# LLMs Know What to Drop: Self-Attention Guided KV Cache Eviction for Efficient Long-Context Inference

**Guangtao Wang** [*], **Shubhangi Upasani, Chen Wu, Darshan Gandhi, Jonathan Li, Changran Hu, Bo Li, Urmish Thakker**
SambaNova Systems, Inc.
Palo Alto, CA 94303, USA
`https://sambanova.ai/`

## Abstract

Efficient long-context inference is critical as large language models (LLMs) adopt context windows of ranging from 128K to 1M tokens. However, the growing key-value (KV) cache and the high computational complexity of attention create significant bottlenecks in memory usage and latency. In this paper, we find that attention in diverse long-context tasks exhibits sparsity, and LLMs implicitly "know" which tokens can be dropped or evicted at the head level after the pre-filling stage. Based on this insight, we propose Self-Attention Guided Eviction (SAGE-KV), a simple and effective KV eviction cache method for long-context inference. After prefilling, our method performs a one-time top-$k$ selection at both the token and head levels to compress the KV cache, enabling efficient inference with the reduced cache. Evaluations on LongBench, InfiniteBench and three long-context LLMs (Llama3.1-8B-Instruct-128k, Llama3-8B-Prolong-512k-Instruct, and Qwen2.5-7B-Instruct-128k) show that SAGE-KV maintains accuracy comparable to full attention while significantly improving efficiency. Specifically, SAGE-KV achieves 4x higher memory efficiency with improved accuracy over the static KV cache selection method StreamLLM, and 2x higher memory efficiency with better accuracy than the dynamic KV cache selection method Quest.

## 1 Introduction

Long-context Large Language Models (LLMs) are essential for tasks like summarization, multi-hop reasoning, question answering, code understanding, personalized chatbots, recommendations, and in-context learning (Zhou et al., 2024; Li et al., 2024; Josh Achiam, 2023; Team et al., 2024). However, their deployment is limited by high computational costs, driven by the KV cache's memory demands and attention computation latency (Li et al., 2024). As attention latency grows with KV cache size, efficient memory and computation management are crucial for real-world feasibility. Addressing these challenges is key to fully leveraging long-context LLMs.

Recent efforts to reduce KV cache requirements and accelerate inference in long-context LLMs have gained increasing attention, mainly by exploiting attention sparsity (Zhang et al., 2023; Liu et al., 2024). Sparsity patterns fall into two main categories: **static** (Xiao et al., 2024c;b) and **dynamic** (Xiao et al., 2024a; Tang et al., 2024; Liu et al., 2024; Sun et al., 2024). Static sparsity methods predefine token selection rules, avoiding runtime computation and enabling faster inference. For instance, StreamLLM (Xiao et al., 2024c) retains sink tokens (early context) and recent tokens, reducing KV cache size without additional selection overhead.

Dynamic sparsity methods adaptively select representative tokens per generation step, often yielding higher accuracy but at increased computational cost. They require careful hyperparameter tuning (e.g., chunk size in InfLLM (Xiao et al., 2024a), or ANN index construction in RetrievalAttention (Liu et al., 2024)) and must retain the full KV cache as a candidate pool, limiting memory

[*]Corresponding author. xjtuwgt@gmail.com, guangtao.wang@sambanovasystems.com

savings. Although offloading KV caches to the CPU reduces GPU memory usage, it incurs high retrieval latency (Sun et al., 2024). As a result, dynamic methods demand sophisticated CPU-GPU coordination (Xiao et al., 2024a; Lee et al., 2024b; He & Zhai, 2024; Liu et al., 2024), increasing implementation complexity.

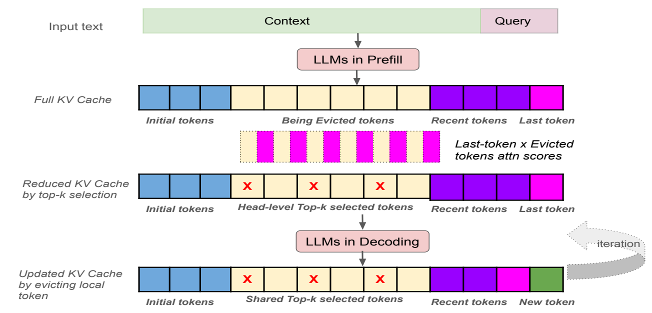

Figure 1: **The illustration of SAGE-KV with the single-pass KV cache selection**. The full KV cache consists of four parts: initial tokens, tokens for eviction, recent tokens, and the last token. To construct a reduced KV cache, we select the top-$k$ evicted tokens based on their attention scores with the last token and concatenate them with the initial and recent tokens. The updated cache is used for continuous token generation, with each new token (green) added to the recent tokens, updating the recent window for the next step.

In this paper, we tackle efficient long-context inference in LLMs by leveraging the observation that, after the pre-filling stage, LLMs naturally focus on critical information. We analyze the sparsity of the attention score and find that the attention heads selectively highlight important tokens. This insight motivates our optimization of KV cache compression at the head level. Based on this, we propose **Self-Attention Guided Eviction for KV Cache** (SAGE-KV) (Fig. 1), a novel method that uses attention scores to guide eviction of KV cache, significantly improving inference efficiency while preserving precision.

SAGE-KV employs a single-pass token-level KV cache selection strategy, compressing the KV cache once after the pre-filling stage using attention scores (Fig. 1). Only the compressed KV cache is retained, eliminating redundant KV selection during token generation, unlike block-level dynamic methods. This reduces computational overhead while preserving essential information for inference. By retaining only the most relevant tokens, SAGE-KV ensures both efficiency and accuracy. It combines the fixed sparsity of static methods—benefiting from their single-pass processing and structured cache—with the context-adaptive selection of dynamic methods. This synergy enables SAGE-KV to achieve superior efficiency while maintaining or even improving performance over existing dynamic block-level KV selection approaches.

Extensive experiments across long-context LLMs and benchmarks validate these advantages. SAGE-KV achieves nearly 4x higher memory efficiency and improved accuracy compared to the static KV eviction method StreamLLM Xiao et al. (2024c), and 2x higher memory efficiency over the dynamic block-level KV eviction method Quest Tang et al. (2024). This simple yet effective approach accelerates long-context inference while remaining easy to integrate, offering a promising solution to the challenges of long-context LLM inference.

## 2 Self-Attention Guided KV Cache Eviction

Given a long input sequence $s = [t_i]_{i=1}^{N}$ of length $N$, consisting of a context followed by a query, the pre-filling step produces the full key-value cache for layer $l$: $\mathbf{P}^l = [(\mathbf{k}_i^l, \mathbf{v}_i^l)]_{i=1}^{N}$. As illustrated in Fig. 1, the proposed method proceeds as follows.

**Step 1: Full KV Cache Partition**. We divide the KV cache $\mathbf{P}^l$ into four parts: (1) initial tokens or sink tokens $\mathbf{S}^l = \mathbf{P}^l_{1:S}$ with length $S$, (2) evicted tokens $\mathbf{E}^l = \mathbf{P}^l_{S+1:S+E}$ with length $E$, (3) recent tokens $\mathbf{R} = \mathbf{P}^l_{S+E+1:N-1}$ with length $R = N - 1 - (S + E)$ and (4) the last token's KV cache $\mathbf{P}^l_N$. Sink and recent tokens are retained separately, as attention analysis shows that initial and most recent tokens typically receive higher attention scores across all heads (Xiao et al., 2024c), which refers to this as the "attention sink".

**Step 2: Representative Token/KV Cache Selection**. We select representative KV cache entries based on the attention scores between the last token of the input sequence and the evicted tokens. Let $\mathbf{q}^l \in \mathcal{R}^{H_q \times d_h}$ denote the query vector corresponding to the last token's KV cache $\mathbf{P}^l_N$, where $H_q$ is the number of query heads, $d_h$ is the head dimension, and $H_q \times d_h = d$, the hidden dimension. In decoder-only LLMs, the last token's hidden representation often serves as an embedding for the entire input sequence (Lee et al., 2024a; BehnamGhader et al., 2024). Thus, $\mathbf{q}^l$ acts as a representative embedding for the full sequence.

For each layer $l$, we use $\mathbf{q}^l$ to select the top-$k$ KV cache entries, $\mathbf{E}^l_{\text{top}_k}$, based on the attention scores with $\mathbf{E}^l$. This yields $H_q$ groups of the top-$k$ KV caches, forming a representative set of key-value pairs for the next-token generation.

**Step 3: Reduced KV Cache Construction**. The reduced KV cache $\mathbf{C}$ is formed by concatenating sink token KV cache, selected top-$k$ token KV cache, recent token KV cache as well as the last KV cache, resulting in $\mathbf{C} = \text{Concat}(\mathbf{S}, \mathbf{E}_{top_k}, \mathbf{R}, \mathbf{P}^l_N)$ with total length $S + k + R + 1$.

**Step 4: Generation/Decoding**. The output is generated using the reduced KV cache $\mathbf{C}$. Each new token's KV pair is added to the recent window $\mathbf{R}$, evicting the oldest entry in $\mathbf{R}$ to maintain its size. This process repeats until generation completes.

## 3 Experiments and Results

### 3.1 Experimental Setup

**Benchmarks and long context LLMs**. We evaluate long-context performance on LongBench (Bai et al., 2023) and InfiniteBench (Zhang et al., 2024) using Llama3.1-8B-Instruct (128k)(Dubey et al., 2024), Llama-3-8B-ProLong-512k-Instruct(Gao et al., 2024), and Qwen2.5-7B-Instruct (128k) (Hui et al., 2024), covering a diverse set of models.

**Baselines**. We compare our method with the following: (1) **Full-Attention Models**: which use standard full attention; (2) **Hugging Face StreamLLM** (Xiao et al., 2024c): the official implementation; (3) **Our StreamLLM Implementation**: which addresses position conflicts[1] in the Hugging Face version by introducing (a) StreamLLM$_R$, which stores pre-RoPE KV cache and applies local window relative positional encoding, and (b) StreamLLM$_{\text{Abs}}$, which uses absolute positional encoding; (See Appendix) (4) **Quest** (Tang et al., 2024): a block-wise top-$k$ selection method; and (5) **InfLLM** (Xiao et al., 2024a): which integrates sink tokens, recent tokens, and block-wise top-$k$ selection. See Appendix for SAGE-KV implementation details.

Table 1: Accuracy Comparison of KV Cache Eviction Methods on LongBench

| Method | 2wikimqa | gov.rep | lcc | mulfqa | mul.news | nar.qa | pass.ret | qasper | rep-p | trec | hotpotqa | musique | qmsum | samsum | tri.qa | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Llama3.1-8B-Instruct (128k) | | | | | | | | | |
| FullAttention | 50.01 | 34.66 | 65.52 | 56.85 | 27.11 | 31.38 | 100.0 | 46.6 | 58.3 | 73.0 | 58.1 | 32.52 | 25.19 | 43.75 | 92.11 | 53.01 |
| StreamLLM$_{HF}$ | 49.47 | 32.66 | 65.37 | 55.54 | 27.05 | 30.98 | 79.0 | 46.27 | 54.87 | 65.0 | 56.20 | 30.35 | 24.45 | 42.89 | 91.93 | 50.14 |
| StreamLLM$_R$ | 49.08 | 33.68 | 65.48 | 55.30 | 27.14 | 31.14 | 83.0 | 45.72 | 55.38 | 73.0 | 57.44 | 30.37 | 24.61 | 43.69 | 91.97 | 51.13 |
| StreamLLM$_{Abs}$ | 48.58 | 33.96 | 65.33 | 55.06 | 27.14 | 30.6 | 84.5 | 45.71 | 56.11 | 73.0 | 57.58 | 30.08 | 24.37 | 43.65 | 92.04 | 51.18 |
| Quest | 45.83 | 35.25 | 64.8 | 55.36 | 27.25 | 28.14 | 99.5 | 44.95 | 59.69 | 70.5 | 55.17 | 31.69 | 25.40 | 42.75 | 82.83 | 51.27 |
| InfLLM | 45.61 | 34.42 | 67.2 | 51.21 | 27.69 | 23.82 | 92.0 | 44.75 | 64.44 | 69.0 | 50.65 | 23.74 | 24.24 | 43.4 | 92.16 | 50.29 |
| SAGE-KV (Ours) | 48.33 | 33.98 | 65.01 | 56.16 | 26.95 | 31.42 | 100.0 | 46.21 | 55.46 | 73.0 | 58.08 | 33.29 | 24.32 | 43.56 | 91.64 | **52.49** |
| | | | | | | | Llama-3-8B-ProLong-512k-Instruct | | | | | | | | | |
| FullAttention | 24.71 | 33.23 | 66.01 | 53.91 | 27.67 | 29.66 | 100.0 | 31.05 | 67.4 | 73.5 | 36.24 | 13.96 | 25.53 | 43.19 | 90.01 | 47.74 |
| StreamLLM$_{HF}$ | 24.06 | 30.78 | 66.11 | 53.16 | 27.73 | 27.79 | 67.5 | 30.66 | 66.0 | 74.0 | 34.64 | 12.57 | 23.88 | 43.55 | 90.48 | 44.86 |
| StreamLLM$_R$ | 25.26 | 32.38 | 65.95 | 51.62 | 27.58 | 26.36 | 68.0 | 30.65 | 66.49 | 74.0 | 35.31 | 11.84 | 24.66 | 43.53 | 89.51 | 44.88 |
| StreamLLM$_{Abs}$ | 25.63 | 32.48 | 65.95 | 52.74 | 27.54 | 26.67 | 68.5 | 30.54 | 65.8 | 74.0 | 34.28 | 11.52 | 24.60 | 43.12 | 89.38 | 44.85 |
| Quest | 24.86 | 31.90 | 65.20 | 50.75 | 28.13 | 26.18 | 99.5 | 29.52 | 67.33 | 72.5 | 35.58 | 14.89 | 25.77 | 43.11 | 86.76 | 46.80 |
| InfLLM | 24.84 | 28.37 | 38.10 | 68.31 | 25.43 | 49.18 | 14.34 | 22.25 | 66.4 | 49.5 | 29.56 | 23.16 | 41.17 | 75.50 | 90.06 | 43.08 |
| SAGE-KV (Ours) | 25.57 | 32.08 | 65.96 | 54.74 | 27.72 | 29.90 | 100.0 | 30.03 | 67.39 | 73.0 | 36.48 | 13.76 | 25.0 | 42.94 | 89.98 | **47.64** |
| | | | | | | | Qwen2.5-7B-Instruct (128k) | | | | | | | | | |
| FullAttention | 46.11 | 33.46 | 62.03 | 51.77 | 25.91 | 28.79 | 100.0 | 44.79 | 66.83 | 73.5 | 58.38 | 29.23 | 24.32 | 47.92 | 88.93 | 52.13 |
| StreamLLM$_{HF}$ | 11.86 | 29.70 | 61.92 | 22.64 | 23.08 | 3.37 | 59.58 | 10.09 | 64.87 | 72.5 | 11.33 | 7.05 | 16.88 | 46.45 | 88.19 | 35.3 |
| StreamLLM$_R$ | 44.92 | 33.2 | 62.18 | 49.10 | 25.73 | 25.85 | 70.0 | 44.79 | 65.88 | 72.0 | 50.73 | 24.83 | 23.20 | 47.73 | 89.41 | 48.64 |
| StreamLLM$_{Abs}$ | 45.46 | 33.23 | 61.82 | 49.63 | 25.71 | 23.65 | 71.5 | 44.79 | 65.29 | 72.5 | 52.61 | 24.9 | 23.39 | 47.45 | 88.64 | 48.70 |
| Quest | 45.70 | 32.73 | 60.45 | 50.41 | 25.77 | 26.34 | 97.83 | 44.70 | 59.95 | 73.0 | 57.65 | 28.79 | 24.50 | 47.02 | 86.78 | 50.77 |
| InfLLM | 43.12 | 32.90 | 52.51 | 60.45 | 24.65 | 48.23 | 28.11 | 22.76 | 63.63 | 69.0 | 43.27 | 23.05 | 45.20 | 66.5 | 88.45 | 47.46 |
| SAGE-KV (Ours) | 45.81 | 32.53 | 61.72 | 50.29 | 25.79 | 28.05 | 100.0 | 44.21 | 63.0 | 72.0 | 55.90 | 29.16 | 23.17 | 47.51 | 88.65 | **51.19** |

Note: We do not compare with H2O, as prior work has already shown that Quest outperforms it (Tang et al., 2024).

### 3.2 Results

**Accuracy comparison** on LongBench is represented in Table 1[2], which shows that:

[1] `https://github.com/huggingface/transformers/issues/35350`

[2] Please refer results of InfiniteBench in Appendix

- SAGE-KV achieves accuracy comparable to full attention by applying self-attention-guided KV cache selection after pre-filling. Unlike per-token generation based selection, it selects KV entries in a single pass using the last input token's attention scores, leveraging LLMs' inherent ability to prioritize key tokens for effective answer generation.

- StreamLLM's static sparse KV selection degrades accuracy in long-context LLMs by discarding essential information from the middle of the input. Our implementation outperforms the Hugging Face (HF) version, with significantly better results on Qwen2.5-7B-Instruction and slightly higher accuracy on Llama3.1-8B-Instruct. The improvement likely stems from a flaw in HF's RoPE rotation reported in Hugging Face Issue (2024), which misaligns key cache positions and increases relative token distances, particularly affecting Qwen2.5. Our implementation corrects this, ensuring more stable performance and underscoring the importance of precise implementation for StreamLLM, especially in position-sensitive models.

- Dynamic sparse KV methods like InfLLM and Quest retrieve blocks per step but underperform SAGE-KV due to imprecise block-wise top-$k$ selection based on pooled vectors (Liu et al., 2024; Tang et al., 2024; Gao et al., 2024). While faster, this approach often misses key tokens. In contrast, SAGE-KV 's token-level selection better approximates full attention, boosting accuracy and highlighting the importance of fine-grained KV management.

**Memory efficiency analysis**. We evaluate KV cache eviction methods on Llama3.1-8B-Instruct, measuring average accuracy across eight LongBench tasks (See Appendix) under token budgets B of 0.5k, 1k, 2k, 4k, and 8k. Results in Fig. 2 reveal the following insights.

(1) All methods perform better with a larger token budget, suggesting improved information retention and accuracy.

(2) SAGE-KV consistently outperforms StreamLLM's static sparse KV selection across all token budgets by reducing the significant information loss caused by discarding middle sections. Instead, SAGE-KV employs a self-attention-guided top-$k$ token selection strategy, preserving critical information within the same budget. This demonstrates the effectiveness of our adaptive KV eviction method.

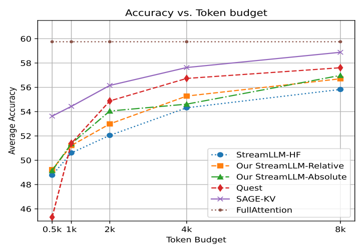

Figure 2: Token Budget Analysis.

(3) With a 2k token budget, SAGE-KV achieves the same accuracy as StreamLLM at 8k, improving memory efficiency by $\sim$4x on LongBench tasks. This highlights the advantage of attention-guided KV eviction in balancing performance and memory usage.

(4) SAGE-KV achieves 2× memory efficiency by matching Quest's performance using half the token budget (4k vs. 8k). Unlike Quest's coarse chunk-level top-$k$ selection, SAGE-KV uses token-level selection for finer context retention, leveraging LLMs' ability to identify important tokens.

## 4 Conclusion and Future Work

We introduce SAGE-KV, an efficient KV cache eviction method for long-context LLM inference. Exploiting attention sparsity, SAGE-KV compresses the KV cache after prefilling for direct use in generation. Our experiments show that SAGE-KV matches the inference speed of static sparse methods while preserving accuracy close to full attention. It achieves $\sim$*4x higher memory efficiency* and *greater accuracy* than the static eviction method StreamLLM, and $\sim$*2x higher memory efficiency* with *better accuracy* than the dynamic method Quest. Additionally, SAGE-KV seamlessly integrates with popular LLM frameworks, including Hugging Face Transformers, Meta's LLaMA, and Alibaba's Qwen, ensuring broad applicability.

**Future Work**. Current long-context tasks focus on short outputs like QA and retrieval, but long-text generation may require more than a single top-$k$ selection. Future work will explore long-output benchmarks and interval-based updates, where the LLM periodically refreshes key tokens to enhance coherence and relevance.

## REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*, 2024.

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

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). *arXiv preprint arXiv:2410.02660*, 2024.

Jiaao He and Jidong Zhai. Fastdecode: High-throughput gpu-efficient llm serving using heterogeneous pipelines. *arXiv preprint arXiv:2403.11421*, 2024.

Hugging Face Issue. SinkCache (StreamLLM) implemented over Post-RoPE Key cache might result in confused position for inference, 2024. URL `https://github.com/huggingface/transformers/issues/35350`.

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

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

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024a.

Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. {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)*, pp. 155–172, 2024b.

Haoyang Li, Yiming Li, Anxin Tian, Tianhao Tang, Zhanchao Xu, Xuejia Chen, Nicole Hu, Wei Dong, Qing Li, and Lei Chen. A survey on large language model acceleration based on kv cache management. *arXiv preprint arXiv:2412.19442*, 2024.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.

Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. In *Forty-first International Conference on Machine Learning*, 2024.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024c.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. Infinite 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)*, pp. 15262–15277, 2024.

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

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

## A Appendix

### A.1 StreamLLM with Relative and Absolute Positions

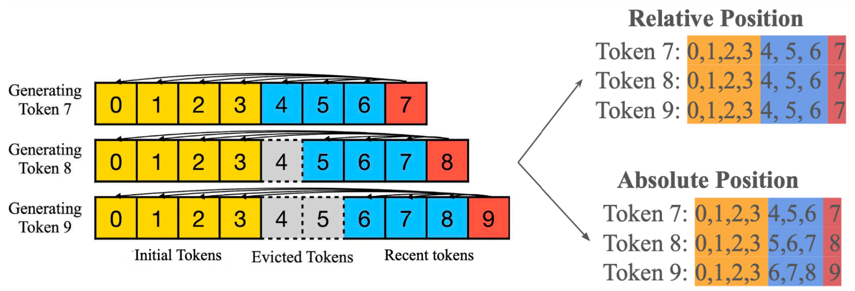

Figure 3: Implementation of Absolute and Relative Positioning in StreamLLM. Relative position assigns indices within StreamLLM's sliding window, dynamically shifting as the window moves to maintain a bounded range. In contrast, absolute position assigns a fixed index to each token based on its original sequence, increasing continuously as new tokens are added.

### A.2 Experimental Results over InfiniteBench

Table 2: Comparison of KV Cache Eviction Methods on InfiniteBench with Different LLMs

| Method | Code.Debug | Choice.En | Math.Find | Number | PassKey | QA.CH | Book.Sum | QA.EN | Diag.EN | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Llama3.1-8B-Instruct (128k) | | | | | |
| FullAttention | 22.59 | 61.57 | 25.14 | 99.49 | 100.0 | 34.53 | 16.82 | 35.44 | 22.5 | 46.45 |
| StreamLLM$_{HF}$ | 22.84 | 62.88 | 25.14 | 3.73 | 6.78 | 27.71 | 13.8 | 31.26 | 17.0 | 23.46 |
| StreamLLM$_{R}$ | 22.59 | 60.7 | 32.57 | 6.61 | 6.78 | 25.33 | 16.49 | 30.35 | 14.5 | 23.99 |
| StreamLLM$_{Abs}$ | 22.59 | 62.88 | 24.57 | 5.59 | 6.78 | 26.71 | 16.0 | 31.05 | 14.5 | 23.41 |
| SAGE-KV (ours) | 22.59 | 62.88 | 26.0 | 94.24 | 100.0 | 30.51 | 16.47 | 33.01 | 20.5 | 45.13 |
| | | | | | Llama-3-8B-ProLong-512k-Instruct | | | | | |
| FullAttention | 38.83 | 63.76 | 20.86 | 100.0 | 100.0 | 28.37 | 8.48 | 33.36 | 10.5 | 44.91 |
| StreamLLM$_{HF}$ | 39.85 | 64.19 | 20.86 | 5.08 | 6.78 | 23.8 | 7.69 | 28.2 | 9.0 | 22.83 |
| StreamLLM$_{R}$ | 36.55 | 63.76 | 21.43 | 6.78 | 6.78 | 22.97 | 9.61 | 28.19 | 9.5 | 22.84 |
| StreamLLM$_{Abs}$ | 36.55 | 63.76 | 22.0 | 6.78 | 6.78 | 22.54 | 9.93 | 27.58 | 10.0 | 22.88 |
| SAGE-KV (ours) | 36.04 | 63.76 | 21.43 | 86.78 | 100.0 | 26.41 | 8.24 | 32.77 | 10.5 | 42.88 |
| | | | | | Qwen2.5-7B-Instruct (128k) | | | | | |
| FullAttention | 33.5 | 49.78 | 33.43 | 94.58 | 96.44 | 28.09 | 12.33 | 15.96 | 11.5 | 41.73 (26.37) |
| StreamLLM$_{HF}$ | 30.46 | 51.97 | 28.0 | 0.51 | 5.08 | 10.35 | 10.98 | 5.72 | 12.5 | 17.29 (21.43) |
| StreamLLM$_{R}$ | 27.41 | 51.09 | 39.43 | 6.78 | 6.78 | 22.79 | 14.97 | 14.77 | 11.0 | 21.67 (25.92) |
| StreamLLM$_{Abs}$ | 31.47 | 48.03 | 28.57 | 5.08 | 5.08 | 22.64 | 14.11 | 13.45 | 9.5 | 19.77 (23.97) |
| SAGE-KV (ours) | 32.23 | 48.47 | 34.86 | 5.25 | 47.29 | 27.66 | 12.76 | 15.58 | 11.0 | 26.12 (26.08) |

Note: For Qwen2.5-7B-Instruct, we also report average accuracy (in parentheses) excluding the "Number" and "PassKey" retrieval tasks.

Table 2 compares the accuracy of Llama-3-based and Qwen2.5 models on InfiniteBench. The results show that:

- SAGE-KV achieves accuracy comparable to full attention in long-context LLaMA models, including LLaMA3.1-8B-Instruct and LLaMA3-8B-ProLong-512k-Instruct. Unlike traditional methods, it selects key tokens in a single pass using the last input token's attention scores, eliminating per-token selection during generation. For Qwen2.5-7B-Instruct, SAGE-KV matches full attention in most tasks, except for two retrieval benchmarks (Number and PassKey), while still outperforming StreamLLM. This highlights LLMs' ability to naturally prioritize critical tokens for effective response generation.
- StreamLLM's static sparse KV selection leads to suboptimal accuracy in long-context LLMs by discarding essential information from the middle of the input sequence. However, our implementation outperforms the official Hugging Face (HF) version, achieving significantly better performance with Qwen2.5-7B-Instruction and slightly higher average accuracy with Llama3.1-8B-Instruct. These findings underscore the importance of precise implementation in evaluating StreamLLM, especially for models sensitive to token-relative positioning, consistent with the results in LongBench (Table 1).

### A.3 Implementation Details of SAGE-KV with LLMs

**Hyper-parameter settings**. For our method SAGE-KV, suppose that the token budget is $B$ and the query group number is $G = h_q/h_v$ where $h_q$ and $h_{kv}$ are the query head number and ke/value head number, respectively, we set the sink window as $B/4$, and $k = \frac{B}{2G}$, and the recent window size as $B/4$. For Llama3.1-8B-Instruct and Llama-3-8B-ProLong-512k-Instruct, we set $B$ = 8192, and thus sink window size = 2048 and $k$ = 1024, the local window size = 2048. For Qwen2.5, since the query group number is 7, we set $B$ = 8192, thus sink window size = 2048, $k$ = 512 and the local window size = 8192 - 2048 - 7 $\cdot$ 512 = 2560. For other baselines, we set the same token budget $B$ 8192. Absolute positioning is applied in SAGE-KV to the reduced KV cache to maintain token order.

**Task names for token budget analysis**. We use the following eight LongBench tasks for hyper-parameter analysis, as in (Tang et al., 2024): "gov-report", "multifieldqa-en", "narrativeqa", "passage-retrieval-en", "qasper", "repobench-p", "hotpotqa", and "triviaqa".