KV pairs (Xiao et al., 2024; Li et al., 2024; Zhang et al., 2023; Cai et al., 2025; Liu et al., 2025; Feng et al., 2025; Park et al., 2025; Tang et al., 2025). In particular, query-agnostic eviction methods compress a context once before future queries are known and reuse the resulting cache across arbitrary subsequent requests (Kim et al., 2025; Jegou & Jeblick, 2026; Chen et al., 2026). Recent methods in this setting largely follow the same formulation: *select a better subset of the original KV pairs*. Although they differ in their importance signals or learned predictors (Kim et al., 2025; Jegou & Jeblick, 2026; Chen et al., 2026; Kim et al., 2026), the resulting cache remains composed only of retained original states. The central question has therefore been which KV pairs should survive under a fixed budget.

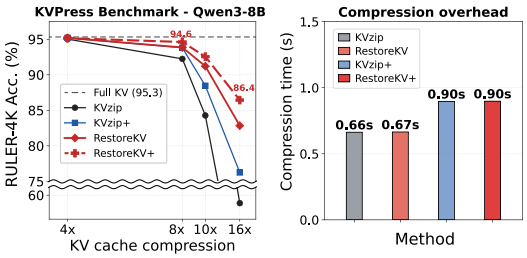

Figure 1: **RestoreKV narrows the gap to full-cache behavior under aggressive compression.** Applied to KVzip+, it reaches 86.4 RULER accuracy at $16\times$ compression on the KVPress Benchmark with negligible overhead.

However, better retention alone does not fully prevent degradation under aggressive eviction. As shown in Fig. 1, the base eviction methods remain close to the full-cache model at mild compression

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.

but diverge sharply as the compression ratio increases, whereas RestoreKV substantially narrows this gap. This motivates a complementary question: rather than relying only on a better subset, can we generate a small context-conditioned complement before eviction to preserve behavior that would otherwise be lost? Although the vulnerable information differs across contexts, the mechanism for generating its compact complement may be learned across them.

In this work, we introduce **RestoreKV**, which learns a small complement to the KV pairs retained by an existing query-agnostic eviction method. Rather than replacing the entire context with a learned compact memory, RestoreKV reuses the frozen prefill cache and processes only eight additional positions in a single LoRA-adapted pass, as illustrated in Fig. 2. The base importance scorer, eviction rule, and retained KV representations remain unchanged. Crucially, the restore generator is trained through predictions made using the combined retained-and-restore cache, rather than a standalone compressed representation. The generated states occupy part of the existing KV budget, and the adapters are disabled after cache construction. This design improves aggressive eviction without increasing the total KV budget or introducing an additional decoding-time operator.

RestoreKV is trained offline through self-distillation from a frozen full-cache teacher. For the same context and query, the student using the final budget-matched cache is optimized to match the teacher's answer-token distributions. Only the restore-token embeddings and lightweight LoRA (Hu et al., 2022) adapters are updated, while the backbone and base evictor remain frozen. Component ablations clarify what the restore pass learns: at a 5% KV budget, optimizing the restore embeddings alone improves KVzip from 38.2 to 42.1, whereas fixed embeddings with learned LoRA adaptation reach 71.9. Adapting only the $q/k/v$ projections further reaches 72.4, close to RestoreKV's 73.2. These results indicate that the gain arises primarily from attention-side adaptation that generates a context-conditioned restore cache, rather than from generic information memorized in the restore embeddings.

Across four model backbones, four long-context benchmarks, and five base eviction methods, RestoreKV consistently narrows the gap to full-cache performance, with larger gains typically observed under tighter budgets. On Qwen3-4B at a 5% KV budget ratio, it improves KVzip from 38.2 to 73.2 on RULER-4K. Applied to KVzip+, RestoreKV reaches 86.4 RULER accuracy at $16\times$ compression on the KVPress Benchmark (Fig. 1). In our 32K-context evaluation, the eight-token restore pass adds only 0.03–0.04 seconds, at most 0.5% of the total compression time, and 84 MB (0.4%) of peak memory, while preserving the query-time KV budget and decoding cost of the base method.

Our contributions are summarized as follows:

- We introduce a complementary restoration perspective on query-agnostic KV eviction: instead of learning only which original KV pairs to retain, we learn a shared mechanism that generates a compact, context-conditioned restore cache to complement the retained context cache.
- We propose RestoreKV, a budget-matched, single-pass plug-in that preserves the base importance scorer and eviction rule. Its LoRA adapters are used only during restore-cache generation, leaving subsequent query processing and decoding unchanged.
- We demonstrate consistent improvements across five eviction methods, four model backbones, and four long-context benchmarks, and show that attention-side adaptation is the primary source of recovery under aggressive eviction.

2 Related Work

2.1 Selection-Based KV Cache Eviction

Most KV cache eviction methods estimate KV importance using signals associated with a specific query, such as query-dependent attention or activation statistics (Li et al., 2024; Zhang et al., 2023). Although effective when the request is already known, these methods cannot construct a compressed cache before future queries are observed. Query-agnostic eviction instead compresses a context once and reuses the resulting cache across arbitrary subsequent requests. KVzip (Kim et al., 2025) estimates the contribution of individual KV pairs through context reconstruction, while KVzip+ (Jegou & Jeblick, 2026) and ContrastKV (Chen et al., 2026) refine the importance signal through output-norm weighting and contrastive objectives, respectively.

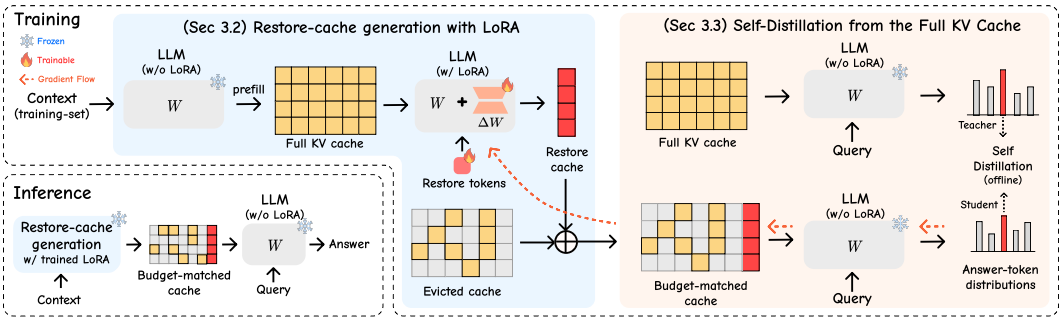

Figure 2: **Overview of RestoreKV.** After context prefill, a single LoRA-adapted restore pass processes n restore tokens (n=8 by default) with causal access to the full KV cache, generating nLH context-conditioned KV pairs. These restore states are combined with the retained context states under the same total KV budget. LoRA is active only during restore-cache generation; subsequent query processing and decoding use the original frozen backbone. **Training (offline):** The restore-token embeddings and LoRA adapters (0.4% of the 4B backbone) are optimized by self-distillation from the frozen full-cache teacher. **Inference:** The trained restore tokens and LoRA adapters generate the restore cache once for each new context.

Recent methods further amortize or improve importance estimation using lightweight trainable modules. KVzap (Jegou & Jeblick, 2026) predicts KVzip+ importance scores directly from hidden states, while Fast KVzip (Kim et al., 2026) learns sink-attention gates through context reconstruction. LookaheadKV (Ahn et al., 2026) considers the query-aware setting and uses lookahead tokens with selectively activated LoRA modules to predict response-induced importance without generating a surrogate response. Despite differences in query availability and scoring mechanisms, these methods share the same selection-based endpoint: learning determines which original KV pairs should survive, while the resulting compressed cache remains a subset of the original cache.

2.2 Synthesized KV Cache Representations

Beyond selecting unmodified KV pairs, recent methods expand the representational space of the compressed cache. KV-Distill (Chari et al., 2025) jointly learns an importance scorer and parameter-efficient adapters that rewrite selected-token representations, using the full-cache predictive distribution as a distillation target. Cartridges (Eyuboglu et al., 2026) instead optimizes a context-specific parameterized KV cache through self-study.

Concurrent work explores several related directions. Attention Matching (Zweiger et al., 2026) constructs compact keys and values by matching per-head attention outputs and attention mass over reference queries. VECTOR (Lin et al., 2026) retains the keys of approximated cache entries and reconstructs their values during generation using an offline-calibrated linear map. Latent Context Compilation (Li et al., 2026) optimizes a disposable LoRA for each context to compile it into buffer-token KV states, while Still (O'Neill et al., 2026) uses learned per-layer compactors to replace the original prefix cache.

RestoreKV instead learns *a small complement to a fixed eviction pipeline*, rather than a standalone context replacement. Its distinction is not the use of learnable tokens or LoRA alone, but their role in generating states optimized for use together with retained original KV pairs. Specifically, full-cache self-distillation supervises predictions from the *combined retained-and-restore cache*, not the restore states alone, while the backbone, base importance scorer, and eviction rule remain fixed. The generated states replace an equal number of retained KV pairs within the same total budget, without rewriting the remaining original states. A separate generator is trained offline for each backbone–evictor configuration and reused across contexts, downstream tasks, and KV budgets. For each new context, a single LoRA-adapted pass over eight additional positions reuses the frozen prefill cache to construct restore states, without per-context optimization. The adapters are then disabled, and subsequent query processing and decoding use the original frozen model, without generation-time reconstruction or an additional attention operator. Query–answer pairs provide offline supervision only: *query-agnostic* refers to constructing the reusable cache without observing its future queries.

3 METHOD

RestoreKV complements an existing query-agnostic KV evictor with a learned restoration mechanism. As illustrated in Fig. 2, it generates a compact, context-conditioned restore cache from the full KV cache before eviction and combines it with the retained context cache under the same total KV budget. The base importance scorer and eviction rule remain unchanged. The restoration mechanism is trained through self-distillation from the frozen full-cache model and is used only during one-time restore-cache construction.

3.1 QUERY-AGNOSTIC EVICTION FORMULATION

Consider a decoder-only Transformer f_θ with L layers and H KV heads. Given a context $x_{1:T}$, the prefill stage produces a full KV cache $\mathcal{C}$ containing TLH KV pairs. Generation conditioned on this cache is denoted by $f_\theta(\cdot \mid \mathcal{C})$.

A query-agnostic eviction method assigns importance scores $\mathbf{s} \in \mathbb{R}^{L\times H\times T}$ without observing future queries. At KV budget ratio r, the total KV budget is

$$\mathcal{B} = \lfloor rTLH \rfloor. \tag{1}$$

The resulting retained context cache is

$$\mathcal{C}' = \mathrm{Evict}\left(\mathcal{C}, \mathbf{s}, \mathcal{B}\right), \tag{2}$$

where $|\mathcal{C}'| = \mathcal{B}$. The operator Evict follows the layer- and head-wise allocation rule of the underlying method. Once constructed, $\mathcal{C}'$ can be reused across arbitrary subsequent queries.

Selection-based eviction constructs $\mathcal{C}'$ solely from original KV pairs in $\mathcal{C}$. RestoreKV keeps the same importance scores and allocation rule, but reserves a small portion of the fixed budget for a generated, context-conditioned restore cache. We next describe its generation, budget-matched composition, and training objective.

3.2 RESTORE-CACHE GENERATION WITH LORA

RestoreKV introduces n learnable restore-token embeddings

$$E = [e_1, \ldots, e_n] \in \mathbb{R}^{n\times d}, \tag{3}$$

where d is the hidden dimension. After context prefill, the restore tokens are processed at positions $T+1, \ldots, T+n$ with causal access to the full KV cache $\mathcal{C}$. A single LoRA-adapted restore pass generates the context-conditioned restore cache

$$\mathcal{C}_{\mathrm{res}} = \mathrm{Restore}_{\theta,\phi}\left(E \mid \mathcal{C}\right), \tag{4}$$

where θ denotes the frozen backbone parameters and ϕ denotes the LoRA parameters (Hu et al., 2022). The restore-token embeddings and LoRA parameters are shared across contexts, whereas $\mathcal{C}_{\mathrm{res}}$ is specific to the current context.

For each adapted linear projection W, LoRA applies

$$\Delta W = \frac{\alpha}{r_{\mathrm{LoRA}}} B_\phi A_\phi, \qquad W_{\mathrm{res}} = W + \Delta W, \tag{5}$$

where r_{LoRA} is the LoRA rank, α is the scaling factor, and A_ϕ and B_ϕ are trainable low-rank matrices, while W remains frozen. LoRA is enabled only during the restore pass; context prefill and all subsequent query processing and decoding use the original backbone f_θ.

Each restore token produces one KV pair per layer and KV head, yielding $|\mathcal{C}_{\mathrm{res}}| = nLH$. Because the restore cache is generated before eviction, it can incorporate information from all context KV pairs, including those subsequently removed.

To preserve the query-time KV budget, RestoreKV reserves nLH of the $\mathcal{B}$ cache slots for $\mathcal{C}_{\mathrm{res}}$ and lets the base evictor fill the remaining $\mathcal{B} - nLH$ slots:

$$\begin{aligned} \widetilde{\mathcal{C}} &= \mathrm{Concat}(\mathrm{Evict}\left(\mathcal{C}, \mathbf{s}, \mathcal{B} - nLH\right), \mathcal{C}_{\mathrm{res}}), \\ \left|\widetilde{\mathcal{C}}\right| &= (\mathcal{B} - nLH) + nLH = \mathcal{B} = |\mathcal{C}'|. \end{aligned} \tag{6}$$

Here, Concat combines the retained context cache and restore cache without changing their position indices. The base method still determines the importance scores and layer- and head-wise allocation of the retained context budget; RestoreKV changes only the cache composition. Retained context KV pairs preserve their original RoPE (Su et al., 2024) phases, while the restore tokens occupy positions $T+1, \ldots, T+n$ and future queries begin at position $T+n+1$.

3.3 Self-Distillation from the Full KV Cache

We train the restore-token embeddings E and LoRA parameters ϕ through self-distillation from the frozen full-cache model (Hinton et al., 2015). Given a training context $x_{1:T}$ and query $q = (q_1, \ldots, q_Q)$, the full-cache teacher first generates an answer $y = (y_1, \ldots, y_M)$. The teacher and the restored-cache student then evaluate the same answer using $\mathcal{C}$ and the final budget-matched cache $\widetilde{\mathcal{C}}$, respectively:

$$\begin{aligned} p_i^{\text{full}}(\cdot) &= p_\theta(\cdot \mid q, y_{<i}, \mathcal{C}), \\ p_i^{\text{res}}(\cdot) &= p_\theta\Big(\cdot \mid q, y_{<i}, \widetilde{\mathcal{C}}\Big). \end{aligned} \tag{7}$$

The query and teacher answer are used only to define the offline distillation target; the context-conditioned restore cache is generated without observing the query, preserving query-agnostic cache construction.

We minimize the token-averaged symmetric KL divergence

$$\mathcal{L}_{\text{distill}} = \frac{1}{2M} \sum_{i=1}^{M} \left[\mathrm{KL}\big(p_i^{\text{full}} \big\| p_i^{\text{res}}\big) + \mathrm{KL}\big(p_i^{\text{res}} \big\| p_i^{\text{full}}\big) \right]. \tag{8}$$

The teacher distributions are detached, and only E and ϕ are updated; the backbone θ and base importance scorer remain frozen. We uniformly sample the KV budget ratio,

$$r \sim \mathcal{U}(r_{\min}, r_{\max}), \qquad 0 < r_{\min} < r_{\max} < 1, \tag{9}$$

so that one shared restoration mechanism supports multiple cache budgets.

3.4 Inference

At inference, RestoreKV generates $\mathcal{C}_{\text{res}}$ once from the full cache before future queries are observed and constructs the final budget-matched cache $\widetilde{\mathcal{C}}$ using Eq. (6). The same cache can then be reused across arbitrary subsequent queries. LoRA is enabled only for the n-token restore pass and disabled thereafter, so all query processing and decoding use the original frozen backbone. Because the restore cache replaces an equal number of retained context KV pairs, RestoreKV preserves the query-time KV budget and decoding cost of the base method; its only additional computation is a one-time forward pass over n positions.

4 Experiments

We evaluate RestoreKV on four benchmarks under aggressive query-agnostic KV cache eviction. The restore pass is trained *once per model and eviction method*, and reused across all downstream tasks without any task-specific tuning.

4.1 Experimental Setup

Datasets. We evaluate on four benchmarks spanning distinct query types, ordered below by increasing context length. RULER (Hsieh et al., 2024) at 4K context is a 13-task synthetic suite covering retrieval, tracing, and aggregation (fixed 4K); we follow the official KVPress evaluation protocol (Devoto et al., 2025). QASPER (Dasigi et al., 2021), taken from LongBench (Bai et al., 2024), is free-form QA over scientific papers (avg. $\sim$5K tokens). QuALITY (Pang et al., 2022) is multiple-choice reading comprehension over long articles ($\sim$6K tokens). LongHealth (Adams et al., 2025) is multiple-choice QA over clinical patient records ($\sim$11K tokens). Matching the query-agnostic setting, each context is compressed once and reused across all of its questions. To assess generalization to longer contexts, we further report results on RULER at 32K and 128K tokens (Table 2), and on SCBench ($\sim$104K tokens) and LongBench in Appendix B.

Models. We conduct experiments on Qwen3-0.6B/4B/8B (Yang et al., 2025) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024). The backbone remains frozen in all experiments; only the restore-token embeddings and LoRA adapters are trained.

Baselines. Our primary base methods are KVzip (Kim et al., 2025), KVzip+ (Jegou & Jeblick, 2026), and ContrastKV (Chen et al., 2026), which use reconstruction-based, output-norm-weighted, and contrastive importance scores, respectively. We additionally evaluate query-agnostic adaptations

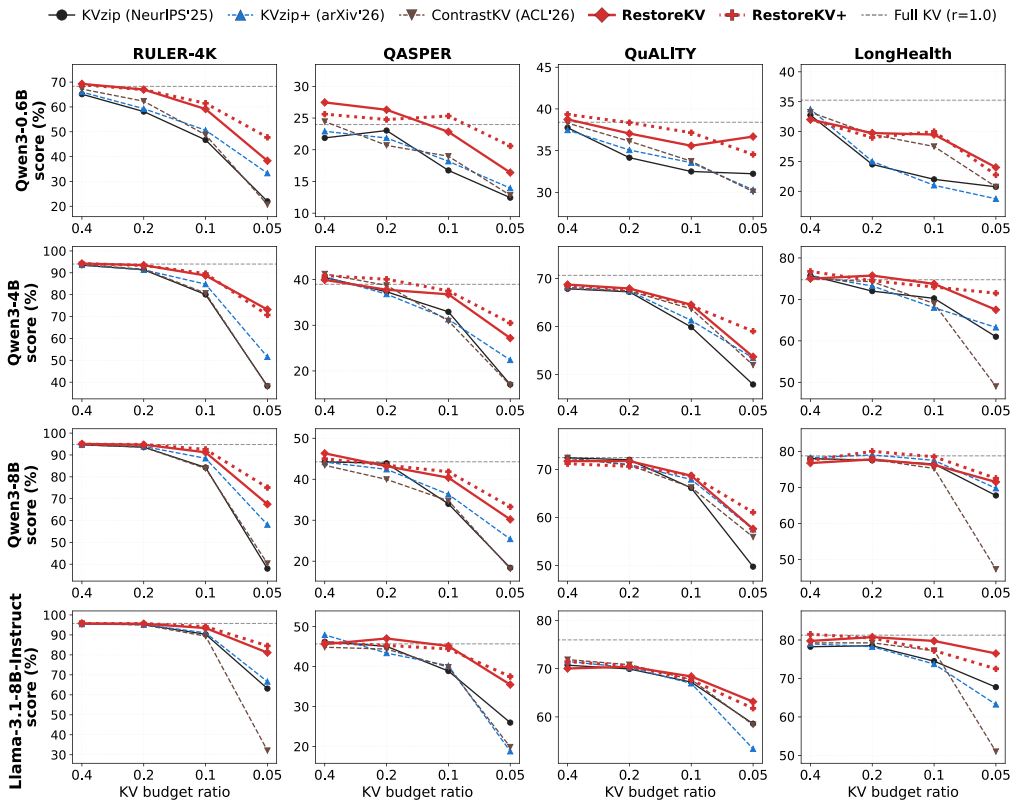

Figure 3: **Performance across four model backbones and four benchmarks.** RestoreKV and RestoreKV+ are applied to KVzip and KVzip+, respectively, under the same KV budget. Both methods reduce compression-induced performance loss, with larger gains under tighter cache budgets.

of SnapKV (Li et al., 2024) and H_2O (Zhang et al., 2023) to test compatibility with different eviction rules; implementation details are provided in the appendix. For every base method, RestoreKV reserves nLH slots from the existing budget for the restore cache, so each paired comparison uses exactly the same query-time KV memory. RestoreKV changes neither the base importance scorer nor its eviction rule. For brevity, figures denote KVzip with RestoreKV as **RestoreKV** and KVzip+ with RestoreKV as **RestoreKV+**; the "+" refers to the base method. All generations use deterministic greedy decoding.

Training details. We use n=8 restore tokens with LoRA rank r_{LoRA}=8 and scaling factor α=16, corresponding to 0.4% of the Qwen3-4B backbone parameters. The training set contains 6.2k context–query pairs derived from 2.5k unique contexts, with some contexts paired with multiple queries. Contexts c are drawn from LongAlpaca (Chen et al., 2024), PG-19 (Rae et al., 2020), and Tulu-3 Flan (Lambert et al., 2024; Longpre et al., 2023). Each context is paired with either a source-provided query or a query generated by the frozen teacher from a fixed set of generic prompts, such as summarization or explanation (Eyuboglu et al., 2026). Details of the training data are given in Appendix C.1. The training data have negligible measured overlap with the evaluation benchmarks, whose contexts are also separable from training contexts in a sentence-embedding analysis (Appendix C.2). During training, KV budget ratios are sampled from $\mathcal{U}(0.025, 0.25)$.

4.2 Main Results

Figure 3 reports performance across all four backbones and benchmarks, with the full KV cache shown as a dashed reference. Differences are modest at mild budgets, but the base eviction methods increasingly diverge from full-cache performance as the budget tightens, whereas RestoreKV remains substantially closer. On Qwen3-4B at r=0.05, RestoreKV improves KVzip from 38.2 to 73.2 on RULER-4K, while RestoreKV+ improves KVzip+ from 51.6 to 70.7 under the same total KV budget. Similar trends appear across the Qwen3 and Llama families and from 0.6B to 8B parameters, showing that the benefit is not confined to one model scale or architecture.

Table 1: **Applicability across KV cache eviction methods.** RestoreKV improves five different base methods on Qwen3-4B under matched KV budgets, with larger gains typically observed at tighter budgets. Each backbone–evictor configuration uses a separately trained generator. Colored subscripts indicate absolute changes from the corresponding baseline.

Method	RULER-4K			QASPER			QuALITY			LongHealth		
	r=0.2	r=0.1	r=0.05	r=0.2	r=0.1	r=0.05	r=0.2	r=0.1	r=0.05	r=0.2	r=0.1	r=0.05
KVzip	91.4	80.1	38.2	38.2	33.9	18.0	66.0	58.5	46.3	71.5	70.0	61.0
+ **Ours**	**93.5** +2.1	**88.8** +8.7	**73.2** +35.0	**39.0** +0.8	**37.7** +3.8	**28.7** +10.7	**67.0** +1.0	**63.3** +4.8	**52.4** +6.1	**75.8** +4.3	**73.8** +3.8	**67.5** +6.5
KVzip+	91.3	84.8	51.6	38.1	32.4	23.3	66.5	60.0	51.9	73.2	67.8	63.2
+ **Ours**	**93.3** +2.0	**89.7** +4.9	**70.7** +19.1	**40.9** +2.8	**38.5** +6.1	**31.6** +8.3	**66.8** +0.3	**62.9** +2.9	**57.9** +6.0	**74.2** +1.0	**73.0** +5.2	**71.5** +8.3
ContrastKV	91.6	80.7	38.0	40.1	31.9	17.8	66.2	62.2	50.7	74.2	69.0	49.0
+ **Ours**	**92.3** +0.7	**84.3** +3.6	**40.2** +2.2	**40.9** +0.8	**36.0** +4.1	**21.6** +3.8	**67.6** +1.4	**63.3** +1.1	**51.9** +1.2	**75.2** +1.0	**70.5** +1.5	**54.0** +5.0
SnapKV	29.7	19.2	12.8	22.4	19.0	14.3	53.9	**53.0**	42.5	41.0	36.8	36.2
+ **Ours**	**38.6** +8.9	**25.9** +6.7	**14.2** +1.4	**29.9** +7.5	**24.0** +5.0	**20.7** +6.4	**59.4** +5.5	51.8 −1.2	**49.9** +7.4	**51.8** +10.8	**41.8** +5.0	**38.5** +2.3
H_2O	8.0	3.5	3.2	22.0	13.8	14.4	59.2	50.4	41.5	60.2	48.8	37.0
+ **Ours**	**17.3** +9.3	**11.6** +8.1	**5.7** +2.5	**30.4** +8.4	**24.7** +10.9	**21.3** +6.9	**60.9** +1.7	**54.9** +4.5	**47.7** +6.2	**64.8** +4.6	**59.0** +10.2	**46.8** +9.8

Applicability across eviction methods. Table 1 evaluates RestoreKV on five KV cache eviction methods: KVzip, KVzip+, ContrastKV, and the query-agnostic variants of SnapKV and H_2O following the KVzip setting (Kim et al., 2025). Although these methods use different importance criteria and eviction procedures, RestoreKV leaves their scorers and eviction rules unchanged and simply replaces part of the selected cache with restore cache under the same total KV budget. RestoreKV improves the base method in most settings, with larger gains typically observed under tighter cache budgets. These results indicate that its effectiveness is not tied to a particular selection strategy and highlight RestoreKV as a broadly compatible plug-in for existing KV cache eviction pipelines.

Generalization to longer contexts. To examine whether the gains of RestoreKV persist as the context grows, we evaluate Llama-3.1-8B-Instruct on RULER at 4K, 32K, and 128K tokens (Table 2). RestoreKV improves KVzip at every length and budget. At r=0.05, the gains are +18.0, +15.9, and +18.5 points at 4K, 32K, and 128K, respectively. At r=0.1, the corresponding gains are +3.1, +5.4, and +8.2 points. RestoreKV is trained only on contexts of up to 15K tokens, yet its gains persist at 128K with a fixed-size ($n = 8$) restore cache.

Table 2: **RULER across context lengths** (Llama-3.1-8B-Instruct).

		r=0.1		r=0.05	
Length	**Full KV**	KVzip	**RestoreKV**	KVzip	**RestoreKV**
4K	95.5	90.3	**93.4** +3.1	63.1	**81.1** +18.0
32K	87.5	75.7	**81.1** +5.4	53.0	**68.9** +15.9
128K	76.8	49.1	**57.3** +8.2	25.8	**44.3** +18.5

Comparison with a Synthesized-Cache Baseline. Attention Matching (AM) (Zweiger et al., 2026) constructs a compact cache for each context by matching per-head attention behavior over reference queries. In contrast, RestoreKV amortizes restoration across contexts and generates only a small, context-conditioned complement in a single pre-eviction pass. As shown in Fig. 4, under the same Qwen3-4B setup and matched KV budgets, RestoreKV+ outperforms AM-fast on RULER-4K and at the tightest LongHealth budget, while AM-fast is stronger at a milder LongHealth budget. At r=0.05, RestoreKV+ achieves 70.7 versus 52.8 on RULER-4K and constructs a 4K-context cache in 0.74 s versus 9.68 s for AM-fast. Thus, RestoreKV+ provides competitive or better accuracy at approximately 13× lower construction time, without per-context query generation or fitting.

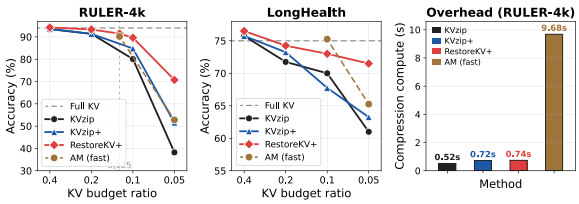

Figure 4: **Comparison with a synthesized-cache baseline** (Qwen3-4B). RestoreKV+ improves token-eviction baselines and remains competitive with AM-fast at ∼13× lower compression time on 4K contexts.

4.3 Analysis of RestoreKV

Source of restoration gains. Table 3 separates the roles of the restore-token embeddings and LoRA adaptation. All restore variants use n=8 and the same total KV budget. At r=0.05, learning only the restore embeddings reaches 42.1, whereas fixed embeddings with LoRA reach 71.9, recov-

Table 3: **Component and LoRA-target ablations on RULER-4K** (Qwen3-4B). The first row is KVzip without a restore cache; the last row is the full RestoreKV configuration. "Fixed" replaces all eight learned restore-token embeddings with the embedding of the line-break token \n.

Restore emb.	LoRA target	#Params	KV budget ratio r 0.2	0.1	0.05
–	–	–	91.4	80.1	38.2
Learned	–	0.02M (0.0005%)	92.7	82.6	42.1
Learned	$q/k/v$	4.0M (0.1%)	93.4	88.2	72.4
Learned	$q/k/v/o$	5.9M (0.15%)	93.5	88.4	72.0
Fixed	$q/k/v/o$+MLP	16.5M (0.4%)	93.5	88.0	71.9
Learned	$q/k/v/o$+MLP	16.5M (0.4%)	**93.5**	**88.8**	**73.2**

ering 96% of RestoreKV's full improvement to 73.2. Restricting LoRA to the $q/k/v$ projections achieves 72.4 with only 4.0M trainable parameters. These results indicate that restoration is driven primarily by attention-side adaptation that generates the context-conditioned restore cache, while learned embeddings and broader backbone adaptation provide only marginal additional gains.

Context-attention recovery. Figure 5 compares the layer-wise attention assigned by the final query token to non-sink context KV pairs. Under aggressive eviction, KVzip suppresses context attention across most layers and misses many of the high-attention peaks observed with the full cache, reducing the average mass from 2.73% to 0.56% and leaving 7 of 36 layers below 0.1%. RestoreKV recovers part of this lost attention across layers, as highlighted by the shaded region, increasing the average mass to 1.30% and raising every layer above 0.1%. Its layer-wise profile also more closely follows the full-cache pattern, reducing KL divergence from 0.198 to 0.167. Consistent with this recovery, the KL divergence of the final predictive distribution from the full-cache model decreases from 7.3 to 3.8. These results indicate that RestoreKV partially restores both the amount and the layer-wise allocation of context attention, accompanied by predictions closer to those of the full-cache model.

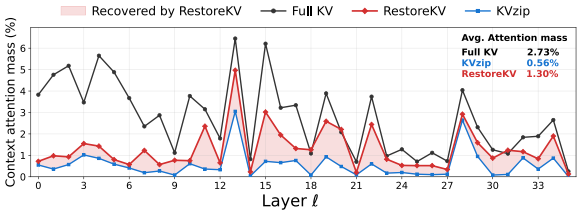

Figure 5: **Layer-wise context attention** (RULER-4K, Qwen3-4B, r=0.05). RestoreKV recovers part of the attention lost under KVzip, raising the average mass from 0.56% to 1.30%.

4.4 Ablations

Number of restore tokens. Table 4 examines the effect of the number of restore tokens n. All variants share the same training recipe and total KV budget. Even a single restore token yields a substantial improvement under aggressive eviction, increasing accuracy from 38.2 to 65.3 at r=0.05. Despite minor fluctuations, performance generally improves as more restore tokens are added up to n=8, which achieves the best accuracy of 73.2. All configurations with restore tokens consistently outperform the no-restore baseline, while increasing n beyond eight provides no further gain. We therefore use n=8 as the default configuration.

Table 4: **Number of restore tokens** (RULER-4K, Qwen3-4B).

n	KV budget ratio r 0.2	0.1	0.05
0 (KVzip)	91.4	80.1	38.2
1	93.0	85.3	65.3
2	93.4	87.6	70.4
4	93.4	87.0	68.5
8 (default)	**93.5**	**88.8**	**73.2**
16	93.3	87.8	69.1

Effect of full-context conditioning. Table 5 compares generating the restore cache before eviction with generating it from the already evicted cache. At r=0.05, the evicted-cache variant improves KVzip from 38.2 to 64.4, indicating that much of the recovery comes from the learned restore transformation rather than full-context access alone. Conditioning on the full cache further improves accuracy to 73.2, providing an additional 8.8 points. Because the two variants use the same restore tokens, adaptation, KV budget, and positional offset, this gap isolates the benefit of

Table 5: **Effect of full-context conditioning** (RULER-4K, Qwen3-4B).

Method	KV budget ratio r 0.2	0.1	0.05
KVzip	91.4	80.1	38.2
From evicted cache	**93.7**	87.6	64.4
From full cache	93.5	**88.8**	**73.2**

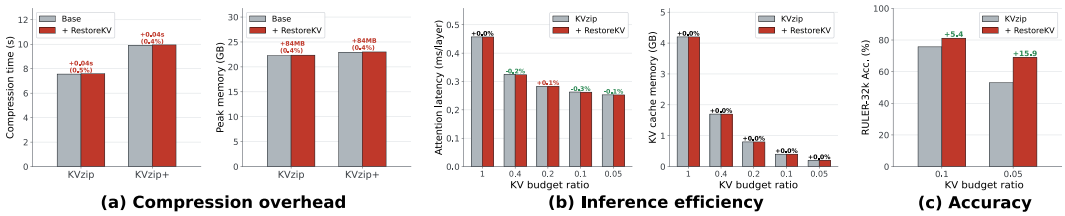

Figure 6: **Efficiency analysis** (Llama-3.1-8B-Instruct, 32K context). (a) Restore-cache generation adds only $\sim$0.04 s ($\leq$0.5%) and 84 MB (0.4%) on top of the base scorer. (b) Attention latency and KV memory are nearly identical to the base method. (c) RestoreKV improves the base method at tight KV budget.

full-context conditioning. Together with Table 3, these results suggest that effective restoration is driven primarily by the learned attention-side transformation, while access to the complete context before eviction provides a substantial complementary benefit.

4.5 Efficiency Analysis

Training cost. RestoreKV is trained only once offline on contexts of up to 15K tokens, and the resulting adapters are reused across all contexts, tasks, and KV budgets without further tuning. With KVzip as the base eviction method, this one-time training takes about 1.5 hours for Llama-3.1-8B-Instruct on a single RTX PRO 6000 GPU (Table 6). Only the restore embeddings and a rank-8 LoRA on the attention and MLP projections are updated, while the long-context prefill and KVzip importance scoring run under `no_grad`, exactly as in deployment. As a result, the training peak memory is on par with KVzip-only inference at 15K tokens, so training fits on the same GPU used for deployment.

Table 6: **Training cost on a single RTX PRO 6000.** Δ: peak memory over KVzip inference at 15K tokens.

Model	Time	Peak mem.	Δ vs. inf.
Llama-3.1-8B	$\sim$1.5 h	22.9 GB	0.1 GB

Inference cost. Figure 6 evaluates RestoreKV on Llama-3.1-8B-Instruct with RULER-32K using an RTX PRO 6000. Constructing the eight-token restore cache before eviction incurs only 0.03–0.04s of additional latency, accounting for at most 0.5% of the end-to-end compression time, including prefill. It also increases peak memory during prefill by only 84 MB (0.4%), due to the resident LoRA weights, whose memory cost is constant with respect to context length (Fig. 6a). This small overhead arises because the restore pass is a single forward pass over eight additional positions after the context has already been prefilled. During decoding, RestoreKV replaces an equal number of context KV pairs and therefore preserves the same total cache budget. Across KV budget ratios from 1.0 to 0.05, its per-layer attention latency is nearly identical to KVzip (0.457–0.253 ms), while KV-cache memory is exactly matched, decreasing from 4.20 to 0.20 GB for both methods (Fig. 6b). Under these identical inference costs, RestoreKV improves the accuracy of KVzip at compressed budgets (Fig. 6c). Thus, RestoreKV improves accuracy at 32K while introducing negligible one-time compression overhead and no measurable additional inference-time cost.

5 Conclusion

We revisited query-agnostic KV cache eviction from a complementary restoration perspective. RestoreKV uses a shared, LoRA-adapted restoration mechanism to generate a compact, context-conditioned restore cache before eviction and combines it with retained original KV pairs under the same total budget. It preserves the base importance scorer and eviction rule, and disables the adapters after cache construction, adding negligible one-time overhead and no query-time KV-memory or decoding cost. Across four backbones, four benchmarks, and five base eviction methods, RestoreKV reduces compression-induced degradation in nearly all settings, with its largest gains under aggressive budgets. Our analyses show that attention-side adaptation is the primary source of recovery, while full-context conditioning provides an additional complementary benefit. A limitation is that the generator is trained per backbone–evictor configuration, with limited transfer across evictors under aggressive compression (Appendix A.4). In practice, it can be trained once for the evictor used in deployment, such as the strongest available one, at a one-time cost of a few GPU hours.

AI USE STATEMENT

We used generative AI tools to implement the proposed method and the experiment code (i.e., as coding assistants), to search for and identify relevant prior work, and to assist with translating drafts of this paper into English. We did not use them to propose or refine hypotheses, to design the research methodology or experiments, to develop the conceptual framework or theoretical models underlying the method, to interpret results, to support qualitative or thematic data analysis, to generate synthetic datasets (the answers used for self-distillation are produced by the frozen backbone itself as part of the proposed method; Appendix C.1), or to clean or reformat datasets. The remaining tasks requiring disclosure — formulating mathematical claims, providing critical ingredients for proving them, and writing proofs — are not applicable to this work, which contains no formal theoretical claims. In writing, we used generative AI tools to edit and polish text and to propose the title and keywords; we did not use them to draft sections of the paper. The authors verified and tested all AI-assisted code, reviewed all AI-assisted text, including translations, and checked every AI-surfaced reference against its original source. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We describe RestoreKV in Section 3 and provide the full training configuration in Appendix C.3, the construction of the training data in Appendix C.1, the data contamination analysis in Appendix C.2, the query-agnostic adaptations of the baselines in Appendix C.4, the evaluation metrics in Appendix C.6, and the hardware and software environment in Appendix C.5. Inference is deterministic, and results across three training seeds are reported in Appendix C.7. An anonymized version of our code is included in the supplementary material, and we will release the code and trained checkpoints upon publication.

REFERENCES

L Adams, F Busch, T Han, JB Excoffier, M Ortala, A Löser, HJWL Aerts, JN Kather, D Truhn, and K Bressem. Longhealth: A question answering benchmark with long clinical documents. *Journal of Healthcare Informatics Research*, 9(3):280–296, 2025.

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. LookaheadKV: Fast and accurate KV cache eviction by glimpsing into the future without generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RVLMGPXt2i.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Tyler A. Chang, Kishaloy Halder, Neha Anna John, Yogarshi Vyas, Yassine Benajiba, Miguel Ballesteros, and Dan Roth. Characterizing and measuring linguistic dataset drift. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8953–8967, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.498. URL https://aclanthology.org/2023.acl-long.498/.

Vivek Chari, Guanghui Qin, and Benjamin Van Durme. Kv-distill: Nearly lossless learnable context compression for llms. *arXiv preprint arXiv:2503.10337*, 2025.

Xingchi Chen, Peiyuan Zong, Ziqiang Gao, Qing Li, Yong Jiang, Fa Zhu, and Hui Li. ContrastKV: Robust KV cache eviction via contrastive signal fusion for multi-query generalization. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9216–9229, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.417. URL https://aclanthology.org/2026.acl-long.417/.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6PmJoRfdaK.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021.

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

Hady Elsahar and Matthias Gallé. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2163–2173, 2019.

Sabri Eyuboglu, Ryan Saul Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Ruoyu Liu, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=0k5w8O0SNg.

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. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yuzhe Gu, Xiyu Liang, Jiaojiao Zhao, and Enmao Diao. OBCache: Optimal brain KV cache pruning for efficient long-context LLM inference. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=f58VZpG6ls.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

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://openreview.net/forum?id=nZeVKeeFYf9.

Simon Jegou and Maximilian Jeblick. Kvzap: Fast, adaptive, and faithful kv cache pruning. *arXiv preprint arXiv:2601.07891*, 2026.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. Kvzip: Query-agnostic kv cache compression with context reconstruction. *Advances in Neural Information Processing Systems*, 38:167563–167591, 2025.

Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast kvzip: Efficient and accurate llm inference with gated kv eviction. *arXiv preprint arXiv:2601.17668*, 2026.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. Scbench: A kv cache-centric analysis of long-context methods. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 66063–66093, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a540b17fb2295c736d5afd6c507acf66-Paper-Conference.pdf.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Zeju Li, Yizhou Zhou, and Qiang Xu. Latent context compilation: Distilling long context into compact portable memory. *arXiv preprint arXiv:2602.21221*, 2026.

Yuping Lin, Jiayuan Ding, Yue Xing, Pengfei He, Jiliang Tang, and Subhabrata Mukherjee. A simple plug-in for improving eviction-based kv cache compression. *arXiv preprint arXiv:2605.23258*, 2026.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. *Advances in Neural Information Processing Systems*, 38:28728–28778, 2025.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *International conference on machine learning*, pp. 22631–22648. PMLR, 2023.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. *arXiv preprint arXiv:1610.06545*, 2016.

Charles O'Neill, Alex Sandomirsky, Harry Partridge, Mudith Jayasekara, and Max Kirkby. Still: Amortized kv cache compaction in a single forward pass. *arXiv preprint arXiv:2606.07878*, 2026.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022.

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:5983–6019, 2025.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

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

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads. In *International Conference on Learning Representations*, volume 2025, pp. 16632–16646, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. doi: 10.52202/075280-1506. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast kv compaction via attention matching. In *Forty-third International Conference on Machine Learning*, 2026.

APPENDIX OVERVIEW

This appendix provides additional experimental results, control experiments, and implementation details supporting the main paper.

- **A Additional Experimental Results**
  - A.1 Applicability to a Learning-Based Scorer
  - A.2 Effect of the Training Ratio Range
  - A.3 Effect of the Query-Position Offset
  - A.4 Effect of Training–Inference Evictor Mismatch
  - A.5 Comparison with Plug-in Methods
- **B Additional Long-Context Benchmark Results**
  - B.1 SCBench Results
  - B.2 LongBench Results
- **C Implementation and Reproducibility Details**
  - C.1 Training Data Construction
  - C.2 Training–Evaluation Data Overlap
  - C.3 Training Setup Details
  - C.4 Baseline Adaptations (SnapKV and H_2O)
  - C.5 Experimental Environment
  - C.6 Evaluation Metrics
  - C.7 Sensitivity to Training Seeds
  - C.8 Training Set Examples

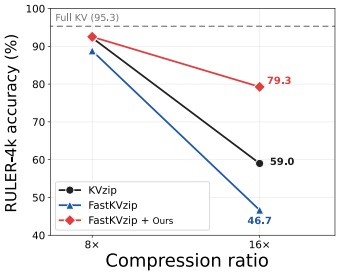

Figure A: **RestoreKV with Fast KVzip** (RULER-4K, Qwen3-8B). At $16\times$, it improves Fast KVzip from 46.7 to 79.3.

Table A: **Ablation on the training ratio range (RULER-4K, Qwen3-4B).** RestoreKV is trained by sampling the retention ratio from $\mathcal{U}(r_{\min}, r_{\max})$. The first row reports the KVzip baseline without restoration.

Training ratio range		KV budget ratio r			
$r_{\min}$	$r_{\max}$	0.4	0.2	0.1	0.05
-	-	93.5	91.4	80.1	38.2
0.025	0.25	**94.3**	**93.5**	**88.8**	**73.2**
0.025	0.50	94.1	93.1	86.1	63.5
0.050	0.25	**94.3**	93.2	86.5	65.6
0.050	0.50	94.0	92.8	84.5	51.5

A Additional Experimental Results

A.1 Applicability to a Learning-Based Scorer

RestoreKV is also compatible with learned importance scorers. Fast KVzip (Kim et al., 2026) replaces KVzip's expensive scoring pass with lightweight per-layer gates that predict importance from hidden states in a single forward pass. Attaching RestoreKV raises its RULER-4K (Hsieh et al., 2024) accuracy from 46.7 to 79.3 at $16\times$ compression (Fig. A), a 32.6-point gain that also exceeds standard KVzip (59.0). Because Fast KVzip retains single-pass scoring, this result shows that RestoreKV can substantially improve a learned scorer without sacrificing its primary efficiency advantage.

A.2 Effect of the Training Ratio Range

Table A examines the effect of the training range used to sample the KV budget ratio, $r \sim \mathcal{U}(r_{\min}, r_{\max})$, on RULER-4K. All configurations achieve comparable performance at the mild evaluation budget of r=0.4. However, the differences become increasingly pronounced as the budget tightens. The default range, $\mathcal{U}(0.025, 0.25)$, achieves the strongest overall performance, reaching 88.8 at r=0.1 and 73.2 at r=0.05. Increasing either the lower bound or the upper bound reduces performance under aggressive eviction. These results indicate that training with sufficient exposure to tight KV budgets is important for effective restoration under severe compression. We therefore use $\mathcal{U}(0.025, 0.25)$ as the default training range.

A.3 Effect of the Query-Position Offset

RestoreKV appends 8 restore tokens after the context, causing subsequent queries to begin 8 positions later than in standard KVzip. This changes the relative RoPE (Su et al., 2024) offsets between the query and the retained context KV pairs, raising the possibility that the position shift itself contributes to the observed improvement. To isolate this effect, we evaluate a control variant of KVzip that uses no restore tokens or restore cache but shifts the query positions by the same $+8$ offset. As shown in Table B, this variant achieves 36.9 at r=0.05, comparable to standard KVzip at 38.2 and substantially below RestoreKV at 73.2. These results indicate that the performance gain cannot be explained by the RoPE offset alone and instead support the contribution of the context-conditioned restore cache.

Table B: **Effect of the query-position offset (RULER-4K, Qwen3-4B).** The same $+8$ offset without restore states stays close to KVzip.

Variant	Restore states	Query offset	KV budget ratio r		
			0.2	0.1	0.05
KVzip	No	0	91.4	80.1	38.2
Offset-only KVzip	No	$+8$	90.5	80.8	36.9
RestoreKV	Yes	$+8$	**93.5**	**88.8**	**73.2**

A.4 Effect of Training–Inference Evictor Mismatch

A central premise of RestoreKV is that its learned restore cache should complement the states retained by the base evictor. We examine whether this mechanism depends on the evictor used during training by fixing the inference evictor to KVzip and comparing checkpoints trained with KVzip and SnapKV (Li et al., 2024) (Table C). Both checkpoints score 93.5 at r=0.20, but their performance diverges as the budget tightens. At r=0.10, the matched checkpoint scores 88.8 versus 81.6 for the mismatched checkpoint; at r=0.05, the gap widens to 73.2 versus 42.0. The mismatched checkpoint still exceeds the KVzip baseline at r=0.05 (42.0 versus 38.2). These results suggest that restoration adapts to the compression behavior of the training-time evictor, while transfer across evictors is limited under aggressive compression.

Table C: **Training–inference evictor mismatch** (RULER-4K, Qwen3-4B).

Training evictor	Inference evictor	KV budget ratio r		
		0.2	0.1	0.05
–	KVzip	91.4	80.1	38.2
SnapKV	KVzip	**93.5**	81.6	42.0
KVzip	KVzip	**93.5**	**88.8**	**73.2**

A.5 Comparison with Plug-in Methods

We compare RestoreKV with two plug-in methods that also operate on top of an existing evictor: OBCache (Gu et al., 2026), which refines importance scores based on output perturbation, and VECTOR (Lin et al., 2026), which reconstructs the values of evicted entries during generation. All methods use the same base evictor and total KV budget (Table D). On KVzip, OBCache degrades accuracy at all budgets, and VECTOR improves it at tight budgets but falls below KVzip at r=0.2. In contrast, RestoreKV improves KVzip at every budget and achieves the best accuracy throughout, reaching 73.2 at r=0.05. On SnapKV, whose base accuracy is already low, all plug-in methods perform comparably, with OBCache slightly ahead at r=0.2 and r=0.05.

Table D: **Comparison with plug-in methods** (RULER-4K, Qwen3-4B).

Method	KV budget ratio r		
	0.2	0.1	0.05
KVzip	91.4	80.1	38.2
+ OBCache-k	87.5	53.6	25.6
+ VECTOR	88.7	86.8	63.6
+ Ours	**93.5**	**88.8**	**73.2**
SnapKV	29.7	19.2	12.8
+ OBCache-k	**40.5**	25.3	**17.1**
+ VECTOR	33.1	24.0	15.2
+ Ours	38.6	**25.9**	14.2

B Additional Long-context Benchmark Results

B.1 SCBench Results

To evaluate generalization to substantially longer contexts, we additionally consider nine SCBench (Li et al., 2025) tasks with an average context length of approximately 104K tokens (Qwen3 tokenizer), following the evaluation protocol of KVzip (Fig. B). These tasks include Retr.KV, Retr.Prefix-Suffix, and Retr.MultiHop for string retrieval; Code.RepoQA, En.QA, and En.MultiChoice for semantic retrieval; and Math.Find, ICL.ManyShot, and En.Sum for global-context understanding. Focusing on the average scores, RestoreKV consistently improves KVzip across all four KV budget ratios, raising the average from 29.4 to 29.7 at r=0.4, from 31.6 to 33.1 at r=0.2, from 30.0 to 32.4 at r=0.1, and from 24.1 to 25.6 at r=0.05. Notably, RestoreKV is trained only on contexts up to 15K tokens (Table F), so these gains reflect generalization to context lengths roughly an order of magnitude beyond those seen during training. These results suggest that the benefit of the restore cache is not limited to short contexts and can extend to longer-context settings.

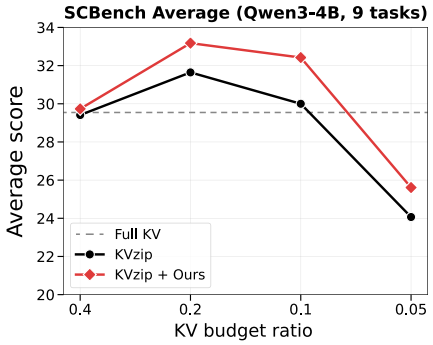

Figure B: **SCBench results** (Qwen3-4B).

Table E: **LongBench results over 16 tasks under aggressive KV compression.** Categories are single-document QA (S-QA), multi-document QA (M-QA), summarization (Summ), few-shot learning (Fewshot), synthetic tasks (Synth), and code tasks (Code). *Rel.* denotes the average score relative to the full KV cache.

Method	S-QA	M-QA	Summ	Fewshot	Synth	Code	Avg	Rel.
Full KV cache								
Llama-3.1-8B-Instruct	44.6	47.6	29.2	54.0	55.1	48.1	46.4	100.0%
KV budget ratio = 0.125								
KVzip	42.7	**43.9**	27.7	61.9	44.0	47.7	44.7	96.3%
KVzip + Ours	**43.6**	43.0	**28.5**	**63.9**	**45.9**	**49.3**	**45.7**	**98.4%**
KV budget ratio = 0.0625								
KVzip	33.3	32.4	24.0	49.1	20.6	41.6	33.5	72.1%
KVzip + Ours	**38.2**	**35.4**	**27.0**	**59.3**	**22.8**	**43.2**	**37.7**	**81.2%**

(a) Llama-3.1-8B-Instruct

Method	S-QA	M-QA	Summ	Fewshot	Synth	Code	Avg	Rel.
Full KV cache								
Qwen3-8B	42.8	48.6	27.6	57.4	50.5	62.6	48.2	100.0%
KV budget ratio = 0.125								
KVzip	38.4	**44.0**	26.2	63.1	**42.4**	56.8	45.2	93.6%
KVzip + Ours	**39.2**	42.6	**26.9**	**64.1**	39.9	**59.9**	**45.4**	**94.2%**
KV budget ratio = 0.0625								
KVzip	23.2	15.5	20.9	50.8	14.3	46.9	28.6	59.3%
KVzip + Ours	**28.1**	**17.7**	**23.8**	**55.0**	**15.0**	**53.0**	**32.1**	**66.5%**

(b) Qwen3-8B

Table F: **RestoreKV training data statistics** (context length in tokens).

Source	Samples	Ctx (mean)	Ctx range
LongAlpaca	2,488	7,306	4,961–15,237
PG-19	2,260	3,071	3,071–3,072
Tulu-3 FLAN	1,500	898	512–9,283
Total	**6,248**	4,236	512–15,237

B.2 LongBench Results

We further evaluate RestoreKV on all 16 LongBench (Bai et al., 2024) tasks following the KVPress protocol. The evaluation comprises 3,750 examples with context lengths ranging from approximately 5K to 15K tokens (Qwen3 tokenizer). Table E reports results aggregated by task category. RestoreKV improves the overall average across both backbones and both cache budgets, with larger gains at the tighter $r{=}0.0625$ budget. At this ratio, the average score increases from 33.5 to 37.7 on Llama-3.1-8B-Instruct and from 28.6 to 32.1 on Qwen3-8B. These results extend the main-paper findings to a broader task suite, with the clearest gains under aggressive KV eviction.

C Implementation and Reproducibility Details

C.1 Training Data Construction

We construct the training set from three sources (Table F). *(i) LongAlpaca:* We select 500 LongAlpaca (Chen et al., 2024) documents containing 2,048–16,000 tokens (Qwen3 tokenizer). We apply the same fixed set of five generic question prompts to each document, covering factual, summarization, multi-hop, method, and comparison types (Eyuboglu et al., 2026). For each prompt, the frozen teacher writes a query and then answers it greedily over the document (≤ 512 tokens). After filtering invalid or failed generations, this yields 2,488 examples. *(ii) PG-19:* We split 50 PG-19 (Rae et al., 2020) books into 3,072-token chunks, using up to 10 chunks per book. We apply the same five prompts to each chunk and generate teacher questions and answers using the same procedure, yielding 2,260 examples after filtering. *(iii) Tulu-3 FLAN few-shot:* We use 1,500 filtered examples from the FLAN v2 subset of `allenai/tulu-3-sft-mixture` on Hugging Face (Lambert et al., 2024; Longpre et al., 2023), paired with teacher-generated responses. The three sources form a training mixture of 6,248 examples. For computational efficiency, teacher responses are generated offline once for each target model and reused throughout training, rather than regenerated at every optimization step.

C.2 Training–Evaluation Data Overlap

Text overlap. Because the training sources in Appendix C.1 partly share domains with the evaluation benchmarks, we examine whether the gains of RestoreKV could stem from data overlap. We measure surface overlap using 13-gram matching (Brown et al., 2020), defined as the fraction of test 13-grams found in the training data, and near-duplication using sentence-embedding similarity (Reimers & Gurevych, 2019). As shown in Table G, overlap is negligible across all four main benchmarks, including pairs drawn from related sources such as QuALITY and PG-19 or QASPER and LongAlpaca. In addition, the backbone remains frozen and no benchmark labels are used during training. Any contamination from pretraining is therefore shared equally by the full-cache model, KVzip, and RestoreKV, and is unlikely to account for the improvement of RestoreKV over KVzip.

Table G: **Overlap between training data and evaluation benchmarks.** Near-dup.: test documents with embedding cosine similarity > 0.9 to any training document.

Benchmark	13-gram	Near-dup.
RULER	$\leq$0.0069	1/1000
QuALITY	$<$0.0001	0/230
QASPER	$<$0.0001	0/281
LongHealth	0.0000	0/20

Distribution shift. Beyond text overlap, we also examine whether training and evaluation data differ in distribution (Chang et al., 2023). We train a logistic-regression classifier on frozen sentence embeddings (Reimers & Gurevych, 2019) to distinguish their source (Lopez-Paz & Oquab, 2016; Elsahar & Gallé, 2019) (balanced classes, 5-fold ROC-AUC). The classifier cannot distinguish two random halves of the training set (AUROC 0.51), but separates training *contexts* from each evaluation benchmark (AUROC 0.95–1.00; Table H). The same holds for the *queries*: the training queries are equally separable from each benchmark's queries (AUROC 0.98–1.00). Thus neither the contexts nor the queries seen during training match those at evaluation, yet RestoreKV's gains persist across all benchmarks under this measured shift.

Table H: **Distribution shift between training and evaluation data.** Domain-classifier AUROC on contexts and queries, where 0.5 means indistinguishable and 1.0 fully separable. Δ is the gain of RestoreKV over KVzip at r=0.05 (Qwen3-4B).

		AUROC		
Train	**Eval**	Ctx	Query	**Δ**
Train	Train	0.51	0.50	—
Train	QuALITY	0.95	0.98	+6.1
	QASPER	0.98	0.98	+10.7
	LongHealth	1.00	1.00	+6.5
	RULER	0.97	1.00	+35.0

C.3 Training Setup Details

We keep the base model frozen and train only the n=8 restore-token embeddings and LoRA adapters (r_{LoRA}=8, α=16, dropout 0) on the attention and MLP projections ($\sim$16.5M parameters, 0.4% of a 4B model). Optimization uses AdamW (β=$(0.9, 0.999)$, weight decay 0.01) with learning rate 2×10^{-4} for the restore tokens and LoRA, a cosine schedule with 50 warmup steps, gradient clipping at norm 1.0, and a batch of one context per step for 5,000 steps. The objective is a symmetric KL distillation loss between the answer-token distributions of the restored-cache student using the final budget-matched cache and the frozen full-cache teacher. During training the KV ratio is sampled from $\mathcal{U}(0.025, 0.25)$ to target extreme-low budgets.

C.4 Baseline Adaptations (SnapKV and H_2O).

We adopt the query-agnostic, prefill-based adaptations of SnapKV (Li et al., 2024) and H_2O (Zhang et al., 2023) used in the KVzip evaluation setup (Kim et al., 2025). In all cases, the context cache is scored and compressed before future queries are observed, and KV importance is evaluated at the individual KV-pair level under the same budget-allocation framework.

SnapKV We use the final w=32 context positions as the observation window and compute their attention over the context keys. For each KV pair, we average attention across the observation-window queries and apply max-pool smoothing with a kernel size of 7. The trailing observation window is always retained. Because the observation window is drawn entirely from the context rather than from a future task query, the resulting cache is query-agnostic.

Table I: **Experimental environment.** Hardware and software configurations used for all experiments.

Hardware	
GPU	NVIDIA RTX PRO 6000 Blackwell (96 GB)
CPU	Intel Xeon Gold 6530
Memory	503 GiB
Software	
OS	Ubuntu 22.04.5 LTS
Framework	PyTorch 2.8.0 (CUDA 12.8)

Table J: **Evaluation metrics used for each benchmark.** We follow the official metric of each benchmark for comparability with prior work.

Benchmark	**Evaluation metric**
RULER	Official string-matching score
QASPER	Maximum token-F1 over reference annotations
QuALITY	Answer-letter accuracy
LongHealth	Answer-letter accuracy
LongBench	Official per-task metric (F1, ROUGE-L, or accuracy)
SCBench	Official task-specific evaluator

Table K: **Sensitivity to Training Seeds.** RULER-4K accuracy on Qwen3-4B, reported as mean±std over three training seeds.

	KV budget ratio			
	0.40	0.20	0.10	0.05
KVzip	93.46	91.38	80.08	38.23
RestoreKV	**94.25** ± 0.04	**93.37** ± 0.11	**88.21** ± 0.51	**72.81** ± 0.57

H_2O We use the prefill-based H_2O adaptation described in KVzip. For each KV pair, its importance is defined as the maximum attention it receives over all causal queries during context prefill. We use maximum rather than mean aggregation, following the finding in KVzip that maximum attention provides better compression performance. This baseline therefore derives importance from prefill self-attention, in contrast to the context-reconstruction attention used by KVzip.

C.5 Experimental Environment

Our experimental environment is summarized in Table I. For the comparison against KVPress Benchmark (Devoto et al., 2025) entries (RULER and LongBench) we implemented our method within the official KVPress codebase and ran those evaluations in its environment, ensuring a fair comparison against published baselines.

C.6 Evaluation Metrics

We follow the official evaluation metric of each benchmark to ensure consistency with its standard evaluation protocol and comparability with prior work. RULER and LongBench are evaluated following the KVPress protocol, while SCBench uses its official task-specific evaluators. QASPER uses the maximum token-level F1 over reference annotations, whereas QuALITY and LongHealth use answer-letter accuracy. Table J summarizes the metrics.

C.7 Sensitivity to Training Seeds

Inference in RestoreKV is deterministic: importance scoring involves no sampling, and decoding is greedy. The remaining stochasticity comes from training, including random initialization and retention-ratio sampling. All main results report the seed-0 run. To assess training robustness, we additionally train RestoreKV with two different seeds and evaluate the three runs on RULER-4K. As shown in Table K, the standard deviation remains below 0.57 across all cache ratios. At $r{=}0.05$, this variation is approximately $60\times$ smaller than the 34.6-point mean improvement over KVzip.

C.8 Training Set Examples

We show representative examples from each of the three sources in our training set (Appendix C.1).

Context: *"Below is a paper. Memorize the paper and answer my question after the paper. The paper begins.*
Abstract. To build Video Question Answering (VideoQA) systems capable of assisting humans in daily activities, seeking answers from long-form videos with diverse and complex events is a must. Existing multi-modal VQA models achieve promising performance on images or short video clips, especially with the recent success of large-scale multi-modal pre-training. However, when extending these methods to long-form videos, new challenges arise: on the one hand, a dense sampling strategy is computationally prohibitive; on the other hand, methods relying on sparse sampling struggle in scenarios where multi-event and multi-granularity visual reasoning are required. In this work, we introduce a new model named Multi-modal Iterative Spatial-temporal Transformer (MIST) to adapt pre-trained models for long-form VideoQA…"

Generated queries:

- *(factual)* "What is the name of the new model introduced in the paper for long-form VideoQA?"
- *(summary)* "What is the main contribution of the MIST model for long-form VideoQA?"
- *(multi-hop)* "What is the main advantage of MIST in handling long-form videos for VideoQA?"
- *(method)* "What specific method performs top-k selection in the iterative spatial-temporal attention layer?"
- *(comparison)* "How does MIST differ from traditional dense spatial-temporal self-attention?"

Figure C: Training set example from LongAlpaca.

Context: *"… For now I will stretch out my hand, that I may smite thee and thy people with pestilence; and thou shalt be cut off from the earth. And in very deed for this cause have I raised thee up, for to shew in thee my power; and that my name may be declared throughout all the earth. As yet exaltest thou thyself against my people, that thou wilt not let them go? Behold, to morrow about this time I will cause it to rain a very grievous hail, such as hath not been in Egypt since the foundation thereof even until now…"* (the plagues of Egypt)

Generated queries:

- *(factual)* "What was the name of the month declared the beginning of months and the first month of the year?"
- *(summary)* "What is the main point of the text regarding the plagues inflicted upon Egypt?"
- *(multi-hop)* "What was the outcome of Pharaoh's request to let the Israelites take their cattle and servants?"
- *(method)* "What specific method did Moses use to cause hail to fall upon Egypt?"
- *(comparison)* "How does the plagues being multiplied contrast with Pharaoh's heart being hardened?"

Figure D: Training set example from PG-19.

Context (the task instruction is embedded in the context and used directly as the query): *"Student A: Halloween is an upcoming American slasher film directed by David Gordon Green and written by Green, Jeff Fradley, and Danny McBride. It is the eleventh installment in the Halloween franchise, and is a direct sequel to Halloween, while disregarding the continuity of the previous sequels… Jamie Lee Curtis and Nick Castle reprise their roles as Laurie Strode and Michael Myers, respectively… Based on this passage, who's playing michael myers in halloween 2018? Student B: OK… The relevant information to answer the above question is: Jamie Lee Curtis and Nick Castle reprise their roles…"*

Figure E: Training set example from Tulu-3 FLAN.