

KV-Retriever: Aligning KV Cache Eviction with Downstream Task Loss

Wenhao Gao^{1,†}, Binqin Shi^{2,†}, Wei Jia³, Aiwei Liu³
Linhao Zhang³, Caixia Yuan¹, Xiaojie Wang^{1,*}, Xiao Zhou^{3,*}

¹Beijing University of Posts and Telecommunications, China

²Xi'an Jiaotong University, China

³Basic Model Technology Center, WeChat AI, Tencent Inc., China

{whgao, yuancx, xjwang}@bupt.edu.cn

{jarviswjia, coveliu, neuronzhang, chappyzhou}@tencent.com

binqinshi@stu.xjtu.edu.cn

Abstract

The prohibitive memory footprint of the Key-Value (KV) cache imposes a critical bottleneck for efficient long-context LLM serving. Current eviction methods, ranging from attention-based heuristics to recent learnable approaches, rely on scoring criteria decoupled from the downstream task performance that ultimately matters. To address this, we introduce **KV-Retriever**, a layer-wise learnable importance scoring module directly optimized by the end-to-end downstream task loss. We demonstrate that aligning the scoring objective with the actual task signal—rather than intermediate proxies such as attention statistics or oracle labels—yields a fundamentally more accurate notion of token importance. Functioning as a lightweight, plug-and-play module with only $\sim 0.01\%$ additional parameters on a frozen 8B backbone, KV-Retriever is compatible with diverse host eviction frameworks. Extensive experiments show that, for each backbone, one trained retriever can be reused unchanged across multiple KV cache compression architectures and delivers consistent improvements on LongBench, LongBench-v2, and RULER. KV-Retriever requires one-time supervised training with golden answers; “plug-and-play” refers to deployment into otherwise unchanged host eviction frameworks, not to training-free adaptation. Our code is available at https://github.com/Taylor-Gavel/kv_retriever.

1 Introduction

While Large Language Models (LLMs) demonstrate robust reasoning capabilities in long-context scenarios—such as long-document QA, multi-turn dialogue, and codebase analysis (Bai et al., 2024; An et al., 2024; Liu et al., 2024), their deployment efficiency is fundamentally constrained by

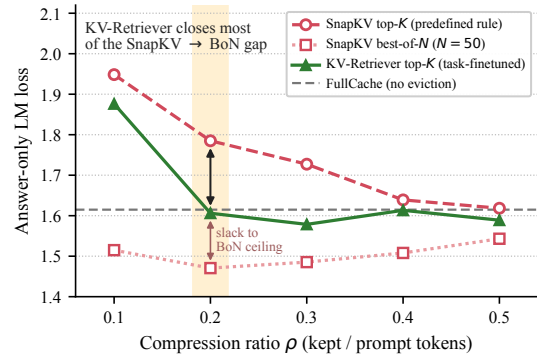


Figure 1: **SnapKV’s deterministic scoring leaves a substantial gap below the compression upper bound; KV-Retriever closes most of it in a single forward.** Answer-only LM loss (lower is better) vs. compression ratio ρ . Four operating points: SnapKV top- K (predefined rule), SnapKV best-of- N ($N=50$; approximate upper bound), KV-Retriever (top- K), and FullCache (no eviction). At $\rho=0.2$ (shaded), the large arrow marks the gap closed by KV-Retriever; the small arrow is the residual to the BoN ceiling.

the quadratic scaling overhead of KV cache relative to sequence length (Zhang et al., 2023; Liu et al., 2023). To address this challenge, KV cache eviction has emerged as a primary compression paradigm (Li et al., 2024; Li et al., 2025). By retaining critical KV entries and discarding redundant ones within a fixed memory budget, these methods aim to preserve the downstream task quality achievable under full attention. The efficacy of such approaches hinges on the definition of “importance”—specifically, the mechanism for assigning a scalar score to each KV token to facilitate filtering.

Existing approaches to scoring KV token importance fall into two broad families. Predefined-heuristic methods derive scores directly from attention statistics—cumulative attention sums, query-window-aggregated attention, or attention reweighted by output-projection norms (Zhang

*Corresponding authors.

[†]Work done while interning at WeChat AI.

et al., 2023; Liu et al., 2023; Li et al., 2024)—without any learning component. A more recent line introduces learnable modules that predict token retention value, yet trains them with intermediate proxies or constrained gating objectives rather than directly ranking sampled keep sets by downstream task loss: future attention patterns, utility rewards, retaining-score regression, or binary keep-or-discard gates (Moschella et al., 2026; Dong et al., 2026; Liu et al., 2026; Huang et al., 2025; Zeng et al., 2024). Despite their structural differences, both families share a defining property: the scoring criterion is not directly optimized by the downstream keep-set task loss used in our method.

This decoupling creates a systematic gap between the keep set a scorer produces and the keep set that would actually minimize task error. To quantify this gap concretely, we take SnapKV (Li et al., 2024) as a representative and construct an approximate upper bound on compression quality: draw $N=50$ keep sets from SnapKV’s attention-score-induced distribution and record the minimum answer-only LM loss (best-of- N ; BoN). As Figure 1 shows, this upper bound sits markedly above SnapKV’s deterministic top- K at every compression ratio—a substantial margin the predefined rule simply cannot close. Crucially, the gap is not structural: KV-Retriever, our task-finetuned scorer (§3), closes most of it with a *single* forward pass, while reaching the upper bound directly requires $N=50$ forwards per query. The bottleneck therefore lies not in the expressiveness of the scoring architecture but in the misalignment between the scoring criterion and the downstream task signal.

Our training method closes this gap by folding the per-query BoN search into the scorer itself, so one deterministic forward on the trained scorer recovers the majority of the headroom between the predefined rule and the reference. We keep each host eviction framework’s scoring and selection operators intact and only re-parameterize their Q,K inputs with identity-initialized per-layer projections. Before training, the composed scorer therefore coincides exactly with its host and only deviates under task signal. We derive the training signal directly from sampled keep-set comparisons: draw S keep sets from π_θ , score each by its answer-only LM loss under the frozen backbone, and update θ by a list-wise ranking objective so high-reward plans receive more probability mass. We call the resulting module **KV-Retriever**: $\sim 1.18\text{M}$ trainable parameters on a frozen Qwen3-8B (0.015%), with the

backbone weights never updated. One retriever is trained per backbone and then reused unchanged across SnapKV, AdaKV, CriticalKV, compression ratios, and benchmarks.

Our contributions are as follows:

- We quantify a budget-monotone gap between SnapKV’s deterministic scoring and its approximate compression upper bound (Figure 1), locating the bottleneck in the misalignment between scoring criterion and downstream task signal.
- We propose **KV-Retriever**, a layer-wise scoring module ($\sim 0.01\%$ extra parameters, identity-initialized) trained end-to-end against the answer-only LM loss via multi-trajectory sampling and a list-wise ranking objective, collapsing N BoN forwards into one deterministic pass (§3).
- One retriever per backbone, reused unchanged across SnapKV, AdaKV, and CriticalKV, closes the diagnosed gap on LongBench, LongBench-v2, and RULER (§4)—with strong zero-shot transfer to the latter two benchmarks unseen during training and to contexts up to 128K.

2 Related Work

2.1 Predefined-Criterion-Based Eviction

Existing KV cache eviction methods score tokens with predefined heuristics derived from attention statistics—cumulative attention (Zhang et al., 2023), persistence-weighted attention across decoding steps (Liu et al., 2023), query-window-aggregated attention (Li et al., 2024), pyramidal per-layer budget reshaping (Cai et al., 2025), or adaptive per-head budget reallocation (Feng et al., 2025a). Despite varying implementations, all are governed by human-designed priors and decoupled from downstream task signals.

2.2 Training-Based Eviction

A more recent line introduces learnable components into eviction (Moschella et al., 2026; Dong et al., 2026; Liu et al., 2026; Huang et al., 2025; Zeng et al., 2024), replacing hand-crafted rules with modules trained by future-utility rewards, future-attention oracles, retaining-score regression, or sigmoid-STE keep gates. Concurrent LKV (Zhou et al., 2026) learns both head-wise budgets (LKV-H) and query-agnostic token selection

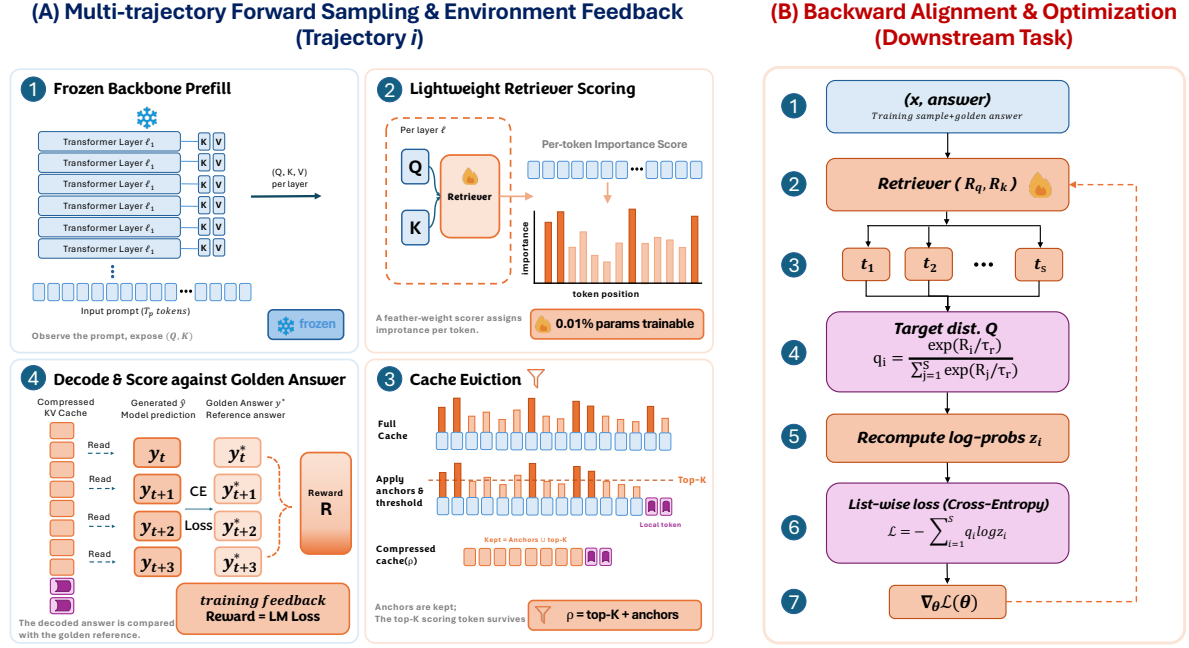


Figure 2: A single training step of KV-Retriever. **(A) Multi-trajectory forward sampling.** The frozen backbone prefills the prompt once. The per-layer retriever—the only trainable component ($\sim 0.01\%$ of backbone parameters)—scores the resulting KV cache; each of S sampled keep sets compresses this shared cache through the unchanged host framework (SnapKV / AdaKV / CriticalKV), and answer decoding yields reward $R_i = -\ell_i$ from the answer-only LM loss on y^* (§3.3). **(B) Backward alignment.** The rewards $\{R_i\}$ are softmax-normalized into a target ranking q (Eq. (8)) and matched to the retriever-recomputed trajectory log-probabilities (Eq. (7)) by a list-wise cross-entropy loss (Eq. (9)); gradients update only R_q, R_k while the backbone stays frozen (§3.4).

from K,V features (LKV-T); its budget component is orthogonal to our host-preserving scorer, while its selection component differs in both architecture and self-distillation signal. KV-Retriever instead aligns a query-dependent scorer directly with answer prediction error.

3 Method

We realize **KV-Retriever**, the construction motivated in the introduction, with three coordinated components: a per-layer learnable scorer with identity initialization that strictly extends the attention prior (§3.2); a stochastic relaxation of top- K retention via Plackett–Luce sampling, whose differentiable trajectory log-probabilities allow the LM loss—acting as a ranking reward—to induce gradients on the per-token scores (§3.3); and a list-wise ranking objective over sampled keep sets whose reward is the LM loss itself (§3.4). We first fix the notation (§3.1). Figure 2 gives the end-to-end overview: panel (A) covers the multi-trajectory forward sampling producing rewards $\{R_i\}$, panel (B) the backward alignment that turns the reward ranking into a list-wise loss whose gradient updates only R_q, R_k .

3.1 Preliminaries: KV Cache Eviction with a Scoring Function

Consider a frozen Transformer LLM processing a length- T_p prompt. At each decoder layer ℓ , we write $\mathbf{K}^{(\ell)}$ for the cached keys and $\mathbf{Q}^{(\ell)}$ for the queries of the most recent w_q tokens, the *observation window* used in SnapKV (Li et al., 2024).

A generic eviction method first assigns per-token importance scores

$$\mathbf{s}^{(\ell)} = a^{(\ell)}(\mathbf{Q}^{(\ell)}, \mathbf{K}^{(\ell)}), \quad (1)$$

then keeps the top-scoring positions per head under a budget induced by a target compression ratio $\rho \in (0, 1]$. Host frameworks differ mainly in their choice of $a^{(\ell)}$: SnapKV (Li et al., 2024) uses query-window-aggregated attention, AdaKV (Feng et al., 2025a) reallocates the same budget adaptively across heads, and CriticalKV (Feng et al., 2025b) further reweights the score by $\|\mathbf{V}\mathbf{W}_O\|_1$. In all cases $a^{(\ell)}$ is a *fixed function of attention statistics*.

Our goal is to preserve $a^{(\ell)}$ and instead learn lightweight transformations of the Q,K inputs it consumes. The host’s softmax normalization, observation-window aggregation, pooling, budget

allocation, and anchor windows remain unchanged, so the retriever is a drop-in re-parameterization rather than a replacement scoring rule.

3.2 KV-Retriever: A Lightweight Layer-wise Scorer

Architecture. At each decoder layer ℓ , KV-Retriever introduces only two weight matrices $\mathbf{R}_q^{(\ell)}, \mathbf{R}_k^{(\ell)} \in \mathbb{R}^{d \times d}$, applied independently to each per-head query and key vector. Writing $\tilde{\mathbf{Q}}^{(\ell)} = \mathbf{R}_q^{(\ell)} \mathbf{Q}^{(\ell)}$ and $\tilde{\mathbf{K}}^{(\ell)} = \mathbf{R}_k^{(\ell)} \mathbf{K}^{(\ell)}$ for these transformed tensors, the retriever score is the composition

$$s_\theta^{(\ell)} = a^{(\ell)}(\tilde{\mathbf{Q}}^{(\ell)}, \tilde{\mathbf{K}}^{(\ell)}). \quad (2)$$

For SnapKV, $a^{(\ell)}$ therefore still performs the original per-query softmax over the observation window, query-window voting, and max-pooling; AdaKV retains its adaptive head budgeting, while CriticalKV retains its $\|\mathbf{V}\mathbf{W}_O\|_1$ reweighting. The only learned change is the geometry of the Q,K inputs.

Identity initialization. A central design choice is to initialize $\mathbf{R}_q^{(\ell)} = \mathbf{R}_k^{(\ell)} = \mathbf{I}_d$. Equation (2) then feeds the host operator exactly the same Q,K tensors as the unmodified method. Consequently, its normalization, aggregation, pooling, scores, and deterministic top- K keep set are recovered exactly at step 0. Training departs from identity only where downstream task feedback indicates that the host’s attention geometry is suboptimal, yielding a host-anchored and stable optimization path.

Parameter footprint. KV-Retriever introduces only $2Ld^2$ trainable parameters in total, approximately 0.01% of a typical 8B backbone; the backbone weights remain frozen throughout training. Exact counts for the models used in our experiments are reported in §B.

3.3 Stochastic Top- K Retention via Plackett–Luce Sampling

The host consumes scores but emits a discrete keep mask, severing the gradient path from the LM loss to θ . We replace this deterministic top- K rule with a stochastic relaxation: a distribution over keep sets that (i) is smooth in θ , (ii) yields trajectory log-probabilities differentiable in θ , so that the LM loss—serving as a ranking reward over sampled keep sets—can drive gradient updates to the retriever scores through a list-wise surrogate, and (iii) recovers top- K in a sharpening limit.

Plackett–Luce factorization. For a fixed layer ℓ and KV head h , let $\mathcal{S}_h^{(\ell)} = (n^{(1)}, \dots, n^{(K)})$ denote the length- K keep set with $K = \lceil \rho T_p \rceil$. This set is drawn by sampling *without replacement* from the softmax of retriever scores at temperature τ_p , equivalent to the Gumbel-Top- k trick (Kool et al., 2019):

$$P(n^{(j)} = n \mid \mathcal{N}_{< j}) = \frac{\exp(s_\theta^{(\ell)}(h, n)/\tau_p) \mathbf{1}[n \notin \mathcal{N}_{< j}]}{\sum_{n' \notin \mathcal{N}_{< j}} \exp(s_\theta^{(\ell)}(h, n')/\tau_p)}, \quad (3)$$

where $\mathcal{N}_{< j} := \{n^{(1)}, \dots, n^{(j-1)}\}$ denotes the already-selected positions. Multiplying the K steps yields the Plackett–Luce factorization

$$\pi_\theta(\mathcal{S}_h^{(\ell)}) = \prod_{j=1}^K P(n^{(j)} \mid \mathcal{N}_{< j}). \quad (4)$$

Layers and KV heads are sampled independently, so the full retention plan

$$\mathcal{S} = \{\mathcal{S}_h^{(\ell)} : \ell \geq \ell_{\text{start}}, h = 1, \dots, H_{\text{kv}}\} \quad (5)$$

has $\pi_\theta(\mathcal{S}) = \prod_{\ell, h} \pi_\theta(\mathcal{S}_h^{(\ell)})$, supplemented by the deterministic anchor windows from §3.2. At inference $\tau_p \rightarrow 0$ collapses the sampler onto deterministic top- K and recovers the host’s standard behavior; finite τ_p at training keeps the sampler stochastic so diverse keep sets are explored.

Multi-trajectory sampling. For each training example with reference answer of length T_a , we draw S retention plans $\mathcal{S}_1, \dots, \mathcal{S}_S \stackrel{\text{iid}}{\sim} \pi_\theta$. The frozen backbone prefills the long prompt once to produce a full KV cache. Each plan then defines a deterministic mask over that shared cache and runs only the reference-answer decode to compute the answer-only LM loss

$$\ell_i = -\frac{1}{T_a} \sum_{t=T_p+1}^{T_p+T_a} \log p_\phi(x_t \mid x_{< t}; \mathcal{S}_i), \quad (6)$$

with reward $R_i = -\ell_i$. Thus S multiplies the short answer-decode term rather than the long prompt prefill. Sharing the same θ across all plans yields paired comparisons of different keep sets on the same input, which the next subsection turns into a learning signal.

3.4 Downstream-Task-Aligned Optimization Objective

The retriever should score token n at layer ℓ highly iff keeping that token improves downstream task quality. Given the trajectories $\{\mathcal{S}_i\}_{i=1}^S$ and rewards $\{R_i\}_{i=1}^S$ from §3.3, we update θ so that π_θ assigns more probability mass to high-reward plans, instantiated as a list-wise ranking loss built on the Plackett–Luce factorization in Eq. (4).

Recomputing trajectory log-probabilities. The S answer-decode passes run without gradients and do not retain activations. We therefore re-evaluate the retriever scores under the current θ on the same input when constructing the differentiable ranking objective. For each (layer ℓ , KV head h), the log-probability of trajectory i ’s token subset $\mathcal{S}_{i,h}^{(\ell)}$ under the current retriever follows directly from Eq. (4):

$$z_{i,h}^{(\ell)} = \log \pi_\theta(\mathcal{S}_{i,h}^{(\ell)}) = \sum_{j=1}^K \log \frac{\exp(s_\theta^{(\ell)}(h, n_i^{(j)})/\tau_p)}{\sum_{n' \notin \mathcal{N}_{< j}^{(i)} \exp(s_\theta^{(\ell)}(h, n')/\tau_p)}, \quad (7)$$

where $n_i^{(j)}$ is the j -th token in $\mathcal{S}_{i,h}^{(\ell)}$ in its original sampling order, and $\mathcal{N}_{< j}^{(i)} := \{n_i^{(1)}, \dots, n_i^{(j-1)}\}$. The cost of $z_{i,h}^{(\ell)}$ is $O(KN)$ per (layer, head, trajectory), and it is fully differentiable in θ .

Reward-induced target distribution. Trajectory rewards are converted into a target distribution over the S trajectories through a temperature-controlled softmax,

$$q_i = \frac{\exp(R_i/\tau_r)}{\sum_{j=1}^S \exp(R_j/\tau_r)}, \quad i = 1, \dots, S, \quad (8)$$

where τ_r is the reward temperature; smaller τ_r sharpens q toward the best trajectory, and $\tau_r \rightarrow \infty$ recovers a uniform target.

List-wise ranking loss. The training objective for layer ℓ and head h is the cross-entropy between q and the trajectory-level policy distribution induced by the log-likelihoods $z_{i,h}^{(\ell)}$,

$$\mathcal{L}_{\text{rank}}^{(\ell,h)} = - \sum_{i=1}^S q_i \log \frac{\exp(z_{i,h}^{(\ell)})}{\sum_{j=1}^S \exp(z_{j,h}^{(\ell)})}, \quad (9)$$

and the final loss aggregates Eq. (9) over active layers $\ell \geq \ell_{\text{start}}$ and KV heads h ,

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{H}|} \sum_{\ell \geq \ell_{\text{start}}} \sum_{h=1}^{H_{\text{kv}}} \mathcal{L}_{\text{rank}}^{(\ell,h)}. \quad (10)$$

Algorithm 1 KV-Retriever training step.

Require: Frozen LM p_ϕ ; KV-Retriever θ ; sample count S ; compression ratio ρ ; temperatures τ_r, τ_p ; start layer ℓ_{start} .

- 1: Prefill the prompt once with p_ϕ to obtain the full KV cache.
- 2: **for** $i = 1$ to S **do**
- 3: Sample plan $\mathcal{S}_i \sim \pi_\theta$ via Eq. (3) at τ_p .
- 4: Compress the shared cache with $\mathcal{S}_i$; decode y^* to obtain ℓ_i (Eq. (6)); set $R_i \leftarrow -\ell_i$.
- 5: **end for**
- 6: Compute target distribution q from $\{R_i\}$ via Eq. (8).
- 7: **for** layer $\ell \geq \ell_{\text{start}}$, head h , trajectory i **do**
- 8: Recompute $z_{i,h}^{(\ell)}$ from current θ via Eq. (7).
- 9: **end for**
- 10: Form $\mathcal{L}(\theta)$ via Eqs. (9)–(10); update θ (backbone frozen).

Equation (6) determines the reward-induced target q , while gradients flow through the recomputed trajectory log-probabilities in Eq. (7). The scorer is therefore aligned with answer prediction error without differentiating through the discrete keep masks or introducing an attention-based supervision proxy.

4 Experiments

We evaluate KV-Retriever along four axes: (i) whether re-parameterizing the Q,K inputs of existing host scorers yields consistent improvements across hosts, models, and benchmarks (§4.2); (ii) whether those gains transfer zero-shot from 8K training examples to contexts up to 128K (§4.3); (iii) how the scorer compares with prior *training-based* eviction methods along the orthogonal axes of scorer architecture and training signal (§4.4); and (iv) which internal design choices are responsible for the gain (§4.5).

4.1 Setup

Backbones, benchmarks, hosts. We freeze three backbones across two families—**Llama-3.1-8B-Instruct** (Grattafiori et al., 2024), **Qwen2.5-7B-Instruct** (Yang et al., 2024), **Qwen3-8B** (Yang et al., 2025)—and evaluate on **LongBench** (Bai et al., 2024) (16 English tasks: QA, summarization, few-shot, synthetic retrieval, code), **LongBench-v2** (Bai et al., 2025) (503 multiple-choice), and **RULER** (Hsieh et al., 2024) (13 NIAH-style sub-tasks at 4K, with a separate 8K–128K length sweep in §4.3). Per-task metrics follow each benchmark’s official protocol (Appendix A). KV-Retriever is hosted by three predefined-scoring frameworks—**SnapKV** (Li et al., 2024), **AdaKV** (Feng et al., 2025a), **CriticalKV** (Feng et al., 2025b)—with their scoring operators, budget allocators, and

Table 1: Plug-in gain at compression ratio $\rho = 0.2$. “w/” rows insert KV-Retriever before the host’s unchanged scoring operator; budget allocation and anchor windows are unchanged. “ Δ ” is the absolute improvement over the corresponding “w/o” row. Values are averaged over the per-task metrics of each benchmark; FullCache is the no-compression reference. **Bold** marks the best per (backbone, benchmark, host).

Host	Variant	Llama-3.1-8B-Instruct			Qwen2.5-7B-Instruct			Qwen3-8B		
		LB	LB-v2	RULER	LB	LB-v2	RULER	LB	LB-v2	RULER
FullCache		47.53	28.43	95.58	48.00	32.80	94.71	47.30	34.59	95.23
SnapKV	w/o	46.85	28.03	80.84	46.83	32.01	75.68	46.45	34.19	75.49
	w/	47.61	28.83	95.58	47.61	32.80	93.90	47.16	34.79	89.93
	Δ	+0.76	+0.80	+14.74	+0.78	+0.79	+18.22	+0.71	+0.60	+14.44
AdaKV	w/o	46.73	28.03	80.18	46.70	31.81	73.96	46.45	34.39	81.47
	w/	47.62	28.63	95.62	47.65	32.60	93.98	47.19	34.79	90.91
	Δ	+0.89	+0.60	+15.44	+0.95	+0.79	+20.02	+0.74	+0.40	+9.44
CriticalKV	w/o	46.86	28.83	83.21	46.87	32.01	78.15	46.52	34.39	79.12
	w/	47.40	29.03	95.55	47.24	32.41	93.19	47.06	34.59	86.86
	Δ	+0.54	+0.20	+12.34	+0.37	+0.40	+15.04	+0.54	+0.20	+7.74

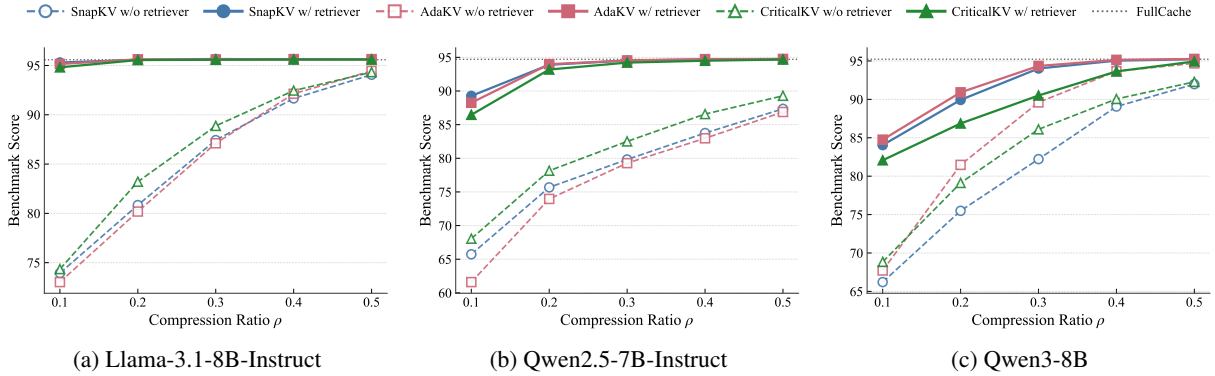


Figure 3: RULER score vs. compression ratio $\rho \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ across three backbones. Each subplot overlays three host pairs (SnapKV, AdaKV, CriticalKV): solid lines denote “w/ KV-Retriever” and dashed lines the predefined-scoring baseline; the grey horizontal line marks FullCache.

anchor-window logic intact; we only insert the Q,K transformations in Eq. (2). **FullCache** is the no-compression reference.

Training and ratios. The compression ratio ρ is swept over $\{0.1, 0.2, 0.3, 0.4, 0.5\}$; headline tables report $\rho = 0.2$ (80% reduction). For each backbone we train one set of weights on a 2WikiMQA/HotpotQA/GovReport/Multi-News mixture (LongBench training pool); the same retriever is reused across *all* hosts and benchmarks. The listwise objective draws $S = 8$ trajectories with $\tau_r = 6$ (Figure 6a); only $\mathbf{R}_q^{(\ell)}, \mathbf{R}_k^{(\ell)}$ receive gradients. Data quotas, packing, and optimizer settings are in Appendix A.

4.2 Plug-in Gain over Predefined-Scoring Hosts

For each host $\in \{\text{SnapKV}, \text{AdaKV}, \text{CriticalKV}\}$ and each backbone, we compare the host’s

predefined-scoring version (“w/o”) with its KV-Retriever drop-in replacement (“w/”).

Table 1 reports the headline comparison at $\rho = 0.2$: inserting KV-Retriever yields a consistent positive Δ across every (backbone, host, benchmark) cell. The gains are concentrated on RULER, where the “w/” rows recover close to the FullCache reference, while LongBench and LongBench-v2 receive smaller but uniformly positive improvements. The pattern holds across all three backbones and all three hosts, indicating that the gain originates from the learned Q,K geometry rather than from host-specific tuning.

Figure 3 shows the full compression-ratio sweep: every “w/ KV-Retriever” curve hugs the FullCache reference, while the “w/o” baselines degrade rapidly as ρ decreases. The gap is monotone in ρ , confirming that the plug-in gain is concentrated in the tight-budget regime.

Out-of-distribution generalization. The retriever is trained on only four LongBench datasets, yet the absolute gains are *largest* on the most distributionally distant benchmark—RULER (+14 to +20 points)—while LongBench-v2 also receives consistent positive transfer (+0.2 to +0.8). This is because the LM-loss signal optimizes for “which tokens reduce prediction error on *any* answer” rather than corpus-specific patterns, yielding a task-general notion of token importance that transfers across formats and domains.

Why RULER shows larger absolute gains. LongBench answers are often recoverable from multiple token subsets, so predefined eviction is already close to the FullCache reference and leaves little gap to close. RULER instead stresses exact retrieval, where evicting one key token can flip the answer. Normalizing Table 1 by the FullCache-reference gap, KV-Retriever closes on average 76% of that gap on LongBench and 86% on RULER. The roughly $18\times$ difference in available gap (0.91 vs. 16.50 points) explains the difference in absolute improvement; it is not an in- versus out-of-distribution effect, since most LongBench tasks, all of LongBench-v2, and all of RULER are unseen during training.

4.3 Zero-shot Generalization to 128K Contexts

The headline RULER results above use 4K inputs. To test the long-context regime directly, we evaluate the Qwen2.5-7B + SnapKV configuration at $\rho = 0.2$ from 8K to 128K. All rows use the same checkpoint trained on examples capped at 8192 tokens, without length-specific retraining.

Table 2: Zero-shot RULER length transfer with Qwen2.5-7B + SnapKV at $\rho = 0.2$. “Closure” is $(\text{Ours} - \text{SnapKV}) / (\text{FullCache} - \text{SnapKV})$.

Length	FullCache	SnapKV	Ours	Δ	Closure
8K	92.18	75.44	88.79	+13.35	80%
16K	91.31	78.51	88.30	+9.79	76%
32K	89.45	77.95	86.13	+8.18	71%
64K	79.85	70.35	77.22	+6.87	72%
128K	66.53	59.56	65.01	+5.45	78%

Absolute improvement decreases from 13.35 points at 8K to 5.45 at 128K as FullCache itself degrades, but the normalized closure remains stable at 71–80% at every length. A retriever trained only at 8K therefore preserves its advantage on inputs up to $16\times$ longer.

One-time training cost and scope. Training one Qwen2.5-7B retriever for one epoch over 20K golden-answer examples takes 26.4 wall-clock hours on 8 GPUs (approximately 211 GPU-hours), with 29 GiB peak memory per GPU at $S = 8$, bf16, and batch size 1. The long prompt is pre-filled once and shared by all sampled keep sets; only their short answer decodes are repeated. This supervised cost is paid once per backbone, after which the retriever is reused across hosts, compression ratios, and benchmarks. Thus “plug-and-play” describes deployment into an unchanged host, not a training-free method.

Across-seed stability. Across three independent Qwen2.5-7B + SnapKV runs, the LongBench improvement is positive for every seed. The standard deviation of the 16-task macro-average Δ is 0.26, 0.08, and 0.02 at $\rho = 0.1, 0.3, 0.5$, respectively, several times smaller than the corresponding improvement.

4.4 Decomposing the Comparison With Prior Training-Based Eviction

Recent training-based eviction scorers (Moschella et al., 2026; Dong et al., 2026; Liu et al., 2026; Huang et al., 2025; Zeng et al., 2024) differ from KV-Retriever along two coupled axes—*scorer architecture* and *training signal*—and their published numbers entangle both with host and backbone confounders. We therefore transplant one axis at a time onto our pipeline, holding the remainder at SnapKV + Qwen2.5-7B-Instruct, and report RULER ($\rho = 0.2$) where differences are most pronounced. Variant-to-baseline mappings are in Appendix C.

Axis 1 — Scorer architecture. We replace the $\mathbf{R}_q, \mathbf{R}_k$ projections with four alternative scorer modules, all trained under the answer LM loss of Eq. (6) and the list-wise objective of §3.4: (i) Q-K projection plus a per-token value head; (ii) a future-attention predictor head over Q/K; (iii) a per-layer sigmoid+STE keep-gate; and (iv) a per-layer retention regression head.

Figure 4(a) shows that all four alternatives, once supervised by the answer LM loss, clear the untrained baseline on RULER by a wide margin—LM-loss supervision lifts even imperfect scorer forms far above the predefined-attention prior (75.68). They nonetheless trail our $\mathbf{R}_q, \mathbf{R}_k$ projections (93.90): the sigmoid+STE keep-gate suffers

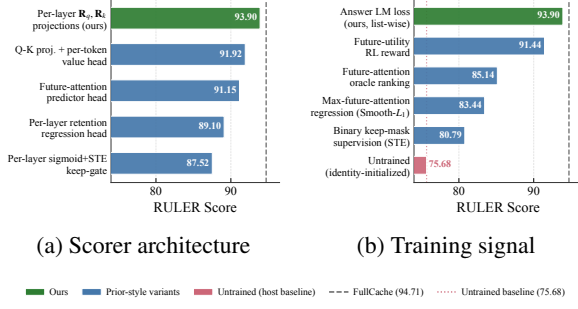


Figure 4: Decomposing the RULER score gap between prior training-based eviction and KV-Retriever along two axes under a unified setting (Qwen2.5-7B-Instruct + SnapKV, $\rho = 0.2$). (a) Architectural ablation: all variants share the answer LM loss and differ *only* in the scorer architecture. (b) Training-signal ablation: all variants share the per-layer $\mathbf{R}_q, \mathbf{R}_k$ scorer and differ *only* in the supervision driving θ . The dashed line marks the FullCache reference (94.71); the dotted line marks the untrained host baseline (75.68).

most (87.52; biased gradients from the straight-through relaxation over a binary output), the Q-K-plus-value-head variant comes nearest (91.92), and the regression and predictor heads fall in between (89.10 and 91.15). The identity-initialized per-token Q-K matching geometry preserves retrieval-intensive scoring precision that auxiliary heads, scalar regression, or binary gates dilute or distort.

Axis 2 — Training signal. Mirroring Axis 1, we hold the scorer at the identity-initialized $\mathbf{R}_q, \mathbf{R}_k$ projections of §3.2 and replace the supervision with: (i) no gradient—identity-init as a strict architectural lower bound; (ii) a future-utility RL reward adapted from KVP (Moschella et al., 2026); (iii) a future-attention oracle ranking adapted from ForesightKV (Dong et al., 2026); (iv) a sigmoid-STE keep-gate objective inspired by Attention-Gate (Zeng et al., 2024); and (v) max-future-attention regression adapted from Locret (Huang et al., 2025). Appendix C gives the exact correspondence.

Figure 4(b) shows every proxy improves over the untrained baseline (75.68)—future-utility RL reward leads at 91.44, with the rest spanning from 80.79 to 85.14—yet none closes the gap to LM-loss supervision (93.90). The strongest proxy still trails by 2.46 RULER points, confirming that the residual gap is attributable to the supervision signal itself: a controlled analog of the in-family headroom that Figure 1 diagnoses on SnapKV, isolated to the choice of training signal under a shared scorer.

Cross-axis takeaway. Reading the two panels together, the architectural spread (Axis 1) is at most 6.4 RULER points, while the supervision spread (Axis 2) reaches 18.2 points against no training. Replacing the supervision is worth roughly $1.3\times$ as much as replacing the architecture among prior-style alternatives. The missing ingredient is therefore quantitatively the training signal, not the scorer form, and only the joint configuration (answer LM loss + $\mathbf{R}_q, \mathbf{R}_k$) reaches near-FullCache performance (93.90 vs. 94.71).

4.5 Ablations and Analyses

Holding the architecture and supervision at our defaults ($\mathbf{R}_q, \mathbf{R}_k$ projections under the answer-only LM loss), we now scan three internal design choices: the ranking-objective form, the reward temperature τ_r , and the sampling count S .

Ablation 1: Choice of ranking objective. We compare five ranking objectives over the S sampled trajectories, holding all other hyperparameters fixed: (i) *Best-vs-worst*: pairwise margin between the highest- and lowest-reward trajectory; (ii) *All-pairs*: margins over every trajectory pair with positive reward gap; (iii) *DPO* (Rafailov et al., 2023): Bradley-Terry preference loss on the best/worst pair; (iv) *GRPO* (Shao et al., 2024): group-relative policy gradient weighted by reward advantage; (v) *List-wise (ours)*: cross-entropy between q (Eq. (8)) and the trajectory log-likelihood distribution. Table 3 shows the list-wise objective strictly outperforms every alternative, consistent with its use of all S trajectories rather than a single pair.

Table 3: Ranking-objective ablation at $\rho = 0.2$ on Qwen2.5-7B-Instruct + SnapKV.

Ranking objective	LB	LB-v2	RULER
Best-vs-worst	46.96	32.21	91.19
All-pairs	47.37	32.41	91.00
DPO (Rafailov et al., 2023)	47.08	32.01	90.00
GRPO (Shao et al., 2024)	47.23	32.21	91.88
List-wise (ours)	47.61	32.80	93.90

Ablation 2: Reward temperature τ_r and sampling count S . The reward temperature τ_r (Eq. (8)) controls how sharply the target q concentrates on the highest-rewarded trajectory; the sampling count S governs how many keep-set comparisons each update sees. Figure 6 sweeps $\tau_r \in \{0.5, 1, 3, 6, 12\}$ and $S \in \{2, 4, 8, 16\}$ with all other hyperparameters fixed: performance peaks

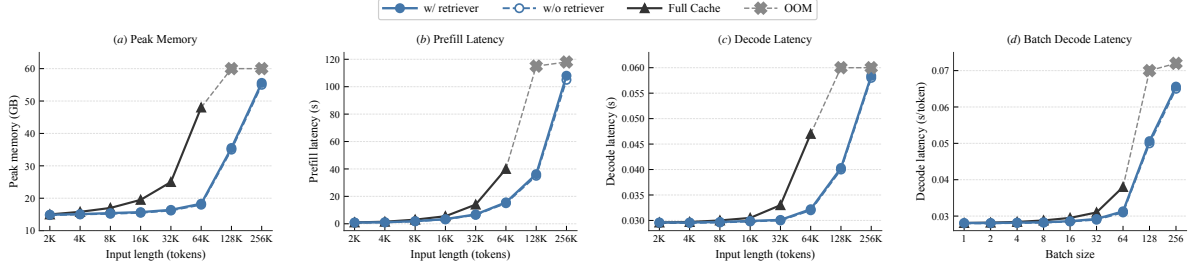


Figure 5: **Efficiency analysis of KV-Retriever.** We compare the host eviction method with (solid) and without (dashed) the retriever module under varying context lengths and batch sizes on a single 80GB GPU with Llama-3.1-8B-Instruct as backbone and SnapKV ($\rho=0.2$) as host. (a) Peak GPU memory. (b) Prefill latency. (c) Per-token decode latency. (d) Batch decode latency (2K context length per batch). The two blue curves overlap almost perfectly, confirming that the retriever adds $<3\%$ overhead across all settings. “Full Cache” (no eviction) is shown for reference; it OOMs at 128K.

at $\tau_r = 6$ and at $S = 8$, with fewer rollouts ($S \leq 4$) underfitting the ranking signal and more ($S = 16$) destabilizing the update, so $\tau_r = 6$ and $S = 8$ are chosen as the defaults.

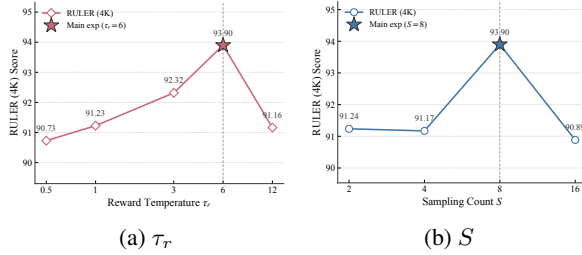


Figure 6: Hyperparameter sensitivity on Qwen2.5-7B-Instruct + SnapKV at $\rho = 0.2$ on RULER (4K). Dashed lines mark defaults $\tau_r = 6$, $S = 8$.

Ablation 3: Transfer across training-time compression ratios. KV-Retriever learns a token ranking rather than a fixed budget: the unchanged host top- K operator can truncate the same ordering at any test ratio. Table 4 verifies this by training three Llama-3.1-8B + SnapKV checkpoints at $\rho_{\text{train}} \in \{0.1, 0.3, 0.5\}$ and evaluating each on RULER across ratios.

Table 4: RULER score under training- and evaluation-ratio transfer. Columns are ρ_{train} ; rows are evaluation ratios.

ρ_{eval}	0.1	0.3	0.5
0.1	82.13	81.92	86.40
0.3	93.58	93.70	94.30
0.5	94.62	94.51	94.80

At evaluation ratios 0.3 and 0.5, all checkpoints agree within 0.7 and 0.3 points. At the tightest ratio 0.1, training with $\rho_{\text{train}} = 0.5$ leads by about four

points, supporting the default: a larger sampled keep set supplies a better-populated global ranking signal that transfers to stricter budgets.

4.6 Inference Efficiency

A practical concern for any learnable scoring module is whether it introduces prohibitive inference overhead. We show that KV-Retriever adds negligible cost, preserving the efficiency gains of the host compression method almost entirely.

Why the overhead is minimal. The retriever runs *once* during prefill—two batched linear projections per layer before the unchanged host operator—and the resulting scores are cached for all subsequent decoding steps (Appendix B), contributing zero additional FLOPs during autoregressive decode. Figure 5 confirms this empirically: across all input lengths (2K–256K) and batch sizes (1–256), the retriever adds <0.6 GB peak memory, $<3\%$ prefill latency, and $<1\%$ decode latency. Both variants reach the same maximum context length before OOM, confirming that the retriever preserves the host’s full serving capacity.

5 Conclusion

Existing KV cache eviction methods score tokens by proxies decoupled from downstream task quality. We proposed **KV-Retriever**, a layer-wise plug-and-play Q,K re-parameterization ($\sim 0.01\%$ extra parameters) aligned with answer loss via Plackett–Luce sampling and a list-wise ranking objective. One retriever per backbone is reused across three host frameworks, compression ratios, and benchmarks, with zero-shot length transfer from 8K training examples to 128K inputs.

Limitations

While our experiments span three backbones, three host frameworks, and three benchmarks, a few minor aspects remain unexplored. First, we have not yet tested KV-Retriever on context lengths beyond 128K tokens; although the method is architecturally length-agnostic, verifying this on ultra-long inputs would further strengthen generality. Second, our current training uses a fixed mixture of four datasets; expanding to a broader data mixture may yield marginal additional gains on niche domains. Third, the reported multi-seed analysis covers Qwen2.5-7B + SnapKV; extending uncertainty estimates to every backbone-host combination would strengthen the robustness analysis. Finally, we report wall-clock efficiency and training cost on a single GPU hardware configuration; profiling across additional datacenter and consumer GPU configurations would complement the current analysis.

Ethical considerations

This paper presents work whose goal is to advance the field of Machine Learning. Since our work concerns context-to-parameter mapping and involves lossy compression mechanisms described in Section 5, which may impact factual recall in entity-intensive tasks, we expect our work to be used under the guidance of human values.

References

- Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14388–14411.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.
- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2025. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664.
- Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2025. [Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling](#). In *Second Conference on Language Modeling*.
- Zican Dong, Peiyu Liu, Junyi Li, Zhipeng Chen, Han Peng, Shuo Wang, and Wayne Xin Zhao. 2026. Foresightkv: Optimizing kv cache eviction for reasoning models by learning long-term contribution. *arXiv preprint arXiv:2602.03203*.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025a. [Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference](#). *Advances in Neural Information Processing Systems*, 38:113152–113188.
- Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. 2025b. [CriticalKV: Optimizing KV cache eviction from an output perturbation perspective](#). *Preprint*, arXiv:2502.03805.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekeshe, Fei Jia, and Boris Ginsburg. 2024. [RULER: What’s the real context size of your long-context language models?](#) In *First Conference on Language Modeling*.
- Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. 2025. [Locret: Enhancing eviction in long-context llm inference with trained retaining heads on consumer-grade devices](#). *Preprint*, arXiv:2410.01805.
- Wouter Kool, Herke van Hoof, and Max Welling. 2019. [Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement](#). *Preprint*, arXiv:1903.06059.
- Haoyang LI, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole HU, Wei Dong, Li Qing, and Lei Chen. 2025. [A survey on large language model acceleration based on KV cache management](#). *Transactions on Machine Learning Research*.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Tianyang Liu, Canwen Xu, and Julian McAuley. 2024. [Repobench: Benchmarking repository-level code auto-completion systems](#). In *The Twelfth International Conference on Learning Representations*.

Yijun Liu, Yixuan Wang, Yuzhuang Xu, Shiyu Ji, Yang Xu, Qingfu Zhu, and Wanxiang Che. 2026. Judge q: Trainable queries for optimized information retention in kv cache eviction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 32240–32248.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhao Xu, Anastasios Kyriklidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364.

Luca Moschella, Laura Manduchi, and Ozan Sener. 2026. [Learning to evict from key-value cache](#). Preprint, arXiv:2602.10238.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Zihao Zeng, Bokai Lin, Tianqi Hou, Hao Zhang, and Zhijie Deng. 2024. [In-context KV-cache eviction for LLMs via attention-gate](#). Preprint, arXiv:2410.12876.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuan-dong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Enshuai Zhou, Yifan Hao, Chao Wang, Rui Zhang, Di Huang, Jiaming Guo, Xing Hu, Zidong Du, Qi Guo, and Yunji Chen. 2026. [LKV: End-to-end learning of head-wise budgets and token selection for LLM KV cache eviction](#). Preprint, arXiv:2605.06676.

A Additional Experimental Setup

Benchmark task composition. **LongBench** comprises 16 English tasks across single-/multi-document QA, summarization, few-shot learning, synthetic retrieval, and code completion. **LongBench-v2** is a 503-question multiple-choice benchmark scored by classification accuracy. **RULER**’s 13 sub-tasks include NIAH single/multikey/multivalue/multiquery, variable tracking, common/frequent word extraction, and synthetic QA, all evaluated at a 4K input length. Per-task metrics follow each benchmark’s official protocol: F1 for QA, ROUGE-L for summarization, accuracy for multi-choice, exact/partial match for synthetic tasks.

Training data details. Each training example is truncated/packed to a maximum sequence length of 8192 tokens, with a per-dataset quota of 5,000 examples and a global cap of 20,000 training examples in total across the 2WikiMQA / HotpotQA / GovReport / Multi-News mixture.

Optimizer settings. We optimize with AdamW at learning rate $1e-4$, per-device batch size 1, on 8 GPUs of a single node (NCCL DDP) for 1 epoch with random seed 42. Rollouts are produced on the SnapKV host with chunk size 1 (per-token scoring) and a fixed training-time compression ratio $\rho_{\text{train}} = 0.5$.

B Implementation Details

Practical scoring choices. We adopt three simple but important practical choices that hold for both training and inference: (1) *Layer-wise application from ℓ_{start}* . The first ℓ_{start} layers run full attention without any score override, since shallow layers tend to encode position-local features whose importance is broadly distributed and provides little signal for compression. (2) *Anchor windows*. A sink window of size w_s at the front and a local window of size w_ℓ at the back are always retained, mirroring StreamingLLM-style protections to stabilize next-token attention. (3) *Per-layer score caching at inference*. Once the prompt’s score $s^{(\ell)}$ is computed at the end of prefill, it is reused throughout decoding without further retriever forwards, so the only added inference-time work is the two per-layer Q,K projections at prefill time; the host scoring operator is unchanged.

Exact parameter counts. KV-Retriever introduces $2L d^2$ trainable weights per backbone. Concretely, this is $\sim 0.92\text{M}$ parameters for Qwen2.5-7B ($L = 28, d = 128$), $\sim 1.18\text{M}$ ($\sim 0.015\%$) for Qwen3-8B ($L = 36, d = 128$), and $\sim 1.05\text{M}$ ($\sim 0.013\%$) for Llama-3.1-8B-Instruct ($L = 32, d = 128$).

Memory-efficient sampling. The long prompt is prefilled once to obtain a shared full KV cache. The S answer-decode passes used to obtain $\{\ell_i\}$ then run without gradients; only the scalar losses ℓ_i and discrete sampled indices $\mathcal{S}_i$ are retained for log-probability re-evaluation. Sampled indices are kept in host memory to bound activation memory across trajectories.

Reward shaping. Beyond the temperature-controlled softmax in Eq. (8), a top- M truncated target—where q is set to a uniform distribution over the M best-rewarded trajectories—can also be used. This is convenient when low-ranked plans carry noisy reward signals.

Distributed-training stability. On individual ranks, some layers may produce no usable ranking term in a step (e.g., $\rho \geq 1$, or all sampled plans coincide), which would detach those layers’ parameters from the autograd graph and stall DDP under `find_unused_parameters=False`. We resolve this by adding a zero-magnitude keep-alive term, $0 \cdot \bar{s}^{(\ell)}$, to such layers, preserving graph connectivity without affecting gradients elsewhere. Because the host eviction method is otherwise unchanged, gradients flow only through the retriever’s two projections at each layer, and the backbone stays fully frozen.

C Correspondence Between Ablation Variants and Prior Training-Based Eviction Methods

This appendix records, for each variant compared in Figure 4, the prior training-based eviction method from which it is adapted (Moschella et al., 2026; Dong et al., 2026; Liu et al., 2026; Huang et al., 2025; Zeng et al., 2024). It also explains why Judge Q does not factor cleanly into either of the two axes considered in §4.4.

Scorer-architecture variants (Axis 1). Figure 4(a) adapts the Q-K + value-head actor-critic of KVP, ForesightKV’s future-attention predictor, Attention-Gate’s sigmoid+STE mask, and Locret’s

retention-regression head. Each replaces $\mathbf{R}_q, \mathbf{R}_k$ in our trainer while retaining answer-LM-loss supervision and a comparable parameter footprint, isolating scorer form from capacity.

Training-signal variants (Axis 2). Figure 4(b) adapts KVP’s future-utility reward, ForesightKV’s future-attention ranking, a sigmoid-STE keep-gate objective inspired by Attention-Gate, and Locret’s max-future-attention regression. All objectives train the same $\mathbf{R}_q, \mathbf{R}_k$ scorer in our trainer, isolating the training signal.

Why Judge Q is excluded. Judge Q learns a query proxy but retains the host’s attention-score form, so it provides no distinct Axis-1 scorer. Its decoded-token attention supervision is also not a self-contained target on keep decisions for Axis 2; we therefore do not force it onto either axis.

Matched comparison with concurrent LKV. With no public LKV (Zhou et al., 2026) implementation available at evaluation time, we transplant its token-selection choices into the matched Qwen2.5-7B + SnapKV, $\rho = 0.2$ RULER setting. Its query-agnostic $[\mathbf{k}; \mathbf{v}]$ MLP and FullCache self-distillation variants score 91.74 and 90.04, respectively, versus 93.90 for KV-Retriever; LKV-H’s learned budget allocation is orthogonal and potentially composable.