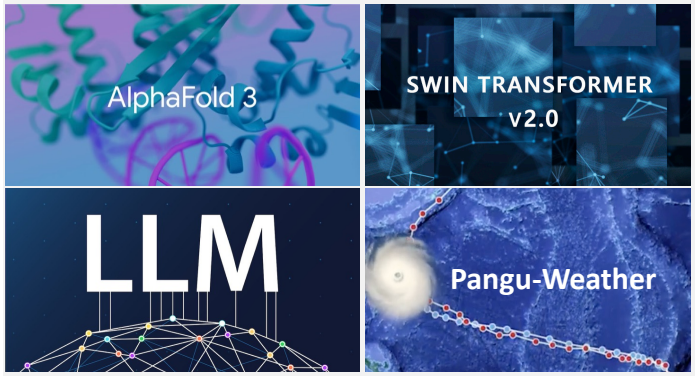
FlashBias: Fast Computation of Attention with Bias

Tsinghua University

Haixu Wu, Minghao Guo, Yuezhou Ma, Yuanxu Sun, Jianmin Wang, Wojciech Matusik, Mingsheng Long#

NEURAL INFORMATION PROCESSING SYSTEMS

Attention with Bias $\mathbf{o} = \mathrm{softmax}(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{C}} + \mathbf{b})\mathbf{v}.$

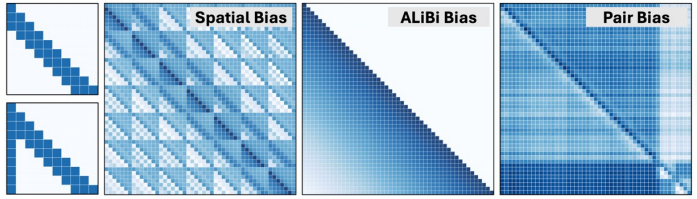


Flex/FlashAttention Fails, Try FlashBias!

1.5x speedup for **Pairformer in Alphafold 3**

2x speedup for **Swin Transformer v2**

Key Challenge: Attention Bias is not sparse.



Have to load *NxN* bias matrix from HBM

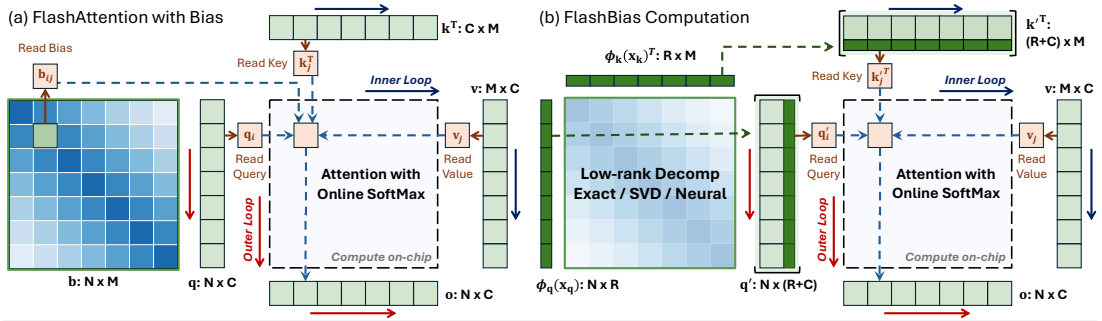
HBM *1.5 TB/s (40 GB)* — Query, Key, Value, **Bias** → / ← Results — **SRAM** 19 TB/s (20 MB)

Get FlashBias kernel at

https://github.com/thuml/FlashBias

wuhaixu98@gmail.com

FlashBias: Reach the Theoretical Efficiency Upper Bound



➢ **Why FlashAttention is fast?** **Underlying low rank assumption**

Given Sequence len *N*, Channel dim *C*, SRAM size *S* and *C=αN, S=βNC*

1) FlashAttention IO Complexity is $\Theta\left(\left(1+\frac{1}{\alpha}\right)\beta\right)$ *smaller than standard attention*

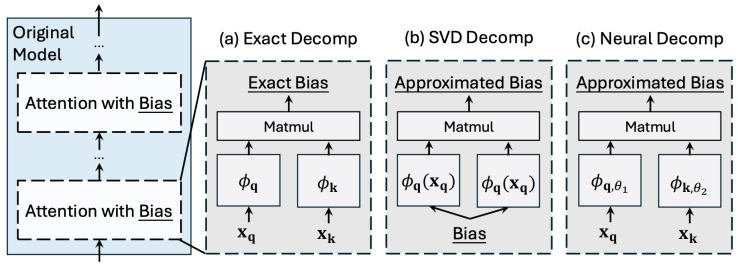
2) Suppose attention weight is of rank R, $\alpha \geq \frac{R}{N}$ *(determines the optimal speedup)*

➢ **FlashBias based on low-rank compressed sensing theory**

1) Low-rank Decomp $\mathbf{b} = f(\mathbf{x_q}, \mathbf{x_k}) = \phi_\mathbf{q}(\mathbf{x_q})\phi_\mathbf{k}(\mathbf{x_k})^\top, \quad \phi_\mathbf{q}, \phi_\mathbf{k} : \mathbb{R}^{C'} \to \mathbb{R}^R.$

2) Fast compute $\mathrm{softmax}(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{C}} + \mathbf{b})\mathbf{v} = \mathrm{softmax}\left(\frac{[\mathbf{q}|\sqrt{C}\phi_\mathbf{q}(\mathbf{x_q})]\,[\mathbf{k}|\phi_\mathbf{k}(\mathbf{x_k})]^\top}{\sqrt{C}}\right)\mathbf{v}$

Theorem: FlashBias achieves optimal efficiency guaranteed by compressed sensing theory

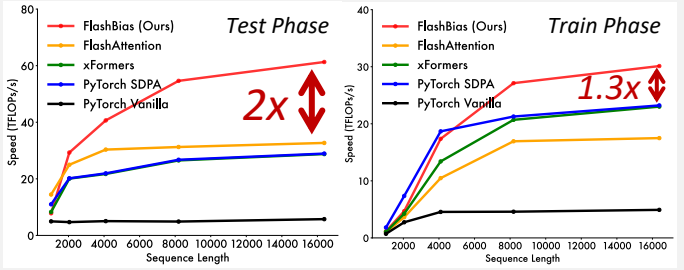


Domain	Bias / Model	Type
Language	ALiBi [29]	(a)
Vision	Swin Trans. [24]	(b)
Science	Spatial Distance	(a)
	Pangu-Weather [4]	(b)
	AlphaFold [1]	(c)

Three concrete instantiations for decomposition: Exact, SVD and Neural Decomp

Speed up without any loss of accuracy

Surpass FlashAttention, PyTorch SDPA, xFormers



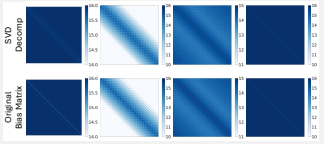
GPT-2 with ALiBi (R=2, Exact decomp)

$f(\mathbf{x}_{\mathbf{q},i}, \mathbf{x}_{\mathbf{k},j}) = i - j\,,\ \phi_\mathbf{q}(\mathbf{x}_{\mathbf{q},i}) = [1, i]\ and\ \phi_\mathbf{k}(\mathbf{x}_{\mathbf{k},j}) = [-j, 1]$

Inference: **52.5→91.6**TFLOPs/s; Train: **42.7→51.8**TFLOPs/s

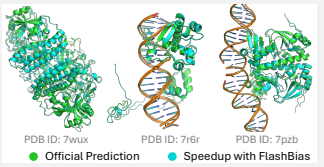
SwinV2-B (R=16, SVD decomp) **0.473s→0.190s**

Method	Acc@1	Acc@5	Time(s)	Mem(MB)
Official Code	87.144%	98.232%	0.473	12829
Pure FlashAttention	9.376%	19.234%	0.180	3957
FlashAttention with Bias	87.142%	98.232%	0.230	11448
FlexAttention [11]	87.142%	98.232%	2.885	25986
INT8 PTQ	86.46%	*Around 22% speed up*		
FlashBias (Ours)	87.186%	98.220%	0.190	9429



AlphaFold 3 (R=94, Neural decomp) **26.8s→18.2s**

Method	PDB ID 7wux		
	pTM ↑	Time(s)	Mem(GB)
Open-sourced Code	0.9500	26.85	13.62
FlashAttention w/o Bias	0.1713	8.27	12.89
FlashAttention w/ Bias	0.9500	20.39	13.62
FlashBias (Ours)	0.9498	18.19	13.62



Ack: AlphaFold 3 is based on the Protenix from ByteDance

Easy to use API: Try FlashBias!

```
>> from flash_bias_triton import flash_bias_func
>> output = flash_bias_func(q, k, v, q_bias, k_bias,
                            mask=None, causal=False,
                            softmax_scale=1/math.sqrt(headdim))
```