

KV Admission: Learning What to Write for Efficient Long-Context LLM Inference

Yen-Chieh Huang^{1,2}, Pi-Cheng Hsiu^{2,†}, Rui Fang¹, Ming-Syan Chen^{1,†}

¹Department of Electrical Engineering, National Taiwan University

²Research Center for Information Technology Innovation, Academia Sinica

[†]Correspondence: pchsiu@citi.sinica.edu.tw, mschen@ntu.edu.tw

Abstract

Long-context LLM inference is bottlenecked by the quadratic attention complexity and linear Key-Value (KV) cache growth. Prior approaches mitigate this via post-hoc selection or eviction but overlook the root inefficiency: *indiscriminate token admission*. In this paper, we formalize KV management as a causal system of three primitives: KV Admission, Selection, and Eviction. We instantiate KV Admission via Write-Gated KV (WG-KV), a lightweight mechanism that learns to predict token utility *before* cache entry. By filtering out redundant states early to maintain a compact global cache alongside a sliding local cache, WG-KV significantly reduces memory usage and accelerates both prefill and decode phases. Our results demonstrate that *learning what to write* is a principled and practical recipe for efficient long-context inference. Code is available at <https://github.com/EMCLab-Sinica/WG-KV>.

1 Introduction

The capability of Large Language Models (LLMs) to process extensive contexts has revolutionized applications ranging from long-document summarization (Bai et al., 2025), repository-level code analysis (Hu et al., 2025), to long-term agentic planning (Erdogan et al., 2025). However, this capability comes at a steep serving cost. Long-context LLM inference is bottlenecked by both the quadratic attention complexity and the linear growth of the Key-Value (KV) cache. In long-context scenarios, the sheer volume of tokens not only incurs massive computational overhead but also exhausts limited GPU memory. Consequently, the latency and resource consumption increase significantly with the context length, rendering long-context LLM inference prohibitively expensive.

To mitigate this bottleneck, prior research has focused on two management strategies: **KV Selection** and **KV Eviction**. Selection-based methods,

such as Quest (Tang et al., 2024) and NSA (Yuan et al., 2025), selectively attend to relevant KVs at runtime yet retain the complete state in memory. Eviction-based methods, such as SnapKV (Li et al., 2024) and R-KV (Cai et al., 2025), retrospectively prune the cache by evicting unneeded tokens after they have accumulated in memory. While effective, these paradigms share a common, fundamental inefficiency: *every generated token is indiscriminately admitted to the growing cache state*, where the system blindly consumes memory resources to accumulate redundant KV states, only to later skip or evict them.

In this paper, we argue that an efficient long-context system requires a third, missing primitive: **KV Admission**. Instead of retrieving information at *read-time* (Selection) or managing memory *post-write* (Eviction), Admission operates *pre-write*, acting as a gatekeeper that predicts the future utility of a token before admitting it to the KV cache.

To realize this, we introduce **Write-Gated KV (WG-KV)**, a novel, learnable KV Admission mechanism. Unlike heuristic approaches, WG-KV employs a lightweight, end-to-end differentiable predictor that evaluates token utility *before* cache entry. During prefilling, this predictor constructs a vertical-slash attention mask, ensuring high-utility tokens are visible to the entire sequence while others are accessible only locally. Simultaneously, it routes generated KV pairs to distinct memory regions: tokens predicted as high-utility are admitted to a persistent global cache, while others are confined to a transient local cache. This selective strategy seamlessly extends to the decoding phase, where we maintain this dual-cache structure to filter redundant states upon KV generation.

Our approach yields substantial efficiency gains through a **hardware-aware implementation** that effectively translates theoretical sparsity into practical wall-clock speedups. We evaluate WG-KV on Llama 3 (Grattafiori et al., 2024) and Qwen3

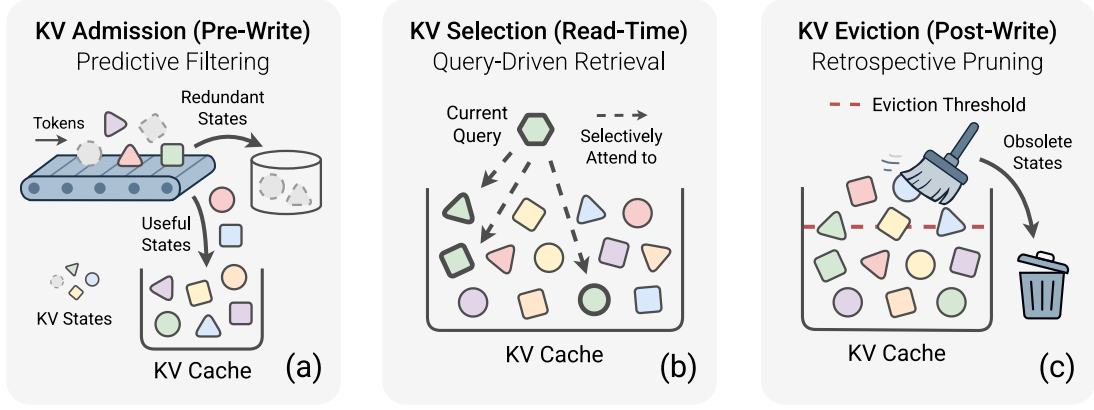


Figure 1: A unified framework for KV management. (a) *KV Admission (Ours)* predictively filters redundant tokens pre-write. (b) *KV Selection* selectively attends to relevant tokens at read-time. (c) *KV Eviction* retrospectively prunes obsolete tokens post-write.

(Yang et al., 2025a) models, demonstrating that WG-KV reduces memory usage by 36–69% and delivers 2.56–4.17x prefill and 1.59–2.63x decode speedups, while maintaining near-lossless task accuracy across diverse long-context benchmarks. By *learning what to write*, we enable LLMs to sustain longer contexts with significantly fewer resources.

2 Background and Motivation

2.1 Long-context attention bottleneck

The core of the Transformer architecture (Vaswani et al., 2017) in LLMs is the attention mechanism. Given a sequence of input tokens, the attention output for a query vector $q_t \in \mathbb{R}^d$ at step t is computed by attending to past key-value pairs $\{(k_i, v_i)\}_{i=1}^t$. Specifically, let $K_{\leq t}, V_{\leq t} \in \mathbb{R}^{t \times d}$ denote the matrices formed by stacking the key and value vectors up to step t . The attention operation is defined as:

$$\text{Attn}(q_t, K_{\leq t}, V_{\leq t}) = \text{Softmax}\left(q_t K_{\leq t}^\top / \sqrt{d}\right) V_{\leq t}$$

In standard autoregressive generation, recomputing $K_{\leq t}$ and $V_{\leq t}$ from all previous tokens at every step t would be computationally prohibitive. To address this, inference engines employ a **KV cache**, storing the key and value vectors for previous tokens. At each step t , the system only computes the new pair (k_t, v_t) , appends it to the cache, and performs attention over the accumulated state. However, as the sequence length increases, the attention mechanism presents distinct bottlenecks during the two phases of inference: *prefill* and *decode*.

During *prefilling*, the model processes the sequence of length N in parallel, with the quadratic complexity of attention $O(N^2)$ quickly dominates execution time (Figure 2a). During *decoding*, the

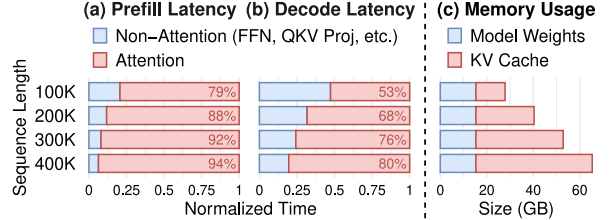


Figure 2: Attention bottleneck in long-context LLM inference. Measured on Llama-3.1-8B on an H200 GPU.

model generates tokens autoregressively, making the process memory-bound as each step requires loading the KV cache from memory. As the sequence grows, the massive data transfer saturates memory bandwidth, driving up latency (Figure 2b), while the expanding cache occupies significant GPU memory (Figure 2c).

2.2 A causal framework for KV management

Formally, we can model the KV cache C_t for a certain attention head at step t as a set of KV pairs stored in memory. In standard LLM inference, the cache management policy is trivial: it is an append-only operation where every new token’s state (k_t, v_t) is indiscriminately accumulated throughout the entire inference process:

$$C_t^{\text{std}} = C_{t-1}^{\text{std}} \cup \{(k_t, v_t)\}$$

This policy leads to linear KV cache growth $|C_t^{\text{std}}| = t$ and becomes the root cause of attention bottleneck. To address this, prior works have introduced constraints on the attention operation or C_t itself. We unify these approaches under a causal framework based on the *timing* and *decision scope* of the operation, as illustrated in Figure 1 and detailed below.

KV Selection (read-time). Selection methods, such as Quest (Tang et al., 2024) and NSA (Yuan et al., 2025), optimize the *read* path. They *approximate* the attention operation by selecting a subset of keys $\mathcal{S}_t \subset \mathcal{C}_t$ relevant to the current query q_t :

$$\text{Attn}(q_t, \mathcal{C}_t) \approx \text{Attn}(q_t, \mathcal{S}_t), \text{ where } \mathcal{S}_t = \text{Select}(q_t, \mathcal{C}_t)$$

While Selection reduces the number of attended tokens per step, the underlying cache $\mathcal{C}_t$ remains fully populated to support future queries, leaving the linear KV cache growth unresolved.

KV Eviction (post-write). Eviction methods, such as SnapKV (Li et al., 2024) and R-KV (Cai et al., 2025), impose a constraint B on the cache size. These methods operate *retrospectively*: they first admit the new token into the cache, and then prune the set based on *past statistics*:

$$\mathcal{C}_t^{\text{new}} = \text{Evict}(\mathcal{C}_t^{\text{old}}) \quad \text{s.t.} \quad |\mathcal{C}_t^{\text{new}}| \leq B$$

Eviction effectively bounds KV memory usage. However, it suffers from a “write-then-throw” inefficiency: the system temporarily admits noise into the KV cache, forcing subsequent queries to waste memory bandwidth and computation attending to these tokens until they are eventually identified as redundant. Furthermore, eviction can be risky; information discarded is permanently lost, potentially degrading generation quality if the evicted tokens are needed for later context (Lee et al., 2024b).

KV Admission (pre-write). We identify *Admission* as the missing third primitive. Unlike Eviction, which reacts to cache overflow, Admission acts as a gatekeeper. It evaluates the *future utility* of a new state (k_t, v_t) *before* it is committed to $\mathcal{C}_t$:

$$\mathcal{C}_t = \text{Admit}(\mathcal{C}_{t-1}, k_t, v_t)$$

Ideally, an admission policy prevents the persistence of noise tokens, ensuring $\mathcal{C}_t$ holds only critical information. This avoids the cost of accumulating redundant states, reducing KV growth and alleviating subsequent attention bottlenecks.

A unified view. As illustrated in Figure 1, these primitives are not mutually exclusive; rather, they operate at different stages of the token lifecycle. An ideal inference system can employ Admission to *filter* the input stream, Selection to *focus* the attention computation, and Eviction to *prune* obsolete history. Together, they form a comprehensive framework that addresses the inefficiencies of long-context LLM inference across the entire token lifecycle. We provide a more comprehensive survey

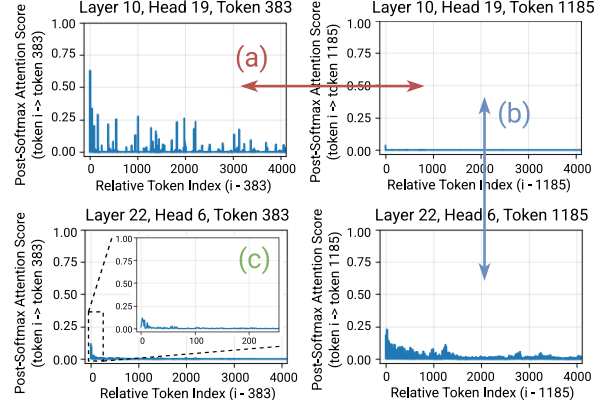


Figure 3: Heterogeneity of token utility.

of existing literature in Appendix A.

2.3 Sparsity and locality in attention patterns

To design an effective KV Admission policy, we must first understand the characteristics of attention patterns. We employ Llama-3.1-8B (Grattafiori et al., 2024) to perform a long-context code summarization task from The Stack dataset (Kocetkov et al., 2023) to observe *which* tokens are attended to and *when*. We identify three key properties:

Skewed utility distribution. Attention scores are not uniformly distributed; they exhibit pronounced sparsity where a small subset of tokens attracts the majority of attention. As shown in Figure 3a, specific tokens (e.g., Token 383 in Layer 10, Head 19) act as high-utility tokens, consistently receiving high attention from subsequent queries. In contrast, other tokens (e.g., Token 1185 in the same head) are virtually ignored. This suggests that a large portion of the KV cache stores “noise” that contributes negligible value to future representations.

Head-specific relevance. The utility of a token is not intrinsic to the token itself but is dependent on the attention head. A token that is critical for one head may be irrelevant for another. For instance, comparing the two heads in Figure 3b, we observe that while Token 1185 serves as a primary attention target for Layer 22, Head 6, it is almost completely ignored by Layer 10, Head 19. Conversely, Token 383, which was ignored by the former, becomes highly salient for the latter. This implies that a unified admission policy (e.g., dropping a token from all heads simultaneously) is suboptimal. An effective admission policy must be fine-grained, making independent decisions for each head.

The necessity of local context. While long-range attention is sparse, local attention exhibits strong recency bias. As illustrated in Figure 3c, Token 383 in Layer 22, Head 6 exemplifies a phenomenon of “transient utility.” Although this token eventually receives near-zero attention from distant queries, it attracts significant attention from its immediate successors. This creates a dilemma for any admission policy: a token cannot be immediately dismissed as universally irrelevant, as it exhibits strong local saliency. A hybrid mechanism is therefore needed to preserve recent tokens briefly for local dense attention before switching to utility-based admission for long-range context.

2.4 Systems challenges with ragged KV states

Implementing a fine-grained admission policy introduces non-trivial systems challenges. In standard dense attention, the KV cache for a sequence is represented as a contiguous tensor. However, independent admission decisions by each head, as motivated in Section 2.3, result in a cache that is “ragged” across two dimensions: (1) *Across heads*: Different heads prioritize different tokens, leading to uneven cache lengths within the same layer. (2) *Across layers*: The level of semantic abstraction varies by depth, causing sparsity rates to diverge between shallow and deep layers.

A naive implementation that simply concentrates admitted tokens leads to severe memory fragmentation, as shown in Figure 4. Pre-allocating maximum-length buffers negates the memory savings, while dynamic reallocation incurs prohibitive runtime overhead. To translate theoretical sparsity into practical wall-clock speedup and memory reduction, the system must leverage a decoupled logical-physical memory mapping to manage these head-wise irregular caches efficiently.

3 Method

In this section, we present **Write-Gated KV (WG-KV)**, a learnable KV Admission mechanism designed to address the aforementioned inference inefficiencies. We first introduce the architectural overview of WG-KV (Section 3.1). Then, we

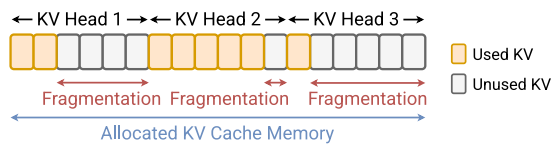


Figure 4: Memory fragmentation with ragged KV states.

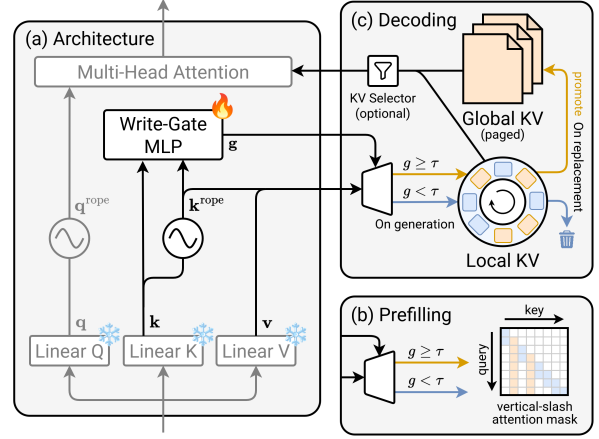


Figure 5: Overview of Write-Gated KV.

derive the differentiable gating mechanism that enables end-to-end learning of token utility (Section 3.2). Finally, we present the training objective for learning the admission policy (Section 3.3).

3.1 Architectural overview

The core philosophy of WG-KV is to filter tokens at the source by *determining which KV states merit long-term retention*. We introduce a lightweight **Write-Gate MLP** (Figure 5a) into the standard attention block to evaluate the gating score $g_t \in [0, 1]$ for each token. Based on this score, the system selectively routes the KV pair to one of two logical memory regions: (1) *Global cache*: A memory region for tokens with high gating score ($g_t \approx 1$). These tokens are retained persistently to support long-term dependencies. (2) *Local cache*: A sliding window of size W_{local} that unconditionally retains recent tokens to support their transient utility (Section 2.3). Upon exiting this window, tokens with low gating score ($g_t \approx 0$) are discarded.

3.2 Differentiable gating mechanism

To learn the gating score g , we derive it using the MLP and subsequently integrate it into the attention formulation. Specifically, for layer l , head h , token t , the input feature $x_{l,h,t}$ is defined by concatenating the pre-RoPE key $k_{l,h,t}$ and the post-RoPE key $k_{l,h,t}^{\text{rope}}$, with both terms normalized via RMSNorm (Su et al., 2024; Zhang and Sennrich, 2019):¹

$$x_{l,h,t} = \left[\text{RMSNorm}(k_{l,h,t}); \text{RMSNorm}(k_{l,h,t}^{\text{rope}}) \right]$$

¹In the context of Grouped-Query Attention (Ainslie et al., 2023), the term “head” specifically refers to the Key-Value (KV) head. We use “head” for simplicity throughout the paper.

The gating score $g_{l,h,t}$ is then formulated as:

$$g_{l,h,t} = \sigma \left(W_{l,h}^2 \cdot \text{GELU}(W_{l,h}^1 \cdot x_{l,h,t} + b_{l,h}^1) + b_{l,h}^2 \right)$$

where $W_{l,h}^1, b_{l,h}^1, W_{l,h}^2, b_{l,h}^2$ are learnable parameters, and σ is the sigmoid function.

Write-Gated Attention. Standard Scaled Dot-Product Attention computes the output for a query $q_i \in \mathbb{R}^d$ against $\{(k_j, v_j)\}_{j \leq i}$ as:²

$$\text{Attn}(q_i, K_{\leq i}, V_{\leq i}) = \sum_{j \leq i} \frac{\exp(q_i \cdot k_j / \sqrt{d})}{\sum_{p \leq i} \exp(q_i \cdot k_p / \sqrt{d})} v_j$$

To simulate the effect of admission during training, we modulate attention scores via mask m_{ij} that provides full visibility within the local window W_{local} while weighting distant tokens by g_j :

$$m_{ij} = \max(\mathbb{1}(i - j < W_{\text{local}}), g_j)$$

where $\mathbb{1}$ is the indicator function. Our proposed Write-Gated Attention is then formulated as:

$$\widetilde{\text{Attn}}(q_i, K_{\leq i}, V_{\leq i}) = \sum_{j \leq i} \frac{\exp(q_i \cdot k_j / \sqrt{d}) \cdot m_{ij}}{\sum_{p \leq i} \exp(q_i \cdot k_p / \sqrt{d}) \cdot m_{ip}} v_j$$

This formulation ensures that if $g_j = 0$, the token j effectively vanishes from the attention of distant queries, mimicking the behavior of not admitting it to the global cache. Conversely, if $g_j = 1$, the token remains visible to all future queries, simulating its admission to the global cache.

Log-space transformation for efficient training. While the multiplicative gating above is theoretically sound, standard attention kernels (Dao et al., 2022) are optimized for $\exp(q \cdot k)$. Introducing a per-token multiplication factor disrupts standard kernel optimizations. To address this, we transform the gating operation into a log-space bias:

$$\exp(q_i \cdot k_j / \sqrt{d}) \cdot m_{ij} = \exp(q_i \cdot k_j / \sqrt{d} + \log(m_{ij}))$$

Write-Gated Attention is thus reformulated as:

$$\widetilde{\text{Attn}}(q_i, K_{\leq i}, V_{\leq i}) = \text{Softmax} \left(q_i K_{\leq i}^\top / \sqrt{d} + B_{\text{gate}} \right) V_{\leq i}$$

where B_{gate} is an attention bias matrix derived from $\log(m_{ij})$. In practice, we compute $\log(m_{ij} + \epsilon)$ with small ϵ for numerical stability. This log-space transformation allows us to leverage generalized attention kernels, such as FlexAttention (Dong et al., 2025), for efficient training on long sequences.

3.3 Training objective

To learn the admission policy, we optimize a joint objective, with the total loss $\mathcal{L}_{\text{total}}$ defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{distill}} + \lambda \mathcal{L}_{\text{sparsity}}$$

²For notational simplicity, we omit the layer (l) and head (h) subscripts in variables like q_t, k_t, v_t and g_t .

$$\mathcal{L}_{\text{sparsity}} = \frac{1}{L \cdot H \cdot T} \sum_{l,h,t} (g_{l,h,t} + g_{l,h,t}(1 - g_{l,h,t}))$$

where $\mathcal{L}_{\text{distill}}$ is the mean squared error of the last-layer hidden states compared to the original full-attention model, and λ serves as a hyperparameter controlling the tradeoff between representation fidelity and KV sparsity. Specifically, the sparsity loss $\mathcal{L}_{\text{sparsity}}$ incorporates two regularization terms for $g_{l,h,t}$: the first term induces sparsity by minimizing global cache admission, while the second penalizes non-binary values, encouraging the gate to converge towards discrete decisions (0 or 1). During inference, we apply a hard threshold τ to binarize g via $\mathbb{1}(g \geq \tau)$ for admission decisions.

4 System Implementation

Implementing WG-KV presents a unique systems challenge: unlike standard Transformers where the KV cache grows uniformly across all heads, our admission policy results in a *ragged cache structure* with highly irregular sequence lengths across heads (as discussed in Section 2.4). To translate theoretical sparsity into wall-clock speedup and memory reduction, we present a **hardware-aware implementation** that leverages paged memory management to handle this irregular cache growth efficiently.

4.1 Dual-cache paged memory management

To address memory fragmentation caused by head-specific admission, we decouple the *logical view* of the cache from its *physical storage*. We partition the logical KV cache of each attention head into the **local cache** and **global cache** (Figure 6a), backed by a unified physical **KV pool** (Figure 6b). To bridge these views, we maintain a per-head **page table** (Figure 6c) that maps the logical pages to non-contiguous physical pages (16 tokens per page). This allows the global cache to grow dynamically without requiring contiguous reallocation.

4.2 Efficient prefilling via sparse attention

In the prefill phase, the system processes the entire input sequence in parallel. The goal is to compute attention and populate the KV cache efficiently.

Vertical-slash sparse attention. We leverage the sparsity pattern dictated by score g to accelerate attention computation. Following the Write-Gated Attention formulation, the attention mask M exhibits a “vertical-slash” pattern (Figure 5b): tokens attend to preceding *globally* high-utility tokens (“vertical”) and their *local* neighborhood (“slash”).

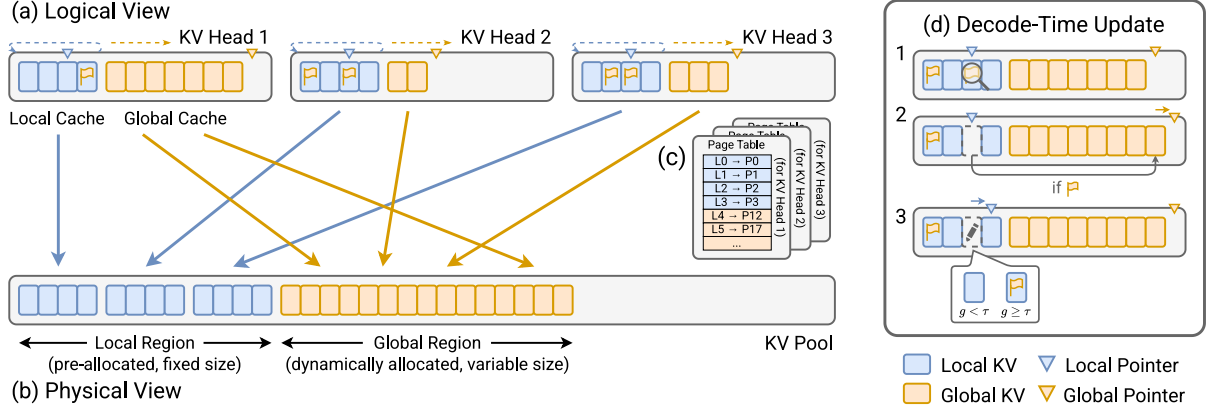


Figure 6: Dual-cache paged memory management. (a–c) To handle ragged cache growth across heads, we decouple the logical view into a fixed-size *local cache* and a growing *global cache*, mapped to a unified physical *KV pool* via per-head *page tables* to prevent memory fragmentation. (d) The cache update logic during decoding. We employ *lazy promotion* where the token exiting the local window is inspected (and possibly promoted) upon replacement.

Formally, for query i , key j , M_{ij} is defined as:

$$M_{ij} = (\mathbb{1}(i - j < W_{\text{local}}) \vee \mathbb{1}(g_j \geq \tau)) \wedge \mathbb{1}(i \geq j)$$

We utilize the sparse FlashAttention kernel from MInference (Jiang et al., 2024) to compute attention efficiently. This avoids the cost of dense $O(N^2)$ attention by skipping masked-out regions.

Initial cache population. Concurrently with attention computation, we write KV states to memory. Tokens within the final W_{local} window are written to the local cache. Tokens preceding this window are written to the global cache only if their $g \geq \tau$; otherwise, they are discarded immediately.

4.3 Efficient decoding via lazy promotion

In the decode phase, tokens are generated autoregressively. Unlike standard inference systems (Kwon et al., 2023; Zheng et al., 2024) that continuously append new KVs to a uniformly growing cache, WG-KV manages the head-wise irregular cache lengths efficiently via a **lazy promotion** strategy. This mechanism enforces the local window size W_{local} via a ring buffer while selectively persisting high-utility states.

As illustrated in Figure 6d, for a given head, upon generating a new token (k_t, v_t) with score g_t , the system (1) inspects the “victim” token at the current **local pointer**. If the victim’s stored score satisfies $g \geq \tau$, it is (2) promoted to the global cache, advancing the **global pointer**. The new token then (3) overwrites the victim in the local cache, and the local pointer is incremented modulo W_{local} . To compute attention over the resulting ragged cache, we can fold the head dimension into the batch dimension to leverage variable-length

PagedAttention (Kwon et al., 2023) kernel, with details provided in Appendix B.

5 Experiments

In this section, we evaluate WG-KV across (1) the sparsity-accuracy tradeoff and (2) practical system efficiency, followed by (3) a qualitative analysis of its admission patterns and additional analyses.

5.1 Experimental setup

Model and training. We implement WG-KV on top of Llama-3.1-8B (Grattafiori et al., 2024) and Qwen3-4B-2507 (Yang et al., 2025a). We freeze the backbone model and train only the lightweight Write-Gate MLP. Training is performed on a subset of FineWeb-Edu (Penedo et al., 2024) with the objective of minimizing $\mathcal{L}_{\text{total}}$. Detailed training configurations are provided in Appendix C.

Evaluation scope. We analyze the results on Llama-3.1-8B in this section. To demonstrate the generalizability of WG-KV across different model architectures, we provide additional evaluation results on Qwen3-4B-2507 in Appendix G.

5.2 Sparsity-accuracy tradeoff

Benchmarks. To evaluate the robustness of our learnable admission policy, we utilize HELMET (Yen et al., 2025), a comprehensive benchmark for long-context understanding. Our evaluation covers 14 tasks, spanning five distinct categories: *Retrieval Augmented Generation*, *Passage Reranking*, *Long-Document QA*, *Summarization*, and *Many-Shot In-Context Learning*. Detailed benchmark descriptions are provided in Appendix D.

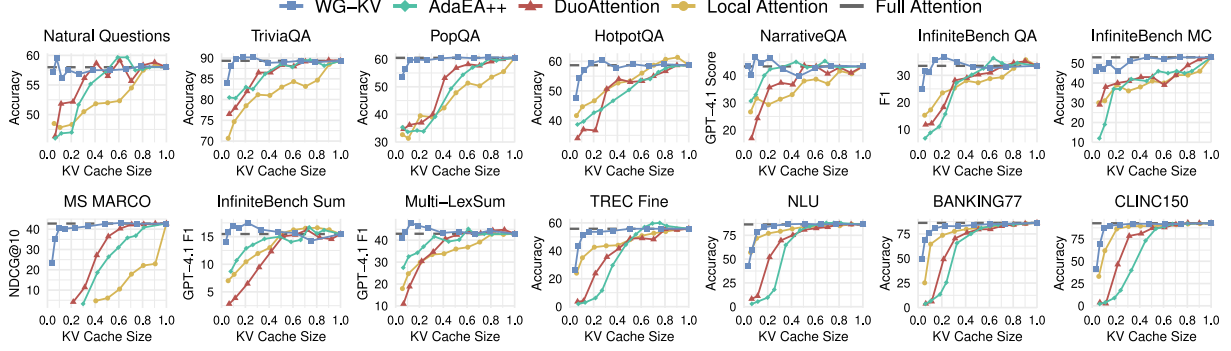


Figure 7: Long-context HELMET task accuracy on Llama-3.1-8B. The x-axis represents the normalized KV cache size. WG-KV consistently outperforms baselines under tight admission budgets.

Baselines. We compare against three representative baselines which we re-contextualize as static (input-agnostic) or training-free admission policies. This setup ensures a fair comparison where all baselines can leverage the system-level efficiency gains of KV Admission as described in Section 4. We evaluate **Local Attention**, which employs a uniform static policy retaining only recent and initial sink tokens (Xiao et al., 2024); **DuoAttention** (Xiao et al., 2025), a head-wise static policy that designates specific heads as retrieval or streaming heads based on pre-computed profiles; and **AdaEA++**, a training-free admission policy adapted from Expected Attention (Devoto et al., 2026) and AdaKV (Feng et al., 2025). Detailed baseline descriptions are provided in Appendix E.

Setup. We investigate the efficacy of different admission policies in preserving task accuracy under varying levels of KV sparsity. For WG-KV, we fix the binarization threshold $\tau = 0.1$ while sweeping λ to control the admission budget to achieve different sparsity targets. For baselines, we vary the sliding window size (Local Attention), the ratio of retrieval heads (DuoAttention), or the score threshold (AdaEA++). Figure 7 shows the results.

Results. WG-KV outperforms baselines across almost all sparsity configurations, with the most prominent advantages under tight admission budgets (e.g., admitting $<40\%$ KVs). Notably, in information-intensive tasks such as *Passage Reranking* (MS MARCO) and *Summarization* (InfiniteBench Sum and Multi-LexSum), WG-KV maintains near-lossless task accuracy even when operating at 90% sparsity (admitting only 10% KVs). In contrast, Local Attention degrades rapidly as the admission budget tightens; DuoAttention and AdaEA++, while more robust, lack the adaptability

to achieve parity with WG-KV. Their reliance on static head profiles or training-free attention statistics prevents them from capturing complex context dependencies, resulting in severe accuracy drops in high-sparsity regimes.

5.3 System efficiency

Setup. Having established WG-KV’s ability to maintain near-lossless accuracy at extreme sparsity, we validate its practical system efficiency under realistic workloads. We measure the end-to-end latency and memory usage using input sequences sampled from the PG-19 (Rae et al., 2019) long-document dataset. We evaluate the model equipped with WG-KV at $\lambda = 0.16$, which achieves approximately 80% sparsity (i.e., admitting 20% KVs), and benchmark it against the standard full-attention model (see Appendix F for detailed experimental setup). Figure 8 shows the results.

Results. Our efficient hardware-aware implementation (Section 4) effectively translates theoretical sparsity into practical system gains. In the *prefill* phase, WG-KV achieves a **2.56–3.85x** speedup across varying sequence lengths, primarily attributed to the vertical-slash sparse attention mechanism, which bypasses the massive computational cost of dense full attention. In the *decode* phase,

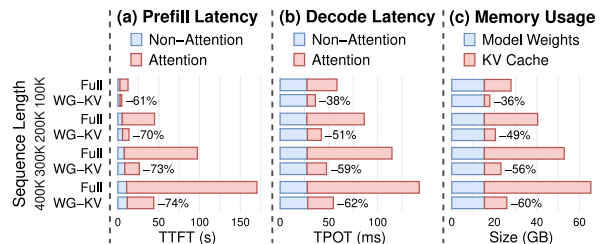


Figure 8: End-to-end latency and memory usage on Llama-3.1-8B.

it achieves a **1.61–2.63x** speedup as the reduced KV cache size alleviates memory transfer overhead and consequently lowers attention latency. This compact KV cache also translates to a **36–60%** reduction in overall memory usage.

Overhead analysis. The introduction of the Write-Gate MLP incurs minimal overhead. In terms of model size, the additional parameters constitute only $\approx 0.4\%$ of the total parameter count in both Llama and Qwen models. In terms of computation, as indicated by the slight increase in non-attention latency (which includes the execution time of the MLP) in Figure 8a-b, the introduced overhead is negligible. The massive reduction in attention computation and memory consumption far outweighs the marginal cost of the MLP.

5.4 Qualitative analysis of admission patterns

To explore the internal mechanics of WG-KV, we visualize the admission decisions across different attention heads in Figure 9, using an excerpt from Wikipedia (2026) as the input text. Remarkably, without any explicit rules, WG-KV autonomously learns highly specialized, head-specific memory allocation strategies. We observe a clear division of labor: certain heads act as structural and syntactic anchors by predominantly admitting function words (Figure 9a), punctuation marks (Figure 9b), or newline characters (Figure 9d). Conversely, other heads selectively persist information-dense tokens such as proper nouns (Figure 9c) and numerical values (Figure 9e). Some heads exhibit complex, non-trivial admission behaviors (Figure 9f-g) that defy simple lexical categorization, possibly tracking high-level semantic dependencies. This emergent specialization demonstrates WG-KV’s capability to intelligently compose a compact yet comprehensive global cache. Full visualizations are provided in Appendix M.

5.5 Additional analyses

To further investigate WG-KV’s characteristics, we provide additional analyses. (1) Appendix H explores the synergy between WG-KV and Selection/Eviction methods. (2) Appendix J analyzes hyperparameters λ and τ , and compares continuous with discrete gating. (3) Appendix K ablates the local cache to validate its necessity. (4) Appendix L demonstrates WG-KV’s input-dependent admission patterns and its adaptive memory allocation across task-specific semantic structures.

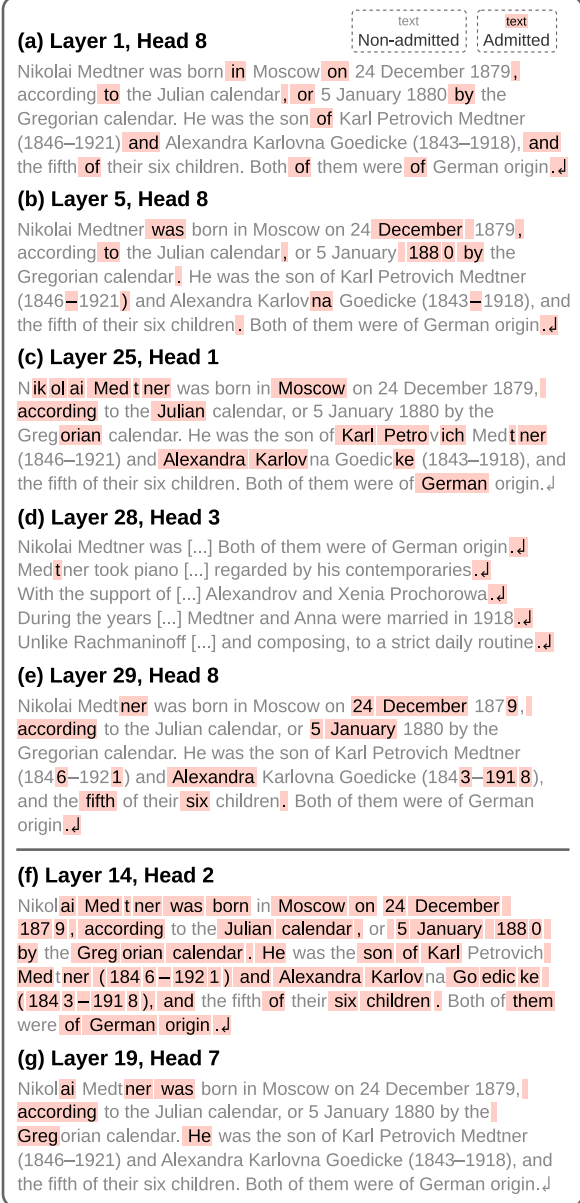


Figure 9: Visualization of admission decisions. Highlighted tokens are admitted to the global cache.

6 Conclusion

In this paper, we address the inefficiency of long-context LLM inference by formalizing KV Admission as a critical third primitive alongside KV Selection and Eviction. We introduce Write-Gated KV (WG-KV), a learnable mechanism that predictively filters redundant states pre-write, achieving substantial efficiency gains while preserving near-lossless task accuracy. Backed by our hardware-aware implementation and extensive evaluations, we demonstrate that *learning what to write* provides a principled and practical recipe for efficient long-context inference.

Limitations

WG-KV is a first step toward learnable KV Admission, but several limitations remain: (1) Our evaluation is restricted to dense models in the 4B–8B parameter range; generalizability to larger scales, Mixture-of-Experts architectures, multi-modal LLMs, and quantized KV cache (Hooper et al., 2024) remains an open question. (2) WG-KV uses a simple two-layer Write-Gate MLP trained on top of a frozen backbone. More expressive gate architectures or fully integrated training that internalizes admission behavior into model weights may yield better sparsity-quality tradeoffs, which we leave for future exploration. (3) While WG-KV is evaluated extensively on HELMET, our study does not yet cover multilingual settings (Kim et al., 2025b), multi-turn dialogues (Deshpande et al., 2025), long-horizon agentic workflows (Qiao et al., 2025), or ultra-long context regimes scaling up to millions of tokens (Rando et al., 2025). These settings may induce different token-utility patterns and require dedicated large-scale evaluation. (4) Our evaluation in Section 5.2 is limited to benchmark accuracy; open-ended generation quality—such as hallucination, faithfulness, long-form coherence, human preference, and response safety—is not explicitly evaluated. Validating WG-KV against these metrics remains an important direction for future research. (5) On the systems side, our efficiency results are limited to a single-batch, single-GPU setup. Realizing these speedups in complex production environments—such as multi-GPU serving (Mei et al., 2025), prefix caching (Pan et al., 2025), and speculative decoding (Miao et al., 2024)—introduces distinct engineering challenges that warrant further system-level exploration.

Ethical Considerations

Because WG-KV filters tokens from the KV cache, there is a risk that critical safety guardrails, alignment instructions, or system prompts could be inadvertently discarded over extended contexts. Adversaries could potentially exploit this mechanism by crafting malicious inputs designed to manipulate the gating scores and intentionally flush safety-critical tokens, thereby facilitating jailbreak attacks (Zou et al., 2023). Future deployments should incorporate targeted mitigation strategies to ensure that efficiency gains do not compromise model safety and alignment.

Acknowledgments

This work was supported in part by the National Science and Technology Council (NSTC), Taiwan, under Grants NSTC 113-2628-E-001-004-MY3, NSTC 114-2221-E-002-180-MY3, and NSTC 115-2223-E-002-001-MY3; Academia Sinica under Grant AS-IA-113-M04-ASSA; and the Center of Data Intelligence: Technologies, Applications, and Systems at National Taiwan University under Grant 115L9009, as part of the Featured Areas Research Center Program within the Higher Education Sprout Project funded by the Ministry of Education (MOE), Taiwan. We also thank Shih-Hsuan Chiu for valuable discussions on this work, and the National Center for High-performance Computing (NCHC), Taiwan, for providing computational resources.

References

- Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. *GQA: Training generalized multi-query transformer models from multi-head checkpoints*. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.
- Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. 2023. *Dynamic context pruning for efficient and interpretable autoregressive transformers*. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Charles Arnal, Konstantin Donhauser, Mohammad Pezeshki, Vivien Cabannes, David Lopez-Paz, and Kartik Ahuja. 2026. *Unveiling simplicities of attention: Adaptive long-context head identification*.
- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. *LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks*. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.
- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. *MS MARCO: A human generated machine reading comprehension dataset*. Preprint, arXiv:1611.09268.
- Adithya Bhaskar, Alexander Wettig, Tianyu Gao, Yihe Dong, and Danqi Chen. 2025. *Cache me if you*

- can: How many KV's do you need for effective long-context LMs? *Preprint*, arXiv:2506.17121.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julien Etzaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. [Lessons from the trenches on reproducible evaluation of language models](#). *Preprint*, arXiv:2405.14782.
- William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. [Reducing transformer key-value cache size with cross-layer attention](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. 2026. [Cache what lasts: Token retention for memory-bounded KV cache in LLMs](#). In *The Fourteenth International Conference on Learning Representations*.
- Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. 2025. [R-KV: Redundancy-aware KV cache compression for reasoning models](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. [Efficient intent detection with dual sentence encoders](#). In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.
- Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Yash Akhauri, Hung-Yueh Chiang, Xilai Dai, Huiqiang Jiang, Yucheng Li, Kai-Chiang Wu, Luis Ceze, and Mohamed S. Abdelfattah. 2026. [xKV: Cross-layer KV-cache compression via aligned singular vector extraction](#).
- Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S. Abdelfattah, and Kai-Chiang Wu. 2025. [Palu: KV-cache compression with low-rank projection](#). In *The Thirteenth International Conference on Learning Representations*.
- Renze Chen, Zhuofeng Wang, Beiquan Cao, Tong Wu, Size Zheng, Xiuhong Li, Xuechao Wei, Shengen Yan, Meng Li, and Yun Liang. 2024a. [ArkVale: Efficient generative LLM inference with recallable key-value eviction](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Yaoqi Chen, Jinkai Zhang, Baotong Lu, Qianxi Zhang, Chengruidong Zhang, Jing Liu, Jingjia Luo, Di Liu, Huiqiang Jiang, Qi Chen, Bailu Ding, Xiao Yan, Jiawei Jiang, Chen Chen, Mingxing Zhang, Cheng Li, Yuqing Yang, Fan Yang, and Mao Yang. 2026. [RetroInfer: A vector storage engine for scalable long-context LLM inference](#). *Preprint*, arXiv:2505.02922.
- Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. 2024b. [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.
- Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. 2025. [MagicPIG: LSH sampling for efficient LLM generation](#). In *The Thirteenth International Conference on Learning Representations*.
- Giulio Corallo and Paolo Papotti. 2024. [FINCH: Prompt-guided key-value cache compression for large language models](#). *Transactions of the Association for Computational Linguistics*, 12:1517–1532.
- Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. 2022. [FlashAttention: Fast and memory-efficient exact attention with IO-awareness](#). In *Advances in Neural Information Processing Systems*.
- Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. 2023. [Flash-decoding for long-context inference](#).
- DeepSeek-AI. 2026. [DeepSeek-V4: Towards highly efficient million-token context intelligence](#).
- DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. [DeepSeek-V3.2: Pushing the frontier of open large language models](#). *Preprint*, arXiv:2512.02556.
- Kaustubh Deshpande, Ved Sirdeshmukh, Johannes Baptist Mols, Lifeng Jin, Ed-Yeremai Hernandez-Cardona, Dean Lee, Jeremy Kritz, Willow E. Primack, Summer Yue, and Chen Xing. 2025. [Multi-Challenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 18632–18702, Vienna, Austria. Association for Computational Linguistics.
- Alessio Devoto, Maximilian Jeblick, and Simon Jégou. 2026. [Expected Attention: KV cache compression by estimating attention from future queries distribution](#).
- Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. [A simple and effective \$l_2\$ norm-based strategy for KV cache compression](#).

- In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499, Miami, Florida, USA. Association for Computational Linguistics.
- Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. 2024. [Get more with LESS: Synthesizing recurrence with KV cache compression for efficient LLM inference](#). In *Forty-first International Conference on Machine Learning*.
- Juechu Dong, BOYUAN FENG, Driss Guessous, Yanbo Liang, and Horace He. 2025. [FlexAttention: A programming model for generating fused attention variants](#). In *Eighth Conference on Machine Learning and Systems*.
- Wei Du, Shubham Toshniwal, Branislav Kisacani, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. 2025. [Nemotron-Math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision](#). *Preprint*, arXiv:2512.15489.
- Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. [Plan-and-Act: Improving planning of agents for long-horizon tasks](#). In *Forty-second International Conference on Machine Learning*.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025. [Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2025. [Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, Hayden Kwok-Hay So, Yu Hua, Ting Cao, Fan Yang, and Mao Yang. 2026. [Sparse attention adaptation for long reasoning](#). In *The Fourteenth International Conference on Learning Representations*.
- Yizhao Gao, Zhichen Zeng, DaYou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. 2025. [SeerAttention: Learning intrinsic sparse attention in your LLMs](#).
- Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. [Model tells you what to discard: Adaptive KV cache compression for LLMs](#). In *The Twelfth International Conference on Learning Representations*.
- Zijie Geng, Jie Wang, Ziqi Liu, Feng Ju, Yiming Li, Xing Li, Mingxuan Yuan, Jianye HAO, Defu Lian, Enhong Chen, and Feng Wu. 2025. [Accurate KV cache eviction via anchor direction projection for efficient LLM inference](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Nathan Godey, Alessio Devoto, Yu Zhao, Simone Scardapane, Pasquale Minervini, Éric de la Clergerie, and Benoît Sagot. 2025. [Q-Filters: Leveraging QK geometry for efficient KV cache compression](#). *Preprint*, arXiv:2503.02812.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The Llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. 2025. [OmniKV: Dynamic context selection for efficient long-context LLMs](#). In *The Thirteenth International Conference on Learning Representations*.
- Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2025. [Squeezed Attention: Accelerating long context length LLM inference](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32631–32652, Vienna, Austria. Association for Computational Linguistics.
- Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. [KVQuant: Towards 10 million context length LLM inference with KV cache quantization](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Ruida Hu, Chao Peng, Jingyi Ren, Bo Jiang, Xiangxin Meng, Qinyun Wu, Pengfei Gao, Xinchun Wang, and Cuiyun Gao. 2025. [Understanding large language model performance in software engineering: A large-scale question answering benchmark](#). In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 3025–3029, New York, NY, USA. Association for Computing Machinery.
- Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. 2025. [Locret: Enhancing eviction in long-context LLM inference with trained retaining heads](#).
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. [Categorical reparameterization with gumbel-softmax](#). In *International Conference on Learning Representations*.

- Simon Jegou and Maximilian Jeblick. 2026. [KVzap: Fast, adaptive, and faithful KV cache pruning](#). *Preprint*, arXiv:2601.07891.
- Huiqiang Jiang, YUCHENG LI, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. [MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Shibo Jie, Yehui Tang, Kai Han, Zhi-Hong Deng, and Jing Han. 2025. [SpeCache: Speculative key-value caching for efficient generation of LLMs](#). In *Forty-second International Conference on Machine Learning*.
- Peng Jin, Bo Zhu, Li Yuan, and Shuicheng YAN. 2025. [MoH: Multi-head attention as mixture-of-head attention](#). In *Forty-second International Conference on Machine Learning*.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. [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)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.
- Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. 2026. [Fast KVzip: Efficient and accurate LLM inference with gated KV eviction](#). *Preprint*, arXiv:2601.17668.
- Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W. Lee, Sangdoo Yun, and Hyun Oh Song. 2025a. [KVzip: Query-agnostic KV cache compression with context reconstruction](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yekyung Kim, Jenna Russell, Marzena Karpinska, and Mohit Iyyer. 2025b. [One ruler to measure them all: Benchmarking multilingual long-context language models](#). In *Second Conference on Language Modeling*.
- Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. 2023. [The Stack: 3 TB of permissively licensed source code](#). *Transactions on Machine Learning Research*.
- Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. [The NarrativeQA reading comprehension challenge](#). *Transactions of the Association for Computational Linguistics*, 6:317–328.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural Questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- 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, SOSP '23*, page 611–626, New York, NY, USA. Association for Computing Machinery.
- Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. 2025. [FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference](#). In *The Thirteenth International Conference on Learning Representations*.
- Adrian Łańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo Ponti. 2025. [Inference-time hyper-scaling with KV cache compression](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. [An evaluation dataset for intent classification and out-of-scope prediction](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.
- HeeJun Lee, Jina Kim, Jeffrey Willette, and Sung Ju Hwang. 2024a. [SEA: Sparse linear attention with estimated attention mask](#). In *The Twelfth International Conference on Learning Representations*.
- HeeJun Lee, Geon Park, Youngwan Lee, Jaduk Suh, Jina Kim, Wonyong Jeong, Bumsik Kim, Hyemin Lee, Myeongjae Jeon, and Sung Ju Hwang. 2025. [A training-free sub-quadratic cost transformer model serving framework with hierarchically pruned attention](#). In *The Thirteenth International Conference on Learning Representations*.
- Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. 2024b. [InfiniGen: Efficient generative inference of large language models with dynamic KV cache management](#). In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, Santa Clara, CA. USENIX Association.
- Xin Li and Dan Roth. 2002. [Learning question classifiers](#). In *COLING 2002: The 19th International Conference on Computational Linguistics*.

- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkatesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. [SnapKV: LLM knows what you are looking for before generation](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. 2025a. [MatryoshkaKV: Adaptive KV compression via trainable orthogonal projection](#). In *The Thirteenth International Conference on Learning Representations*.
- Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. 2025b. [Twilight: Adaptive attention sparsity with hierarchical top-p pruning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. 2024. [MiniCache: KV cache compression in depth dimension for large language models](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. 2025a. [RetrievalAttention: Accelerating long-context LLM inference via vector retrieval](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. 2025b. [ClusterKV: Manipulating LLM KV cache in semantic space for recallable compression](#). In *Proceedings of the 62nd Annual ACM/IEEE Design Automation Conference, DAC '25*. IEEE Press.
- Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. [Benchmarking natural language understanding services for building conversational agents](#). *Preprint*, arXiv:1903.05566.
- Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhao Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. [Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. [Learning sparse neural networks through l0 regularization](#). In *International Conference on Learning Representations*.
- Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, and 6 others. 2025. [MoBA: Mixture of block attention for long-context LLMs](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. [The concrete distribution: A continuous relaxation of discrete random variables](#). In *International Conference on Learning Representations*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [When not to trust language models: Investigating effectiveness of parametric and non-parametric memories](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.
- Yixuan Mei, Yonghao Zhuang, Xupeng Miao, Juncheng Yang, Zhihao Jia, and Rashmi Vinayak. 2025. [Helix: Serving large language models over heterogeneous GPUs and network via max-flow](#). In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1, ASPLOS '25*, page 586–602, New York, NY, USA. Association for Computing Machinery.
- Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. [SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification](#). In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS '24*, page 932–949, New York, NY, USA. Association for Computing Machinery.
- Luca Moschella, Laura Manduchi, and Ozan Sener. 2026. [Learning to evict from key-value cache](#).
- Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. 2024. [Dynamic memory compression: Retrofitting LLMs for accelerated inference](#). In *Forty-first International Conference on Machine Learning*.
- Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. [Transformers are multi-state RNNs](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741, Miami, Florida, USA. Association for Computational Linguistics.
- Zaifeng Pan, AJJKUMAR PATEL, Yipeng Shen, Zhengding Hu, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. 2025. [KVFlow: Efficient prefix caching for accelerating LLM-based multi-agent workflows](#). In *The Thirty-ninth Annual*

- Conference on Neural Information Processing Systems*.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The FineWeb datasets: Decanting the web for the finest text data at scale](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. [Benchmarking agentic workflow generation](#). In *The Thirteenth International Conference on Learning Representations*.
- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. [Compressive transformers for long-range sequence modelling](#). *Preprint*, arXiv:1911.05507.
- Stefano Rando, Luca Romani, Alessio Sampieri, Luca Franco, John Yang, Yuta Kyuragi, Fabio Galasso, and Tatsunori Hashimoto. 2025. [LongCodeBench: Evaluating coding LLMs at 1M context windows](#). In *Second Conference on Language Modeling*.
- Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2024. [SparQ attention: Bandwidth-efficient LLM inference](#). In *Forty-first International Conference on Machine Learning*.
- Utkarsh Saxena, Gobinda Saha, Sakshi Choudhary, and Kaushik Roy. 2024. [Eigen Attention: Attention in low-rank space for KV cache compression](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15332–15344, Miami, Florida, USA. Association for Computational Linguistics.
- Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. 2022. [Multi-LexSum: Real-world summaries of civil rights lawsuits at multiple granularities](#). In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Prajwal Singhanian, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. 2024. [Loki: Low-rank keys for efficient sparse attention](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. [RoFormer: Enhanced transformer with rotary position embedding](#). *Neurocomputing*, 568:127063.
- Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. 2025. [ShadowKV: KV cache in shadows for high-throughput long-context LLM inference](#). In *Forty-second International Conference on Machine Learning*.
- 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 *Forty-first International Conference on Machine Learning*.
- Philippe Tillet, H. T. Kung, and David Cox. 2019. [Triton: an intermediate language and compiler for tiled neural network computations](#). In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA. Association for Computing Machinery.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Zheng Wang, Boxiao Jin, Yuming Chang, Zhongzhi Yu, and Minjia Zhang. 2025. [Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks](#).
- Wikipedia. 2026. [Nikolai Medtner - Wikipedia](#).
- 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.
- Haoyi Wu and Kewei Tu. 2024. [Layer-Condensed KV cache for efficient inference of large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11175–11188, Bangkok, Thailand. Association for Computational Linguistics.
- Huaijin Wu, Lianqiang Li, Hantao Huang, Tu Yi, Jihang Zhang, Minghui Yu, and Junchi Yan. 2025a. [HShare: Fast LLM decoding by hierarchical key-value sharing](#). In *The Thirteenth International Conference on Learning Representations*.
- Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2025b. [Retrieval head mechanistically explains long-context factuality](#). In *The Thirteenth International Conference on Learning Representations*.
- 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 *The Thirteenth International Conference on Learning Representations*.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. [Efficient streaming language models with attention sinks](#). In *The Twelfth International Conference on Learning Representations*.
- Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. 2025a. [XAttention: Block sparse attention with antidiagonal scoring](#). In *Forty-second International Conference on Machine Learning*.
- Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. 2025b. [ThinK: Thinner key cache by query-driven pruning](#). In *The Thirteenth International Conference on Learning Representations*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. [PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3258–3270, Bangkok, Thailand. Association for Computational Linguistics.
- Qingyue Yang, Jie Wang, Xing Li, Zhihai Wang, Chen Chen, Lei Chen, Xianzhi Yu, Wulong Liu, Jianye HAO, Mingxuan Yuan, and Bin Li. 2025b. [Attention-Predictor: Temporal patterns matter for KV cache compression](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. 2025c. [LServe: Efficient long-sequence LLM serving with unified sparse attention](#). In *Eighth Conference on Machine Learning and Systems*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Yao Yao, Zuchao Li, and Hai Zhao. 2024. [SirLLM: Streaming infinite retentive LLM](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2611–2624, Bangkok, Thailand. Association for Computational Linguistics.
- Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. [LongProc: Benchmarking long-context language models on long procedural generation](#). In *Second Conference on Language Modeling*.
- Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2025. [HELMET: How to evaluate long-context models effectively and thoroughly](#). In *The Thirteenth International Conference on Learning Representations*.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. [Native Sparse Attention: Hardware-aligned and natively trainable sparse attention](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria. Association for Computational Linguistics.
- Zihao Zeng, Bokai Lin, TianQi Hou, Hao Zhang, and Zhijie Deng. 2025. [In-context KV-cache eviction for LLMs via attention-gate](#).
- Biao Zhang and Rico Sennrich. 2019. [Root mean square layer normalization](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. 2025a. [PQCache: Product quantization-based KVCache for long context LLM inference](#). *Proc. ACM Manag. Data*, 3(3).
- Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. 2025b. [SparseAttention: Accurate and training-free sparse attention accelerating any model inference](#). In *Forty-second International Conference on Machine Learning*.
- Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024a. [∞Bench: Extending long context evaluation beyond 100K tokens](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.
- Yifan Zhang and Team Math-AI. 2025. [math-ai/aime25 · datasets at hugging face](#).
- Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. 2024b. [CaM: Cache merging for memory-efficient LLMs](#)

inference. In *Forty-first International Conference on Machine Learning*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuan-dong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. *H2O: Heavy-hitter oracle for efficient generative inference of large language models*. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. *SGLang: Efficient execution of structured language model programs*. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. *Universal and transferable adversarial attacks on aligned language models*. Preprint, arXiv:2307.15043.

A Extended Related Work

Static and head-level attention specialization.

Studies on attention dynamics reveal that token utility is structurally skewed across heads and positions. Xiao et al. (2024) identify “attention sinks,” demonstrating that retaining a few initial tokens allows for stable streaming inference without the full KV cache. Building on this, Ge et al. (2024) propose profiling attention heads to statically categorize them for distinct retention policies. Mechanistic studies (Wu et al., 2025b) further reveal that specific “retrieval heads” are uniquely critical for long-range dependencies. Leveraging these characteristics, Tang et al. (2025), Xiao et al. (2025), and Bhaskar et al. (2025) enhance inference efficiency by statically identifying retrieval-critical heads to retain full context, while restricting others to local context. Unlike these static heuristics, our approach introduces a learnable, input-dependent admission mechanism that captures dynamic token utility patterns at runtime.

Read-time KV Selection. A large body of work accelerates long-context inference by selectively attending to a sparse, query-dependent subset of cached KV pairs at each step. These methods approximate attention while maintaining the complete underlying cache for future queries. Existing approaches exploit sparsity in various dimensions, including temporal (Lee et al., 2025; Jie et al., 2025; Yang et al., 2025b), channel (Ribar et al., 2024), inter-layer (Hao et al., 2025; Wu et al., 2025a), head-wise (Jin et al., 2025; Arnal et al., 2026),

top- p (Lin et al., 2025b), and low-rank (Lee et al., 2024a; Singhanian et al., 2024) structures. Some retrieval-oriented designs treat the KV cache as an indexed memory (Chen et al., 2024a; Liu et al., 2025a; Hooper et al., 2025; Zhang et al., 2025a; Liu et al., 2025b; Chen et al., 2025, 2026). Systems-oriented designs further integrate selection with KV-aware paging (Tang et al., 2024; Yang et al., 2025c), runtime management (Lee et al., 2024b; Sun et al., 2025), sparse prefill patterns (Jiang et al., 2024; Lai et al., 2025; Zhang et al., 2025b; Xu et al., 2025a), or learnable indexers (Gao et al., 2025; Lu et al., 2025; Yuan et al., 2025; Gao et al., 2026; DeepSeek-AI et al., 2025; DeepSeek-AI, 2026). Our write-time admission mechanism is complementary to these read-time strategies, allowing for compound efficiency gains by filtering redundant states prior to selection.

Post-write KV Eviction. Another line of work focuses on maintaining a bounded cache by retrospectively pruning previously accumulated states. Representative methods employ statistics derived from past attention (Liu et al., 2023; Zhang et al., 2023; Oren et al., 2024) or intrinsic token properties (Yao et al., 2024; Devoto et al., 2024; Godey et al., 2025; Devoto et al., 2026) to evict tokens. Other approaches refine the eviction criteria through learnable interaction scores (Anagnostidis et al., 2023; Zeng et al., 2025), importance estimation (Huang et al., 2025; Jegou and Jeblick, 2026; Moschella et al., 2026; Bui et al., 2026; Kim et al., 2026), prompt-guided heuristics (Li et al., 2024; Chen et al., 2024b; Yang et al., 2024; Corallo and Papotti, 2024; Geng et al., 2025), reconstruction objectives (Kim et al., 2025a), or reasoning trace redundancy (Łańcucki et al., 2025; Fu et al., 2025; Cai et al., 2025). Unlike Admission, which pre-filters redundant states at the source, Eviction manages cache bounds by removing obsolete history; thus, the two serve distinct, complementary roles in the token lifecycle.

Representation-level KV compression. A parallel line of research seeks to compress the KV representations themselves. Instead of discarding tokens, merging-based methods aggregate information by combining keys and values based on semantic similarity (Wang et al., 2025), weighted accumulation (Zhang et al., 2024b), low-rank residual recovery (Dong et al., 2024), or learnable mechanisms (Nawrot et al., 2024; DeepSeek-AI, 2026). Additionally, dimensionality reduction techniques

exploit inherent redundancy in hidden states by projecting KV pairs into lower-rank subspaces (Saxena et al., 2024; Sun et al., 2025; Chang et al., 2025; Xu et al., 2025b; Lin et al., 2025a), leveraging inter-layer redundancy (Liu et al., 2024; Wu and Tu, 2024; Brandon et al., 2024; Chang et al., 2026), or applying quantization (Hooper et al., 2024). These lines of work are largely orthogonal to our admission mechanism.

B PagedAttention Kernel Compatibility

Our head-specific admission policy results in a ragged KV cache structure where sequence lengths vary significantly across heads. To perform attention operations over this irregular structure, we can employ vLLM’s PagedAttention (Kwon et al., 2023) kernel, which natively supports variable sequence lengths across the batch dimension. By reshaping the attention input—folding the head dimension (H) into the batch dimension (B)—the computation can be treated as $B \times H$ independent sequences of varying lengths. This allows existing serving systems to apply industry-standard CUDA kernels to compute attention over the ragged KV cache seamlessly.

C Additional Details for Training

Training is performed on a subset of FineWeb-Edu (Penedo et al., 2024), using samples with lengths ranging from 4K to 32K. The training consists of 7,500 samples with a batch size of 1, totaling approximately 63M tokens (merely 0.00042% of Llama-3.1-8B’s ≈ 15 T pre-training tokens). Each training run requires approximately 6 hours on a single H100 GPU. We employ the AdamW optimizer (Loshchilov and Hutter, 2019) with a weight decay of 0.01 and a cosine learning rate schedule (linear warmup for the first 10% of steps) peaking at 0.001. To simulate instruction-following scenarios, we prepend a generic prompt (“*Please analyze and summarize the following text:\n\n*”) to each sample.

For reasoning models, we augment the training set with an additional 7,500 samples from Nemotron-Math-v2 (Du et al., 2025). While this addition doubles the training cost, we empirically observe that the resulting model consistently outperforms the version trained without this augmentation on reasoning-intensive benchmarks.

D Additional Details for HELMET Benchmark

We evaluate our method on 14 tasks from the HELMET benchmark (Yen et al., 2025). The tasks are categorized as follows:

- **Retrieval augmented generation:** Evaluated on Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), and HotpotQA (Yang et al., 2018).
- **Passage reranking:** Evaluated on MS MARCO (Bajaj et al., 2018).
- **Long-document QA:** Evaluated on NarrativeQA (Kočíský et al., 2018), InfiniteBench QA (Zhang et al., 2024a), and InfiniteBench MC (Zhang et al., 2024a).
- **Summarization:** Evaluated on InfiniteBench Sum (Zhang et al., 2024a) and Multi-LexSum (Shen et al., 2022).
- **Many-shot in-context learning:** Evaluated on TREC Fine (Li and Roth, 2002), NLU (Liu et al., 2019), BANKING77 (Casanueva et al., 2020), and CLINC150 (Larson et al., 2019).

All tasks are evaluated with 32K context length using the official evaluation scripts provided by HELMET.

E Additional Details for Admission Baselines

We compare our method against static (input-agnostic) or training-free admission baselines. The specific configurations for each baseline are detailed below.

Local Attention. A uniform static admission policy that only admits the most recent tokens while retaining a small set of initial tokens (Xiao et al., 2024) to serve as attention sinks.

DuoAttention (Xiao et al., 2025). A head-wise static policy that categorizes attention heads into retrieval heads (which retain the full KV cache) and streaming heads (which retain the local sliding window and initial attention sinks). We utilize the official profiles provided by the authors for the Llama-3.1-8B model. For Qwen3-4B-2507, as no official profile is available, we apply a similar optimization-based identification method described in the paper to determine the head allocation.

AdaEA++. Expected Attention (EA, [Devoto et al., 2026](#)) and AdaKV ([Feng et al., 2025](#)) are originally designed as post-write KV Eviction methods. To fairly benchmark them under our KV Admission framework, we adapt them into AdaEA++, a pre-write training-free admission policy. Using the same FineWeb-Edu subset utilized for training WG-KV (Appendix C), we first precompute the mean and covariance of queries for each head offline. We then apply EA’s closed-form scoring on these samples and adopt AdaKV’s head-wise allocation strategy to find the top- k score thresholds for a target admission budget. During inference, AdaEA++ calculates the expected attention score for each newly generated KV pair using the pre-computed query statistics. If the score surpasses the threshold, the token is admitted. This adaptation allows AdaEA++ to function as a strong training-free admission baseline that is fully compatible with our efficient system implementation (Section 4).

To ensure a fair comparison, we standardize the cache retention configurations across all baselines. Specifically, we fix the attention sink size to 128 for Local Attention and DuoAttention, and set the local sliding window size to 256 for DuoAttention and AdaKV++ to align with WG-KV.

F Detailed Experimental Setup for System Efficiency

This section details the software stack, attention kernels, and evaluation setup underpinning the latency results reported in Section 5.3. By rigorously accounting for all runtime overheads, we demonstrate that our results reflect the real execution time of a fully functional system prototype rather than theoretical estimates.

F.1 Software stack and attention kernels

To ensure a fair comparison, both the full-attention baseline and WG-KV are built upon Hugging Face Transformers ([Wolf et al., 2020](#)). We utilize `AutoModelForCausalLM` for inference but replace the core attention kernels as follows:

Prefill phase. We employ vLLM’s `sparse_attn_func` kernel, which is an efficient sparse FlashAttention kernel based on MInference ([Jiang et al., 2024](#)). The full-attention baseline operates with a standard causal mask, whereas WG-KV utilizes the kernel to enforce the vertical-slash sparse attention pattern.

Decode phase. We employ a custom Triton-based ([Tillet et al., 2019](#)) attention kernel. The full-attention baseline operates with a standard KV cache, whereas WG-KV utilizes the kernel to handle the ragged KV cache structure. Although this ragged structure can be mapped to vLLM’s Page-dAttention ([Kwon et al., 2023](#)) kernel via batch-folding (as detailed in Appendix B), we empirically observed that extreme variance in sequence lengths across the folded batch introduces workload imbalance, which hinders optimal GPU utilization. Our custom kernel mitigates this by dynamically balancing the workload across heads via a split-reduce strategy ([Dao et al., 2023](#)), which redistributes the ragged KV blocks across all heads into uniform-sized chunks to maximize hardware efficiency.

By standardizing the underlying attention kernels, we ensure that the observed speedups are attributable solely to algorithmic efficiency rather than differences in kernel optimizations.

F.2 System implementation and overhead accounting

Our evaluations measure the performance of a fully implemented dual-cache paged memory system. The reported latency includes the elapsed time for all CPU-side and GPU-side operations required to handle the admission decisions and the resulting ragged KV cache structure. Specifically, the timing loop encompasses the forward computation of the Write-Gate MLP, the allocation of the KV cache, the management of the page table mapping, and the generation of block indices required by the `sparse_attn_func` kernel. During the decoding phase, the measurement also includes the full overhead of the lazy promotion logic. This involves the operations to check the victim token, update the page tables, and migrate tokens from local to global cache when necessary. This comprehensive accounting ensures that the reported speedups faithfully translate to real-world efficiency.

F.3 Evaluation setup

To accurately reflect realistic workloads, evaluations are performed using input sequences sampled from the PG-19 ([Rae et al., 2019](#)) long-document dataset. All experiments are conducted on a single H200 GPU with a batch size of 1. Latency is recorded via CUDA Events after one warm-up pass. We report the end-to-end Time to First Token (TTFT) for the prefill phase, and the average

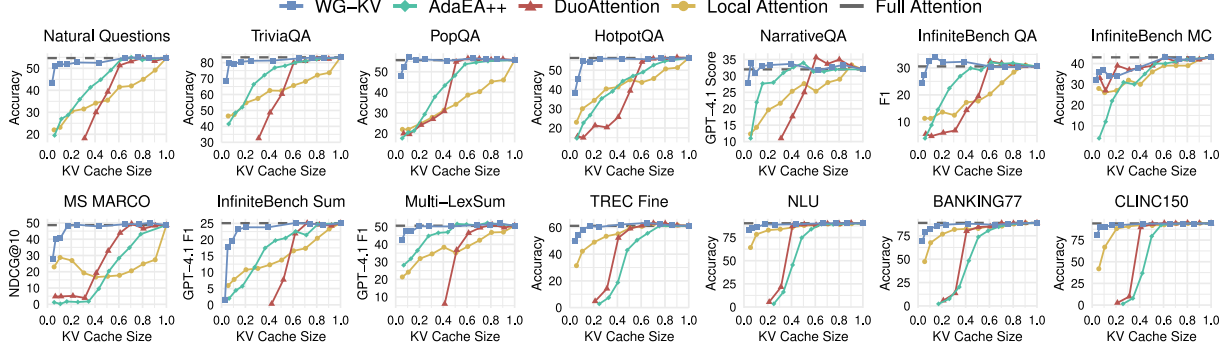


Figure 10: Long-context HELMET task accuracy on Qwen3-4B-2507. The trends align with the results on Llama-3.1-8B, showing WG-KV’s generalizability across model architectures.

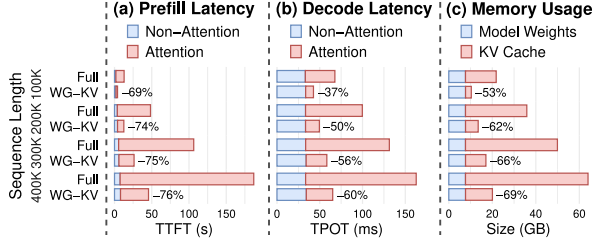


Figure 11: End-to-end latency and memory usage on Qwen3-4B-2507.

Time per Output Token (TPOT) over 300 generation steps for the decode phase.

G Evaluation Results on Qwen Model

To demonstrate the generalizability of WG-KV across different model architectures, we perform additional evaluations on the non-reasoning variant of Qwen3-4B-2507 (Yang et al., 2025a).

G.1 Sparsity-accuracy tradeoff

Figure 10 shows the evaluation results on the HELMET benchmark. Similar to the observations on Llama-3.1-8B (Figure 7), WG-KV maintains high accuracy on Qwen3-4B-2507 across varying admission budgets, consistently outperforming baselines across diverse tasks.

G.2 System efficiency

Following the same experimental setup used for Llama-3.1-8B, we measure the end-to-end latency and memory usage on Qwen3-4B-2507 at $\lambda = 0.32$, which achieves approximately 80% sparsity (i.e., admitting 20% KVs). As shown in Figure 11, WG-KV achieves a **3.23–4.17x** speedup in prefill and a **1.59–2.50x** speedup in decode across varying sequence lengths. Furthermore, WG-KV reduces memory usage by **53–69%** compared to the standard full-attention model.

H Synergy with Selection and Eviction

As conceptualized in Section 2.2, the three KV management primitives operate at distinct stages of the token lifecycle and are inherently complementary. In this section, we demonstrate that WG-KV can be seamlessly integrated with both read-time KV Selection and post-write KV Eviction methods to achieve compound gains.

H.1 Synergy with KV Selection

We investigate the interplay between WG-KV and Quest (Tang et al., 2024), a representative KV Selection method, using Llama-3.1-8B³. During decoding, Quest selects relevant KVs by utilizing key landmarks⁴ to estimate the criticality of each KV page. When evaluated at a selection budget of 2048 KVs per query, WG-KV effectively pre-filters redundant tokens, reducing KV memory usage (Figure 12a). This compacted cache then shrinks the candidate pool for selection, decreasing memory access per decode step (Figure 12b), which further translates into lower attention latency (Figure 12c).

Notably, these compound efficiency gains do not come at the expense of model capability. As shown in Figure 12d, integrating WG-KV achieves nearly identical HELMET accuracy to standalone Quest, indicating that performing selection on a cache already filtered by WG-KV preserves representation fidelity. Figure 13 further supports these findings by showing that this robustness is consistently maintained across varying selection budgets, highlighting the composable nature of pre-write Admission and read-time Selection.

³In this evaluation, WG-KV operates with $\lambda = 0.16$, admitting $\approx 20\%$ KVs.

⁴For Quest, “key landmarks” refer to the per-page minimum and maximum key vectors.

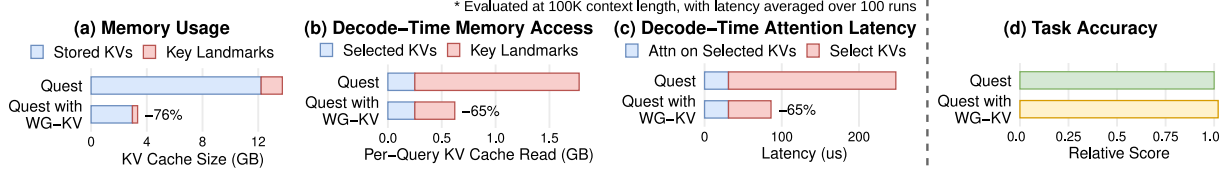


Figure 12: Synergy between WG-KV and Quest. By pre-filtering redundant tokens, WG-KV effectively (a) reduces KV memory usage. This compacted cache shrinks the candidate pool for selection, leading to (b) lower decode-time memory access and (c) reduced attention latency. Furthermore, integrating WG-KV (d) maintains nearly identical task accuracy compared to standalone Quest.

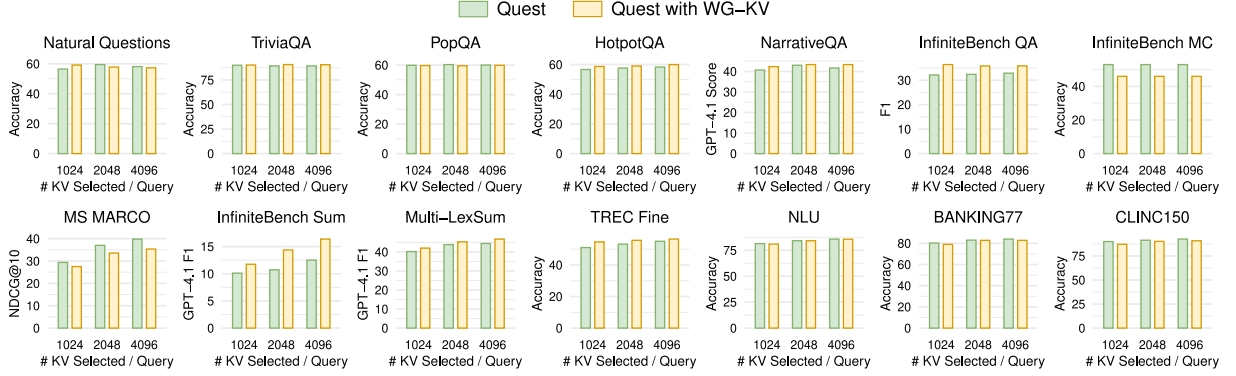


Figure 13: Long-context HELMET task accuracy under varying selection budgets. The results demonstrate that integrating WG-KV with Quest consistently matches the accuracy of standalone Quest, highlighting the composable nature of pre-write Admission and read-time Selection.

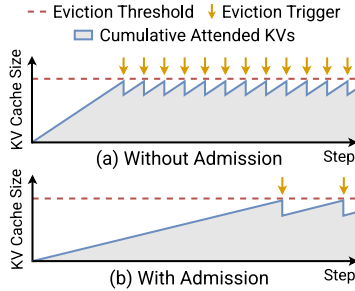


Figure 14: Conceptual illustration of the synergy between KV Admission and Eviction.

H.2 Synergy with KV Eviction

Figure 14 conceptually illustrates the synergy between KV Admission and Eviction. Without Admission, rapid KV accumulation leads to steep cache growth, causing the system to frequently hit the eviction threshold and trigger aggressive evictions (Figure 14a). By pre-filtering redundant tokens, Admission flattens the growth, delaying the onset of eviction and reducing its frequency (Figure 14b). The smaller shaded area under this flattened curve also translates to a lower cumulative attention cost over the generation process.

We empirically validate this synergy by coupling WG-KV with SnapKV (Li et al., 2024), a representative KV Eviction method, in reasoning-intensive

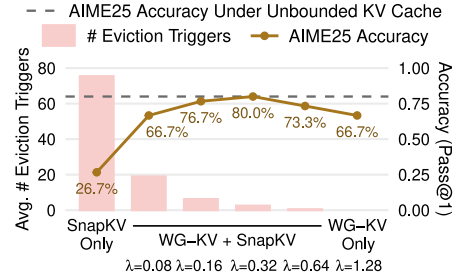


Figure 15: Synergy between WG-KV and SnapKV on AIME25 under strict memory bounds.

scenarios where long thinking traces rapidly consume KV cache memory. We evaluate this on the AIME25 benchmark (Zhang and Math-AI, 2025) using the reasoning variant of Qwen3-4B-2507 under strict memory limits. As illustrated in Figure 15, relying solely on SnapKV (Eviction) leads to catastrophic degradation (26.7% accuracy); this occurs because the “write-then-throw” inefficiency allows noise to flood the cache, triggering frequent and premature evictions that inadvertently discard critical context. Conversely, relying solely on WG-KV (Admission) to satisfy strict memory limits forces the policy to be overly aggressive, causing the model to reject potentially useful information and “starve” itself. However, combining these primi-

tives yields a superior tradeoff. WG-KV acts as a gatekeeper that filters noise at the source, effectively reducing the frequency of eviction triggers and allowing the eviction policy to focus on pruning *obsolete* history. This synergy stabilizes the reasoning process and restores accuracy to 80.0%, matching the performance of the unbounded KV baseline while strictly adhering to memory limits. Detailed experimental setup and analysis are provided in Appendix I.

I Details of Eviction Synergy for Long-Chain Reasoning

This section details the experimental setup and further analysis of the Admission-Eviction synergy introduced in Appendix H.2, particularly in reasoning-intensive scenarios where intermediate “thinking tokens” cause rapid cache expansion. While WG-KV effectively filters redundant tokens at the source, real-world hardware imposes strict limits on memory capacity. This constraint is particularly pronounced in modern high-throughput serving systems (Kwon et al., 2023; Zheng et al., 2024), where multiple concurrent requests compete for a shared memory pool. To sustain inference over prolonged generation phases—such as the extended reasoning trajectories seen in complex mathematical problem-solving—the system must enforce a hard memory budget by retrospectively pruning obsolete history. Here, we demonstrate how coupling pre-write Admission with post-write Eviction effectively reserves this limited capacity exclusively for critical context, yielding a superior memory-accuracy tradeoff compared to utilizing either primitive in isolation.

I.1 Eviction policy implementation

To evaluate the synergy between KV Admission and Eviction, we integrate WG-KV with SnapKV to manage the cache under strict memory constraints by enforcing a hard KV cache budget such that the average capacity per head is 4096 tokens. When the cache size exceeds this budget, an eviction process is triggered to remove the bottom 10% of KV pairs with the lowest importance scores.

The eviction policy operates independently for each KV head. Consider a specific head with cached keys $K \in \mathbb{R}^{N \times d}$ and a corresponding set of query heads $\mathcal{G}$ (as in Grouped-Query Attention). We define the importance score S_j for the j -th key-value pair based on queries from the most

recent window of size $W_{\text{obs}} = 256$, denoted as $\{Q_{\text{obs}}^{(h)} \in \mathbb{R}^{W_{\text{obs}} \times d} \mid h \in \mathcal{G}\}$. The scoring process proceeds in three steps:

1. **Attention computation:** We first compute the post-softmax attention scores $A^{(h)} \in \mathbb{R}^{W_{\text{obs}} \times N}$ for each query head $h \in \mathcal{G}$ against the keys K :

$$A^{(h)} = \text{Softmax} \left(Q_{\text{obs}}^{(h)} K^\top / \sqrt{d} \right)$$

2. **Score aggregation:** To capture the utility of a key across the observation window, we aggregate the scores by maximizing over the query heads in $\mathcal{G}$ and summing over W_{obs} :

$$S_j^{\text{raw}} = \sum_{i=1}^{W_{\text{obs}}} \max_{h \in \mathcal{G}} A_{i,j}^{(h)}$$

3. **Local smoothing:** Finally, we apply a max-pooling operation with a kernel size of $W_{\text{pool}} = 5$ over the sequence length N :

$$S = \text{MaxPool} \left(S^{\text{raw}}, W_{\text{pool}} \right)$$

I.2 Quantitative analysis

We evaluate the impact of WG-KV on the AIME25 benchmark using the reasoning variant of Qwen3-4B-2507. We analyze the system’s behavior in two distinct scenarios: (a) an unbounded memory setting to isolate the behavior of WG-KV, and (b) a bounded memory setting to analyze its synergy with SnapKV.

Admission alone is insufficient under strict bounds. Figure 16a illustrates the tradeoff when no size limit is enforced on the KV cache. Increasing λ reduces the KV cache size but progressively degrades accuracy. In the bounded setting (Figure 16b), relying solely on Admission to avoid hitting the memory limit necessitates an overly aggressive policy. For instance, with $\lambda = 1.28$, the admission rate is low enough that the eviction trigger count drops to zero (the cache never fills up). However, this comes at a steep cost: accuracy drops to 66.7%. This indicates that when using Admission alone, the model is forced to reject tokens that are useful, starving itself of information to satisfy the memory constraint.

Eviction alone leads to catastrophic degradation. Conversely, relying solely on Eviction (the “Off” baseline in Figure 16b) results in a catastrophic accuracy drop. Without WG-KV, the cache grows rapidly with noise tokens, triggering frequent and premature evictions that inadvertently discard critical information. Consequently, accuracy collapses to 26.7%.

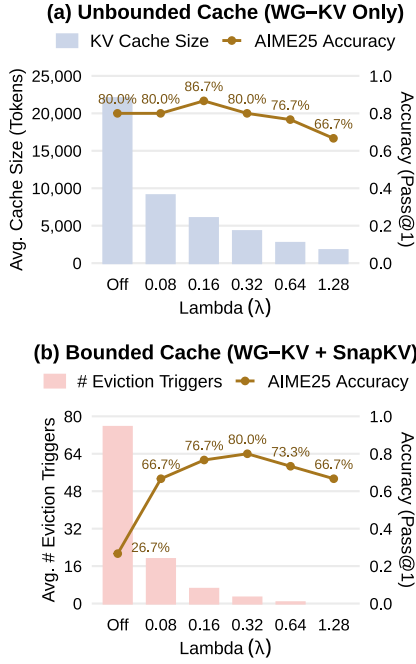


Figure 16: Admission-Eviction synergy on AIME25 using Qwen3-4B-2507. “Off” denotes the baseline where WG-KV is disabled, and the reported metrics (KV Cache Size and # Eviction Triggers) are averaged across all samples.

Optimal performance requires combining Admission and Eviction. The synergy between the two primitives achieves the best of both worlds. By setting a moderate admission budget ($\lambda = 0.32$), WG-KV prevents redundant tokens from being admitted into the global cache. This filtering process allows the eviction policy to focus on its primary strength: identifying and removing *obsolete* information that has outlived its usefulness. As shown in Figure 16b, pairing WG-KV with SnapKV allows the model to achieve 80% accuracy under strict memory constraints—matching the performance of the original model with unbounded memory as shown in Figure 16a (“Off”). This result confirms that Admission and Eviction are not mutually exclusive but complementary strategies essential for memory-constrained long-chain reasoning.

I.3 Qualitative analysis

To qualitatively illustrate the synergy between KV Admission and Eviction, Figure 17 visualizes the token lifecycle during an AIME25 reasoning trace. The visualization reveals an intriguing phenomenon: fundamental semantic anchors, such as the initial problem statement (Figure 17a), critical sub-conclusions (Figure 17c), and the final definitive answer validation (Figure 17e), are admitted

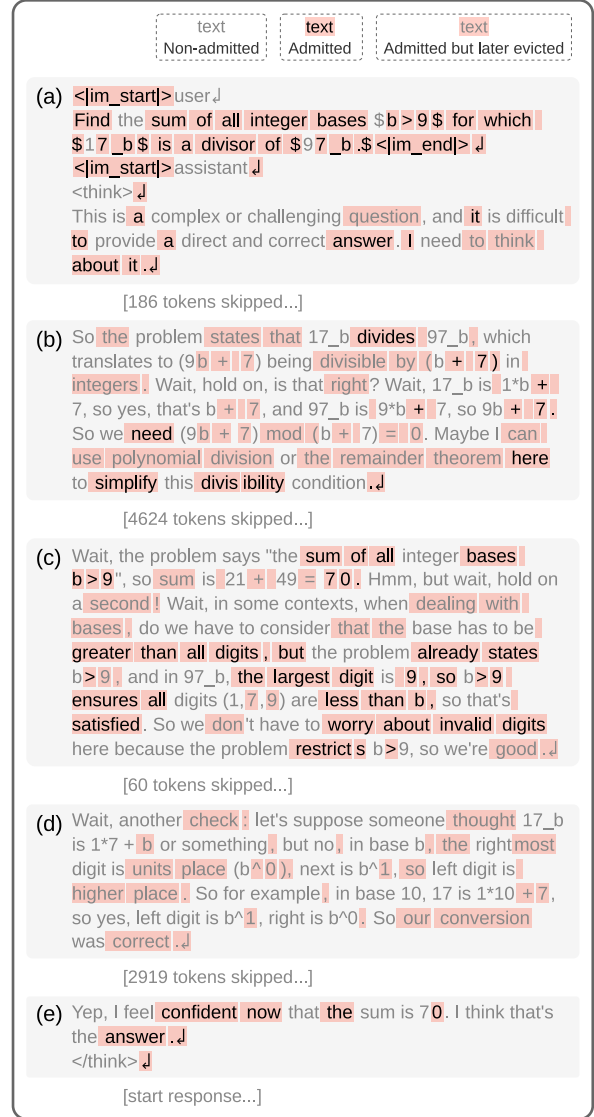


Figure 17: Visualization of the token lifecycle during an AIME25 reasoning trace using Qwen3-4B-2507 on Layer 25, Head 3.

and persistently retained to guide the global reasoning trajectory and formulate the final response. Conversely, intermediate exploratory hypotheses and trial calculations (Figure 17b and Figure 17d) are initially admitted by WG-KV to support immediate derivation steps, but as the model reaches intermediate milestones and transitions to subsequent logical phases, these temporary “scratchpad” states are retrospectively identified as obsolete and pruned by SnapKV. This demonstrates how the joint policy effectively distills convoluted thinking traces into a compact cache without sacrificing the information needed for accurate problem-solving.

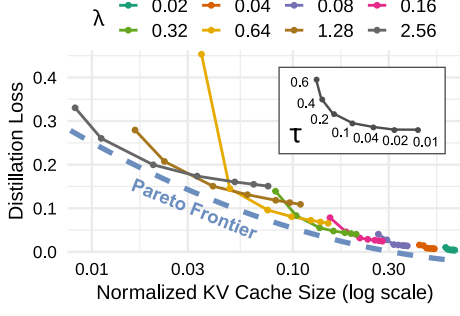


Figure 18: Impact of λ and τ on sparsity-loss tradeoff.

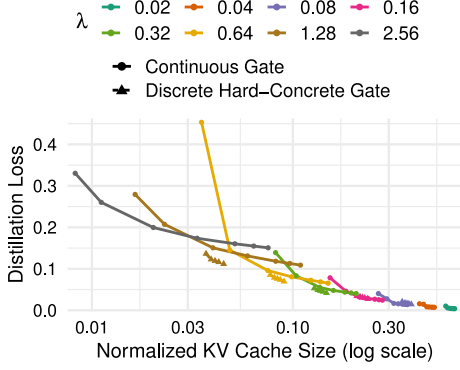


Figure 19: Comparison of continuous and discrete gating on the sparsity-loss tradeoff.

J Hyperparameter Analysis

J.1 Impact of λ and τ

In this section, we provide a detailed analysis of the impact of the training-time hyperparameter λ and the inference-time binarization threshold τ . Figure 18 plots the average distillation loss $\mathcal{L}_{\text{distill}}$ computed on a held-out validation set of 750 FineWeb-Edu samples against the normalized KV cache size. By sweeping λ and τ , we trace the Pareto frontier of the model’s sparsity-loss tradeoff. Increasing λ effectively drives the model towards higher sparsity (admitting less KV’s), albeit with an increase in $\mathcal{L}_{\text{distill}}$. Empirically, we observe that fixing $\tau \approx 0.1$ consistently yields near-optimal operating points along this frontier across various λ settings. We therefore adopt $\tau = 0.1$ for all of our experiments.

J.2 Continuous versus discrete gating

Since admission is inherently a discrete decision, binarizing a continuously relaxed gate (Sections 3.2 and 3.3) raises the concern of a train-inference mismatch: a token with an intermediate gating score remains partially visible during training, yet is either fully retained or entirely discarded at inference. To quantify this gap, we retrain the Write-Gate MLP with hard-concrete gates (Louizos et al.,

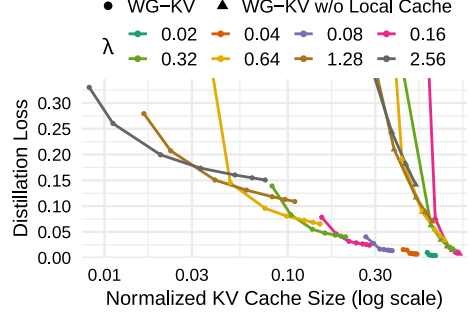


Figure 20: Impact of local cache on the sparsity-loss tradeoff. Removing local cache leads to a sharp surge in distillation loss.

2018), reinterpreting the MLP output as an unnormalized log-probability $\log \alpha$ (temperature 0.667 with stretch interval from -0.1 to 1.1) and replacing $\mathcal{L}_{\text{sparsity}}$ with the closed-form expected L_0 norm. Owing to the stretch-and-clip construction, the resulting gate is exactly binary for the vast majority of tokens and is therefore nearly invariant to τ by design, serving as a mismatch-free reference point.

Figure 19 compares the two variants. Across the admission budgets adopted in this paper, the two frontiers are nearly identical, indicating that the binarization term in $\mathcal{L}_{\text{sparsity}}$ suffices to keep the learned policy close to a discrete one and that the relaxation gap is small in practice. The two diverge only under aggressive budgets, where the discrete gate traces a modestly lower frontier.

Since the two perform comparably at the budgets we target, we adopt the continuous gate, which avoids the additional gradient variance and temperature sensitivity of stochastic sampling (Maddison et al., 2017; Jang et al., 2017).

K Ablation Study on Local Cache

To validate the necessity of local cache within WG-KV, we conducted an ablation study by removing the local cache. Specifically, we retrained the model using the same objective described in Section 3.3, but constrained the local window size W_{local} to 1. This configuration forces the model to rely exclusively on the learnable gate to decide whether to retain any token *immediately upon generation*, effectively eliminating the grace period for recent tokens.

Figure 20 compares the full WG-KV system against the variant without local cache (“WG-KV w/o Local Cache”). We plot the average distillation loss $\mathcal{L}_{\text{distill}}$ on a held-out validation set of 750 FineWeb-Edu samples against the normalized KV

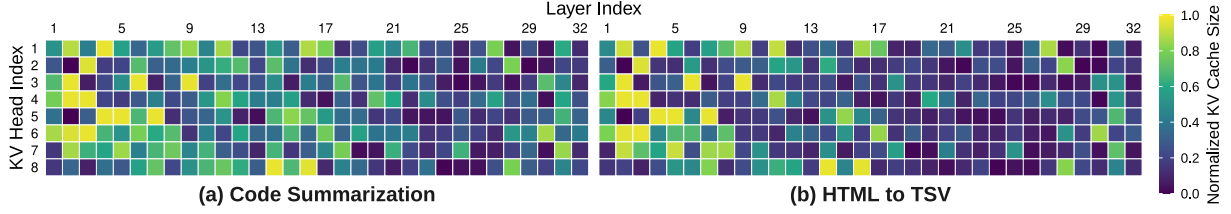


Figure 21: Visualization of input-dependent admission patterns. The heatmaps show the normalized KV cache size across each head for Llama-3.1-8B with WG-KV. We compare the (a) *Code Summarization* task, which exhibits a relatively uniform distribution, against the (b) *HTML to TSV* task, which shows a sparser pattern. This contrast demonstrates that WG-KV learns an input-dependent policy based on the semantic structure of the task.

cache size.

The results highlight a sharp degradation in the variant without local cache as admission budget tightens. This empirical evidence strongly substantiates the “transient utility” hypothesis discussed in Section 2.3: recent tokens, even those with negligible long-term utility, attract dense local attention. Without a dedicated local cache to provide a temporary grace period, the admission policy is forced to make premature retention decisions for these high-saliency states. Consequently, the system fails to decouple immediate context needs from long-term retention, confirming that the dual-cache design—combining a transient local cache with a persistent global cache—is essential for efficient and robust long-context inference.

L Analysis of WG-KV’s Input-Dependent Admission Patterns

To demonstrate that WG-KV learns an input-dependent admission policy rather than degenerating into input-agnostic head specialization, we provide both qualitative and quantitative analyses in this section.

L.1 Qualitative analysis

To show that WG-KV adapts its memory allocation based on the input context, Figure 21 visualizes the distribution of KV cache sizes across all attention heads using Llama-3.1-8B with $\lambda = 0.08$. We compare two distinct long-context tasks: (a) *Code Summarization*, utilizing a Python sample from The Stack (Kocetkov et al., 2023), and (b) *HTML to TSV*, employing a structured data extraction sample from LongProc (Ye et al., 2025).

Observations. The heatmaps reveal distinct retention characteristics tailored to the input task:

1. **Task sensitivity.** For (a) *Code Summarization*, the cache distribution is relatively uniform, in-

dicating that understanding code dependencies requires retaining context across diverse heads. In contrast, for (b) *HTML to TSV*, the pattern is significantly sparser, suggesting that the relevant information for this task is highly concentrated. The model effectively filters out most tokens, likely preserving primarily the critical structural cues (e.g., tags) to perform the extraction.

2. **Head-specific decisions.** Consistent with Figure 3, the admission policy is fine-grained. Specific heads retain nearly full history, while adjacent heads may discard most tokens.

These findings demonstrate that WG-KV adaptively modulates its memory allocation in response to the semantic structures of the task, dynamically transitioning between high-retention and high-sparsity modes.

L.2 Quantitative analysis

To further justify that admission decisions are dynamically driven by the input rather than relying on input-agnostic head profiles, we conduct a cross-sequence replay experiment. We randomly sample 1,000 pairs of 8K-length sequences (S_1, S_2) from the PG-19 (Rae et al., 2019) long-document dataset and feed them into Llama-3.1-8B with $\lambda = 0.08$. We measure the average MSE of the last-layer hidden states for S_1 against the standard full-attention model under three policies:

1. **Own Policy:** Using S_1 ’s own dynamically generated admission decisions.
2. **Replay Policy:** Forcing S_1 to adopt the exact admission decisions generated by S_2 .
3. **Random Policy:** Randomly admitting tokens for S_1 while matching the exact retention ratio of S_2 .

Table 1: Average last-layer hidden states MSE under different admission policies.

Own Policy	Replay Policy	Random Policy
0.01480	0.10297	0.10271

If WG-KV merely degenerated into learning input-agnostic head specialization, the *Replay Policy* should yield an MSE close to *Own Policy* and outperform *Random Policy*. However, as shown in Table 1, the *Replay Policy* performs almost identically to the *Random Policy* and is significantly worse than *Own Policy*. This stark contrast quantitatively confirms that WG-KV’s admission decisions are strictly input-dependent and dynamically tailored to the context of the given sequence.

M Full Visualizations of WG-KV’s Admission Patterns

In this section, we provide the full visualizations of WG-KV’s admission patterns on Llama-3.1-8B at $\lambda = 0.16$.

Figures 22 to 28 present the full-text admission patterns corresponding to the cropped snippets shown in Figure 9a-g. These visualizations demonstrate the fine-grained division of labor among individual attention heads, ranging from lexical and structural anchors to those capturing complex semantic dependencies.

Figure 29 further shows the aggregated admission decisions across all attention heads. The color intensity corresponds to the proportion of heads that admit a given token into their global cache. We observe a strong cross-head consensus for specific token categories: special control tokens (e.g., `<|begin_of_text|>`), key entities, proper nouns, and semantically dense words are highly admitted. Conversely, function words and pronouns are rarely admitted. This emergent behavior, interestingly, closely mirrors human reading cognition, where semantic anchors are committed to memory while syntactic fillers are glossed over—a strategy the model learns entirely without explicit linguistic supervision.

N Artifacts and Intended Use

Our work utilizes publicly available models, including Llama-3.1-8B (Grattafiori et al., 2024), Qwen3-4B-2507 (Yang et al., 2025a); datasets, including PG-19 (Rae et al., 2019), The Stack (Kocetkov et al., 2023), FineWeb-Edu (Penedo et al., 2024),

Nemotron-Math-v2 (Du et al., 2025), HELMET (Yen et al., 2025), AIME25 (Zhang and Math-AI, 2025), LongProc (Ye et al., 2025); software, including Triton (Tillet et al., 2019), Hugging Face Transformers (Wolf et al., 2020), FlashAttention (Dao et al., 2022), vLLM (Kwon et al., 2023), LM Evaluation Harness (Biderman et al., 2024), MInference (Jiang et al., 2024), DuoAttention (Xiao et al., 2025), FlexAttention (Dong et al., 2025), KVPress (Devoto et al., 2026). Our use of these artifacts is strictly for academic research and complies with their respective open-source licenses and intended uses.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>↓
 ↓
 Cutting Knowledge Date: December 2023.↓
 Today Date: 26 Jul 2024.↓
 ↓
 <|eot_id|><|start_header_id|>user<|end_header_id|>↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

Figure 22: Full visualization of admission decisions for Layer 1, Head 8.

<|begin_of_text|> <|start_header_id|>system <|end_header_id|> ↓
 ↓
 Cutting Knowledge Date : December 2023 ↓
 Today Date : 26 Jul 2024 ↓
 ↓
 <|eot_id|> <|start_header_id|>user <|end_header_id|> ↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

Figure 23: Full visualization of admission decisions for Layer 5, Head 8.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>↓
 ↓
 Cutting Knowledge Date: December 2023↓
 Today Date: 26 Jul 2024↓
 ↓
 <|eot_id|><|start_header_id|>user<|end_header_id|>↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

Figure 24: Full visualization of admission decisions for Layer 25, Head 1.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>↓
 ↓
 Cutting Knowledge Date: December 2023↓
 Today Date: 26 Jul 2024↓
 ↓
 <|eot_id|><|start_header_id|>user<|end_header_id|>↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

Figure 25: Full visualization of admission decisions for Layer 28, Head 3.

<|begin_of_text|> <|start_header_id|>system<|end_header_id|> ↓
 ↓
 Cutting Knowledge Date: December 2023 ↓
 Today Date: 26 Jul 2024 ↓
 ↓
 <|eot_id|> <|start_header_id|>user<|end_header_id|> ↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin. ↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries. ↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa. ↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918. ↓

Figure 26: Full visualization of admission decisions for Layer 29, Head 8.

<|begin_of_text|> <|start_header_id|>system<|end_header_id|> ↓
 ↓
 Cutting Knowledge Date: December 2023 ↓
 Today Date: 26 Jul 2024 ↓
 ↓
 <|eot_id|> <|start_header_id|>user<|end_header_id|> ↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin. ↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries. ↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa. ↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918. ↓

Figure 27: Full visualization of admission decisions for Layer 14, Head 2.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>↓
 ↓
 Cutting Knowledge Date: December 2023.↓
 Today Date: 26 Jul 2024.↓
 ↓
 <|eot_id|><|start_header_id|>user<|end_header_id|>↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

Figure 28: Full visualization of admission decisions for Layer 19, Head 7.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>↓
 ↓
 Cutting Knowledge Date: December 2023.↓
 Today Date: 26 Jul 2024.↓
 ↓
 <|eot_id|><|start_header_id|>user<|end_header_id|>↓
 ↓
 Nikolai Medtner was born in Moscow on 24 December 1879, according to the Julian calendar, or 5 January 1880 by the Gregorian calendar. He was the son of Karl Petrovich Medtner (1846–1921) and Alexandra Karlovna Goedicke (1843–1918), and the fifth of their six children. Both of them were of German origin.↓
 ↓
 Medtner took piano lessons from his mother until the age of ten. He also had lessons from his mother's brother, Fyodor Goedicke (the father of his more famous cousin Alexander Goedicke). He entered the Moscow Conservatory in 1891, and graduated nine years later in 1900, at the age of 20, receiving the Anton Rubinstein prize, having studied under Pavel Pabst, Wassily Sapellnikoff, Vasily Safonov and Sergei Taneyev among others. Despite his conservative musical tastes, Medtner's compositions and his pianism were highly regarded by his contemporaries.↓
 ↓
 With the support of Taneyev, Medtner rejected a career as a performer and turned to composition, partly inspired by Ludwig van Beethoven's late piano sonatas and string quartets. His composing career began professionally in 1903, when he started publishing his music, and it began to be performed. With the publication of his First Piano Sonata in F minor, he was noticed by Sergei Rachmaninoff, who would remain a friend of Medtner's throughout his life, as well as a supporter of his composing. Among his students in that period were Alexander Vasilyevich Alexandrov and Xenia Prochorowa.↓
 ↓
 During the years leading up to the 1917 Russian Revolution, Medtner lived at home with his parents. During that time, he fell in love with Anna Mikhaylovna Bratenskaya (1877–1965), a respected violinist and the young wife of his older brother Emil. Later, when World War I broke out, Emil was interned in Germany where he had been studying. It was during a holiday in Crimea with his friend and fellow composer-pianist Alexander Goldenweiser that Medtner composed the first of his three piano concerti. Medtner generously gave Anna the freedom to marry his brother, and Medtner and Anna were married in 1918.↓

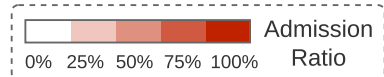


Figure 29: Aggregated admission decisions across all attention heads. Color intensity denotes the proportion of heads admitting each token. High cross-head consensus is observed on structural control tokens, key entities, and semantically dense words, whereas function words and pronouns exhibit lower admission ratios.