MoBA: Mixture of Block Attention for Long-Context LLMs

Technical Report

Enzhe Lu[1] **Zhejun Jiang**[1] **Jingyuan Liu**[1] **Yulun Du**[1] **Tao Jiang**[1] **Chao Hong**[1]
Shaowei Liu[1] **Weiran He**[1] **Enming Yuan**[1] **Yuzhi Wang**[1] **Zhiqi Huang**[1] **Huan Yuan**[1]
Suting Xu[1] **Xinran Xu**[1] **Guokun Lai**[1] **Yanru Chen**[1] **Huabin Zheng**[1] **Junjie Yan**[1]
Jianlin Su[1] **Yuxin Wu**[1] **Neo Y. Zhang**[1] **Zhilin Yang**[1]
Xinyu Zhou[1,‡] **Mingxing Zhang**[2,*] **Jiezhong Qiu**[3,‡ *†]

[1] Moonshot AI [2] Tsinghua University [3] Zhejiang Lab/Zhejiang University

Abstract

Scaling the effective context length is essential for advancing large language models (LLMs) toward artificial general intelligence (AGI). However, the quadratic increase in computational complexity inherent in traditional attention mechanisms presents a prohibitive overhead. Existing approaches either impose strongly biased structures, such as sink or window attention which are task-specific, or radically modify the attention mechanism into linear approximations, whose performance in complex reasoning tasks remains inadequately explored.

In this work, we propose a solution that adheres to the "less structure" principle, allowing the model to determine where to attend autonomously, rather than introducing predefined biases. We introduce Mixture of Block Attention (MoBA), an innovative approach that applies the principles of Mixture of Experts (MoE) to the attention mechanism. This novel architecture demonstrates superior performance on long-context tasks while offering a key advantage: the ability to seamlessly transition between full and sparse attention, enhancing efficiency without the risk of compromising performance. MoBA has already been deployed to support Kimi's long-context requests and demonstrates significant advancements in efficient attention computation for LLMs. Our code is available at `https://github.com/MoonshotAI/moba`.

1 Introduction

The pursuit of artificial general intelligence (AGI) has driven the development of large language models (LLMs) to unprecedented scales, with the promise of handling complex tasks that mimic human cognition. A pivotal capability for achieving AGI is the ability to process, understand, and generate long sequences, which is essential for a wide range of applications, from historical data analysis to complex reasoning and decision-making processes. This growing demand for extended context processing can be seen not only in the popularity of long input prompt understanding, as showcased by models like Kimi (MoonshotAI 2023), Claude (Anthropic 2023) and Gemini (Reid et al. 2024), but also in recent explorations of long chain-of-thought (CoT) output capabilities in Kimi k1.5 (Team et al. 2025), DeepSeek-R1 (D. Guo et al. 2025), and OpenAI o1/o3 (Guan et al. 2024).

However, extending the sequence length in LLMs is non-trivial due to the quadratic growth in computational complexity associated with the vanilla attention mechanism (Waswani et al. 2017). This challenge has spurred a wave of research aimed at improving efficiency without sacrificing performance. One prominent direction capitalizes on the inherent sparsity of attention scores. This sparsity arises both mathematically — from the softmax operation, where various sparse attention patterns have be studied (H. Jiang et al. 2024) — and biologically (Watson et al. 2025), where sparse connectivity is observed in brain regions related to memory storage.

[**] `zhang_mingxing@mail.tsinghua.edu.cn`
[†‡] Co-corresponding authors. Xinyu Zhou (`zhouxinyu@moonshot.cn`), Jiezhong Qiu (`jiezhongqiu@outlook.com`)

Existing approaches often leverage predefined structural constraints, such as sink-based (G. Xiao et al. 2023) or sliding window attention (Beltagy et al. 2020), to exploit this sparsity. While these methods can be effective, they tend to be highly task-specific, potentially hindering the model's overall generalizability. Alternatively, a range of dynamic sparse attention mechanisms, exemplified by Quest (Tang et al. 2024), Minference (H. Jiang et al. 2024), and RetrievalAttention (Di Liu et al. 2024), select subsets of tokens at inference time. Although such methods can reduce computation for long sequences, they do not substantially alleviate the intensive training costs of long-context models, making it challenging to scale LLMs efficiently to contexts on the order of millions of tokens. Another promising alternative way has recently emerged in the form of linear attention models, such as Mamba (Dao and Gu 2024), RWKV (Peng, Alcaide, et al. 2023; Peng, Goldstein, et al. 2024), and RetNet (Sun et al. 2023). These approaches replace canonical softmax-based attention with linear approximations, thereby reducing the computational overhead for long-sequence processing. However, due to the substantial differences between linear and conventional attention, adapting existing Transformer models typically incurs high conversion costs (Mercat et al. 2024; J. Wang et al. 2024; Bick et al. 2025; M. Zhang et al. 2024) or requires training entirely new models from scratch (A. Li et al. 2025). More importantly, evidence of their effectiveness in complex reasoning tasks remains limited.

Consequently, a critical research question arises: How can we design a robust and adaptable attention architecture that retains the original Transformer framework while **adhering to a "less structure" principle, allowing the model to determine where to attend without relying on predefined biases?** Ideally, such an architecture would transition seamlessly between full and sparse attention modes, thus maximizing compatibility with existing pre-trained models and enabling both efficient inference and accelerated training without compromising performance.

Thus, we introduce Mixture of Block Attention (MoBA), a novel architecture that builds upon the innovative principles of Mixture of Experts (MoE) (Shazeer et al. 2017) and applies them to the attention mechanism of the Transformer model. MoE has been used primarily in the feedforward network (FFN) layers of Transformers (Lepikhin et al. 2020; Fedus et al. 2022; Zoph et al. 2022), but MoBA pioneers its application to long context attention, allowing dynamic selection of historically relevant blocks of key and values for each query token. This approach not only enhances the efficiency of LLMs but also enables them to handle longer and more complex prompts without a proportional increase in resource consumption. MoBA addresses the computational inefficiency of traditional attention mechanisms by partitioning the context into blocks and employing a gating mechanism to selectively route query tokens to the most relevant blocks. This block sparse attention significantly reduces the computational costs, paving the way for more efficient processing of long sequences. The model's ability to dynamically select the most informative blocks of keys leads to improved performance and efficiency, particularly beneficial for tasks involving extensive contextual information.

In this paper, we detail the architecture of MoBA, firstly its block partitioning and routing strategy, and secondly its computational efficiency compared to traditional attention mechanisms. We further present experimental results that demonstrate MoBA's superior performance in tasks requiring the processing of long sequences. Our work contributes a novel approach to efficient attention computation, pushing the boundaries of what is achievable with LLMs in handling complex and lengthy inputs.

2 Method

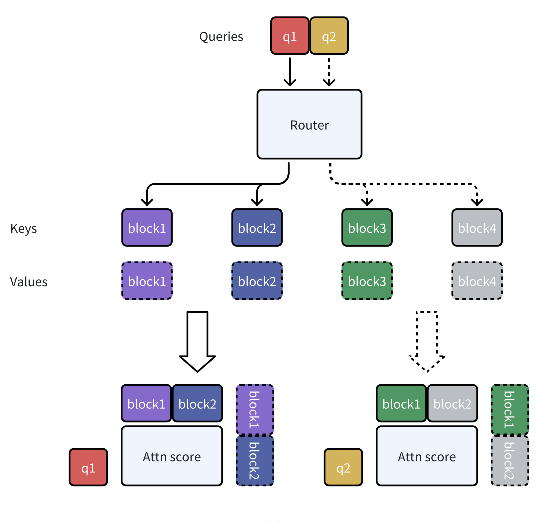
In this work, we introduce a novel architecture, termed Mixture of Block Attention (MoBA), which extends the capabilities of the Transformer model by dynamically selecting historical segments (blocks) for attention computation. MoBA is inspired by techniques of Mixture of Experts (MoE) and sparse attention. The former technique has been predominantly applied to the feedforward network (FFN) layers within the Transformer architecture, while the latter has been widely adopted in scaling Transformers to handle long contexts. Our method is innovative in applying the MoE principle to the attention mechanism itself, allowing for more efficient and effective processing of long sequences.

2.1 Preliminaries: Standard Attention in Transformer

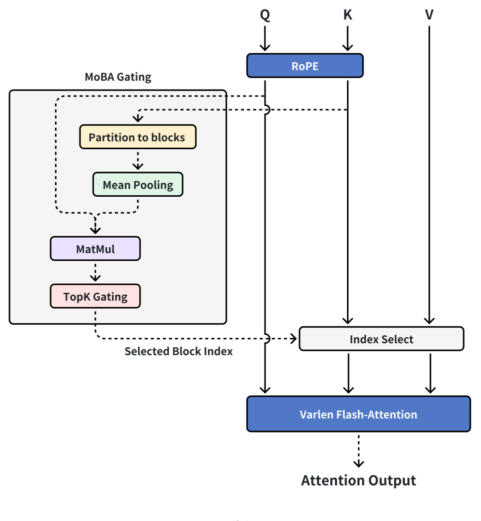
We first revisit the standard Attention in Transformers. For simplicity, we revisit the case where a single query token $\boldsymbol{q} \in \mathbb{R}^{1\times d}$ attends to the N key and value tokens, denoting $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N\times d}$, respectively. The standard attention is computed as:

$$\mathrm{Attn}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{Softmax}\big(\boldsymbol{q}\boldsymbol{K}^{\top}\big)\boldsymbol{V}, \tag{1}$$

where d denotes the dimension of a single attention head. We focus on the single-head scenario for clarity. The extension to multi-head attention involves concatenating the outputs from multiple such single-head attention operations.



(a)



(b)

Figure 1: Illustration of mixture of block attention (MoBA). **(a)** A running example of MoBA; **(b)** Integration of MoBA into Flash Attention.

2.2 MoBA Architecture

Different from standard attention where each query tokens attend to the entire context, MoBA enables each query token to only attend to a subset of keys and values:

$$\mathrm{MoBA}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{Softmax}\Big(\boldsymbol{q}\boldsymbol{K}[I]^\top\Big)\boldsymbol{V}[I], \tag{2}$$

where $I \subseteq [N]$ is the set of selected keys and values.

The key innovation in MoBA is the block partitioning and selection strategy. We divide the full context of length N into n blocks, where each block represents a subset of subsequent tokens. Without loss of generality, we assume that the context length N is divisible by the number of blocks n. We further denote $B = \frac{N}{n}$ to be the block size and

$$I_i = [(i-1) \times B + 1, i \times B] \tag{3}$$

to be the range of the i-th block. By applying the top-k gating mechanism from MoE, we enable each query to selectively focus on a subset of tokens from different blocks, rather than the entire context:

$$I = \bigcup_{g_i > 0} I_i. \tag{4}$$

The model employs a gating mechanism, as g_i in Equation 4, to select the most relevant blocks for each query token. The MoBA gate first computes the affinity score s_i measuring the relevance between query $\boldsymbol{q}$ and the i-th block, and applies a top-k gating among all blocks. More formally, the gate value for the i-th block g_i is computed by

$$g_i = \begin{cases} 1 & s_i \in \mathrm{Topk}\left(\{s_j | j \in [n]\}, k\right) \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $\mathrm{Topk}(\cdot, k)$ denotes the set containing k highest scores among the affinity scores calculated for each block. In this work, the score s_i is computed by the inner product between $\boldsymbol{q}$ and the mean pooling of $\boldsymbol{K}[I_i]$ along the sequence dimension:

$$s_i = \langle \boldsymbol{q}, \mathrm{mean\_pool}(\boldsymbol{K}[I_i]) \rangle \tag{6}$$

A Running Example. We provide a running example of MoBA at Figure 1a, where we have two query tokens and four KV blocks. The router (gating network) dynamically selects the top two blocks for each query to attend. As shown in Figure 1a, the first query is assigned to the first and second blocks, while the second query is assigned to the third and fourth blocks.

It is important to maintain causality in autoregressive language models, as they generate text by next-token prediction based on previous tokens. This sequential generation process ensures that a token cannot influence tokens that come before it, thus preserving the causal relationship. MoBA preserves causality through two specific designs:

Causality: No Attention to Future Blocks. MoBA ensures that a query token cannot be routed to any future blocks. By limiting the attention scope to current and past blocks, MoBA adheres to the autoregressive nature of language modeling. More formally, denoting $\mathrm{pos}(\boldsymbol{q})$ as the position index of the query $\boldsymbol{q}$, we set $s_i = -\infty$ and $g_i = 0$ for any blocks i such that $\mathrm{pos}(\boldsymbol{q}) < i \times B$.

Current Block Attention and Causal Masking. We define the ”current block” as the block that contains the query token itself. The routing to the current block could also violate causality, since mean pooling across the entire block can inadvertently include information from future tokens. To address this, we enforce that each token must be routed to its respective current block and apply a causal mask during the current block attention. This strategy not only avoids any leakage of information from subsequent tokens but also encourages attention to the local context. More formally, we set $g_i = 1$ for the block i where the position of the query token $\mathrm{pos}(\boldsymbol{q})$ is within the interval I_i. From the perspective of Mixture-of-Experts (MoE), the current block attention in MoBA is akin to the role of shared experts in modern MoE architectures (Dai et al. 2024; A. Yang et al. 2024), where static routing rules are added when expert selection.

Next, we discuss some additional key design choices of MoBA, such as its block segmentation strategy and the hybrid of MoBA and full attention.

Fine-Grained Block Segmentation. The positive impact of fine-grained expert segmentation in improving mode performance has been well-documented in the Mixture-of-Experts (MoE) literature (Dai et al. 2024; A. Yang et al. 2024). In this work, we explore the potential advantage of applying a similar fine-grained segmentation technique to MoBA. MoBA, inspired by MoE, operates segmentation along the context-length dimension rather than the FFN intermediate hidden dimension. Therefore our investigation aims to determine if MoBA can also benefit when we partition the context into blocks with a finer grain. More experimental results can be found in Section 3.1.

Hybrid of MoBA and Full Attention. MoBA is designed to be a substitute for full attention, maintaining the same number of parameters without any addition or subtraction. This feature inspires us to conduct smooth transitions between full attention and MoBA. Specifically, at the initialization stage, each attention layer has the option to select full attention or MoBA, and this choice can be dynamically altered during training if necessary. A similar idea of transitioning full attention to sliding window attention has been studied previous work (X. Zhang et al. 2024). More experimental results can be found in Section 3.2.

Comparing to Sliding Window Attention and Attention Sink. Sliding window attention (SWA) and attention sink are two popular sparse attention architectures. We demonstrate that both can be viewed as special cases of MoBA. For sliding window attention (Beltagy et al. 2020), each query token only attends to its neighboring tokens. This can be interpreted as a variant of MoBA with a gating network that keeps selecting the most recent blocks. Similarly, attention sink (G. Xiao et al. 2023), where each query token attends to a combination of initial tokens and the most recent tokens, can be seen as a variant of MoBA with a gating network that always selects both the initial and the recent blocks. The above discussion shows that MoBA has stronger expressive power than sliding window attention and attention sink. Moreover, it shows that MoBA can flexibly approximate many static sparse attention architectures by incorporating specific gating networks.

Overall, MoBA's attention mechanism allows the model to adaptively and dynamically focus on the most informative blocks of the context. This is particularly beneficial for tasks involving long documents or sequences, where attending to the entire context may be unnecessary and computationally expensive. MoBA's ability to selectively attend to relevant blocks enables more nuanced and efficient processing of information.

2.3 Implementation

We provide a high-performance implementation of MoBA, by incorporating optimization techniques from FlashAttention (Dao, D. Fu, et al. 2022) and MoE (Rajbhandari et al. 2022). Figure 2 demonstrates the high efficiency of MoBA, while we defer the detailed experiments on efficiency and scalability to Section 3.4. Our implementation consists of five major steps:

- Determine the assignment of query tokens to KV blocks according to the gating network and causal mask.
- Arrange the ordering of query tokens based on their assigned KV blocks.
- Compute attention outputs for each KV block and the query tokens assigned to it. This step can be optimized by FlashAttention with varying lengths.

Algorithm 1 MoBA (Mixture of Block Attention) Implementation

```
Require: Query, key and value matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times h \times d}$; MoBA hyperparameters (block size $B$ and top-$k$); $h$
    and $d$ denote the number of attention heads and head dimension. Also denote $n = N/B$ to be the number of
    blocks.
 1: // Split KV into blocks
 2: $\{\tilde{\mathbf{K}}_i, \tilde{\mathbf{V}}_i\}$ = split_blocks$(\mathbf{K}, \mathbf{V}, B)$, where $\tilde{\mathbf{K}}_i, \tilde{\mathbf{V}}_i \in \mathbb{R}^{B \times h \times d}, i \in [n]$
 3: // Compute gating scores for dynamic block selection
 4: $\bar{\mathbf{K}}$ = mean_pool$(\mathbf{K}, B) \in \mathbb{R}^{n \times h \times d}$
 5: $\mathbf{S} = \mathbf{Q}\bar{\mathbf{K}}^\top \in \mathbb{R}^{N \times h \times n}$
 6: // Select blocks with causal constraint (no attention to future blocks)
 7: $\mathbf{M}$ = create_causal_mask$(N, n)$
 8: $\mathbf{G}$ = topk$(\mathbf{S} + \mathbf{M}, k)$
 9: // Organize attention patterns for computation efficiency
10: $\mathbf{Q}^s, \tilde{\mathbf{K}}^s, \tilde{\mathbf{V}}^s$ = get_self_attn_block$(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}})$
11: $\mathbf{Q}^m, \tilde{\mathbf{K}}^m, \tilde{\mathbf{V}}^m$ = index_select_moba_attn_block$(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}}, \mathbf{G})$
12: // Compute attentions seperately
13: $\mathbf{O}^s$ = flash_attention_varlen$(\mathbf{Q}^s, \tilde{\mathbf{K}}^s, \tilde{\mathbf{V}}^s$, causal=True)
14: $\mathbf{O}^m$ = flash_attention_varlen$(\mathbf{Q}^m, \tilde{\mathbf{K}}^m, \tilde{\mathbf{V}}^m$, causal=False)
15: // Combine results with online softmax
16: $\mathbf{O}$ = combine_with_online_softmax$(\mathbf{O}^s, \mathbf{O}^m)$
17: return $\mathbf{O}$
```

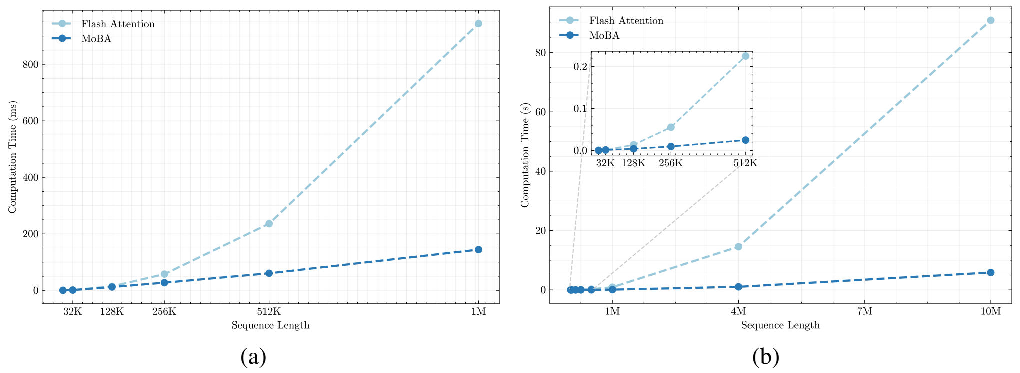


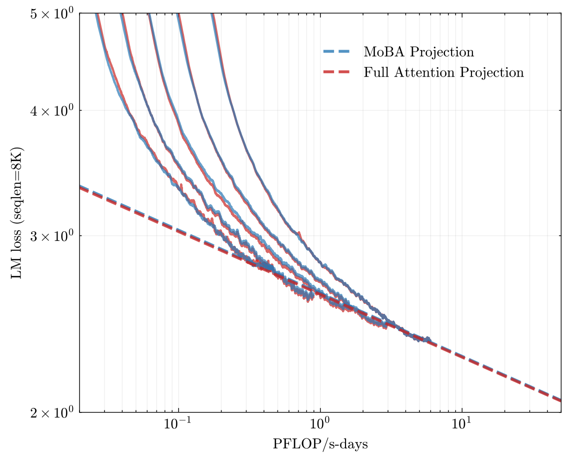
Figure 2: Efficiency of MoBA vs. full attention (implemented with Flash Attention). **(a)** 1M Model speedup evaluation: Computation time scaling of MoBA versus Flash Attention on 1M model with increasing sequence lengths (8K-1M). **(b)** Fixed Sparsity Ratio scaling: Computation time scaling comparison between MoBA and Flash Attention across increasing sequence lengths (8K-10M), maintaining a constant sparsity ratio of 95.31% (fixed 64 MoBA blocks with variance block size and fixed top-k=3).

- Re-arrange the attention outputs back to their original ordering.
- Combine the corresponding attention outputs using online Softmax (i.e., tiling), as a query token may attend to its current block and multiple historical KV blocks.

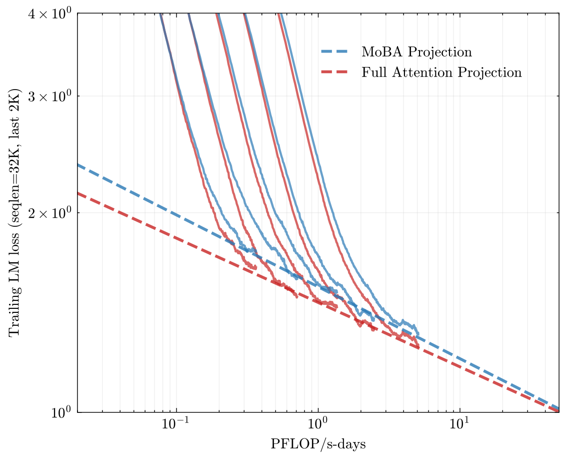
The algorithmic workflow is formalized in Algorithm 1 and visualized in Figure 1b, illustrating how MoBA can be implemented based on MoE and FlashAttention. First, the KV matrices are partitioned into blocks (Line 1-2). Next, the gating score is computed according to Equation 6, which measures the relevance between query tokens and KV blocks (Lines 3-7). A top-k operator is applied on the gating score (together with causal mask), resulting in a sparse query-to-KV-block mapping matrix $\boldsymbol{G}$ to represent the assignment of queries to KV blocks (Line 8). Then, query tokens are arranged based on the query-to-KV-block mapping, and block-wise attention outputs are computed (Line 9-12). Notably, attention to historical blocks (Line 11 and 14) and the current block attention (Line 10 and 13) are computed separately, as additional causality needs to be maintained in the current block attention. Finally, the attention outputs are rearranged back to their original ordering and combined with online softmax (Line 16) (Milakov et al. 2018; H. Liu et al. 2023).

Model Param	Head	Layer	Hidden	Training Token	Block size	TopK
568M	14	14	1792	10.8B	512	3
822M	16	16	2048	15.3B	512	3
1.1B	18	18	2304	20.6B	512	3
1.5B	20	20	2560	27.4B	512	3
2.1B	22	22	2816	36.9B	512	3

Table 1: Configuration of Scaling Law Experiments



(a)



(b)

L(C)	MoBA	Full
LM loss (seqlen=8K)	$2.625 \times C^{-0.063}$	$2.622 \times C^{-0.063}$
Trailing LM loss (seqlen=32K, last 2K)	$1.546 \times C^{-0.108}$	$1.464 \times C^{-0.097}$

(c)

Figure 3: Scaling law comparison between MoBA and full attention. **(a)** LM loss on validation set (seqlen=8K); **(b)** trailing LM loss on validation set (seqlen=32K, last 1K tokens); **(c)** fitted scaling law curve.

3 Experiments

3.1 Scaling Law Experiments and Ablation Studies

In this section, we conduct scaling law experiments and ablation studies to validate some key design choices of MoBA.

Scalability w.r.t. LM Loss. To assess the effectiveness of MoBA, we perform scaling law experiments by comparing the validation loss of language models trained using either full attention or MoBA. Following the Chinchilla scaling law (Hoffmann et al. 2022), we train five language models of varying sizes with a sufficient number of training tokens to ensure that each model achieves its training optimum. Detailed configurations of the scaling law experiments can be found in Table 1. Both MoBA and full attention models are trained with a sequence length of 8K. For MoBA models, we set the block size to 512 and select the top-3 blocks for attention, resulting in a sparse attention pattern with sparsity up to $1 - \frac{512 \times 3}{8192} = 81.25\%$[3]. In particular, MoBA serves as an alternative to full attention, meaning that it does not introduce new parameters or remove existing ones. This design simplifies our comparison process, as the only difference across all experiments lies in the attention modules, while all other hyperparameters, including the learning rate and batch size, remain constant. As shown in Figure 3a, the validation loss curves for MoBA and full attention display very similar scaling trends. Specifically, the validation loss differences between these two attention mechanisms remain consistent within a range of $1e-3$. This suggests that MoBA achieves scaling performance that is comparable to full attention, despite its sparse attention pattern with sparsity up to 75%.

[3]Since we set top-k=3, thus each query token can attend to at most 2 history block and the current block.

Long Context Scalability. However, LM loss may be skewed by the data length distribution (An et al. 2024), which is typically dominated by short sequences. To fully assess the long-context capability of MoBA, we assess the **LM loss of trailing tokens (trailing LM loss, in short)**, which computes the LM loss of the last few tokens in the sequence. We count this loss only for sequences that reach the maximum sequence length to avoid biases that may arise from very short sequences. A detailed discussion on trailing tokens scaling can be found in the Appendix A.1

These metrics provide insights into the model's ability to generate the final portion of a sequence, which can be particularly informative for tasks involving long context understanding. Therefore, we adopt a modified experimental setting by increasing the maximum sequence length from 8k to 32k. This adjustment leads to an even sparser attention pattern for MoBA, achieving a sparsity level of up to $1 - \frac{512\times3}{32768} = 95.31\%$. As shown in Figure 3b, although MoBA exhibits a marginally higher last block LM loss compared to full attention in all five experiments, the loss gap is progressively narrowing. This experiment implies the long-context scalability of MoBA.

Ablation Study on Fine-Grained Block Segmentation. We further ablate the block granularity of MoBA. We carry out a series of experiments using a 1.5B parameter model with a 32K context length. The hyperparameters of block size and top-k are adjusted to maintain a consistent level of attention sparsity. Specifically, we divide the 32K context into 8, 16, 32, 64, and 128 blocks, and correspondingly select 2, 4, 8, 16, and 32 blocks, ensuring an attention sparsity of 75% across these configurations. As shown in Figure 4, MoBA's performance is significantly affected by block granularity. Specifically, there is a performance difference of 1e-2 between the coarsest-grained setting (selecting 2 blocks from 8) and the settings with finer granularity. These findings suggest that fine-grained segmentation appears to be a general technique for enhancing the performance of models within the MoE family, including MoBA.

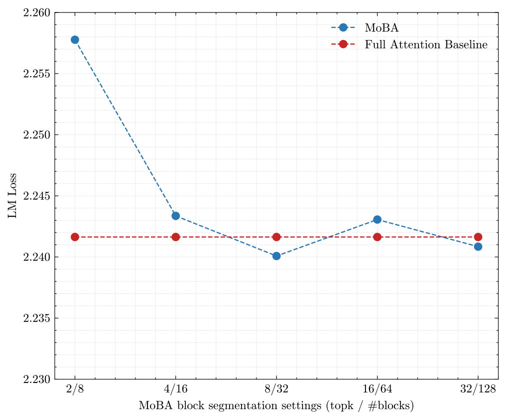


Figure 4: Fine-Grained Block Segmentation. The LM loss on validation set v.s. MoBA with different block granularity.

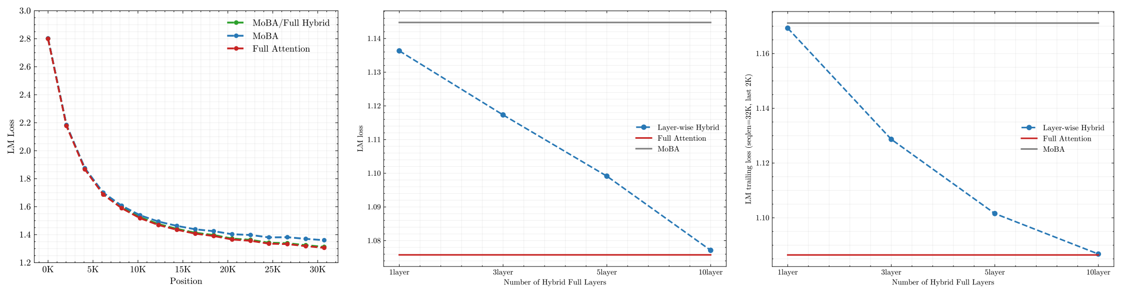
3.2 Hybrid of MoBA and Full Attention

As discussed in Section 2, we design MoBA to be a flexible substitute for full attention, so that it can easily switch from/to full attention with minimal overhead and achieve comparable long-context performance. In this section, we first show seamless transition between full attention and MoBA can be a solution for efficient long-context pre-training. Then we discuss the layer-wise hybrid strategy, mainly for the performance of supervised fine-tuning (SFT).

MoBA/Full Hybrid Training. We train three models, each with 1.5B parameters, on 30B tokens with a context length of 32K tokens. For the hyperparameters of MoBA, the block size is set to 2048, and the top-k parameter is set to 3. The detailed training recipes are as follows:

- MoBA/full hybrid: This model is trained using a two-stage recipe. In the first stage, MoBA is used to train on 90% of the tokens. In the second stage, the model switches to full attention for the remaining 10% of the tokens.
- Full attention: This model is trained using full attention throughout the entire training.
- MoBA: This model is trained exclusively using MoBA.

We evaluate their long-context performance via position-wise language model (LM) loss, which is a fine-grained metric to evaluate lm loss at each position within a sequence. Unlike the vanilla LM loss, which is computed by averaging the LM loss across all positions, the position-wise LM loss breaks down the loss for each position separately. Similar metrics have been suggested by previous studies (Xiong et al. 2023; Reid et al. 2024), who noticed that



(a) (b) (c)

Figure 5: Hybrid of MoBA and full attention. **(a)** position-wise LM loss for MoBA, full attention, and MoBA/full hybrid training; **(b)** SFT LM loss w.r.t the number of full attention layers in layer-wise hybrid; **(c)** SFT trailing LM loss (seqlen=32K, last 2K) w.r.t the number of full attention layers in layer-wise hybrid.

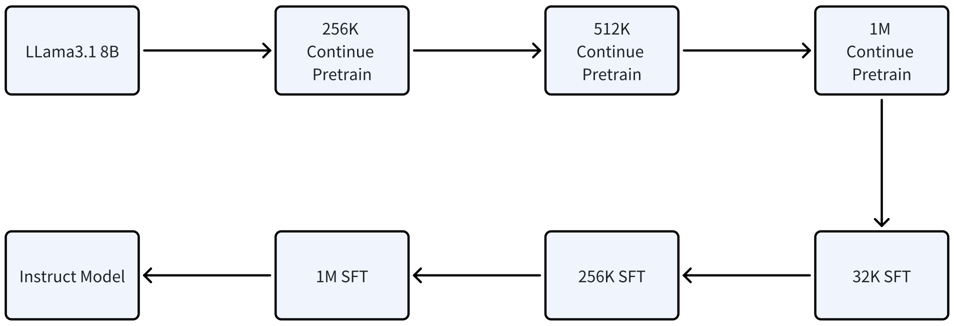


Figure 6: The continual pre-training and SFT recipes.

position-wise LM loss follows a power-law trend relative to context length. As shown in Figure 5a, the MoBA-only recipe results in higher position-wise losses for trailing tokens. Importantly, our MoBA/full hybrid recipe reaches a loss nearly identical to that of full attention. This result highlights the effectiveness of the MoBA/full hybrid training recipe in balancing training efficiency with model performance. More interestingly, we have not observed significant loss spikes during the switch between MoBA and full attention, again demonstrating the flexibility and robustness of MoBA.

Layer-wise Hybrid. This flexibility of MoBA encourages us to delve into a more sophisticated strategy — the layer-wise hybrid of MoBA and full attention. We investigate this strategy with a particular focus on its application during the supervised fine-tuning (SFT). The motivation for investigating this strategy stems from our observation that MoBA sometimes results in suboptimal performance during SFT, as shown in Figure 5b. We speculate that this may be attributed to the loss masking employed in SFT — prompt tokens are typically excluded from the loss calculation during SFT, which can pose a sparse gradient challenge for sparse attention methods like MoBA. Because it may hinder the backpropagation of gradients, which are initially calculated from unmasked tokens, throughout the entire context. To address this issue, we propose a hybrid approach — switching the last several Transformer layers from MoBA to full attention, while the remaining layers continue to employ MoBA. As shown in Figure 5b and Figure 5c, this strategy can significantly reduce SFT loss.

3.3 Large Language Modeling Evaluation

We conduct a thorough assessment of MoBA across a variety of real-world downstream tasks, evaluating its performance in comparison to full attention models. For ease of verification, our experiments begin with the Llama 3.1 8B Base Model, which is used as the starting point for long-context pre-training. This model, termed Llama-8B-1M-MoBA, is initially trained with a context length of 128K tokens, and we gradually increase the context length to 256K, 512K, and 1M tokens during the continual pre-training. To ease this transition, we use position interpolation method (S. Chen et al. 2023) at the start of the 256K continual pre-training stage. This technique enables us to extend

Benchmark	**Llama-8B-1M-MoBA**	**Llama-8B-1M-Full**
AGIEval [0-shot]	0.5144	**0.5146**
BBH [3-shot]	0.6573	**0.6589**
CEval [5-shot]	**0.6273**	0.6165
GSM8K [5-shot]	**0.7278**	0.7142
HellaSWAG [0-shot]	0.8262	**0.8279**
Loogle [0-shot]	**0.4209**	0.4016
Competition Math [0-shot]	0.4254	**0.4324**
MBPP [3-shot]	**0.5380**	0.5320
MBPP Sanitized [0-shot]	**0.6926**	0.6615
MMLU [0-shot]	0.4903	**0.4904**
MMLU Pro [5-shot][CoT]	0.4295	**0.4328**
OpenAI HumanEval [0-shot][pass@1]	0.6951	**0.7012**
SimpleQA [0-shot]	0.0465	**0.0492**
TriviaQA [0-shot]	**0.5673**	0.5667
LongBench @32K [0-shot]	**0.4828**	0.4821
RULER @128K [0-shot]	0.7818	**0.7849**

Table 2: Performance comparison between MoBA and full Attention across different evaluation benchmarks.

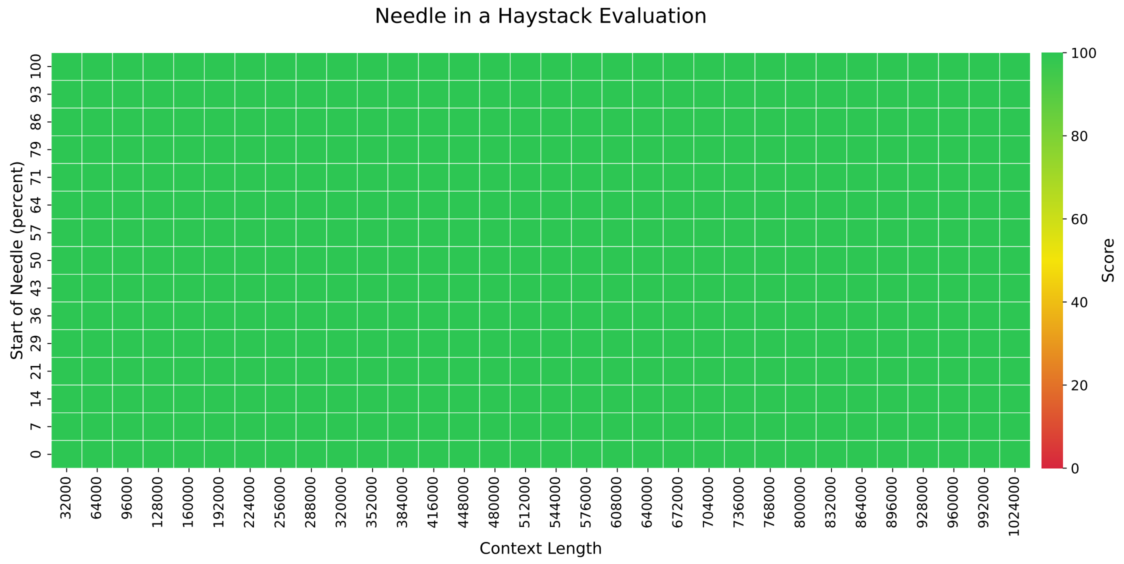


Figure 7: Performance of LLama-8B-1M-MoBA on the Needle in the Haystack benchmark (upto 1M context length).

the effective context length from 128K tokens to 1M tokens. After completing the 1M continuous pre-training, MoBA is activated for 100B tokens. We set the block size to 4096 and the top-K parameter to 12, leading to an attention sparsity of up to $1 - \frac{4096 \times 12}{1M} = 95.31\%$. To preserve some full attention capabilities, we adopt the layer-wise hybrid strategy — the last three layers remain as full attention, while the other 29 full attention layers are switched to MoBA. For supervised fine-tuning, we follow a similar strategy that gradually increases the context length from 32K to 1M. The baseline full attention models (termed Llama-8B-1M-Full) also follow a similar training strategy as shown in Figure 6, with the only difference being the use of full attention throughout the process. This approach allows us to directly compare the performance of MoBA with that of full attention models under equivalent training conditions.

The evaluation is performed on several widely used long-context benchmarks. In particular, across all evaluation tasks, MoBA is used for prefill only, while we switch to full attention during generation for better performance. As shown in Table 2, Llama-8B-1M-MoBA exhibits a performance that is highly comparable to that of Llama-8B-1M-Full. It is particularly noteworthy that in the longest benchmark, RULER, where MoBA operates at a sparsity level of up to $1 - \frac{4096 \times 12}{128K} = 62.5\%$, Llama-8B-1M-MoBA nearly matches the performance of Llama-8B-1M-Full, with a score of 0.7818 compared to 0.7849. For context lengths of up to 1M tokens, we evaluate the model using the traditional Needle in the Haystack benchmark. As shown in Figure 7, Llama-8B-1M-MoBA demonstrates satisfactory performance even with an extended context length of 1 million tokens.

3.4 Efficiency and Scalability

The above experimental results show that MoBA achieves comparable performance not only regarding language model losses but also in real-world tasks. To further investigate its efficiency, we compare the forward pass time of the attention layer in two models trained in Section 3.3 — Llama-8B-1M-MoBA and Llama-8B-1M-Full. We focus solely on the attention layer, as all other layers (e.g., FFN) have identical FLOPs in both models. As shown in Figure 2a, MoBA is more efficient than full attention across all context lengths, demonstrating a sub-quadratic computational complexity. In particular, it achieves a speedup ratio of up to 6.5x when prefilling 1M tokens.

We also explore the length scalability of MoBA by gradually increasing the context length to 10 million tokens. To maintain a constant attention sparsity, we keep the top-k value and number of MoBA Block fixed while proportionally increasing the block size. To reach the 10M context length, we expanded tensor parallelism (Shoeybi et al. 2019) toward the query head level, Specifically, we broadcast key and value tensors across distributed query heads, effectively addressing GPU memory limitations while preserving computational efficiency. As shown in Figure 2b, MoBA demonstrates superior efficiency compared to standard Flash Attention when scaling to longer sequences. Specifically, at 10M tokens moba achieves a speedup ratio of 16x reduction in attention computation time. The inset graph in the top figure, focusing on shorter sequences (32K to 512K), shows that even though both methods perform comparably at smaller scales, MoBA's computational advantage becomes increasingly evident as sequences grow longer, highlighting its particular strength in processing extremely long sequences.

Overall, the high efficiency of MoBA can be attributed to two key innovations: (1) the block sparse attention mechanism, and (2) the optimized implementation combining Mixture-of-Experts (MoE) and FlashAttention, as described in Section 2.3. These techniques effectively address the quadratic complexity limitation of full attention, reducing the computational complexity to a more economical sub-quadratic scale.

4 Related Work

The development of efficient attention (Tay, Dehghani, et al. 2020) mechanisms has been a critical area of research in the field of natural language processing, particularly with the rise of Large Language Models (LLMs). As the demand for handling longer sequences and reducing computational costs grows, efficeint attention techniques have emerged as a promising solution to reduce the quadratic complexity of self-attention mechanisms while maintaining model performance.

Static Sparse Patterns: Significant efforts, such as Sparse Transformer (Child et al. 2019), Star-Transformer (Q. Guo et al. 2019), BlockBERT (Qiu et al. 2019), Longformer (Beltagy et al. 2020), GMAT (Gupta et al. 2020), ETC (Ainslie, Ontanon, et al. 2020), BigBird (Zaheer et al. 2020), LongT5 (M. Guo et al. 2021) and LongNet (J. Ding et al. 2023), have been dedicated to the design of static attention patterns in LLMs. Their choices of static attention patterns can encompass strided and fixed attention, window attention, global token attention, random attention, dilated attention, block sparse attention, or any combinations of them. In the realm of multimodal models, static sparse attention mechanisms have also been developed, such as axial attention (Ho et al. 2019) for 2D images and spatial-temporal attention (Z. Zheng et al. 2024) for 3D videos.

Dynamic Sparse Patterns: Different from static patterns, dynamic sparse attention techniques adaptively determine which tokens to attend. Reformer (Kitaev et al. 2020) and Routing Transformer (Roy et al. 2021) respectively employ locality-sensitive hashing (LSH) and K-means to cluster tokens, and attend to clusters rather than the full context. Memorizing Transformers (Yuhuai Wu et al. 2022) and Unlimiformer (Bertsch et al. 2024) dynamically attend to tokens selected by the k-nearest-neighbor (kNN) algorithms. CoLT5 (Ainslie, Lei, et al. 2023) designs a routing modules to select the most important queries and keys. Sparse Sinkhorn Attention (Tay, Bahri, et al. 2020) learns to permute blocks from the input sequence, allowing dynamic block sparse attention computation.

Training-free Sparse Attention: In addition to the previously discussed approaches that study training sparse attention models, there are also strategies designed to incorporate sparse attention mechanisms to enhance the efficiency of the two primary stages of model inference — either the prefill stage or the decode stage, or both of them. During the prefill optimization phase, the complete prompt can be utilized for attention profiling, which allows for the exploration of more intricate sparse attention patterns. For instance, MoA (T. Fu et al. 2024), Minference (H. Jiang et al. 2024), and SeerAttention (Y. Gao et al. 2024) have investigated sparse attention configurations such as A-shape, vertical-slash, and dynamic block sparsity. In the context of decode optimization, considerable work has been dedicated to compressing and pruning the KV-cache to achieve a balance between the quality and speed of text generation. Notable efforts in this area include H2O (Z. Zhang et al. 2024), StreamingLLM (G. Xiao et al. 2023), TOVA (Oren et al. 2024), FastGen (Ge et al. 2023) and Quest (Tang et al. 2024). Quest, in particular, can be viewed as MoBA with a smaller block size and a specialized block representation function which combines both min and max pooling. Another work closely related to MoBA is Longheads (Y. Lu et al. 2024) which can be viewed as MoBA with a top-1 gating network, meaning that each query selects the most relevant KV blocks for attention.

Beyond Traditional Attention Architecture: Another line of research investigates novel model architectures that deviate from the conventional attention mechanism. As architectures change, these methods require training models from scratch and are unable to reuse pre-trained Transformer-based models. Studies in this domain have explored architectures that are inspired by Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), State Space Models (SSMs), or Linear Attention (Katharopoulos et al. 2020), Examples of such models include Hyena (Poli et al. 2023), Performer (Choromanski et al. 2020), Linformer (S. Wang et al. 2020), RWKV Peng, Alcaide, et al. 2023, Mamba (Gu et al. 2023), RetNet (Sun et al. 2023), etc.

In summary, the landscape of efficient attention techniques is diverse, encompassing sparse patterns that range from static to dynamic, optimization objectives that span from training to inference, and architectures that extend from traditional attention mechanisms to innovative alternatives. Each method presents unique advantages and trade-offs, and the choice of technique often depends on the specific requirements of the application, such as the maximum sequence length, computational resources, and the desired balance between efficiency and performance. As research in this area continues to evolve, it is expected that these methods will play a crucial role in enabling LLMs to tackle increasingly complex tasks while maintaining efficiency and scalability.

5 Conclusion

In this paper, we introduce Mixture of Block Attention (MoBA), a novel attention architecture inspired by the principles of Mixture of Experts (MoE) that aims to enhance the efficiency and scalability of large language models (LLMs) for long-context tasks. MoBA addresses the computational challenges associated with traditional attention mechanisms by partitioning the context into blocks and employing a dynamic gating mechanism to selectively route query tokens to the most relevant KV blocks. This approach not only reduces computational complexity but also maintains model performance. Moreover, it allows for seamless transitions between full and sparse attention. Through extensive experiments, we demonstrated that MoBA achieves performance comparable to full attention while significantly improving computational efficiency. Our results show that MoBA can scale effectively to long contexts, maintaining low LM losses and high performance on various benchmarks. Additionally, MoBA's flexibility allows it to be integrated with existing models without substantial training cost, making it a practical continual pre-training solution for enhancing long-context capabilities in LLMs. In summary, MoBA represents a significant advancement in efficient attention, offering a balanced approach between performance and efficiency. Future work may explore further optimizations of MoBA's block-selection strategies, investigate its application to other modalities, and study its potential for improving generalization in complex reasoning tasks.

References

Ainslie, Joshua, Tao Lei, et al. "Colt5: Faster long-range transformers with conditional computation". In: *arXiv preprint arXiv:2303.09752* (2023).

Ainslie, Joshua, Santiago Ontanon, et al. "ETC: Encoding long and structured inputs in transformers". In: *arXiv preprint arXiv:2004.08483* (2020).

An, Chenxin et al. "Why Does the Effective Context Length of LLMs Fall Short?" In: *arXiv preprint arXiv:2410.18745* (2024).

Anthropic. *Introducing 100K Context Windows*. `https://www.anthropic.com/news/100k-context-windows`. 2023.

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

Bertsch, Amanda et al. "Unlimiformer: Long-range transformers with unlimited length input". In: *Advances in Neural Information Processing Systems* 36 (2024).

Bick, Aviv et al. "Transformers to ssms: Distilling quadratic knowledge to subquadratic models". In: *Advances in Neural Information Processing Systems* 37 (2025), pp. 31788–31812.

Chen, Shouyuan et al. "Extending Context Window of Large Language Models via Positional Interpolation". In: *arXiv preprint arXiv:2401.06066* (2023).

Child, Rewon et al. "Generating long sequences with sparse transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

Choromanski, Krzysztof et al. "Rethinking attention with performers". In: *arXiv preprint arXiv:2009.14794* (2020).

Dai, Damai et al. "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models". In: *arXiv preprint arXiv:2401.06066* (2024).

Dao, Tri, Dan Fu, et al. “Flashattention: Fast and memory-efficient exact attention with io-awareness”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16344–16359.

Dao, Tri and Albert Gu. “Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality”. In: *arXiv preprint arXiv:2405.21060* (2024).

Ding, Jiayu et al. “Longnet: Scaling transformers to 1,000,000,000 tokens”. In: *arXiv preprint arXiv:2307.02486* (2023).

Fedus, William, Barret Zoph, and Noam Shazeer. “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity”. In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39.

Fu, Tianyu et al. “Moa: Mixture of sparse attention for automatic large language model compression”. In: *arXiv preprint arXiv:2406.14909* (2024).

Gao, Yizhao et al. “SeerAttention: Learning Intrinsic Sparse Attention in Your LLMs”. In: *arXiv preprint arXiv:2410.13276* (2024).

Ge, Suyu et al. “Model tells you what to discard: Adaptive kv cache compression for llms”. In: *arXiv preprint arXiv:2310.01801* (2023).

Gu, Albert and Tri Dao. “Mamba: Linear-time sequence modeling with selective state spaces”. In: *arXiv preprint arXiv:2312.00752* (2023).

Guan, Melody Y et al. “Deliberative alignment: Reasoning enables safer language models”. In: *arXiv preprint arXiv:2412.16339* (2024).

Guo, Daya et al. “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning”. In: *arXiv preprint arXiv:2501.12948* (2025).

Guo, Mandy et al. “LongT5: Efficient text-to-text transformer for long sequences”. In: *arXiv preprint arXiv:2112.07916* (2021).

Guo, Qipeng et al. “Star-transformer”. In: *arXiv preprint arXiv:1902.09113* (2019).

Gupta, Ankit and Jonathan Berant. “Gmat: Global memory augmentation for transformers”. In: *arXiv preprint arXiv:2006.03274* (2020).

Ho, Jonathan et al. “Axial attention in multidimensional transformers”. In: *arXiv preprint arXiv:1912.12180* (2019).

Hoffmann, Jordan et al. “Training compute-optimal large language models”. In: *arXiv preprint arXiv:2203.15556* (2022).

Jiang, Huiqiang et al. “Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention”. In: *arXiv preprint arXiv:2407.02490* (2024).

Katharopoulos, Angelos et al. “Transformers are rnns: Fast autoregressive transformers with linear attention”. In: *International conference on machine learning*. PMLR. 2020, pp. 5156–5165.

Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. “Reformer: The efficient transformer”. In: *arXiv preprint arXiv:2001.04451* (2020).

Lepikhin, Dmitry et al. “Gshard: Scaling giant models with conditional computation and automatic sharding”. In: *arXiv preprint arXiv:2006.16668* (2020).

Li, Aonian et al. “Minimax-01: Scaling foundation models with lightning attention”. In: *arXiv preprint arXiv:2501.08313* (2025).

Liu, Di et al. “Retrievalattention: Accelerating long-context llm inference via vector retrieval”. In: *arXiv preprint arXiv:2409.10516* (2024).

Liu, Hao and Pieter Abbeel. “Blockwise Parallel Transformer for Large Context Models”. In: *arXiv preprint arXiv:2305.19370* (2023).

Lu, Yi et al. “LongHeads: Multi-Head Attention is Secretly a Long Context Processor”. In: *arXiv preprint arXiv:2402.10685* (2024).

Mercat, Jean et al. “Linearizing Large Language Models”. In: *arXiv preprint arXiv:2405.06640* (2024).

Milakov, Maxim and Natalia Gimelshein. “Online normalizer calculation for softmax”. In: *arXiv preprint arXiv:1805.02867* (2018).

MoonshotAI. *Kimi Chat*. `https://kimi.moonshot.cn/`. 2023.

Oren, Matanel et al. “Transformers are multi-state rnns”. In: *arXiv preprint arXiv:2401.06104* (2024).

Peng, Bo, Eric Alcaide, et al. “Rwkv: Reinventing rnns for the transformer era”. In: *arXiv preprint arXiv:2305.13048* (2023).

Peng, Bo, Daniel Goldstein, et al. “Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence”. In: *arXiv preprint arXiv:2404.05892* (2024).

Poli, Michael et al. “Hyena hierarchy: Towards larger convolutional language models”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 28043–28078.

Qiu, Jiezhong et al. “Blockwise self-attention for long document understanding”. In: *arXiv preprint arXiv:1911.02972* (2019).

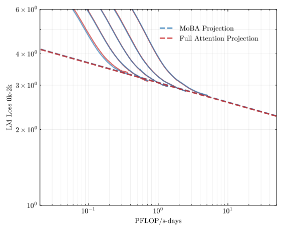
Rajbhandari, Samyam et al. “Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale”. In: *International conference on machine learning*. PMLR. 2022, pp. 18332–18346.

Reid, Machel et al. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. In: *arXiv preprint arXiv:2403.05530* (2024).
Roy, Aurko et al. “Efficient content-based sparse attention with routing transformers”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 53–68.
Shazeer, Noam et al. “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer”. In: *arXiv preprint arXiv:1701.06538* (2017).
Shoeybi, Mohammad et al. “Megatron-lm: Training multi-billion parameter language models using model parallelism”. In: *arXiv preprint arXiv:1909.08053* (2019).
Sun, Yutao et al. “Retentive network: A successor to transformer for large language models”. In: *arXiv preprint arXiv:2307.08621* (2023).
Tang, Jiaming et al. “Quest: Query-Aware Sparsity for Efficient Long-Context LLM Inference”. In: *arXiv preprint arXiv:2406.10774* (2024).
Tay, Yi, Dara Bahri, et al. “Sparse sinkhorn attention”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9438–9447.
Tay, Yi, Mostafa Dehghani, et al. “Efficient transformers: A survey. arXiv”. In: *arXiv preprint arXiv:2009.06732* (2020).
Team, Kimi et al. “Kimi k1. 5: Scaling Reinforcement Learning with LLMs”. In: *arXiv preprint arXiv:2501.12599* (2025).
Wang, Junxiong et al. “The mamba in the llama: Distilling and accelerating hybrid models”. In: *arXiv preprint arXiv:2408.15237* (2024).
Wang, Sinong et al. “Linformer: Self-attention with linear complexity”. In: *arXiv preprint arXiv:2006.04768* (2020).
Waswani, A et al. “Attention is all you need”. In: *NIPS*. 2017.
Watson, Jake F et al. “Human hippocampal CA3 uses specific functional connectivity rules for efficient associative memory”. In: *Cell* 188.2 (2025), pp. 501–514.
Wu, Yuhuai et al. “Memorizing transformers”. In: *arXiv preprint arXiv:2203.08913* (2022).
Xiao, Guangxuan et al. “Efficient streaming language models with attention sinks”. In: *arXiv preprint arXiv:2309.17453* (2023).
Xiong, Wenhan et al. “Effective Long-Context Scaling of Foundation Models”. In: *arXiv preprint arXiv:2309.16039* (2023).
Yang, An et al. “Qwen2. 5 Technical Report”. In: *arXiv preprint arXiv:2412.15115* (2024).
Zaheer, Manzil et al. “Big bird: Transformers for longer sequences”. In: *Advances in neural information processing systems* 33 (2020), pp. 17283–17297.
Zhang, Michael et al. “LoLCATs: On Low-Rank Linearizing of Large Language Models”. In: *arXiv preprint arXiv:2410.10254* (2024).
Zhang, Xuan et al. “Simlayerkv: A simple framework for layer-level KV cache reduction”. In: *arXiv preprint arXiv:2410.13846* (2024).
Zhang, Zhenyu et al. “H2o: Heavy-hitter oracle for efficient generative inference of large language models”. In: *Advances in Neural Information Processing Systems* 36 (2024).
Zheng, Zangwei et al. *Open-Sora: Democratizing Efficient Video Production for All*. Mar. 2024. URL: `https://github.com/hpcaitech/Open-Sora`.
Zoph, Barret et al. “St-moe: Designing stable and transferable sparse expert models”. In: *arXiv preprint arXiv:2202.08906* (2022).

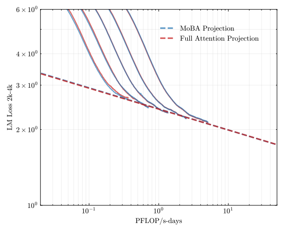
A Appendix

A.1 Long Context Scalability

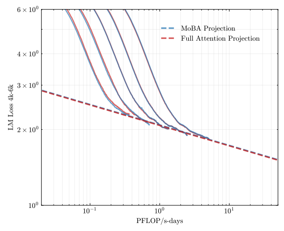
To address the bias in natural data distribution that favors short contexts, we strategically segmented the overall sequences into discrete segments based on their actual positions. For example, the segment spanning positions 30K-32K exclusively reflects losses associated with documents exceeding 30K context lengths and also masks the positions from 30K to 32K. This approach ensures a more balanced and representative evaluation across different context lengths. In our exploration of long-context scalability, we made a pivotal discovery: the trailing tokens account for the majority of the performance discrepancy between the full context baseline and the newly proposed sparse attention architectures. Consequently, we streamlined the long-context scaling process by focusing on trailing token scaling. This not only simplifies the computational requirements but also significantly enhances the efficiency and effectiveness of investigating long-context scenarios. This finding holds substantial implications for the development of more efficient and scalable attention mechanisms in the future.



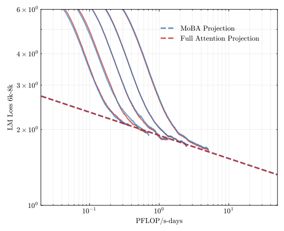
(a) Scaling law (0-2k)



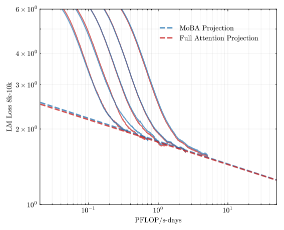
(b) Scaling law (2-4k)



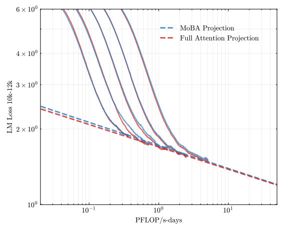
(c) Scaling law (4-6k)



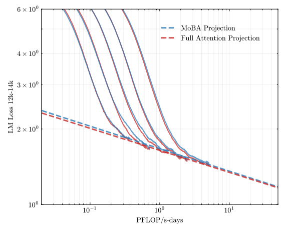
(d) Scaling law (6-8k)



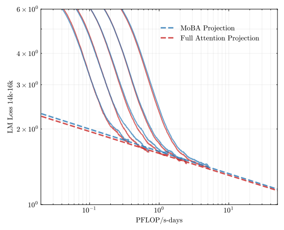
(e) Scaling law (8-10k)



(f) Scaling law (10-12k)

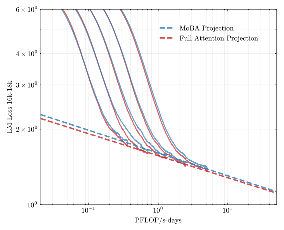


(g) Scaling law (12-14k)

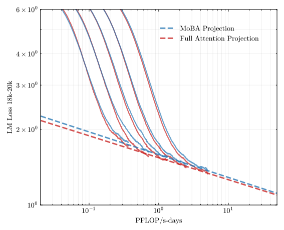


(h) Scaling law (14-16k)

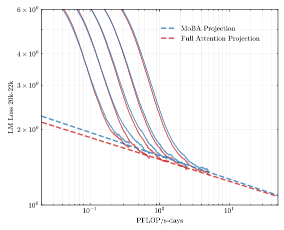
Figure 8: Scaling laws for positions 0-16k



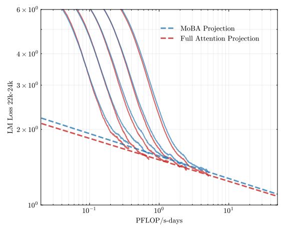
(i) Scaling law (16-18k)



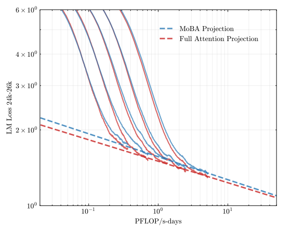
(j) Scaling law (18-20k)



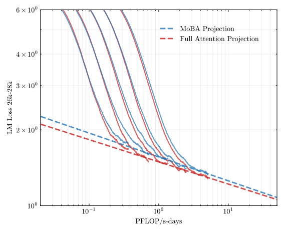
(k) Scaling law (20-22k)



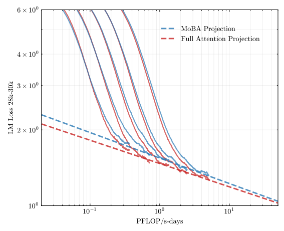
(l) Scaling law (22-24k)



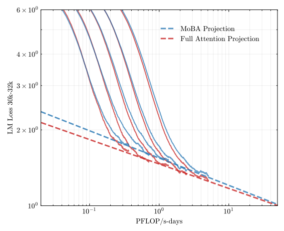
(m) Scaling law (24-26k)



(n) Scaling law (26-28k)



(o) Scaling law (28-30k)



(p) Scaling law (30-32k)

Figure 8: Scaling laws for positions 16-32k

Table 3: Loss scaling with different positions

LM Loss Position Range	MoBA	Full
0K - 2K	$3.075 \times C^{-0.078}$	$3.068 \times C^{-0.078}$
2K - 4K	$2.415 \times C^{-0.084}$	$2.411 \times C^{-0.083}$
4K - 6K	$2.085 \times C^{-0.081}$	$2.077 \times C^{-0.081}$
6K - 8K	$1.899 \times C^{-0.092}$	$1.894 \times C^{-0.092}$
8K - 10K	$1.789 \times C^{-0.091}$	$1.774 \times C^{-0.089}$
10K - 12K	$1.721 \times C^{-0.092}$	$1.697 \times C^{-0.087}$
12K - 14K	$1.670 \times C^{-0.089}$	$1.645 \times C^{-0.088}$
14K - 16K	$1.630 \times C^{-0.089}$	$1.600 \times C^{-0.087}$
16K - 18K	$1.607 \times C^{-0.090}$	$1.567 \times C^{-0.087}$
18K - 20K	$1.586 \times C^{-0.091}$	$1.542 \times C^{-0.087}$
20K - 22K	$1.571 \times C^{-0.093}$	$1.519 \times C^{-0.086}$
22K - 24K	$1.566 \times C^{-0.089}$	$1.513 \times C^{-0.085}$
24K - 26K	$1.565 \times C^{-0.091}$	$1.502 \times C^{-0.085}$
26K - 28K	$1.562 \times C^{-0.095}$	$1.493 \times C^{-0.088}$
28K - 30K	$1.547 \times C^{-0.097}$	$1.471 \times C^{-0.091}$
30K - 32K	$1.546 \times C^{-0.108}$	$1.464 \times C^{-0.097}$