MoBA: Mixture of Block Attention for Long-Context LLMs

Jiezhong Qiu

Hangzhou Institute of Medicine Chinese Academy of Sciences

MoBA

- Paper: https://arxiv.org/pdf/2502.13189
- Code: https://github.com/MoonshotAI/MoBA

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,‡ *†]

MoBA

- Full Attention

$$\mathrm{Attn}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{Softmax}\left(\boldsymbol{q}\boldsymbol{K}^\top\right)\boldsymbol{V}$$

- MoBA:

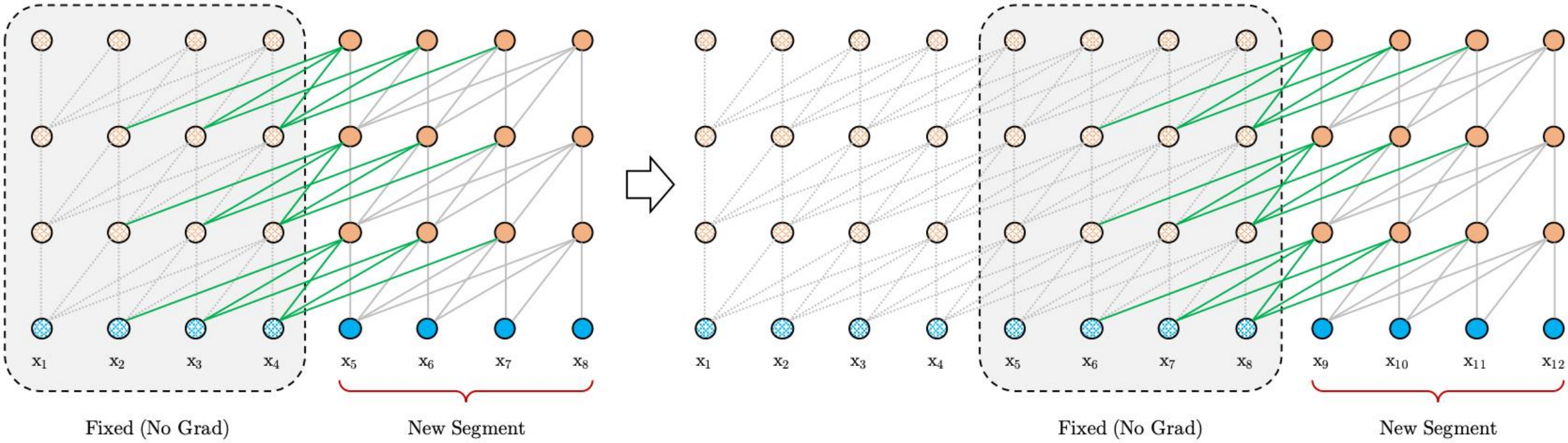
$$\mathrm{MoBA}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{Softmax}\left(\boldsymbol{q}\boldsymbol{K}[I]^\top\right)\boldsymbol{V}[I]$$

$$I_i = [(i-1) \times B + 1, i \times B] \qquad I = \bigcup_{g_i > 0} I_i.$$

$$g_i = \begin{cases} 1 & s_i \in \mathrm{Topk}\left(\{s_j | j \in [n]\}, k\right) \\ 0 & \text{otherwise} \end{cases} \qquad s_i = \langle \boldsymbol{q}, \mathrm{mean\_pool}(\boldsymbol{K}[I_i]) \rangle$$

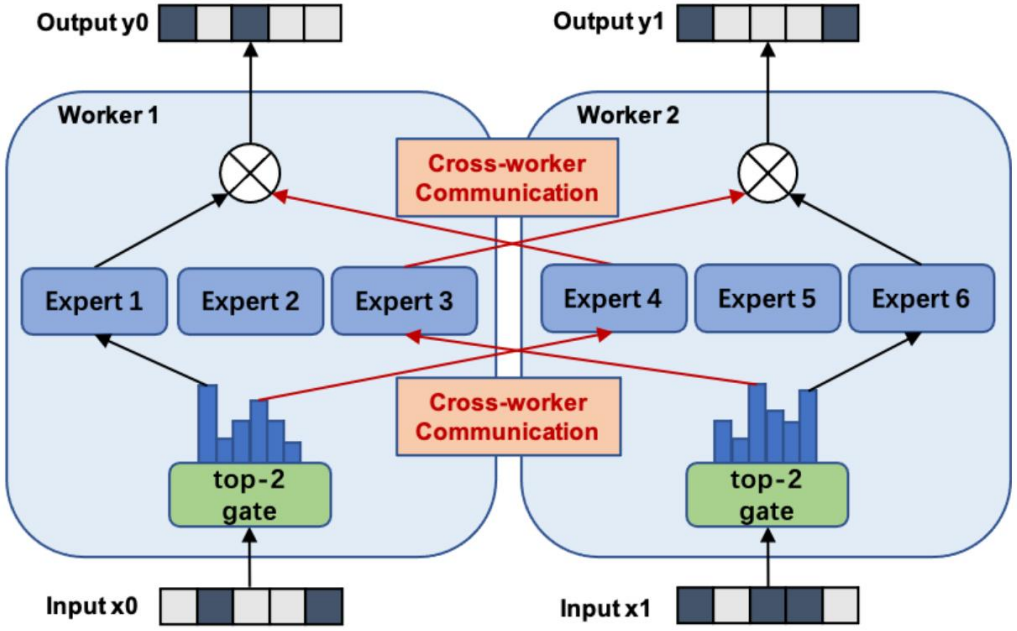
Transformer–XL

- Transformer–XL can be viewed as special case of MoBA where
  - a pre–defined gating function that only attends to the most recent block
  - now grad to historical blocks



FastMoE

- Expert-parallel
- Easy-to-use in Megatron




```
model = ...

from fmoe.megatron import fmoefy
model = fmoefy(model, fmoe_num_experts=<number of experts per worker>)

train(model, ...)
```

He, Jiaao, **Jiezhong Qiu**, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. "Fastmoe: A fast mixture-of-expert training system." arXiv preprint arXiv:2103.13262 (2021).
https://github.com/laekov/fastmoe

Implementation

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.

// Split KV into blocks
$\{\tilde{\mathbf{K}}_i, \tilde{\mathbf{V}}_i\} = \text{split\_blocks}(\mathbf{K}, \mathbf{V}, B)$, where $\tilde{\mathbf{K}}_i, \tilde{\mathbf{V}}_i \in \mathbb{R}^{B \times h \times d}, i \in [n]$
// Compute gating scores for dynamic block selection
$\bar{\mathbf{K}} = \text{mean\_pool}(\mathbf{K}, B) \in \mathbb{R}^{n \times h \times d}$
$\mathbf{S} = \mathbf{Q}\bar{\mathbf{K}}^\top \in \mathbb{R}^{N \times h \times n}$
// Select blocks with causal constraint (no attention to future blocks)
$\mathbf{M} = \text{create\_causal\_mask}(N, n)$
$\mathbf{G} = \text{topk}(\mathbf{S} + \mathbf{M}, k)$
// Organize attention patterns for computation efficiency
$\mathbf{Q}^s, \tilde{\mathbf{K}}^s, \tilde{\mathbf{V}}^s = \text{get\_self\_attn\_block}(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}})$
$\mathbf{Q}^m, \tilde{\mathbf{K}}^m, \tilde{\mathbf{V}}^m = \text{index\_select\_moba\_attn\_block}(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}}, \mathbf{G})$
// Compute attentions seperately
$\mathbf{O}^s = \text{flash\_attention\_varlen}(\mathbf{Q}^s, \tilde{\mathbf{K}}^s, \tilde{\mathbf{V}}^s, \text{causal=True})$
$\mathbf{O}^m = \text{flash\_attention\_varlen}(\mathbf{Q}^m, \tilde{\mathbf{K}}^m, \tilde{\mathbf{V}}^m, \text{causal=False})$
// Combine results with online softmax
$\mathbf{O} = \text{combine\_with\_online\_softmax}(\mathbf{O}^s, \mathbf{O}^m)$
return $\mathbf{O}$

Combine with online softmax

$$o_1 = Attn(q, K[I_1], V[I_1]) = \sum_{i=1}^{a} \frac{\exp(x_i)}{\exp(lse_1)} v_i$$

$$o_2 = Attn(q, K[I_2], V[I_2]) = \sum_{i=a+1}^{a+b} \frac{\exp(x_i)}{\exp(lse_2)} v_i$$

$$lse_1 = \log \sum_{i=1}^{a} \exp(x_i)$$

$$lse_2 = \log \sum_{i=a+1}^{a+b} \exp(x_i)$$

$$o = Attn(q, K[I_1||I_2], V[I_1||I_2]) = \sum_{i=1}^{a+b} \frac{\exp(x_i)}{\exp(lse)} v_i$$

$$m = \max(lse_1, lse_2)$$

$$lse - m = \log[\exp(lse_1 - m) + \exp(lse_2 - m)]$$

$$o = \exp[(lse_1 - m) - (lse - m)]o_1 + \exp[(lse_2 - m) - (lse - m)]o_2$$

Speedup

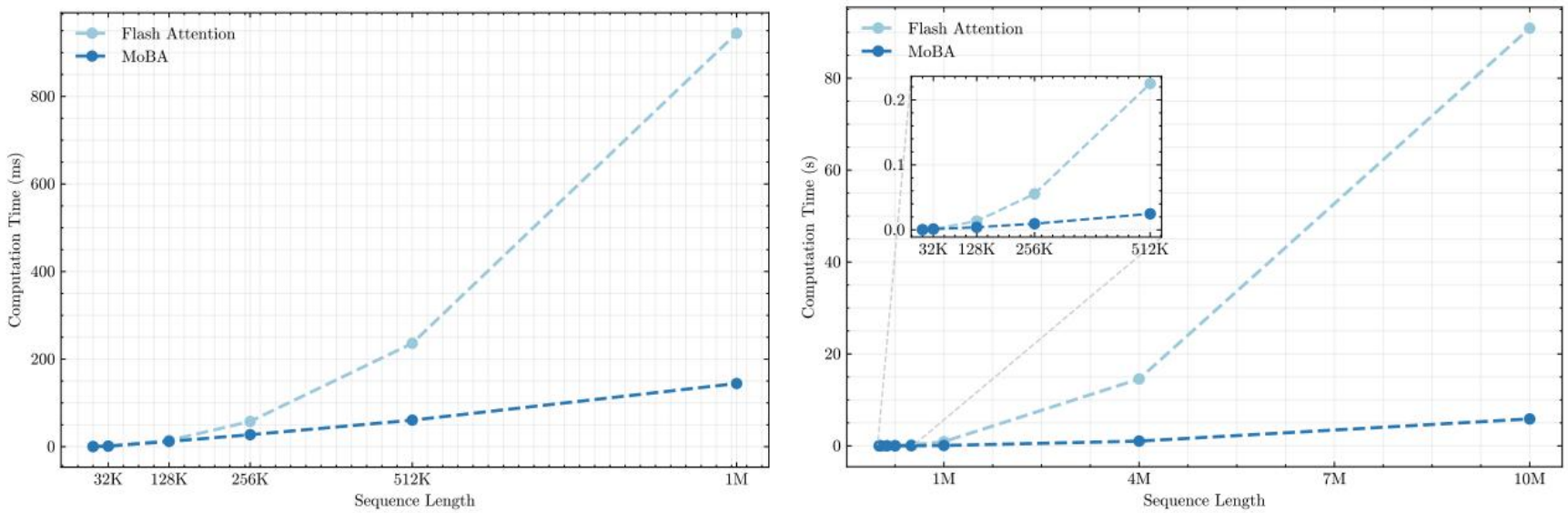
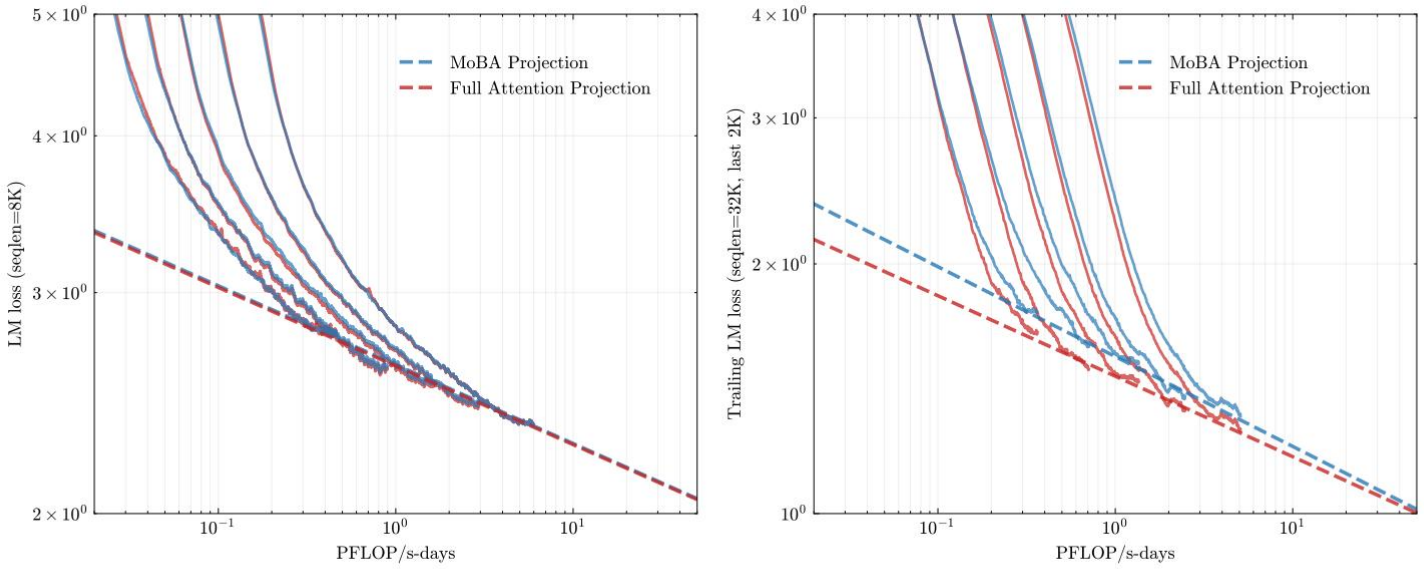


Figure 2: Efficiency of MoBA vs. full attention (implemented with Flash Attention). **(a)** 1M Model speedup evaluation: Computation time scaling of MoBA versus Flash Attention on 1M model with increasing sequence lengths (8K-1M). **(b)** Fixed Sparsity Ratio scaling: Computation time scaling comparison 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).

Scaling Law



(a) (b)

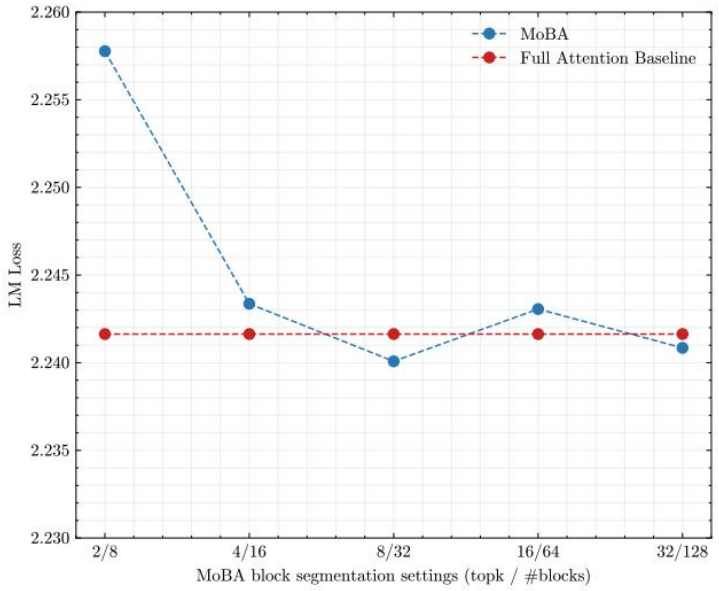
L(C)	MoBA	Full
LM loss (seqlen=8K)	$2.625 \times C^{-0.063}$	$2.622 \times C^{-0.063}$
Trailing LM loss (seqlen=32K, last 2K)	$1.546 \times C^{-0.108}$	$1.464 \times C^{-0.097}$

(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.

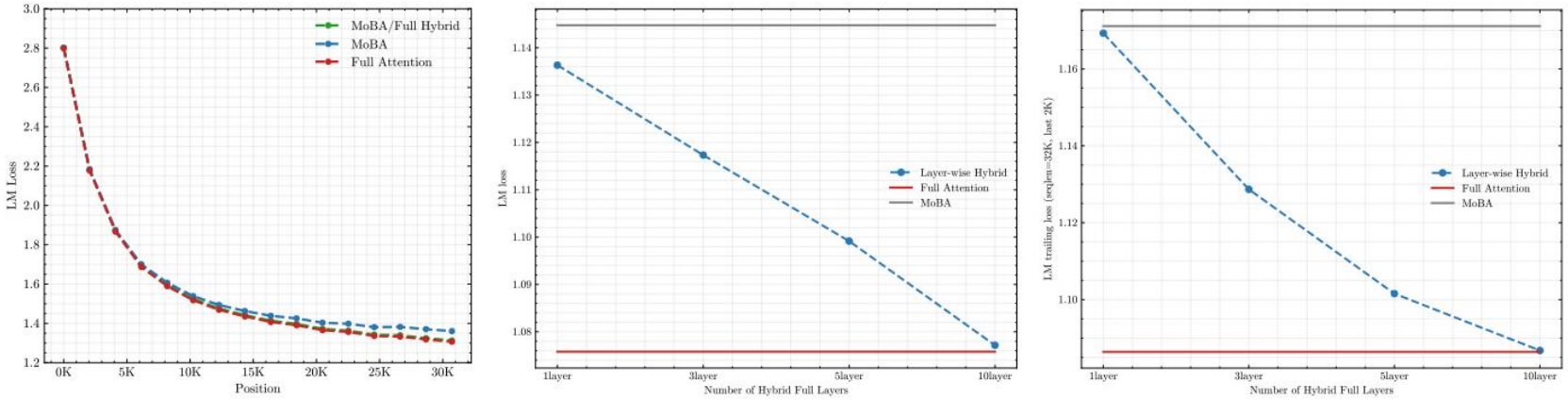
Fine-grained expert -> fine-grained block

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.



Hybrid Strategy of MoBA

- Hybrid Training: 90% tokens with MoBA + 10% with Full Attention
- Layer Hybrid: switch the last several layers from MoBA to full attention



(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.

The post-training recipes (from short to long)

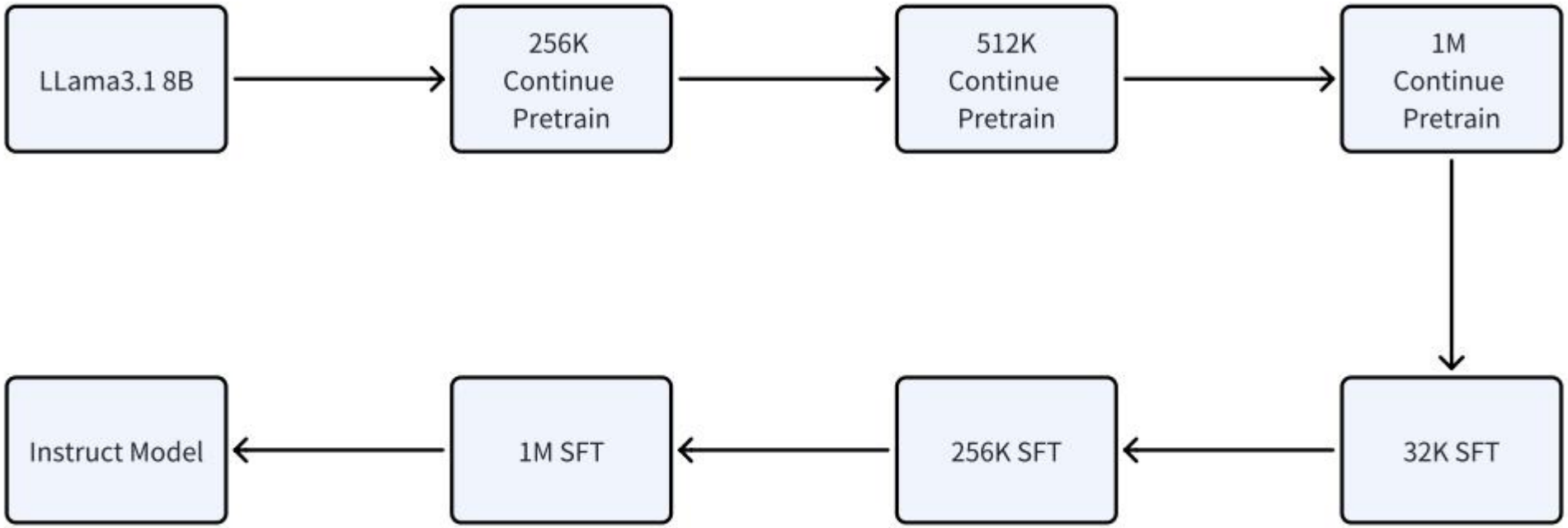


Figure 6: The continual pre-training and SFT recipes.

MoBA v.s. Full

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.

From MoE to MoBA

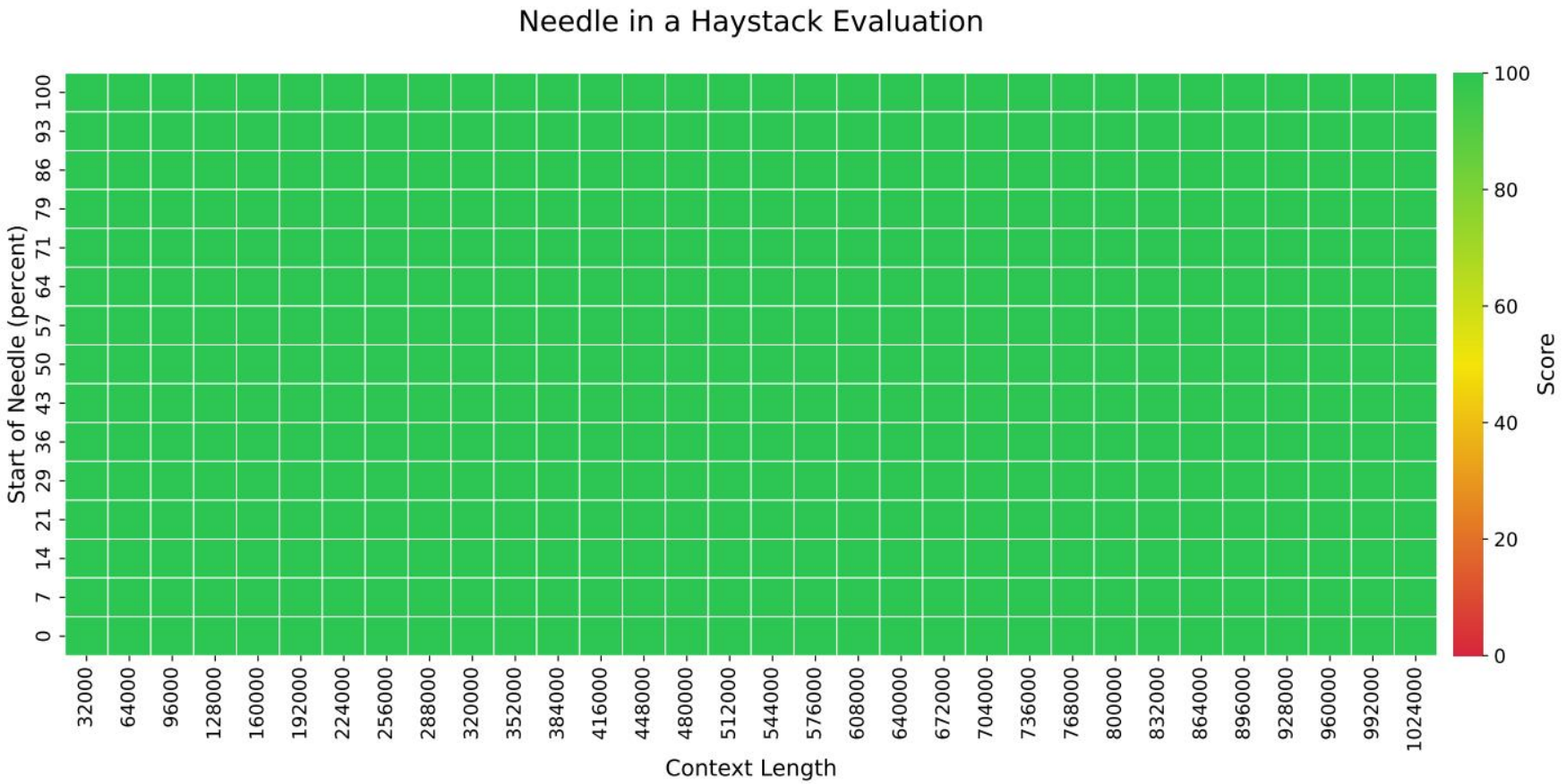
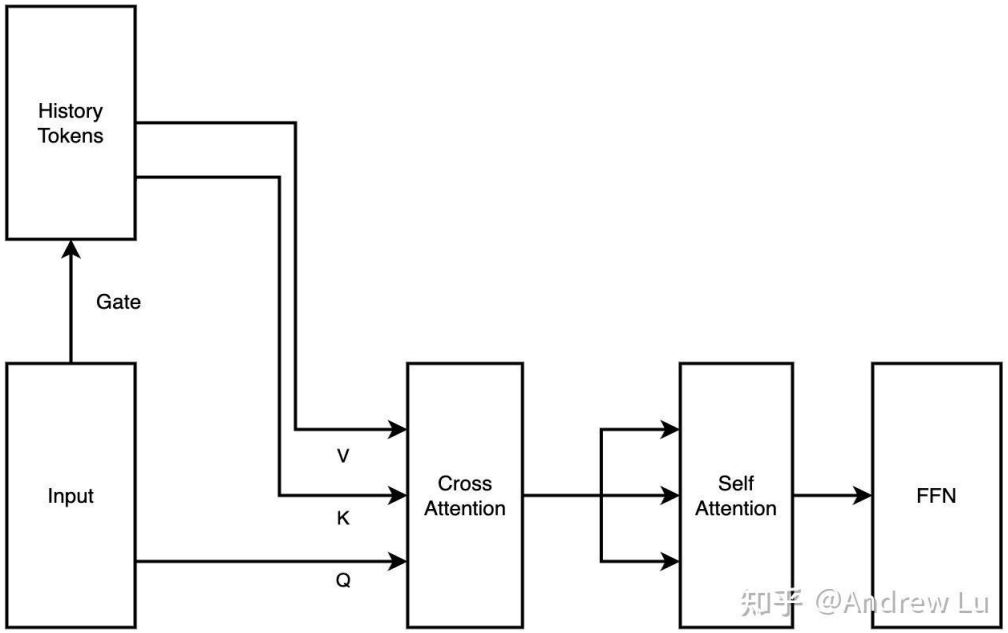


Figure 7: Performance of LLama-8B-1M-MoBA on the Needle in the Haystack benchmark (upto 1M context length).

MoBA v0.5

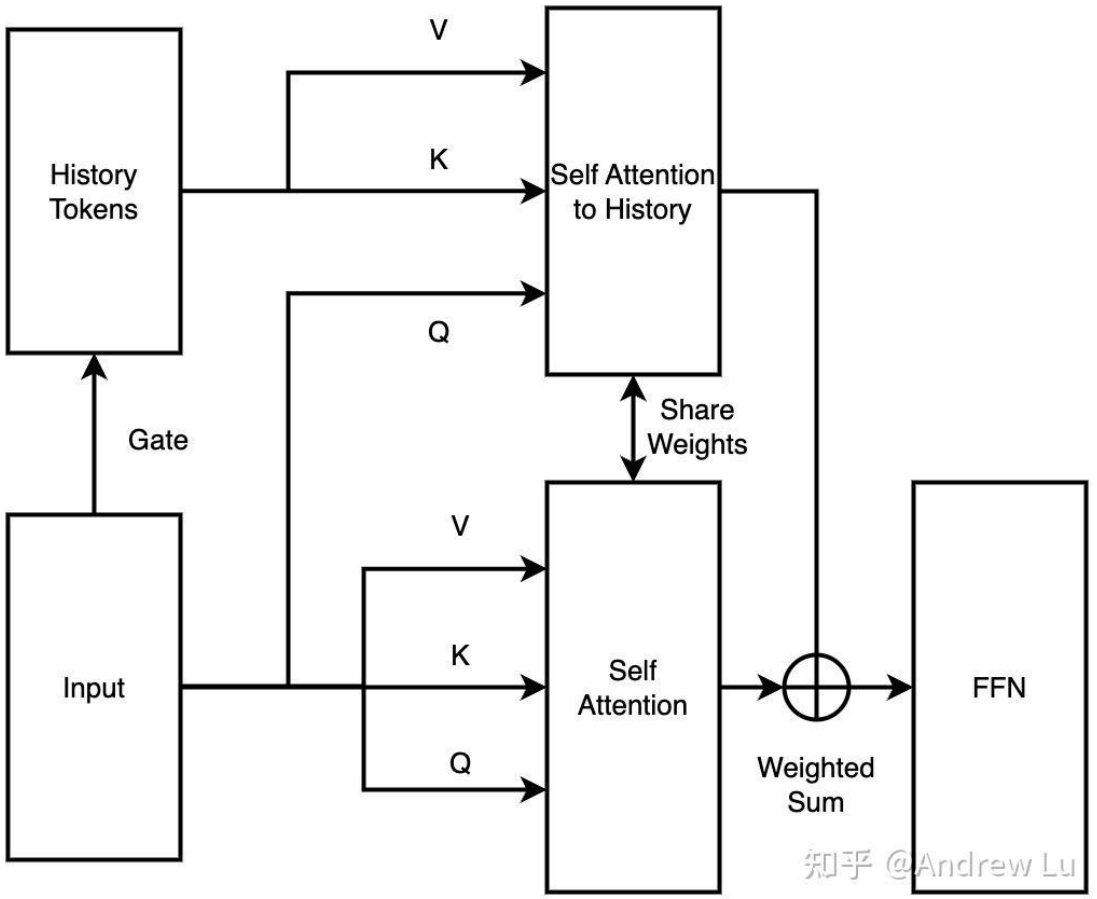
- Cross Attention + Self Attention + FFN
- Bad news: introduce new layers/parameters



MoBA v1.0

- MoBA but with a MoE-style weighted sum

$$o = \sum_i g_i Attn(q, K[I_i], V[I_i])$$



MoBA with Context Parallel

- MoE has expert parallel to support a large number of experts
- Place KV blocks distributedly ...
- Bad news: load balance...

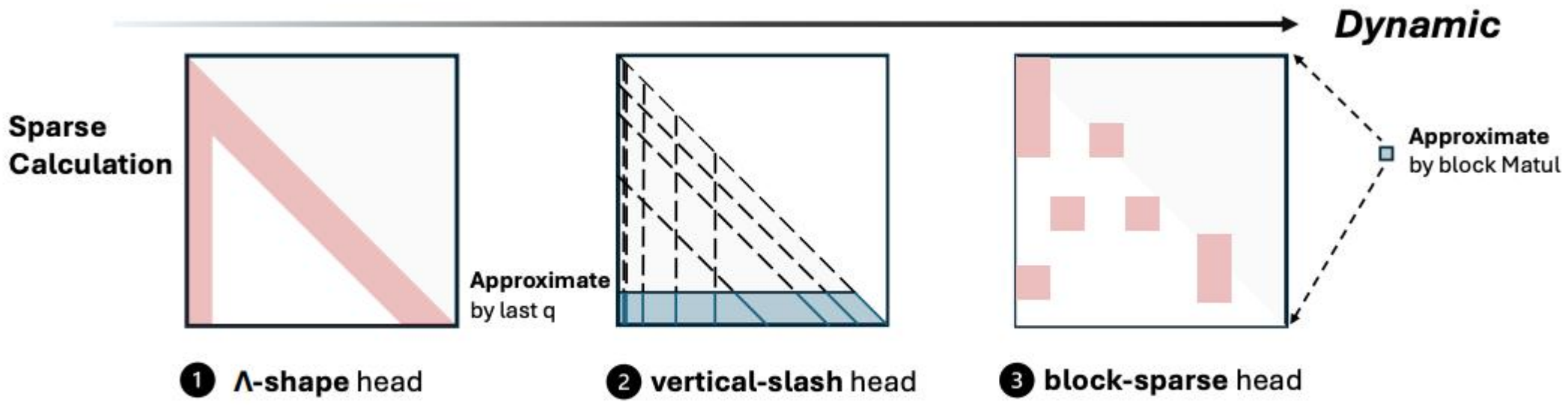


Figure 4: The three sparse methods in MInference.

Thank You!