CARVEKV: CROSS-LAYER ADAPTIVE LATENT VECTOR EVICTION FOR EFFICIENT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-Head Latent Attention (MLA) reduces the per-token memory footprint of the KV cache, yet cache storage still grows linearly with context length. Existing token-eviction methods are largely formulated for expanded KV representations or make independent layer-wise decisions, leaving open whether eviction can be performed directly and consistently in MLA's compressed latent space. We show that latent-token importance rankings persist substantially across Transformer depth, even after controlling for causal eligibility and token position. Based on this observation, we propose CarveKV, a training-free MLA-native eviction framework that aggregates rank-normalized importance estimates across multiple layers and combines prompt-wide and final-window attention views through effective-support routing. CarveKV produces a single shared keep set across layers and scores compressed latent and decoupled-RoPE caches directly, without persistently reconstructing expanded per-head KV tensors. Across 80 natural-language and code windows, eligibility-normalized pairwise Spearman correlation reaches 0.651-0.654 and remains above 0.50 under within-position controls. On eight LongBench tasks at a matched 60% physical-cache budget, CarveKV retains 96.6% of full-cache quality. Under matched 8K-32K system settings, it also reduces prefill latency by 1.0-3.0%, with comparable decode throughput and peak memory.

1 INTRODUCTION

Long-context language models must preserve increasingly long prefixes during autoregressive decoding. They reuse attention states through the key–value (KV) cache, whose size grows linearly with sequence length and can dominate GPU memory. This growth restricts serving batch sizes and makes memory a central obstacle to efficient long-context deployment. A natural way to alleviate this bottleneck is to compress the cache representation associated with each token. Multi-Head Latent Attention (MLA) (Liu et al., 2024a) reduces per-token cost by caching compact low-rank latents instead of full per-head keys and values, but total storage remains linear in the retained token count. Token eviction is therefore a natural complement. Yet many eviction strategies define importance using expanded keys, values, or full attention maps. Reconstructing these representations solely to prune compressed entries adds computation and temporary memory that an MLA-native method should avoid. Independent layer-wise eviction also produces different token layouts across depth, whereas a shared physical layout is easier to integrate with cache interfaces that expect compatible sequence positions.

This motivates our central empirical question: does latent-token importance persist across Transformer depth strongly enough to support a shared eviction decision? Prior work observes persistence in conventional KV caches (Liu et al., 2023), but its extent in MLA's latent space is not established. We measure rankings across ten layers on 40 natural-language and 40 code windows. After dividing each token's accumulated attention mass by the number of sampled queries causally eligible to attend to it, mean pairwise Spearman correlation remains 0.651 and 0.654 in the two domains, with top-30% keep-set Jaccard overlap of 0.498 in both. Correlation computed within matched position bins remains 0.507 and 0.505. We call this empirical regularity cross-layer persistence of latent importance. The agreement remains substantial after controlling for causal eligibility and coarse token position, suggesting that it cannot be attributed solely to causal exposure. Persistence therefore does not imply that layers are interchangeable. Instead, the persistence provides a common signal that

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.

allows evidence from multiple layers to be aggregated into a single shared keep set. This motivates aggregation across multiple depths instead of relying on one fixed scoring layer.

Building on this observation, we introduce CarveKV, a training-free eviction framework for compressed MLA caches. It scores resident latents with complete absorbed-content and decoupled-RoPE logits, modulates global attention mass by a value-aware term, and protects sink and recent tokens. A rank-normalized committee estimates global importance, an effective-support router then reserves a data-dependent quota for observed final-window prefill attention and fills the remaining slots globally. The resulting mask is shared across layers, and an optional knee-and-coverage heuristic can select a retained budget when no fixed target is supplied. CarveKV avoids persistently materializing expanded KV tensors and requires neither training nor changes to model weights.

Our main contributions are:

- We establish cross-layer persistence of latent importance on 80 natural-language and code windows, using causal-eligibility and within-position-bin controls in addition to rank correlation and retained-set overlap.
- We propose CarveKV, which combines a RoPE-complete rank-coverage committee with effective-support routing to construct a globally shared eviction decision without a manually fixed fusion weight.
- We evaluate the resulting quality–memory trade-off under fixed, matched physical-cache budgets. Across eight LongBench tasks with 100 examples per task, fixed 60% retention preserves 96.6% of full-cache quality and remains within 0.098 macro points of an MLA-native SnapKV-style adaptation. Under matched system conditions, CarveKV also lowers prefill latency by 1.0–3.0% across 8K–32K contexts.

Together, these results position cross-layer latent eviction as a complementary stage after architectural cache compression: MLA reduces the representation stored for every token, while CarveKV selectively reduces how many token positions remain resident.

2 Background and Related Work

KV-cache efficiency can be addressed at the systems, architectural, and token levels (Zhou et al., 2024). PagedAttention (Kwon et al., 2023) improves physical allocation by paging cache blocks, but does not change how many states the model logically retains. Multi-Query Attention (Shazeer, 2019) and Grouped-Query Attention (Ainslie et al., 2023) instead share key–value heads, reducing the representation stored per position. Both directions improve serving efficiency, yet the remaining cache still grows as $\mathcal{O}(T)$ with context length.

Token selection attacks this remaining length dimension by removing positions judged unlikely to support future decoding. H_2O (Zhang et al., 2023) and Scissorhands (Liu et al., 2023) exploit accumulated or persistent attention importance, while SnapKV (Li et al., 2024) and PyramidKV (Cai et al., 2024) summarize observation-window behavior. StreamingLLM (Xiao et al., 2023) protects attention sinks together with a recent window. FastGen (Ge et al., 2024), DMC (Nawrot et al., 2024), and NACL (Chen et al., 2024) further vary how utility and budgets are assigned. Quest, TOVA, RazorAttention, and LazyLLM (Tang et al., 2024; Oren et al., 2024; Tang et al., 2025; Fu et al., 2024) make selection query-dependent or prune during prefill. Although their policies differ, these methods are generally formulated using expanded keys, values, or attention statistics. Applying the same decision rules to MLA can therefore require an additional representation or a cache-specific reformulation.

Other compression families change what is stored rather than which positions survive. KVQuant (Hooper et al., 2024) and HqeKV (Wang et al., 2026) lower per-entry precision while retaining token count, making quantization complementary to eviction but leaving token-wise decoding bandwidth linear in T. IndexMem (Yang et al., 2026) instead summarizes removed representations in a residual memory module. Such residual paths soften the information loss of eviction, but do not by themselves determine a shared keep set for a compressed latent cache.

DeepSeek-V2/V3 (Liu et al., 2024a;b) introduce Multi-Head Latent Attention (MLA), which maps each hidden state through a low-rank bottleneck (Wang et al., 2020) and caches $\mathbf{c}_t^{KV} \in \mathbb{R}^{d_c}$ with

$d_c \ll n_h d_h$. Together with a decoupled RoPE key (Su et al., 2024), the evaluated configuration stores $d_c + d_R = 576$ scalars per token rather than $2n_h d_h = 4096$ for standard MHA, approximately a $7\times$ reduction. Weight absorption evaluates the content term against cached latents without materializing expanded content keys. TransMLA (Meng et al., 2025) and MHA2MLA (Ji et al., 2025) retrofit paths to this parameterization. MTLA (Deng & Woodland, 2026) learns temporal merging, whereas LCA (You et al., 2026) combines query-aware pooling with positional anchors. LCA replaces the original attention operator with group-wise condensation: semantic latent vectors are aggregated through query-aware pooling, whereas positional keys are preserved through anchor selection. Its reported efficiency relies on continued fine-tuning of the modified model and a fused Triton implementation. CarveKV instead preserves the pretrained model weights and MLA operator, and performs a training-free, post-prefill cache decision. The two methods therefore address adjacent but distinct operating regimes: LCA adapts the attention architecture for aggressive joint compute–memory reduction, whereas CarveKV provides a plug-in cache policy for existing MLA checkpoints. Whereas these approaches compress or aggregate latent representations, CarveKV specifically asks which latent token positions should remain resident after architectural compression.

Cross-Layer Shared Attention (Mu et al., 2024) and layer dropping (Men et al., 2025) exploit redundancy across depth by changing attention computation or removing layers. CarveKV leaves the model structure and weights unchanged. It uses cross-layer persistence to score compressed cache entries and commit one shared token decision without persistently materializing expanded KV. Its focus is therefore training-free, MLA-compatible token eviction rather than architectural cross-layer sharing.

3 METHOD

CarveKV is a latent-space cache eviction method designed for MLA inference. For a given prefill sequence, it scores compressed latent and RoPE-key entries from global and final-window query views, routes a data-dependent share of the keep budget to each view, and executes one shared eviction decision.

Let N denote the total prefix length. After protecting the first N_s sink tokens and last N_r recent tokens, CarveKV scores the mid-token region $\mathcal{M} = \{N_s, \ldots, N - N_r - 1\}$, whose size is $n = |\mathcal{M}| = N - N_s - N_r$; thus N always denotes the full prefix and n denotes the scored mid tokens.

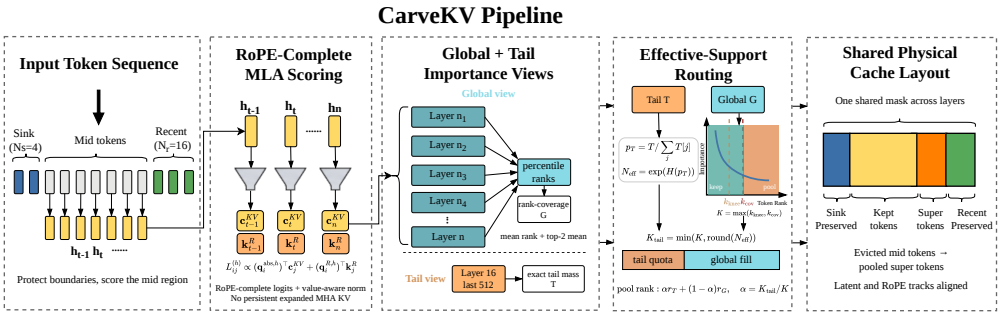


Figure 1: Overview of CarveKV's latent-cache eviction pipeline. Sink and recent tokens are protected, while mid-region latent entries are scored using RoPE-complete MLA attention at multiple layers. Cross-layer percentile ranks form a global score, and exact attention from the final query window supplies a tail score. Effective-support routing allocates the keep budget between the two views, after which one shared mask is applied across layers.

CarveKV protects sink and recent entries and ranks the mid-token region. Layers $\{4, 6, 8, 12, 16\}$ score at most 512 prompt-wide queries with complete MLA logits, their percentile ranks form a committee at decision layer D=16. Exact attention from the final 512 queries supplies a tail view, whose effective support sets its quota, and the global view fills the remainder. For optional variable-budget operation, we additionally use a knee-and-coverage heuristic defined in Appendix B; all

primary quality comparisons instead use fixed matched physical budgets. Evicted tokens may be merged into super tokens (Kim et al., 2024).

3.1 RoPE-Complete Query and Value-Aware Scoring

In MLA (Liu et al., 2024a), given hidden state $\mathbf{h}_t$, a shared low-rank latent is computed as $\mathbf{c}_t^{KV} = W_{DKV}\,\mathbf{h}_t \in \mathbb{R}^{d_c}$, from which the content key and value are recovered:

$$\mathbf{k}_t^C = W_{UK}\,\mathbf{c}_t^{KV}, \qquad \mathbf{v}_t = W_{UV}\,\mathbf{c}_t^{KV}. \tag{1}$$

Let d_C and d_R denote the per-head content and decoupled-RoPE dimensions, respectively. For head h, define the absorbed content query

$$\mathbf{q}_i^{\mathrm{abs},h} \;\triangleq\; \big(W_{UK}^{(h)}\big)^\top \mathbf{q}_i^{C,h} \;\in\; \mathbb{R}^{d_c}. \tag{2}$$

The exact MLA attention logit contains both content and positional terms:

$$L_{ij}^{(h)} = \frac{\big(\mathbf{q}_i^{\mathrm{abs},h}\big)^\top \mathbf{c}_j^{KV} + \big(\mathbf{q}_i^{R,h}\big)^\top \mathbf{k}_j^R}{\sqrt{d_C + d_R}}. \tag{3}$$

CarveKV evaluates Eq. (3) directly against the cached latent $\mathbf{c}_j^{KV}$ and cached RoPE key $\mathbf{k}_j^R$. Weight absorption therefore avoids materializing an expanded content key $\mathbf{k}_j^C \in \mathbb{R}^{H\times 128}$ while retaining both terms of the model's MLA logit. For $i, j \in \mathcal{M}$, we causally mask the logit for head h:

$$\widehat{L}_{ij}^{(h)} = \begin{cases} L_{ij}^{(h)} & j \le i, \\ -\infty & j > i, \end{cases} \tag{4}$$

and set $P_{ij}^{(h)} = \mathrm{softmax}(\widehat{L}_{i,\cdot}^{(h)})_j$.

At each score layer ℓ, CarveKV chooses $Q_g = \min(512, n)$ deterministic midpoint queries spanning all n scored mid-region rows:

$$i_q = N_s + \left\lfloor \frac{(q+\frac{1}{2})n}{Q_g} \right\rfloor, \qquad q = 0, \ldots, Q_g - 1. \tag{5}$$

This prompt-wide stratification prevents a short contiguous observation window from defining the global score. The query-aware mass is

$$A^{(\ell)}[j] = \frac{1}{H} \sum_{h=1}^{H} \sum_{q=0}^{Q_g-1} P_{i_q j}^{(\ell,h)}. \tag{6}$$

As before, broad attention mass is modulated by the mean per-head ℓ_2 norm of the token's value projection (Edelman et al., 2024):

$$\nu^{(\ell)}[j] = \frac{1}{H} \sum_h \left\| W_{UV}^{(\ell,h)} \mathbf{c}_j^{KV} \right\|_2, \quad \bar{\nu}^{(\ell)}[j] = \frac{\nu^{(\ell)}[j]}{n^{-1} \sum_{u\in\mathcal{M}} \nu^{(\ell)}[u] + \varepsilon}, \quad \mathcal{I}^{(\ell)}[j] = A^{(\ell)}[j]\bar{\nu}^{(\ell)}[j]. \tag{7}$$

Sampling reduces score computation from $\mathcal{O}(Hn^2)$ to $\mathcal{O}(HKn)$. Projected values are computed only transiently for their norms and are never stored as an expanded KV cache.

3.2 Sink and Recency Token Protection

Two boundary regions are unconditionally retained and excluded from scoring. We protect the first N_s=4 attention sinks (Xiao et al., 2023; Sun et al., 2024) because deleting these highly attended tokens can destabilize decoding, not because they lack causal support. We also protect the last N_r=16 tokens (Zhang et al., 2023; Li et al., 2024; 2025a): few sampled global queries can attend to them, so $A^{(\ell)}[j]$ may understate their future decode-time utility. Appendix C gives the set definitions.

Table 1: CodeSearchNet/Python corpus-level PPL for shared masks selected at individual layers under a fixed 70% original-mid-token budget. The no-eviction baseline has PPL 6.8466. The independent layer-wise masking baseline has PPL 8.2162 and applies a separately selected binary attention mask at each layer. It is a heuristic baseline, not an oracle.

Layer	PPL	Gap vs Baseline	Gap vs Layer-wise
L0	26.5820	+19.7353	+18.3657
L1	13.8248	+6.9781	+5.6085
L2	15.4165	+8.5699	+7.2003
L4	9.1393	**+2.2926**	+0.9230
L6	8.2559	**+1.4093**	+0.0397
L8	8.3845	+1.5378	+0.1682
L12	7.3071	+0.4605	-0.9091
L16	7.6199	+0.7732	-0.5964

3.3 Cross-Layer Shared Indices and Multi-Layer Committee Decision

Cross-layer persistence motivates one shared index set $\mathcal{K}$: aligned importance rankings allow evidence from several depths to support one cache-wide decision. In our Hugging Face implementation, this also preserves a uniform physical layout compatible with DynamicCache's layer-wise concatenation (Wolf et al., 2019). This is an implementation benefit, not an MLA requirement. Let $k^\star \le n$ be the number of original mid tokens retained after accounting for protected entries and super tokens. A single-layer policy at $\ell^\star$ defines

$$\mathcal{K} = \mathrm{Top}_{k^\star}\left(\mathcal{I}^{(\ell^\star)}\right) \cup \mathcal{K}_{\mathrm{sink}} \cup \mathcal{K}_{\mathrm{recent}}, \tag{8}$$

where $\mathrm{Top}_{k^\star}(\cdot)$ retains the $k^\star$ highest-scored positions in $\mathcal{M}$, sink/recency tokens lie outside this budget.

Table 1 asks which layer best represents the others. Single-layer eviction improves sharply through L6 and then plateaus; together with the rank and overlap measurements below, this supports estimating a shared $\mathcal{K}$ from shallow and intermediate representations.

We use committee $\mathcal{S} = \{4, 6, 8, 12, 16\}$. Because raw scores are not comparable across depth, each becomes a percentile rank $r^{(\ell)}[j] \in [0, 1]$. Let $\mathcal{S}_2(j) \subset \mathcal{S}$ contain the two layers with the largest $r^{(\ell)}[j]$. The global score is

$$G[j] = \frac{1}{2}\left(\frac{1}{|\mathcal{S}|}\sum_{\ell\in\mathcal{S}} r^{(\ell)}[j] + \frac{1}{2}\sum_{\ell\in\mathcal{S}_2(j)} r^{(\ell)}[j]\right). \tag{9}$$

This construction is a robustness mechanism: single-layer rankings can vary across tasks and metrics. The committee reduces sensitivity to that choice rather than aiming to beat the best layer on every benchmark; selecting that layer separately per benchmark would be post-hoc tuning unavailable when the downstream workload is unknown.

3.4 Effective-Support Tail Routing

The global score can dilute tokens important to imminent decoding. At D=16, we compute $T[j]$ from the exact MLA attention weights observed over the final $Q_t = \min(512, N)$ prefill queries, not from future decode-time attention. For $j \in \mathcal{M}$ with nonzero total tail mass, let $p_j = T[j]/\sum_{u\in\mathcal{M}} T[u]$ and define

$$N_{\mathrm{eff}} = \exp\left(-\sum_{j\in\mathcal{M}} p_j \log p_j\right), \quad k_{\mathrm{tail}} = \min\left(k^\star, \left\lfloor N_{\mathrm{eff}} + \tfrac{1}{2}\right\rfloor\right). \tag{10}$$

Here $1 \le N_{\mathrm{eff}} \le n$, zero tail mass gives $k_{\mathrm{tail}} = 0$. The tail score selects this quota and G fills the remaining $k^\star - k_{\mathrm{tail}}$ slots. Pooling uses $\alpha r_T + (1-\alpha) r_G$, where $\alpha = k_{\mathrm{tail}}/k^\star$. At D=16, the keep set is applied retroactively to layers $0, \ldots, D$ and reused thereafter, retention is thus post-prefill rather than transient peak memory. Pseudocode is in Appendix D.

3.5 SOFT EVICTION VIA SUPER TOKENS

Because a low-scored token may matter later (Xiao et al., 2023), CarveKV merges evicted mid tokens into super tokens (Li et al., 2025b; Bolya et al., 2022). Each complete non-overlapping chunk of ρ_p evicted tokens yields one super token; any remainder of fewer than ρ_p tokens is dropped. All soft-eviction experiments use ρ_p=4. Thus

$$B_{\text{phys}} = N_s + N_r + k^\star + \left\lfloor \frac{n - k^\star}{\rho_p} \right\rfloor .$$

Latent and RoPE components use complementary pooling, matched-budget baselines use the same reconstruction. Details are in Appendix E.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Primary experiments use DeepSeek-Coder-V2-Lite (Zhu et al., 2024) (16B, 128K context, 27 layers, d_c=512). Transfer experiments use MiniCPM-3-4B (Hu et al., 2024), Qwen2.5-7B (Hui et al., 2024) converted with TransMLA (Meng et al., 2025), and Llama2-7B (Touvron et al., 2023) converted with MHA2MLA (Ji et al., 2025). Cross-model and earlier diagnostic runs use PyTorch 2.3.0 (Paszke et al., 2019), FlashAttention-2 (Dao, 2024), CUDA 12.1, and an NVIDIA vGPU-48GB or 80 GB H800. Tables 2 and 4 report BF16 measurements on the same AutoDL vGPU-48GB allocation under PyTorch 2.12.1+cu130. The fair fixed-budget test uses five warm-ups and 30 runs per method at 8K, 16K, and 32K, generates 512 tokens, and exactly matches the 60% retained physical latent/RoPE bytes across methods. CarveKV and SnapKV-style MLA are paired; H_2O-style MLA uses a separate run of the same protocol. Table 2 instead reports one optional heuristic configuration as a descriptive systems measurement, not as evidence that adaptive selection improves quality. CarveKV is training-free. Conversion details and the protocol matrix distinguishing the PPL diagnostics are in the Appendix G.

4.2 CROSS-LAYER IMPORTANCE PERSISTENCE EXPERIMENT

We probe persistence at ten backbone layers, {L0, L1, L2, L4, L6, L8, L10, L12, L16, L20}, on 40 independent 4096-token WikiText-103 and 40 CodeSearchNet/Python windows. Each layer uses the same 512 prompt-wide stratified queries and RoPE-complete MLA logits. For token j, let e_j be the number of sampled queries causally eligible to attend to it. We compare the raw value-aware score with its eligibility-normalized form on the common candidate set $e_j > 0$. Spearman $\rho(\ell_1, \ell_2)$ measures rank preservation, while top-30% keep-set overlap is

$$J(\ell_1, \ell_2) = \frac{|\mathcal{K}^{(\ell_1)} \cap \mathcal{K}^{(\ell_2)}|}{|\mathcal{K}^{(\ell_1)} \cup \mathcal{K}^{(\ell_2)}|}. \tag{11}$$

Across all 45 layer pairs, the mean raw Spearman correlation is 0.536 on natural language and 0.529 on code. Eligibility normalization increases these means to 0.651 and 0.654 rather than erasing the agreement. The corresponding top-30% Jaccard means increase from 0.381/0.380 to 0.498/0.498. For reference, independently selected 30% sets have the large-n reference $0.3/(2-0.3) \approx 0.176$. As a stricter position control, we compute correlation separately within eight contiguous token-position bins and average by bin size. The normalized within-bin means remain 0.507 and 0.505.

Within the committee $\mathcal{S} = \{4, 6, 8, 12, 16\}$, eligibility-normalized Spearman correlation is 0.731 on natural language and 0.737 on code, the within-bin values are 0.632 and 0.635, and normalized Jaccard overlap is 0.540 and 0.543. Thus the shared signal persists after controlling for causal eligibility and coarse position, while variation across layer pairs still supports aggregation rather than treating layers as interchangeable. Full definitions, domain-specific results, pairwise statistics, and controlled heatmaps are in the Appendix J.

These controls separate persistence from two trivial causes. Eligibility normalization removes the advantage of early positions visible to more sampled queries, while within-bin correlation compares

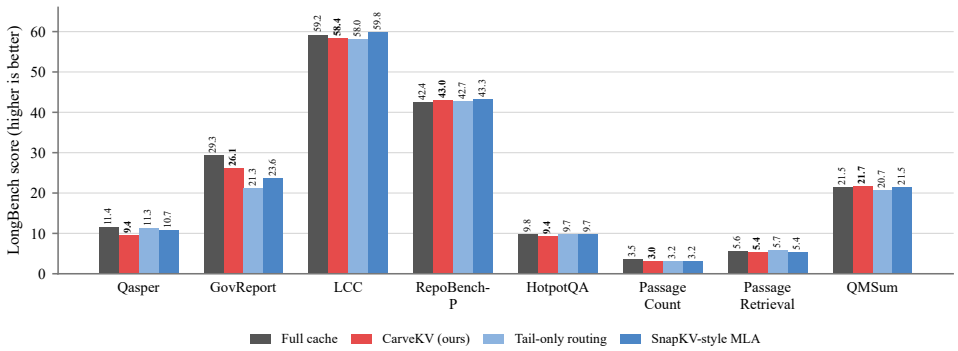


Figure 2: Task-level scores across all eight LongBench tasks (100 examples per task) at matched 60% physical cache retention ratio. The corresponding eight-task macro scores are 22.832, 22.049, 21.571, and 22.147 for full cache, CarveKV, tail-only routing, and SnapKV-style MLA, respectively. SnapKV-style MLA is our in-house adaptation.

tokens at similar coarse locations. The remaining agreement therefore reflects shared ranking structure rather than exposure alone, its higher value within $\mathcal{S}$ also supports forming the committee from intermediate layers.

4.3 Comprehensive Evaluation and Ablation Study

Seven configurations cover efficiency, language-modeling quality, multi-needle retrieval, and code cross-reference. Full protocols are in the Appendix H.

Our length-stratified evaluation on eight LongBench (Bai et al., 2024) tasks uses a fixed 60% physical cache retention ratio budget, 100 examples per task, and an 8192-token context cap. The evaluation includes Qasper, GovReport, LCC, RepoBench-P, HotpotQA, PassageCount, PassageRetrieval-en, and QMSum, covering long-document question answering, summarization, multi-hop reasoning, retrieval, and code understanding (Figure 2). This fixed-budget comparison is separate from the optional settings used in the descriptive diagnostics below.

Across all eight tasks, CarveKV scores 22.049 (96.6% of full-cache 22.832), improves on tail-only routing (21.571) by 0.478 points, and remains within 0.098 points of our in-house SnapKV-style MLA (22.147). On HotpotQA, PassageCount, PassageRetrieval-en, and QMSum, it retains 97.8% of full-cache quality (9.863 vs. 10.083) and is within 0.090 points of SnapKV-style MLA (9.954), supporting the global–tail design across heterogeneous workloads.

A complementary exact-byte sweep from 40–80% retention further shows that CarveKV supplies the 60% and 80% operating points on the joint quality–memory Pareto frontier. This sweep is separate from the evaluation summarized in Figure2: in particular, the 22.147 SnapKV-style MLA score in Figure2 and the 20.372 score at 80% retention in Appendix L are obtained under different experimental settings and are not points from the same sweep. At 80% retention, CarveKV reaches a macro score of 22.837 versus 20.372 for SnapKV-style MLA, leading on seven of eight tasks while using 20% fewer physical cache bytes than full cache. Full protocol and operating-point results are provided in Appendix L.

Because all methods retain the same physical number of entries, these results primarily compare selection quality. On GovReport, CarveKV reaches 26.146, versus 23.569 for the SnapKV-style MLA and 21.277 for tail-only routing, indicating that prompt-wide global coverage is especially useful for long-form summarization. On the two code tasks, CarveKV retains 99.7% of the full-cache average. On the QMSum task, it reaches 21.669, above both the MLA-adapted baseline (21.532) and full cache (21.506). Detailed task scores, router allocations, and paired intervals are in the supplement. System tests span 4K–64K contexts (Table 2). Full curves are in the Appendix M.

Across 4K–64K, the optional $\tau 60$ configuration retains 53.0–64.2% of the cache under a 0.60 coverage floor and length-aware physical bounds. This is a descriptive realized-retention range, not a comparison against an equal-average fixed budget. Except for the 4K point, single-run prefill mea-

Table 2: Descriptive system metrics for the optional adaptive coverage-floor configuration (τ=0.60, ρ_p=4) and the MLA baseline across context lengths on an AutoDL vGPU-48GB. Compress = retained KV fraction (%), Prefill = prefill latency (ms), One-step decode = the latency of one cached single-token forward after prefill (ms; not end-to-end TTFT), and Retained KV = physical size of the post-prefill cache tensors after eviction (GB), not peak allocated GPU memory. Arrows indicate that lower is better.

Context (tokens)	Configuration	Compress↓ (%)	Prefill↓ (ms)	One-step decode↓ (ms)	Retained KV↓ (GB)
4096	baseline	100.0	986	76.6	0.119
	τ60	**64.2**	583	38.8	**0.076**
8192	baseline	100.0	1936	47.7	0.237
	τ60	**62.0**	1972	43.1	**0.147**
16384	baseline	100.0	6749	51.3	0.475
	τ60	**59.0**	6849	40.0	**0.280**
32768	baseline	100.0	25262	52.8	0.949
	τ60	**56.0**	25486	42.1	**0.532**
65536	baseline	100.0	110052	66.7	1.898
	τ60	**53.0**	110461	44.3	**1.006**

Table 3: Suffix PPL on CodeSearchNet/Python and Multi-needle NIAH recall. Baseline PPL is 2.495. Each NIAH entry contains 100 trials across five depths. The compressed rows are descriptive configurations, not an adaptive-versus-fixed budget comparison. τ90 uses committee scoring, τ=0.90, and ρ_p=4, hd_τ90 disables pooling, while single_L6/L12 replaces the committee with the indicated layer under the same τ90 budget rule. Arrows indicate metric direction (↓ lower is better, ↑ higher is better).

Configuration	Suffix PPL↓	NIAH Recall↑	
		$ctx = 4096$	$ctx = 8192$
Baseline	2.495	92% (92/100)	93% (93/100)
τ90	**2.515**	83% (83/100)	90% (90/100)
hd_τ90	2.516	84% (84/100)	90% (90/100)
single_L6	2.522	90% (90/100)	91% (91/100)
single_L12	2.523	87% (87/100)	82% (82/100)

surements remain within 2.1% of baseline, while one-step cached decode latency is 38.8–44.3 ms versus 47.7–66.7 ms for baseline. The matched CarveKV–SnapKV-style MLA comparison in Table 4 isolates policy overhead at the same physical cache size and generation length.

The near-linear cache savings and consistent one-step decode improvement are expected from reducing the state scanned during decoding. These are post-prefill measurements: because the shared decision is made at D=16, earlier layers temporarily retain the full prefix.

Table 3 reports mean window-level suffix PPL and multi-needle NIAH recall from 100 strict-match trials per context (five depths and twenty target values). Full suffix-scoring, target, distractor, seed, and answer-extraction protocols are in the Appendix H.

τ90 has the lowest compressed PPL (2.515), only 0.020 above full cache. The calibrated NIAH protocol gives full-cache recall of 92%/93% at 4K/8K, while the displayed compressed variants span 83–90%/82–91%. Among compressed variants, single_L6 is highest at 4K and 8K. Results for τ85 and fixed_ratio_70 are retained in the Appendix H.

The reversal between τ90's lower PPL and single_L6's higher NIAH recall shows that the preferred scoring choice depends on the metric. This motivates a fixed committee as a robust, task-agnostic choice rather than post-hoc selection of the best layer for each benchmark.

Table 4: Fair MLA-native systems comparison on one NVIDIA vGPU-48GB. All methods retain exactly the same physical latent/RoPE cache bytes (60%), use the same context and 512-token generation, and avoid persistent expanded KV. Values are means over 30 runs after five warm-ups. CarveKV/SnapKV are paired. Prefill covers the prompt forward through the final-position vocabulary projection. Selecting the first token adds sub-millisecond overhead, so the nearly redundant GPU TTFT column is omitted. Lower is better except for decode throughput.

Context (tokens)	Method	Cache (MiB)↓	Peak (GiB)↓	Prefill (s)↓	Decode (tok/s)↑	E2E (s)↓
8192	CarveKV (ours)	145.8	30.78	**1.962**	**22.95**	**24.243**
	SnapKV-style MLA	145.8	30.78	2.022	22.83	24.428
	H_2O-style MLA	145.8	30.78	2.044	22.89	24.368
16384	CarveKV (ours)	291.6	32.29	**6.805**	22.37	**29.654**
	SnapKV-style MLA	291.6	32.29	6.924	22.49	29.747
	H_2O-style MLA	291.6	32.29	6.989	**22.43**	29.771
32768	CarveKV (ours)	583.2	35.32	**25.363**	22.45	**48.127**
	SnapKV-style MLA	583.2	35.32	25.623	**22.47**	48.362
	H_2O-style MLA	583.2	35.32	25.801	22.46	48.553

4.4 Comparison with Eviction References

We perform systems-equivalent comparisons against unofficial SnapKV-style and H_2O-style MLA adaptations on DeepSeek-Coder-V2-Lite and one NVIDIA vGPU-48GB. SnapKV-style ranks latents with exact absorbed-content and decoupled-RoPE logits over the final 512 prefill queries. H_2O-style instead accumulates this exact MLA attention mass over 512 prompt-wide stratified queries and protects a recent window. Neither adaptation persists expanded KV; both use a shared top-k mask. All methods retain exactly the same physical latent/RoPE bytes, process the same 8K, 16K, and 32K prompts, and generate the same 512 tokens. These are MLA-native adaptations for controlled comparison, not the official implementations (Zhang et al., 2023; Li et al., 2024).

Table 4 shows that CarveKV reduces mean prefill latency by 3.0%, 1.7%, and 1.0% at 8K, 16K, and 32K relative to SnapKV-style MLA, the paired 95% intervals exclude zero. End-to-end latency is 0.8%, 0.3%, and 0.5% lower. Against prompt-wide H_2O-style scoring, CarveKV lowers prefill latency by 4.0%, 2.6%, and 1.7% with identical cache bytes and near-identical peak allocation. Decode throughput differs by at most 0.3%, while end-to-end latency falls by 0.5%, 0.4%, and 0.9%. Together with the matched-budget quality results, these measurements show no systems penalty for the global committee and effective-support router relative to either reference.

The quality and systems results answer complementary questions. LongBench and the exact-byte sweep test whether the shared router selects useful entries at controlled storage, whereas Table 4 fixes cache bytes and generation length to expose policy overhead. Together, they indicate that CarveKV's quality retention is not purchased through a larger physical cache or slower decoding.

5 Conclusion

CarveKV is a training-free, MLA-native cache-eviction framework motivated by the empirical cross-layer persistence of latent-token importance. It scores resident latent and decoupled-RoPE caches with the complete MLA logit and a value-aware term, avoiding reconstruction of expanded per-head KV tensors. A rank-normalized committee produces a shared keep set, while effective-support routing divides the mid-token budget between prompt-wide coverage and observed final-window prefill attention. Protected sink and recent tokens preserve stable context, and soft eviction summarizes removed tokens as budget-accounted super tokens. Under controlled physical-cache budgets, CarveKV retains 96.6% of full-cache LongBench quality at 60% retention, remains within 0.098 macro points of an MLA-native SnapKV-style adaptation, and reduces prefill latency by 1.0–3.0% at 8K–32K with comparable peak memory and decode throughput. Future work will reduce transient prefill memory through earlier decisions and fused scoring-pooling kernels, support batched variable-length routing, evaluate broader MLA architectures and workloads, and investigate failure modes of pooled RoPE summaries and safe decode-time mask updates.

AI USE STATEMENT

Generative AI tools were used for language editing, structural revision, and code-review assistance. The authors reviewed the resulting text, equations, claims, and code and take responsibility for the final content and artifacts.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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)*, pp. 3119–3137, 2024.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. Nacl: A general and effective kv cache eviction framework for llm at inference time. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7913–7926, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

Keqi Deng and Phil Woodland. Multi-head temporal latent attention. *Advances in Neural Information Processing Systems*, 38:162828–162851, 2026.

Ezra Edelman, Nikolaos Tsilivis, Surbhi Goel, Benjamin L Edelman, et al. The evolution of statistical induction heads: In-context learning markov chains. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

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

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling deepseek's multi-head latent attention in any transformer-based llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 33313–33328, 2025.

Minchul Kim, Shangqian Gao, Yen-Chang Hsu, Yilin Shen, and Hongxia Jin. Token fusion: Bridging the gap between token pruning and token merging. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1372–1381. IEEE, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Runchao Li, Yao Fu, Mu Sheng, Xianxuan Long, Haotian Yu, and Pan Li. Faedkv: Infinite-window fourier transform for unbiased kv cache compression. *arXiv preprint arXiv:2507.20030*, 2025a.

Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *International Journal of Computer Vision*, 133(10):6794–6812, 2025b.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

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

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Fanxu Meng, Pingzhi Tang, Xiaojuan Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. Transmla: Multi-head latent attention is all you need. *arXiv preprint arXiv:2502.07864*, 2025.

Yongyu Mu, Yuzhang Wu, Yuchun Fan, Chenglong Wang, Hengyu Li, Qiaozhi He, Murun Yang, Tong Xiao, and Jingbo Zhu. Cross-layer attention sharing for large language models. *arXiv preprint arXiv:2408.01890*, 2024.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M Ponti. Dynamic memory compression: Retrofitting llms for accelerated inference. *arXiv preprint arXiv:2403.09636*, 2024.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state rnns. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 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.

Mingjie Sun, Xinlei Chen, J Zico Kolter, and Zhuang Liu. Massive activations in large language models. *arXiv preprint arXiv:2402.17762*, 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.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

He Wang, Yu Gu, Fangfang Li, Zhigang Wang, Zhenghao Liu, Ning Wang, Xiaohua Li, and Ge Yu. Hqekv: Towards hybrid quantization and eviction for kv cache in long-context llm inference. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4138–4153, 2026.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. *URL https://arxiv. org/abs/2309.17453*, pp. 1, 2023.

Xintong Yang, Hao Gu, Binxing Xu, Lujun Li, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Yike Guo, and Sirui Han. Indexmem: Learned kv-cache eviction with latent memory for long-context llm inference. *arXiv preprint arXiv:2605.25475*, 2026.

Zeng You, Yaofo Chen, Qiuwu Chen, Ying Sun, Shuhai Zhang, Yingjian Li, Yaowei Wang, and Mingkui Tan. Latent-condensed transformer for efficient long context modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25656–25671, 2026.

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

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y Wu, Yukun Li, Huazuo Gao, Shirong Ma, et al. Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024.