TRIDENTKV: OPTIMIZING KV CACHE EVICTION WITH DIVERSITY-AWARE BUDGET ALLOCATION AND SPATIAL-TEMPORAL SCORE AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context inference in large language models (LLMs) is increasingly bottlenecked by the growing memory and computation cost of the key-value (KV) cache. While KV cache eviction provides a training-free solution, existing methods typically optimize cache allocation or importance aggregation without explicitly accounting for behavioral diversity across layers and attention heads or localized variations in token importance. We introduce TridentKV, a training-free KV cache eviction framework that improves cache utilization through diversity-aware budget allocation and spatial-temporal score aggregation. TridentKV consists of three complementary components: (1) uniqueness-based layer budget allocation, which assigns larger cache budgets to layers exhibiting distinctive attention behaviors; (2) behavior-guided head budget allocation, which clusters attention heads by behavioral similarity and restricts budget competition within each cluster to preserve diverse attention patterns; and (3) spatial-temporal token score aggregation, which jointly captures importance variations across head clusters and query window to preserve locally salient tokens. Experiments on LongBench, RULER, and Needle-in-a-Haystack across Llama, Mistral, and Qwen models show that TridentKV robustly preserves long-context performance under constrained cache budgets. With a cache budget of 512, it retains over 99.6% of full-cache performance on LongBench and outperforms the strongest evaluated baselines on 32K RULER by 2.7%–5.4%, while achieving up to 2.76× decoding speedup and a 36.2% reduction in peak memory at 128K context length. We provide our code in the supplementary material to facilitate straightforward reproduction of our method.

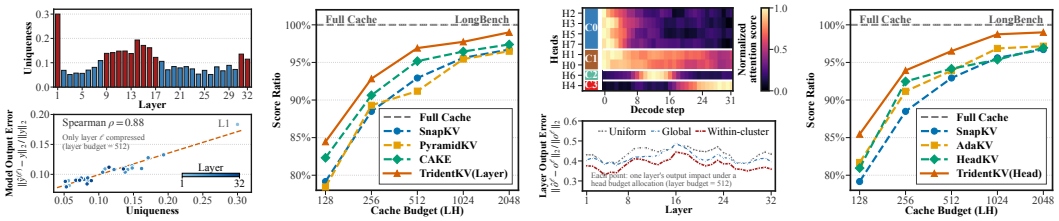
1 INTRODUCTION

Large language models (LLMs) (Achiam et al., 2023) have become a dominant paradigm for multi-turn dialogue (Glm et al., 2024), question answering (Kamalloo et al., 2023), and code generation (Guo et al., 2023), etc. However, as context windows expand, efficient inference is increasingly limited by the key-value (KV) cache, whose memory consumption scales linearly with sequence length and incurs substantial memory-access overhead. To alleviate this bottleneck, recent studies have explored KV cache compression techniques, such as low-bit quantization (Liu et al., 2024e; Hooper et al., 2024), low-rank projection (Chang et al., 2025; Bhatnagar et al., 2026), and KV cache eviction, etc. Among them, KV cache eviction is particularly appealing because it requires no model retraining while reducing both cache memory and attention computation.

Earlier KV cache eviction methods (Zhang et al., 2023; Li et al., 2024; Zhang et al., 2025) primarily focused on importance scoring, often using multiple recent queries (query window) to obtain diverse importance estimates for past KV entries. Recent studies have extended this focus to budget allocation across layers or attention heads and score aggregation for combining multiple observed scores for each cache entry. This shift reflects a broader transition from determining what to retain toward jointly considering where to allocate cache capacity and how to aggregate retention evidence.

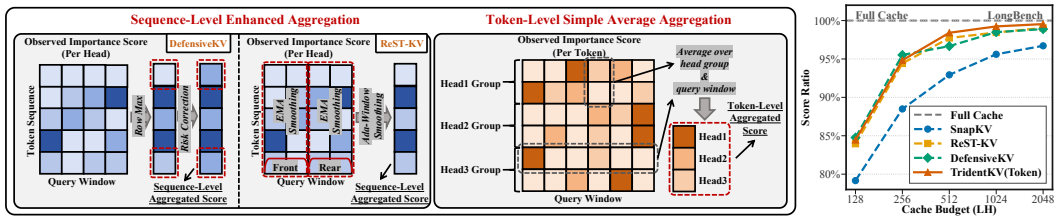
Existing budget allocation methods have explored non-uniform cache distribution at both the layer and head levels. Layer-wise methods (Cai et al., 2024; Qin et al., 2025) assign different capacities based on layer-specific attention characteristics or importance, while head-wise methods (Feng et al.,

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.



(a) Layer uniqueness. (b) Performance. (c) Head clusters. (d) Performance.

Figure 1: Behavioral diversity across layers and attention heads, its influence on model/layer outputs, and the performance of TridentKV in leveraging such diversity on Llama-3.1-8B-Instruct.



(a) Previous score aggregation at the sequence and token level. (b) Performance.

Figure 2: Prior score aggregation overlooks spatial-temporal characteristics, whereas TridentKV captures both dimensions to achieve improved performance (Llama-3.1-8B-Instruct).

2026; Fu et al., 2025) redistribute budgets according to head-level importance. However, these approaches mainly rely on unit-specific signals or score magnitudes, without explicitly accounting for behavioral diversity. This limitation is particularly critical under aggressive compression, where allocating excessive capacity to units with similar behaviors wastes scarce cache resources and may underrepresent distinctive ones. Figure 1a shows that layer attention uniqueness varies substantially and strongly correlates with the output impact of layer-wise compression, motivating larger budgets for more distinctive layers. At the head level, Figure 1c shows that attention heads exhibit clustered behavioral patterns, and restricting budget competition within behaviorally similar groups yields lower layer output error than uniform or global allocation. These observations motivate diversity-aware budget allocation that preserves complementary behaviors at both layer and head levels.

While budget allocation determines cache capacity distribution, reliable eviction also depends on how token importance is aggregated. As illustrated in Figure 2a, early eviction methods (Li et al., 2024; Zhang et al., 2025) simply average observations across the query window and shared query heads, which can dilute a strong but localized response. Recent approaches (An et al., 2026; Guo et al., 2026) improve upon this by adopting max-based aggregation with risk correction or modeling smooth importance variations across token positions, but they primarily focus on sequence-level information, leaving token-level signal largely unexplored. Under GQA (Grouped Query Attention (Ainslie et al., 2023)), token-level importance is jointly characterized by the temporal dimension of the query window and the spatial dimension of query heads sharing the same KV head. Explicitly modeling importance across both dimensions can prevent potentially important tokens from being discarded when local salience is obscured. This motivates a spatial-temporal aggregation scheme that yields more robust token-level importance estimates and more reliable eviction decisions.

Building upon these observations, we propose TridentKV, a training-free KV cache eviction framework that jointly considers diversity-aware budget allocation across layers and heads and spatial-temporal token score aggregation. At the layer level, TridentKV allocates cache budgets according to cross-layer attention uniqueness, assigning more capacity to layers with distinctive attention behaviors. Within each layer, behavior-guided head clustering restricts budget competition to heads with similar attention behaviors, reserving capacity for each behavioral group to support behavioral diversity. Within each head cluster, a 2D spatial-temporal pooling scheme is employed to jointly aggregate observed importance across the query window and shared query heads, reducing the risk of discarding tokens with locally salient signals. Together, these three components form the three prongs of TridentKV, enabling more effective and robust KV cache eviction while remaining orthogonal to prior work focused on importance scoring mechanisms.

We evaluate TridentKV on LongBench (Bai et al., 2024), RULER (Hsieh et al., 2024), and Needle-in-a-Haystack (Liu et al., 2024b) using three models from the Llama (Grattafiori et al., 2024), Mis-

tral (Jiang et al., 2023), and Qwen (Bai et al., 2023) families, demonstrating its effectiveness in preserving long-context capabilities under constrained cache budgets. With a cache budget of 512, TridentKV retains over 99.6% of the full-cache average score on LongBench across all three models and improves the average score on 32K RULER by 2.7%–5.4% relative to the strongest evaluated baseline for each model. At the same budget, TridentKV achieves a 2.76× decoding speedup and reduces peak memory by 36.2% relative to full-cache inference at a context length of 128K. Ablation studies further show that each of the three components improves the baseline individually, with their combination achieving the best overall performance, supporting the joint design of diversity-aware budget allocation and spatial-temporal score aggregation.

2 Related Work

KV cache eviction has emerged as an effective approach to reduce the memory and computational overhead of long-context LLM inference. Early methods such as StreamingLLM (Xiao et al., 2024b) retain only recent tokens, potentially discarding useful information. Scissorhands (Liu et al., 2023) preserves persistently important tokens, while H2O (Zhang et al., 2023) combines recent tokens with attention-based "heavy hitters". SnapKV (Li et al., 2024) identifies important positions from a short observation window, and ClusterAttn (Zhang et al., 2025) exploits clustering structures in attention distributions for cache eviction. More recent methods explore importance criteria beyond raw attention scores. For example, CriticalKV (Feng et al., 2025) estimates KV importance through output perturbation, jointly incorporating attention responses and value information.

Another line of work studies cache budget allocation across layers and attention heads. PyramidKV (Cai et al., 2024) exploits layer-wise differences in information concentration, while CAKE (Qin et al., 2025) dynamically allocates cache according to layer-specific spatial and temporal attention characteristics. At the head level, AdaKV (Feng et al., 2026) improves cache utilization through adaptive global redistribution, and HeadKV (Fu et al., 2025) assigns larger budgets to heads that are more important for contextual retrieval and reasoning. Although these approaches outperform uniform allocation, they mainly assess heads or layers independently and do not explicitly account for behavioral diversity across layers and attention heads.

Recent studies further improve eviction robustness through score aggregation. Unlike the average aggregation commonly used in prior work, DefensiveKV (Guo et al., 2026) adopts worst-case-oriented aggregation to reduce the risk of discarding locally important tokens, whereas ReST-KV (An et al., 2026) combines exponential moving average smoothing and adaptive observation window to obtain more stable importance estimates. However, under GQA, each KV head serves multiple query heads, making token-level importance jointly dependent on cross-head responses and temporal query dynamics. Approaches above both operate at the sequence level, without explicitly modeling these signals. In contrast, TridentKV optimizes across granularities via uniqueness-based layer budget allocation, behavior-guided head budget allocation and spatial-temporal token score aggregation, thereby preserving complementary attention behaviors and enhancing eviction robustness.

3 Methodology

3.1 Preliminary

KV Cache in LLM Generation. Autoregressive inference comprises prefilling and decoding. Under GQA, layer ℓ has H KV heads, each shared by G query heads, indexed by $h \in \{1, \dots, H\}$ and $g \in \{1, \dots, G\}$, respectively. Prefilling caches $\mathbf{K}_h^\ell = \mathbf{H}^\ell \mathbf{W}_{K,h}^\ell \in \mathbb{R}^{n \times d_h}$ and $\mathbf{V}_h^\ell = \mathbf{H}^\ell \mathbf{W}_{V,h}^\ell \in \mathbb{R}^{n \times d_h}$ from input hidden states $\mathbf{H}^\ell \in \mathbb{R}^{n \times d}$, where n is the input length. At decoding step t, hidden state $\mathbf{h}_t^\ell$ produces query $\mathbf{q}_{h,g,t}^\ell = \mathbf{h}_t^\ell \mathbf{W}_{Q,h,g}^\ell \in \mathbb{R}^{1 \times d_h}$ and each query attends to its shared cache:

$$\mathbf{A}_{h,g,t}^\ell = \mathrm{softmax}\left(\frac{\mathbf{q}_{h,g,t}^\ell (\mathbf{K}_h^\ell)^\top}{\sqrt{d_h}} \right) \tag{1}$$

Head outputs $\mathbf{z}_{h,g,t}^\ell = \mathbf{A}_{h,g,t}^\ell \mathbf{V}_h^\ell$ are concatenated and projected as $\mathbf{o}_t^\ell = \mathrm{Concat}_{h,g}(\mathbf{z}_{h,g,t}^\ell) \mathbf{W}_O^\ell$. Each step appends new KV states to the caches for subsequent decoding. By reusing historical KV states, KV caching avoids redundant computation and substantially accelerates autoregressive

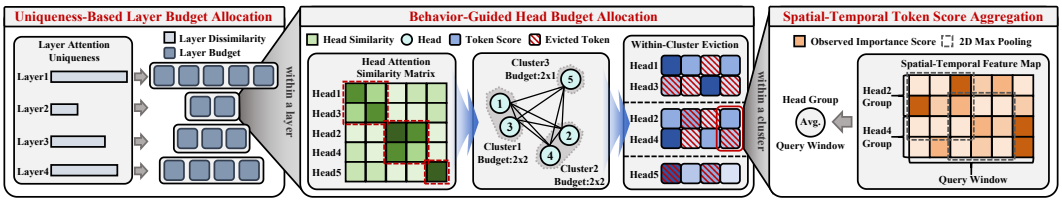


Figure 3: Overview of TridentKV. The framework combines uniqueness-based layer budget allocation, behavior-guided head budget allocation, and spatial-temporal token score aggregation to preserve complementary attention behaviors and enhance eviction robustness.

generation. However, the cache grows linearly with sequence length, while each decoding step must retrieve and attend to an increasing number of KV entries, leading to higher memory consumption, memory-access overhead, and attention cost, particularly in long-context inference.

KV Cache Eviction. KV cache eviction mitigates the above overhead by replacing the full cache $(\mathbf{K}_h^\ell, \mathbf{V}_h^\ell)$ with a compressed cache $(\widehat{\mathbf{K}}_h^\ell, \widehat{\mathbf{V}}_h^\ell)$ that preserves information relevant to future queries. Existing methods can be broadly characterized by three components: importance scoring, budget allocation, and score aggregation, which determine what to retain, where to allocate cache capacity, and how to combine multiple importance observations, respectively:

- **Importance Scoring.** To estimate the relevance of cached entries, A short query window comprising the last m queries in the prompt, $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_m]$, is commonly used. Each query–entry pair receives an importance score based on the query $\mathbf{q}_j$ and the cached key–value pair $(\mathbf{k}_i, \mathbf{v}_i)$. Collectively, these scores form an observation matrix $\mathbf{I} \in \mathbb{R}^{H\times G\times m\times n}$, where $I_{h,g,j,i}$ denotes the importance of cache entry i to the query j of group g in head h.
- **Budget Allocation.** Given an overall cache budget B_{total}, an allocation rule distributes the available capacity across layers and attention heads. The resulting budget $B_{\ell,h}$ specifies the number of KV entries retained by head h in layer ℓ and satisfies $\sum_{\ell,h} B_{\ell,h} = B_{\text{total}}$.
- **Score Aggregation.** Existing methods typically aggregate the observations in $\mathbf{I}$ by averaging over the query window and the query heads within each group, yielding per-entry scores $S \in \mathbb{R}^{H\times n}$. For each KV head h in layer ℓ, entries are ranked by these scores, and the $B_{\ell,h}$ highest-scoring entries are retained in the compressed cache.

In this work, we target widely used eviction methods that derive importance scores from attention weights, with a focus on optimizing budget allocation and score aggregation to improve cache utilization and eviction reliability under constrained budgets.

3.2 Uniqueness-Based Layer Budget Allocation

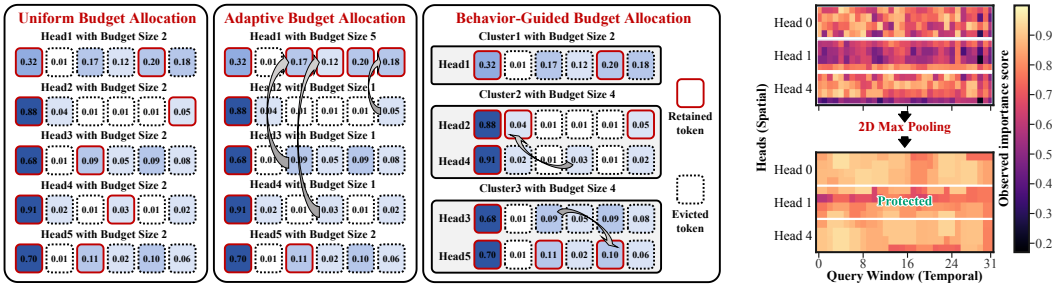
The first prong of TridentKV allocates a limited cache budget across layers. Existing methods (Cai et al., 2024; Qin et al., 2025) typically assess layers separately using signals such as attention concentration or entropy, overlooking cross-layer behavioral diversity. Under a constrained budget, this can under-allocate capacity to layers with distinctive attention patterns. To promote coverage of diverse attention behaviors, we assign more capacity to layers with distinctive attention patterns.

We characterize each layer by its attention distribution over a recent query window, which is also the common signal used by mainstream KV eviction methods to estimate token importance. Let $\mathbf{A}_{h,g}^\ell \in \mathbb{R}^{m\times n}$ denote the attention between the query window of query head g and the n historical keys associated with KV head h at layer ℓ. We represent the attention behavior of layer ℓ as:

$$\mathbf{A}^\ell = [\mathbf{A}_{h,g}^\ell]_{h=1,g=1}^{H,G} \in \mathbb{R}^{H\times G\times m\times n}, \qquad \mathbf{A}_{h,g}^\ell = \text{softmax}\left(\frac{\mathbf{Q}_{h,g}^\ell (\mathbf{K}_h^\ell)^\top}{\sqrt{d_h}}\right) \tag{2}$$

where H is the number of KV heads and each serves G query heads under GQA. We then measure the uniqueness of each layer relative to the layer with the most similar attention pattern. Specifically, we define the uniqueness score of each layer as the minimum cosine dissimilarity between its attention behavior and that of any other layer:

$$u_\ell = 1 - \max_{r\neq\ell} \frac{\langle \mathbf{A}^\ell, \mathbf{A}^r \rangle_F}{\|\mathbf{A}^\ell\|_F \|\mathbf{A}^r\|_F} \tag{3}$$



(a) Head budget competition under different allocation methods. (b) 2D feature map.

Figure 4: Details of behavior-guided budget allocation and spatial-temporal score aggregation.

where $\langle \cdot, \cdot \rangle_F$ and $\|\cdot\|_F$ denote the Frobenius inner product and norm, respectively. Under this nearest-neighbor criterion, a high uniqueness score indicates that a layer's attention pattern remains distinct even from its most similar peer, while a low score indicates close similarity to another layer. We use uniqueness as a demand weight to prioritize capacity for distinctive attention patterns.

We retain the query window in every KV head, reserving LHm entries within B_{total}, where L is the number of layers. The remaining budget is allocated in proportion to the uniqueness scores:

$$B_\ell = Hm + B_\ell^{\text{free}}, \qquad B_\ell^{\text{free}} = B_{\text{free}} \frac{u_\ell}{\sum_{r=1}^{L} u_r}, \qquad B_{\text{free}} = B_{\text{total}} - LHm \tag{4}$$

Thus, $\sum_\ell B_\ell = B_{\text{total}}$. This allocation rule balances free cache capacity per unit of uniqueness demand, prioritizing layers with distinctive attention patterns. By incorporating cross-layer attention diversity into budget allocation, TridentKV aims to improve layer-level cache utilization and support coverage of diverse layer behaviors.

3.3 Behavior-Guided Head Budget Allocation

The second prong distributes the layer budget B_ℓ across its H KV heads. Conventional strategies exhibit opposite limitations: a uniform split (Li et al., 2024; Zhang et al., 2025) prevents capacity from adapting to heads with stronger token importance, whereas unrestricted global competition (Feng et al., 2026; Fu et al., 2025) allows similar high-scoring heads to crowd out behaviorally distinctive ones. We therefore restrict budget competition to clusters of heads with similar attention patterns, reserving capacity for each cluster to support coverage across behavioral groups.

We first derive the attention behavior of each KV head from Eq. 2 by averaging the G query heads that share its keys and values:

$$\bar{\mathbf{A}}_h^\ell = \frac{1}{G} \sum_{g=1}^{G} \mathbf{A}_{h,g}^\ell \in \mathbb{R}^{m \times n} \tag{5}$$

For heads h and h', we then quantify their behavioral similarity and distance using cosine similarity:

$$c_{hh'}^\ell = \frac{\langle \bar{\mathbf{A}}_h^\ell, \bar{\mathbf{A}}_{h'}^\ell \rangle_F}{\|\bar{\mathbf{A}}_h^\ell\|_F \|\bar{\mathbf{A}}_{h'}^\ell\|_F}, \quad d_{hh'}^\ell = 1 - c_{hh'}^\ell \tag{6}$$

Next, we perform complete-link agglomerative clustering (Defays, 1977) using a layer-adaptive threshold. For clusters C_a and C_b, the complete-link distance and the threshold for layer ℓ are:

$$D_\ell(C_a, C_b) = \max_{h \in C_a,\, h' \in C_b} d_{hh'}^\ell, \quad \tau_\ell = \frac{2}{H(H-1)} \sum_{1 \le h < h' \le H} d_{hh'}^\ell \tag{7}$$

Starting from H singleton clusters, we repeatedly merge the closest pair as long as $D_\ell(C_a, C_b) \le \tau_\ell$ for that pair, producing partition $\mathcal{C}_\ell$. Complete linkage prevents chains of locally similar heads from collapsing into diffuse clusters, while the mean pairwise distance adapts the stopping criterion to each layer's behavioral diversity without requiring a predefined number of clusters.

The resulting partition decomposes B_ℓ into independent cluster-level budget pools, assigning each cluster $C \in \mathcal{C}_\ell$ capacity in proportion to its number of heads:

$$B_{\ell,C} = \frac{|C|}{H} B_\ell, \qquad \sum_{C \in \mathcal{C}_\ell} B_{\ell,C} = B_\ell, \qquad \sum_{h \in C} B_{\ell,h} = B_{\ell,C} \tag{8}$$

The allocatable cluster budget is $B^{\text{free}}_{\ell,C} = B_{\ell,C} - |C|m = \frac{|C|}{H} B^{\text{free}}_{\ell}$. As shown in Figure 4a, heads compete for $B^{\text{free}}_{\ell,C}$ only within their own cluster, and capacity cannot move across clusters. Within each cluster, token importance determines the distribution of capacity among heads, while separate cluster budgets support coverage across behavioral groups. Taken together, the layer- and head-level stages form TridentKV's diversity-aware budget allocation, which uses attention diversity to guide capacity allocation and promote broad behavioral coverage under a fixed memory budget.

3.4 Spatial-Temporal Token Score Aggregation

The third prong determines which head–token pairs occupy the free cluster budget $B^{\text{free}}_{\ell,C}$. Directly averaging observations over the query window and the query heads associated with each KV head may suppress locally salient responses, while aggregating across all heads may involve responses arising from behaviorally unrelated heads. To preserve local salience without mixing distinct behaviors, we perform a robust aggregation over decoupled spatial-temporal domains, treating the query window as the temporal domain and each cluster as an independent spatial domain.

For cached token i and cluster $C \in \mathcal{C}_\ell$, we arrange the unaggregated token-level importance scores from Eq. 2 into a cluster-local spatial-temporal feature map (Figure 4b):

$$\mathbf{X}^{\ell,C}_i[\phi_C(h,g),t] = \mathbf{A}^{\ell}_{h,g}[t,i], \qquad \mathbf{X}^{\ell,C}_i \in \mathbb{R}^{|C|G\times m} \tag{9}$$

where $h \in C$, $g \in \{1,\dots,G\}$, $t \in \{1,\dots,m\}$, and ϕ_C maps (h,g) to a row under a fixed cluster-local ordering that preserves the original head order (see Appendix G for an ablation). We then apply a 2D max pooling to the cluster-local spatial-temporal feature map and average the pooled responses to obtain the spatial-temporal score for each head–token pair:

$$\mathbf{P}^{\ell,C}_i = \text{MaxPool}_{2\text{D}}\left(\mathbf{X}^{\ell,C}_i; k\times k\right), \qquad \mathbf{P}^{\ell,C}_i \in \mathbb{R}^{|C|G\times m} \tag{10}$$

$$S_{\ell,h,i} = \frac{1}{Gm}\sum_{g=1}^{G}\sum_{t=1}^{m} \mathbf{P}^{\ell,C(h)}_i\left[\phi_{C(h)}(h,g),t\right], \qquad h \in C(h) \tag{11}$$

where $C(h)$ is the cluster containing head h and we use kernel size $k = 3$ in our implementation. 2D max pooling here jointly captures spatial-temporal responses, preventing locally salient signals from being immediately diluted, thereby reducing the risk of erroneously evicting potentially important tokens. The resulting spatial-temporal score then resolves the within-cluster allocation from Section 3.3. Let $\mathcal{W} = \{n-m+1,\dots,n\}$ denote the retained query window. For each cluster C, we rank head–token pairs outside $\mathcal{W}$ and select the $B^{\text{free}}_{\ell,C}$ highest ones:

$$\mathcal{I}_{\ell,h} = \left\{i : (h,i) \in \mathcal{R}_{\ell,C(h)}\right\} \cup \mathcal{W}, \qquad \mathcal{R}_{\ell,C} = \text{TopK}_{B^{\text{free}}_{\ell,C}}\left(\{S_{\ell,h,i}\}_{(h,i)\in C\times([n]\setminus\mathcal{W})}\right) \tag{12}$$

Here, $[n] = \{1,\dots,n\}$, $\mathcal{R}_{\ell,C} \subseteq C \times ([n] \setminus \mathcal{W})$ is the set of selected prefix head–token pairs, and $\mathcal{I}_{\ell,h}$ contains all retained token indices for head h. Thus, the compressed KV cache is as follows:

$$\widehat{\mathbf{K}}^{\ell}_h = \mathbf{K}^{\ell}_h[\mathcal{I}_{\ell,h},:], \qquad \widehat{\mathbf{V}}^{\ell}_h = \mathbf{V}^{\ell}_h[\mathcal{I}_{\ell,h},:] \tag{13}$$

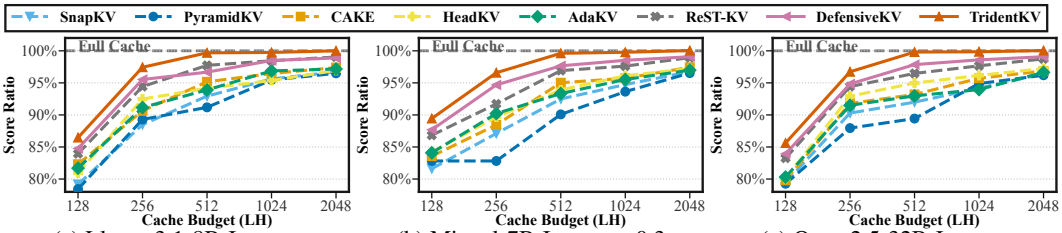
The number of retained tokens may vary across heads within the same cluster, while each cluster exactly satisfies $\sum_{h\in C}|\mathcal{I}_{\ell,h}| = B_{\ell,C}$, and $\sum_{C\in\mathcal{C}_\ell} B_{\ell,C} = B_\ell$ preserves the layer budget, including the retained window (see pseudocode in Appendix C).

Overall, the coarse-to-fine pipeline of TridentKV first protects distinctive layers, then allocates capacity across distinct head behaviors, and finally conducts robust spatial-temporal score aggregation, enabling more effective and reliable KV cache eviction under constrained budgets. Besides, Appendix B provides theoretical analysis and proofs for these three components.

4 Experiments

4.1 Experimental Setup

Models. We evaluate TridentKV on 3 open-source LLMs: Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-32B-Instruct (Bai et al., 2023), both supporting context lengths of up to 128K, and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), supporting up to 32K.



(a) Llama-3.1-8B-Instruct (b) Mistral-7B-Instruct-v0.3 (c) Qwen2.5-32B-Instruct

Figure 5: Overview of averaged generation quality across 16 datasets on LongBench.

Baselines. We compare TridentKV against 7 representative KV eviction methods: SnapKV (Li et al., 2024), AdaKV (Feng et al., 2026), PyramidKV (Cai et al., 2024), HeadKV (Fu et al., 2025), CAKE (Qin et al., 2025), ReST-KV (An et al., 2026), and DefensiveKV (Guo et al., 2026). SnapKV improves token importance estimation, PyramidKV, CAKE, AdaKV and HeadKV adopt adaptive budget allocation across layers or attention heads, while ReST-KV and DefensiveKV optimize the aggregation of importance scores. These baselines cover the main design paradigms of existing KV eviction methods. As TridentKV improves both budget allocation and score aggregation, these baselines provide a well-aligned and comprehensive comparison for evaluating its effectiveness.

Benchmarks. We assess TridentKV on 3 widely used long-context benchmarks: (1) LongBench (Bai et al., 2024), which evaluates long-context comprehension over 16 datasets covering 6 task categories; (2) RULER (Hsieh et al., 2024), a challenging benchmark comprising 13 tasks across 4 categories for evaluating long-context capabilities; and (3) Needle-in-a-Haystack (Liu et al., 2024b), which measures a model's ability to retrieve relevant information within lengthy input sequences.

Implementation Details. We evaluate TridentKV and all baselines under varying total cache budgets $B_{\text{total}} = bLH$, where b denotes the average number of KV pairs retained per KV head, L is the number of layers and H is the number of KV heads per layer. Baseline hyperparameters follow their official default settings, and all methods use a $m = 32$ observation query window. Following prior works (Li et al., 2024; Zhang et al., 2025; Guo et al., 2026), experiments are accelerated with FlashAttention (Dao, 2024), with cache eviction applied only once per layer during prefilling. Notably, AdaKV, PyramidKV, DefensiveKV, and TridentKV are implemented as plug-in modules on top of SnapKV. All experiments are conducted on NVIDIA A800 80GB GPUs.

4.2 LongBench Evaluation

We evaluate TridentKV's ability to preserve long-context understanding on the LongBench benchmark under cache budgets ranging from 128 to 2048. Figure 5 reports the average score normalized by the corresponding full-cache performance. Across all three models and all evaluated cache budgets, TridentKV consistently achieves the highest average score among the compared KV eviction methods. Table 1 further presents the per-dataset results at $B_{\text{total}} = 512LH$. At this cache budget, TridentKV improves the average score by approximately 2.0% over the strongest evaluated baseline on each model. At the same time, its average score drops by at most 0.39% relative to full-cache performance on all models. These results demonstrate that TridentKV retains nearly all of the full-cache performance while consistently outperforming strong KV eviction baselines under the same cache constraint. Detailed per-dataset results for all evaluated cache budgets are provided in Appendix J.

4.3 RULER Evaluation

We further evaluate TridentKV's ability to preserve long-context retrieval and reasoning on the RULER benchmark. We set the context length to 32K tokens and sample 100 instances per task. Table 2 presents the per-task results for all evaluated models. Across all three models, TridentKV achieves the highest average score among the compared KV eviction methods, with a relative improvement of approximately 2.7%–5.4% over strongest baselines on each model. The gains extend across multi-key, multi-query, and multi-value retrieval as well as question answering, demonstrating that TridentKV better preserves task-relevant information under the same cache constraint. Additional results under different cache budgets are provided in Appendix H.

Table 1: Detailed scores of 16 datasets on LongBench ($B_{\text{total}} = 512LH$).

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	Avg.
Llama-3.1-8B-Instruct, $B_{\text{total}} = 512LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	29.35	34.41	36.54	54.01	36.46	27.93	33.63	25.33	26.63	56.11	92.29	51.34	7.19	84.19	73.69	61.46	45.66
PyramidKV	28.69	26.37	**41.56**	49.78	34.12	25.62	31.00	24.37	26.71	**63.69**	89.66	43.19	10.48	83.06	72.54	**66.12**	44.81
CAKE	29.37	37.94	40.67	53.99	38.77	31.49	34.28	25.39	26.69	59.24	92.27	50.79	7.21	**95.43**	72.49	52.30	46.77
HeadKV	29.39	34.40	38.78	56.16	36.82	26.37	33.95	25.35	26.65	62.06	92.23	50.81	7.23	84.19	72.52	63.41	46.27
AdaKV	29.41	34.44	38.80	56.87	36.84	26.39	31.33	25.37	26.67	62.08	92.25	50.83	8.71	84.21	72.56	61.48	46.14
ReST-KV	29.45	**38.67**	40.67	56.85	40.65	**32.29**	34.30	25.43	26.73	62.04	96.03	43.21	**13.17**	94.89	72.50	61.44	48.02
DefensiveKV	29.43	34.46	40.69	56.18	38.79	28.22	34.32	25.41	26.75	62.08	96.05	51.36	9.28	89.61	73.94	63.43	47.50
TridentKV	**29.47**	37.19	41.54	**57.42**	**40.77**	30.81	**34.34**	**25.45**	**26.77**	63.67	**97.50**	**53.98**	10.46	91.71	**76.66**	66.10	**48.99**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 512LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	23.67	30.95	36.59	46.39	34.40	24.40	33.73	25.56	26.25	62.79	89.46	46.88	6.69	88.77	65.85	60.34	43.92
PyramidKV	23.69	33.58	43.15	46.41	36.47	24.48	33.81	**25.70**	26.37	64.15	89.48	46.80	6.71	63.96	60.36	59.20	42.77
CAKE	26.86	31.35	43.09	50.57	36.41	24.38	33.77	25.60	26.33	64.13	96.15	46.84	6.17	84.14	65.87	60.26	45.12
HeadKV	26.88	31.39	36.63	50.61	34.42	24.42	32.14	25.58	26.29	63.52	92.20	46.76	6.21	89.94	**65.97**	60.32	44.58
AdaKV	26.92	31.37	36.61	**50.67**	31.97	24.44	31.29	25.62	26.27	62.96	91.80	46.82	6.23	89.96	65.91	60.28	44.32
ReST-KV	26.94	**33.63**	43.11	50.59	**36.49**	24.67	33.75	25.68	26.31	71.45	91.78	46.78	6.19	93.73	65.89	59.17	46.01
DefensiveKV	26.90	33.54	43.13	50.63	36.43	24.46	33.79	25.64	26.35	71.43	96.13	46.86	6.69	93.71	65.93	60.30	46.37
TridentKV	**26.96**	33.56	**49.15**	50.65	36.45	**27.86**	**34.38**	25.66	**26.39**	**73.57**	**96.17**	**46.90**	**6.73**	**96.06**	65.95	**60.36**	**47.30**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 512LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	28.37	28.51	33.09	57.58	49.01	36.12	32.45	24.32	24.75	62.90	88.23	55.80	10.38	79.87	60.37	43.29	44.69
PyramidKV	27.04	28.03	38.87	55.65	49.05	33.47	30.69	20.88	24.85	69.73	85.93	46.99	12.69	72.30	57.53	41.34	43.44
CAKE	27.02	32.98	38.83	55.63	49.03	33.45	32.41	24.26	24.83	65.27	88.25	54.84	**12.73**	85.98	57.51	41.30	45.27
HeadKV	30.76	27.99	38.81	61.91	48.99	36.10	32.47	24.28	24.81	68.76	88.31	55.74	12.63	79.83	63.58	43.27	46.14
AdaKV	29.36	28.01	33.45	61.93	46.13	36.14	32.43	24.30	24.77	68.78	88.29	55.78	10.08	79.85	**63.60**	39.82	45.17
ReST-KV	30.78	31.87	38.85	63.59	54.25	38.78	32.49	24.34	24.79	69.71	88.27	**56.30**	12.71	**88.50**	60.35	34.34	46.87
DefensiveKV	30.80	34.42	44.14	63.61	55.92	38.80	32.51	24.36	24.87	71.46	92.44	55.76	12.65	77.08	60.34	41.32	47.53
TridentKV	**30.82**	**36.84**	**46.81**	**63.63**	**58.56**	**38.82**	**32.53**	**24.38**	**24.89**	**71.48**	**94.24**	55.72	12.67	78.94	60.31	**45.20**	**48.49**

Table 2: Detailed scores of 13 datasets on RULER ($B_{\text{total}} = 512LH$).

Method	cwe	fwe	niah_mk1	niah_mk2	niah_mk3	niah_mq	niah_mv	niah_s1	niah_s2	niah_s3	qa_1	qa_2	vt	Avg.
Llama-3.1-8B-Instruct, 32K RULER, $B_{\text{total}} = 512LH$														
FullCache	45.00	95.33	100.00	100.00	100.00	98.25	99.75	100.00	100.00	100.00	84.00	62.00	99.50	91.06
SnapKV	15.80	91.00	95.00	34.00	28.00	95.00	91.25	**100.00**	**100.00**	97.00	34.00	42.00	97.00	70.77
CAKE	26.70	92.33	98.00	57.00	52.00	96.75	94.75	**100.00**	**100.00**	**100.00**	49.00	49.00	**98.20**	77.98
AdaKV	27.10	92.00	98.00	55.00	50.00	97.00	94.75	**100.00**	**100.00**	99.00	46.00	49.00	98.10	77.38
ReST-KV	31.40	92.33	**99.00**	60.00	53.00	97.25	95.75	**100.00**	**100.00**	**100.00**	53.00	51.00	98.10	79.29
DefensiveKV	33.30	**92.67**	**99.00**	65.00	56.00	97.25	95.75	**100.00**	**100.00**	**100.00**	53.00	51.00	98.10	80.08
TridentKV	**37.80**	92.33	98.00	**71.00**	**63.00**	**98.75**	**97.50**	**100.00**	**100.00**	**100.00**	**58.00**	**55.00**	98.10	**82.27**
Mistral-7B-Instruct-v0.3, 32K RULER, $B_{\text{total}} = 512LH$														
FullCache	30.00	95.67	94.00	69.00	31.00	94.25	95.00	99.00	100.00	100.00	60.00	59.00	90.60	78.27
SnapKV	31.10	92.33	19.00	13.00	5.00	14.25	10.50	61.00	26.00	2.00	37.00	52.00	75.00	33.71
CAKE	32.50	92.67	27.00	20.00	**11.00**	22.25	21.00	67.00	39.00	7.00	43.00	53.00	79.40	39.60
AdaKV	28.70	92.67	22.00	13.00	7.00	18.75	14.50	53.00	33.00	3.00	41.00	53.00	73.80	34.88
ReST-KV	40.40	**93.67**	33.00	14.00	8.00	29.50	28.75	71.00	48.00	9.00	45.00	53.00	**80.00**	42.56
DefensiveKV	33.20	93.33	**42.00**	16.00	7.00	38.25	39.25	70.00	53.00	**18.00**	45.00	53.00	79.20	45.17
TridentKV	**40.50**	**93.67**	40.00	**22.00**	**11.00**	**41.00**	**41.00**	**73.00**	**59.00**	16.00	**48.00**	**54.00**	79.60	**47.60**
Qwen2.5-32B-Instruct, 32K RULER, $B_{\text{total}} = 512LH$														
FullCache	90.50	96.00	99.00	90.00	90.00	100.00	99.25	100.00	100.00	100.00	82.00	74.00	100.00	93.90
SnapKV	88.10	92.67	55.00	20.00	8.00	61.00	53.75	**100.00**	98.00	99.00	40.00	49.00	99.90	66.49
CAKE	89.30	**94.67**	86.00	48.00	37.00	89.25	87.25	**100.00**	**100.00**	**100.00**	60.00	62.00	**100.00**	81.04
AdaKV	88.60	93.00	66.00	24.00	13.00	72.00	66.75	99.00	**100.00**	**100.00**	48.00	54.00	99.90	71.10
ReST-KV	89.00	94.33	89.00	50.00	36.00	91.75	90.25	**100.00**	**100.00**	**100.00**	63.00	64.00	**100.00**	82.10
DefensiveKV	**89.60**	94.33	90.00	53.00	40.00	92.50	91.00	**100.00**	**100.00**	**100.00**	63.00	64.00	99.90	82.87
TridentKV	89.50	94.33	**93.00**	**60.00**	**47.00**	**96.50**	**95.25**	**100.00**	**100.00**	**100.00**	**66.00**	**66.00**	99.90	**85.19**

4.4 Needle-in-a-Haystack Evaluation

To visualize TridentKV's ability to retain information across long contexts, we select a sample from the niah_s1 task in RULER using Qwen2.5-32B-Instruct model. As shown in Figure 6, TridentKV achieves the best overall retrieval performance and remains robust across different context lengths and insertion depths. More visualizations on other models are provided in Appendix I.

4.5 Ablation Studies

Table 3 evaluates the contributions of TridentKV's components and the design choices for budget allocation and score aggregation on Llama-3.1-8B-Instruct using the LongBench benchmark. Each component improves the baseline individually, and the full framework achieves the best overall performance. The allocation ablations favor nearest-neighbor layer uniqueness and behavior-guided head budget competition with adaptive clustering thresholds over the tested alternatives. For score aggregation, jointly capturing spatial and temporal information within each cluster consistently out-

Table 3: Ablation studies with $B_{\text{total}} = 512LH$. (L/H/T: TridentKV at layer/head/token level).

(a) Components (Base–SnapKV)

Variant	Score	Variant	Score
Base	45.66	+L+H	48.06
+L	47.62	+L+T	48.68
+H	47.42	+H+T	48.72
+T	48.36	TridentKV	**48.98**

(b) Head budget competition

Strategy	Score
Uniform (SnapKV)	45.66
Global (AdaKV)	46.14
Random clustering	46.13
TridentKV (Head)	**47.42**

(c) Spatial-temporal aggregation

Aggregation	Score
Temporal-only	47.14
Spatial-only (cross-cluster)	45.22
Spatial-only (within-cluster)	47.73
Spatial-temporal	**48.36**

(d) Layer uniqueness formulation

Metric	Score
Uniform (SnapKV)	45.66
Average distance	46.53
Nearest neighbor (ours)	**47.62**

(e) Head clustering threshold

Rule (TridentKV (Head))	Score
Fixed cluster number (3)	46.24
Fixed threshold τ (0.025)	46.59
Adaptive threshold τ_ℓ	**47.42**

(f) Pooling kernel size

Kernel Size (k)	Score
TridentKV (Token)–3x3	**48.36**
TridentKV (Token)–5x5	48.07
TridentKV (Token)–7x7	47.83

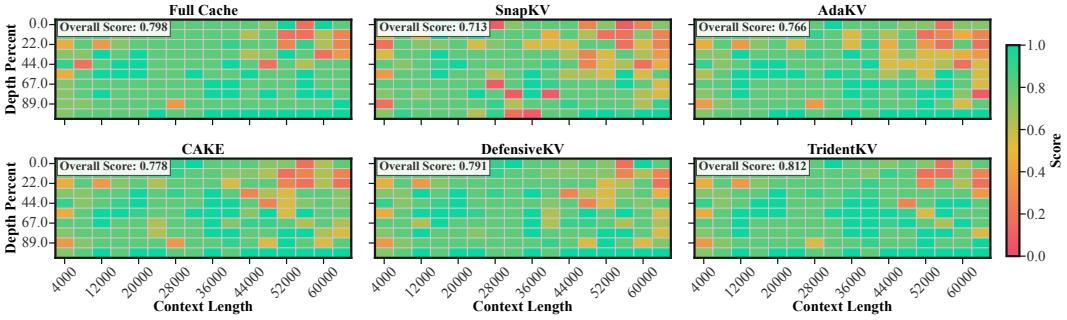


Figure 6: Needle-in-a-Haystack test results on Qwen2.5-32B-Instruct with $B_{\text{total}} = 512LH$.

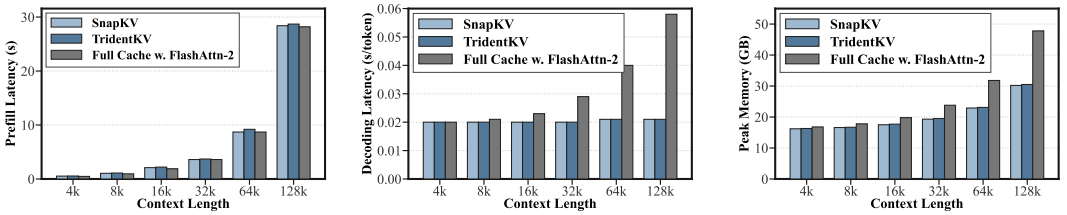


Figure 7: Efficiency analysis on Llama-3.1-8B-Instruct with $B_{\text{total}} = 512LH$.

performs modeling either dimension alone or mixing unrelated information across clusters, while a smaller pooling kernel is already sufficient to achieve strong performance.

4.6 Efficiency Analysis

We evaluate the efficiency of TridentKV on Llama-3.1-8B-Instruct across context lengths from 4K to 128K. We set $B_{\text{total}} = 512LH$ because this budget already preserves nearly all of the full-cache performance on LongBench (Table 1), allowing us to assess efficiency at a cache budget that maintains generation quality. Figure 7 compares prefill latency, per-token decoding latency, and peak memory against SnapKV and full-cache inference with FlashAttention-2. TridentKV achieves up to a 2.76× decoding speedup and a 36.2% reduction in peak memory relative to full-cache inference, both at a context length of 128K. Compared with SnapKV, TridentKV maintains comparable decoding latency and peak memory with only a modest increase in prefill latency. Together with the LongBench results, these findings show that TridentKV preserves generation quality under a compact cache budget while retaining the efficiency benefits of KV cache eviction.

5 Conclusion

We introduce TridentKV, a training-free KV cache eviction framework that improves cache utilization through diversity-aware budget allocation and spatial-temporal score aggregation. By preserving distinctive attention behaviors across layers and heads while retaining locally salient tokens, TridentKV consistently outperforms strong eviction baselines under constrained cache budgets. Experiments across multiple models and long-context benchmarks demonstrate that TridentKV preserves near-full-cache performance while substantially reducing decoding cost and memory usage, providing an effective solution for efficient long-context LLM inference.

AI USE STATEMENT

We used generative AI tools to assist with identifying potentially relevant literature for the related work, polishing the manuscript for clarity and coherence, and developing the theoretical proofs presented. The assistance with proofs included exploring proof strategies and formulating intermediate derivations. The authors take full responsibility for the selection and interpretation of cited literature, the correctness of the theoretical arguments, and the final content of the manuscript, including material developed with AI assistance.

ETHICS STATEMENT

TridentKV reduces the memory and computational requirements of long-context LLM inference, which may broaden access to these models on resource-constrained hardware. Our evaluation uses existing models and the LongBench, RULER, and Needle-in-a-Haystack benchmarks. As a general-purpose inference optimization, TridentKV may also lower the cost of harmful applications and does not itself mitigate biases, harmful outputs, or privacy risks associated with the underlying models. KV cache eviction can discard context needed to interpret instructions or factual evidence, potentially affecting output reliability and safety. The benchmarks evaluated here measure task performance and efficiency; they do not establish safety behavior or fairness is preserved under compression. Deployment should therefore include application-specific evaluation of these properties at the intended cache budget, alongside appropriate safeguards for model use and sensitive input data.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 defines TridentKV's budget allocation and token score aggregation procedures. Algorithm 1 in Appendix C provides pseudocode, including observation-window retention, integer budget allocation, head ordering, and pooling settings. Section 4.1 identifies the evaluated model variants, benchmarks, baselines, cache budget convention, implementation settings, and GPU hardware. Appendices L and M describe the LongBench datasets and RULER tasks, respectively. The assumptions, derivations, and proofs underlying our theoretical analysis are provided in Appendix B, together with the conditions under which the guarantees apply. We provide our code in the supplementary material to facilitate straightforward reproduction of our method.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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.

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. Rest-kv: Robust kv cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. *arXiv preprint arXiv:2605.08840*, 2026.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 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.

Priyansh Bhatnagar, Ashkan Moradifirouzabadi, Se-Hyun Yang, SeungJae Lee, Jungwook Choi, and Mingu Kang. Star-kv: Low-rank kv cache compression via soft thresholding for adaptive rank control. *arXiv preprint arXiv:2606.08382*, 2026.

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.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed Abdelfattah, and Kai-Chiang Wu. Palu: Kv-cache compression with low-rank projection. In *International Conference on Learning Representations*, volume 2025, pp. 50222–50249, 2025.

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

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

Daniel Defays. An efficient algorithm for a complete link method. *The computer journal*, 20(4): 364–366, 1977.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1074–1084, 2019.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. Criticalkv: Optimizing kv cache eviction from an output perturbation perspective. *arXiv preprint arXiv:2502.03805*, 2025.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2026.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 99269–99290, 2025.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pp. 70–79, 2019.

Team Glm, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pp. 12098–12107. PMLR, 2023.

Haoyu Guo, Junlin Lv, S Kevin Zhou, Xike Xie, et al. Defensivekv: Taming the fragility of kv cache eviction in llm inference. In *International Conference on Learning Representations*, volume 2026, pp. 43407–43432, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

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.

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

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 conference of the north American chapter of the association for computational linguistics: Human language technologies*, pp. 1419–1436, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Ehsan Kamalloo, Nouha Dziri, Charles LA Clarke, and Davood Rafiei. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, pp. 5591–5606, 2023.

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

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.

Xin Li and Dan Roth. Learning question classifiers. In *Coling 2002: The 19th international conference on computational linguistics*, 2002.

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.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024a.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *Advances in Neural Information Processing Systems*, 38:54358–54385, 2026.

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

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. In *International Conference on Learning Representations*, volume 2024, pp. 47832–47850, 2024c.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, et al. Cachegen: Kv cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024d.

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.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024e.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Heyi Tang, Feng Ren, Teng Ma, Shangming Cai, Yineng Zhang, Mingxing Zhang, et al. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *ACM Transactions on Storage*, 22(4):1–38, 2026.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. Cake: Cascading and adaptive kv cache eviction with layer preferences. In *International Conference on Learning Representations*, volume 2025, pp. 89945–89977, 2025.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*, 2023.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian Yen, Avner May, Tianqi Chen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 6835–6850, 2025.

Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. Eigen attention: Attention in low-rank space for kv cache compression. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15332–15344, 2024.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Cam-Tu Nguyen. LAVa: Layer-wise KV cache eviction with dynamic budget allocation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13672–13692, November 2025.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

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

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in neural information processing systems*, 37:119638–119661, 2024a.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024b.

Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. Longspec: Long-context lossless speculative decoding with efficient drafting and verification. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1826–1844, 2026.

Yifei Yang, Zouying Cao, Qiguang Chen, Libo Qin, Dongjie Yang, Hai Zhao, and Zhi Chen. Kvsharer: Efficient inference via layer-wise dissimilar kv cache sharing. *arXiv preprint arXiv:2410.18517*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25805–25813, 2025.

Minwei Zhang, Haifeng Sun, Jingyu Wang, Shaolong Li, Wanyi Ning, Qi Qi, Zirui Zhuang, and Jianxin Liao. Clusterattn: Kv cache compression under intrinsic attention clustering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14451–14473, 2025.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first international conference on machine learning*, 2024.

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.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, 2021.

A Additional Related Works

Beyond KV eviction, prior work suggests that the KV-cache bottleneck is inherently multi-dimensional, motivating compression along several complementary axes. KV merging methods, such as CaM (Zhang et al., 2024), KVMerger (Wang et al., 2024), MiniCache (Liu et al., 2024a) and KVSharer (Yang et al., 2024), exploit redundancy across tokens or layers by consolidating multiple cache entries into compact representations, thereby preserving information that would otherwise be discarded by eviction. KV quantization methods, including KIVI (Liu et al., 2024e), KVQuant (Hooper et al., 2024), and QJL (Zandieh et al., 2025), instead reduce numerical precision while largely preserving the temporal extent of the cache, making them naturally complementary to structural compression techniques. Another line of work applies low-rank compression to the representation dimension. Methods such as Eigen Attention (Saxena et al., 2024) and Palu (Chang et al., 2025) exploit redundancy in key-value features and can, in principle, be combined with both token-level compression and quantization.

Several systems techniques address orthogonal sources of KV-cache overhead. PagedAttention (Kwon et al., 2023) reduces memory fragmentation and duplication through paged KV-cache management, CacheGen (Liu et al., 2024d) compresses serialized KV states to lower transfer costs, and Mooncake (Qin et al., 2026) introduces a distributed KV-centric storage hierarchy to reduce expensive GPU memory residency. KV-cache compression also interacts closely with decoding-time computation. TriForce (Sun et al., 2024), MagicDec (Sadhukhan et al., 2025), and LongSpec (Yang et al., 2026) leverage reduced or selectively accessed caches to enable efficient speculative decoding, while query-aware sparse-attention methods such as QUEST (Tang et al., 2024) and SparQ (Ribar et al., 2023) reduce the amount of historical KV state accessed for each query.

Notably, sparse attention and KV cache eviction are conceptually related but address different aspects of inference efficiency (Xiao et al., 2024a; Tang et al., 2024; Jiang et al., 2024; Liu et al., 2026). KV cache eviction reduces memory consumption by retaining only a selected subset of cached key-value entries, whereas sparse attention preserves the entire KV cache and selectively attends to only a small fraction of entries during computation. As a result, sparse attention primarily reduces computational overhead without directly decreasing the memory footprint of the KV cache. These two approaches are therefore largely orthogonal and can potentially be combined. A promising direction for future work is to first apply KV cache eviction to substantially reduce cache size and subsequently employ sparse attention to further accelerate attention computation. Such a hybrid strategy could improve both memory and computational efficiency during long-context inference.

B Theoretical Analysis of TridentKV

TridentKV reduces eviction error by allocating capacity to distinctive layers, preserving complementary head behaviors, and retaining locally salient tokens. We bound output error by discarded attention mass and derive how each component improves the resulting bound. The analysis accounts for cluster-dependent changes in attention and spatial-temporal response variation, yielding explicit improvement terms for diversity-aware allocation and spatial-temporal aggregation.

B.1 From Retained Attention to Output Error

Let $\mathcal{W}$ be the mandatory observation window, $\mathcal{P} = [n] \setminus \mathcal{W}$ the prefix, and $\mathcal{I}_{\ell,h} \supseteq \mathcal{W}$ the retained indices of KV head h. Local comparisons use the same queries, keys, values, and output projection, with attention renormalized after eviction. Retained attention mass is assumed positive. The results concern attention outputs and, through a fixed-forward-pass argument, continuous model outputs.

Lemma B.1 (Output error is controlled by discarded attention mass)**.** *For one query head, let a_i be its full-cache attention weights, v_i its values, and $\mathcal{I}$ its retained set. Define*

$$\delta = \sum_{i \notin \mathcal{I}} a_i < 1, \qquad z = \sum_i a_i v_i, \qquad \widehat{z} = \frac{\sum_{i \in \mathcal{I}} a_i v_i}{1 - \delta}.$$

If $D = \max_{i,j} \|v_i - v_j\|_2$, then

$$z - \widehat{z} = \sum_{i \notin \mathcal{I}} a_i (v_i - \widehat{z}), \qquad \|z - \widehat{z}\|_2 \le D\delta. \tag{14}$$

Proof. Substituting $(1-\delta)\widehat{z} = \sum_{i\in\mathcal{I}} a_i v_i$ gives the identity. The retained output $\widehat{z}$ is a convex combination of retained values, so $\|v_i - \widehat{z}\|_2 \le D$. The triangle inequality yields the bound. Equality is attained when all retained values equal one vector and all evicted values equal another vector at distance D.

Define the mean attention and mean layer-output error as

$$\bar{a}_{\ell,h,i} = \frac{1}{Gm}\sum_{g=1}^{G}\sum_{t=1}^{m} \mathbf{A}^{\ell}_{h,g}[t,i], \qquad \varepsilon_\ell = \frac{1}{m}\sum_{t=1}^{m} \|\mathbf{o}^{\ell}_t - \widehat{\mathbf{o}}^{\ell}_t\|_2.$$

Let D^{val}_ℓ bound the value diameters in layer ℓ, and set $C_\ell = GD^{\mathrm{val}}_\ell \|\mathbf{W}^{\ell}_O\|_2$, using the spectral norm. Applying Lemma B.1 to each query head, bounding the norm of the concatenated perturbations by their sum, and averaging over queries gives

$$\varepsilon_\ell \le C_\ell \sum_{h=1}^{H}\sum_{i\notin\mathcal{I}_{\ell,h}} \bar{a}_{\ell,h,i} = C_\ell \left(H - \sum_{h=1}^{H}\sum_{i\in\mathcal{I}_{\ell,h}} \bar{a}_{\ell,h,i} \right). \tag{15}$$

Thus, increasing retained attention by Δ decreases this output-error upper bound by $C_\ell\Delta$. This is the retained-mass optimization principle underlying the comparisons below. We compare bounds under shared constants and evaluation conditions; realized errors additionally depend on value directions.

Propagation to model outputs. For a fixed forward pass with historical KV states held fixed, let Λ_ℓ bound the Lipschitz gain from the attention output of layer ℓ to final logits, including the remaining operations in that layer. Form hybrid computations by compressing the first j layers, for $j = 0, \dots, L$. Consecutive hybrids share the input to the layer being switched. If ε_ℓ denotes its local error at that hybrid input, the triangle inequality gives

$$\varepsilon_{\mathrm{model}} \le \sum_{\ell=1}^{L} \Lambda_\ell \varepsilon_\ell. \tag{16}$$

Here $\varepsilon_{\mathrm{model}}$ is the mean Euclidean logit difference over the evaluated queries. Bounds used in this expression must cover the corresponding hybrid inputs.

B.2 Layer Allocation Reduces the Largest Calibrated Error Bound

For the normalized, vectorized attention patterns $a_\ell = \mathrm{vec}(\mathbf{A}^\ell)/\|\mathbf{A}^\ell\|_F$, Eq. 3 gives

$$u_\ell = \frac{1}{2}\min_{r\neq\ell} \|a_\ell - a_r\|_2^2. \tag{17}$$

This identity follows from $\|a_\ell - a_r\|_2^2 = 2 - 2\langle a_\ell, a_r\rangle$ and identifies uniqueness with separation from the nearest layer behavior. The association between uniqueness and compression-induced output error in Figure 8 motivates the following explicit calibration model.

Let $b_\ell = B^{\mathrm{free}}_\ell$, $B = B_{\mathrm{free}} > 0$, and $U = \sum_\ell u_\ell$. Assume $u_\ell > 0$ and that, over the budgets being compared,

$$\Lambda_\ell \varepsilon_\ell(b_\ell) \le e_\ell(b_\ell) := \varepsilon_0 + \kappa \frac{u_\ell}{b_\ell}, \qquad \varepsilon_0 \ge 0, \quad \kappa > 0. \tag{18}$$

The shared constants model a common residual floor and inverse-budget decay with uniqueness-dependent sensitivity. This is a calibration assumption about output error; the geometric identity alone does not imply this decay law.

Theorem B.1 (Strict improvement over uniform layer allocation)**.** *For positive real-valued budgets with $\sum_\ell b_\ell = B$, suppose the proportional allocation is within layer capacities. TridentKV's rule $b^{\mathrm{T}}_\ell = Bu_\ell/U$ minimizes $\mathcal{E}_{\max}(b) := \max_\ell e_\ell(b_\ell)$, with*

$$\min_{\substack{b_\ell>0 \\ \sum_\ell b_\ell = B}} \max_\ell \frac{u_\ell}{b_\ell} = \frac{U}{B}, \qquad \mathcal{E}_{\max}(b^{\mathrm{T}}) = \varepsilon_0 + \kappa\frac{U}{B}. \tag{19}$$

For uniform budgets $b_\ell^{\mathrm{U}} = B/L$ and $u_{\max} = \max_\ell u_\ell$,

$$\begin{aligned} \mathcal{E}_{\max}(b^{\mathrm{U}}) - \mathcal{E}_{\max}(b^{\mathrm{T}}) &= \frac{\kappa}{B}(Lu_{\max} - U), \\ \frac{\mathcal{E}_{\max}(b^{\mathrm{T}}) - \varepsilon_0}{\mathcal{E}_{\max}(b^{\mathrm{U}}) - \varepsilon_0} &= \frac{U}{Lu_{\max}} \le 1. \end{aligned} \tag{20}$$

The improvement is strict whenever the uniqueness scores are not all equal.

Proof. For $t = \max_\ell u_\ell/b_\ell$, summing $u_\ell \le t b_\ell$ gives $U \le tB$. Hence $t \ge U/B$, attained by b_ℓ^{T} in every layer. Uniform allocation gives $t = Lu_{\max}/B$. Subtracting the two calibrated bounds proves the result; $Lu_{\max} > U$ precisely when some $u_\ell < u_{\max}$.

The theorem quantifies the benefit of prioritizing distinctive layers: proportional allocation reduces the largest calibrated error bound by supplying more capacity to high-uniqueness layers. It optimizes the largest layer contribution, which is distinct from minimizing their sum in Eq. 16. For feasible integer budgets $\widehat{b}_\ell \ge (1-\rho) b_\ell^{\mathrm{T}}$ with $0 \le \rho < 1$, direct substitution yields

$$\mathcal{E}_{\max}(\widehat{b}) \le \varepsilon_0 + \frac{\kappa U}{(1-\rho)B}. \tag{21}$$

Thus the improvement survives rounding whenever $U/(1-\rho) < Lu_{\max}$. For capacity-clipped allocations with positive budgets, the corresponding bound is $\varepsilon_0 + \kappa \max_\ell u_\ell/\widehat{b}_\ell$.

B.3 Head Allocation Combines Adaptation with Behavioral Protection

Complete-link clustering gives a concrete locality property. Every output cluster satisfies

$$\max_{h,h' \in C} d_{hh'}^\ell \le \tau_\ell, \qquad \left\| \frac{\bar{\mathbf{A}}_h^\ell}{\|\bar{\mathbf{A}}_h^\ell\|_F} - \frac{\bar{\mathbf{A}}_{h'}^\ell}{\|\bar{\mathbf{A}}_{h'}^\ell\|_F} \right\|_F \le \sqrt{2\tau_\ell}. \tag{22}$$

Singletons satisfy the first inequality, and each permitted merge preserves it for all existing and new pairs. Induction proves the claim; the normalized-distance identity gives the second inequality. Budget competition therefore takes place within groups of bounded behavioral dissimilarity.

Fix a layer, its partition, and feasible integer quotas $b_C = B_{\ell,C}^{\mathrm{free}}$. Write $\mathcal{Q}_C = C \times \mathcal{P}$ and let $s_{h,i} \ge 0$ be fixed selection scores. For a retained prefix set $\mathcal{R}_C \subseteq \mathcal{Q}_C$, define

$$T_C(\mathcal{R}_C) = \sum_{(h,i) \in \mathcal{Q}_C \setminus \mathcal{R}_C} s_{h,i}. \tag{23}$$

When $s_{h,i} \ge \bar{a}_{\ell,h,i}$, $C_\ell \sum_C T_C$ bounds layer-output error. Both raw mean attention and the pooled scores in Eq. 11 satisfy this requirement on the observation window.

Theorem B.2 (Within-cluster adaptation improves on uniform head budgets)**.** *Let $\mathcal{R}_C^{\mathrm{T}}$ contain the b_C largest scores in $\mathcal{Q}_C$. For any selection $\mathcal{R}_C^0$ with $|\mathcal{R}_C^0| = b_C$,*

$$\begin{aligned} \Delta_C &:= \sum_{\mathcal{R}_C^{\mathrm{T}} \setminus \mathcal{R}_C^0} s_{h,i} - \sum_{\mathcal{R}_C^0 \setminus \mathcal{R}_C^{\mathrm{T}}} s_{h,i} \ge 0, \\ C_\ell \sum_C T_C(\mathcal{R}_C^{\mathrm{T}}) &= C_\ell \sum_C T_C(\mathcal{R}_C^0) - C_\ell \sum_C \Delta_C. \end{aligned} \tag{24}$$

In particular, this holds for uniform head budgets whose retained counts sum to the prescribed cluster quotas. If $d_C = |\mathcal{R}_C^{\mathrm{T}} \setminus \mathcal{R}_C^0|$ exchanged pairs satisfy a score separation of at least $\mu_C > 0$ between the newly retained and removed pairs, then $\Delta_C \ge d_C \mu_C$.

Proof. The two difference sets have equal size. Every selected score in a TopK set is at least every unselected score, so pairing the newly retained and removed entries gives $\Delta_C \ge 0$, or $\Delta_C \ge d_C \mu_C$ under the margin condition. Subtracting the discarded-score sums gives the equality. Summing over clusters completes the proof.

TridentKV adapts head budgets within each behavioral group to maximize retained score under the group's reserved capacity. At the same cluster capacity, it retains at least as much score as a uniform split, with a strict output-bound improvement when $C_\ell > 0$ and an exchange has a positive margin. Uniform allocation is a feasible comparator whenever b_ℓ/H is integral and $b_C = |C|b_\ell/H$; for rounded allocations, the comparison uses their actual cluster totals.

Why reserve capacity across clusters? Separate quotas also protect against changes in the relative importance of distinct behaviors. To formalize this advantage, suppose evaluated attention satisfies the cluster-dependent envelope

$$\widetilde{\widetilde{a}}_{\ell,h,i} \le \gamma_C s_{h,i} + \eta_\ell, \qquad h \in C, \quad i \in \mathcal{P}, \quad \gamma_C \ge 0, \quad \eta_\ell \ge 0. \tag{25}$$

The factors γ_C represent group-dependent growth relative to the score estimate. With shared value-diameter bounds and b_ℓ retained prefix pairs, a common output certificate for all compared sets is

$$\widetilde{\varepsilon}_\ell(\mathcal{R}) \le \mathcal{E}_\gamma(\mathcal{R}) := C_\ell \left[\sum_C \gamma_C T_C(\mathcal{R}_C) + \eta_\ell \big(H(n-m) - b_\ell \big) \right]. \tag{26}$$

Theorem B.3 (Behavioral protection improves on global competition under a shift margin)**.** *Let $\mathcal{R}^{\mathrm{G}}$ select the globally largest b_ℓ scores, with $q_C = |\mathcal{R}^{\mathrm{G}} \cap \mathcal{Q}_C|$, and use the same deterministic tie order for all TopK operations. Let $\mathcal{R}^{\mathrm{T}}$ use TridentKV's quotas b_C. Sort each cluster's scores as $s_{C,(1)} \ge \cdots \ge s_{C,(|\mathcal{Q}_C|)}$. For clusters receiving additional capacity and those releasing capacity, respectively, define*

$$\mathcal{C}_+ = \{C : q_C < b_C\}, \quad A_C = \sum_{j=q_C+1}^{b_C} s_{C,(j)}; \qquad \mathcal{C}_- = \{C : q_C > b_C\}, \quad D_C = \sum_{j=b_C+1}^{q_C} s_{C,(j)}.$$

Under Eq. 25,

$$\mathcal{E}_\gamma(\mathcal{R}^{\mathrm{G}}) - \mathcal{E}_\gamma(\mathcal{R}^{\mathrm{T}}) = C_\ell \left[\sum_{C \in \mathcal{C}_+} \gamma_C A_C - \sum_{C \in \mathcal{C}_-} \gamma_C D_C \right]. \tag{27}$$

Consequently, for $C_\ell > 0$, TridentKV has a strictly smaller output-error certificate whenever the weighted recovered mass exceeds the weighted sacrificed mass. A sufficient condition, when both cluster sets are nonempty and the denominators are positive, is

$$\frac{\min_{C \in \mathcal{C}_+} \gamma_C}{\max_{C \in \mathcal{C}_-} \gamma_C} > \frac{\sum_{C \in \mathcal{C}_-} D_C}{\sum_{C \in \mathcal{C}_+} A_C}. \tag{28}$$

Proof. A global TopK set contains the first q_C entries within each cluster. Increasing its quota to b_C removes exactly A_C from the discarded-score sum; decreasing the quota adds exactly D_C. Substitute these changes into Eq. 26. The tolerance term cancels because both sets retain b_ℓ pairs. Bounding growth factors below on $\mathcal{C}_+$ and above on $\mathcal{C}_-$ proves the sufficient condition.

The shift margin identifies when behavioral reservations are useful: a group assigned little capacity by its observed scores becomes sufficiently important at evaluation. TridentKV preserves entries from that group before the shift occurs. The theorem compares against unrestricted global competition with matched scores. When all γ_C are equal, global TopK minimizes the unweighted certificate. When behavioral importance changes, TridentKV's reservations yield the improvement quantified in Eq. 27 under the stated shift margin.

Constructed example with strictly smaller actual output error. Consider four KV heads with $G = m = 1$, two prefix entries, and one retained window entry. The first two heads have attention $(0.45, 0.40, 0.15)$ and the other two have $(0.20, 0.19, 0.61)$. Identical pairs have distance zero and all cross-pair distances equal $d > 0$, so $\tau_\ell = 2d/3$ and complete linkage yields two clusters of size two. Pooling leaves these cluster-constant maps unchanged. With four free entries, global TopK retains both prefix entries in each head of the first cluster. TridentKV reserves two entries for each cluster and retains the first prefix entry in every head.

Now let attention change to $(0.05, 0.05, 0.90)$ in the first cluster and $(0.60, 0.10, 0.30)$ in the second. This satisfies Eq. 25 with $(\gamma_1, \gamma_2) = (1, 3)$ and $\eta_\ell = 0$. The weighted recovered mass is $3(2 \times 0.20) = 1.20$, exceeding the sacrificed mass $2 \times 0.40 = 0.80$. Both attention maps are realizable with fixed two-dimensional keys: take $k_{h,i} = \sqrt{2}(\log a^{\text{obs}}_{h,i}, \log a^{\text{eval}}_{h,i})$ and queries $(1, 0)$ and $(0, 1)$, respectively. Assign value $(1, 0)$ to both prefix entries and $(0, 0)$ to the window entry in every head, and let the output projection sum the first coordinates of the four head outputs. The full-cache output is 1.6. Global eviction produces 0.2, while TridentKV produces $2(0.05/0.95)+2(0.60/0.90)$. Therefore,

$$\varepsilon^{\text{G}}_\ell = 1.4, \qquad \varepsilon^{\text{T}}_\ell = \frac{46}{285} \approx 0.1614 < 1.4. \tag{29}$$

This analytical example exhibits an actual output-error reduction with the same values, projection, retained window, and total cache budget.

B.4 Spatial-Temporal Pooling Improves the Local-Robustness Bound

For a cell $p = (\phi_C(h, g), t)$, let $\mathcal{N}_k(p)$ contain the valid cells in its centered $k \times k$ neighborhood within cluster C. With the nonnegative maps in Eq. 9, unit stride, and zero padding,

$$\mathbf{P}^{\ell,C}_i(p) = \max_{q \in \mathcal{N}_k(p)} \mathbf{X}^{\ell,C}_i(q).$$

Since the center belongs to its neighborhood,

$$S_{\ell,h,i} \geq \bar{a}_{\ell,h,i}. \tag{30}$$

More generally, consider valid evaluated attention maps satisfying

$$\widetilde{\mathbf{X}}^{\ell,C}_i(p) \leq \gamma_C \mathbf{P}^{\ell,C}_i(p) + \eta_\ell. \tag{31}$$

Averaging over a KV head's Gm cells yields Eq. 25 with $s_{h,i} = S_{\ell,h,i}$. For $\gamma_C = 1$ and $\eta_\ell = 0$, the envelope covers the original observations and any valid local reassignment of rows $\widetilde{\mathbf{X}}_i(p) = \mathbf{X}_i(\psi(p))$, where $\psi(p) \in \mathcal{N}_k(p)$ is shared across tokens. Thus it captures a response moving to nearby query positions or query heads.

Theorem B.4 (Pooled selection improves on direct averaging)**.** *Fix cluster quotas and let $\mathcal{R}^{\text{avg}}$ select within each cluster using raw mean attention $\bar{a}_{\ell,h,i}$, while $\mathcal{R}^{\text{T}}$ uses pooled scores $S_{\ell,h,i}$. Under Eq. 31, both selections obey*

$$\widetilde{\varepsilon}_\ell(\mathcal{R}) \leq \mathcal{E}_{\text{pool}}(\mathcal{R}) := C_\ell \left[\sum_C \gamma_C \sum_{\mathcal{Q}_C \setminus \mathcal{R}_C} S_{\ell,h,i} + \eta_\ell \big(H(n-m) - b_\ell \big) \right]. \tag{32}$$

Define the pooled retained-score gain

$$\Delta^{\text{pool}}_C = \sum_{\mathcal{R}^{\text{T}}_C} S_{\ell,h,i} - \sum_{\mathcal{R}^{\text{avg}}_C} S_{\ell,h,i} \geq 0.$$

Then

$$\mathcal{E}_{\text{pool}}(\mathcal{R}^{\text{T}}) = \mathcal{E}_{\text{pool}}(\mathcal{R}^{\text{avg}}) - C_\ell \sum_C \gamma_C \Delta^{\text{pool}}_C \leq \mathcal{E}_{\text{pool}}(\mathcal{R}^{\text{avg}}). \tag{33}$$

The inequality is strict when $C_\ell > 0$ and some cluster with $\gamma_C > 0$ has a positive pooled retained-score gain.

Proof. Average Eq. 31, apply Eq. 15, and sum over evicted pairs to obtain the common certificate. Theorem B.2 applied to $s_{h,i} = S_{\ell,h,i}$ gives $\Delta^{\text{pool}}_C \geq 0$. The total pooled score and number of evicted pairs are fixed, so subtracting the two certificates gives Eq. 33.

This comparison evaluates both selections with the same uncertainty envelope. Pooling raises the priority of locally salient entries before selection; the improvement comes from retaining more of that locally robust score. The amplification remains spatially and temporally bounded:

$$\sum_{h \in C} \bar{a}_{\ell,h,i} \leq \sum_{h \in C} S_{\ell,h,i} \leq k^2 \sum_{h \in C} \bar{a}_{\ell,h,i}. \tag{34}$$

For the upper inequality, replace each neighborhood maximum by its sum; each input cell appears in at most k^2 neighborhoods. Consequently, a local response can support nearby observations without spreading across behavioral clusters.

B.5 Combined Output-Error Guarantee

Suppose Eq. 31 holds at the hybrid inputs in Eq. 16, with constants $\gamma_{\ell,C}$ and η_ℓ shared by the compared selections. Equations 16 and 32 give

$$\varepsilon_{\mathrm{model}}(\mathcal{R}) \le \mathcal{E}_{\mathrm{model}}(\mathcal{R}) := \sum_{\ell=1}^{L} \Lambda_\ell C_\ell \left[\sum_{C\in\mathcal{C}_\ell} \gamma_{\ell,C} \sum_{\mathcal{Q}_{\ell,C}\setminus\mathcal{R}_{\ell,C}} S_{\ell,h,i} + \eta_\ell\big(H(n-m)-b_\ell\big) \right]. \quad (35)$$

Corollary B.1 (Retained-score gains reduce the model-output certificate)**.** *Let a baseline $\mathcal{R}^0$ and TridentKV use the same layer budgets and total cache budget. Define*

$$\Delta_{\ell,C}^{0\to\mathrm{T}} = \sum_{\mathcal{R}^{\mathrm{T}}_{\ell,C}} S_{\ell,h,i} - \sum_{\mathcal{R}^{0}_{\ell,C}} S_{\ell,h,i}.$$

Under the common envelopes above,

$$\mathcal{E}_{\mathrm{model}}(\mathcal{R}^0) - \mathcal{E}_{\mathrm{model}}(\mathcal{R}^{\mathrm{T}}) = \sum_{\ell=1}^{L} \Lambda_\ell C_\ell \sum_{C\in\mathcal{C}_\ell} \gamma_{\ell,C}\Delta_{\ell,C}^{0\to\mathrm{T}}. \quad (36)$$

For matched cluster quotas, every gain is nonnegative when $\mathcal{R}^{\mathrm{T}}$ selects pooled TopK scores. This covers uniform head allocation with compatible quotas and direct-average selection. For global score competition, the shift-margin condition of Theorem B.3 gives a positive layer gain. A positive gain in a layer with $\Lambda_\ell C_\ell > 0$, with nonnegative gains elsewhere, strictly decreases the model-output certificate.

Proof. Subtract Eq. 35 for the two selections. Total pooled scores cancel within each cluster, and the tolerance terms cancel under matched layer budgets. The sign conclusions follow from Theorems B.2, B.3, and B.4.

Together, these results give each component a comparative role. Uniqueness-based allocation strictly lowers the largest calibrated layer-error bound relative to uniform allocation when layer uniqueness varies. Within-cluster adaptation and pooled selection reduce a common output-error certificate by explicit retained-score gains. Behavioral reservations additionally improve on global competition when the importance gained by protected groups exceeds the score mass sacrificed elsewhere. The fixed-forward-pass guarantee applies to continuous outputs under the stated attention envelopes; an autoregressive rollout also requires those envelopes to remain valid as queries and KV states evolve.

C Complete Algorithm

Algorithm 1 summarizes TridentKV's prefill cache compression. The last m tokens are retained within B_{total}, and the remaining capacity is allocated across layers and head clusters to select earlier tokens. The superscript free denotes capacity after window reservation. The routine ALLOCATE() produces capacity-constrained integer quotas through proportional allocation and largest-remainder rounding.

D More Visualizations of Layer Uniqueness

We examine six sample-wise profiles over 32 layers to further characterize the cross-layer heterogeneity that motivates uniqueness-based budget allocation (Section 3.2). Recall that the uniqueness score u_ℓ in Eq. 3 measures the cosine dissimilarity between a layer's attention behavior and its closest counterpart among the remaining layers. A larger score therefore indicates that the layer is less similar to even its nearest behavioral neighbor. Figure 8 pairs these uniqueness profiles with relative model output errors. In the lower-panel axis labels, y denotes the model output with the full KV cache, and $\hat{y}^{(\ell)}$ denotes the model output when only layer ℓ is compressed. The notation $\|\cdot\|_2$ denotes the Euclidean norm, and the error is normalized by $\|y\|_2$.

The six displayed profiles exhibit a recurring, non-monotonic pattern across depth. The first layer has the highest uniqueness score, while the middle layers form a broad region of elevated scores

Algorithm 1 TridentKV with mandatory observation-window retention

Require: Prompt KV caches $\{\mathbf{K}_h^\ell, \mathbf{V}_h^\ell\}$, final-window queries $\{\mathbf{Q}_{h,g}^\ell\}$;
prompt length n, integer budget B_{total}, window size $m = 32$, kernel $k = 3$
Ensure: Compressed caches and original token indices $\{\widehat{\mathbf{K}}_h^\ell, \widehat{\mathbf{V}}_h^\ell, \mathcal{I}_{\ell,h}\}$

$m \leftarrow \min(m, n)$; $\mathcal{W} \leftarrow \{n - m + 1, \ldots, n\}$
$B_{\text{total}} \leftarrow \min(B_{\text{total}}, LHn)$; require $B_{\text{total}} \geq LHm$
if $B_{\text{total}} = LHn$ **then**
  return full caches with $\mathcal{I}_{\ell,h} = [n]$ for every (ℓ, h)
end if
Compute $\mathbf{A}^\ell$ for all layers from the final m queries using Eq. 2
Compute u_ℓ from Eq. 3 for all layers
$(B_\ell^{\text{free}})_{\ell=1}^L \leftarrow \textsc{Allocate}(B_{\text{total}} - LHm,\ (u_\ell)_\ell,\ (H(n-m))_\ell)$
for each layer ℓ **do**
  $B_\ell \leftarrow Hm + B_\ell^{\text{free}}$
  Compute $\bar{\mathbf{A}}_h^\ell$, $d_{hh'}^\ell$, and τ_ℓ using Eqs. 5–7
  $\mathcal{C}_\ell \leftarrow \{\{1\}, \ldots, \{H\}\}$
  while $|\mathcal{C}_\ell| > 1$ **do**
    $(C_a, C_b) \leftarrow \arg\min_{C_a \neq C_b} D_\ell(C_a, C_b)$
    if $D_\ell(C_a, C_b) > \tau_\ell$ **then**
      break
    end if
    Replace C_a, C_b in $\mathcal{C}_\ell$ by $C_a \cup C_b$
  end while
  $(B_{\ell,C}^{\text{free}})_{C \in \mathcal{C}_\ell} \leftarrow \textsc{Allocate}(B_\ell^{\text{free}},\ (|C|)_C,\ (|C|(n-m))_C)$
  for each cluster $C \in \mathcal{C}_\ell$ **do**
    $B_{\ell,C} \leftarrow |C|m + B_{\ell,C}^{\text{free}}$
    Order (h, g) by original head indices to define ϕ_C
    Build $\mathbf{X}_i^{\ell,C}$ for $i \notin \mathcal{W}$ using Eq. 9
    Compute $\mathbf{P}_i^{\ell,C} \leftarrow \text{MaxPool}_{2\text{D}}(\mathbf{X}_i^{\ell,C}; k \times k)$
      with unit stride and zero padding of width $(k-1)/2$
    Average pooled responses to obtain $S_{\ell,h,i}$ using Eq. 11
    $\mathcal{R}_{\ell,C} \leftarrow \text{TopK}_{B_{\ell,C}^{\text{free}}}\{S_{\ell,h,i} : (h, i) \in C \times ([n] \setminus \mathcal{W})\}$
    for each head $h \in C$ **do**
      $\mathcal{I}_{\ell,h} \leftarrow \text{sort}\big(\mathcal{W} \cup \{i : (h, i) \in \mathcal{R}_{\ell,C(h)}\}\big)$
      Gather $\widehat{\mathbf{K}}_h^\ell, \widehat{\mathbf{V}}_h^\ell$ using Eq. 13
    end for
  end for
end for
return $\{\widehat{\mathbf{K}}_h^\ell, \widehat{\mathbf{V}}_h^\ell, \mathcal{I}_{\ell,h}\}$

relative to most preceding and subsequent layers. Scores rise again at the final layers. Although the magnitudes vary across samples, these recurring features illustrate how behavioral distinctiveness can vary across layers, motivating a budget schedule that adapts to layer uniqueness.

Across the six samples, layers with higher uniqueness generally exhibit larger relative model output errors after compression. This consistent association supports using uniqueness to guide layer-wise budget allocation, with more capacity assigned to distinctive layers to better preserve model outputs.

E More Visualizations of Head Clustering and Head Budget Allocation

Figure 9 extends Figure 1c to six samples, relating head attention trajectories to the choice of head budget allocation. The upper panels organize heads by behavioral similarity, while the lower panels compare the relative layer output errors of Uniform, Global, and Within-cluster allocation under matched layer budgets. In the lower-panel axis labels, o^ℓ denotes the output of layer ℓ with the full KV cache, and $\hat{o}^\ell$ denotes its output under the corresponding head budget allocation strategy. The error uses the Euclidean norm $\|\cdot\|_2$ and is normalized by $\|o^\ell\|_2$. Lower error indicates closer agreement with the full-cache output.

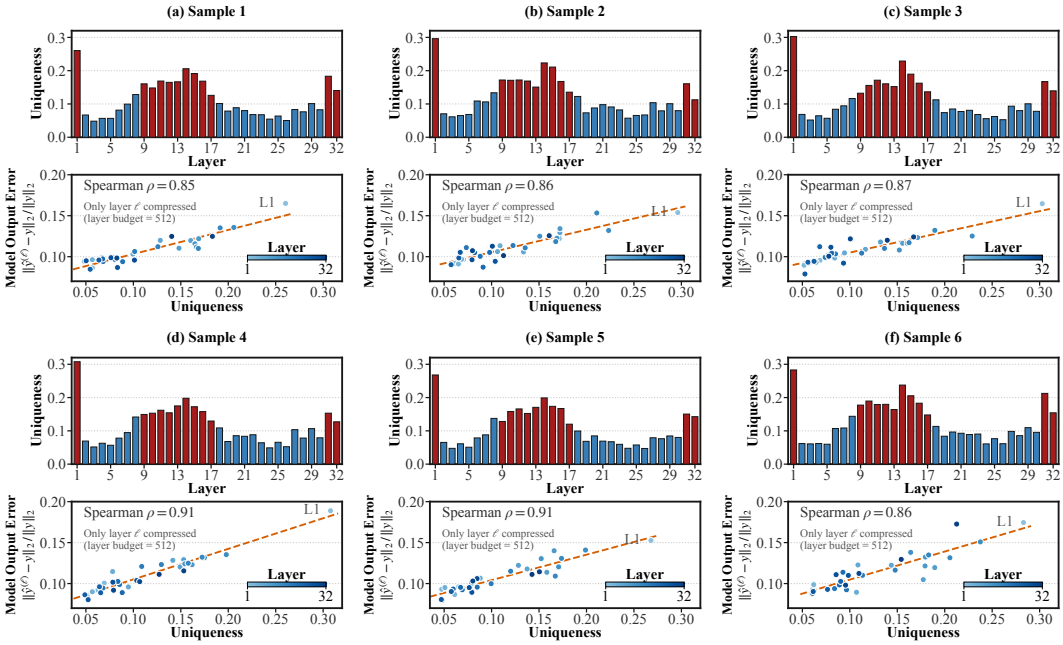


Figure 8: Layer uniqueness profiles (upper panels) and relative model output errors (lower panels) across six samples. Each point corresponds to compressing only one layer with a budget of 512, allowing us to examine the differences across layers.

These allocation strategies differ in which heads can compete for capacity. Uniform allocation assigns equal capacity to each head, while Global allocation allows all heads to compete for a shared layer budget. Within-cluster allocation restricts this competition to heads within the same behavioral group. Heads with similar attention trajectories can therefore share capacity, while distinct groups retain separate reservations, as described in Section 3.3.

To examine the behavioral structure underlying these groups, Figure 10 shows pairwise cosine similarities among eight KV heads in six selected layers, with heads reordered by cluster membership. Several multi-head clusters exhibit high internal similarity, while singleton clusters retain heads outside these groups. Cluster membership and size vary across layers: layer 13 contains groups of two, five, and one heads, whereas layer 25 contains groups of one, three, and four heads. This variation motivates deriving the partition from each layer's attention behavior.

With the behavioral groups defined, Figure 11 examines how capacity is distributed among them. We aggregate AdaKV's head budgets using the same cluster partition shown for TridentKV, allowing a direct comparison of group-level budget shares. In sample 1 at layer L1, two three-head clusters receive 50.7% and 6.8% of AdaKV's layer budget, respectively; TridentKV reserves 37.5% for each. In sample 2 at layer L0, two four-head clusters receive 19.3% and 80.7% under AdaKV, compared with 50.0% each under TridentKV. Thus, unrestricted competition based on head-level scores can assign substantially different capacities to behavioral groups of the same size.

The equal shares assigned to equal-sized groups by TridentKV follow from its cluster-level reservation rule. As specified in Eq. 8, each cluster receives a fraction $|C|/H$ of its layer's budget, while its constituent heads compete for that shared capacity according to token importance. This rule preserves capacity in proportion to the number of heads in each behavioral group and allows the allocation to adapt within the group, connecting the behavioral partition to the budget distribution.

F More Visualizations of Spatial-Temporal Token Score Aggregation

Figure 12 illustrates how cluster-local spatial-temporal pooling preserves local salience before score averaging (Section 3.4). The examples cover layers 2, 5, and 8, with a 32-token query window and four query heads per KV head. Within each cluster, the upper heatmap displays the input importance map and the lower heatmap displays its output after 3×3 max pooling with unit stride and padding

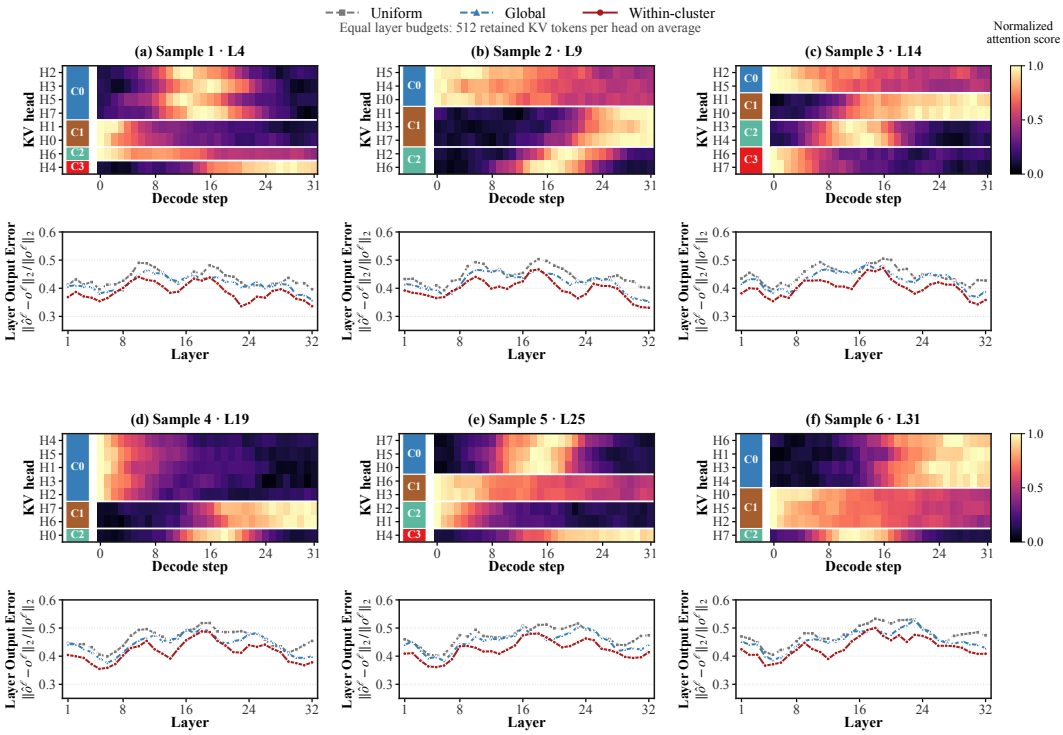


Figure 9: Upper panels show per-head normalized attention scores over 32 decode steps, with original head IDs reordered by cluster membership. Lower panels compare relative layer output errors under Uniform, Global, and Within-cluster head budget allocation across 32 layers at matched total layer budgets (512 retained KV tokens per head on average).

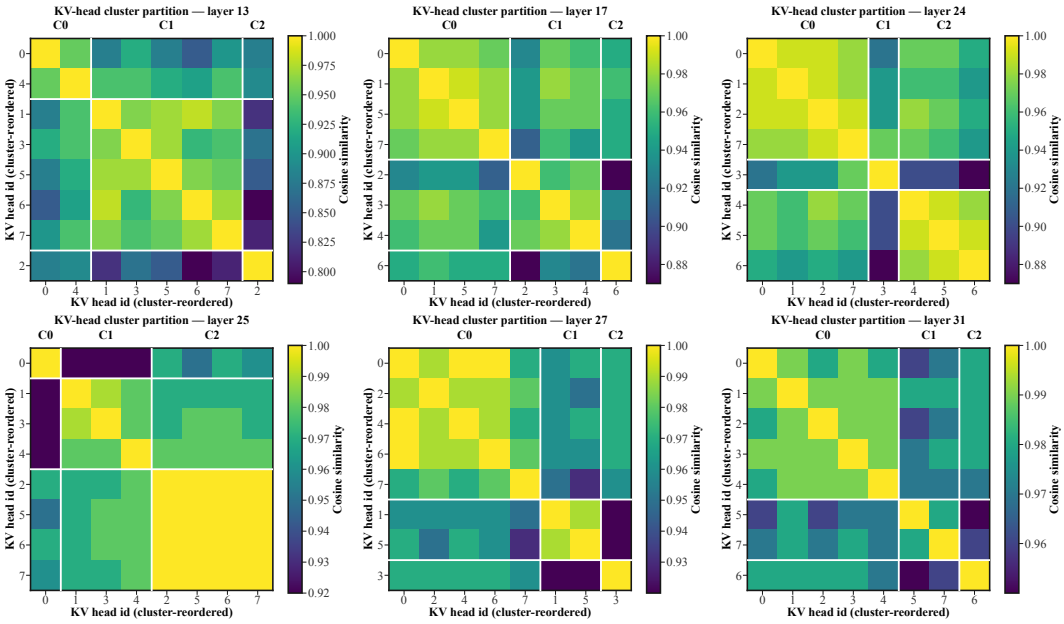


Figure 10: Behavioral similarity and cluster partitions of KV heads in six selected layers. Each heatmap shows pairwise cosine similarity, with head IDs reordered by cluster membership.

that preserves the map dimensions. Pooling operates jointly across query positions and query-head rows, independently for each cluster.

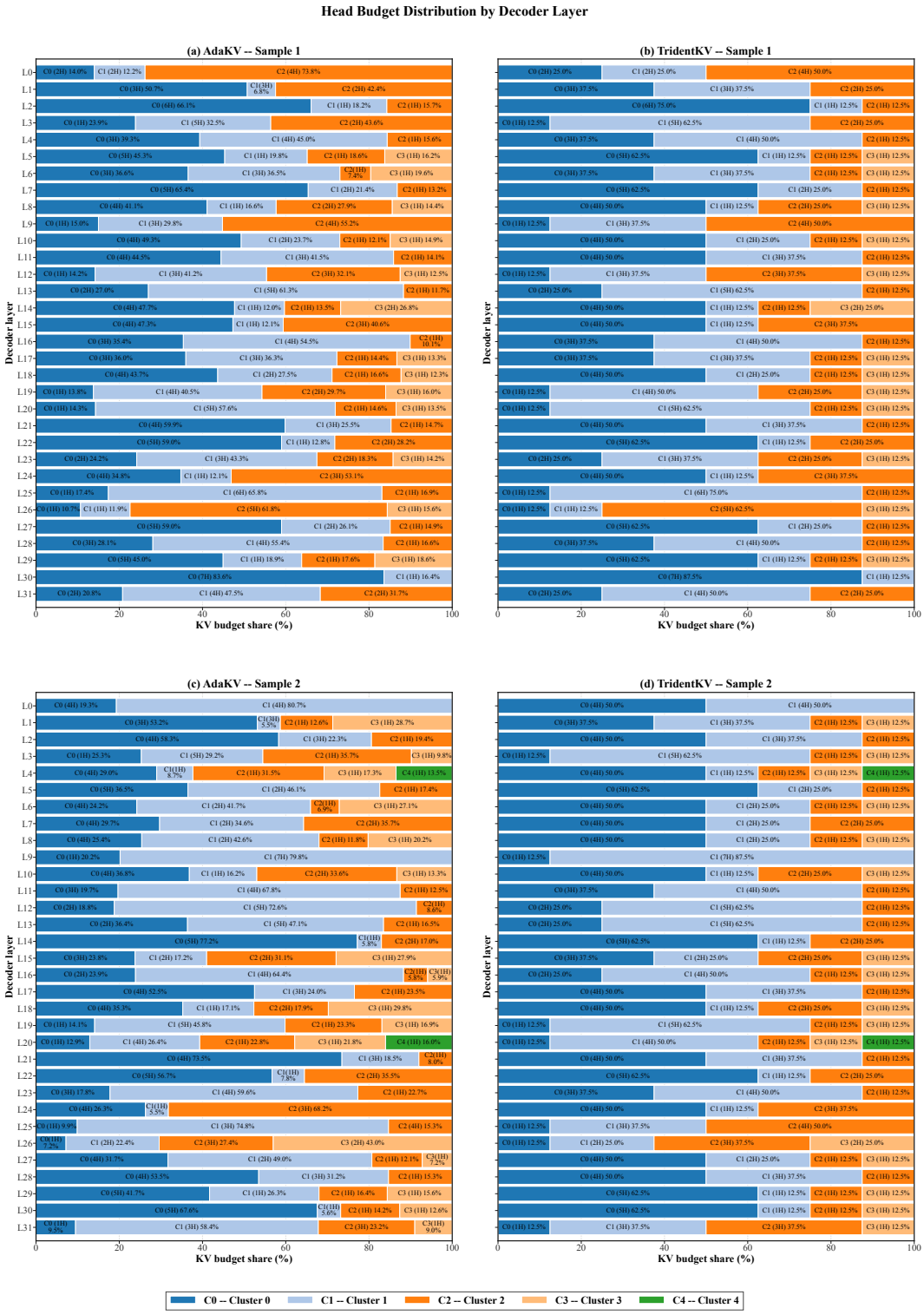


Figure 11: Within-cluster KV budget shares under AdaKV and TridentKV for two samples. Panels (a,b) and (c,d) compare the methods using a shared partition within each sample and layer. Each bar represents one layer's budget; segment labels give the cluster ID, head count, and budget share.

The paired maps show that local maxima extend their influence to neighboring positions before the responses are averaged. For example, in Cluster 0 of layer 8, localized high responses in the input become broader regions of elevated scores after pooling. This transformation allows a response confined to a few query positions or query heads to contribute more strongly to the aggregated score

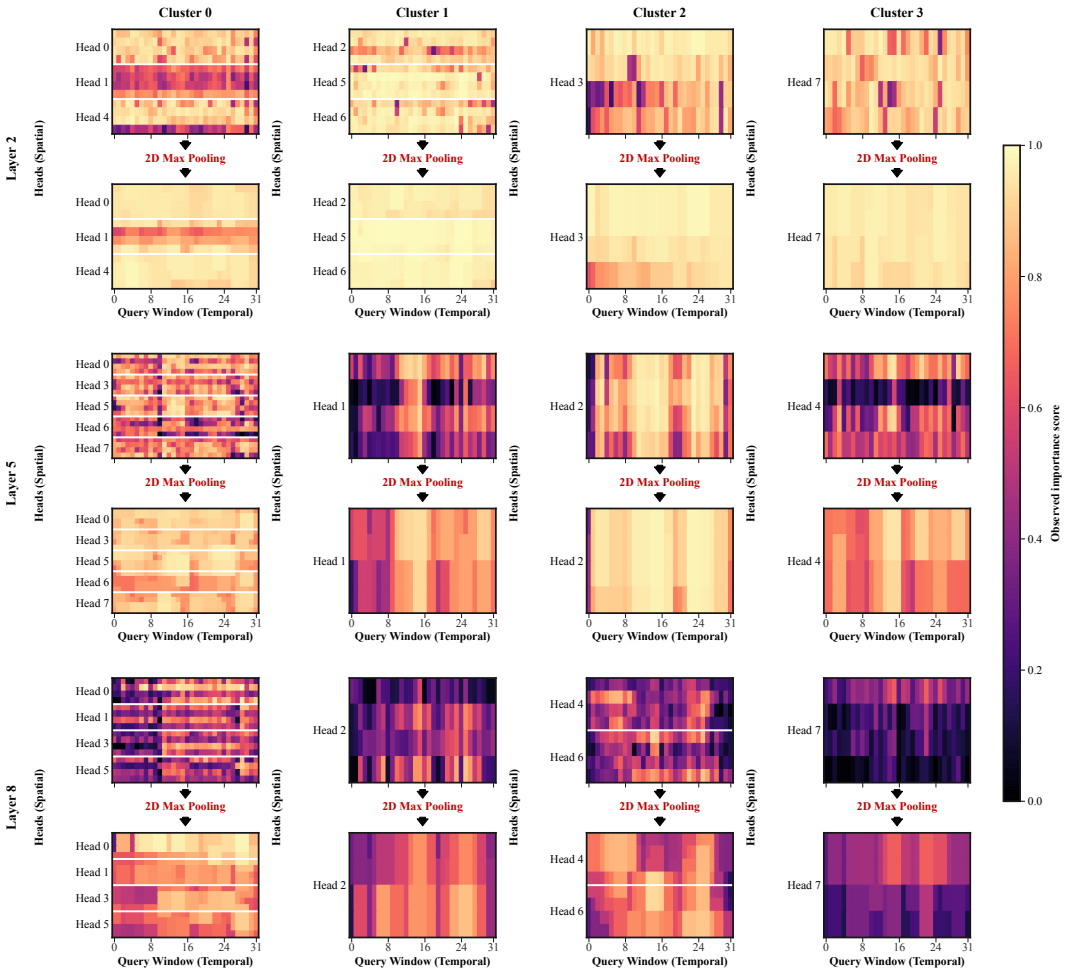


Figure 12: Spatial-temporal feature maps of head clusters before and after 2D max pooling. Each pair shows the input map (top) and the 3×3 max-pooled map (bottom) for one cluster. Columns index query positions; each KV-head block contains four query-head rows, with white lines separating KV heads and pooling is confined to individual clusters.

Table 4: Within-cluster head-ordering ablation of LongBench on Llama-3.1-8B-Instruct.

Head ordering	$B_{\text{total}} = 512LH$	$B_{\text{total}} = 1024LH$
Original (default)	48.36	49.01
Reversed	48.30	48.99
Random	48.39	48.95
Similarity-guided	48.32	49.03

in Eq. 11. Spatial pooling also remains active in singleton KV-head clusters, such as Cluster 1 of layer 5, because each KV head serves four query heads. In all cases, pooling neighborhoods remain within the cluster, retaining the separation between behavioral groups established by head clustering.

G Effect of Within-Cluster Head Ordering

Table 4 shows that different within-cluster head orderings yield comparable performance across both cache budgets, indicating that 2D pooling is insensitive to head orderings within the same cluster. Keeping pooling within each cluster is sufficient to maintain its effectiveness in these settings, supporting the use of the original head order without additional sorting overhead. Figure 13 illustrates the input and pooled responses under different head orderings in an example cluster.

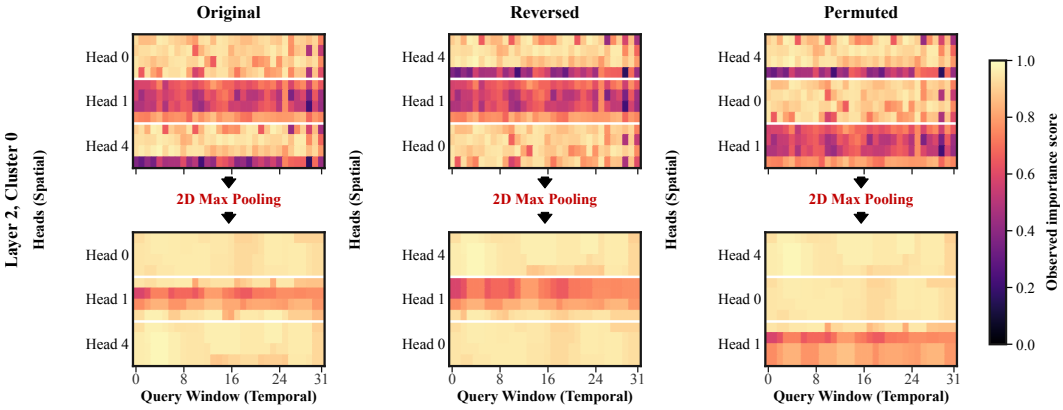


Figure 13: Effect of head ordering in Layer 2, Cluster 0. The top row shows input maps under the original, reversed, and a fixed random ordering of KV-head blocks, preserving query-head order within each block. The bottom row shows cluster-local 3×3 max-pooled maps.

Table 5: Detailed scores of 13 datasets on RULER ($B_{\text{total}} = 1024LH$).

Method	cwe	fwe	niah_mk1	niah_mk2	niah_mk3	niah_mq	niah_mv	niah_s1	niah_s2	niah_s3	qa_1	qa_2	vt	Avg.
Llama-3.1-8B-Instruct, 32K RULER, $B_{\text{total}} = 1024LH$														
FullCache	45.00	95.33	100.00	100.00	100.00	98.25	99.75	100.00	100.00	100.00	84.00	62.00	99.50	91.06
SnapKV	21.70	92.00	96.00	39.00	33.00	95.50	92.25	**100.00**	**100.00**	97.00	39.00	46.00	97.50	73.00
CAKE	33.40	93.67	98.00	62.00	57.00	97.50	95.50	**100.00**	**100.00**	**100.00**	54.00	54.00	**98.70**	80.29
AdaKV	33.70	93.00	98.00	59.00	55.00	97.50	95.50	**100.00**	**100.00**	99.00	51.00	54.00	98.60	79.56
ReST-KV	38.50	93.67	**99.00**	65.00	58.00	98.00	96.75	**100.00**	**100.00**	**100.00**	58.00	56.00	98.60	81.66
DefensiveKV	40.30	**94.00**	**99.00**	69.00	61.00	98.00	96.50	**100.00**	**100.00**	**100.00**	58.00	56.00	**98.70**	82.35
TridentKV	**44.70**	93.00	98.00	**75.00**	**67.00**	**98.50**	**97.25**	**100.00**	**100.00**	**100.00**	**63.00**	**60.00**	97.80	**84.17**
Mistral-7B-Instruct-v0.3, 32K RULER, $B_{\text{total}} = 1024LH$														
FullCache	30.00	95.67	94.00	69.00	31.00	94.25	95.00	99.00	100.00	100.00	60.00	59.00	90.60	78.27
SnapKV	34.80	92.67	22.00	16.00	8.00	16.75	13.25	62.00	28.00	5.00	39.00	54.00	75.90	35.95
CAKE	35.60	93.00	30.00	23.00	**15.00**	25.00	24.00	68.00	41.00	10.00	45.00	**55.00**	80.40	41.92
AdaKV	31.00	93.00	25.00	16.00	10.00	21.25	17.00	55.00	35.00	6.00	43.00	**55.00**	74.70	37.07
ReST-KV	**42.90**	**94.00**	36.00	17.00	12.00	32.50	31.75	72.00	50.00	13.00	47.00	**55.00**	**81.00**	44.93
DefensiveKV	36.00	**94.00**	**45.00**	19.00	10.00	40.75	41.75	72.00	55.00	**21.00**	47.00	**55.00**	80.30	47.45
TridentKV	**42.90**	**94.00**	42.00	**24.00**	**15.00**	**42.50**	**42.50**	**73.00**	**60.00**	18.00	**49.00**	**55.00**	79.40	**49.02**
Qwen2.5-32B-Instruct, 32K RULER, $B_{\text{total}} = 1024LH$														
FullCache	90.50	96.00	99.00	90.00	90.00	100.00	99.25	100.00	100.00	100.00	82.00	74.00	100.00	93.90
SnapKV	89.30	93.67	58.00	25.00	14.00	63.75	57.00	**100.00**	98.00	99.00	44.00	52.00	**100.00**	68.75
CAKE	90.90	**95.67**	88.00	54.00	44.00	91.00	89.25	**100.00**	**100.00**	**100.00**	65.00	66.00	**100.00**	83.37
AdaKV	89.70	93.67	69.00	29.00	19.00	74.25	69.50	99.00	**100.00**	**100.00**	52.00	58.00	99.80	73.30
ReST-KV	90.70	95.33	91.00	56.00	43.00	93.25	92.00	**100.00**	**100.00**	**100.00**	68.00	69.00	**100.00**	84.48
DefensiveKV	**91.30**	95.33	92.00	59.00	47.00	93.50	92.25	**100.00**	**100.00**	**100.00**	68.00	69.00	99.70	85.16
TridentKV	90.70	94.67	**94.00**	**65.00**	**54.00**	**96.50**	**95.50**	100.00	100.00	100.00	**71.00**	**71.00**	99.20	**87.04**

H More Results on RULER Benchmark

Table 5 extends the main-text RULER comparison in Table 2 from $B_{\text{total}} = 512LH$ to $1024LH$, covering the same 13 tasks and three models at a context length of 32K tokens. The listed scores retain the main-text ranking: TridentKV has the highest average among the eviction methods on all three models and ranks first or ties for first on 10 of the 13 tasks for each model. Its advantage thus spans multiple tasks and model families at the larger cache budget.

I More Results on Needle-in-a-Haystack

We extend the Needle-in-a-Haystack evaluation in Figure 6 to examine whether TridentKV preserves retrieval quality across models under tighter cache constraints. The additional settings comprise Llama-3.1-8B-Instruct and Qwen2.5-32B-Instruct with $B_{\text{total}} = 256LH$, and Mistral-7B-Instruct-v0.3 with $B_{\text{total}} = 128LH$. For each setting, we compare TridentKV with SnapKV, AdaKV, CAKE, and DefensiveKV, using Full Cache as the reference. Figures 14, 15, and 16 demonstrate that TridentKV consistently outperforms the other eviction methods across all three model–budget combinations. Notably, the performance gap between TridentKV and the next-best method is more

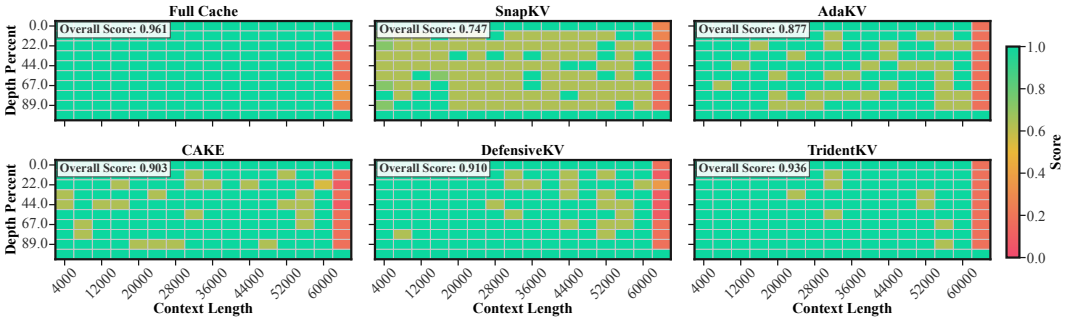


Figure 14: Needle-in-a-Haystack test results on Llama-3.1-8B-Instruct with $B_{\text{total}} = 256LH$.

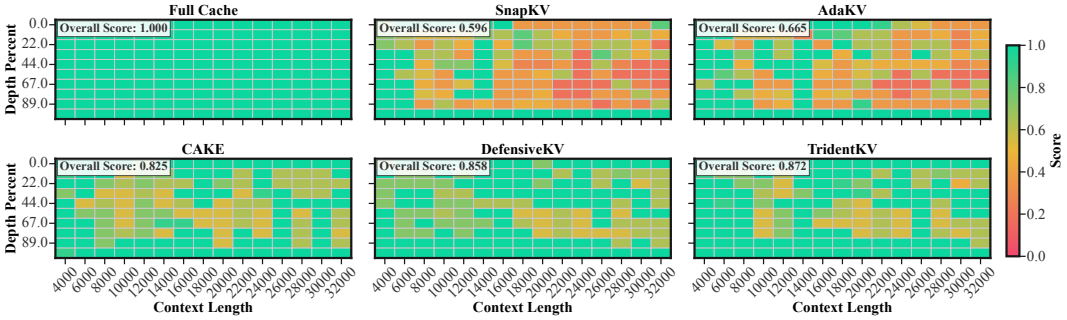


Figure 15: Needle-in-a-Haystack test results on Mistral-7B-Instruct-v0.3 with $B_{\text{total}} = 128LH$.

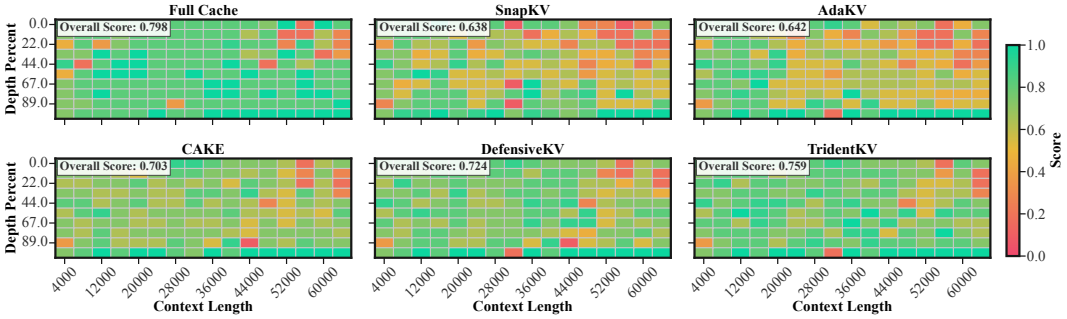


Figure 16: Needle-in-a-Haystack test results on Qwen2.5-32B-Instruct with $B_{\text{total}} = 256LH$.

pronounced at smaller cache budgets, highlighting its effectiveness in preserving retrieval quality under tighter constraints.

J Detailed Scores on LongBench

Tables 6, 7, 8, and 9 provide detailed per-dataset scores on the 16 LongBench datasets at cache budgets of 128, 256, 1024, and 2048, respectively, for Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen2.5-32B-Instruct. Together with the results at a budget of 512 in Table 1, these tables cover all evaluated cache budgets. TridentKV has the highest reported average score among the compared KV eviction methods in all 15 model–budget combinations.

K Comparison with LAVa on LongBench

We additionally compare TridentKV with LAVa (Shen et al., 2025), a KV cache eviction method that dynamically allocates budgets across both layers and heads. To focus on budget allocation, we evaluate a variant of TridentKV that uses diversity-aware budget allocation alone, with the proposed

Table 6: Detailed scores of 16 datasets on LongBench ($B_{\text{total}} = 128LH$).

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama-3.1-8B-Instruct, $B_{\text{total}} = 128LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	21.84	26.31	29.99	44.95	28.74	21.10	25.29	18.56	19.68	50.05	90.22	40.28	5.95	82.18	65.07	52.19	38.90
PyramidKV	21.56	24.10	31.22	43.54	27.98	20.40	24.49	18.22	19.59	52.25	89.14	37.78	6.65	81.70	64.52	53.50	38.54
CAKE	22.96	28.52	32.50	46.55	30.71	23.10	26.70	19.56	20.72	52.63	90.74	41.67	6.34	87.41	66.11	50.98	40.45
HeadKV	22.49	27.04	31.40	46.54	29.62	21.34	26.10	19.13	20.28	52.85	90.51	41.03	6.18	82.71	65.52	53.74	39.78
AdaKV	22.88	27.49	31.87	47.29	30.07	21.71	25.82	19.47	20.63	53.39	90.71	41.55	6.65	83.03	66.02	53.66	40.14
ReST-KV	23.40	**29.17**	33.00	48.00	31.71	**23.71**	27.15	19.94	21.12	54.08	92.45	40.07	7.84	**87.53**	66.63	54.36	41.26
DefensiveKV	**23.92**	28.66	33.66	48.53	31.84	23.19	27.75	**20.41**	**21.62**	54.82	92.64	43.12	7.15	85.77	67.77	55.71	41.66
TridentKV	23.91	29.16	**34.48**	**49.53**	**32.96**	23.70	**28.30**	20.40	21.61	**55.95**	**93.57**	**44.52**	**7.85**	87.41	**69.29**	**57.20**	**42.49**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 128LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	19.84	25.52	32.70	40.80	28.26	19.33	26.81	20.06	20.64	58.16	89.26	39.10	4.84	84.40	57.94	52.34	38.75
PyramidKV	20.68	27.27	35.74	42.08	29.92	20.19	27.89	20.96	21.56	59.86	89.80	40.34	5.10	76.94	57.50	53.29	39.32
CAKE	21.17	26.22	35.22	42.78	29.45	19.81	27.44	20.57	21.17	59.34	92.07	39.84	4.86	83.12	58.72	53.10	39.68
HeadKV	21.58	26.70	33.95	43.39	29.40	20.21	27.47	20.96	21.57	59.73	90.76	40.40	4.99	85.54	59.32	53.71	39.98
AdaKV	21.62	26.73	33.97	43.44	28.75	20.24	27.27	21.00	21.59	59.58	90.62	40.44	5.00	85.56	59.34	53.73	39.93
ReST-KV	22.35	28.18	36.77	44.45	30.89	20.99	28.80	21.72	22.32	63.32	90.95	41.46	5.19	87.50	60.35	54.44	41.23
DefensiveKV	**22.59**	28.45	37.12	44.82	31.18	21.18	29.11	**21.96**	**22.58**	63.65	**92.69**	41.83	**5.39**	87.67	60.71	55.15	41.63
TridentKV	22.58	**29.25**	**39.74**	**45.75**	**32.00**	**22.76**	**30.07**	21.95	22.57	**65.25**	92.68	**42.75**	5.38	**89.11**	**61.60**	**56.08**	**42.47**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 128LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	22.41	23.47	29.57	50.99	42.02	28.72	25.18	18.22	18.87	58.74	85.06	46.26	8.43	74.12	51.85	33.69	38.60
PyramidKV	22.33	23.61	31.46	50.78	42.40	28.31	25.00	17.57	19.14	61.30	84.44	44.02	9.11	71.85	51.35	33.49	38.51
CAKE	22.05	24.61	31.10	50.38	42.01	27.99	25.14	18.19	18.88	59.46	85.05	45.95	8.97	76.33	50.94	33.11	38.76
HeadKV	22.91	23.24	30.99	52.20	41.88	28.59	25.07	18.12	18.79	60.50	85.01	46.10	8.90	73.98	52.73	33.55	38.91
AdaKV	22.89	23.58	29.95	52.68	41.51	28.99	25.42	18.41	19.07	60.96	85.25	46.58	8.46	74.36	53.20	33.01	39.02
ReST-KV	24.10	25.45	32.42	54.37	45.04	30.67	26.33	19.13	19.81	62.37	85.80	47.90	9.47	**78.42**	53.33	32.59	40.45
DefensiveKV	**24.26**	26.29	34.08	54.60	45.77	30.87	26.51	**19.28**	**19.98**	63.16	87.52	47.97	9.53	74.41	53.56	34.69	40.78
TridentKV	24.25	**27.49**	**35.47**	**55.30**	**47.27**	**31.48**	**27.09**	19.27	19.97	**63.83**	**88.60**	**48.64**	**9.81**	76.33	**54.23**	**36.41**	**41.59**

Table 7: Detailed scores of 16 datasets on LongBench ($B_{\text{total}} = 256LH$).

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama-3.1-8B-Instruct, $B_{\text{total}} = 256LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	26.28	31.27	35.34	51.02	33.98	25.43	30.17	22.52	23.81	56.09	92.13	46.22	7.47	85.42	70.38	58.15	43.48
PyramidKV	26.75	29.64	37.60	50.59	34.04	25.39	30.12	22.84	24.46	59.22	91.55	44.62	8.65	85.51	70.72	60.38	43.88
CAKE	27.01	33.10	37.42	51.95	35.49	27.16	31.14	23.19	24.51	57.96	92.37	46.98	7.76	89.54	70.78	56.28	44.54
HeadKV	28.11	33.28	38.14	53.95	36.17	26.74	32.23	24.16	25.52	60.13	92.75	48.34	8.15	86.51	71.90	60.96	45.44
AdaKV	27.48	32.59	37.41	53.39	35.45	26.13	30.78	23.60	24.94	59.38	92.54	47.56	8.34	86.15	71.28	59.62	44.79
ReST-KV	28.58	35.02	39.22	54.71	37.82	28.92	32.81	24.60	25.97	60.65	94.10	46.63	10.02	**90.13**	72.33	60.89	46.40
DefensiveKV	**29.30**	34.59	40.08	55.40	38.11	28.45	33.61	**25.26**	**26.66**	61.52	**94.29**	49.95	9.22	88.79	73.48	62.33	46.94
TridentKV	29.29	**36.15**	**41.11**	**56.57**	**39.46**	**29.92**	**34.35**	25.25	26.65	**62.76**	94.28	**51.53**	**10.03**	89.87	**74.96**	**63.90**	**47.88**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 256LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	22.21	28.30	35.87	44.27	31.21	21.64	29.68	22.42	23.06	61.52	90.47	42.52	5.54	85.86	61.34	55.85	41.36
PyramidKV	20.73	27.30	35.76	42.12	29.95	20.22	27.91	20.99	21.59	59.86	89.74	40.37	5.11	76.63	57.52	53.32	39.32
CAKE	23.35	28.71	38.09	45.87	32.09	21.88	29.99	22.68	23.33	62.26	92.87	42.87	5.48	84.50	61.69	56.18	41.99
HeadKV	24.07	29.53	37.13	46.86	32.38	22.57	30.34	23.37	24.03	63.01	91.80	43.80	5.69	86.89	62.64	57.17	42.58
AdaKV	24.39	29.88	37.53	47.30	32.04	22.88	30.46	23.70	24.34	63.23	91.81	44.24	5.79	87.09	63.02	57.58	42.83
ReST-KV	24.59	30.76	39.69	47.54	33.59	23.14	31.42	23.91	24.56	66.12	91.86	44.50	5.84	88.50	63.27	57.51	43.55
DefensiveKV	25.86	32.20	41.33	49.25	35.10	24.31	32.90	**25.15**	**25.84**	67.61	**93.74**	46.20	**6.37**	89.18	64.84	59.48	44.96
TridentKV	**26.64**	**33.05**	**44.05**	**50.20**	**35.99**	**26.03**	**33.93**	25.14	25.83	**69.13**	93.73	**47.15**	6.36	**90.43**	**65.70**	**60.40**	**45.86**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 256LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	27.36	28.54	35.33	57.55	48.56	34.43	30.50	22.52	23.27	64.97	88.11	52.88	10.71	78.80	58.40	39.83	43.86
PyramidKV	26.23	27.65	36.19	56.06	47.64	32.82	29.20	20.88	22.64	66.20	87.01	49.30	11.03	75.84	56.62	38.37	42.73
CAKE	27.51	30.44	37.68	57.63	49.26	34.26	31.06	22.98	23.78	66.30	88.40	53.27	11.67	81.12	58.19	39.89	44.59
HeadKV	28.90	29.26	38.01	59.88	49.59	35.37	31.39	23.24	24.03	67.67	88.56	53.90	11.78	79.47	**60.37**	40.82	45.14
AdaKV	28.09	28.85	35.95	59.40	48.24	34.94	30.95	22.89	23.66	67.24	88.36	53.42	10.84	79.16	59.90	39.31	44.45
ReST-KV	29.42	30.93	38.61	60.98	51.79	36.73	31.93	23.70	24.49	68.50	88.79	**54.69**	12.06	**82.55**	59.99	38.76	45.87
DefensiveKV	**29.43**	31.69	40.18	60.99	52.31	36.75	31.95	**23.71**	**24.52**	69.05	90.11	54.53	12.05	79.01	60.00	40.84	46.07
TridentKV	29.42	**33.24**	**41.89**	**61.91**	**54.06**	**37.66**	**32.80**	23.70	24.51	**69.88**	**91.14**	54.68	**12.56**	81.12	60.36	**42.91**	**46.99**

spatial-temporal score aggregation disabled. Table 10 illustrates this comparison on 16 LongBench datasets across three models at a total cache budget of $B_{\text{total}} = 512LH$.

Table 8: Detailed scores of 16 datasets on LongBench ($B_{\text{total}} = 1024LH$).

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama-3.1-8B-Instruct, $B_{\text{total}} = 1024LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	29.90	35.24	39.54	55.49	38.10	28.98	34.08	25.81	27.23	60.46	93.38	50.69	8.80	87.53	74.00	62.45	46.98
PyramidKV	29.91	32.95	41.30	54.47	37.61	28.45	33.48	25.70	27.44	62.91	92.72	48.48	9.94	87.34	73.84	64.02	46.91
CAKE	30.01	36.41	40.91	55.60	38.92	30.15	34.37	25.91	27.33	61.49	93.39	50.65	8.87	90.84	73.73	59.82	47.40
HeadKV	29.59	34.88	39.86	55.76	37.86	28.19	33.83	25.52	26.93	61.86	93.26	50.15	8.70	87.35	73.35	62.67	46.86
AdaKV	30.43	35.81	40.81	56.94	38.80	29.00	33.91	26.29	27.72	62.79	93.51	51.15	9.49	87.79	74.14	63.02	47.60
ReST-KV	30.68	**37.34**	41.65	57.21	40.22	**31.03**	35.08	26.52	27.96	63.04	94.67	49.13	10.96	**91.00**	74.32	63.27	48.38
DefensiveKV	30.87	36.28	41.88	57.24	39.87	29.99	35.29	26.69	**28.15**	63.27	94.71	51.81	9.85	89.52	74.92	64.06	48.40
TridentKV	**30.97**	37.20	**42.25**	**57.73**	**40.57**	30.86	**35.39**	**26.78**	28.14	**63.85**	**95.13**	**52.72**	**10.97**	90.84	**75.79**	**64.97**	**49.01**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 1024LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	25.62	32.28	40.29	48.95	35.40	24.99	33.76	25.88	26.57	65.89	92.02	47.18	6.61	88.08	65.71	60.45	44.98
PyramidKV	25.42	32.86	42.02	48.69	35.80	24.83	33.56	25.72	26.41	66.06	91.98	46.90	6.56	80.84	63.86	59.85	44.46
CAKE	26.70	32.53	42.39	50.35	36.13	25.09	33.90	25.99	26.70	66.41	94.01	47.32	6.49	86.78	65.85	60.56	45.45
HeadKV	27.06	32.92	40.87	50.80	35.94	25.44	33.80	26.33	27.04	66.63	92.96	47.73	6.61	88.67	66.28	61.00	45.63
AdaKV	26.87	32.69	40.61	50.56	34.95	25.25	33.30	26.14	26.83	66.24	92.79	47.49	6.55	88.57	66.02	60.74	45.35
ReST-KV	27.37	33.94	43.19	51.17	36.91	25.81	34.63	26.65	27.34	69.30	92.92	48.11	6.70	89.97	66.58	61.01	46.35
DefensiveKV	**27.69**	34.28	43.60	51.60	37.28	26.06	35.01	**26.97**	**27.69**	69.66	**94.31**	48.55	**6.96**	90.12	**66.96**	61.74	46.78
TridentKV	27.68	**34.81**	**45.97**	**52.16**	**37.81**	**27.56**	**35.72**	26.96	27.68	**70.81**	94.30	**49.11**	6.95	**91.17**	66.95	**62.28**	**47.37**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 1024LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	29.23	30.47	37.48	59.81	50.87	36.55	32.50	24.17	24.97	67.04	89.07	55.18	11.64	80.32	60.63	42.07	45.75
PyramidKV	29.54	31.06	40.07	60.08	51.76	36.54	32.72	23.73	25.67	69.78	88.77	53.42	12.76	78.74	60.63	42.33	46.10
CAKE	29.40	32.45	39.89	59.90	51.58	36.39	33.10	24.67	25.52	68.35	89.34	55.60	12.68	82.53	60.45	42.15	46.50
HeadKV	30.51	30.88	39.84	61.72	51.52	37.15	33.08	24.63	25.47	69.33	89.33	55.82	12.62	80.71	**62.21**	42.70	46.72
AdaKV	29.29	30.07	37.32	60.82	49.72	36.28	32.23	23.95	24.75	68.52	88.96	54.89	11.43	80.13	61.31	40.73	45.65
ReST-KV	31.03	32.60	40.44	62.80	53.70	38.53	33.63	25.12	25.94	70.13	89.55	56.59	12.92	**83.64**	61.83	40.59	47.44
DefensiveKV	31.31	33.64	42.34	63.10	54.52	38.84	33.92	**25.35**	**26.21**	**70.91**	90.89	56.73	13.04	80.46	62.12	43.02	47.90
TridentKV	**31.65**	**34.71**	**43.52**	**63.47**	**55.70**	**39.21**	**34.28**	25.34	26.20	70.90	**91.57**	**57.10**	**13.23**	82.53	62.20	**44.55**	**48.51**

Table 9: Detailed scores of 16 datasets on LongBench ($B_{\text{total}} = 2048LH$).

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama-3.1-8B-Instruct, $B_{\text{total}} = 2048LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	30.50	35.88	40.21	56.18	38.76	29.55	34.71	26.35	27.78	61.12	93.56	51.39	9.02	87.84	74.53	63.10	47.53
PyramidKV	30.44	33.51	41.92	55.10	38.21	28.96	34.04	26.19	27.96	63.50	92.90	49.10	10.17	87.63	74.33	64.60	47.41
CAKE	30.51	36.95	41.47	56.18	39.47	30.64	34.91	26.37	27.80	62.04	93.54	51.23	9.05	91.04	74.18	60.38	47.86
HeadKV	30.43	35.79	40.81	56.74	38.79	28.99	34.73	26.28	27.71	62.79	93.51	51.14	9.02	87.78	74.13	63.60	47.64
AdaKV	30.60	35.97	41.00	57.13	38.98	29.15	34.08	26.44	27.87	62.97	93.57	51.34	9.55	87.87	74.28	63.20	47.75
ReST-KV	31.00	**37.68**	42.01	57.58	40.56	**31.35**	35.41	26.81	28.26	63.39	94.75	49.50	11.10	**91.12**	74.59	63.61	48.67
DefensiveKV	**31.08**	36.50	42.11	57.48	40.10	30.19	**35.51**	26.88	**28.35**	63.49	94.77	52.05	9.93	89.62	75.10	64.28	48.59
TridentKV	31.07	37.34	**42.39**	**57.87**	**40.72**	30.99	35.50	**26.91**	28.34	**63.98**	**95.14**	**52.86**	**11.11**	91.04	**75.89**	**65.09**	**49.14**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 2048LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	26.56	33.36	41.47	50.18	36.53	25.92	34.86	26.83	27.54	67.00	92.40	48.41	6.93	88.63	66.81	61.61	45.94
PyramidKV	26.75	34.39	43.71	50.42	37.39	26.14	35.12	27.07	27.77	67.61	92.49	48.62	7.00	81.86	65.44	61.50	45.83
CAKE	27.51	33.44	43.40	51.38	37.09	25.88	34.83	26.80	27.52	67.34	94.26	48.35	6.74	87.30	66.78	61.54	46.26
HeadKV	27.58	33.51	41.51	51.46	36.55	25.94	34.40	26.85	27.57	67.22	93.15	48.40	6.78	88.98	66.87	61.63	46.15
AdaKV	27.57	33.48	41.47	51.45	35.77	25.94	34.10	26.85	27.54	67.04	93.04	48.39	6.77	88.97	66.83	61.59	46.05
ReST-KV	**28.00**	**34.65**	43.97	51.96	37.65	26.41	35.35	**27.28**	27.98	69.98	93.15	48.90	6.90	90.29	67.29	61.76	46.97
DefensiveKV	27.94	34.59	43.93	51.93	37.59	26.32	35.32	27.23	27.95	69.94	**94.40**	48.87	**7.04**	90.28	67.26	62.05	47.04
TridentKV	27.99	34.64	**46.02**	**52.22**	**37.87**	**27.58**	**35.77**	27.27	**28.21**	**70.84**	94.39	**49.17**	7.03	**91.16**	**67.52**	**62.32**	**47.50**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 2048LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	30.35	31.59	38.72	61.08	52.20	37.79	33.68	25.17	25.98	68.21	89.59	56.49	12.19	81.14	61.89	43.37	46.84
PyramidKV	30.19	31.71	40.80	60.81	52.52	37.25	33.41	24.29	26.26	70.42	89.09	54.17	13.09	79.24	61.35	43.08	46.73
CAKE	29.98	33.04	40.54	60.55	52.27	37.02	33.71	25.19	26.05	68.94	89.61	56.27	12.97	82.90	61.11	42.81	47.06
HeadKV	31.05	31.41	40.44	62.32	52.16	37.74	33.65	25.11	25.96	69.87	89.59	56.44	12.89	81.10	62.80	43.31	47.24
AdaKV	30.63	31.42	38.80	62.33	51.30	37.76	33.65	25.13	25.96	69.88	89.59	56.46	12.08	81.12	**62.81**	42.28	46.95
ReST-KV	31.60	33.15	41.06	63.40	54.35	39.15	34.22	**25.61**	26.44	70.68	89.80	**57.22**	13.20	**83.99**	62.44	41.21	47.97
DefensiveKV	31.55	33.88	42.60	63.36	54.80	39.11	34.18	25.57	26.43	71.15	91.00	57.00	13.16	80.63	62.38	43.28	48.13
TridentKV	**31.62**	**34.67**	**43.49**	**63.44**	**55.68**	**39.19**	**34.25**	25.60	**26.47**	**71.20**	**91.48**	57.07	**13.21**	82.90	62.80	**44.53**	**48.60**

L Details of 16 Datasets in LongBench

As a widely used long-context benchmark, LongBench consists of 16 datasets across six task domains: single-document question answering (QA) (Kočiskỳ et al., 2018; Dasigi et al., 2021), multi-document QA (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022), summarization (Huang et al., 2021; Zhong et al., 2021; Fabbri et al., 2019), few-shot learning (Joshi et al., 2017; Gliwa et al.,

Table 10: Comparison with LAVa on 16 LongBench datasets at $B_{\text{total}} = 512LH$.

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	Hotpot	2WikiQA	Musique	GovRep	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama-3.1-8B-Instruct, $B_{\text{total}} = 512LH$																	
Full Cache	29.49	44.72	55.75	57.44	48.93	32.55	34.36	25.47	26.79	72.50	92.31	43.23	7.25	99.75	63.38	52.32	49.14
SnapKV	29.35	34.41	36.54	54.01	36.46	27.93	33.63	25.33	26.63	56.11	92.29	51.34	7.19	84.19	73.69	61.46	45.66
PyramidKV	28.69	26.37	**41.56**	49.78	34.12	25.62	31.00	24.37	26.71	**63.69**	89.66	43.19	10.48	83.06	72.54	**66.12**	44.81
CAKE	29.37	**37.94**	40.67	53.99	38.77	**31.49**	34.28	25.39	26.69	59.24	92.27	50.79	7.21	**95.43**	72.49	52.30	46.77
HeadKV	29.39	34.40	38.78	56.16	36.82	26.37	33.95	25.35	26.65	62.06	92.23	50.81	7.23	84.19	72.52	63.41	46.27
AdaKV	29.41	34.44	38.80	56.87	36.84	26.39	31.33	25.37	26.67	62.08	92.25	50.83	8.71	84.21	72.56	61.48	46.14
LAVa	29.16	37.49	39.29	56.26	39.17	31.41	34.10	25.33	26.53	61.06	95.54	42.52	**12.78**	93.61	71.71	60.36	47.27
TridentKV	**29.44**	36.26	40.51	**57.12**	**39.76**	30.04	**34.31**	**25.42**	**26.74**	62.09	**95.79**	**52.64**	10.20	89.43	**74.75**	64.46	**48.06**
Mistral-7B-Instruct-v0.3, $B_{\text{total}} = 512LH$																	
Full Cache	26.98	38.24	50.16	50.69	37.45	27.88	34.40	25.72	26.41	75.50	88.96	46.92	6.75	97.25	65.99	60.38	47.48
SnapKV	23.67	30.95	36.59	46.39	34.40	24.40	33.73	25.56	26.25	62.79	89.46	46.88	6.69	88.77	65.85	60.34	43.92
PyramidKV	23.69	**33.58**	43.15	46.41	**36.47**	24.48	33.81	**25.70**	26.37	64.15	89.48	46.80	6.71	63.96	60.36	59.20	42.77
CAKE	26.86	31.35	43.09	50.57	36.41	24.38	33.77	25.60	26.33	64.13	96.15	46.84	6.17	84.14	65.87	60.26	45.12
HeadKV	26.88	31.39	36.63	50.61	34.42	24.42	32.14	25.58	26.29	63.52	92.20	46.76	6.21	89.94	**65.97**	60.32	44.58
AdaKV	26.92	31.37	36.61	**50.67**	31.97	24.44	31.29	25.62	26.27	62.96	91.80	46.82	6.23	89.96	65.91	60.28	44.32
LAVa	26.85	33.21	42.59	50.40	36.16	24.44	33.61	25.63	26.22	70.89	91.40	46.59	6.14	93.26	65.61	58.84	45.74
TridentKV	**26.94**	33.39	**47.73**	50.53	36.31	**27.06**	**34.06**	25.65	**26.38**	**71.45**	**96.16**	**46.89**	**6.72**	**93.56**	65.78	**60.35**	**46.81**
Qwen2.5-32B-Instruct, $B_{\text{total}} = 512LH$																	
Full Cache	30.84	46.19	52.82	63.65	59.70	38.84	32.55	24.40	24.91	71.50	88.31	47.01	12.75	99.75	49.70	34.36	48.58
SnapKV	28.37	28.51	33.09	57.58	49.01	36.12	32.45	24.32	24.75	62.90	88.23	**55.80**	10.38	79.87	60.37	43.29	44.69
PyramidKV	27.04	28.03	38.87	55.65	49.05	33.47	30.69	20.88	24.85	69.73	85.93	46.99	12.69	72.30	57.53	41.34	43.44
CAKE	27.02	32.98	38.83	55.63	49.03	33.45	32.41	24.26	24.83	65.27	88.25	54.84	**12.73**	85.98	57.51	41.30	45.27
HeadKV	30.76	27.99	38.81	61.91	48.99	36.10	32.47	24.28	24.81	68.76	88.31	55.74	12.63	79.83	63.58	43.27	46.14
AdaKV	29.36	28.01	33.45	61.93	46.13	36.14	32.43	24.30	24.77	68.78	88.29	55.78	10.08	79.85	**63.60**	39.82	45.17
LAVa	30.20	29.99	36.54	62.58	52.09	37.63	32.06	24.20	24.50	68.12	87.55	55.00	12.42	**86.48**	58.91	32.61	45.68
TridentKV	**30.79**	**36.09**	**45.86**	**62.83**	**57.38**	**38.04**	**32.50**	**24.35**	**24.87**	**70.04**	**92.34**	55.25	12.55	77.34	59.16	**44.29**	**47.73**

2019; Li & Roth, 2002), synthetic tasks (Bai et al., 2024), and code generation (Guo et al., 2023; Liu et al., 2024c). The average token length across all 16 datasets is 6,711. Table 11 provides detailed information on the 16 datasets in LongBench.

Table 11: Details of 16 datasets in LongBench.

Task	Task Type	Eval metric	Avg len	Language	Sample Num
NarrativeQA	Single-Doc. QA	F1	18,409	EN	200
Qasper	Single-Doc. QA	F1	3,619	EN	200
MultiFieldQA-en	Single-Doc. QA	F1	4,559	EN	150
HotpotQA	Multi-Doc. QA	F1	9,151	EN	200
2WikiMultihopQA	Multi-Doc. QA	F1	4,887	EN	200
MuSiQue	Multi-Doc. QA	F1	11,214	EN	200
GovReport	Summarization	Rouge-L	8,734	EN	200
QMSum	Summarization	Rouge-L	10,614	EN	200
MultiNews	Summarization	Rouge-L	2,113	EN	200
TREC	Few-shot Learning	Accuracy	5,177	EN	200
TriviaQA	Few-shot Learning	F1	8,209	EN	200
SAMSum	Few-shot Learning	Rouge-L	6,258	EN	200
PassageCount	Synthetic	Accuracy	11,141	EN	200
PassageRetrieval-en	Synthetic	Accuracy	9,289	EN	200
LCC	Code	Edit Sim	1,235	Python/C#/Java	500
RepoBench-P	Code	Edit Sim	4,206	Python/Java	500

M Details of 13 Tasks in RULER

RULER (Hsieh et al., 2024) comprises 13 configurable tasks across four categories: retrieval, multi-hop tracing, aggregation, and question answering (QA). Table 12 summarizes the configurations of the original paper. Context length is controlled by varying distractor text, word-list size, or document count. The original evaluation uses 500 examples per task per length at 4K, 8K, 16K, 32K, 64K, and 128K tokens, scored with recall-based accuracy.

Table 12: Details of the 13 RULER tasks (Hsieh et al., 2024). NIAH denotes needle-in-a-haystack; essays are by Paul Graham.

Task	Task type	Input and configuration	Required output
niah_s1	Retrieval	Word–number pair; repeated noise sentences.	One number (passkey).
niah_s2	Retrieval	Word–number pair; essays.	One number.
niah_s3	Retrieval	Word–UUID pair; essays.	One UUID.
niah_mk1	Retrieval	Four word–number pairs; essays.	One queried value.
niah_mk2	Retrieval	Word–number pairs fill the context.	One queried value.
niah_mk3	Retrieval	UUID–UUID pairs fill the context.	One queried value.
niah_mv	Retrieval	One word key, four number values; essays.	All four values.
niah_mq	Retrieval	Four word–number pairs, four queries; essays.	All four queried values.
vt	Multi-hop tracing	One assignment chain, four hops; noise sentences.	All five linked variables.
cwe	Aggregation	Ten common words occurring 30 times each; others three times each.	Ten common words.
fwe	Aggregation	Synthetic words with Zeta frequencies ($\alpha = 2$); top rank reserved for noise.	Three most frequent words.
qa_1	QA	SQuAD; supporting passage mixed with distractor passages.	Single-hop answer.
qa_2	QA	HotpotQA; supporting passages mixed with distractor passages.	Multi-hop answer.

N LIMITATIONS

TridentKV is a training-free plug-in framework that enhances KV cache eviction through diversity-aware budget allocation and spatial-temporal score aggregation. For diversity-aware allocation, attention patterns characterize behavioral diversity across layers and heads, while differences in value representations and downstream effects are not directly captured by the similarity measure. The layer-allocation guarantee in Appendix B depends on an explicit relation between uniqueness, budget, and output sensitivity. Cluster quotas yield optimal score retention within each group, while their advantage over unrestricted competition depends on how behavioral importance changes. Value information and downstream sensitivity could further improve these allocation rules. For spatial-temporal aggregation, the design is primarily motivated by GQA, where multiple query heads share a KV head, and its effectiveness under other attention architectures remains to be systematically evaluated. Finally, as TridentKV operates as a plug-in enhancement to existing importance-scoring methods, its overall performance may also vary with the underlying scoring scheme.