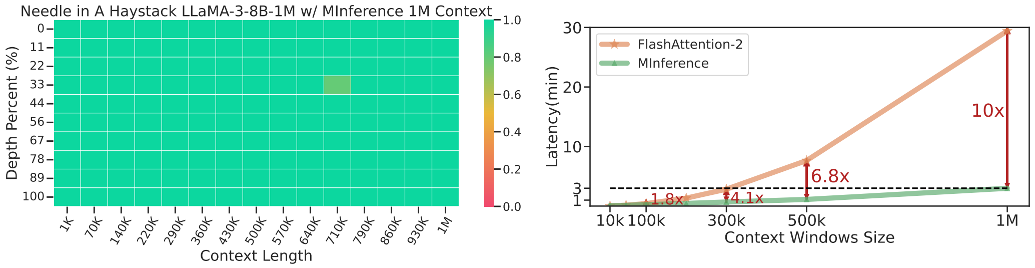
MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention

Huiqiang Jiang[†] **Yucheng Li**[◇,†] **Chengruidong Zhang**[†] **Qianhui Wu, Xufang Luo,**
Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, Lili Qiu
Microsoft Corporation, [◇]University of Surrey
`{hjiang,chengzhang,yuqyang}@microsoft.com,yucheng.li@surrey.ac.uk`

Abstract

The computational challenges of Large Language Model (LLM) inference remain a significant barrier to their widespread deployment, especially as prompt lengths continue to increase. Due to the quadratic complexity of the attention computation, it takes 30 minutes for an 8B LLM to process a prompt of 1M tokens (i.e., the pre-filling stage) on a single A100 GPU. Existing methods for speeding up pre-filling often fail to maintain acceptable accuracy or efficiency when applied to long-context LLMs. To address this gap, we introduce **MInference** (*Million-tokens Inference*), a sparse calculation method designed to accelerate pre-filling of long-sequence processing. Specifically, we identify three unique patterns in long-context attention matrices—the *A-shape*, *Vertical-Slash*, and *Block-Sparse*—that can be leveraged for efficient sparse computation on GPUs. We determine the optimal pattern for each attention head offline and dynamically build sparse indices based on the assigned pattern during inference. With the pattern and sparse indices, we perform efficient sparse attention calculations via our optimized GPU kernels to significantly reduce the latency in the pre-filling stage of long-context LLMs. Our proposed technique can be directly applied to existing LLMs without any modifications to the pre-training setup or additional fine-tuning. By evaluating on a wide range of downstream tasks, including InfiniteBench, RULER, PG-19, and Needle In A Haystack, and models including LLaMA-3-1M, GLM-4-1M, Yi-200K, Phi-3-128K, and Qwen2-128K, we demonstrate that MInference effectively reduces inference latency by up to $10\times$ for pre-filling on an A100, while maintaining accuracy. Our code is available at `https://aka.ms/MInference`.



(a) Needle In A Haystack

(b) Latency Speedup

Figure 1: Attention weights, especially in long-context LLMs, exhibit up to 96.8% sparsity in contexts of 128K. We propose **MInference**, leveraging dynamic sparse attention to accelerate the pre-filling stage of long-context LLM inference. It achieves up to 10x speedup for 1M contexts on a single A100, as shown in (b), and matches or surpasses baselines, as demonstrated by Needle In A Haystack [Kam23] in (a) on LLaMA-3-8B-1M [Gra24].

[†] Equal contribution. [◇] Work during internship at Microsoft.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

Large language models (LLMs) have entered the era of long-context processing, with some of them supporting context windows ranging from 128K to 10M tokens [Gra24, RST$^+$24, LYZA24, YCL$^+$24, AJA$^+$24, DA24]. These extended context windows enable LLMs to unlock a multitude of complex real-world applications, such as repository-level code understanding [BSK$^+$24, JYW$^+$23, POC$^+$23], long-document question-answering [CPG$^+$23, LZD$^+$24], self-play reasoning [Ope24], extreme-label in-context learning [LZD$^+$24], and long-horizon agent tasks [Wen23].

However, due to the quadratic complexity of attention, it can take several minutes for the model to process the input prompt (i.e., the pre-filling stage) and then start to produce the first token, which leads to unacceptable Time To First Token experience, thus greatly hinders the wide application of long-context LLMs. As shown in Fig. 2a, when serving LLaMA-3-8B on a single A100 machine, the model would keep users waiting for 3 minutes to finish the pre-filling stage given a prompt of 300K tokens, and this number increases to 30 minutes for a prompt of 1M tokens. The overhead of self-attention computation exceeds 90% of the total pre-filling latency, which makes it the major bottleneck in long-context processing of LLMs. Previous research has shown that the attention matrices are highly sparse [LQC$^+$22, DSY24], which has led to the development of fixed sparse attention methods such as Longformer [BPC20] and BigBird [ZGD$^+$20]. However, prior studies have also noted that attention distributions vary significantly across different inputs [LCW21, LQC$^+$22]. This dynamic nature prevents prior sparse methods from being used directly on long-context LLMs without expensive training or fine-tuning. But if the dynamic sparse attention patterns could be efficiently predicted online, the pre-filling latency of long-context LLMs could be significantly reduced by calculating only the most important part of the attention weights.

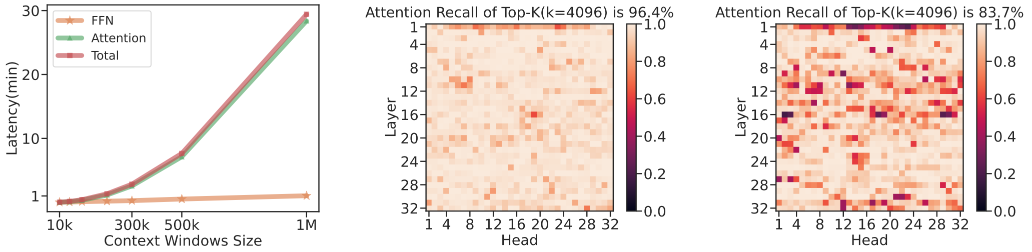
Building upon this idea, we present **MInference**, a technique that reduces 95% of FLOPs in the attention computation to significantly accelerate the pre-filling stage of long-context LLM inference via dynamic sparse attention. Unlike existing dynamic sparse attention methods that introduce large computational overhead to estimate attention patterns with low-rank hidden dimensions [LQC$^+$22, RCHG$^+$24], our method is designed specifically for long-context scenarios with minimal overhead in estimation. Specifically, we conduct extensive analysis and identify three general patterns of sparse attention in long-context LLMs: *A-shape* pattern, *Vertical-Slash* pattern, and *Block-Sparse* pattern. Based on these findings, we introduce a kernel-aware search method to assign the optimal attention pattern for each head. Importantly, instead of fixed attention masks in prior studies, we perform an efficient online approximation to build a dynamic sparse mask for each head according to their assigned pattern and particular inputs. For example, to build a dynamic sparse mask for a specific prompt on one *Vertical-Slash* head, we use a partial of attention weight consisting of the last last_q query and key vectors (i.e. $\boldsymbol{Q}_{[-\text{last\_q}:]}$ and $\boldsymbol{K}$) to estimate the most important indices of the vertical and slash lines globally on the attention matrix. For *Block-Sparse* heads, we perform mean pooling on both query and key vectors in blocks of 64 and calculate the block-level attention weights to determine the most important blocks and thereby obtain a block-sparse dynamic mask. After obtaining the dynamic sparse mask, three optimized GPU kernels are used, which we developed for the above three sparse patterns. These kernels are based on the dynamic sparse compilers PIT [ZJZ$^+$23], Triton [TKC19] and FlashAttention [Dao24], which enable extremely efficient computation of dynamic sparse attention.

Extensive experiments are conducted on various Long-context LLMs, including LLaMA-3-8B-1M [Gra24], GLM-4-9B-1M [GZX$^+$24], and Yi-9B-200K [YCL$^+$24], across benchmarks with context lengths over 1M tokens, such as InfiniteBench [ZCH$^+$24], RULER [HSK$^+$24], Needle In A Haystack [Kam23], and PG-19 [RPJ$^+$20]. Needle In A Haystack was also tested on Phi-3-Mini-128K [AJA$^+$24] and Qwen-2-7B-128K [BBC$^+$23]. Results show that MInference speeds up the pre-filling stage by up to $10\times$ for 1M contexts with LLaMA-3-8B on a single A100, reducing latency from 30 minutes to 3 minutes per prompt, while maintaining or improving accuracy.

2 Attention Heads: Dynamic, Sparse, and Characteristic

2.1 Attention is Dynamically Sparse

The sparsity of attention weights in pre-trained LLMs, especially in long-context scenarios, has been well-documented [LQC$^+$22, RCHG$^+$24, LWD$^+$23, XTC$^+$24]. As shown in Fig. 2b, for an attention



(a) Attention incurs heavy cost.

(b) Attention is sparse.

(c) Sparsity of attention is dynamic.

Figure 2: (a) Latency breakdown of the pre-filling stage. (b) How much attention scores can top-k (k=4096) columns cover in a 128k context. (c) Less attention scores are retrieved when reusing the top-k indices from another examples, indicating its dynamic nature. Visualizations are based on LLaMa-3-8B with a single A100.

matrix of size $128k \times 128k$, retaining only the top 4k columns recalls 96.8% of the total attention. In other words, each token is attending to a limit number of tokens despite the long sequence it is processing.

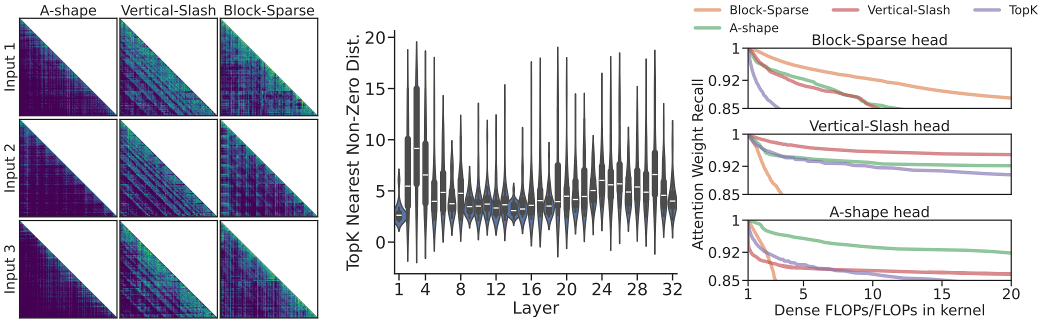
On the other hand, although the sparse nature of attention matrices is shared across different inputs, the exact distributions of sparse pattern are highly dynamic. That is to say, a token at a given position only attends to a subset of the sequence in self-attention, and the exact tokens it attends to are highly context-dependent and vary significantly across different prompts. This dynamism has been mathematically demonstrated in prior studies [LCW21, LCW23]. As depicted in Fig. 2c, if we take the top 4k columns found in Fig. 2b and apply it on another prompt of 128k, the recall of attention would drop largely to 83.7%.

2.2 Attention Sparsity Exhibits Patterns

Table 1: Comparison of different sparse patterns.

Patterns	A-shape	Vertical-Slash	Block-Sparse	Top-K
Spatial Distribution	Static structured	Dynamic structured	Dynamic structured	Dynamic fine-grained
Latency on GPU	Low	Medium	Low	High
Time to build the index	Zero	Small	Small	High

Although the sparsity distribution of attention matrix is dynamic, previous works [XTC$^+$24, HWP$^+$24] have shown that they exhibit certain patterns in the two-dimensional space such as spatial clustering. Through our analysis of long-context prompts of various lengths and tasks, we have



(a) Attention patterns

(b) Attention is spatial clustering

(c) Attention pattern recall

Figure 3: (a) Visualization of attention weights from different attention heads. For different prompts and tasks, the pattern of the same head is relatively consistent, but the sparse indices are dynamically changing.(b) Distance of the top-10 nearest non-zero element in the attention matrix. (c) Attention recall distribution using our identified patterns, where FLOPs in the kernel refer to the real FLOPs required for sparse attention computing using on GPUs. Here, a 1x64 block size is used for the *Vertical-Slash* pattern, and a 64x64 block size is used for others on GPUs. All visualization are based on LLaMA-3-8B-Instruct-262K [Gra24].

categorized such attention sparse patterns into the *A-shape*, *Vertical-Slash* (VS), and *Block-Sparse* patterns, as shown in Fig. 3a and Fig. 4. Table 1 details the characteristics and differences between these three patterns.

***A-shape* pattern** The attention weights of these types of heads are concentrated on initial tokens and local windows [XTC+24, HWP+24], exhibiting relatively higher stability.

***Vertical-Slash* (VS) pattern** The attention weights are concentrated on specific tokens (vertical lines) [MJ23] and tokens at fixed intervals (slash lines). The positions of vertical and slash lines in this pattern dynamically change with the context content and exhibit a certain sparsity, making them difficult to be encompassed by local windows and *A-shape* patterns.

***Block-Sparse* pattern** This sparsity pattern is the most dynamic, exhibiting a more dispersed distribution. Despite its dynamism, the attention weights maintain some characteristics of spatial clustering, which we identify as the block-sparse pattern. We analyzed the distances between non-zero attention weights and their top-k nearest non-zero neighbors within a 128k prompt as shown in Fig. 3b. The results indicate that across layers and heads, the distances between nearest non-zero values are generally concentrated around 5, suggesting a strong spatial clustering of the attention weights.

The point of these three patterns is that we can leverage them to perform highly efficient sparse computing for the attention matrix in long-context LLMs. In Fig. 3c, we test how efficient is our indentified patterns retrieving attention scores with limit computing cost on GPU (FLOPs). First, attention heads are labeled with one of the sparse pattern (detail see §3.2). Then we demonstrate our patterns are significantly more efficient compared to other sparse methods [RCHG+24, XTC+24, PPJF24]. Specifically, with the same amount of FLOPs, our patterns achieve a notable higher recall on attention scores, which can potentially lead to better accuracy. For example, previous Top-K methods [RCHG+24, XTC+24, PPJF24] struggle with the *Block-Sparse* pattern as they focus on specific tokens globally, while our pattern retrieves attention scores more efficiently and accurately. We example how we use these patterns on long-context LLMs and how we implement optimized GPU kernels for these patterns in §3.

3 MInference 1.0

Following the analysis in §2, we propose **MInference** to accelerate the pre-filling stage of long-context LLMs, consisting of three steps: 1) Offline attention pattern identification for each head; 2) Dynamic build of sparse indices w.r.t. the pattern; 3) Sparse attention calculation with optimized GPU kernels.

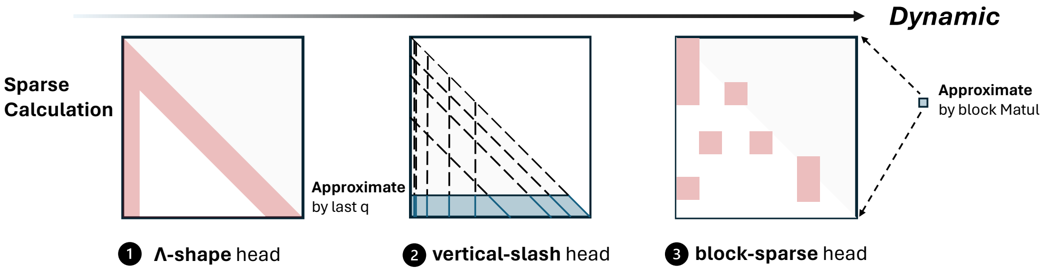


Figure 4: The three sparse methods in MInference.

3.1 Problem Formulation

When accelerating the pre-filling stage of long-context LLMs with sparse attention computing, the attention matrix can be formulated as follows:

$$\boldsymbol{A}(\boldsymbol{M}) = \text{Softmax}(\frac{1}{\sqrt{d}}\boldsymbol{Q}\boldsymbol{K}^\top - c(1-\boldsymbol{M})), \tag{1}$$

where $M_{i,j} \in \{0, 1\}$ represents the dynamic sparse mask for item i, j of the attention matrix. Here, c is a large constant, such as 1e5, ensuring that the less important attention weights for which $M_{i,j} = 0$ have values approaching zero after the softmax, i.e., $A_{i,j} \approx 0$.

The goal of the dynamic sparse attention system is to achieve greater speedup with minimal overhead while retaining as much of the attention weights as possible. Formally, this can be expressed as:

$$\begin{aligned} \min \quad & |\boldsymbol{A}(\boldsymbol{M}) - \boldsymbol{A}_{\text{dense}}|, \\ \min \ \ & t_{\text{sparse}}(\boldsymbol{M}) + t_{\text{overhead}}(\boldsymbol{M}), \end{aligned} \tag{2}$$

where t_{sparse} and t_{overhead} represent the time spent on dynamic sparse attention computation and estimation of the approximate dynamic sparse pattern, respectively.

3.2 Speedup of Long-context LLM Inference via Dynamic Sparse Attention

Kernel-Aware Optimal Sparse Pattern Search To achieve the best accuracy with limited FLOPs budget, we propose an offline Kernel-Aware Optimal Sparse Pattern Search method. In this step, we determine which sparse pattern will be used for each attention head, and the optimal setting for the pattern in real calculation (e.g., the number of vertical/slash lines in *VS* pattern; or the number of top-k blocks in *BS* patterns). As shown in Algorithm 1, we first create the search space based on a target FLOPs for each pattern, ensuring all potential candidates (i.e., different patterns with different settings) have similar computational cost. *Kernel-aware* here indicates the computational cost reflects the real FLOPs in GPU kernels, instead of conceptual estimations, which is crucial to achieve the optimal acceleration.

Next, we go through the search space with a reference example to decide the optimal pattern and setting. Specifically, we use recall of the attention output as the objective criterion when searching for the best pattern. This approach leverages FlashAttention [Dao24] to reduce GPU memory overhead and incorporates the information from the $\boldsymbol{V}$ matrix, enabling end-to-end selection of the best pattern, which further enhances performance.

Algorithm 1 Kernel-Aware Sparse Pattern Search

```
Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, patterns $p$, search space $\rho$, target FLOPs $t$, initialized search space $\sigma$
# Build kernel-aware search space
for $i \leftarrow 1$ to $|\sigma|$ do
    $t_i \leftarrow$ FLOPs_in_kernel$(\sigma_i)$
    while $|t_i - t| > \epsilon$ do
        $\sigma_i \leftarrow$ ChangeSpace$(\sigma_i, p_i)$
        $t_i \leftarrow$ FLOPs_in_kernel$(\sigma_i)$
    end while
    $\rho \leftarrow \rho \cup \sigma_i$
end for
# Search for optimal head pattern
$p_{\text{best}} \leftarrow \phi$
$\boldsymbol{y} \leftarrow$ Softmax$(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d})$
for $i \leftarrow 1$ to $|\rho|$ do
    $\boldsymbol{y}_i \leftarrow$ SparseAttention$(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d}, \rho_i)$
    $p_{\text{best}} \leftarrow$ argmin$(\boldsymbol{y}_i - \boldsymbol{y}, p_{\text{best}})$
end for
return $p_{\text{best}}$
```

Sparsity Indices Approximation and Dynamic Sparse Attention Calculation During the inference stage, we will perform an online estimation on the attention matrix to dynamically determine the

Algorithm 2 Vertical-Slash Head

```
Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, $k_v, k_s \in \mathbb{N}$
# Approximate vertical and slash pattern (last_q = 64)
$\widehat{\boldsymbol{A}} \leftarrow \text{softmax}\left(\boldsymbol{Q}_{[-\text{last\_q}:]}\boldsymbol{K}^\top/\sqrt{d} + \boldsymbol{m}_{\text{casual}}\right)$
# Indices of top $k_v$ vertical line, sum in vertical
$\boldsymbol{i}_v \leftarrow \text{argtopk}\left(\text{sum}_v(\widehat{\boldsymbol{A}}), k_v\right)$
# Indices of top $k_s$ slash line, sum in slash
$\boldsymbol{i}_s \leftarrow \text{argtopk}\left(\text{sum}_s(\widehat{\boldsymbol{A}}), k_s\right)$
# Build sparse attention index
$\boldsymbol{i}_{vs} \leftarrow \text{sparseformat}(\boldsymbol{i}_v, \boldsymbol{i}_s)$
# Final dynamic sparse attention scores (only index block)
$\boldsymbol{A} \leftarrow \text{softmax}\left(\text{sparse}(\boldsymbol{Q}\boldsymbol{K}^\top, \boldsymbol{i}_{vs})/\sqrt{d}\right)$
# Sparse mixed scores and values
$\boldsymbol{y} \leftarrow \text{sparse}(\boldsymbol{A}\boldsymbol{V}, \boldsymbol{i}_{vs})$
return $\boldsymbol{y}$
```

Algorithm 3 Block-Sparse Head

```
Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, $k_b \in \mathbb{N}$
# Approximate block-sparse pattern (block_size = 64)
$\widehat{\boldsymbol{Q}} \leftarrow \text{MeanPooling}(\boldsymbol{Q}, block\_size)$
$\widehat{\boldsymbol{K}} \leftarrow \text{MeanPooling}(\boldsymbol{K}, block\_size)$
$\widehat{\boldsymbol{A}} \leftarrow \text{softmax}\left(\widehat{\boldsymbol{Q}}\widehat{\boldsymbol{K}}^\top/\sqrt{d} + \boldsymbol{m}_{\text{casual}}\right)$
# Indices of top $k$ blocks
$\boldsymbol{i}_b \leftarrow \text{argtopk}\left(\widehat{\boldsymbol{A}}, k_b\right)$
# Build sparse attention index
$\boldsymbol{i}_b \leftarrow \text{sparseformat}(\boldsymbol{i}_b)$
# Final dynamic sparse attention scores (only index block)
$\boldsymbol{A} \leftarrow \text{softmax}\left(\text{sparse}(\boldsymbol{Q}\boldsymbol{K}^\top, \boldsymbol{i}_b)/\sqrt{d}\right)$
# Sparse mixed scores and values
$\boldsymbol{y} \leftarrow \text{sparse}(\boldsymbol{A}\boldsymbol{V}, \boldsymbol{i}_b)$
return $\boldsymbol{y}$
```

spatial distribution our sparse indices, based on the assigned patterns and the exact input. After that, we conduct the sparse attention computations with our optimized GPU kernels. The implementation details of our kernels can be found in Appendix C.4. Noted that the sparse mask is static for *A-shape* heads, so there is no overhead in building the dynamic masks, and only sparse calculation is required.

(i) Vertical-Slash head. As shown in Algorithm 2, due to the continuity of vertical and slash lines, we matmul the last query vector $\boldsymbol{Q}_{[-\text{last\_q}:]}$ and key vector $\boldsymbol{K}$ to produce the estimated attention matrix $\widehat{\boldsymbol{A}}$, which, in turn, is used to determine the indices for the vertical $\boldsymbol{i}_v$ and slash $\boldsymbol{i}_s$ lines. After obtaining the sparse indices for the vertical and slash lines, we convert them into a sparse format $\boldsymbol{i}_{vs}$. Using these sparse indices, we perform block-sparse calculations of the attention weights and attention output.

(ii) Block-Sparse head. Per Algorithm 3, mean pooling is applied on $\boldsymbol{Q}$ and $\boldsymbol{K}$ to obtain $\widehat{\boldsymbol{Q}}$ and $\widehat{\boldsymbol{K}}$, respectively. The two matrices are multiplied to get the estimated block-level attention weights $\widehat{\boldsymbol{A}}$. Since the mean pooling and matrix multiplication operations are commutative, the resulting attention weights are approximately equivalent to the actual attention weights after mean pooling. This allows us to approximate the actual attention weights' block-sparse pattern with minimal overhead. Similarly, we build a sparse index $\boldsymbol{i}_b$ and use it to compute the sparse attention weights and attention output.

4 Experiments

In this section, we investigate two questions: **(i) How effective is MInference?** We evaluate our method on three general long-context benchmarks: InfiniteBench [ZCH$^+$24], RULER [HSK$^+$24], and the Needle In A Haystack task [Kam23], as well as the long-context language modeling task [RPJ$^+$20]. These benchmarks cover long-context QA, multi-hop QA, math reasoning, aggregation tasks, summarization, retrieval tasks, and code debugging, allowing us to assess MInference's effectiveness across a wide range of long-context scenarios. **(ii) How efficient is MInference?** We delve into the end-to-end latency and its breakdown to evaluate the efficiency of MInference. Additional experimental, latency results, and analysis can be found in Appendix D, E, and F.

Implementation Details Our experiments use four state-of-the-art long-context LLMs: LLaMA-3-8B-Instruct-262k[1], LLaMA-3-8B-Instruct-1048k[2], GLM-4-9B-1M [GZX$^+$24], and Yi-9B-200K [YCL$^+$24]. Additionally, we tested Needle in A Haystack [Kam23] on Phi-3-Mini-128K [AJA$^+$24] and Qwen2-7B-128K [BBC$^+$23], as detailed in Appendix D.1. To guarantee stable results, we use greedy decoding in all experiments. We provide a simple custom implementation of our method in PyTorch, built on FlashAttention [Dao24], Triton [TKC19], and the dynamic sparse compiler PIT [ZJZ$^+$23]. We set the target FLOPs t to 1k global tokens and 4k local windows in the *A-shape* pattern. We set last_q $= 64$ and $block\_size = 64$ in the *Vertical-Slash* and *Block-Sparse* patterns, respectively. The latency experiments are conducted on a single Nvidia A100 GPU in the bfloat16 format. More details are provided in Appendix C.2.

Dataset & Evaluation Metrics We use the provided metrics and scripts from the following benchmarks for evaluation. More details about dataset can be found in Appendix C.1.

(i) InfiniteBench [ZCH$^+$24]: This benchmark consists of 10 tasks, including retrieval tasks such as PassKey retrieval, Number retrieval, and KV retrieval, as well as representative realistic tasks like question-answering, coding, dialogue, and summarization. The average context length of InfiniteBench is about 214K tokens.

(ii) RULER [HSK$^+$24]: A challenging long-context benchmark consisting of 4 categories and 13 complex tasks, including retrieval, multi-hop tracing and aggregation, and QA tasks. It contains subsets with different prompt lengths up to 128k tokens, allowing us to determine the actual context window size of the model based on the results.

(iii) Needle In A Haystack [Kam23]: A long-context retrieval benchmark testing LLMs' performance with context window sizes up to 1M tokens where information placed at various positions.

[1] https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-262k

[2] https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k

Table 2: Performance of different methods with different base models on InfiniteBench [ZCH+24].

Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
LLaMA-3-8B-262K	20.2	12.4	67.3	6.0	12.9	22.1	26.6	100.0	100.0	14.4	38.2
StreamingLLM	21.0	8.2	40.2	10.0	10.4	**25.9**	30.0	86.8	5.1	0.8	23.8
StreamingLLM w/ dilated	20.1	9.4	44.5	**15.5**	11.2	20.5	27.5	5.0	87.5	0.5	24.2
StreamingLLM w/ strided	17.3	8.2	27.5	14.5	11.2	19.5	27.5	4.0	2.1	1.0	13.3
InfLLM	**24.1**	7.8	45.0	6.0	11.4	19.5	32.9	**100.0**	**100.0**	1.2	34.8
Ours w/ static	19.9	8.6	43.2	3.5	8.9	20.6	25.1	92.4	96.3	0.2	31.9
Ours	20.5	**12.9**	**65.9**	7.5	**12.5**	22.3	**33.1**	**100.0**	**100.0**	**12.8**	**38.8**
Yi-9B-200K	8.2	10.6	64.2	1.0	17.3	21.3	23.4	99.8	100.0	28.8	37.5
StreamingLLM	5.4	**14.2**	38.0	**4.0**	18.8	18.8	22.3	39.2	6.1	1.6	16.8
StreamingLLM w/ dilated	5.7	4.2	15.0	0.0	18.2	0.0	2.9	0.0	0.0	0.0	4.2
StreamingLLM w/ strided	6.1	4.5	9.8	0.0	16.9	0.0	3.1	1.5	0.0	0.0	4.6
InfLLM	6.3	13.0	45.9	2.5	**21.5**	20.6	34.6	85.3	88.1	1.4	31.9
Ours w/ static	5.8	12.6	48.5	3.0	12.6	20.8	**25.1**	60.9	38.5	1.0	22.9
Ours	**7.9**	11.2	**64.2**	1.0	17.9	**24.1**	23.1	**99.5**	**100.0**	**27.6**	**37.7**
GLM-4-9B-1M	28.3	9.7	68.6	39.5	12.1	29.4	38.9	100.0	100.0	41.0	46.7
StreamingLLM	27.7	6.4	40.2	12.5	10.8	27.7	21.1	97.1	25.6	0.6	27.0
InfLLM	28.0	7.3	45.0	14.0	10.7	27.9	**39.4**	98.0	**100.0**	2.6	37.3
Ours	**28.8**	**9.6**	**68.6**	**38.5**	**12.0**	**30.7**	39.1	**100.0**	**100.0**	**43.0**	**47.0**

(iv) PG-19 [RPJ+20]: Following StreamingLLM [XTC+24] and H2O [ZSZ+24], we use PG-19 for long-context language modeling tasks with prompts up to 100k tokens.

Baselines We include five training-free sparse attention approaches as our baselines: 1) StreamingLLM [XTC+24], which corresponds to the *A-shape* pattern. We use 1k global tokens and 4k local windows in all our experiments; 2) StreamingLLM w/ dilated [BPC20], which sets dilated local windows with intervals in the local windows direction. We use 1k global tokens and 8k dilated attention windows with an interval of 1; 3) StreamingLLM w/ strided [CGRS19], which retains local windows while adding dilated attention. We use 1k global tokens, 2k local windows, and 4k dilated attention windows with an interval of 1; 4) InfLLM [XZH+24], which uses a memory unit to process streaming long sequences. Following the paper, we set 128 global tokens and 8k local windows in all experiments; 5) Ours w/ static, which utilizes static sparse indices in the *Vertical-Slash* and *Block-Sparse* heads. For all baselines, we perform sparse computation only during the pre-filling stage, while retaining dense computation during the decoding stage.

InfiniteBench As shown in Table 2, MInference achieves the best overall performance on InfiniteBench compared to baseline methods. Remarkably, MInference matches or even slightly surpasses the performance of the original full attention baseline on some tasks, despite the significant acceleration it provided. From the perspective of different tasks, our method not only performs well in natural language tasks such as summarization, QA, and code, but also maintains the original model's performance on retrieval-related tasks. Baseline methods such as StreamingLLM, on the

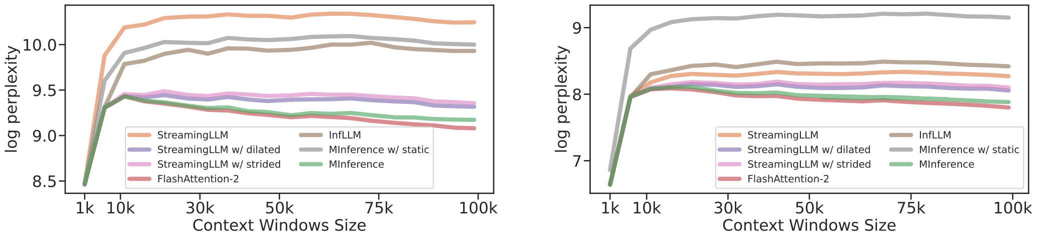
Table 3: Performance (%) of different models and different methods on RULER [HSK+24] evaluated at lengths from 4k to 128k.

Methods	Claimed	Effective	4K	8K	16K	32K	64K	128K	Avg.
LLaMA-3-8B-262K	262K	16K	97.2	91.8	87.3	80.8	77.4	72.2	84.4
StreamingLLM	-	4K	97.2	38.1	37.5	17.2	14.2	9.4	35.0
StreamingLLM w/ dilated	-	<4K	23.4	0.7	1.4	18.8	16.5	15.6	12.7
StreamingLLM w/ strided	-	<4K	2.0	0.7	0.6	0.6	0.7	1.3	1.0
InfLLM	-	4K	89.4	79.8	70.1	55.6	43.0	39.5	62.9
Ours	-	**32K**	**97.7**	**91.2**	**88.5**	**85.0**	**82.3**	**77.6**	**87.0**
Yi-9B-200K	200K	8K	91.9	90.2	78.8	76.3	68.1	62.9	78.1
StreamingLLM	-	4K	91.9	37.8	33.9	18.6	13.0	12.8	34.3
StreamingLLM w/ dilated	-	<4K	44.8	42.8	38.5	29.8	26.8	23.9	34.4
StreamingLLM w/ strided	-	<4K	2.6	0.7	0.6	0.6	1.2	0.5	1.1
InfLLM	-	<4K	80.3	83.9	60.7	45.2	38.6	30.2	56.5
Ours	-	8K	**92.3**	**89.7**	**79.0**	**73.8**	**64.7**	**56.9**	**74.7**
GLM-4-9B-1M	1M	64K	93.8	91.6	89.3	87.4	85.2	80.8	88.0
StreamingLLM	-	4K	93.8	66.9	58.5	51.4	45.9	39.1	59.3
InfLLM	-	8K	**94.7**	89.5	76.4	66.5	56.8	53.5	72.9
Ours	-	64K	94.6	**93.1**	**91.0**	**89.6**	**85.5**	**84.0**	**89.6**

contrary, struggle with these retrieval tasks. Additionally, on tasks such as dialogue QA, using local attention mechanisms can better handle these tasks, while our performance is closer to the original results, indicating that our method is not solely based on local windows. Extending the local windows' intervals in StreamingLLM, i.e., w/ dilated and w/ strided, has minimal impact on the model's performance.

RULER To further reveal the true potential of our method in long-context LLMs, we evaluate MInference with the state-of-the-art long-context challenge, RULER. As shown in Table 3, MInference effectively maintains the long-context performance even in complex multi-hop or aggregation tasks in RULER. It even outperforms the original full attention for testing lengths beyond 32K, achieving effective context windows of 32K and 64K (context with performance over 85% is considered effective [HSK+24]) in LLaMA-3-8B-262K and GLM-4-9B-1M.

Language Modeling Following the approach of StreamingLLM [XTC+24] and H2O [ZSZ+24], we evaluate our methods against baselines on the language modeling task based on the PG-19 dataset [RPJ+20]. As shown in 5, our method yields best results compared to other sparse approaches, and exhibits minimal divergence compared to the full attention baseline. For prompts of 100K token, our perplexity is only 0.2 higher than the full attention, but lower than StreamingLLM for 0.25 and 0.75 on the Yi-9B-200K and LLaMA-3-262K models respectively.



(a) LLaMA-3-8B-Instruct-262K

(b) Yi-9B-200K

Figure 5: Perplexity results on PG-19 [RPJ+20] using different models and methods.

Needle In A Haystack Comparing Fig. 1a to Fig. 6, our method effectively retains the ability to process information at different positions across various context windows, ranging from 1k to 1M tokens. In contrast, methods like StreamingLLM and InfLLM (as shown in Appendix D.1), while effective at reducing latency, experience a sharp decline in performance once critical information extends beyond the range of global tokens and local windows.

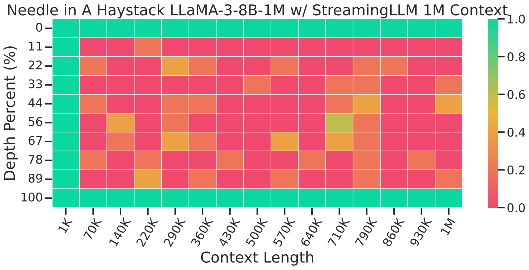


Figure 6: Results on Needle In A Haystack of StreamingLLM [XTC+24] in LLaMA-3-8B-1M.

Ablation Study To evaluate the contributions of different components in MInference, we introduce four variants for the ablation study: (1) Ours w/ static, which uses a static sparse mask in the *Vertical-Slash* and *Block-Sparse* patterns; (2) Ours w/ only A-shape, which is equivalent to StreamingLLM; (3) Ours w/ only block-sparse, which uses only the *Block-Sparse* pattern in the dynamic sparse calculation. (4) Ours w/ only vertical-slash, which uses only the *Vertical-Slash* pattern in the dynamic sparse calculation.

Tables 2, 3, and 4 present the ablation results. It first proves that using static indices significantly degrades LLM performance, especially in highly dynamic tasks like KV retrieval, where accuracy nearly drops to zero. This highlight the necessity of our dynamic strategy and the effectiveness of our dynamically built sparse indices. Additionally, remove any pattern from the three leads to varying degrees of performance degradation. Specifically, "only A-shape" can only capture information

Table 4: Performance of different ablation methods using LLaMA-3-8B-Instruct-262K on InfiniteBench [ZCH+24].

Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
Ours	20.5	12.9	65.9	7.5	12.5	22.3	33.1	100.0	100.0	12.8	38.8
Ours w/ only block-sparse	12.4	3.4	5.7	6.0	3.1	12.2	24.0	59.5	60.3	0.0	18.7
Ours w/ only vertical-slash	19.6	12.0	62.1	9.5	11.7	21.6	29.1	100.0	100.0	5.0	37.1

within local windows. The "only block-sparse" variant using only the *BS* pattern, also results in significant performance declines. On the other hand, "only vertical-slash" manages to preserve most of the performance due to its balance between dynamicity and the StreamingLLM pattern, but still fall behind the full version of our method.

Latency Fig. 1b and 10 shows the latency and breakdown of MInference across different context windows on a single A100. At 100K, 300K, 500K, and 1M tokens, our method achieves speedups of 1.8×, 4.1×, 6.8×, and 10×, respectively. It reduces the pre-filling latency from 30 mins to 3 mins on a single A100 for a prompt of 1M token. By further utilizing tensor parallel [LMZ+24] and context parallel [LZA24, JTZ+24], this latency can be reduced to 22 seconds on 8x A100 GPUs. This significantly lowers the deployment cost of long-context LLMs and enhances user experience. And since our kernel is implemented based on Triton, it can be easily ported to other devices and achieve similar speedups, such as on the H100 or MI300X. Additionally, analyzing the latency breakdown, we found about 5%-20% of the overhead is spent on dynamic sparse index building, while the remaining time is spent on dynamic sparse calculation.

Integrate with KV cache compression methods We also combined MInference with a state-of-the-art KV cache compression method SnapKV [LHY+24], as shown in Table 5. This proves our method is compatible with KV cache compression techniques. For most tasks, performance remains nearly unchanged, with the average score even showing a slight increase, which further demonstrates the potential practical value of our method as an optimization for serving long-context LLMs. This phenomenon is also observed in other works, such as ShadowKV [SCB+24].

Table 5: Performance of different methods on InfiniteBench [ZCH+24] using SnapKV [LHY+24] in the decoding stage.

Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
LLaMA-3 w/ SnapKV	18.0	**11.8**	65.5	2.5	12.0	21.3	26.6	**100.0**	**100.0**	1.8	36.0
Ours w/ SnapKV	**18.9**	11.7	**66.4**	**6.5**	**12.1**	**21.8**	**33.1**	**100.0**	**100.0**	**2.0**	**37.3**

Scaling-up on Larger LLMs We also evaluated MInference on larger LLMs, such as LLaMA-3-70B-1M[3]. As shown in Table 6, MInference maintains strong performance even in larger models. Notably, in dynamic tasks such as KV retrieval, MInference can match or even slightly improve performance compared to full attention. In contrast, baselines like InfLLM generally struggle with tasks such as KV retrieval.

Table 6: Performance of different methods using LLaMA-3-70B-Instruct-262K on InfiniteBench [ZCH+24].

Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
LLaMA-3-70B-262K	20.7	10.3	84.2	9.5	14.0	33.2	61.7	97.0	100.0	34.0	46.5
StreamingLLM	20.5	8.5	52.0	**10.0**	12.6	27.4	61.1	14.0	10.0	0.0	21.6
InfLLM	**24.1**	8.1	57.0	**10.0**	12.9	27.4	52.3	**100.0**	**100.0**	0.0	39.2
Ours	20.6	**10.1**	**83.4**	**10.0**	**14.1**	**34.1**	**61.9**	**100.0**	**100.0**	**39.0**	**47.3**

5 Related Works

Sparse Attention Due to the quadratic complexity of the attention mechanism, many previous works have focused on sparse attention to improve the efficiency of Transformers. These methods include static sparse patterns, cluster-based sparse approaches, and dynamic sparse attention.

[3] https://huggingface.co/gradientai/Llama-3-70B-Instruct-Gradient-262k

Static sparse patterns include techniques such as sliding windows [JSM$^+$23, AJA$^+$24], dilated attention [CGRS19, SGR$^+$21, DMD$^+$23], and mixed sparse patterns [BPC20, ZGD$^+$20, LCSR21]. Cluster-based sparse methods include hash-based [KKL20] and kNN-based [RSVG21, NŁC$^+$24] methods. All of the above methods require pre-training the model from scratch, which makes them infeasible to be directly used as a plugin for reay-to-use LLMs. Recently, there has been work [DG24, ZAW24] to unify state space models [GGR22, GD24, DG24], and linear attention [KVPF20, SDH$^+$23] into structured masked attention. Additionally, some works [WZH21, LQC$^+$22, RCHG$^+$24] leverage the dynamic nature of attention to predict sparse patterns dynamically. However, these approaches often focus on low-rank hidden states during the dynamic pattern approximation or use post-statistical methods to obtain the sparse mask, which introduce substantial overhead in the estimation step, making them less useful for long-context LLMs.

Scaling Context Windows of LLMs Recent research has focused on expanding the context window of pre-trained LLMs, that enables LLMs to handle more complex real-life applications [JYW$^+$23, POC$^+$23]. These methods can be categorized into: 1) Staged pre-training [NXH$^+$23, FPN$^+$24]; 2) Modifying or interpolating position embeddings [PSL22, CWCT23, PQFS24, DZZ$^+$24]; 3) Utilizing external memory modules for context storage [BANG23, TSP$^+$23, XZH$^+$24]; 4) Expanding computations across multiple devices in a distributed manner [LZA24]. However, these methods do not alleviate the high inference costs in long-context processing.

Long-Context LLM Inference Recent studies [Fu24] have tackled the high computational cost of attention and substantial KV cache storage in long-context scenarios from two angles: pre-filling and decoding. Pre-filling optimizations are primarily categorized as State Space Models [GGR22, GD24], linear attention methods [SDH$^+$23, PAA$^+$23], memory-based methods [MFG24, HBK$^+$24], hybrid methods [LLB$^+$24, RLL$^+$24], and prompt compression methods [LDGL23, JWL$^+$23, JW$^+$24, PWJ$^+$24]. However, these approaches require training from scratch or additional overhead and are difficult to implement directly in pretrained long-context LLMs. Recently, some studies [MEL24, XZH$^+$24, LCL$^+$24] have focused on using kNN or cluster-based sparse attention to accelerate LLM inference. However, these methods often lead to reduced accuracy, limited speedup, or are restricted to CPU scenarios.

In contrast, optimizations for the decoding stage are divided into [LJW$^+$24]: 1) Reusing attention KV to reduce KV cache storage [Sha19, ALTdJ$^+$23, SDZ$^+$24, DA24, NŁC$^+$24]; 2) Static KV cache dropping [XTC$^+$24, HWP$^+$24]; 3) Dynamic KV cache dropping [ZSZ$^+$24, LDL$^+$24, GZL$^+$24, OHY$^+$24, LHY$^+$24, APB$^+$24]; 4) Dynamic KV cache offloading [RCHG$^+$24, DHJ$^+$24, TZZ$^+$24, LCL$^+$24, CSY$^+$24, SCB$^+$24]; 5) Methods for restoring performance loss due to KV cache compression [AAJ$^+$24, DYZ$^+$24]; 6) Hierarchical speculative decoding methods [SCY$^+$24, CTS$^+$24]; 7) KV cache quantitation [LYJ$^+$24]. Nevertheless, these methods do not address the heavy computational burden of the attention in the pre-filling stage.

6 Conclusion

This paper addresses the expensive computational cost and the unacceptable latency of the attention calculations in the pre-filling stage of long-context LLMs. We propose MInference, a method that accelerates the pre-filling stage by leveraging dynamic sparse attention with spatial aggregation patterns. Specifically, we categorize attention heads into three types: *A-shape*, *Vertical-Slash*, and *Block-Sparse*. Using a kernel-aware optimal sparse pattern search method, we identify the optimal pattern for each head. Subsequently, we utilize a fast approximation approach to build dynamic sparse masks for different inputs, and then apply these mask to perform sparse attention calculations. Experimental results on benchmarks such as InfiniteBench, RULER, language modeling, and Needle In A Haystack demonstrate that our method effectively maintains the long-context capabilities of LLMs while achieving up to a 10× speedup, reducing the latency from 30 minutes to 3 minutes per prompt for 1 million token prompts on a single A100 GPU. Additionally, we have found that similar dynamic sparse attention patterns also exist in both multi-modal LLMs [WWL$^+$24] and encoder-decoder LLMs [RSR$^+$20]. Using MInference for pre-filling stage inference acceleration holds great promise.

References

[AAJ+24] Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127, 2024.

[AJA+24] Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Caio César Teodoro Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, Allie Del Giorno, Gustavo de Rosa, Matthew Dixon, Ronen Eldan, Dan Iter, Amit Garg, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Jamie Huynh, Mojan Javaheripi, Xin Jin, Piero Kauffmann, Nikos Karampatziakis, Dongwoo Kim, Mahoud Khademi, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Chen Liang, Weishung Liu, Eric Lin, Zeqi Lin, Piyush Madan, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Xia Song, Masahiro Tanaka, Xin Wang, Rachel Ward, Guanhua Wang, Philipp Witte, Michael Wyatt, Can Xu, Jiahang Xu, Sonali Yadav, Fan Yang, Ziyi Yang, Donghan Yu, Chengruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

[ALTdJ+23] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. 2023.

[APB+24] Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurelien Lucchi, and Thomas Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 2024.

[BANG23] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[BBC+23] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *ArXiv preprint*, abs/2309.16609, 2023.

[BPC20] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *ArXiv preprint*, abs/2004.05150, 2020.

[BSK+24] Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Arun Iyer, Suresh Parthasarathy, Sriram Rajamani, B Ashok, and Shashank Shet. Codeplan: Repository-level coding using llms and planning. *Proceedings of the ACM on Software Engineering*, 1(FSE):675–698, 2024.

[CGRS19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *ArXiv preprint*, abs/1904.10509, 2019.

[CPG+23] Avi Caciularu, Matthew E Peters, Jacob Goldberger, Ido Dagan, and Arman Cohan. Peek across: Improving multi-document modeling via cross-document question-answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1989, 2023.

[CSY+24] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm generation. *ArXiv preprint*, abs/2410.16179, 2024.

[CTS+24] Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *ArXiv*, abs/2408.11049, 2024.

[CWCT23] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *ArXiv preprint*, abs/2306.15595, 2023.

[CWW+24] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024.

[DA24] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

[Dao24] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[DG24] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning*, 2024.

[DHJ+24] Jincheng Dai, Zhuowei Huang, Haiyun Jiang, Chen Chen, Deng Cai, Wei Bi, and Shuming Shi. Sequence can secretly tell you what to discard. *ArXiv preprint*, abs/2404.15949, 2024.

[DMD+23] Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. Longnet: Scaling transformers to 1,000,000,000 tokens. *ArXiv preprint*, abs/2307.02486, 2023.

[DSY24] Yichuan Deng, Zhao Song, and Chiwun Yang. Attention is naturally sparse with gaussian distributed input. *ArXiv preprint*, abs/2404.02690, 2024.

[DYZ+24] Harry Dong, Xinyu Yang, Zhenyu Zhang, Zhangyang Wang, Yuejie Chi, and Beidi Chen. Get more with LESS: Synthesizing recurrence with KV cache compression for efficient LLM inference. In *Forty-first International Conference on Machine Learning*, 2024.

[DZZ+24] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. LongroPE: Extending LLM context window beyond 2 million tokens. In *Forty-first International Conference on Machine Learning*, 2024.

[FPN+24] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *Forty-first International Conference on Machine Learning*, 2024.

[Fu24] Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *ArXiv preprint*, abs/2405.08944, 2024.

[GD24] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

[GGR22] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[Gra24] Gradient. Llama-3 8b instruct gradient 4194k (v0.1), 2024.

[GZL+24] 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. In *The Twelfth International Conference on Learning Representations*, 2024.

[GZX+24] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *ArXiv preprint*, abs/2406.12793, 2024.

[HBK+24] Namgyu Ho, Sangmin Bae, Taehyeon Kim, hyunjik.jo, Yireun Kim, Tal Schuster, Adam Fisch, James Thorne, and Se-Young Yun. Block transformer: Global-to-local language modeling for fast inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[HSK+24] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[HWP+24] 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. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico, 2024. Association for Computational Linguistics.

[JSM+23] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *ArXiv preprint*, abs/2310.06825, 2023.

[JTZ+24] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Reza Yazdani Aminadabi, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. System optimizations for enabling training of extreme long sequence transformer models. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, pages 121–130, 2024.

[JW+24] Huiqiang Jiang, Qianhui Wu, , Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[JWL+23] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore, 2023. Association for Computational Linguistics.

[JYW+23] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2023.

[Kam23] Greg Kamradt. Needle in a haystack - pressure testing llms, 2023.

[KKL20] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[KVPF20] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 2020.

[LCL+24] 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*, 2409.10516, 2024.

[LCSR21] François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics.

[LCW21] Valerii Likhosherstov, Krzysztof Choromanski, and Adrian Weller. On the expressive power of self-attention matrices. *ArXiv preprint*, abs/2106.03764, 2021.

[LCW23] Valerii Likhosherstov, Krzysztof Choromanski, and Adrian Weller. On the expressive flexibility of self-attention matrices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(7):8773–8781, 2023.

[LDGL23] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, 2023.

[LDL+24] 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, 2024.

[LHY+24] 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. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[LJW+24] Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. *arXiv preprint arXiv:2412.10319*, 2024.

[LLB+24] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *ArXiv preprint*, abs/2403.19887, 2024.

[LLWL24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[LMZ+24] Zhiqi Lin, Youshan Miao, Quanlu Zhang, Fan Yang, Yi Zhu, Cheng Li, Saeed Maleki, Xu Cao, Ning Shang, Yilei Yang, Weijiang Xu, Mao Yang, Lintao Zhang, and Lidong Zhou. nnscaler: Constraint-guided parallelization plan generation for deep learning training. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, 2024.

[LQC+22] Liu Liu, Zheng Qu, Zhaodong Chen, Fengbin Tu, Yufei Ding, and Yuan Xie. Dynamic sparse attention for scalable transformer acceleration. *IEEE Transactions on Computers*, 71(12):3165–3178, 2022.

[LWD+23] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR, 2023.

[LYJ+24] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning*, 2024.

[LYZA24] Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with ringattention. *ArXiv preprint*, abs/2402.08268, 2024.

[LZA24] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ringattention with blockwise transformers for near-infinite context. In *The Twelfth International Conference on Learning Representations*, 2024.

[LZD$^+$24] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *ArXiv preprint*, abs/2404.02060, 2024.

[MEL24] Yuzhen Mao, Martin Ester, and Ke Li. Iceformer: Accelerated inference with long-sequence transformers on CPUs. In *The Twelfth International Conference on Learning Representations*, 2024.

[MFG24] Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *ArXiv preprint*, abs/2404.07143, 2024.

[MJ23] Amirkeivan Mohtashami and Martin Jaggi. Random-access infinite context length for transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[NŁC$^+$24] Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Forty-first International Conference on Machine Learning*, 2024.

[NXH$^+$23] Erik Nijkamp, Tian Xie, Hiroaki Hayashi, Bo Pang, Congying Xia, Chen Xing, Jesse Vig, Semih Yavuz, Philippe Laban, Ben Krause, Senthil Purushwalkam, Tong Niu, Wojciech Kryściński, Lidiya Murakhovs'ka, Prafulla Kumar Choubey, Alex Fabbri, Ye Liu, Rui Meng, Lifu Tu, Meghana Bhat, Chien-Sheng Wu, Silvio Savarese, Yingbo Zhou, Shafiq Joty, and Caiming Xiong. Xgen-7b technical report. *ArXiv preprint*, abs/2309.03450, 2023.

[OHY$^+$24] Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[Ope24] OpenAI. Learning to reason with llms, 2024.

[PAA$^+$23] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the ACL: EMNLP 2023*, pages 14048–14077, Singapore, 2023. Association for Computational Linguistics.

[POC$^+$23] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

[PPJF24] Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Fast attention over long sequences with dynamic sparse flash attention. *Advances in Neural Information Processing Systems*, 36, 2024.

[PQFS24] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[PSL22] Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[PWJ+24] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the ACL 2024*, pages 963–981, Bangkok, Thailand and virtual meeting, 2024. Association for Computational Linguistics.

[RCHG+24] Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient LLM inference. In *Forty-first International Conference on Machine Learning*, 2024.

[RLL+24] Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *ArXiv preprint*, abs/2406.07522, 2024.

[RPJ+20] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[RST+24] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv preprint*, abs/2403.05530, 2024.

[RSVG21] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[SCB+24] 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*, abs/2410.21465, 2024.

[SCY+24] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. In *First Conference on Language Modeling*, 2024.

[SDH+23] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *ArXiv preprint*, abs/2307.08621, 2023.

[SDZ+24] Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[SGR+21] Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James Tin-Yau Kwok. Sparsebert: Rethinking the importance analysis in self-attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9547–9557. PMLR, 2021.

[Sha19] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *ArXiv preprint*, abs/1911.02150, 2019.

[TDT$^+$23] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*, 2023.

[TKC19] Philippe Tillet, Hsiang-Tsung 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*, pages 10–19, 2019.

[tri23] Triton implementation of the flash attention v2. Technical report, OpenAI, 2023.

[TSP$^+$23] Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. Focused transformer: Contrastive training for context scaling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[TZZ$^+$24] 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.

[Wen23] Lilian Weng. Llm-powered autonomous agents. *lilianweng.github.io*, 2023.

[WWL$^+$24] Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. LOOK-M: Look-once optimization in KV cache for efficient multimodal long-context inference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4065–4078, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[WZH21] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE, 2021.

[XTC$^+$24] 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*, 2024.

[XZH$^+$24] 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*, 2024.

[YCL$^+$24] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *ArXiv preprint*, abs/2403.04652, 2024.

[ZAW24] Itamar Zimerman, Ameen Ali, and Lior Wolf. A unified implicit attention formulation for gated-linear recurrent sequence models. *ArXiv preprint*, abs/2405.16504, 2024.

[ZCH$^+$24] Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[ZGD$^+$20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[ZJZ$^+$23] Ningxin Zheng, Huiqiang Jiang, Quanlu Zhang, Zhenhua Han, Lingxiao Ma, Yuqing Yang, Fan Yang, Chengruidong Zhang, Lili Qiu, Mao Yang, et al. Pit: Optimization of dynamic sparse deep learning models via permutation invariant transformation. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 331–347, 2023.

[ZSZ$^+$24] 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, 2024.

A Limitations

As the context length decreases, the time required to build the dynamic index becomes more significant as attention computation time decreases. For example, with a 10k context, the time spent on building the index increases from 5% to 30%, resulting in overall end-to-end latency approaching that of FlashAttention. However, this overhead proportion gradually decreases as the prompt lengthens. Additionally, when using a higher sparsity rate, the model performance may noticeably decline.

B Broader Impacts

MInference effectively accelerates the inference of long-context LLMs, facilitating their deployment and application. By enabling lower latency, it can reduce the deployment costs of LLMs, especially for long-context LLMs, helping to democratize access to advanced AI. It also promotes further research and development in related fields.

C Experiment Details

C.1 Dataset Details

InfiniteBench [ZCH$^+$24] includes 10 tasks designed to test various aspects of long-context processing. Specifically, these tasks cover entire novel summarization, open-form question answering based on novels, multiple-choice question answering on novels, question answering on long drama scripts, question answering on Chinese texts, debugging large code repositories, identifying the largest/smallest number in arrays, and retrieval tasks with varying pattern lengths. The average token length for these tasks is 214k, and they include 3,992 examples.

RULER [HSK$^+$24] is a recent synthetic benchmark suite for long-context evaluation with 13 complex tasks across four categories. The retrieval category includes Single Needle-in-a-Haystack (S-NIAH), where a single key-value pair is inserted into noisy text, and the model must retrieve the value. Multi-keys Needle-in-a-Haystack (MK-NIAH) involves multiple keys, and the model retrieves one specific value among hard distractors. The Multi-values Needle-in-a-Haystack (MV-NIAH) task requires retrieving all values associated with a single key, while the Multi-queries Needle-in-a-Haystack (MQ-NIAH) task involves retrieving values for multiple keys. The Multi-hop Tracing category includes Variable Tracking (VT), where the model traces and returns all variable names pointing to the same value through variable bindings. The aggregation category introduces Common Words Extraction (CWE), where the model identifies the top-K common words from a mixture of common and uncommon words, and Frequent Words Extraction (FWE), where the model identifies the most frequent words from a Zeta distribution. The Question Answering (QA) category extends existing short-context QA datasets by adding distracting paragraphs, challenging the model to answer questions based on relevant information surrounded by distractors. These tasks provide a comprehensive evaluation of long-context modeling capabilities, covering multi-hop reasoning, aggregation, and complex question answering. Following [HSK$^+$24], we test models on 4K, 8K, 16K, 32K, 64K, and 128K context lengths, including 2,600 examples per length.

Needle In A Haystack task [Kam23] evaluates the performance of retrieval-augmented generation (RAG) systems by embedding specific, targeted information (the "needle") within a large, complex body of text (the "haystack"). The test assesses a language model's ability to identify and utilize this specific piece of information amidst a vast amount of data. Both RULER and the needle test iterate over various context lengths and document depths (where the ground-truth is placed in the prompt) to measure the long-context performance. Here we scale the Needle In A Haystack task to 1M context length, including 750 examples.

PG-19 [RPJ$^+$20] The perplexity on long text is also often used by researchers to evaluate the language modeling performance of long-context LLMs. PG-19 is a suitable test set for this task, as it includes texts as long as 500K tokens. Perplexity is used as the metric indicating how well a model predicts the next token in a sequence. Our experiments are conducted on 1,000 random samples from PG-19 that are longer than 100K tokens.

C.2 Additional Implementation Details

Our experiments are based on a number of state-of-the-art long-context LLMs: 1) LLaMA-3-8B-Instruct-262k[4] is a LLaMA-3 variant with further NTK-aware interpolation and minimal fine-tuning with Ring Attention, which achieved SOTA results on long-context assessments such as the Needle In A Haystack test; 2) LLaMA-3-8B-Instruct-1048k[5] is similar to LLaMA-3-8B-Instruct-262k, but supports context lengths up to 1M tokens; 3) Yi-9B-200K [YCL$^+$24] is a SOTA LLM that balances long-context performance with general capabilities; 4) Phi-3-Mini-128K [AJA$^+$24] a small but powerful language model that offers capabilities equivalent to models ten times its size with up to 128K context window powered by LongRoPE [DZZ$^+$24]; 5) Qwen2-7B-128K [BBC$^+$23] is a recently release update of Qwen series model with up to 128K context window that achieve superior or comparable performance compared to LLaMA-3; 6) GLM-4-9B-1M [GZX$^+$24] has been improved from its predecessor in terms of a 1M context window, performance on downstream tasks and inference efficiency. To guarantee stable results, we use greedy decoding in all tests. Our kernel implementations are developed and optimized based on the dynamic sparse compiler PIT [ZJZ$^+$23] in the Triton language [TKC19]. The latency experiments are done on a single Nvidia A100 GPU using bfloat16. We provide a simple custom implementation of attention in PyTorch, building on FlashAttention and Triton.

We set the target FLOPs t to be the same as 1k global tokens and 4k local window tokens in the *A-shape* pattern. The step size of ChangeSpace is set to 50, with the corresponding search space shown in Table 7. Additionally, we use only one sample as our validation set from KV retrieval synthetic data with 30k token inputs, which exhibits strong generalization and stability across different lengths and domains. The search time is approximately 15 minutes on a single A100. Additionally, we use the same optimal sparse pattern configuration for both the LLaMA-3-8B-Instruct-262K model and the LLaMA-3-8B-Instruct-1M model. The specific distribution is shown in Fig. 11.

Table 7: Kernal-aware optimal head pattern search space. In this context, *A-shape* represents the global tokens and local window number, *Vertical-Slash* represents the Top-K number of vertical and diagonal lines, and *Block-Sparse* represents the Top-K number of blocks retained.

Patterns	Search Space
A-shape	$\{(1024, 4096)\}$
Vertical-Slash	$\{(30, 2048), (100, 1800), (500, 1500), (3000, 200)\}$
Block-Sparse	$\{100\}$

C.3 Single A100 Implementation Details

The original PyTorch implementation[6] of the LLaMA model causes an out-of-memory error on a single A100 (80G) when the prompt exceeds 50k tokens. To enable running 1M prompt inference on a single A100, we implemented the following optimizations:

1. **Tensor Splitting**: We split the Attention by head and the MLP by sequence dimension. In long-context scenarios, where computation is the bottleneck, this splitting keeps GPU utilization at 100%, and the overhead of splitting is negligible;
2. **Reduction of Intermediate Variables**: We minimized intermediate variable allocation by removing the attention mask and implementing causal mask logic directly within the kernel;
3. **Elimination of Unnecessary Computations**: In long-context scenarios, only the logits corresponding to the last token in the prompt phase are meaningful. Thus, we only retain the computation of the LM Head Linear layer for the last token.

[4] https://huggingface.co/gradientai/Llama-3-70B-Instruct-Gradient-262k

[5] https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k

[6] https://github.com/huggingface/transformers/blob/main/src/transformers/models/llama/modeling_llama.py

C.4 Kernel Implementation

C.4.1 Block-Sparse Flash Attention

Our *Block-Sparse* kernel implementation is based on the Triton version of the FlashAttention kernel [tri23]. With the selected block index as an additional input, each thread block loops through the top-K blocks in a row. As discussed in FlashAttention [Dao24], the latency of the block-sparse FlashAttention kernel is linearly related to the number of blocks, and the speedup ratio (compared to the dense FlashAttention kernel) is approximately as,

$$s_p = \frac{S}{2B \times k_b} \tag{3}$$

C.4.2 Vertical-Slash Attention

The *Vertical-Slash* attention includes two custom kernels: the *Vertical-Slash* sparse index kernel and the *Vertical-Slash* sparse FlashAttention kernel.

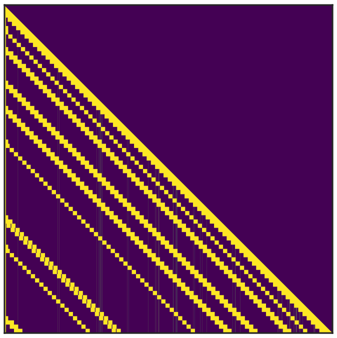


Figure 7: The dynamic sparse mask for the vertical-slash pattern using LLaMA-3-8B in the summarization task [ZCH+24]. Yellow areas indicate the computed parts. Slash lines use 64×64 blocks, while vertical lines use 1×64 blocks.

The *Vertical-Slash* sparse index kernel in Algorithm 4 builds the index for each row of blocks. Since a slash line segment can be masked by a square block, our attention mask is a mix of blocks and columns, as shown in Fig. 7. We apply a point-range two-way merge algorithm where vertical indexes are treated as points and slash indexes are converted to ranges given the row index. The output consists of two parts: merged ranges and separate column indexes, where the ranges are represented by block indexes. The time complexity to build an index for a row is $O(k_v + k_s)$.

The *Vertical-Slash* sparse FlashAttention kernel in Algorithm 5 is a mix of the block-sparse attention kernel and the PIT [ZJZ+23] sparse attention kernel. PIT is a technology that loads sparse data into dense compute blocks via a Permutation Invariant Transformation. A thread block first loops through the block indexes as described in the previous section (block part) and then loops through the column indexes grouped by block size (PIT part). The latency of this hybrid kernel is linearly related to the total area of blocks and columns.

D Additional Experiment Results

D.1 Needle In A Haystack

In addition to the Needle In A Haystack results for LLaMA-3-Instruct-1M shown in §4, we also present the LLaMA-3-Instruct-1M using InfLLM results in Fig. 8, and results for GLM-4-9B-1M, Yi-9B-200K, Phi-3-Mini-128K, and Qwen2-7B-128K, shown in Fig. 9. Compared to Full Attention, using MInference has minimal impact on the ability to understand semantic information across different context windows and

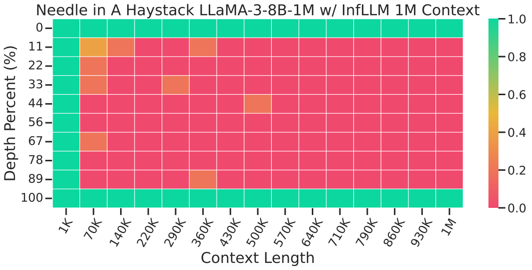
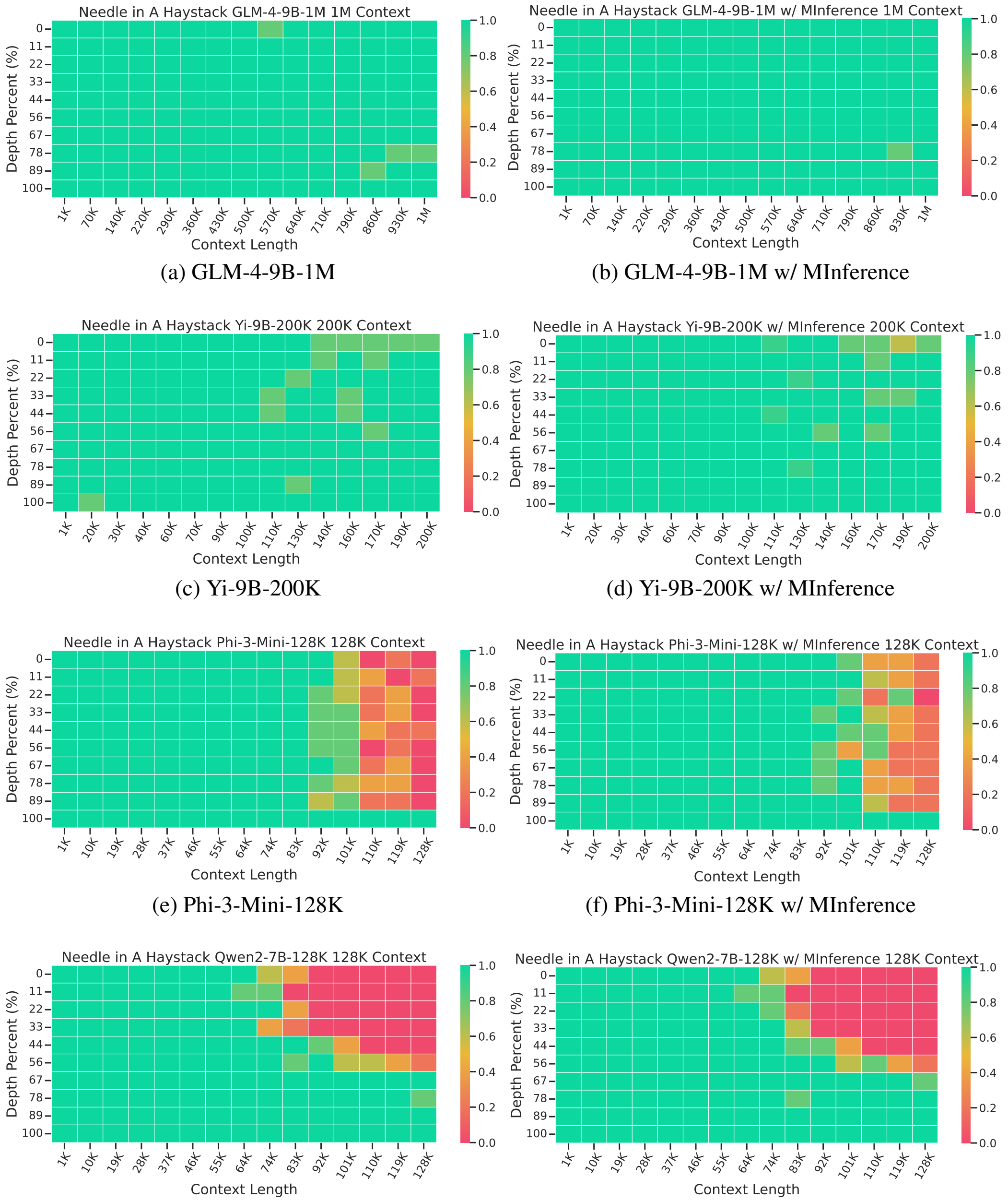


Figure 8: Results on Needle In A Haystack using InfLLM in LLaMA-3-8B-Instruct-1M.

needle depths. There is even a slight performance improvement around the 100k context length using Yi-9B-200K and Phi-3-Mini-128K.



(a) GLM-4-9B-1M

(b) GLM-4-9B-1M w/ MInference

(c) Yi-9B-200K

(d) Yi-9B-200K w/ MInference

(e) Phi-3-Mini-128K

(f) Phi-3-Mini-128K w/ MInference

(g) Qwen2-7B-128K

(h) Qwen2-7B-128K w/ MInference

Figure 9: Needle In A Haystack [Kam23] results using GLM-4-9B-1M [GZX+24], Yi-9B-200K [YCL+24], Phi-3-Mini-128K [AJA+24], and Qwen2-7B-128K [BBC+23].

D.2 Latency Breakdown

Fig. 10 shows the micro-benchmark results of the three attention patterns proposed in this paper, as well as FlashAttention. It can be seen that Vertical-Slash is the slowest among the three patterns, but it still achieves a 13x speedup compared to FlashAttention under 1M context windows. A-shape is slightly faster than Vertical-Slash, but at 1M, A-shape is 50% slower than Vertical-Slash. Block-Sparse is the fastest, achieving a 30x speedup over FlashAttention under 1M context windows.

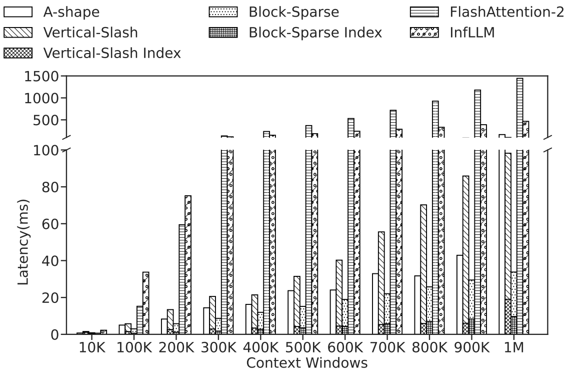


Figure 10: The latency breakdown of a single attention kernel for three patterns and FlashAttention [Dao24] across different context windows in a single A100, including the index time for dynamic sparse approximation and building dynamic sparsity. At 10k tokens, the latency of the four kernels is very close and all are less than 1ms. At 1M tokens, the latency for A-shape is 164ms.

The estimation and index-building time for the dynamic sparse pattern accounts for approximately 5%-15% and 25% of the total time for Vertical-Slash and Block-Sparse patterns, respectively. The index-building overhead is higher for Block-Sparse mainly due to the time-consuming MeanPooling and block-level matmul computations. Additionally, the memory overhead for sparse indexing is relatively small, remaining within 160MB for a LLaMA-3-8B model in 1M context.

D.3 Additional Ablation Study

Table 8: Performance of different ablation methods using LLaMA-3-8B-Instruct-262K on InfiniteBench [ZCH+24]. It is important to note that due to kernel limitations, we must retain at least one vertical and one slash. Therefore, "ours w/ only vertical" retains the top-1 slash, and "ours w/ only slash" retains the top-1 vertical.

Methods	En.Sum	En.QA	En.MC	En.Dia	Zh.QA	Code.Debug	Math.Find	Retr.PassKey	Retr.Num	Retr.KV	Avg.
Ours	20.5	12.9	65.9	7.5	12.5	22.3	33.1	100.0	100.0	12.8	38.8
Ours w/ only vertical	13.7	6.2	30.1	2.0	6.5	7.9	1.7	65.4	52.7	0.0	18.6
Ours w/ only slash	18.4	11.5	60.1	3.0	11.4	22.1	28.4	100.0	100.0	4.2	35.9

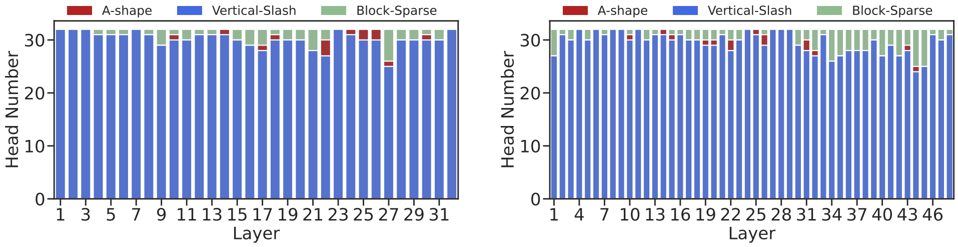
To further analyze the role of dynamic vertical and slash lines in the *Vertical-Slash* pattern for sparse computation, we introduce a new set of ablation studies as follows: 1) Ours w/ only vertical, which only uses vertical lines and the top-1 slash line in *Vertical-Slash* pattern. 2) Ours w/ only slash, which only uses slash lines and the top-1 vertical line in *Vertical-Slash* pattern. The corresponding top-K quantities are set after converting based on FLOPs in kernel.

As shown in Table 8, using only vertical lines results in a significant performance drop, especially in retrieval tasks, where performance is similar to only using block-sparse. In contrast, using only slash lines retains most of the performance, but in highly dynamic tasks such as KV retrieval, performance further decreases, with an average performance drop of 2.9% compared to Ours.

E Pattern Distribution

Fig. 11 shows the distribution of the optimal head configuration obtained through our search. Firstly, most of the patterns are the *Vertical-Slash* pattern (>90%). However, according to the ablation study, using only the *Vertical-Slash* pattern significantly impacts performance in highly dynamic tasks like KV retrieval. Secondly, the *Block-Sparse* pattern is primarily distributed in several intermediate to later layers, while the *A-shape* pattern is found in the middle layers. Although the optimal patterns vary slightly across different models, they generally align with these observations.

Additionally, we used the same configuration for two versions of LLaMA in our experiments, and the results show that the 1M model also performs very well, with nearly perfect results in the Needle In A Haystack task. This demonstrates the generalizability of the optimal sparse pattern.



(a) LLaMA-3-8B-Instruct-262K/1M

(b) Yi-9B-200K

Figure 11: Distribution of three sparse head patterns in different models. We use the same optimal sparse pattern configuration for both LLaMA-3-8B-Instruct-262K and LLaMA-3-8B-Instruct-1M.

F Sparsity in Kernel Distribution

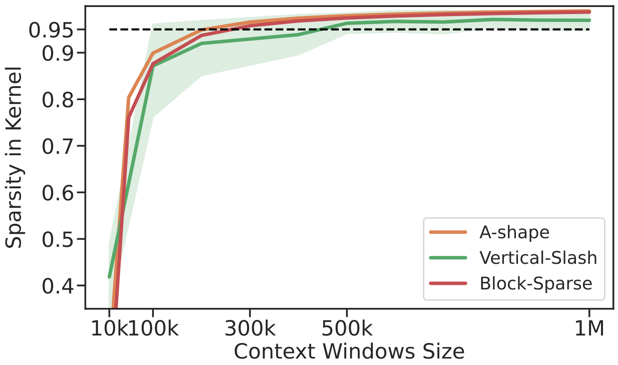


Figure 12: The distribution of sparsity in the kernel across different context windows refers to the proportion of the kernel that is actually computed after block coverage, compared to the sparsity rate when using FlashAttention with a causal mask.

As shown in Fig. 12, the sparsity distribution of the three patterns during the actual kernel computation process is displayed. It can be seen that when the context windows exceed 200k, the actual sparsity of all three patterns surpasses 90%. Even considering a 20% index-building overhead, this ensures that the kernel achieves a speedup of over 8×. Furthermore, when the context windows exceed 500k, the sparsity relative to FlashAttention exceeds 95%, with a theoretical speedup of over 15×.

G Does This Dynamic Sparse Attention Pattern Exist Only in Auto-Regressive LLMs or RoPE-Based LLMs?

Similar vertical and slash line sparse patterns have been discovered in BERT [SGR$^+$21] and multi-modal LLMs [WWL$^+$24]. Additionally, as shown in Fig. 13, we analyzed the distribution of attention patterns in T5 across different heads. It is evident that there are vertical and slash sparse patterns even in bidirectional attention.

Recent studies [WWL$^+$24] have analyzed sparse attention patterns in multi-modal LLMs, revealing the presence of vertical and slash patterns in models like LLaVA [LLWL24] and InternVL [CWW$^+$24]. Using MInference for pre-filling stage inference acceleration holds great promise.

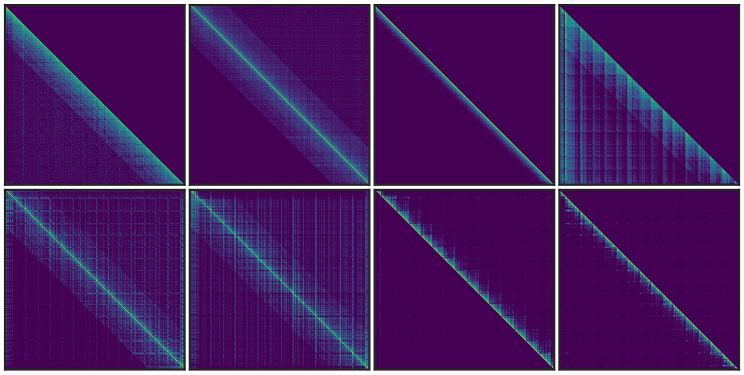


Figure 13: The sparse pattern in T5-style Encoder Attention using Flan-UL2 [TDT$^+$23] on the Summarization dataset [ZCH$^+$24].

H Case Study

Table 9 presents a comparison of the generation performance for various methods on the EN.SUM task (200K input length) from InfiniteBench based on the LLaMA-3-8B-262K model. The original summary provides a comprehensive and coherent narrative, detailing the Bronwyn family's trip to the Kindergarten and touching on themes such as nostalgia, loss, and the passage of time. StreamingLLM's summary, although looks coherent, introduces elements that are not present in the original story, leading to serious factual errors. For example, it mentions a boat trip to a school for boys and specific details like fishermen, sandwiches, and a spot where men were drowned. These details deviate from the original story, which is about the Bronwyn family preparing for a trip to the Kindergarten. In addition, the summaries generated by StreamingLLM with dilated and strided techniques are largely incoherent, consisting primarily of repetitive and nonsensical characters, indicating a failure to produce meaningful content. In stark contrast, the summary generated by our proposed method offers a detailed and coherent narrative, comparable to the original, with a clear depiction of the story's main events and themes. This includes the preparation of the Bronwyn family for their trip, the characterization of family members and guests, and the exploration of deeper themes such as love, marriage, and the search for meaning. The results demonstrate the superiority of our proposed method in generating high-quality, human-like summaries over the baseline methods.

Table 10 compares the performance of various methods on the Retrieve.KV task (200K input length) using the LLaMA-3-8B-262K model. The original method demonstrates perfect retrieval, correctly predicting the exact strings of the ground truth for both examples. StreamingLLM, again, generates predictions that looks coherent and real, but factually incorrect. In addition, StreamingLLM with dilated and strided techniques, and our method with a static pattern, fail significantly, producing outputs that are either repetitive sequences of characters or nonsensical strings, indicating their inability to accurately retrieve the required key-value pairs. Our method, however, performs on par with the original, accurately retrieving and predicting the exact key-value pairs for both examples. This demonstrates the superior capability of our method in handling KV retrieval tasks, providing precise and reliable outputs consistent with the ground truth. The results highlight our method's effectiveness and robustness compared to the baselines, making it a reliable choice for such tasks.

Table 9: Comparison of generated results by different methods using LLaMA-3-8B-Instruct-262K in summarization task [ZCH$^+$24].

Methods	Summary
Original	The story follows the Bronwyn family and their guests as they prepare for a day trip to the Kindergarten, a mysterious destination that holds great significance for them. The story is told from multiple perspectives, including that of Pete Lucy, who is staying with the family and is struggling to find her place in their lives. As the day progresses, the family members prepare for their trip, and Pete reflects on her own past experiences with the Bronwyns. The story ends with the family setting off on their journey, leaving Pete alone on the lawn to contemplate her own feelings and the meaning of life. The Kindergarten is a symbol of the family's past and their memories, and the story explores themes of nostalgia, loss, and the passage of time. Ultimately, the story suggests that even though the past cannot be changed, it can still be celebrated and remembered.
StreamingLLM	The story follows a family on a boat trip to the Kindergarten, a school for boys. The father, Mr Bronwyn, is reading a book and seems to be in a trance. The children, Seb and Annalisa, are excited about the trip but also a little scared. They pass by a group of fishermen and Mr Bronwyn shares out sandwiches. They come across a spot where three men were drowned and Mr Bronwyn takes out his watch, making a calculation. They reach the Kindergarten and Mr Bronwyn jumps out of the boat, holding a parcel. The story ends with Pete Lucy, who has been watching the family from the shore, feeling relieved that the trip is over and that Mr Bronwyn has reached his destination. The story is told from multiple perspectives, including Pete Lucy's, who is watching the family from the shore and reflecting on his own life and relationships.
StreamingLLM w/ dilated	A (
StreamingLLM w/ strided	The GraphicsUnit36.OOAKA new (
Ours w/ static	def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def
Ours	The story follows the Bronwyn family and their guests as they prepare for a trip to the Kindergarten, a remote island off the coast of Scotland. The family is made up of Mr. Bronwyn, a philosopher, and his wife, Mrs. Bronwyn, who is known for her beauty and her ability to surround and protect her family. The guests include Kevin Emma, a young man who is studying for a fellowship, and Castor Kiki, a botanist who is also a widower. The story explores themes of love, marriage, and the search for meaning in life. It also touches on the idea of the other, or the person who is different from oneself and cannot be understood or judged by conventional standards. Ultimately, the story is about the search for truth and the desire to connect with others on a deeper level.

Algorithm 4 Vertical-Slash Index

```
Input: vertical indexes $\boldsymbol{i}_v \in \mathbb{N}^{k_v}$, slash indexes $\boldsymbol{i}_s \in \mathbb{N}^{k_s}$
# Sort vertical and slash indexes
$\boldsymbol{i}_v \leftarrow \text{IncrementalSort}\left(\boldsymbol{i}_v\right)$
$\boldsymbol{i}_s \leftarrow \text{DescendingSort}\left(\boldsymbol{i}_s\right)$
# Calculate block number (block_size $B$)
$N \leftarrow \lceil \frac{S}{B} \rceil$
# Initialize outputs
block count $\boldsymbol{c}_{\text{blk}} \in \mathbb{N}^N$, block index $\boldsymbol{i}_{\text{blk}} \in \mathbb{N}^{N \times k_s}$, column count $\boldsymbol{c}_{\text{col}}\mathbb{N}^N$, column index $\boldsymbol{i}_{\text{col}} \in \mathbb{N}^{N \times k_v}$
# Parallelized in GPU
for $i \leftarrow 1$ to $N$ do
  $j_v \leftarrow 1$
  # Find the first slash line that crosses the row
  $j_s \leftarrow \text{biset\_left}(\boldsymbol{i}_s, i \times B)$
  # Define the range by slash index
  $r_{\text{start}} \leftarrow (i-1) \times B - \boldsymbol{i}_s^{j_s}$
  $r_{\text{end}} \leftarrow i \times B - \boldsymbol{i}_s^{j_s}$
  # Merge points (vertical indexes) and ranges (slash indexes)
  while $s_v \leq k_s$ do
    if $j_v \leq k_v$ and $\boldsymbol{i}_v^{j_v} < r_{\text{end}}$ then
      # Record the point if not in the range
      if $\boldsymbol{i}_v^{j_v} < r_{\text{start}}$ then
        $\boldsymbol{c}_{\text{col}}^i \leftarrow \boldsymbol{c}_{\text{col}}^i + 1, \boldsymbol{i}_{\text{col}}^{i, \boldsymbol{c}_{\text{col}}^i} \leftarrow \boldsymbol{i}_v^{j_v}$
      end
      $j_v \leftarrow j_v + 1$
    else
      $s_v \leftarrow s_v + 1$
      # Update the range
      if $(i-1) \times B - \boldsymbol{i}_s^{j_s} > r_{\text{end}}$ then
        # Record the last range
        $s \leftarrow r_{\text{start}}$
        while $s < r_{\text{end}}$ do
          $\boldsymbol{c}_{\text{blk}}^i \leftarrow \boldsymbol{c}_{\text{blk}}^i + 1$
          $\boldsymbol{i}_{\text{blk}}^{i, \boldsymbol{c}_{\text{blk}}^i} \leftarrow s$
          $s \leftarrow s + B$
        end while
        # Calculate the new range
        $r_{\text{start}} \leftarrow (i-1) \times B - \boldsymbol{i}_s^{j_s}$
        $r_{\text{end}} \leftarrow i \times B - \boldsymbol{i}_s^{j_s}$
      else
        # Extend the range
        $r_{\text{end}} \leftarrow r_{\text{end}} + B$
      end
    end
  end while
  # Record the last range
  $s \leftarrow r_{\text{start}}$
  while $s < r_{\text{end}}$ do
    $\boldsymbol{c}_{\text{blk}}^i \leftarrow \boldsymbol{c}_{\text{blk}}^i + 1$
    $\boldsymbol{i}_{\text{blk}}^{i, \boldsymbol{c}_{blk}^i} \leftarrow s, s \leftarrow s + B$
  end while
end for
return $\boldsymbol{c}_{\text{blk}}, \boldsymbol{i}_{\text{blk}}, \boldsymbol{c}_{\text{col}}, \boldsymbol{i}_{\text{col}}$
```

Algorithm 5 Vertical-Slash Flash Attention

```
Input: $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{S \times d_h}$, block count $\boldsymbol{c}_{\text{blk}} \in \mathbb{N}^N$, block index $\boldsymbol{i}_{\text{blk}} \in \mathbb{N}^{N \times k_s}$, column count $\boldsymbol{c}_{\text{col}} \in \mathbb{N}^N$, column index $\boldsymbol{i}_{\text{col}} \in \mathbb{N}^{N \times k_v}$
Scale $\tau \leftarrow \sqrt{\frac{1}{d_h}}$
Initialize $\boldsymbol{O} \leftarrow (0)^{S \times d_h} \in \mathbb{R}^{S \times d_h}$
# Parallelized in GPU
for $i \leftarrow 1$ to $N$ do
  Load $\boldsymbol{Q}_{\text{chip}} \leftarrow \boldsymbol{Q}^{i \times B:(i+1) \times B} \in \mathbb{R}^{B \times d_h}$
  Initialize $\boldsymbol{O}_{\text{chip}} \leftarrow (0)^{B \times d_h} \in \mathbb{R}^{B \times d_h}$
  Initialize $\boldsymbol{m} \leftarrow (-\inf)^B \in \mathbb{R}^B$
  Initialize $\boldsymbol{l} \leftarrow (0)^B \in \mathbb{R}^B$
  # Loop through block indexes: block sparse flash attention
  for $j \leftarrow 1$ to $\boldsymbol{c}_{\text{blk}}^i$ do
    Block start $s \leftarrow \boldsymbol{i}_{\text{blk}}^{i,j}$
    Load $\boldsymbol{K}_{\text{chip}} \leftarrow \boldsymbol{K}^{s:s+B} \in \mathbb{R}^{B \times d_h}$
    Load $\boldsymbol{V}_{\text{chip}} \leftarrow \boldsymbol{V}^{s:s+B} \in \mathbb{R}^{B \times d_h}$
    $\boldsymbol{S} \leftarrow \tau \boldsymbol{Q}_{\text{chip}} \boldsymbol{K}_{\text{chip}}^T$
    $\boldsymbol{S} \leftarrow \text{mask}(\boldsymbol{S})$
    $\boldsymbol{m}_{new}^i \leftarrow \max(\boldsymbol{m}^i, \text{rowmax}(\boldsymbol{S})) \in \mathbb{R}^B$
    $\boldsymbol{S} \leftarrow \boldsymbol{S} - \boldsymbol{m}_{new}^i$
    $\boldsymbol{P} \leftarrow \exp(\boldsymbol{S})$
    $\boldsymbol{l}_{new}^i \leftarrow \text{rowsum}(\boldsymbol{S}))$
    $\boldsymbol{\alpha} \leftarrow \exp(\boldsymbol{m}^i - \boldsymbol{m}_{new}^i)$
    $\boldsymbol{l}^i \leftarrow \boldsymbol{\alpha} \boldsymbol{l}^i + \boldsymbol{l}_{new}^i$
    $\boldsymbol{O}_{\text{chip}} \leftarrow \boldsymbol{\alpha} \boldsymbol{O}_{\text{chip}} + \boldsymbol{P} \boldsymbol{V}_{\text{chip}}$
  end for
  # Loop through column indexes : PIT sparse flash attention
  $j \leftarrow 0$
  while $j < \boldsymbol{c}_{\text{col}}^j$ do
    $\boldsymbol{cols} \leftarrow \boldsymbol{i}_{\text{col}}^{i,j:j+B} \in \mathbb{N}^B$
    Load $\boldsymbol{K}_{\text{chip}} \leftarrow \boldsymbol{K}^{\boldsymbol{cols}} \in \mathbb{R}^{B \times d_h}$
    Load $\boldsymbol{V}_{\text{chip}} \leftarrow \boldsymbol{V}^{\boldsymbol{cols}} \in \mathbb{R}^{B \times d_h}$
    $\boldsymbol{S} \leftarrow \tau \boldsymbol{Q}_{\text{chip}} \boldsymbol{K}_{\text{chip}}^T$
    $\boldsymbol{S} \leftarrow \text{mask}(\boldsymbol{S})$
    $\boldsymbol{m}_{new}^i \leftarrow \max(\boldsymbol{m}^i, \text{rowmax}(\boldsymbol{S})) \in \mathbb{R}^B$
    $\boldsymbol{S} \leftarrow \boldsymbol{S} - \boldsymbol{m}_{new}^i$
    $\boldsymbol{P} \leftarrow \exp(\boldsymbol{S})$
    $\boldsymbol{l}_{new}^i \leftarrow \text{rowsum}(\boldsymbol{S}))$
    $\boldsymbol{\alpha} \leftarrow \exp(\boldsymbol{m}^i - \boldsymbol{m}_{new}^i)$
    $\boldsymbol{l}^i \leftarrow \boldsymbol{\alpha} \boldsymbol{l}^i + \boldsymbol{l}_{new}^i$
    $\boldsymbol{O}_{\text{chip}} \leftarrow \boldsymbol{\alpha} \boldsymbol{O}_{\text{chip}} + \boldsymbol{P} \boldsymbol{V}_{\text{chip}}$
    $j \leftarrow j + B$
  end while
  # Write outputs
  $\boldsymbol{O}_{\text{chip}} \leftarrow \text{diag}(\boldsymbol{l}^i)^{-1} \boldsymbol{O}_{\text{chip}}$
  Save $\boldsymbol{O}_i \leftarrow \boldsymbol{O}_{\text{chip}}$
end for
```

Table 10: Comparison of generated results by different methods using LLaMA-3-8B-Instruct-262K in KV Retrieval task [ZCH+24].

Methods	Summary
Original	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: "5e6b7b90-710d-4953-9b18-3e96b2cadbf2" 2. Ground Truth: f1ec3dd8-b6e7-4af5-a589-fda0b7693f93 Prediction: "f1ec3dd8-b6e7-4af5-a589-fda0b7693f93" (a string of length 32).
StreamingLLM	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: 2b114db0-d87e-42d2-9b4c-0b1f115976ad.assistant 2. Ground Truth: bbf3fa73-6217-4e6e-923c-8349bd286c3d Prediction: "d829ce18-6339-4f77-8c04-31fc7ec33619".
StreamingLLM w/ dilated	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: 5 (2. Ground Truth: f1ec3dd8-b6e7-4af5-a589-fda0b7693f93 Prediction: "def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 def solverome2
StreamingLLM w/ strided	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: "def solverome2 def solverome2 def solverome2 def solverome2 def solverome2 2. Ground Truth: f1ec3dd8-b6e7-4af5-a589-fda0b7693f93 Prediction: "0 (
Ours w/ static	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: "def solverome2 def solverome2 def solverome2 def solverome2 def 2. Ground Truth: f1ec3dd8-b6e7-4af5-a589-fda0b7693f93 Prediction: "def solverome2 def solverome2 def solverome2 def solverome2 def
Ours	1. Ground Truth: 5e6b7b90-710d-4953-9b18-3e96b2cadbf2 Prediction: "5e6b7b90-710d-4953-9b18-3e96b2cadbf2" 2. Ground Truth: f1ec3dd8-b6e7-4af5-a589-fda0b7693f93 Prediction: "f1ec3dd8-b6e7-4af5-a589-fda0b7693f93" (a string of length 32).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We discuss the contributions and scope in detail and accurately in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of MInference in Appendix A.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The methods discussed and proposed in this paper are based on empirical observations and do not involve theoretical proof.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our experiments and implementation in §4 and Appendix C. Additionally, we have released our code at `https://aka.ms/MInference`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide detailed descriptions of our experiments and implementation in §4 and Appendix C. Additionally, we have released our code at `https://aka.ms/MInference`.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our experiments and implementation in §4 and Appendix C. Additionally, we have released our code at `https://aka.ms/MInference`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: We follow previous works by conducting tests on public benchmarks, without including error bars or similar information.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our experiments and implementation details including computation resources in §4 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the guidelines of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of MInference in Appendix B.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not involve the release of new models or data.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The LLMs, datasets, and codebase used in our work comply with open-source licenses and can be used for scientific research.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have released our code at `https://aka.ms/MInference`.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve data annotation or crowdsourcing.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.