Autonomy-of-Heads: Data-Free Sparse Attention from Frozen Query-Key Geometry

Yehan Yang[1,2], **Junyuan Shang**[4,†], **Yang Li**[1,2], **Guanqun Zhao**[3,4], **Shuohuan Wang**[4], **Dianhai Yu**[4]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Beijing University of Posts and Telecommunications [4] Baidu Inc.
{yangyehan25z,liyang23s}@ict.ac.cn, zhao-guanqun@bupt.edu.cn
{shangjunyuan, wangshuohuan, yudianhai}@baidu.com

arXiv:2608.06849v1 [cs.CL] 7 Aug 2026

Abstract

Long-context LLM inference is bottlenecked by quadratic attention computation and growing KV-cache costs. Existing sparse attention and KV-compression methods typically decide which tokens or heads to preserve from runtime attention scores, observation windows, calibration prompts, or learned gates, making head diagnosis input-dependent and costly to deploy. We propose Autonomy-of-Heads (AoH), a data-free method that identifies retrieval and streaming heads from the spectral geometry of query-key projections. AoH defines the kernel attention operator $M_h = W_K^{h\top} W_Q^h$ and uses its effective-rank as a weight-space measure of head function: concentrated spectra indicate a small number of dominant query-key matching directions and are associated with retrieval heads, whereas diffuse spectra indicate the absence of a dominant global matching direction and are associated with streaming heads. We further derive an efficient d_{head}-dimensional computation that avoids constructing the full $d_{\text{model}} \times d_{\text{model}}$ matrix. We conducted extensive experiments across models demonstrating that at 50% sparsity, AoH retains 96.5% of Full Attention performance on average while reducing prefill and decode latency by up to 41.4% and 66.0%, respectively, and KV-cache memory by 50.0% at 256K tokens.

1 Introduction

Long-context inference is increasingly central to LLM applications, particularly in agentic workflows (team at Anthropic, 2024; Team et al., 2025) and reasoning scenarios. But standard attention incurs quadratic score computation and a linearly growing KV cache. Existing sparse-attention and KV-compression methods usually decide what to keep from runtime behavior: heavy-hitter attention mass (Zhang et al., 2023; Wan et al., 2024), local windows or selectors (Xiao et al., 2024; Fu et al., 2025; Mohtashami and Jaggi, 2023; DeepSeek-AI et al., 2025), cross-layer reuse (Deshmukh et al., 2025; Gao et al., 2026; Bai et al., 2026; Brandon et al., 2024), or learned head gates (Xiao et al., 2025; Lin et al., 2026). As shown in Figure 1, these approaches are effective but input-dependent or rely on learned gates, and calibration procedures. We ask a question: *Can frozen weights alone provide a useful prior for attention-head function, independent of runtime attention scores, calibration prompts, or additional training?*

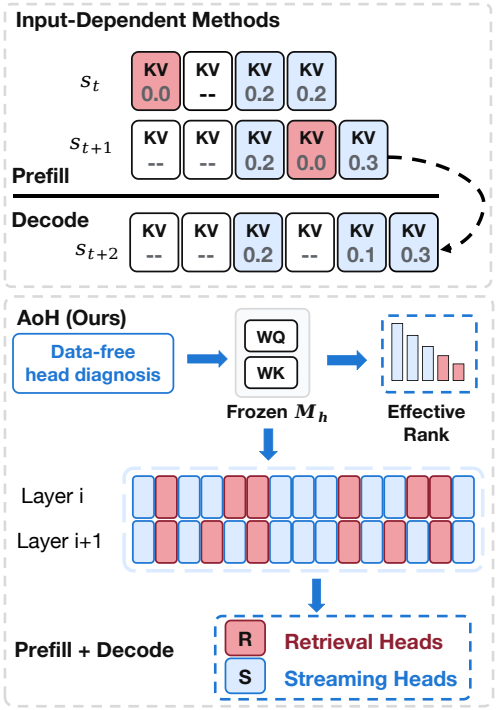

Figure 1: Input-Dependent Methods vs. AoH.

As illustrated in Figure 2, we observe that heads with concentrated spectra tend to exhibit retrieval-style attention, selectively attending to contextually relevant tokens, whereas heads with diffuse spectra mainly focus on sink and recent tokens.

This observation leads to our key insight: heads know what they know! The head-specific atten-

[†] Corresponding author.
Project: https://undground.fun/aoh/

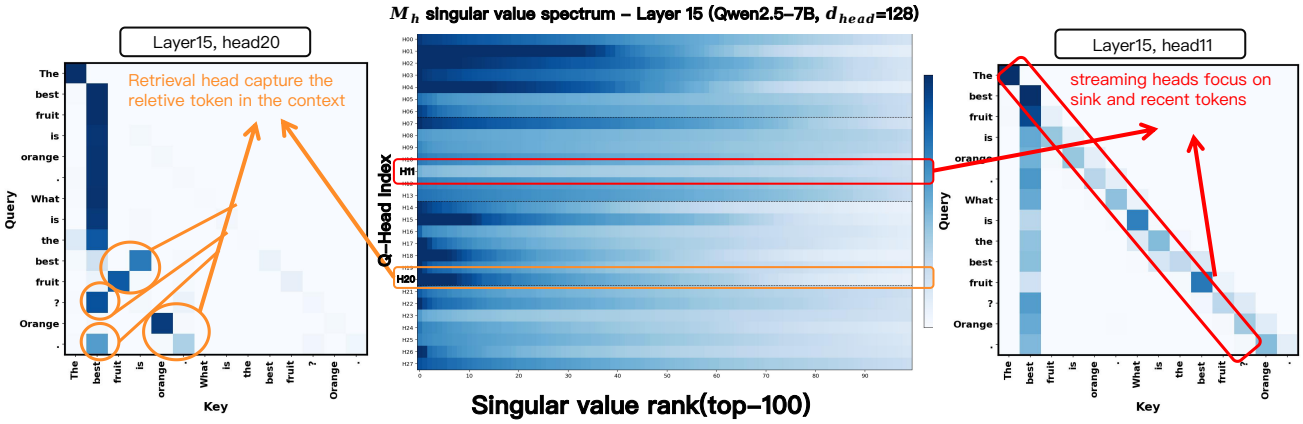

Figure 2: Visualization of the M_h singular value spectrum and attention maps in the Qwen2.5-7B model for the sentence *"The best fruit is orange. What is the best fruit? Orange."*, showing that concentrated spectra correspond to retrieval heads while uniform spectra correspond to streaming heads. Left: Retrieval heads (e.g., Layer 15, Head 20) attend selectively to contextually relevant tokens, requiring full attention.Center: M_h singular value spectrum heatmap; orange-highlighted rows (concentrated spectrum) are retrieval heads, red-highlighted rows (uniform spectrum) are streaming heads. Right: Streaming heads (e.g., Layer 15, Head 11) focus on sink and recent tokens, sufficient with sliding-window attention.

tion operator is already encoded in the frozen query-key projections. During decoding, the score of head h can be written as $\text{scores}_{h,i} = X_{\text{ctx}} W_K^{h\top} W_Q^h x_i$, where the middle operator is head-specific. We therefore define the **kernel attention matrix** $M_h = W_K^{h\top} W_Q^h$, whose spectral geometry characterizes the query-key matching directions used by head h.

We propose **Autonomy-of-Heads** (AoH), a data-free head-selection criterion based on the effective rank of M_h (Roy and Vetterli, 2007). A low effective rank indicates a few dominant matching directions and suggests a retrieval role requiring global context; a high effective rank indicates a diffuse spectrum and suggests a streaming role that can use sink and recent-window attention. Because AoH depends only on frozen weights, head labels are computed once before any prompt is processed, enabling sparse attention from prefill rather than after runtime observation. For efficient deployment, we show that the nonzero singular values of M_h can be computed from a $d_{\text{head}} \times d_{\text{head}}$ proxy, avoiding construction of the full $d_{\text{model}} \times d_{\text{model}}$ matrix. We then use the resulting head labels to build AoH-guided sparse attention: retrieval heads retain global attention, while streaming heads use bounded sink and recent-window caches. This keeps the sparse policy simple, training-free, and compatible with GQA and FlashAttention-style implementations.

We evaluate AoH on LongBench across Models. At 50% sparsity[1], AoH remains close to Full Attention and consistently outperforms baselines, random and reversed head selection, showing that the effective-rank ordering captures meaningful head-function structure rather than an arbitrary sparse subset. Efficiency results further show that AoH reduces prefill/decode latency and KV-cache memory at long context lengths. Our contributions are summarized as follows: (1) We introduce AoH, a data-free and training-free method for identifying retrieval and streaming heads directly from frozen query-key geometry.(2) We develop an effective-rank classifier for M_h and an efficient d_{head}-dimensional computation of its nonzero spectrum. (3)We instantiate the AoH as a simple sparse-attention policy and evaluate it on three long-context LLMs, where it preserves strong accuracy and improves inference efficiency at 50% sparsity.

2 Related Work

2.1 Layer-wise Sparse Attention

A major line of efficient long-context work sparsifies attention at the token or layer level. Early sparse-attention architectures such as Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) replace dense attention with fixed local, random, and global patterns, reducing the

[1]We define sparsity as $\text{sparsity} = 1 - \frac{N_{\text{full}}}{N_{\text{total}}}$, where N_{full} is the number of full-attention heads and N_{total} is the total number of attention heads. Since streaming heads retain only a small sink-plus-recent cache ($128 + 256$ tokens in this paper), this cache is negligible at long contexts such as 32K, 64K, and 128K. We therefore use the equivalent KV-cache budget approximation $\text{KV budget} \approx 1 - \text{sparsity}$ in the following analysis.

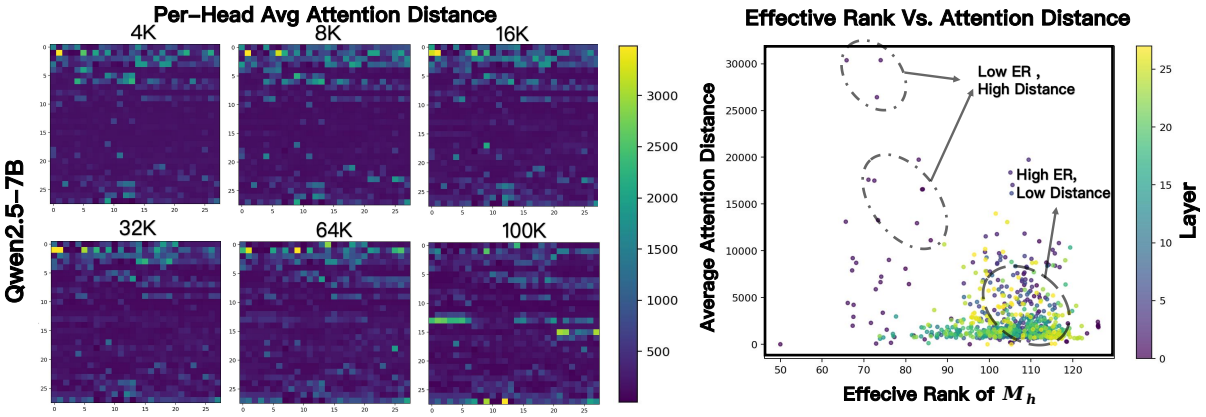

Figure 3: Empirical relationship between effective-rank and head attention distance on Qwen2.5-7B. Left: Per-head average attention distance heatmaps under context lengths from 4K to 100K. The layer–head distance patterns remain largely consistent across context lengths. Right: Scatter plot of effective-rank versus average attention distance, with points colored by layer. Heads with lower effective-rank tend to attend farther into the context, while high-ER heads concentrate on recent tokens.

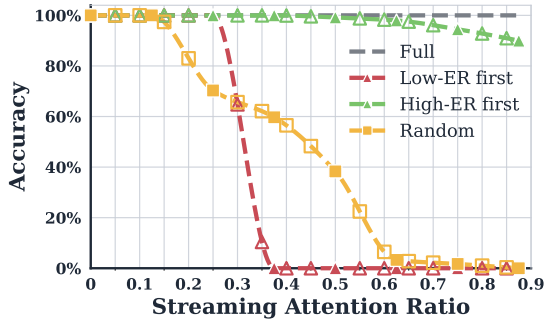

Figure 4: Passkey retrieval under progressive head-to-streaming conversion. Restricting low-ER heads to sink-and-recent attention rapidly collapses retrieval accuracy, while restricting high-ER heads has little impact.

quadratic cost of self-attention. For pretrained LLM inference, StreamingLLM (Xiao et al., 2024) combines attention sinks with a recent window, while token-eviction and KV-selection methods such as H_2O (Zhang et al., 2023), D_2O (Wan et al., 2024), NACL (Chen et al., 2024), SnapKV (Li et al., 2024), and Quest (Tang et al., 2024) retain tokens or KV caches using observed attention statistics, proxy/observation tokens, randomized eviction, or query-aware runtime estimates. These methods are effective, but they primarily decide *which tokens* to keep from input-dependent signals. AoH instead asks which heads should retain global access before seeing any input, using only frozen query-key weights.

2.2 Head-wise Sparse Attention

A second line of work exploits the functional heterogeneity of attention heads. RazorAttention (Tang et al., 2025) observes that only a small number of retrieval heads need long-range cache access, while most heads focus on local context; it keeps full cache for retrieval heads and compresses non-retrieval heads. DuoAttention (Xiao et al., 2025) similarly separates retrieval and streaming heads, using full KV cache for retrieval heads and lightweight cache for streaming heads, but obtains head labels through a trained gate. LycheeDecode (Lin et al., 2026) uses HardKuma-based routing for head classification and also studies cross-layer reuse. These works motivate head-aware sparse attention, but their head diagnosis is tied to attention observations, learned gates, or task-dependent procedures. AoH contributes a complementary data-free classifier: it assigns head or KV-group roles from the spectral geometry of frozen query-key weights.

2.3 Cross-layer Sharing

Cross-layer methods reduce redundancy by sharing KV states, representations, or sparse indices across adjacent layers. CLA (Brandon et al., 2024) shares key/value heads between layers, and YOCO (Sun et al., 2024) redesigns the decoder so that global KV cache is stored once. KASCADE (Deshmukh et al., 2025), HySparse (Gao et al., 2026), and IndexCache (Bai et al., 2026) exploit cross-layer stability for sparse index or KV reuse; HySparse in particular derives both sparse-layer token selection and KV cache from a preceding full-attention layer. These methods are largely orthogonal to AoH: they exploit redundancy across layers, while AoH identifies which heads should retain global access before observing any input. As a result, AoH can serve as a data-free head prior for cross-layer or token-selection systems without replacing their runtime selectors.

3 Observations

Before introducing Autonomy-of-Heads (AoH), we first ask whether frozen query-key geometry reflects stable and functional differences among attention heads? We present two empirical observations that motivate our work and answer this question.

Observation 1: Long-range attention behavior is stable and negatively associated with effective-rank (ER). In Figure 3, the left heatmaps show that per-head average attention distance on Qwen2.5-7B remains structurally stable across 4K–100K contexts, suggesting that long-range attention is a persistent head-level property rather than a prompt-length artifact. The right panel further shows that lower-ER heads tend to attend farther into the context, whereas higher-ER heads are more local, suggesting an overall negative association between ER and long-range attention behavior.

Observation 2: Low-ER heads are functionally important. Figure 4 evaluates passkey retrieval when different KV heads are progressively converted to streaming attention.[2] We compare three conversion orders: Low-ER first, High-ER first, and Random. For each Streaming Attention ratio, we evaluate exact-match passkey retrieval accuracy over 100 samples, with passkeys inserted at 20%, 40%, 60%, and 80% depths of the 32K context. The results show a clear functional separation. Restricting high-ER heads to sink-plus-recent attention has little effect on retrieval accuracy. In contrast, converting low-ER heads causes accuracy to collapse rapidly, indicating that they are essential for accessing remote information. Random conversion lies between the two strategies.

Together, these observations motivate Autonomy-of-Heads: using ER as a data-free weight-space criterion to assign low-ER heads to full attention and high-ER heads to sink-and-recent attention.

4 Autonomy-of-Heads

In this section, we will introduce the AoH method in detail. Prior work commonly distinguishes retrieval heads, which support long-range content lookup, from streaming heads, which mainly rely on sink and recent tokens (Xiao et al., 2025; Lin et al., 2026; Shaikh et al., 2026). AoH asks whether this distinction can be approximated without calibration prompts, runtime attention traces, or trained gates. During decoding, for the i-th query token with context $X_{\text{ctx}} \in \mathbb{R}^{T \times d_{\text{model}}}$, the attention scores for head h are: [3]

$$\underbrace{\text{scores}_{h,i}}_{T\times\text{batch-size}} = \underbrace{X_{\text{ctx}}}_{T\times d_{\text{model}}} \cdot \underbrace{W_K^{h\top}}_{d_{\text{model}}\times d_{\text{head}}} \cdot \underbrace{W_Q^{h}}_{d_{\text{head}}\times d_{\text{model}}} \cdot \underbrace{x_i}_{d_{\text{model}}\times\text{batch-size}} \tag{1}$$

For the same input, X_{ctx} and x_i are shared across heads; the head-specific component is the middle operator. We therefore define the kernel attention matrix:

$$M_h = W_K^{h\top} W_Q^h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}} \tag{2}$$

which acts as a frozen query-key matching operator for head h.

The matrix M_h connects query-side information demand to key-side information supply. For each attention head h, we compute the singular value decomposition

$$M_h = U_h \Sigma_h V_h^\top, \tag{3}$$

where $M_h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$, and $U_h, V_h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ are orthogonal matrices. Σ_h is a diagonal matrix containing the singular values of M_h in descending order. Let $\sigma_{h,1} \geq \cdots \geq \sigma_{h,r_h} > 0$ denote the nonzero singular values, where $r_h = \text{rank}(M_h)$. The right singular directions in V_h represent query-side directions that activate the head, whereas the left singular directions in U_h represent key-side directions that can be matched in the context. The singular values in Σ_h quantify the strength of these query-key matching directions. Let r_h be the number of nonzero singular values of head h, let $\hat{\sigma}_k$ denote the normalized weight of its k-th singular direction, and let $\text{eff_rank}(h)$ denote the effective rank of the head:

$$\hat{\sigma}_k = \frac{\sigma_{h,k}}{\sum_{j=1}^{r_h} \sigma_{h,j}}, \qquad k = 1, \ldots, r_h,$$
$$\text{eff_rank}(h) = \exp\left(-\sum_{k=1}^{r_h} \hat{\sigma}_k \log \hat{\sigma}_k \right) \in [1, r_h] \tag{4}$$

Here, $\text{eff_rank}(h) \in [1, r_h]$ is the exponential of the Shannon entropy of the normalized singular-value distribution. A low ER indicates a concentrated spectrum dominated by a few query-key

[2] We conduct the study on Llama3.1-8B-Instruct at 32K context length. Converted KV heads retain only the first 128 sink tokens and the most recent 256 tokens, while the remaining heads keep full-context KV Cache.

[3] Here, we consider only the case without additions such as RoPE. The ablation study in Section 6.4 shows that RoPE-aware AoH produces highly consistent head rankings with vanilla AoH.

matching directions, whereas a high ER indicates a diffuse spectrum spread across many directions. We therefore classify low-ER heads as retrieval heads and high-ER heads as streaming heads:

- Retrieval Heads: low ER heads whose attention can be driven by a small number of content-matching directions and may need to search globally over the context.
- Streaming Heads: high ER heads with more diffuse spectra and no small set of dominant global matching directions. They are less likely to require content-specific global retrieval.

Optimized Implementation. Directly constructing $M_h \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ and performing SVD is unnecessary and expensive. From Eq. (1),

$$\text{rank}(M_h) \le \min\Big(\text{rank}(W_K^{h\top}), \text{rank}(W_Q^h)\Big) \le d_{\text{head}}$$

Although M_h is a $d_{\text{model}} \times d_{\text{model}}$ matrix, its nonzero spectrum is limited by the d_{head} bottleneck. Using Sylvester's determinant theorem Karapiperi et al. (2015), AB and BA share the same nonzero eigenvalues. Therefore, the nonzero eigenvalues of $M_h^\top M_h$ can be obtained from the smaller proxy

$$C_h = (W_Q^h W_Q^{h\top})(W_K^h W_K^{h\top}) \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}, \quad (5)$$

with

$$\sigma_k(M_h) = \sqrt{\lambda_k(C_h)} \quad (6)$$

The detailed derivation is presented in Appendix L. This reduces the computation from operating on a $d_{\text{model}} \times d_{\text{model}}$ matrix to a $d_{\text{head}} \times d_{\text{head}}$ eigenvalue problem, with cost $O(d_{\text{head}}^2 \cdot d_{\text{model}})$. For Qwen2.5-7B ($d_{\text{head}} = 128$, $d_{\text{model}} = 3584$), this is approximately 4.6×10^7 operations versus 5.9×10^{10} for the full matrix construction.

Group-level Classification. For each KV Group g, every constituent Query Head h independently evaluates its effective-rank from its per-head Kernel Attention Matrix M_h, computed using the shared group key projection W_K^g and its own query projection W_Q^h. We aggregate these per-head ranks into a single group-level score by taking the *mean* across all member heads in the group. Within each layer, KV groups are then ranked by this score, and the $k = \lceil (1-s), G \rceil$ groups with the lowest ER are labelled Retrieval Groups (where s is the target sparsity and G the number of KV groups), while the remainder are Streaming Groups. All Query Heads within a group inherit the same label, so the classification maps cleanly back to the Q-Head level without ambiguity.

Algorithm 1 AoH Head Classification

Require: Frozen weights $\{W_Q^{(l)h}, W_K^{(l)h}\}$, layers $l \in [L]$, heads $h \in [H]$, budget k
Ensure: Retrieval / Streaming head sets $\{\mathcal{R}^{(l)}, \mathcal{S}^{(l)}\}_{l=1}^{L}$

for $l = 1$ **to** L **do**
  for $h = 1$ **to** H **do**
    $QQ = W_Q^{(l)h}(W_Q^{(l)h})^\top \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}$
    $KK = W_K^{(l)h}(W_K^{(l)h})^\top \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}$
    $C_h^{(l)} = QQ \cdot KK \triangleright \lambda_k(C_h^{(l)}) = \sigma_k^2(M_h^{(l)})$
    $\sigma_k = \sqrt{\max(\text{eig}(C_h^{(l)}),\ 0)}$
    $\hat{\sigma}_k = \frac{\sigma_k}{\sum_j \sigma_j}$
    $\text{eff_rank}^{(l)}(h) = \exp(-\sum_k \hat{\sigma}_k \log \hat{\sigma}_k)$
  end for
  $\pi^{(l)} = \text{argsort}_h[\text{eff_rank}^{(l)}(h)]$ ▷ ascending order
  $\mathcal{R}^{(l)} = \{h \in [H] \mid \text{eff_rank}^{(l)}(h) \le \text{eff_rank}^{(l)}(\pi_k^{(l)})\}$ ▷ lowest-k eff_rank ⇒ Retrieval
  $\mathcal{S}^{(l)} = [H] \setminus \mathcal{R}^{(l)}$ ▷ remaining ⇒ Streaming
end for

5 Deploying LLMs With AoH

Head Classification: Algorithm 1 classifies each head as retrieval or streaming using the ER of its kernel attention matrix. For head h in layer l, we compute the d_{head}-dimensional proxy $C_h^{(l)} = (W_Q^{(l)h} W_Q^{(l)h\top})(W_K^{(l)h} W_K^{(l)h\top})$, whose eigenvalues equal the squared nonzero singular values of $M_h^{(l)}$, and evaluate

$$\text{eff_rank}^{(l)}(h) = \exp\left(-\sum_k \frac{\sqrt{\lambda_k(C_h^{(l)})}}{\sum_j \sqrt{\lambda_j(C_h^{(l)})}} \log \frac{\sqrt{\lambda_k(C_h^{(l)})}}{\sum_j \sqrt{\lambda_j(C_h^{(l)})}}\right) \quad (7)$$

Reordering: Before deployment, we preprocess the model by reordering the output channels of the Query, Key, and Value projection weights according to the attention heads, which are sorted by ER in ascending order and the lowest-k units are designated retrieval heads. The budget k controls the retrieval/streaming ratio. This reordering groups retrieval heads and streaming heads, allowing for efficient slicing and concatenation operations when managing the KV cache for these two types of heads within a layer, rather than relying on scattering and gathering operations. For GQA models, the same decision is lifted to KV-group granularity

to preserve grouped-cache efficiency; details are in Section K.

Decode Stage: Each layer applies its assigned strategy based on its head type, which uses layer-specific full K/V states for Retrieval Heads and only a fixed-size window cache plus sink cache for Streaming Heads. Each head type is computed independently; their outputs are concatenated along the head dimension and projected through a shared output matrix:

$$o^{(l)} = \mathrm{Concat}\left(\underbrace{\mathrm{Attn}\left(q_{\mathcal{R}}^{(l)}, \mathrm{KV}_{\mathcal{R}}^{(l)}\right)}_{\text{Full Attn.}}, \underbrace{\mathrm{Attn}\left(q_{\mathcal{S}}^{(l)}, \mathrm{KV}_{\mathcal{S}}^{(l)}\right)}_{\text{SWA}}\right) W_O^{(l)} \tag{8}$$

where $\mathrm{Concat}(\cdot)$ denotes concatenation along the head axis, followed by reshaping to d_{model}.

Chunked-Prefill Stage: AoH is compatible with standard chunked prefill (Agrawal et al., 2023; Kwon et al., 2023) using FlashAttention-2 (Dao, 2023). Retrieval Heads attend to the accumulated prefix and retain their layer-specific $O(T)$ KV states. For Streaming Heads, after processing each chunk of size K, we evict all but s_{sink} sink tokens and s_{recent} recent tokens. Thus, each subsequent chunk attends to at most $s_{\text{sink}} + s_{\text{recent}}$ tokens. For a sequence of length T, this reduces the prefill cost from $O(T^2)$ to $O(TK)$ for fixed cache sizes, while bounding persistent KV storage by $O(s_{\text{sink}}+s_{\text{recent}})$ and peak working memory by $O(K)$.

This design requires no specialized kernels beyond standard chunked FlashAttention and remains compatible with batched serving, which can further enhance LLM efficiency in serving scenarios with large sizes.

6 Experiment

6.1 Setups

Datasets and models We evaluate on the LongBench (Bai et al., 2024), a comprehensive long-context understanding suite covering 21 tasks across six categories; Appendix D lists the task names. Each task is evaluated using its official metric, and we report the arithmetic mean over all 21 tasks as the primary aggregate score. The main models are Qwen2.5-7B (Yang et al., 2024) (28 layers, 28 Q-heads, 4 KV-heads, GQA group size 7) and Qwen3-8B (Yang et al., 2025) (36 layers, 32 Q-heads, 8 KV-heads).

Implementation Details and baseline We implement AoH in PyTorch (Paszke et al., 2019) with RoPE (Su et al., 2024), RMSNorm (Zhang and Sennrich, 2019), Eager Attention (Vaswani et al., 2017; Wolf et al., 2020). Unless otherwise stated, streaming heads use sink size $s_{\text{sink}} = 128$ and recent-window size $s_{\text{recent}} = 256$. We compare AoH with Full Attention, H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024), Quest (Tang et al., 2024), DuoAttention (Xiao et al., 2025), and two controlled head-selection baselines: $\text{Random}_{\text{avg}}$ and Reverse. For all baseline methods, we ensure a fair comparison under comparable KV-cache costs. Table 1 summarizes the KV-cache storage and attended-KV budgets used by each method. Except for Quest, which uses a small attended-token budget but still stores the full KV cache, the sparse baselines are configured with comparable storage and attention budgets. All LongBench evaluations use a maximum sequence length of 32,768 and greedy decoding.

Table 1: KV Budget-matched setting for LongBench evaluation. H denotes Heads. RH denotes retrieval heads, and SH denotes streaming heads. Storage KV denotes the KV-cache footprint, while attended KV denotes the KV entries participating in attention computation.

Method	Sparse Policy	Storage KV	Attended KV
Full Attention	32K tokens × 100% H	100%	100%
Quest	32K tokens × 100% H	100%	≈6.25%
SnapKV	16K tokens × 100% H	≈50%	≈50%
H_2O	16K tokens × 100% H	≈50%	≈50%
DuoAttention	32K × 50% RH; 384 × 50% SH	≈50.6%	≈50.6%
AoH	32K × 50% RH; 384 × 50% SH	≈50.6%	≈50.6%

6.2 Main Results

Table 2 reports LongBench results on Qwen3-8B[4] and Llama3.1-8B-Instruct[5], with Qwen2.5-7B[6] results provided in Appendix F. With 50% sparsity, AoH achieves average scores of 38.78 and 47.55 on the two main models, respectively, remaining close to Full Attention while reducing KV budgets by half. The gap to dense attention is small: AoH trails Full Attention by only 0.32 points on Qwen3-8B and 1.10 points on Llama3.1-8B. AoH also outperforms the sparse baselines in average score across all models. The Random and Reverse ablations further validate the ER criterion: under the same sparsity budget, AoH substantially outperforms both random head selection and reversed ER selection. These results show that low-rank query-key geometry identifies heads that are especially

[4] https://huggingface.co/Qwen/Qwen3-8B

[5] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[6] https://huggingface.co/Qwen/Qwen2.5-7B

Table 2: LongBench results on Qwen3-8B and Llama3.1-8B-Instruct. All AoH variants use a retrieval/streaming ratio of 50%. Random$_{\text{avg}}$ randomly selects the same number of retrieval heads, and Reverse keeps the highest ER heads as retrieval heads. AoH-RoPE uses a RoPE-aware ER score computed with relative RoPE rotations up to $\Delta = 32$K.

Model	Method	SQA	MQA	Summ	Fewshot	Synthetic	Code	Avg
Qwen3-8B	*Baselines*							
	Full Attention (Dense)	25.19	15.97	21.96	63.49	66.67	57.38	**39.10**
	SnapKV	24.04	14.62	20.25	53.80	66.45	54.09	36.11
	Quest	22.87	13.71	22.71	57.94	63.82	55.82	36.76
	H_2O (Eviction-based)	17.80	13.71	21.05	59.92	59.63	57.69	35.44
	DuoAttention (Head-wise)	22.04	14.63	20.86	58.65	64.32	56.27	36.68
	Ours							
	AoH	24.18	16.59	21.51	62.55	66.18	58.27	<u>38.78</u>
	AoH-RoPE	24.20	16.54	21.48	62.36	66.43	58.38	38.78
	AoH-Random$_{avg}$	14.72	12.99	19.40	45.03	48.59	52.89	29.53
	AoH-Reverse	10.61	11.87	17.73	39.15	11.96	48.56	21.45
Llama3.1-8B -Instruct	*Baselines*							
	Full Attention (Dense)	47.55	41.31	26.11	63.45	67.43	52.79	**48.65**
	SnapKV	42.98	41.07	24.89	55.03	67.83	51.16	45.80
	Quest	44.94	37.10	25.54	60.86	62.97	52.47	45.85
	H_2O (Eviction-based)	35.18	35.72	24.47	59.84	60.98	53.73	43.39
	DuoAttention (Head-wise)	44.38	35.74	24.14	61.49	63.57	54.14	45.81
	Ours							
	AoH	46.54	39.28	25.94	61.85	64.95	54.58	<u>47.55</u>
	AoH-RoPE	46.63	39.09	25.57	62.89	64.54	54.42	47.58
	AoH-Random$_{avg}$	19.71	23.83	20.29	48.04	31.83	52.88	30.89
	AoH-Reverse	17.52	20.87	18.99	34.46	8.84	47.84	23.31

important for preserving long-context performance.

6.3 Efficiency Results

AoH reduces KV-cache memory nearly proportionally to the fraction of streaming heads, and its latency benefits become more pronounced at longer contexts where attention and cache access dominate runtime. As shown in Figure 5 and Table 11, at 75% sparsity, AoH achieves up to $3.24\times$ prefill speedup and $9.14\times$ decode speedup at 256K context, while reducing KV-cache memory by up to $3.98\times$ on Llama3.1-8B. Additional efficiency results on Qwen2.5-7B and Qwen3-8B are provided in Appendix H.

6.4 Ablation Studies

Ablation of head ordering. We first isolate whether AoH's ER ordering provides a meaningful head-function prior. We compare AoH with two controlled alternatives under the same sparsity budget. Random$_{\text{avg}}$ selects the same number of retrieval heads uniformly at random, while Reverse flips the AoH ordering by treating high ER heads as retrieval heads. Table 2 shows the quality of the head-ranking criterion. We also show the detailed results in Section I.

Ablation of hyperparameters. We ablate the main hyperparameters: the sparsity ratio and the streaming-cache size. As shown in Figure 6, AoH achieves the best trade-off between performance and computational overhead at around 50% sparsity. We further study the sensitivity to sink size and recent-window size. As shown in Table 3, AoH is relatively robust to these streaming-cache hyperparameters across a range of sparsity levels. We choose a sink size of 128 and a recent-window size of 256, which provides a good balance between performance and KV-cache reduction.

Ablation of Kernel spectral metric. We further compare effective-rank (Roy and Vetterli, 2007) with three data-free spectral baselines computed from the same frozen query-key kernel matrix. For each attention head, we form $M_h = W_K^{h\top} W_Q^h$, or equivalently compute its singular spectrum from the low-dimensional Gram proxy

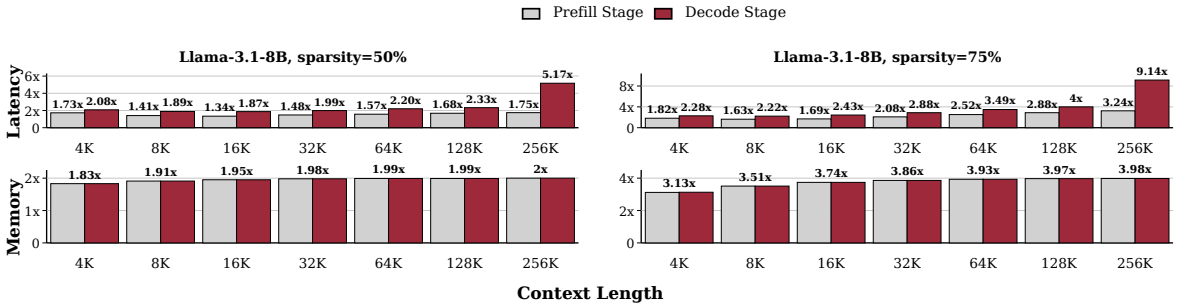

Figure 5: AoH Prefill and Decoding Efficiency Across Context Lengths vs. Full Attention. Gray and red bars report AoH gains in the prefill and decode stages. The top row shows latency speedup, and the bottom row shows KV-cache memory reduction under 50% and 75% sparsity.

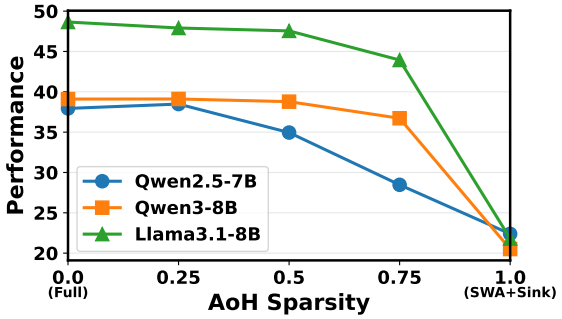

Figure 6: Performance comparison of three LLMs under different AoH sparsity.

$(W_Q^h W_Q^{h\top})(W_K^h W_K^{h\top})$. We then derive three alternative head-selection scores: the Frobenius norm $\|M_h\|_F$, which measures the overall kernel energy; the spectral norm $\|M_h\|_2 = \sigma_1$, which measures the strongest query-key matching direction; and the stable rank $\frac{\|M_h\|_F^2}{\|M_h\|_2^2}$ (Rudelson and Vershynin, 2007), which estimates spectral dimensionality while emphasizing dominance by the largest singular value. Under the same retrieval-head budgets as AoH, we rank heads within each layer according to each metric and generate the corresponding retrieval/streaming heads. As shown in Table 4, effective-rank (Ours) achieves the best average score. This indicates that AoH benefits from measuring spectral concentration rather than simply selecting heads with large kernel magnitude or a single dominant singular direction.

Ablation of GQA group-level aggregation. We also ablate how query-head scores are aggregated into GQA group-level decisions. As shown in Table 5, mean and min aggregation perform nearly identically, with average scores of 47.55 and 47.60, respectively. The small difference suggests that AoH is not sensitive to this choice. In contrast, max aggregation is noticeably worse, dropping to 45.82 on average. This indicates that max aggregation is too coarse: a single high-rank query head can dominate the group score and obscure lower-rank retrieval-oriented heads within the same KV group. We therefore use mean aggregation as the default because it is stable, simple, and less sensitive to individual outlier heads.

Table 3: LongBench-Qasper results of AoH sink-size and recent-window sensitivity on Llama-3.1-8B-Instruct. Bold indicates the settings for our main experiment.

Sink	Recent	sp=0.25	sp=0.50	sp=0.75	sp=1.0
128	128	43.97	43.15	37.01	10.13
64	128	44.15	41.53	35.57	11.96
32	128	43.82	40.75	34.68	10.84
16	128	43.66	41.00	35.61	10.53
4	128	43.74	41.37	35.63	10.95
128	**256**	**43.86**	**43.34**	**39.20**	**13.53**
64	256	44.55	41.60	38.29	14.39
32	256	44.16	41.21	37.73	13.12
16	256	43.89	41.40	37.20	13.42
4	256	44.28	41.71	39.54	14.14
128	512	44.53	43.54	42.95	17.41
64	512	44.01	42.42	39.52	16.21
32	512	44.44	41.56	38.39	16.91

Ablation of AoH-RoPE. Vanilla AoH computes the ER of each head from the frozen query-key kernel without explicitly inserting positional rotations. To examine whether RoPE changes the head-level ordering used by AoH, we construct a RoPE-aware variant. For a relative distance Δ, we insert the RoPE relative rotation R_Δ into the head-specific QK kernel and compute

$$C_{h,\Delta}^{(l)} = \left(R_\Delta W_Q^{(l)h} W_Q^{(l)h\top} R_\Delta^\top\right)\left(W_K^{(l)g(h)} W_K^{(l)g(h)\top}\right) \tag{9}$$

where h denotes a query head in layer l and $g(h)$ denotes its corresponding KV group. We compute the ER of $C_{h,\Delta}^{(l)}$ as the RoPE-aware AoH score. For each maximum relative distance D, we average scores over logarithmically spaced $\Delta \le D$, com-

Table 4: Ablation of spectral metrics for head classification on LongBench with Llama-3.1-8B-Instruct at 50% sparsity. ER denotes effective-rank, the metric used by AoH. The best result in each row is **bolded**.

Metric	Frobenius	Spectral	Stable	**ER (Ours)**
SQA	33.55	36.46	23.02	**46.54**
MQA	28.97	35.29	27.85	**39.28**
Summ	22.40	23.03	21.17	**25.94**
Fewshot	50.11	54.46	56.04	**61.85**
Synthetic	64.20	42.99	22.22	**64.95**
Code	54.14	52.78	51.30	**54.58**
Avg	40.05	39.59	32.46	**47.55**

Table 5: Ablation of GQA group-level aggregation strategies on LongBench with Llama-3.1-8B-Instruct at 50% sparsity. The best result is **bolded**.

Metric	GQA-Max	GQA-Min	GQA-Mean(ours)
SQA	44.94	45.90	**46.54**
MQA	38.44	38.03	**39.28**
Summ	24.38	25.26	**25.94**
Fewshot	56.72	**63.47**	61.85
Synthetic	66.20	**66.35**	64.95
Code	52.86	**54.91**	54.58
Avg	45.82	**47.60**	47.55

pare the resulting head ordering with vanilla AoH using layer-wise Spearman correlation, and report the mean across layers.

As shown in Table 6, vanilla AoH and RoPE-aware AoH produce highly consistent head rankings across all tested models and distance ranges. The mean layer-wise Spearman correlation remains above 0.98 even at 128K relative distance. As shown in Table 2, vanilla AoH performs comparably to AoH-RoPE on both Qwen3-8B and Llama-3.1-8B-Instruct. This suggests that although RoPE changes the relative-position phase of query-key interactions, it largely preserves the head-level spectral ordering exploited by AoH.

7 Conclusion

We presented Autonomy-of-Heads (AoH), a data-free and training-free method for identifying retrieval and streaming heads from frozen query-key geometry. By analyzing the effective-rank of the kernel attention matrix $M_h = W_K^{h\top} W_Q^h$, AoH replaces calibration data and learned gates with a one-time weight-space classifier. Across multiple long-context LLMs, AoH preserves 96.5% Full Attention performance on average at 50% sparsity and consistently outperforms random, reversed head selection and trainging-free baselines. On the hardware side, AoH achieves up to $3.24\times$ prefill speedup and $9.14\times$ decode speedup at 256K context on Llama3.1-8B, while reducing KV-cache memory by up to $3.98\times$ under 75% sparsity. More broadly, AoH provides a simple and extensible head prior for sparse attention, and can be combined with existing KV-cache compression or token-selection methods to improve long-context inference without additional training.

Table 6: RoPE-aware AoH ranking stability. We report the mean layer-wise Spearman correlation between vanilla AoH scores and RoPE-aware AoH scores under different maximum relative distances.

Model	**1K**	**4K**	**8K**	**16K**	**32K**	**64K**	**128K**
Qwen2.5-7B	0.992	0.991	0.991	0.990	0.990	0.990	0.989
Qwen3-8B	0.987	0.987	0.986	0.986	0.985	0.984	0.984
Llama3.1-8B	0.990	0.989	0.988	0.987	0.987	0.986	0.985

Limitations

Our evaluation focuses on decoder-only LLMs with standard attention variants, including GQA-based models. Although the effective-rank criterion is simple and architecture-agnostic at the level of query-key projections, we have not exhaustively tested it on encoder-decoder models, multimodal LLMs, or models with heavily modified attention mechanisms.

AoH uses a fixed sparsity budget to split heads into retrieval and streaming groups. In this work, the budget is selected manually and kept constant across layers for simplicity. A more adaptive budget, potentially varying by layer, model, or deployment constraint, may further improve the accuracy–efficiency trade-off. We leave automatic budget selection and finer-grained head policies for future work.

References

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S Gulavani, and Ramachandran Ramjee. 2023. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills. *arXiv preprint arXiv:2308.16369*.

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

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. 2026. Indexcache: Accelerating sparse attention via cross-layer index reuse. *arXiv preprint arXiv:2603.12201*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. Reducing transformer key-value cache size with cross-layer attention. *Advances in Neural Information Processing Systems*, 37:86927–86957.

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

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Aixin Liu DeepSeek-AI, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Dhruv Deshmukh, Saurabh Goyal, Nipun Kwatra, and Ramachandran Ramjee. 2025. Kascade: A practical sparse attention method for long-context llm inference. *arXiv preprint arXiv:2512.16391*.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. 2025. Sliding window attention training for efficient large language models. *arXiv preprint arXiv:2502.18845*.

Yizhao Gao, Jianyu Wei, Qihao Zhang, Yu Cheng, Shimao Chen, Zhengju Tang, Zihan Jiang, Yifan Song, Hailin Zhang, Liang Zhao, and 1 others. 2026. Hysparse: A hybrid sparse attention architecture with oracle token selection and kv cache sharing. *arXiv preprint arXiv:2602.03560*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Anna Karapiperi, Michela Redivo-Zaglia, and Maria Rosaria Russo. 2015. Generalizations of sylvester's determinantal identity. *arXiv preprint arXiv:1503.00519*.

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

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 22947–22970. Curran Associates, Inc.

Gang Lin, Dongfang Li, Zhuoen Chen, Yukun Shi, Xuhui Chen, Baotian Hu, and Min Zhang. 2026. Lycheedecode: Accelerating long-context llm inference via hybrid-head sparse decoding. *arXiv preprint arXiv:2602.04541*.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Random-access infinite context length for transformers. *Advances in Neural Information Processing Systems*, 36:54567–54585.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Olivier Roy and Martin Vetterli. 2007. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.

Mark Rudelson and Roman Vershynin. 2007. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):21–es.

Khalid Shaikh, Asmit Kumar Singh, Rebecca Christopher Dsouza, and Shikhar Shiromani. 2026. Linear predictability of attention heads in large language models. *arXiv preprint arXiv:2603.13314*.

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

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang,

and Furu Wei. 2024. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2025. Razorattention: Efficient KV cache compression through retrieval heads. In *The Thirteenth International Conference on Learning Representations*.

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

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, and 1 others. 2025. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*.

Engineering team at Anthropic. 2024. Building effective agents. *https://www.anthropic.com/engineering/building-effective-agents*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, Longyue Wang, and 1 others. 2024. D2o: Dynamic discriminative operations for efficient long-context inference of large language models. *arXiv preprint arXiv:2406.13035*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, volume 2025, pages 37228–37253.

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

Xiaomi MiMo Team. 2026. Mimo-v2.5. https://huggingface.co/collections/XiaomiMiMo/mimo-v25.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in neural information processing systems*, 32.

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

7.1 Appendices

A Further discussion on the kernel attention matrix M_h

The head-specific part of attention is encoded in the frozen query-key projections. During decoding, the attention score of head h for a query token can be written as $\text{scores}_{h,i} = X_{\text{ctx}} W_K^{h\top} W_Q^h x_i$, where the context matrix X_{ctx} and query vector x_i are shared across heads, while the middle operator $W_K^{h\top} W_Q^h$ is head-specific. We therefore define the **kernel attention matrix** $M_h = W_K^{h\top} W_Q^h$, a frozen query-key matching operator whose spectral geometry summarizes how head h maps query-side information demand to key-side information supply. Intuitively, the spectrum of M_h describes how many dominant query-key matching directions a head relies on. A concentrated spectrum indicates that a few stable matching directions dominate the head's behavior, which is consistent with retrieval heads that search globally for relevant content. A diffuse spectrum indicates that no small set of global matching directions dominates, which is consistent

with streaming heads that mainly rely on sink and recent tokens.

B Additional Observations

To test whether this association also appears beyond Qwen2.5-7B, we quantify it on Qwen3-8B in Figure 7. For each layer, we compute Spearman correlations across heads between effective-rank and three empirical behavior metrics: average attended distance, attention entropy, and far-token mass. Distance and entropy exclude sink tokens, and far-token mass measures attention probability assigned to tokens farther than 8K positions. On Qwen3-8B under 16K–64K contexts, effective-rank is consistently negatively correlated with long-range attention behavior, indicating that frozen query-key geometry provides a meaningful signal of head function.

C AoH Forms the Upper Envelope

Shown in Figure 8, AoH forms the upper envelope of the accuracy–sparsity trade-off, with all baseline methods lying below the AoH curve under comparable sparsity budgets.

D LongBench Introduction

We evaluate AoH on the LongBench (Bai et al., 2024), a comprehensive long-context understanding suite covering 21 tasks across six categories: single-document QA (narrativeqa, qasper, multifieldqa_en, multifieldqa_zh), multi-document QA (hotpotqa, 2wikimqa, musique, dureader), summarization (gov_report, qmsum, multi_news, vcsum), few-shot learning (trec, triviaqa, lsht, samsum), synthetic retrieval (passage_count, passage_retrieval_en, passage_retrieval_zh), and code completion (lcc, repobench-p).

E Details on Baseline Budgets

Table 7 reports the persistent KV-cache memory used by each baseline at the maximum 32K LongBench context length with BF16 KV caches. Full Attention stores all KV states and defines the 100% reference budget. Quest has the same KV storage footprint because it retains the full cache, although only a subset of entries participates in attention computation. SnapKV and H_2O allocate a 16K-token cache for all heads, corresponding to roughly 50% of the full KV storage. For head-wise methods, DuoAttention and all AoH variants keep full 32K caches for 50% retrieval heads and only a 512-token sink/recent cache for the remaining streaming heads, giving a storage budget of approximately 50.6%. Since AoH, AoH-Random, and AoH-Reverse use identical KV budgets, their differences in LongBench accuracy reflect the quality of the retrieval-head assignment.

Table 7: KV-cache memory footprint under the 32K LongBench setting. KV memory is computed for the maximum 32K context using BF16 KV caches.

Method	Qwen2.5-7B KV	Qwen3-8B KV	Llama3.1-8B KV
Full Attention	1.75 GB	4.50 GB	4.00 GB
Quest	1.75 GB	4.50 GB	4.00 GB
SnapKV	0.875 GB	2.25 GB	2.00 GB
H_2O	0.875 GB	2.25 GB	2.00 GB
DuoAttention	0.885 GB	2.276 GB	2.023 GB
AoH (sp=50%)	0.885 GB	2.276 GB	2.023 GB
AoH-Random (sp=50%)	0.885 GB	2.276 GB	2.023 GB
AoH-Reverse (sp=50%)	0.885 GB	2.276 GB	2.023 GB

F Additional LongBench Results

Table 9 shows the additional LongBench results on Qwen2.5-7B. We also include a complementary MoE evaluation to examine whether AoH remains applicable beyond dense transformer models.

Experimental setup for MoE models. We further evaluate AoH on LongBench using Qwen3-30B-A3B-Instruct-2507, a Mixture-of-Experts (MoE) model with 30B total parameters and 3B activated parameters per token. Compared with dense models, MoE architectures introduce an additional architectural component through expert routing; however, AoH operates only on the attention layers, where it compresses and allocates the KV cache at the attention-head level. We keep all hyperparameters, including attention sink tokens and window size, identical to the default AoH configuration used in the main LongBench table, without MoE-specific tuning.

Reults for MoE models. As shown in Table 8, AoH remains effective on the MoE model: with only 50% KV heads kept, it achieves an average LongBench score of 48.79, close to the Full Attention score of 50.95, while preserving comparable performance on the code category.

G Additional PassKeyRetrieval Results

Figure 9 shows the single-needle passkey retrieval accuracy of AoH at 50% sparsity across context lengths from 1K to 128K and insertion depths from 0% to 100%. AoH maintains near-perfect retrieval accuracy across almost all settings, with degrada-

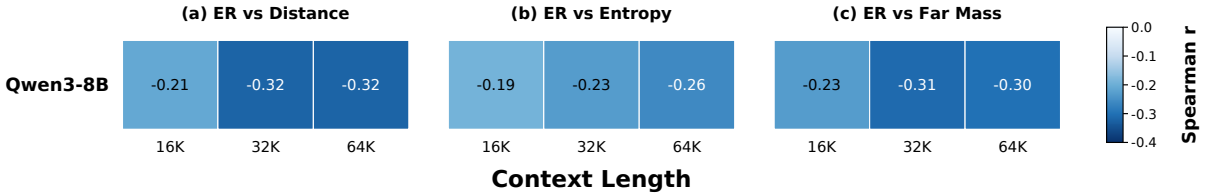

Figure 7: Correlation between effective-rank and empirical attention behavior. Negative correlations indicate that higher-ER heads are more local, while lower-ER heads are more associated with long-range attention.

Table 8: LongBench results on Qwen3-30B-A3B-Instruct-2507 (MoE).

Method	SQA	MQA	Summ	Fewshot	Synthetic	Code	Avg
Full	49.03	44.34	21.99	66.78	72.00	62.62	**50.95**
AoH ($sp = 50\%$)	44.89	42.01	20.55	64.75	70.17	62.66	48.79

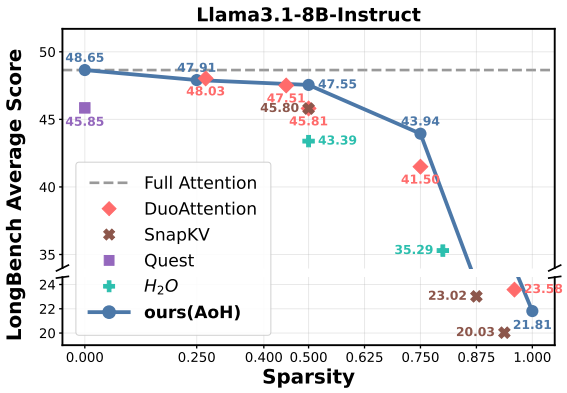

Figure 8: Accuracy-efficiency trade-off of AoH.

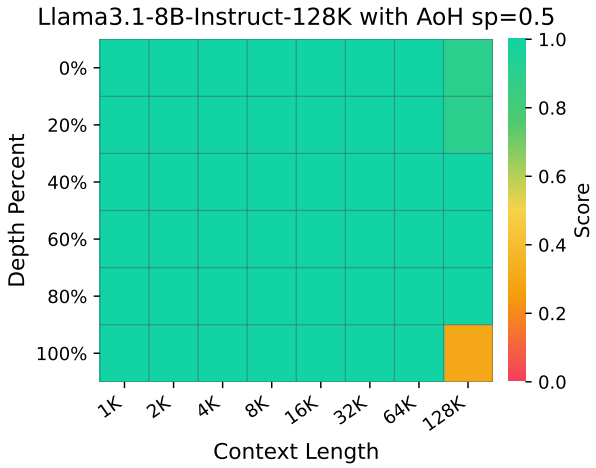

Figure 9: Single-needle passkey retrieval accuracy of AoH at 50% sparsity across context lengths and insertion depths.

tion only at the most challenging 128K context and 100% insertion depth.

H Additional Efficiency Results

Measurement protocol. To isolate the effect of head-level sparsity from low-level kernel engineering, the decode latencies reported here are measured under a deliberately simple attention backend. Both Full and AoH use *eager* attention (Vaswani et al., 2017; Wolf et al., 2020), i.e. vanilla PyTorch $\mathrm{softmax}(\frac{QK^\top}{\sqrt{d}})V$ (with the softmax accumulated in FP32 and cast back to bf16). All numbers are single-sequence ($\mathrm{batchsize} = 1$), single-token decode latencies in bf16.

Results Tables 11, 13, and 12 provide the full efficiency results for AoH on LLaMA3.1-8B, Qwen3-8B, and Qwen2.5-7B, including prefill latency, decode latency, and KV-cache memory.

I Additional Ablation Results

Ablation of head ordering. Figure 10 shows that AoH consistently preserves stronger LongBench performance across task categories and KV-cache budgets. $\mathrm{Random}_{\mathrm{avg}}$ performs substantially worse, showing that sparse attention is sensitive to which heads retain global context. Reverse is usually the weakest variant, confirming that the direction of the effective-rank criterion matters: low-rank query-key geometry identifies retrieval-oriented heads, whereas high-rank heads are better treated as streaming heads. Together with the aggregate results in Table 2, this ablation shows the quality of the head-ranking criterion.

Hyperparameter sensitivity on ShortBench. We also use MMLU as a representative short-context knowledge task. Shown in Tables 10,

Table 9: LongBench results on Qwen2.5-7B. All AoH variants use a retrieval/streaming ratio of 0.5 unless otherwise stated. $Random_{\text{avg}}$ randomly selects the same number of retrieval heads, and Reverse keeps the highest effective-rank heads as retrieval heads. AoH-RoPE-aware uses a RoPE-aware effective-rank score computed with relative RoPE rotations up to $\Delta = 32$K. sp refers to the sparsity ratio: $\text{sparsity} = 1 - \frac{N_{\text{full}}}{N_{\text{total}}}$.

Method	SQA	MQA	Summ	Fewshot	Synthetic	Code	Avg
Baselines							
Full Attention (Dense)	31.19	33.45	24.39	61.83	28.51	54.93	**37.94**
SnapKV	28.98	23.66	22.47	51.42	29.47	52.02	33.27
Quest	28.75	29.92	22.00	58.62	20.37	51.34	34.33
H_2O	21.16	25.62	20.96	55.38	10.19	51.19	29.79
DuoAttention (Head-wise)	21.78	21.97	18.31	52.68	6.14	50.48	27.54
Ours							
AoH (sp=50%)	29.93	29.28	22.05	60.04	19.11	55.68	<u>34.95</u>
AoH-RoPE-aware(sp=50%)	29.88	29.60	22.13	59.41	18.66	55.31	34.79
$AoH - Random_{\text{avg}}$(sp=50%)	17.22	23.79	18.55	45.12	5.81	50.39	25.57
$AoH - Reverse$(sp=50%)	15.53	21.94	18.41	38.93	6.56	47.51	23.52

Table 10: ShortBench-MMLU accuracy of AoH sink-size and recent-window sensitivity on Llama-3.1-8B-Instruct.

Sink	Recent	sp=0.25	sp=0.50	sp=0.75	sp=1.0
128	128	0.600	0.595	0.605	0.600
64	128	0.600	0.595	0.605	0.600
32	128	0.600	0.595	0.605	0.600
16	128	0.600	0.595	0.605	0.600
4	128	0.600	0.595	0.605	0.600
128	256	0.600	0.595	0.605	0.600
64	256	0.600	0.595	0.605	0.600
32	256	0.600	0.595	0.605	0.600
16	256	0.600	0.595	0.605	0.600
4	256	0.600	0.595	0.605	0.600
128	512	0.600	0.595	0.605	0.600
64	512	0.600	0.595	0.605	0.600
32	512	0.600	0.595	0.605	0.600

AoH is not sensitive to these hyperparameters. On MMLU, accuracy is almost unchanged across all combinations and sparsity levels, showing clear robustness in the short-context setting.

J AoH-Guided Sparse Attention: H2Share

AoH has the strongest scalability and configurability; therefore, in this section, we introduce an AoH-guided cross-layer shared sparse Attention structure, H2Share. Building on AoH, H2Share exploits both head heterogeneity and adjacent-layer stability through three offline steps: head or KV-group classification, anchor–reuse block assignment, and cross-layer head/group mapping. The online policy is then determined by the layer role and head type. All offline steps use frozen weights only; no training or calibration prompts are required. Figure 11 summarizes the architecture. Importantly, H2Share is an *index-only* sharing method: the only object transferred across layers is a set of integer token positions, never key/value vectors.

J.1 Head Classification

Algorithm 1 classifies each head as retrieval or streaming using the effective-rank of its kernel attention matrix. For head h in layer l, we compute the d_{head}-dimensional proxy $C_h^{(l)} = (W_Q^{(l)h} W_Q^{(l)h\top})(W_K^{(l)h} W_K^{(l)h\top})$, whose eigenvalues equal the squared nonzero singular values of $M_h^{(l)}$, and evaluate

$$\text{eff_rank}^{(l)}(h) = \exp\left(-\sum_{k=1}^{r_h} \hat{\sigma}_k \log \hat{\sigma}_k\right),$$
$$\text{where } \hat{\sigma}_k = \frac{\sqrt{\lambda_k(C_h^{(l)})}}{\sum_{j=1}^{r_h} \sqrt{\lambda_j(C_h^{(l)})}}, \quad k = 1, \ldots, r_h. \tag{10}$$

For each layer, heads are sorted by effective-rank in ascending order and the lowest-k units are designated retrieval heads. The budget k controls the retrieval/streaming ratio. For GQA models, the same decision is lifted to KV-group granularity to preserve grouped-cache efficiency; details are in Section K.

Table 11: Prefill/decode latency and KV memory of AoH on LLaMA-3.1-8B across sparsity levels and context lengths. Speedups and memory reduction are relative to Full Attention.

Ctx Len	Prefill Latency (s, total)			Prefill Mem (GB)			Decode Latency (ms/tok)			Decode Mem (GB)		
	Full	AoH	Spd↑	Full	AoH	Mem↑	Full	AoH	Spd↑	Full	AoH	Mem↑
LLaMA-3.1-8B (sparsity=25%)												
4K	1.382	0.870	1.59×	0.537	0.415	1.29×	22.00	11.55	1.90×	0.540	0.420	1.29×
8K	2.632	2.165	1.22×	1.074	0.818	1.31×	23.65	14.45	1.64×	1.076	0.820	1.31×
16K	6.307	5.703	1.11×	2.148	1.623	1.32×	31.55	20.87	1.51×	2.150	1.625	1.32×
32K	19.592	18.097	1.08×	4.295	3.234	1.33×	53.20	35.24	1.51×	4.298	3.236	1.33×
64K	68.447	62.124	1.10×	8.590	6.455	1.33×	99.53	62.26	1.60×	8.593	6.457	1.33×
128K	259.748	219.358	1.18×	17.180	12.898	1.33×	199.55	122.47	1.63×	17.183	12.900	1.33×
256K	1073.530	882.520	**1.22**×	34.360	25.780	**1.33**×	838.80	237.20	**3.54**×	34.362	25.784	**1.33**×
LLaMA-3.1-8B (sparsity=50%)												
4K	1.382	0.800	1.73×	0.537	0.294	1.83×	22.00	10.60	2.08×	0.540	0.290	1.83×
8K	2.632	1.861	1.41×	1.074	0.562	1.91×	23.65	12.53	1.89×	1.076	0.563	1.91×
16K	6.307	4.698	1.34×	2.148	1.099	1.95×	31.55	16.91	1.87×	2.150	1.100	1.95×
32K	19.592	13.250	1.48×	4.295	2.173	1.98×	53.20	26.76	1.99×	4.298	2.174	1.98×
64K	68.447	43.587	1.57×	8.590	4.320	1.99×	99.53	45.21	2.20×	8.593	4.321	1.99×
128K	259.748	154.225	1.68×	17.180	8.615	1.99×	199.55	85.49	2.33×	17.183	8.616	1.99×
256K	1073.530	615.020	**1.75**×	34.360	17.200	**2.00**×	838.80	162.27	**5.17**×	34.362	17.206	**2.00**×
LLaMA-3.1-8B (sparsity=75%)												
4K	1.382	0.759	1.82×	0.537	0.172	3.12×	22.00	9.64	2.28×	0.540	0.170	3.13×
8K	2.632	1.611	1.63×	1.074	0.306	3.51×	23.65	10.65	2.22×	1.076	0.307	3.51×
16K	6.307	3.730	1.69×	2.148	0.575	3.74×	31.55	13.00	2.43×	2.150	0.575	3.74×
32K	19.592	9.427	2.08×	4.295	1.112	3.86×	53.20	18.49	2.88×	4.298	1.112	3.86×
64K	68.447	27.184	2.52×	8.590	2.185	3.93×	99.53	28.50	3.49×	8.593	2.186	3.93×
128K	259.748	90.084	2.88×	17.180	4.333	3.97×	199.55	49.90	4.00×	17.183	4.333	3.97×
256K	1073.530	331.840	**3.24**×	34.360	8.630	**3.98**×	838.80	91.73	**9.14**×	34.362	8.628	**3.98**×

J.2 Anchor–Reuse Blocks and Head-Mapped Index Reuse

Anchor–reuse blocks. We partition the L transformer layers into consecutive, non-overlapping blocks of size B. The first layer of each block is the **Anchor Layer**; the remaining $B-1$ layers are **Reuse Layers**. In an Anchor Layer, AoH-identified retrieval heads compute global attention and select top-k important-token indices according to their attention scores. These indices are made available to Reuse Layers in the same block. Streaming heads do not participate in cross-layer reuse and are served by sink tokens plus a recent-window cache.

Index-only reuse. Unlike HySparse (Gao et al., 2026), which shares both token selection and KV cache from a preceding full-attention layer, H2Share transfers only selected token indices. Each Reuse Layer computes its own keys and values, then its retrieval heads attend sparsely to the layer-specific K/V states at the borrowed positions. This design preserves the pretrained layer-specific projection geometry while still reducing attention computation. The shared-KV variant is used only as a negative ablation in Figure 12, where it collapses due to cross-layer projection-space mismatch.

Head mapping. Naively giving every Reuse Retrieval Head the indices selected by the same-indexed Anchor Head assumes that head indices have identical functions across adjacent layers. This assumption is unreliable. We instead map each Reuse Retrieval Head $h' \in \mathcal{R}^{(l')}$ to the functionally closest Retrieval Head $h \in \mathcal{R}^{(l)}$ in the Anchor Layer using the normalized Frobenius inner product between their kernel attention matrices: By the cyclic invariance of the trace, all quantities in Eq. (11) are computed efficiently in d_{head} space without ever forming the $d_{\text{model}} \times d_{\text{model}}$ matrix M_h; the detailed proof is in Appendix L. Each Reuse Retrieval Head uses the important-token indices selected by its best-matched Anchor Retrieval Head.

Inference. Let $\mathcal{R}^{(l)}$ and $\mathcal{S}^{(l)}$ denote the retrieval and streaming heads in layer l. An Anchor Layer computes global attention for retrieval heads and sliding-window attention for streaming heads:

$$o^{(l)} = \Big[\mathrm{Attn}\Big(q_{\mathcal{R}}^{(l)}, K_{\mathcal{R}}^{(l)}, V_{\mathcal{R}}^{(l)}\Big) \;\Big|\; \mathrm{SWA}\Big(q_{\mathcal{S}}^{(l)}, K_{\mathcal{S}}^{(l)}, V_{\mathcal{S}}^{(l)}\Big) \Big] W_O^{(l)}. \tag{12}$$

Table 12: Prefill/decode latency and KV memory of AoH on Qwen2.5-7B across sparsity levels and context lengths. Speedups and memory reduction are relative to Full Attention. AoH decode latency uses eager two-path attention with whole-step CUDA-graph capture.

Ctx Len	Prefill Latency (s, total)			Prefill Mem (GB)			Decode Latency (ms/tok)			Decode Mem (GB)		
	Full	AoH	Spd↑	Full	AoH	Mem↓	Full	AoH	Spd↑	Full	AoH	Mem↓
Qwen2.5-7B (sparsity=25%)												
4K	1.273	0.947	**1.35**×	0.235	0.182	1.29×	19.92	9.91	2.01×	0.236	0.183	1.29×
8K	2.115	1.975	1.07×	0.470	0.358	1.31×	21.52	12.04	1.79×	0.471	0.359	1.31×
16K	5.056	5.129	0.99×	0.940	0.710	1.32×	26.62	17.09	1.56×	0.941	0.711	1.32×
32K	16.031	14.673	1.09×	1.879	1.415	1.33×	43.83	27.81	1.58×	1.880	1.416	1.33×
64K	50.914	46.750	1.09×	3.758	2.824	1.33×	78.83	48.38	1.63×	3.759	2.825	1.33×
128K	191.469	169.305	1.13×	7.516	5.643	1.33×	160.44	94.63	1.70×	7.517	5.644	1.33×
256K	776.550	642.693	1.21×	15.032	11.280	**1.33**×	312.31	181.92	**1.72**×	15.034	11.281	**1.33**×
Qwen2.5-7B (sparsity=50%)												
4K	1.273	0.765	1.66×	0.235	0.128	1.83×	19.92	9.29	2.14×	0.236	0.129	1.83×
8K	2.115	1.737	1.22×	0.470	0.246	1.91×	21.52	10.86	1.98×	0.471	0.246	1.91×
16K	5.056	5.109	0.99×	0.940	0.481	1.95×	26.62	14.10	1.89×	0.941	0.481	1.95×
32K	16.031	11.362	1.41×	1.879	0.951	1.98×	43.83	21.75	2.02×	1.880	0.951	1.98×
64K	50.914	34.670	1.47×	3.758	1.890	1.99×	78.83	35.94	2.19×	3.759	1.891	1.99×
128K	191.469	134.345	1.43×	7.516	3.769	1.99×	160.44	66.98	2.40×	7.517	3.770	1.99×
256K	776.550	456.776	**1.70**×	15.032	7.527	**2.00**×	312.31	127.32	**2.45**×	15.034	7.528	**2.00**×
Qwen2.5-7B (sparsity=75%)												
4K	1.273	0.768	1.66×	0.235	0.075	3.12×	19.92	7.61	2.62×	0.236	0.076	3.13×
8K	2.115	1.543	1.37×	0.470	0.134	3.51×	21.52	7.68	2.80×	0.471	0.134	3.51×
16K	5.056	3.440	1.47×	0.940	0.251	3.74×	26.62	8.32	3.20×	0.941	0.252	3.74×
32K	16.031	8.161	1.96×	1.879	0.486	3.86×	43.83	9.38	4.67×	1.880	0.487	3.86×
64K	50.914	21.675	2.35×	3.758	0.956	3.93×	78.83	10.80	7.30×	3.759	0.956	3.93×
128K	191.469	66.267	2.89×	7.516	1.896	3.96×	160.44	13.63	11.77×	7.517	1.896	3.97×
256K	776.550	222.189	**3.50**×	15.032	3.775	**3.98**×	312.31	21.53	**14.50**×	15.034	3.775	**3.98**×

It records the selected indices $\mathcal{I}_h^{(l)}$ for each Anchor Retrieval Head. For a Reuse Layer l', each Retrieval Head h' obtains indices from its mapped Anchor Head $m(h')$ and attends only to its own layer-specific K/V states at those positions:

$$o^{(l')} = \Big[\mathrm{Attn}\Big(q_{\mathcal{R}}^{(l')}, K_{\mathcal{R}}^{(l')}[\mathcal{I}_{m(\cdot)}^{(l)}], V_{\mathcal{R}}^{(l')}[\mathcal{I}_{m(\cdot)}^{(l)}]\Big) \Big| \mathrm{SWA}\Big(q_{\mathcal{S}}^{(l')}, K_{\mathcal{S}}^{(l')}, V_{\mathcal{S}}^{(l')}\Big) \Big] W_O^{(l')}. \tag{13}$$

Thus, H2Share reuses cross-layer information only at the level of discrete important-token positions; all key/value vectors remain layer-specific.

J.3 Inference Phase

Prefill Stage: $W_Q^{(l)}, W_K^{(l)}, W_V^{(l)}$ are reordered offline along the head dimension so that Retrieval and Streaming Heads form contiguous slices, making all head-type splits simple tensor indexing with no gather overhead. Retrieval Heads process the full prompt via standard FlashAttention-2 (Dao, 2023), maintaining layer-specific $O(T)$ KV states. Streaming Heads adopt chunked prefilling: the prompt is divided into fixed-size chunks, and after each chunk, the KV cache is immediately trimmed to retain only sink tokens and the most recent window, bounding memory to $O(s_{\text{sink}} + s_{\text{recent}})$ regardless of text length.

Decode Stage: Each layer applies its assigned strategy based on its role (Anchor or Reuse) and head type (Retrieval or Streaming). An Anchor Layer l uses layer-specific full K/V states for Retrieval Heads and a fixed-size window cache for Streaming Heads. Each head type is computed independently; their outputs are concatenated along the head dimension and projected through a shared output matrix:

$$o^{(l)} = \Big[\underbrace{\mathrm{Attn}\Big(q_{\mathcal{R}}^{(l)}, \mathrm{KV}_{\mathcal{R}}^{(l)}\Big)}_{\text{Full Attention}}, \underbrace{\mathrm{Attn}\Big(q_{\mathcal{S}}^{(l)}, \mathrm{KV}_{\mathcal{S}}^{(l)}\Big)}_{\text{SlidingWindowAttention}} \Big] W_O^{(l)} \tag{14}$$

Table 13: Prefill/decode latency and KV memory of AoH on Qwen3-8B across sparsity levels and context lengths. Speedups and memory reduction are relative to Full Attention.

Ctx Len	Prefill Latency (s, total)			Prefill Mem (GB)			Decode Latency (ms/tok)			Decode Mem (GB)		
	Full	AoH	Spd↑	Full	AoH	Mem↓	Full	AoH	Spd↑	Full	AoH	Mem↓
Qwen3-8B (sparsity=25%)												
4K	1.305	1.075	**1.21**×	0.604	0.467	1.29×	28.82	14.05	2.05×	0.607	0.469	1.29×
8K	2.511	2.492	1.01×	1.208	0.920	1.31×	28.90	17.86	1.62×	1.211	0.922	1.31×
16K	7.064	6.694	1.06×	2.416	1.826	1.32×	38.42	25.46	1.51×	2.419	1.828	1.32×
32K	22.458	20.701	1.09×	4.832	3.638	1.33×	64.40	40.65	1.58×	4.835	3.640	1.33×
64K	76.034	69.077	1.10×	9.664	7.262	1.33×	116.43	71.29	1.63×	9.667	7.264	1.33×
128K	292.425	257.593	1.14×	19.327	14.510	1.33×	235.80	144.21	1.64×	19.330	14.512	1.33×
256K	1155.898	1016.460	1.14×	38.655	29.005	**1.33**×	459.90	277.59	**1.66**×	38.658	29.007	**1.33**×
Qwen3-8B (sparsity=50%)												
4K	1.305	1.473	0.89×	0.604	0.330	1.83×	28.82	12.99	2.22×	0.607	0.332	1.83×
8K	2.511	2.313	1.09×	1.208	0.632	1.91×	28.90	15.62	1.85×	1.211	0.634	1.91×
16K	7.064	5.581	1.27×	2.416	1.236	1.95×	38.42	20.78	1.85×	2.419	1.238	1.95×
32K	22.458	15.762	1.42×	4.832	2.444	1.98×	64.40	31.11	2.07×	4.835	2.446	1.98×
64K	76.034	51.437	1.48×	9.664	4.860	1.99×	116.43	51.99	2.24×	9.667	4.862	1.99×
128K	292.425	181.280	1.61×	19.327	9.692	1.99×	235.80	101.45	2.32×	19.330	9.694	1.99×
256K	1155.898	691.531	**1.67**×	38.655	19.356	**2.00**×	459.90	192.29	**2.39**×	38.658	19.357	**2.00**×
Qwen3-8B (sparsity=75%)												
4K	1.305	0.873	1.50×	0.604	0.193	3.12×	28.82	11.89	2.42×	0.607	0.194	3.13×
8K	2.511	1.900	1.32×	1.208	0.345	3.51×	28.90	13.34	2.17×	1.211	0.345	3.51×
16K	7.064	4.408	1.60×	2.416	0.646	3.74×	38.42	16.12	2.38×	2.419	0.647	3.74×
32K	22.458	11.478	1.96×	4.832	1.250	3.86×	64.40	21.82	2.95×	4.835	1.251	3.86×
64K	76.034	32.813	2.32×	9.664	2.458	3.93×	116.43	33.13	3.51×	9.667	2.459	3.93×
128K	292.425	105.451	2.77×	19.327	4.874	3.96×	235.80	59.71	3.95×	19.330	4.875	3.97×
256K	1155.898	384.765	**3.00**×	38.655	9.706	**3.98**×	459.90	108.87	**4.22**×	38.658	9.707	**3.98**×

where $\mathrm{Concat}(\cdot)$ denotes concatenation along the head axis, followed by reshaping to d_{model}.

After processing the full context, the Anchor Layer records the top-k token indices $\mathcal{I}_h^{(l)}$ per Retrieval Head h for reuse within the block. A Reuse Layer l' borrows only these integer indices via the head mapping $\mathcal{M}^{(l\to l')}$ and computes its own K/V states independently: each Retrieval Head computes sparse attention over the mapped k positions, while Streaming Heads compute sliding-window attention independently.

$$o^{(l')} = \Bigg[\underbrace{\mathrm{Attn}\Big(q_{\mathcal{R}}^{(l')},\ \mathrm{KV}_{\mathcal{R}}^{(l')}\Big[\mathcal{I}^{(l)}_{\mathcal{M}^{(l\to l')}(\cdot)}\Big]\Big)}_{\text{Sparse Attention (borrowed indices, own KV)}},\ \underbrace{\mathrm{Attn}\Big(q_{\mathcal{S}}^{(l')},\ \mathrm{KV}_{\mathcal{S}}^{(l')}\Big)}_{\text{SWA}}\Bigg] W_O^{(l')} \tag{15}$$

where $\mathrm{KV}_{\mathcal{R}}^{(l')}\Big[\mathcal{I}^{(l)}_{\mathcal{M}^{(l\to l')}(\cdot)}\Big]$ denotes the Reuse Layer's own K/V states indexed by the top-k positions borrowed from the mapped Anchor Head. The borrowed object is only the discrete index set; no Anchor-Layer K/V vector is transferred. For Retrieval Heads, the attention scan is restricted to $O(k)$ positions; for Streaming Heads, a fixed-size sliding-window KV cache is dynamically maintained.

K GQA Extension

Our proposed solutions are all based on Multi-head Attention (MHA) (Cordonnier et al., 2020), but modern LLMs (Xiaomi MiMo Team, 2026; Grattafiori et al., 2024; Yang et al., 2024, 2025) increasingly adopt Grouped Query Attention (GQA) (Ainslie et al., 2023), where H Query Heads are grouped into G KV Groups, with $\frac{H}{G}$ Query Heads sharing a single KV Head within each group. Therefore, the retention decision must also be made at the KV-Group level. If a classification strategy targeting individual query heads were adopted, partitioning the KV cache within a group would lose the storage efficiency of GQA.

Group-level Classification: For each KV Group g, every constituent Query Head h independently evaluates its effective-rank from its per-head Kernel Attention Matrix M_h, computed using the shared

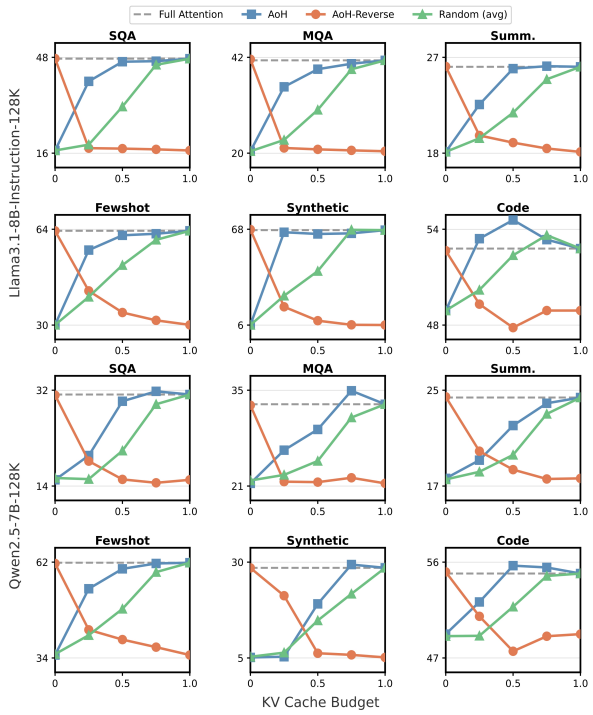

Figure 10: Ablation of AoH head classification under different KV-cache budgets on LongBench. We compare AoH with Full Attention, Random$_{\text{avg}}$, and Reverse.

group key projection W_K^g and its own query projection W_Q^h. We aggregate these per-head ranks into a single group-level score by taking the *mean* across all member heads in the group. Within each layer, KV groups are then ranked by this score, and the $k = \lceil (1-s), G \rceil$ groups with the lowest effective-rank are labelled Retrieval Groups (where s is the target sparsity and G the number of KV groups), while the remainder are Streaming Groups. All Query Heads within a group inherit the same label, so the classification maps cleanly back to the Q-Head level without ambiguity.

Group-level Head Mapping: The cross-layer semantic similarity matrix reduced from $H \times H$ (MHA) to $G \times G$ (GQA), with one entry per KV Group pair. For each group pair, the pairwise Q-Head similarities are aggregated (mean) to produce a single group-to-group score. The mapping then assigns each Reuse Retrieval Group to its most similar Anchor Retrieval Group, and all Query Heads within that group borrow the same set of top-k token indices.

Weight Reordering: Offline permutation operates at two granularities simultaneously: $W_K^{(l)}$ and $W_V^{(l)}$ are reordered along the KV-Group axis according to the group classification permutation, while $W_Q^{(l)}$ is reordered along the Q-Head axis by expanding each group index into its $\frac{H}{G}$ constituent Q-Head indices. After reordering, Retrieval and Streaming Groups remain contiguous in memory, preserving full compatibility with FlashAttention 's (Dao, 2023) native GQA kernel support with no additional gather overhead.

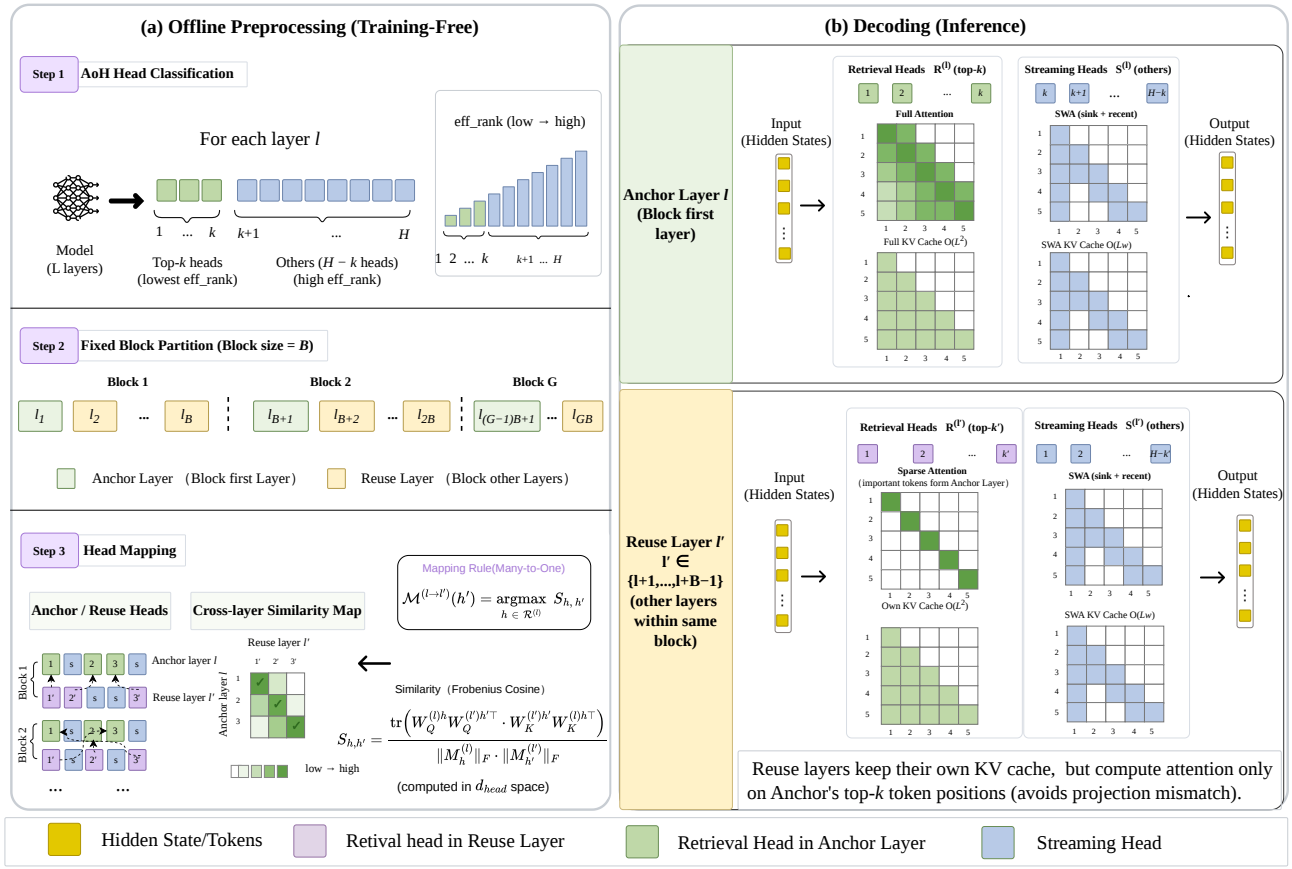

Figure 11: Overview of H2Share. **Left**: Offline preprocessing classifies heads with AoH, partitions layers into anchor–reuse blocks, and builds cross-layer head mappings. **Right**: During decoding, anchor retrieval heads compute full attention and select important-token indices; reuse retrieval heads borrow only these indices and attend to their own layer-specific keys and values at the mapped positions; streaming heads use sliding-window attention.

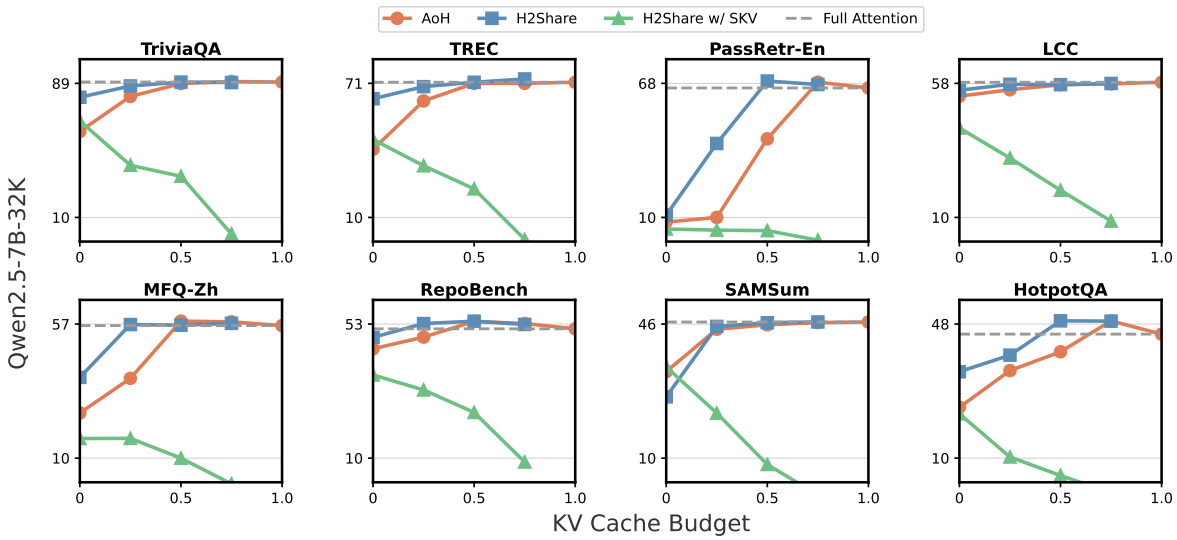

Figure 12: Ablation Experiments on LongBench. H2Share offers a better KV budget-accuracy trade-off.

$$\mathcal{M}_{h,h'}^{(l\to l')} = \frac{\left\langle M_h^{(l)},\, M_{h'}^{(l')} \right\rangle_F}{\left\| M_h^{(l)} \right\|_F \cdot \left\| M_{h'}^{(l')} \right\|_F}, \quad h \in \{1,\dots,H\}^{(l)},\ h' \in \{1,\dots,H\}^{(l')}, \quad \mathcal{M}^{(l\to l')} \in \mathbb{R}^{H\times H}$$

$$\mathcal{M}_{h,h'}^{(l\to l')} = \frac{\operatorname{tr}\Big(W_Q^{(l)h}\, W_Q^{(l')h'\top} \cdot W_K^{(l')h'}\, W_K^{(l)h\top}\Big)}{\sqrt{\operatorname{tr}\Big(W_Q^{(l)h}\, W_Q^{(l)h\top} \cdot W_K^{(l)h}\, W_K^{(l)h\top}\Big)} \cdot \sqrt{\operatorname{tr}\Big(W_Q^{(l')h'}\, W_Q^{(l')h'\top} \cdot W_K^{(l')h'}\, W_K^{(l')h'\top}\Big)}}. \tag{11}$$

L Proof: Trace Reduction to d_{head} Space

We prove that the Frobenius inner product $\langle M_h^{(l)}, M_{h'}^{(l')} \rangle_F$ and the Frobenius norms $\|M_h^{(l)}\|_F$ in Eq. (11) can be computed entirely in d_{head} space, without ever constructing the $d_{\text{model}} \times d_{\text{model}}$ kernel attention matrix M_h.

Setup

Recall the kernel attention matrix for head h at layer l:

$$M_h^{(l)} = W_K^{(l)h\top} W_Q^{(l)h} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}, \qquad W_Q^{(l)h},\, W_K^{(l)h} \in \mathbb{R}^{d_{\text{head}} \times d_{\text{model}}}. \tag{16}$$

Since $\operatorname{rank}(M_h^{(l)}) \le d_{\text{head}} \ll d_{\text{model}}$, direct construction of $M_h^{(l)}$ requires $O(d_{\text{model}}^2 \cdot d_{\text{head}})$ operations. We show that both quantities reduce to traces of $d_{\text{head}} \times d_{\text{head}}$ matrices.

Frobenius Inner Product

Proposition 1. $\langle M_h^{(l)}, M_{h'}^{(l')} \rangle_F = \operatorname{tr}\Big(W_Q^{(l)h}\, W_Q^{(l')h'\top} \cdot W_K^{(l')h'}\, W_K^{(l)h\top} \Big)$.

Proof. By definition of the Frobenius inner product and the trace identity $\langle A, B\rangle_F = \mathrm{tr}(A^\top B)$:

$$\langle M_h^{(l)},\, M_{h'}^{(l')}\rangle_F = \mathrm{tr}\Big(M_h^{(l)\top} M_{h'}^{(l')}\Big)$$
$$= \mathrm{tr}\left(\underbrace{W_Q^{(l)h\top}}_{d_{\text{model}}\times d_{\text{head}}}\ \underbrace{W_K^{(l)h}}_{d_{\text{head}}\times d_{\text{model}}}\ \underbrace{W_K^{(l')h'\top}}_{d_{\text{model}}\times d_{\text{head}}}\ \underbrace{W_Q^{(l')h'}}_{d_{\text{head}}\times d_{\text{model}}}\right). \tag{17}$$

The trace in (17) operates on a $d_{\text{model}} \times d_{\text{model}}$ matrix, which is expensive. We apply the *cyclic invariance* of the trace, $\mathrm{tr}(ABCD) = \mathrm{tr}(DABC)$, with

$$A = W_Q^{(l)h\top}, \quad B = W_K^{(l)h}, \quad C = W_K^{(l')h'\top}, \quad D = W_Q^{(l')h'}.$$

Cycling D to the front:

$$\mathrm{tr}(ABCD) = \mathrm{tr}(DABC) = \mathrm{tr}\left(\underbrace{W_Q^{(l')h'} W_Q^{(l)h\top}}_{d_{\text{head}}\times d_{\text{head}}} \cdot \underbrace{W_K^{(l)h} W_K^{(l')h'\top}}_{d_{\text{head}}\times d_{\text{head}}}\right). \tag{18}$$

Finally, using $\mathrm{tr}(A) = \mathrm{tr}(A^\top)$ on (18):

$$= \mathrm{tr}\Big(W_Q^{(l)h} W_Q^{(l')h'\top} \cdot W_K^{(l')h'} W_K^{(l)h\top}\Big),$$

which is the trace of a $d_{\text{head}} \times d_{\text{head}}$ matrix product. This completes the proof. □

Frobenius Norm

Corollary 1. $\|M_h^{(l)}\|_F^2 = \mathrm{tr}\Big(W_Q^{(l)h}\, W_Q^{(l)h\top} \cdot W_K^{(l)h}\, W_K^{(l)h\top}\Big)$.

Proof. Setting $l' = l$ and $h' = h$ in Proposition 1 gives

$$\|M_h^{(l)}\|_F^2 = \langle M_h^{(l)},\, M_h^{(l)}\rangle_F = \mathrm{tr}\Big(W_Q^{(l)h} W_Q^{(l)h\top} \cdot W_K^{(l)h} W_K^{(l)h\top}\Big).$$ □

Complexity

Both quantities now require computing two $d_{\text{head}} \times d_{\text{head}}$ matrices and taking their trace:

- $W_Q^{(l)h} W_Q^{(l')h'\top} \in \mathbb{R}^{d_{\text{head}}\times d_{\text{head}}}$: cost $O(d_{\text{head}}^2 \cdot d_{\text{model}})$.
- $W_K^{(l')h'} W_K^{(l)h\top} \in \mathbb{R}^{d_{\text{head}}\times d_{\text{head}}}$: cost $O(d_{\text{head}}^2 \cdot d_{\text{model}})$.
- Trace of $d_{\text{head}} \times d_{\text{head}}$ product: cost $O(d_{\text{head}}^3)$.

This is a reduction from $O(d_{\text{model}}^3)$ (direct construction + inner product of M_h) to $O(d_{\text{head}}^2 \cdot d_{\text{model}})$. For Qwen3-8B ($d_{\text{head}} = 128$, $d_{\text{model}} = 4096$), this yields a factor of $\approx d_{\text{model}}/d_{\text{head}} = 32\times$ speedup.