

SALE : LOW-BIT ESTIMATION FOR EFFICIENT SPARSE ATTENTION IN LONG-CONTEXT LLM PREFILLING

Anonymous authors

Paper under double-blind review

ABSTRACT

Many advanced Large Language Model (LLM) applications require long-context processing, but the self-attention module becomes a bottleneck during the prefilling stage of inference due to its quadratic time complexity with respect to sequence length. Existing sparse attention methods accelerate attention computation by skipping less significant regions of the attention map. However, these approaches typically perform coarse-grained inspection of the attention map, resulting in their suboptimal performance. In this paper, we propose **SALE**, a fine-grained sparse attention method that accelerates the long-context prefilling stage of LLM with negligible loss in model accuracy. SALE achieves fast and accurate fine-grained attention map estimation using low-bit quantized query-key products to approximate attention weights, followed by the application of a novel *Relative Attention Score* metric to assess the importance of query-key pairs. This design enables us to accurately identify important regions in the attention map, thereby constructing a highly sparse attention mask.

We implement a custom CUDA kernel in SALE optimized for hardware efficiency, reducing overhead to approximately 11% of the full attention latency. Notably, SALE requires no parameter training and can be seamlessly integrated into existing systems with trivial code modifications. Experiments on long-context benchmarks demonstrate that our method outperforms existing approaches in accuracy-efficiency trade-offs, achieving at least $3.36\times$ speedups on Llama-3.1-8B for sequences longer than 64K while maintaining model quality.

1 INTRODUCTION

With the growing demand for ultra-long context understanding in complex applications such as long book summarization (Kryściński et al., 2022; Porwal et al., 2023; Chang et al., 2024), long document question-answering (Caciularu et al., 2023; Pang et al., 2022; Fan et al., 2019), and repository-level code completion (Wang et al., 2024a;b), state-of-the-art Large Language Models (LLM) are now equipped with increasingly longer context window (Grattafiori et al., 2024; Yang et al., 2025; Team et al., 2025; DeepSeek-AI et al., 2025). Most LLMs employ a decoder-only Transformer architecture (Vaswani et al., 2017), where the self-attention module serves as the core component to enable powerful language understanding capabilities. However, during the prefilling stage of LLM inference, the self-attention module exhibits quadratic time complexity with respect to the number of input tokens. This makes it the primary performance bottleneck, as computational costs increase rapidly with longer contexts (Fu, 2024; Jiang et al., 2024).

In recent years, numerous research studies have attempted to accelerate prefilling by computing only the important regions of attention maps, based on the observation that attention maps in LLMs are significantly sparse (Deng et al., 2024). These methods, referred to as *sparse attention*, use *sparse masks* to indicate the specific regions of the attention map to be computed. Some sparse attention methods utilize sparse masks with static patterns, such as stride pattern (Child et al., 2019), window pattern (Zaheer et al., 2020; Beltagy et al., 2020), or streaming pattern (Xiao et al., 2024b; Han et al., 2023). However, static sparse masks often result in severe performance degradation, as the real sparse patterns of LLM attention maps are highly dynamic across various input contents (Lai et al., 2025; Jiang et al., 2024).

To adapt to such dynamism, several methods attempt to predict critical attention regions by analyzing the attention map. For instance, MInference (Jiang et al., 2024) and SampleAttention (Zhu et al., 2024), decompose the sparse attention pattern into combinations of multiple vertical or slash lines, and predict the positions of these lines by analyzing the attention score distribution of a subset of query tokens. Another series of sparse attention methods, such as FlexPrefill (Lai et al., 2025), SpargeAttn (Zhang et al., 2025b), and HiP Attention (Lee et al., 2025), treat the attention map as the concatenation of blocks. They dynamically skip certain attention computations at the block granularity. These methods often construct a representative token for each consecutive query/key chunk and build sparse masks based on the product between representative tokens. Although existing dynamic sparse attention methods can accelerate the prefilling stage of LLM inference to some extent, they fail to achieve a satisfactory accuracy-efficiency trade-off, as their inspection approaches are either too coarse-grained or insufficiently comprehensive.

In this paper, we propose **SALE**, a novel block-Sparse Attention technique based on Low-bit Estimation of attention weights, to significantly accelerate the long-context prefilling stage of LLM inference with negligible loss in model accuracy. SALE is built on a fast and accurate framework for fine-grained attention map inspection. By performing element-wise importance analysis on the entire attention map, SALE is capable of constructing highly sparse attention masks, while ensuring that the output error is bounded within an acceptable tolerance range. We propose two key components to implement this framework. First, we utilize low-bit quantized query-key products (QK) to approximate attention weights. This process runs efficiently on modern GPUs, leveraging two key factors: the use of high-throughput low-bit Tensor Core instructions and the reduction in global memory access. Second, we propose a novel *Relative Attention Score* metric to evaluate the importance of query-key pairs. Observing that the attention scores in the sink (beginning) and local (end) regions of each attention map row tend to be relatively higher (Xiao et al., 2024b; Gu et al., 2025), we determine the importance of a query-key pair based on the relative magnitude between its attention weight and the attention weights within the sink and local regions. Compared to common practice that uses the original attention scores (Zhang et al., 2023; Li et al., 2024; Zhang et al., 2024a) as indicators, our design is more efficient because it avoids materializing the entire attention map in DRAM. It also supports adaptive adjustment of the sparsity level according to the input, which allows SALE to robustly handle input samples of varying complexities. SALE further adopts several kernel optimization techniques to optimize hardware efficiency.

SALE incurs no additional training overhead and can be seamlessly integrated into existing inference systems. We conduct comprehensive experiments on various long-context processing benchmarks using two LLMs, Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen-2.5-32B-Instruct (Yang et al., 2024), to verify the effectiveness of our method. Experimental results demonstrate that our method delivers a speed-up of at least $3.36\times$ when processing sequences longer than 64K tokens, while maintaining negligible accuracy loss. It achieves superior accuracy-efficiency trade-off compared to the baseline methods.

2 RELATED WORKS

Sparse LLM inference Many previous works try to leverage the sparsity nature of transformer model to accelerate LLM inference from different perspectives. One line of research exploits input text sparsity to dynamically prune context irrelevant to the user’s query (Jha et al., 2024; Liu et al., 2025; Shi et al., 2024; Jiang et al., 2023; Li et al., 2023b). While these methods can significantly reduce LLM inference latency for relatively simple prompts, they severely degrade generation quality when processing complex inputs (Yuan et al., 2024).

Numerous studies have observed sparsity patterns in self-attention modules, where only a small subset of attention map elements are much larger than the rest. Some methods (Xiao et al., 2024b; Fu et al., 2024; Xiao et al., 2024a) use predefined static sparsity patterns to prune the attention map. However, these methods suffer from accuracy degradation as the attention sparsity distribution varies among different input contexts (Jiang et al., 2024; Lai et al., 2025). Other methods assume that the distribution follows certain structures, such as Vertical-Slash or Block-Sparse. Some of them (Jiang et al., 2024; Zhu et al., 2024) try to dynamically predict the location of important regions by examining the exact attention scores of several tokens. Others (Gao et al., 2024; Zhang et al., 2025b; Lai et al., 2025; Lee et al., 2025) regard the attention map of compressed tokens, which are

generated from continuous token chunks, as the proxy of real attention map. All these methods fail to achieve accurate predictions due to their overly coarse-grained approximations of attention maps.

In contrast to the aforementioned approaches, several alternatives to self-attention have emerged to circumvent its quadratic complexity. Notable examples include: (1) natively sparse attention algorithms (Yuan et al., 2025; Lu et al., 2025), (2) linear attention mechanisms (Peng et al., 2023; Yang et al., 2023), and (3) state-space models (Gu & Dao, 2024; Dao & Gu, 2024). However, these methods impose significant adoption costs as they require full model training.

During the decoding stage, methods like SparQ (Ribar et al., 2023) and InfiniGen (Lee et al., 2024) compress the channels of query / key tokens to efficiently approximate the attention scores. Retrieval-based approaches (Zhang et al., 2024a; Chen et al., 2025; Liu et al., 2024) leverage vector-retrieval techniques to approximately sort the attention scores of previous input tokens. Several existing algorithms compress tokens by analyzing attention maps during the prefilling stage. These approaches either eliminate redundant tokens (Zhang et al., 2023; Liu et al., 2023; Li et al., 2024; Ge et al., 2023; Devoto et al., 2024) or perform token merging (Zhang et al., 2024d; Zandieh et al., 2024). Our method is orthogonal to these optimizations and can be combined to further enhance end-to-end LLM inference efficiency.

Attention kernel optimization Many CUDA kernel optimization techniques (Dao et al., 2022; Dao, 2023; Shah et al., 2024; Sanovar et al., 2024) leverage hardware features to accelerate the computation of the original full attention. Although these methods accelerate computation, they still require full attention calculations and fail to fully exploit the inherent sparsity of attention maps.

3 METHOD

3.1 PROBLEM FORMULATION

We denote the query, key and value matrices as Q , K and V , respectively, while the corresponding vectors at token offset i are q_i, k_i, v_i . Let N represent the sequence length and d represent the hidden size. The shapes of Q , K and V are all $N \times d$. Matrix M is the sparse attention mask with a shape of $N \times N$. Single-head self-attention module can be mathematically formalized as below:

$$\text{Attn}(Q, K, V, M) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}} + M\right) \cdot V \quad (1)$$

During the computation of self-attention, the attention weight matrix S is defined as $S = QK^T / \sqrt{d}$, and the attention score matrix P is defined as $P = \text{Softmax}(S + M)$. The sparse attention mask is formed by $M = M_c + M_s$, where $M_c, M_s \in \{0, -\infty\}$ represent the causal mask and the sparse mask respectively. Based on the mathematical properties of Softmax function, if an item $M[i, j]$ in matrix M is $-\infty$, its corresponding attention score will be zero. Therefore, we can skip the attention computation at this position.

For block-sparse attention, query and key tokens are divided into continuous blocks of sizes b_q, b_k along the sequence dimension. We denote the query, key token block at position j as Q_j, K_j , which have the shapes of $b_q \times d$ and $b_k \times d$ respectively. For simplicity, we assume $b_q \mid N, b_k \mid N$, and denote $N_q = N/b_q, N_k = N/b_k$. As shown in Figure 1(a), the attention map can be viewed as the concatenation of $N_q \cdot N_k$ attention blocks, each of shape $b_q \times b_k$. Block sparse attention skips certain computations at the block level. To formulate, we denote $M_{bs} \in \{0, 1\}$ as *block-level sparse mask*, and values of sparse mask M_s depend on M_{bs} :

$$M_s[i, j] = \begin{cases} 0, & \text{if } M_{bs}[\lfloor i/b_q \rfloor, \lfloor j/b_k \rfloor] = 1, \\ -\infty, & \text{if } M_{bs}[\lfloor i/b_q \rfloor, \lfloor j/b_k \rfloor] = 0 \end{cases} \quad (2)$$

In other words, the attention computation between Q_i, K_j, V_j will be skipped if $M_{bs}[i, j]$ is zero. Block-sparse attention aims to maximize sparsity in matrix M_{bs} while bounding the approximation error relative to full attention within a tolerable threshold.

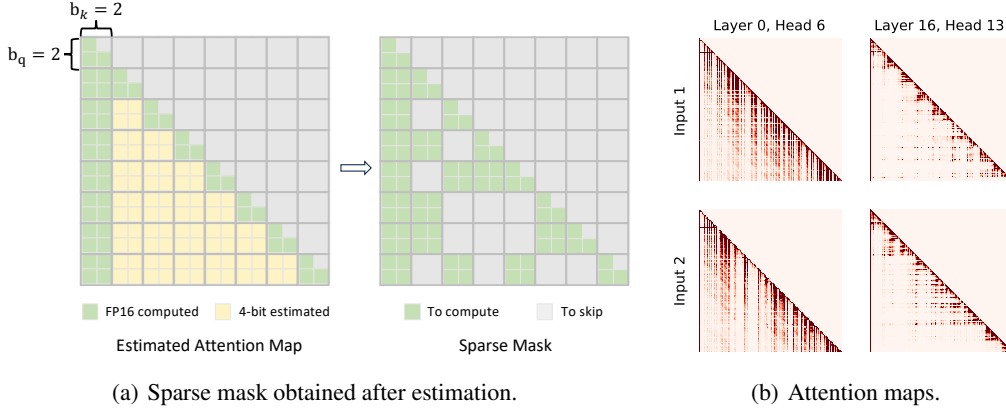


Figure 1: (a) Illustration of SALE. The whole 16×16 attention map is viewed as concatenation of many 2×2 blocks. We first estimate the attention weights in an element-wise manner, and then construct a sparse mask at the block level based on these estimations. (b) Attention maps of two different attention heads in Llama-3.1-8B-Instruct when processing different input sequences.

3.2 BLOCK SELECTION VIA FINE-GRAINED IMPORTANCE APPROXIMATION

Considering the dynamic nature of the attention pattern in LLMs (Jiang et al., 2024; Lai et al., 2025), constructing "Sparse and Accurate" M_{bs} is highly challenging. To achieve this goal, SALE evaluates the "importance" of each position in the attention map. Such a framework enables SALE to examine the attention map at a fine-grained level, allowing robust handling of various dynamic attention patterns. We propose two key designs—*Low-bit Attention Weight Approximation* and *Relative Importance Approximation*—to ensure the efficient and accurate operation of SALE.

Low-bit attention weight approximation SALE first calculates the attention weights for all query-key pairs. Rather than using full-precision floating-point Q and K matrices, SALE computes attention weights using low-bit quantized versions $\tilde{Q}$ and $\tilde{K}$ for approximation, significantly minimizing additional overhead with high-throughput low-bit Tensor Core instructions and reduced DRAM access. For instance, on RTX 4090 and RTX 5090 GPUs, the throughput of 4-bit and 8-bit Tensor Core instructions is $8\times$ and $4\times$ that of 16-bit (full precision) Tensor Core instructions, respectively (NVIDIA, 2023; 2025). Meanwhile, the overhead of loading 4-bit and 8-bit QK matrices from DRAM is $1/4$ and $1/2$ of that required for loading 16-bit QK matrices, respectively.

Regarding the selection of the quantization scheme, we choose to quantize query-key matrices to 4-bit, which maintains high selection accuracy while maximizing the acceleration capability of the low-bit instructions in GPU Tensor Cores. We provide more detailed discussion about quantization scheme in section H.

Relative importance approximation Denoting approximated attention weights as $\tilde{S}$, the next step is to evaluate the "importance" of each attention block. In related works (Zhang et al., 2023; Li et al., 2024; Liu et al., 2025; 2023), a commonly used metric is the attention score, obtained by applying *Softmax* function to attention weights. Compared to existing practices, our approach performs importance computation on-the-fly, bypassing the need to compute full attention scores. We propose *Relative Attention Score* as our importance metric instead of the original attention scores.

Our design is based on an observation in many related studies (Xiao et al., 2024b; Gu et al., 2025; Xiao et al., 2024a). As shown in Figure 1(b), attention scores within the "sink and local regions" (i.e. the beginning and end of each row) maintain consistently large values, while the region exhibits consistent size across diverse input sequences. Motivated by this pattern, we assess "importance" by comparing $\tilde{S}[i, j]$ with the attention weights from the sink and local regions. Specifically, before examining those blocks located in the middle of the sequence, we first compute full precision QK matrix multiplication on blocks in the sink and local areas. Denoting the indices of key tokens within the sink and local regions as I_{SL} , this process yields two intermediate values, $\tilde{m}_i$ and $\tilde{l}_i$. We then compute the *Relative Attention Score* $\tilde{P}[i, j]$ base on these two intermediate results, which can be

formulated as follows:

$$\tilde{m}_i = \max_{j \in I_{SL}} S[i, j], \quad \tilde{l}_i = \sum_{j \in I_{SL}} e^{S[i, j] - \tilde{m}_i}, \quad \tilde{P}[i, j] = \frac{e^{\tilde{S}[i, j] - \tilde{m}_i}}{\tilde{l}_i}$$

If all $\tilde{P}[i, j]$ values in a block are smaller than the threshold τ (e.g. 0.004), this block is marked as non-critical, and the attention computation for this block will be skipped. The procedure for determining the threshold value $\tau \in (0, 1)$ is elaborated in Section 3.3.

3.3 PER-HEAD THRESHOLD CALIBRATION

Figure 1(b) illustrates the attention score distributions of two attention heads of Llama-3.1-8B-Instruct, exhibiting inconsistent sparsity levels. Thus, applying the same τ for all heads may lead to suboptimal performance. To address the issue, we propose an offline calibration procedure to determine the optimal τ value for each head, which ensures negligible output errors while maximizing sparsity.

We adopt the L_1 distance between the output of SALE and the output of full attention as the error metric, which can be formulated as $Err(\tau) = \|O - \tilde{O}\|_1 / N$. O is the result of the original attention, $\tilde{O}$ is the result of SALE, and N represents sequence length. At the beginning of the calibration, τ is initially set to be a relatively large threshold τ_0 (e.g. 0.008). We then progressively reduce the sparsity level by halving the value of τ until $Err(\tau)$ falls below θ , where θ is the predefined error bound. By tuning θ , we can control the sparsity level of SALE. In Section F, we provide a detailed explanation of the correlations between these hyperparameters.

3.4 KERNEL OPTIMIZATION

For kernel implementation, we propose several optimization techniques to further improve the hardware efficiency of SALE. Here, we describe some core designs, and more detailed information about the optimizations can be found in Section G.

Computation tiling SALE targets long context processing scenarios, where materializing the intermediate results of QK GEMM would occupy a large amount of DRAM space and introduce extensive I/O overhead. Therefore, we implement SALE based on the framework of FlashAttention2 (Dao, 2023) to avoid this issue. Specifically, the computation tasks of different query blocks will be scheduled to different cooperative thread arrays (CTA) for parallel processing. Each CTA iteratively checks the importance between its corresponding query block and different key blocks.

Estimation-computation disaggregation In consideration of performance issues, SALE separates "important block selection" and "sparse attention computation" into two distinct CUDA kernels, which are referred to as *Selection-Pass* and *Computation-Pass*, respectively. During Selection-Pass, we select important attention regions at the block granularity and record the coordinates of these blocks. We provide the corresponding pseudo code in Section J for a more detailed description. After Selection-Pass, we compute attention output on selected blocks in the following Computation-Pass.

A potentially simpler and more efficient implementation is the "One-pass" alternative: upon identifying an important key block, immediately compute the output for that block. However, such a design is not feasible in practice, as it fails to reduce computational workload on the one hand and is incompatible with the hardware characteristics of GPUs on the other. We provide a more detailed discussion on this issue in Section I.

Relative attention score comparison Directly computing *Relative Attention Score* is time-consuming as it consists of multiple complex hardware instructions, including floating point division and exponential function. Considering that $\tilde{l}_i$ and $\tilde{m}_i$ do not change after computation in sink and local areas, we optimize this comparison by following mathematical transformation:

$$\frac{e^{\tilde{S}[i, j] - \tilde{m}_i}}{\tilde{l}_i} \geq \tau \iff \tilde{S}[i, j] \geq \ln(\tau \cdot \tilde{l}_i + \tilde{m}_i) \quad (3)$$

The comparison between the *Relative Attention Score* and τ can then be accomplished using a single floating point comparison instruction. It is worth noting that we also mitigate potential overflow issues caused by the exponential function.

Integration with SageAttention The final stage of SALE is Computation-Pass. We employ the QK 8-bit quantization strategy proposed in SageAttention (Zhang et al., 2025a) to further accelerate Computation-Pass while maintaining negligible precision loss.

4 EXPERIMENTS

4.1 SETTINGS

Models Most of the experiments are conducted using Llama-3.1-8B-Instruct (Grattafiori et al., 2024) (**Llama-3.1**). We also use Qwen2.5-32B-Instruct (Yang et al., 2024) (**Qwen-2.5**) to validate the effectiveness of our method on larger-scale LLM. Both models support a context length of 128K.

Implementation details We implement Selection-Pass in C++ CUDA and use Triton (Tillet et al., 2019) compiler to accelerate the quantization process. We implement the quantized Computation-Pass based on the open-source code of SpargeAttn (Zhang et al., 2025b). For those hyper-parameters mentioned in Section 3.2, we use block size $b_q = 64$ and $b_k = 32$. We constrain the sink area size to 32 tokens and the local area size to no more than 256 tokens. During offline calibration, we set the initial threshold $\tau_0 = 0.008$, and use error bounds of $\theta = 0.4$ for Llama-3.1 and $\theta = 2.0$ for Qwen-2.5 by default. All latency experiments are conducted on a server with 8 GeForce RTX 4090 GPUs without using tensor-parallel (Shoeybi et al., 2019) or context-parallel (Li et al., 2023a) techniques. In the Section B, we provide additional information about the implementation.

Baselines To demonstrate the advantages of SALE, we compare it with four strong baselines for self-attention acceleration in long-context processing: **FlashAttention2**(FA2) (Dao, 2023), **Minference**(Minfer) (Jiang et al., 2024), **FlexPrefill**(Flex) (Lai et al., 2025), and **SparseAttn**(Sparge) (Zhang et al., 2025b). FA2 computes standard full attention, which serves as an oracle. The other three methods employ sparse attention mechanisms. All experimental results are based on their publicly available implementation. We use $\gamma = 0.95$ for Llama-3.1 and $\gamma = 0.98$ for Qwen-2.5 when evaluating FlexPrefill. We use $(l_1 = 0.08, l_2 = 0.09)$ for Llama-3.1, and $(l_1 = 0.04, l_2 = 0.05)$ for Qwen-2.5 when evaluating SpargeAttn. For Minference, we select the optimal sparse pattern based on its open-source code. Additionally, to investigate the performance of these methods under varying sparsity levels, we prepare multiple sets of hyperparameters based on their publicly available codes. For FlexPrefill and SpargeAttn, as described in their papers, we adjust their sparsity levels by tuning γ and (l_1, l_2) , respectively. For Minference, since its open-source implementation configures all heads with the *Vertical-Slash* pattern, the sparsity rate is adjusted by varying the total number of vertical and slash lines across all heads. We provide more details in Section C.

Metrics To validate the effectiveness of SALE, we assess model quality using long-context benchmarks (see Section 4.2) and measure efficiency through latency measurements. All latency results in the experimental section focus solely on the attention computation time across all layers during the LLM prefilling phase. Our latency measurements include all online operations, such as quantization, block selection, and index selection. In some experiments, we report the end-to-end (E2E) latency on certain datasets, which is computed by summing the latency of all samples in the dataset.

4.2 ACCURACY EVALUATION

Following common practice (Zhang et al., 2025b; Jiang et al., 2024; Lai et al., 2025; Zhang et al., 2024a; Gao et al., 2024; Li et al., 2024), we adopt three long-context understanding benchmarks to compare the generation quality of our method with other baselines. Achieving higher scores on these baselines indicates better performance. (1) **LongBench** (Bai et al., 2024): A comprehensive benchmark covering diverse long-text applications, including single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, etc. The context lengths of most input samples are below 32K tokens. (2) **InfiniteBench** (Zhang et al., 2024c): A benchmark designed to evaluate the capability of processing excessively long context (exceeding 100K tokens). It comprises several challenging synthetic tasks such as Retrieve.KV and Math.Find, as well as other real-world tasks including QA and summarization based on fake books or fake dialogues. (3) **Needle-In-A-Haystack** (Kamradt, 2023): A widely-used long-context retrieval task. It requires the LLM to locate a randomly inserted sentence at various positions within a real-world context.

Table 1: LongBench evaluation results of different methods. We use **boldface** to denote the highest value and underline to indicate the second-highest value.

Tasks	Llama-3.1					Qwen-2.5				
	FA2	Minfer	Flex	Sparge	SALE	FA2	Minfer	Flex	Sparge	SALE
NarrativeQA	29.93	24.92	28.29	<u>29.62</u>	28.95	29.20	<u>31.27</u>	29.80	29.19	32.21
Qasper	<u>44.82</u>	44.29	44.55	43.73	45.33	<u>45.79</u>	45.05	45.53	44.61	45.95
MultiFieldQA	54.65	53.71	<u>55.34</u>	56.02	55.18	<u>53.25</u>	53.01	52.61	51.66	53.37
HotpotQA	<u>55.81</u>	52.00	55.38	54.57	55.83	<u>64.68</u>	64.59	64.78	63.94	63.95
2WikiMQA	<u>46.16</u>	44.10	43.43	47.08	42.61	60.87	60.82	62.98	61.13	<u>62.33</u>
MuSiQue	<u>30.41</u>	25.72	30.07	31.40	30.10	39.89	41.38	39.46	39.22	<u>40.54</u>
GovReport	<u>35.29</u>	35.09	34.64	35.04	35.45	30.38	30.59	30.78	30.36	<u>30.66</u>
QMSum	25.25	<u>25.47</u>	25.83	25.12	25.33	23.06	23.16	23.10	<u>23.18</u>	23.42
TREC	72.50	<u>72.00</u>	70.50	71.00	70.50	<u>73.50</u>	<u>73.50</u>	74.50	<u>73.50</u>	73.00
TriviaQA	<u>91.65</u>	91.18	89.81	92.68	90.47	87.68	88.40	89.40	<u>88.81</u>	87.97
SAMSum	43.67	<u>43.73</u>	43.18	43.18	44.19	45.67	45.92	46.43	<u>46.41</u>	45.92
LSHT	46.50	<u>46.00</u>	41.00	45.50	46.50	45.79	47.50	44.17	47.00	<u>47.21</u>
Count	<u>6.72</u>	3.25	2.59	5.89	7.09	12.67	13.67	3.57	9.22	<u>13.38</u>
Retrieval	<u>99.50</u>	97.00	82.00	84.00	100.00	99.50	<u>99.25</u>	92.25	98.83	98.25
Average	48.77	47.03	46.18	47.48	<u>48.39</u>	50.85	<u>51.29</u>	49.88	50.57	51.30
Speedup (64K)	1.00×	1.07×	2.21×	<u>3.11×</u>	3.36×	1.00×	1.25×	1.39×	<u>2.55×</u>	3.28×

LongBench Table 1 presents the LongBench evaluation results comparing SALE with baseline approaches. In the second row of the table, we use abbreviations introduced in Section 4.1 to denote each method. In the last two rows, we report the average scores and the latency speedup. The latency metric, measured on a 64K-length Needle-In-A-Haystack input, confirms the premise of our accuracy test, that SALE achieves the highest speedup when processing long contexts.

The results on both models show that SALE achieves superior accuracy among all sparse attention baselines. In addition, when our method is applied, Llama-3.1 exhibits only marginal performance degradation while Qwen-2.5 even shows improvement. We attribute this improvement to our method’s ability to potentially filter noisy information during the prefilling phase, thereby enhancing the model’s comprehension capabilities.

InfiniteBench Table 2 presents the test scores of InfiniteBench, evaluating the capability of processing extremely long inputs. As shown in the table, our method also achieves the best accuracy-efficiency trade-off on InfiniteBench.

Needle-In-A-Haystack We evaluate the Needle-In-A-Haystack (NIAH) task using Llama-3.1, with results visualized in Figure 2. The average score and end-to-end speedup for each method are annotated above their respective plots. Our method achieves a $3.81\times$ speedup with only a 0.1% drop in average score compared to FlashAttention2, outperforming all other sparse attention baselines.

4.3 EFFICIENCY EVALUATION

Single input speedup We first compare the latency of different methods when processing a single input. The results are presented in Figure 3(a). We conduct experiments using Llama-3.1 and report

Table 2: InfiniteBench evaluation results of different methods. We use **boldface** to denote the highest value and underline to indicate the second-highest value.

Tasks	Llama-3.1					Qwen-2.5				
	FA2	Minfer	Flex	Sparge	SALE	FA2	Minfer	Flex	Sparge	SALE
Retrieve.KV	<u>55.60</u>	20.00	38.00	47.20	56.40	4.00	7.60	4.80	4.60	<u>5.40</u>
En.MC	<u>67.25</u>	55.02	68.56	66.38	66.38	63.70	<u>63.32</u>	59.80	61.50	62.80
Math.Find	34.29	34.86	30.00	<u>34.57</u>	30.57	41.40	45.71	<u>47.20</u>	41.70	52.00
En.QA	15.12	13.96	<u>14.18</u>	13.51	13.19	6.70	<u>6.85</u>	6.70	6.80	6.90
En.Dia	16.50	13.50	17.00	<u>17.50</u>	19.00	27.50	30.00	<u>29.00</u>	25.50	27.50
Average	37.75	27.47	33.55	35.83	<u>37.11</u>	28.66	<u>30.70</u>	29.5	28.02	30.92
Speedup (64K)	1.00×	1.07×	2.21×	<u>3.11×</u>	3.36×	1.00×	1.25×	1.39×	<u>2.55×</u>	3.28×

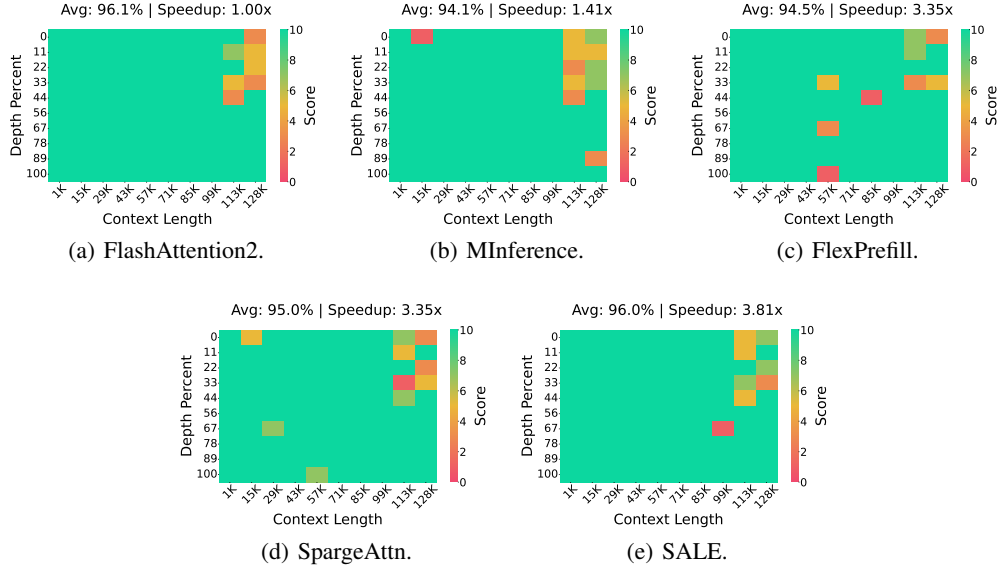


Figure 2: Needle-In-A-Haystack evaluation results.

the speedup of each method relative to FlashAttention2. To illustrate how latency scales with the number of tokens, we prepare five input samples of different lengths. These samples are obtained by truncating a single 128K-length input from the Needle-In-A-Haystack task.

Our method demonstrates consistent speedups over FlashAttention2 across all sequence lengths while outperforming all sparse attention baselines in most cases. At 8K context length, SALE achieves a $1.39\times$ speedup. It further exhibits greater speedup as context length increases—reaching $3.88\times$ at 128K—owing to sparser attention patterns. It should be noted that the hyperparameters of each sparse method in this test are aligned with those in Section 4.2, indicating that our speedup evaluation is based on SALE achieving optimal accuracy performance.

Accuracy vs efficiency We adjust the computation budget of each method following the approach described in Section 4.1 to analyze the accuracy-efficiency trade-offs. Considering that the speedup achieved by dynamic sparse attention methods may vary depending on the input content, we evaluate the end-to-end latency of all methods on both LongBench and InfiniteBench for comprehensive comparison. The results, shown in Figure 4, demonstrate the superior performance of our method on both datasets.

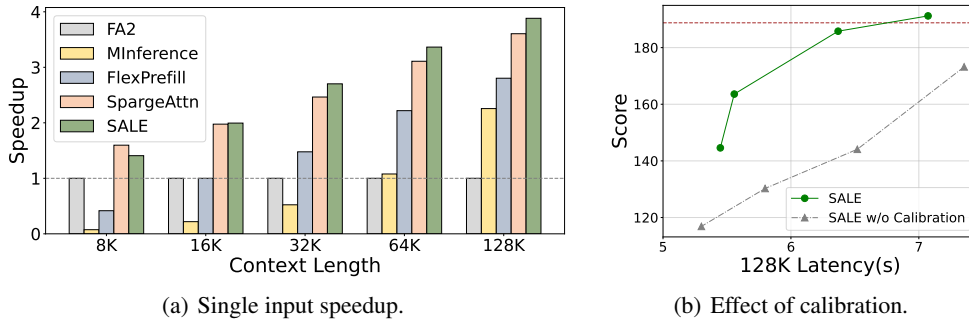


Figure 3: (a) Speedup in single-input processing. (b) Comparison between SALE v.s. SALE w/o Calibration on InfiniteBench. The brown horizontal dashed line represents the score achieved by FlashAttention2.

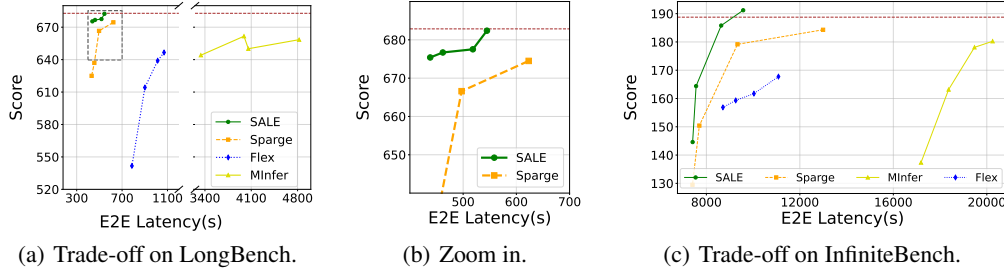


Figure 4: Evaluation of accuracy-efficiency trade-offs. The brown horizontal dashed line represents the score achieved by FlashAttention2. (a) Performance on LongBench under different sparsity levels. (b) A magnified view focusing on the region enclosed by the dashed box in (a). (c) Performance on InfiniteBench under different sparsity levels.

4.4 ABLATION STUDY

In this section, we evaluate the latency of each stage in SALE and assess the impact of per-head threshold calibration. Additionally, in Section D and Section H, we provide additional experimental results to demonstrate the block selection accuracy of SALE. We also present sparsity measurement results of these sparse methods across contexts of varying lengths in Section E, which serves as a supplement to our previous comparison.

Latency breakdown We report the latency breakdown results of SALE under various input lengths in Table 3. All experiments use Llama-3.1, with reported timings reflecting end-to-end execution across all 32 model layers. In the second-to-last line, we show the execution time ratio of Quantization and Selection-Pass operations relative to full attention latency. In the final line, we present the speedup of Computation-Pass compared to full attention. The results demonstrate that our method introduces acceptable computational overhead, with its relative cost decreasing as sequence length grows. Furthermore, Computation-Pass shows greater speedups with longer context lengths, reflecting improved sparsity level at scale.

Table 3: Latency breakdown (ms).

Context length	8K	16K	32K	64K	128K
Quantization	11	21	47	99	208
Selection-Pass	14	48	166	634	2562
Computation-Pass	51	137	378	1117	3599
FA2	106	416	1597	6224	24731
Overhead ratio	23.9%	16.7%	13.3%	11.5%	11.1%
Computation-Pass speedup	2.08×	3.04×	4.23×	5.57×	6.87×

Threshold calibration To demonstrate the performance gain brought by per-head threshold calibration, we set all heads in Llama-3.1 to share the same τ , which is referred to as *SALE w/o Calibration*. As shown in Figure 3(b), setting different τ values according to the characteristics of attention patterns across different heads yields substantial performance gains.

5 CONCLUSION

In this paper, we propose a block-Sparse Attention technique based on Low-bit Estimation. By performing fine-grained estimation of the attention map, we achieve a better accuracy-efficiency trade-off. Specifically, we estimate the attention weights using low-bit quantized queries and keys, and assess the importance of query-key pairs using our *Relative Attention Score* metric. Furthermore, we introduce several CUDA kernel optimization techniques to ensure the efficiency of sparse mask construction on hardware. Experimental results demonstrate that our approach achieves the best trade-off among existing sparse attention baselines, delivering a speedup of at least $3.36\times$ when processing sequences longer than 64K tokens while maintaining negligible accuracy loss.

6 ETHICS STATEMENT

Our submission does not violate ICLR Code of Ethics.

7 REPRODUCIBILITY STATEMENT

In Section 4, Section B and Section C, we provide details regarding the experiments as well as the implementation of our method.

We also upload our code to <https://anonymous.4open.science/r/SALE-7209> for experiments result reproducibility.

REFERENCES

- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL <https://aclanthology.org/2024.acl-long.172/>.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- Avi Caciularu, Matthew Peters, Jacob Goldberger, Ido Dagan, and Arman Cohan. Peek across: Improving multi-document modeling via cross-document question-answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1970–1989, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.110. URL <https://aclanthology.org/2023.acl-long.110/>.
- Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. Boookscore: A systematic exploration of book-length summarization in the era of LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=7Ttk3RzDeu>.
- Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=ALzTQUgW8a>.
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.
- Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- Tri Dao and Albert Gu. Transformers are ssms: generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.
- Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song,

- Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL <https://arxiv.org/abs/2412.19437>.
- Yichuan Deng, Zhao Song, and Chiwun Yang. Attention is naturally sparse with gaussian distributed input. *CoRR*, abs/2404.02690, 2024. doi: 10.48550/ARXIV.2404.02690. URL <https://doi.org/10.48550/arXiv.2404.02690>.
- Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024.
- Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering, 2019. URL <https://arxiv.org/abs/1907.09190>.
- Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. Moa: Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*, 2024.
- Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis, 2024. URL <https://arxiv.org/abs/2405.08944>.
- Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.
- Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.
- 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, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelder van der Linde,

Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Rapparthi, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collet, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpiere Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippas Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro

- Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=tEYskw1VY2>.
- Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=78Nn4QJTEN>.
- Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. *arXiv preprint arXiv:2308.16137*, 2023.
- Siddharth Jha, Lutfi Eren Erdogan, Sehoon Kim, Kurt Keutzer, and Amir Gholami. Characterizing prompt compression methods for long context inference. *arXiv preprint arXiv:2407.08892*, 2024.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LlmLingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.
- Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.
- Greg Kamradt. Needle in a haystack - pressure testing llms, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack?tab=readme-ov-file.
- Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization, 2022. URL <https://arxiv.org/abs/2105.08209>.
- Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=OfjIlbelrT>.
- Heejun Lee, Geon Park, Youngwan Lee, Jaduk Suh, Jina Kim, Wonyong Jeong, Bumsik Kim, Hyemin Lee, Myeongjae Jeon, and Sung Ju Hwang. A training-free sub-quadratic cost transformer model serving framework with hierarchically pruned attention. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=PTcMzQgKmn>.

- 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, 2024.
- Dacheng Li, Rulin Shao, Anze Xie, Eric P Xing, Xuezhe Ma, Ion Stoica, Joseph E Gonzalez, and Hao Zhang. Distflashattn: Distributed memory-efficient attention for long-context llms training. *arXiv preprint arXiv:2310.03294*, 2023a.
- Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*, 2023b.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.
- 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.
- Jingyu Liu, Beidi Chen, and Ce Zhang. Speculative prefill: Turbocharging ttft with lightweight and training-free token importance estimation. *arXiv preprint arXiv:2502.02789*, 2025.
- Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.
- Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.
- NVIDIA. Nvidia ada gpu architecture. Technical report, NVIDIA, 2023. URL <https://images.nvidia.cn/aem-dam/Solutions/geforce/ada/nvidia-ada-gpu-architecture.pdf>.
- NVIDIA. Nvidia rtx blackwell gpu architecture. Technical report, NVIDIA, 2025. URL <https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf>.
- Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. Quality: Question answering with long input texts, yes!, 2022. URL <https://arxiv.org/abs/2112.08608>.
- Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.
- Siddhant Porwal, Laxmi Bewoor, and Vivek Deshpande. Transformer based implementation for automatic book summarization. *arXiv preprint arXiv:2301.07057*, 2023.
- Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*, 2023.
- Rya Sanovar, Srikant Bharadwaj, Renee St Amant, Victor Rühle, and Saravan Rajmohan. Lean attention: Hardware-aware scalable attention mechanism for the decode-phase of transformers. *arXiv preprint arXiv:2405.10480*, 2024.
- Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

- Zhenmei Shi, Yifei Ming, Xuan-Phi Nguyen, Yingyu Liang, and Shafiq Joty. Discovering the gems in early layers: Accelerating long-context llms with 1000x input token reduction. *arXiv preprint arXiv:2409.17422*, 2024.
- Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Naveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Pettrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Naveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pöder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL <https://arxiv.org/abs/2503.19786>.
- Philippe Tillet, H. T. Kung, and David Cox. 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, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL <https://doi.org/10.1145/3315508.3329973>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Chong Wang, Jian Zhang, Yebo Feng, Tianlin Li, Weisong Sun, Yang Liu, and Xin Peng. Teaching code llms to use autocompletion tools in repository-level code generation, 2024a. URL <https://arxiv.org/abs/2401.06391>.

- Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. RlCoder: Reinforcement learning for repository-level code completion, 2024b. URL <https://arxiv.org/abs/2407.19487>.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- 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*, 2024a.
- 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*, 2024b. URL <https://openreview.net/forum?id=NG7sS51zVF>.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, et al. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.
- Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.
- Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhuo Xu, et al. Kv cache compression, but what must we give in return? a comprehensive benchmark of long context capable approaches. *arXiv preprint arXiv:2407.01527*, 2024.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.
- Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.
- Amir Zandieh, Insu Han, Vahab Mirrokni, and Amin Karbasi. Subgen: Token generation in sublinear time and memory. *arXiv preprint arXiv:2402.06082*, 2024.
- Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *arXiv preprint arXiv:2407.12820*, 2024a.
- Jintao Zhang, Haofeng Huang, Pengl Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2 technical report: Accurate 4 bit attention for plug-and-play inference acceleration. *arXiv preprint arXiv:2411.10958*, 2024b.
- Jintao Zhang, Jia wei, Pengl Zhang, Jun Zhu, and Jianfei Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL <https://openreview.net/forum?id=OL44KtasKc>.
- Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025b.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞ bench: Extending long context evaluation beyond 100k tokens, 2024c. URL <https://arxiv.org/abs/2402.13718>.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first International Conference on Machine Learning*, 2024d.

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.

Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, et al. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *arXiv preprint arXiv:2406.15486*, 2024.

A LLM USAGE

During the writing process of this paper, we used LLMs to assist in polishing the writing and employed AI tools to support the visualization of experimental results.

B ADDITIONAL IMPLEMENTATION DETAILS

For the quantization algorithm scheme, we use the smoothing technique mentioned in SageAttention2 (Zhang et al., 2024b) to improve the quality of QK quantization, which introduce negligible overhead. While the quality of quantization affects the accuracy performance of SALE, it should be noted that SALE is designed to be orthogonal to the quantization algorithm.

We select five input samples from the Retrieve.KV task in InfiniteBench to perform calibration for SALE, and the final configuration must satisfy the error bound requirement across all five samples. The per-head threshold calibration for Llama-3.1 on RTX4090 server takes approximately five minutes to complete.

C ADDITIONAL EXPERIMENT DETAILS

For model inference, we leverage the transformers (Wolf et al., 2020) library to build an execution pipeline and replace the default self-attention module with sparse methods. We use greedy decoding to avoid randomness during generation, and use the default chat template to construct the input prompt. We use the same set of input samples as our method to search the optimal hyperparameters for SparseAttention. Since the open-source code of MInference only uses one input sample for calibration, we employed the first sample from this set for its sparse pattern searching.

During evaluation process, to ensure proper model behavior, we truncate samples that exceed the maximum context window length. Following common practice, we retain the tokens from both the beginning and the end of the sequence and remove those from the middle portion. For all these benchmarks and tasks, we employ the official evaluation scripts from their respective open-source repositories to assess model outputs.

For the data format during model inference, we employed BFloat16 for FlexPrefill due to requirements specified in its repository, while Float16 was used for all other methods.

D 4BIT SELECTION-PASS VS 16BIT SELECTION PASS

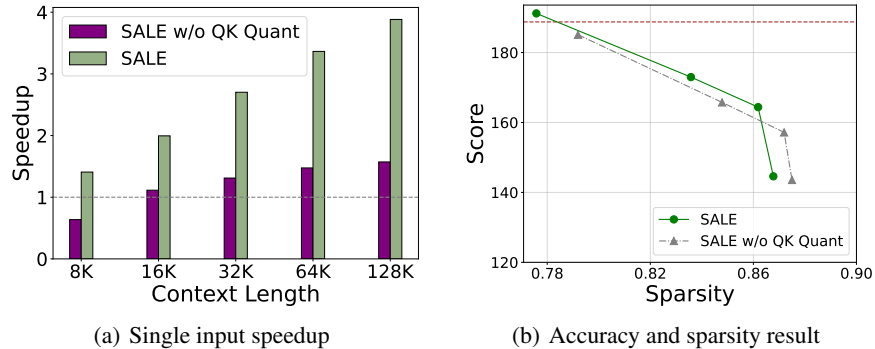


Figure 5: Comparison between SALE and SALE w/o QK Quant. (a)Single input speedup. (b) Comparison between SALE v.s. SALE w/o QK Quant on InfiniteBench. The brown horizontal dashed line represents the score achieved by FlashAttention2.

To evaluate the effectiveness of 4-bit attention weight approximation, we further conducted experiments using original-precision (16-bit) QK matrices to inspect the attention map, which is referred to as *SALE w/o QK Quant*. The result is shown in Figure 5. We measure the single input speedup of two

methods under varying input lengths, using the same set of input samples as in the Figure 3(a). The result indicates that using original-precision QK to estimate attention weights leads to a significant increase in computational overhead.

We further evaluate the accuracy and attention sparsity of both methods based on Llama-3.1, where corresponding data points for the two methods are obtained using the same θ . We use the scores from InfiniteBench to represent accuracy. Attention sparsity metric is defined as the ratio of the number of skipped attention blocks to the total number of attention blocks, and the results presented here are measured when processing contexts of 128K length. As observed, under identical hyperparameter settings, *SALE w/o QK Quant* achieves higher attention sparsity while showing a slight performance drop on InfiniteBench. This may be attributed to the limited precision of current Int4 quantization techniques, which can cause certain approximated attention weights to exceed their true values, thereby leading to more blocks being selected.

E SPARSITY STATISTIC ANALYSIS

To enable a more transparent comparison between SALE and other sparse attention baselines introduced in the Section 4, we tested the sparsity results of these methods on the input samples from LongBench. We divided all inputs into three groups according to their context lengths and computed the average sparsity rate along with its standard deviation for each group during processing. The hyperparameters used for each sparse method here are consistent with those described in Section 4.1. The results are in Table 4, with data in each cell formatted as (mean, std):

Table 4: Sparsity comparison

Sparse methods	MInfer	Flex	Sparge	SALE
0-20K	(0.138, 0.029)	(0.604, 0.035)	(0.184, 0.016)	(0.517, 0.028)
20K-40K	(0.356, 0.043)	(0.676, 0.046)	(0.279, 0.022)	(0.651, 0.028)
> 40K	(0.471, 0.044)	(0.677, 0.026)	(0.339, 0.025)	(0.678, 0.020)

The experimental results show that our scheme has obvious advantages in terms of sparsity rate compared with MInference and SpargeAttn. When compared with FlexPrefill, our sparsity rate is lower for context length $\leq 40K$, but higher for context length $> 40K$.

It should be noted that a higher sparsity rate of FlexPrefill here does not mean better performance. According to the results in Figure 4(a), SALE can achieve both higher scores and faster end-to-end latency than FlexPrefill under various sparsity rates.

F HYPERPARAMETERS EXPLANATION

τ controls the sparsity rate of a specific attention head. When the relative attention score of an attention block is less than τ , this block will be ignored in the Computation Pass. Therefore, as τ increases, more blocks will be skipped, leading to a higher sparsity rate, decreased accuracy, and faster attention computation. The value of τ can vary across different heads, and its specific value is determined during the calibration process.

τ_0 represents the initial value of τ in calibration process for each attention head. It will gradually decrease during the calibration process until the output error requirement is met. Therefore, it is sufficient for τ_0 to have a relatively large initial value. τ_0 does not affect the speedup and accuracy performance of the model.

θ , on the other hand, is a hyperparameter that adjusts the global sparsity rate. When θ increases, the τ values for each head will also increase, and the sparsity level of SALE get higher. In our Figure 4, we have assessed the performance of SALE under various values of θ .

G ADDITIONAL KERNEL OPTIMIZATION TECHNIQUE

Reduction in dequantization operations Theoretically, whether an attention block is skipped only depends on the comparison between the largest *Relative Attention Score* with τ . By employing per-thread quantization strategy proposed in (Zhang et al., 2024b), we make all quantized attention weight elements held by each thread share the same quantization scale. This ensures that the largest *Relative Attention Score* and the largest approximated attention weight occur at the same position. Therefore, only the largest approximated attention weight needs to be dequantized, which saves many low-throughput operations such as datatype conversion.

Segment level all-reduce Due to the hardware characteristics of GPU Tensor Cores, the QK GEMM for an attention block is collectively executed by multiple threads within a GPU thread blocks with the output of the QK GEMM stored in a distributed manner across different threads. Since the elements held by each thread are invisible to other threads, we need to perform a CTA-wise all-reduce operation on the relative importance comparison results in each thread. This ensures that all threads within the thread blocks reach a unified judgment on the importance of the current block.

In fact, this all-reduce operation is costly: it introduces multiple `warp_shfl` instructions, `_syncthreads` instructions, and shared memory access instructions. Since the Selection-Pass does not require computing attention output, we employ specific optimization techniques to reduce the number of all-reduce operations. Specifically, each thread can temporarily store the relative importance of multiple consecutive blocks as multiple bits in local variables, and use a single all-reduce operation to achieve CTA-wise consistency. This allows us to save a significant number of all-reduce operations.

H QUANTIZATION SCHEME ANALYSIS

SALE uses quantized QK to estimate attention weights in Selection-Pass. Theoretically, the fewer the number of bits used in quantization, the faster the estimation of attention weights will be, as this allows the use of faster Tensor Core instructions and results in lower DRAM memory access overhead. Considering that the minimum bit width supported by Tensor Cores in current mainstream computing cards is 4-bit, the most efficient quantization scheme we can choose is 4-bit quantization for QK. Lower quantization bit widths cannot yield additional performance benefits.

However, lower-precision quantization also introduces greater errors, possibly leading to a decrease in the accuracy of important block selection in Selection-Pass. To demonstrate the block selection accuracy under 4-bit quantization, We tested the “recall rate” between the blocks selected using 4-bit QK and those selected using full precision floating-point QK. The “recall rate” metric can be defined as “the number of blocks commonly selected by both methods” divided by “the total number of blocks selected using full precision QK”. The input samples used in the test are aligned with those in Figure 3(a), and the tests are also based on the Llama-3.1-8B-Instruct model. We obtained the statistical results under different context lengths, which are demonstrated in Table 5:

Table 5: Block selection recall rate

Context length	8K	16K	32K	64K	128K
Recall rate	98.2%	97.7%	97.2%	96.6%	95.4%

The experimental results demonstrate that 4-bit QK is capable of accurately selecting important blocks. The “Sparsity-Accuracy” curve presented in Figure 5(b) also indicates that quantizing QK to 4-bit can achieve block selection accuracy comparable to that of full-precision QK. Thus, SALE adopts 4-bit scheme which enables it to balance accuracy and efficiency.

I TWO PASS DESIGN RATIONALE

In contrast to our “Estimation-computation disaggregation” (“Two-pass”) kernel design mentioned in Section 3.4, the “One-pass” design integrates these two processes into a single kernel. Specifically,

when iterating through a key block, if the current attention block is determined to be important, the output for the current block is computed directly (i.e. “on-the-fly pruning”).

Theoretically, this design eliminates the need to store the coordinates of important blocks, and can reuse the computation results of low-bit QK GEMM, which is expected to run faster. However, in practice, the “One-pass” design is not feasible. Two primary reasons are listed here:

4-bit quantization issue In fact, the results of 4-bit QK GEMM cannot be directly used for computing attention output, as this would lead to significant accuracy degradation. To illustrate this point, in Table 6, we present the accuracy experiment results of using 4-bit QK in the computation stage. We refer to this scheme as “4+4”, whereas SALE corresponds to “4+8”. We also tested a scheme that calculates attention scores using only 4-bit QK without block pruning, denoted as “4 dense”. All 4-bit quantization in the table has adopted the technique mentioned in SageAttention2 (Zhang et al., 2024b) to improve quantization quality. We report the scores on two tasks of InfiniteBench in Table 6:

Table 6: Accuracy performance comparison

Tasks	Retrieve.KV	En.MC
4+4	35.8	63.30
4 dense	41.4	60.26
SALE (4+8)	56.4	66.38

Our experimental results show that the “4+4” scheme leads to a degradation in model performance. Even if we do not prune any blocks, a significant score drop still occurs. Therefore, we choose to use QK with a precision of at least 8 bits in the Computation Pass.

Implementation challenges According to the hardware characteristics of GPUs, “on-the-fly pruning” cannot be efficiently implemented on GPUs. The requirement for immediate decisions imposed by the “on-the-fly pruning” means that we have to perform an all-reduce operation each time after computing the QK GEMM. Based on the analysis provided in Section G, it will introduce much more CTA-wise all-reduce operations compared to SALE, which is not efficient.

In addition, “on-the-fly pruning” is unfriendly to the GPU memory hierarchy. Due to the high DRAM access latency of GPUs, high-performance CUDA implementations typically pre-issue memory access instructions for data that will be computed to achieve overlap between memory access and computation. However, the nature of “on-the-fly pruning” dictates that we must decide whether to issue the memory access instruction only after completing the QK GEMM, and then wait for the data transmission of value tensor, which waste the computational resource on SM. Otherwise, if we issue memory access instructions for the value tensor for all blocks, the unnecessary memory access will also impair performance.

Based on the above analysis, even if we adopt the “8+8” scheme to attempt reusing the results of 8-bit QK GEMM, it still cannot match the efficiency of “Two-pass” design.

J SELECTION-PASS ALGORITHM

Algorithm 1: Selection-Pass

Input: $Q, K \in \mathbb{R}^{N \times d}$, 4-bit quantized matrices $\tilde{Q}, \tilde{K} \in \mathbb{Z}^{N \times d}$, threshold τ , block size b_q, b_k , local area size l .

- 1 $N_q \leftarrow N/b_q, N_k \leftarrow N/b_k, N_{local} \leftarrow l/b_k$;
- 2 Split Q, K into blocks $Q_i \in \mathbb{R}^{b_q \times d}, K_j \in \mathbb{R}^{b_k \times d}$, split $\tilde{Q}, \tilde{K}$ into blocks $\tilde{Q}_i \in \mathbb{Z}^{b_q \times d}, \tilde{K}_j \in \mathbb{Z}^{b_k \times d}$
- for** $i = 0$ to $N_q - 1$ **do**
- 3 $I_{SL} \leftarrow \{0\} \cup [i - N_{local}, i - 1]$; // Block indices of sink-local area
- 4 $\tilde{m}, \tilde{l} \in \mathbb{R}^{b_q}, \tilde{m} \leftarrow -\infty, \tilde{l} \leftarrow 0$; // Initialize intermediate result
- 5 **for** $j \in I_{SL}$ **do**
- 6 **if** $j \neq 0$ **then**
- 7 $\tilde{m}_\Delta \leftarrow \tilde{m} - \text{rowmax}(Q_i K_j^T / \sqrt{d})$; $\tilde{l} \leftarrow \tilde{l} \cdot \exp(\tilde{m}_\Delta)$;
- 8 **end**
- 9 $\tilde{m} \leftarrow \text{rowmax}(Q_i K_j^T / \sqrt{d})$; // Ignore causal mask
- 10 $\tilde{l} \leftarrow \tilde{l} + \text{rowsum}(\exp(\frac{Q_i K_j^T}{\sqrt{d}} - \tilde{m}))$;
- 11 $M_{bs}[i, j] \leftarrow 1$
- 12 **end**
- 13 **for** $j \leftarrow 1$ to $(i - N_{local} - 1)$ **do**
- 14 $\tilde{S}_{ij} \leftarrow \text{Dequantize}(\tilde{Q}_i \tilde{K}_j^T) / \sqrt{d}$; // Approximate attention weight
- 15 $\tilde{P}_{ij} \leftarrow \exp(\tilde{S}_{ij} - \tilde{m}) / \tilde{l}$; // Compute Relative Attention Score
- 16 $M_{bs}[i, j] \leftarrow \max(\tilde{P}_{ij}) \geq \tau$;
- 17 **end**
- 18 **end**

Output: Block-level sparse mask M_{bs}
