Advanced Attention Mechanisms - II

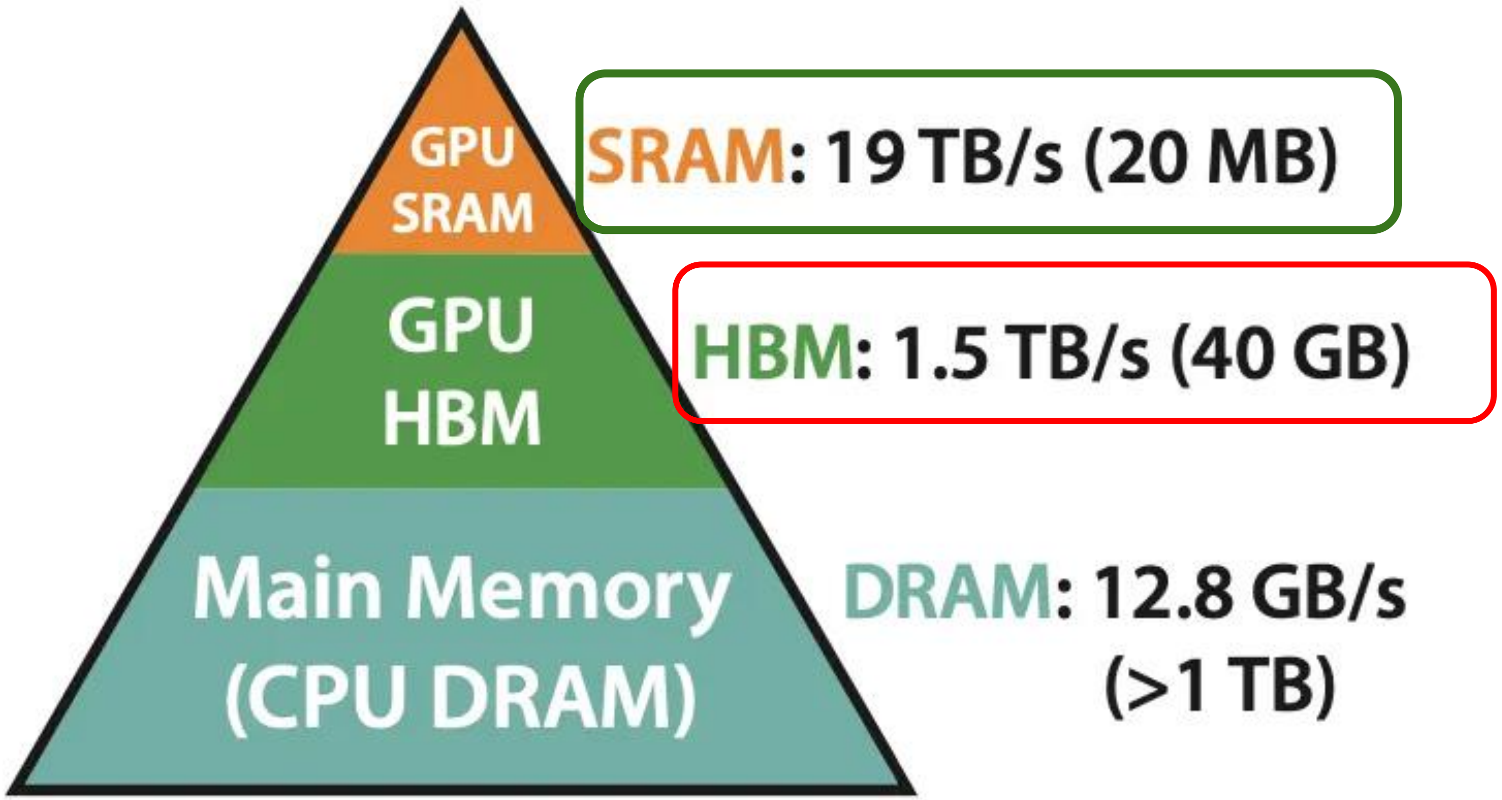
Large Language Models: Introduction and Recent Advances

ELL881 · AIL821

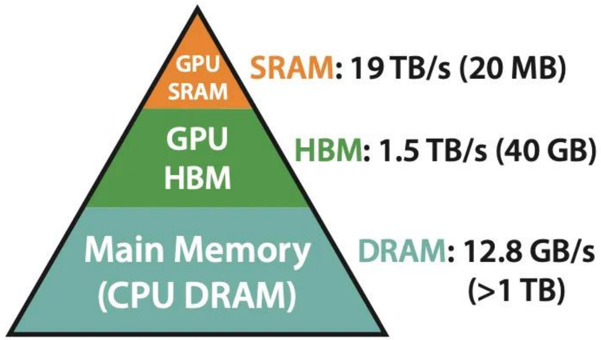
Sourish Dasgupta
Assistant Professor, DA-IICT, Gandhinagar
https://daiict.ac.in/faculty/sourish-dasgupta

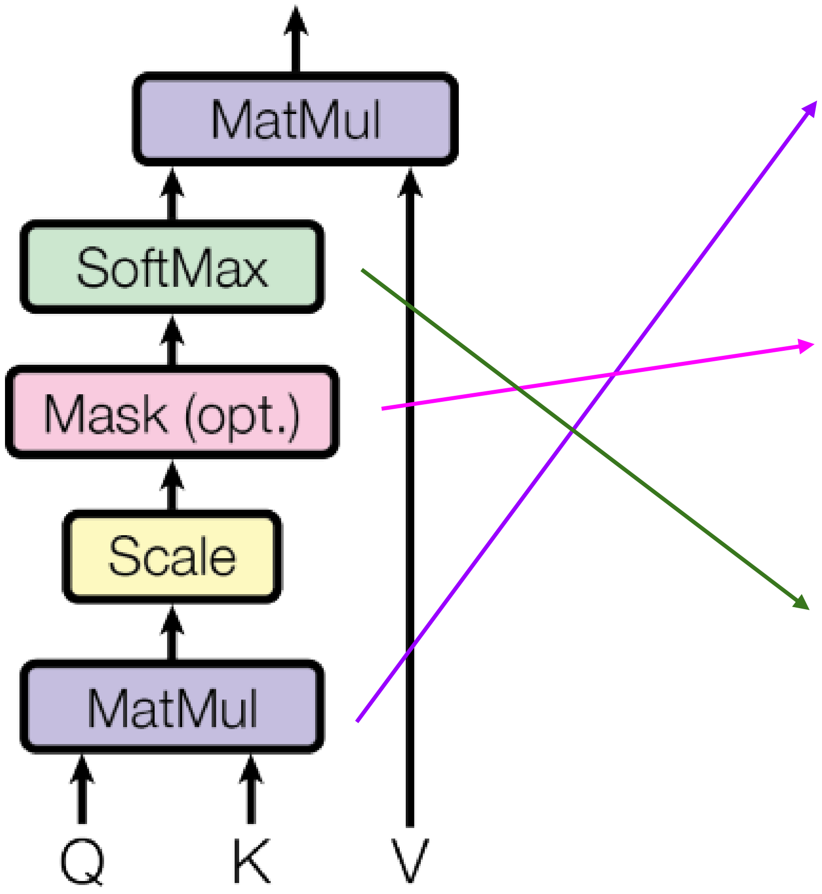
Can we optimize without performance degradation?

A bit more about the GPU



What was happening so far:

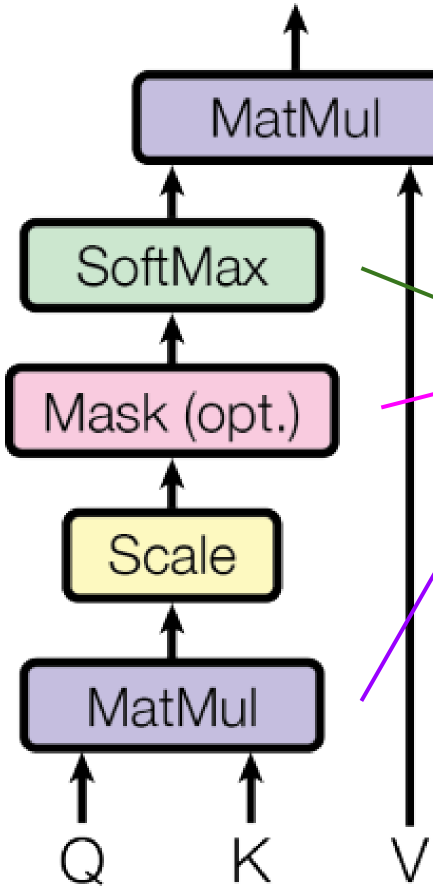




1. Matmul_op (Q,K)
   a. Read Q,K to SRAM (read-op)
   b. Compute matmul A=QxK (compute-op)
   c. Write A to HBM (write-op)
2. Mask_op
   a. Read A to SRAM (read-op)
   b. Mask A into A' (compute-op)
   c. Write A' to HBM (write-op)
3. Softmax_op
   a. Read A' to SRAM (read-op)
   b. Softmax A' into A'' (compute-op)
   c. Write A'' to HBM (write-op)

Year: 2022; NeurIPS

The magic: *Fused Kernel* (GPU Operations)!



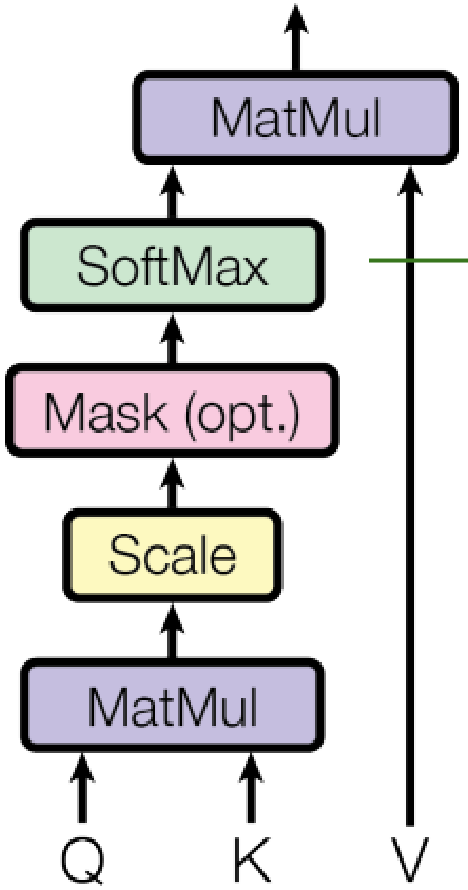
Flash Attention

1. Read Q,K to SRAM
2. Compute A = QxK
3. Mask A into A'
4. Softmax A' into A''
5. Write A'' to HBM

1. Matmul_op (Q,K)
   a. Read Q,K to SRAM (read-op)
   b. Compute matmul A=QxK (compute-op)
   c. Write A to HBM (write-op)
2. Mask_op
   a. Read A to SRAM (read-op)
   b. Mask A into A' (compute-op)
   c. Write A' to HBM (write-op)
3. Softmax_op
   a. Read A' to SRAM (read-op)
   b. Softmax A' into A'' (compute-op)
   c. Write A'' to HBM (write-op)

Year: 2022; NeurIPS

The magic does not end here! More optimization

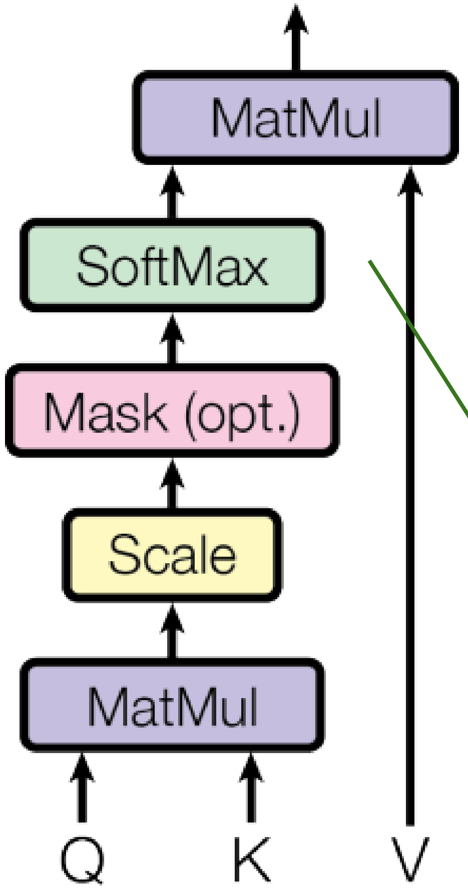


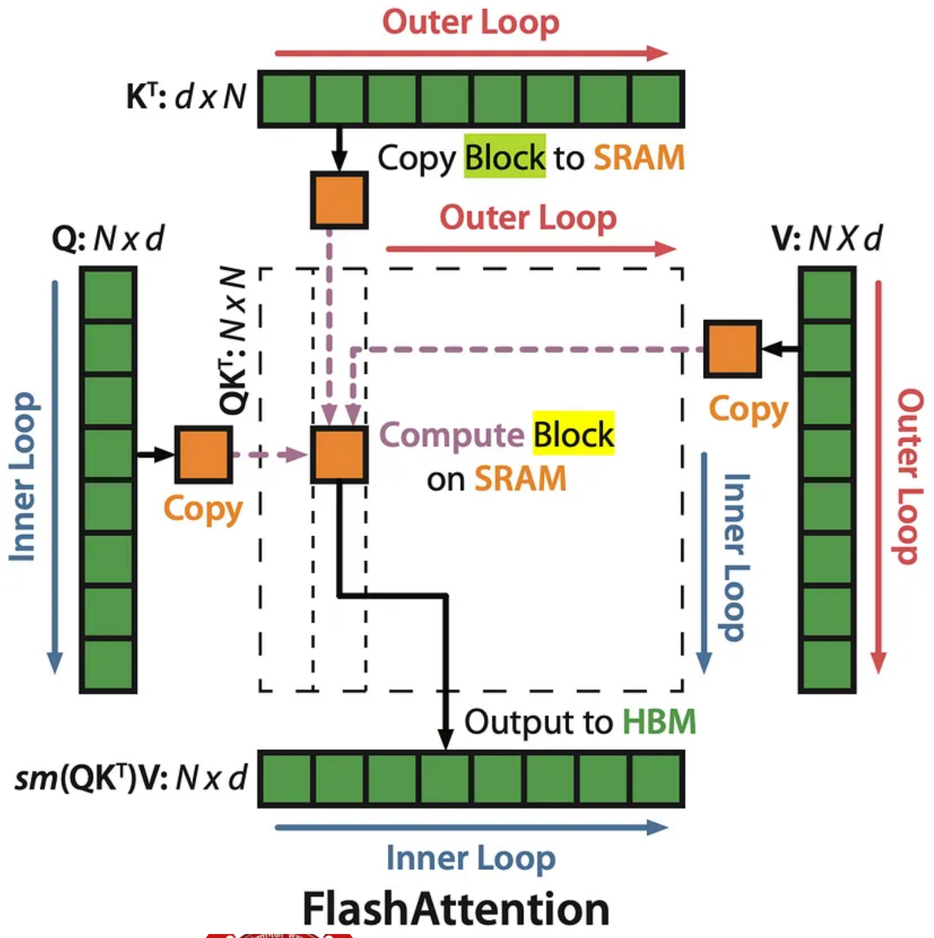
$$s(x_i) = \frac{e^{x_i}}{\boxed{\sum_{j=1}^{N} e^{x_j}}}$$

Gets computed for every row - *Problem!*

Year: 2022; NeurIPS

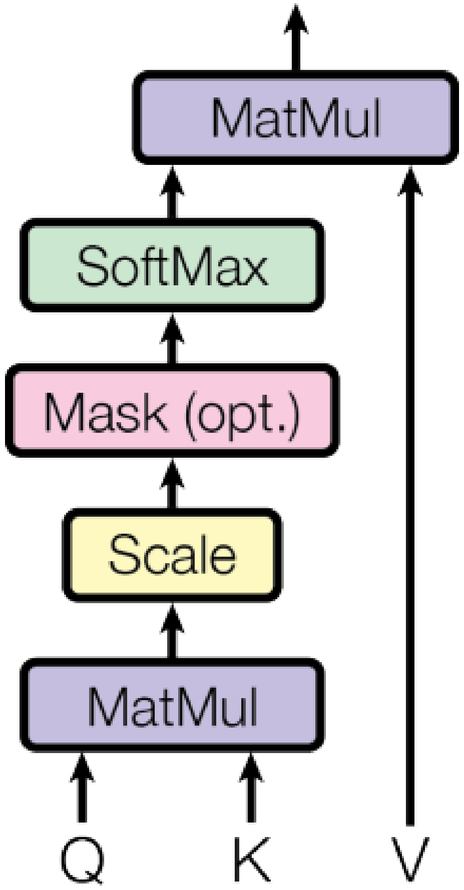
The magic does not end here! *Tiling*



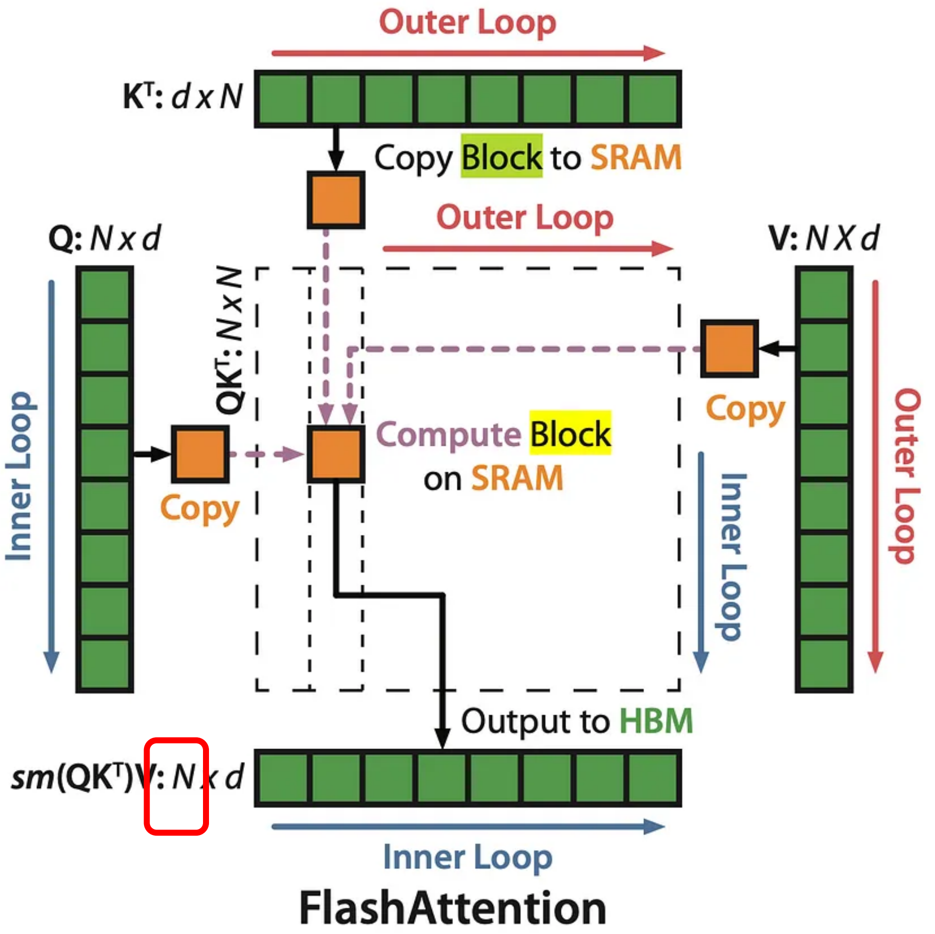


Year: 2022; NeurIPS

Does the story end here? What's the problem?

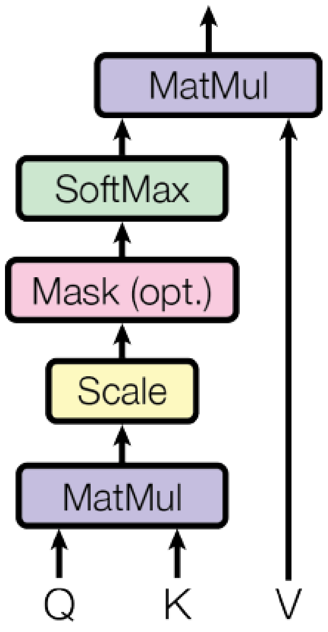


$$s(x_i) = \frac{e^{x_i}}{\boxed{\sum}_{j=1}^{N} e^{x_j}}$$



Year: 2022; NeurIPS

The softmax denominator problem



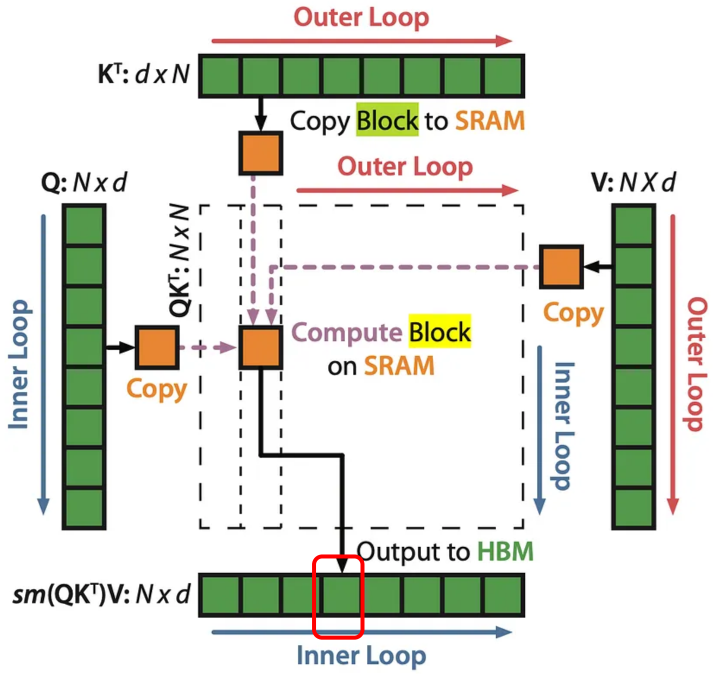
$$Q = [1] \qquad K = [1, 2, 3] \qquad V = [2, 4, 8]$$

$$A = QK^T = [1, 2, 3] \qquad V = [2, 4, 8]$$

$$O = \text{softmax}(A)V$$

$$O = \frac{N}{D} = \frac{2e^1 + 4e^2 + 8e^3}{\boxed{e^1 + e^2 + e^3}}$$

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$



FlashAttention

$$\text{At} \quad i = 0$$

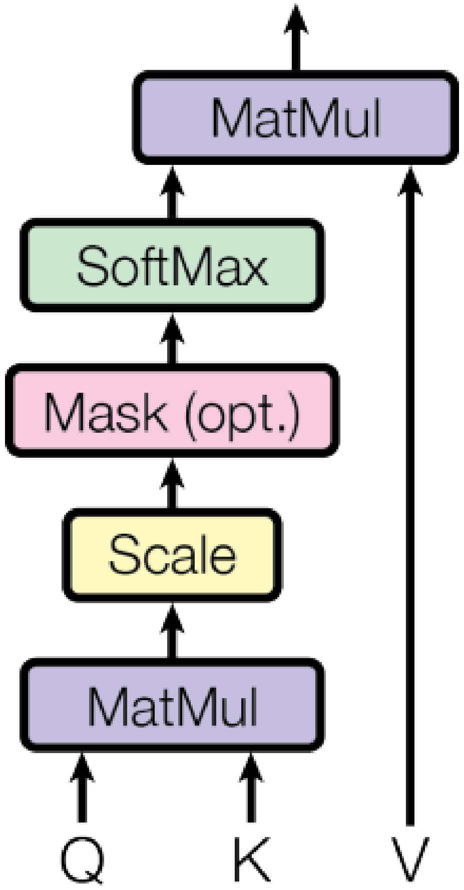
$$D_b = e^1$$

$$N_b = 2e^1$$

$$O = \frac{1}{e^1}[0 + 2e^1]$$

Year: 2022; NeurIPS

Summary Statistics - the final touch!



$Q = [1] \quad K = [1,2,3] \quad V = [2,4,8]$

$A = QK^T = [1,2,3] \quad V = [2,4,8]$

$D = 0, O = 0 \longrightarrow O = \dfrac{2e^1 + 4e^2 + 8e^3}{e^1 + e^2 + e^3}$

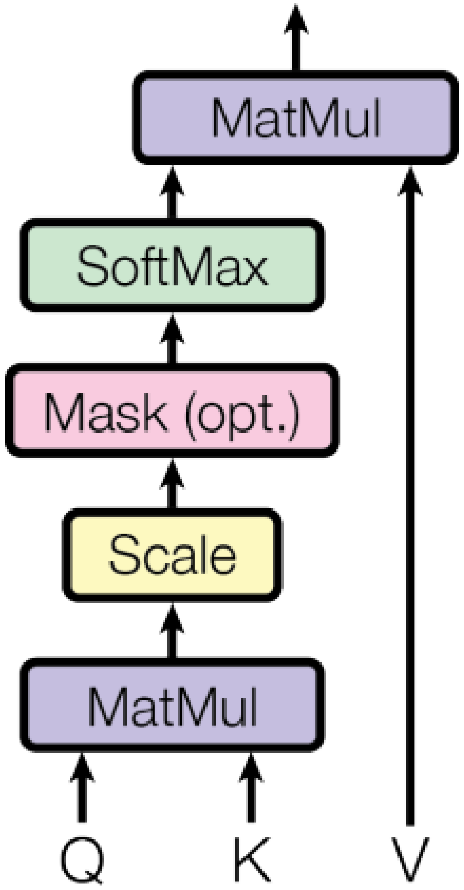
At $i = 0$	At $i = 1$	At $i = 2$
$D_b = e^1$	$D_b = e^2$	$D_b = e^3$
$N_b = 2e^1$	$N_b = 4e^2$	$N_b = 8e^3$
$O = \frac{1}{e^1}[0 + 2e^1]$	$O = \frac{1}{e^1 + e^2}[\boxed{2e^1} + 4e^2]$	$O = \frac{1}{e^1 + e^2 + e^3}[\boxed{2e^1 + 4e^2} + 8e^3]$
$D = e^1$	$D = e^1 + e^2$	$D = e^1 + e^2 + e^3$

$D * O$

$D * O$

Year: 2022; NeurIPS

Summary Statistics - the final touch!



$Q = [1] \quad K = [1, 2, 3] \quad V = [2, 4, 8]$

$A = QK^T = [1, 2, 3] \quad V = [2, 4, 8]$

$O = \text{softmax}(A)V$

$$O = \frac{N}{D} = \frac{2e^1 + 4e^2 + 8e^3}{e^1 + e^2 + e^3}$$

$D = 0, O = 0$

```
# Treat each element as a block,
# so we have three blocks
for i in range(3):
```

$D_b = exp(Q[i] \times K[i])$

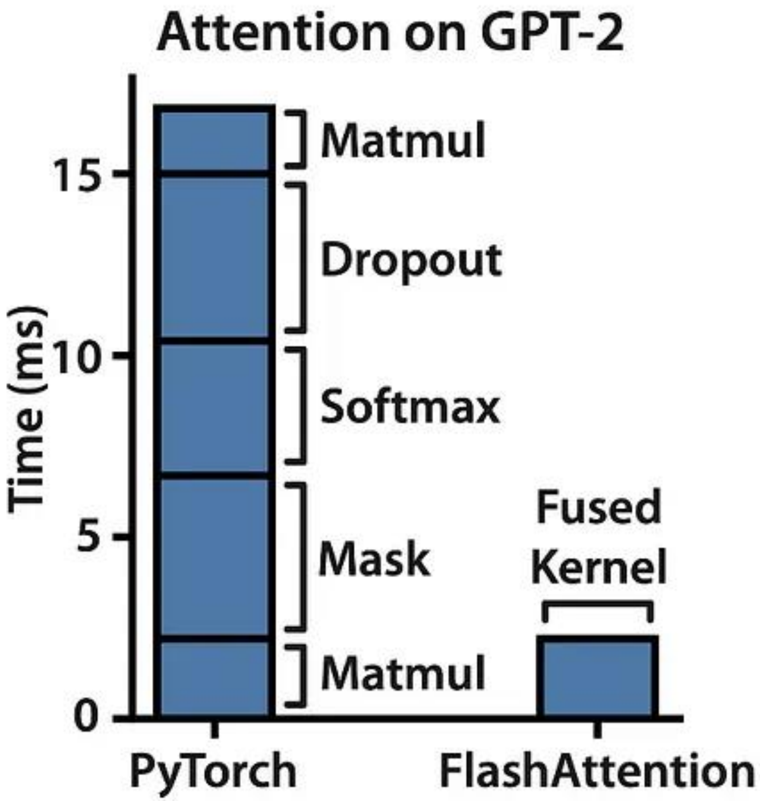
$N_b = V[i] * exp(Q[i] \times K[i])$

$\boxed{O = \frac{1}{D + D_b}[D * O + N_b]}$

$D = D + D_b$

Year: 2022; NeurIPS

How well did they do?



BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FlashAttention (ours)	**17.4** ± 1.4

*Time: O(N*d)*
*Space: O(N*d)*

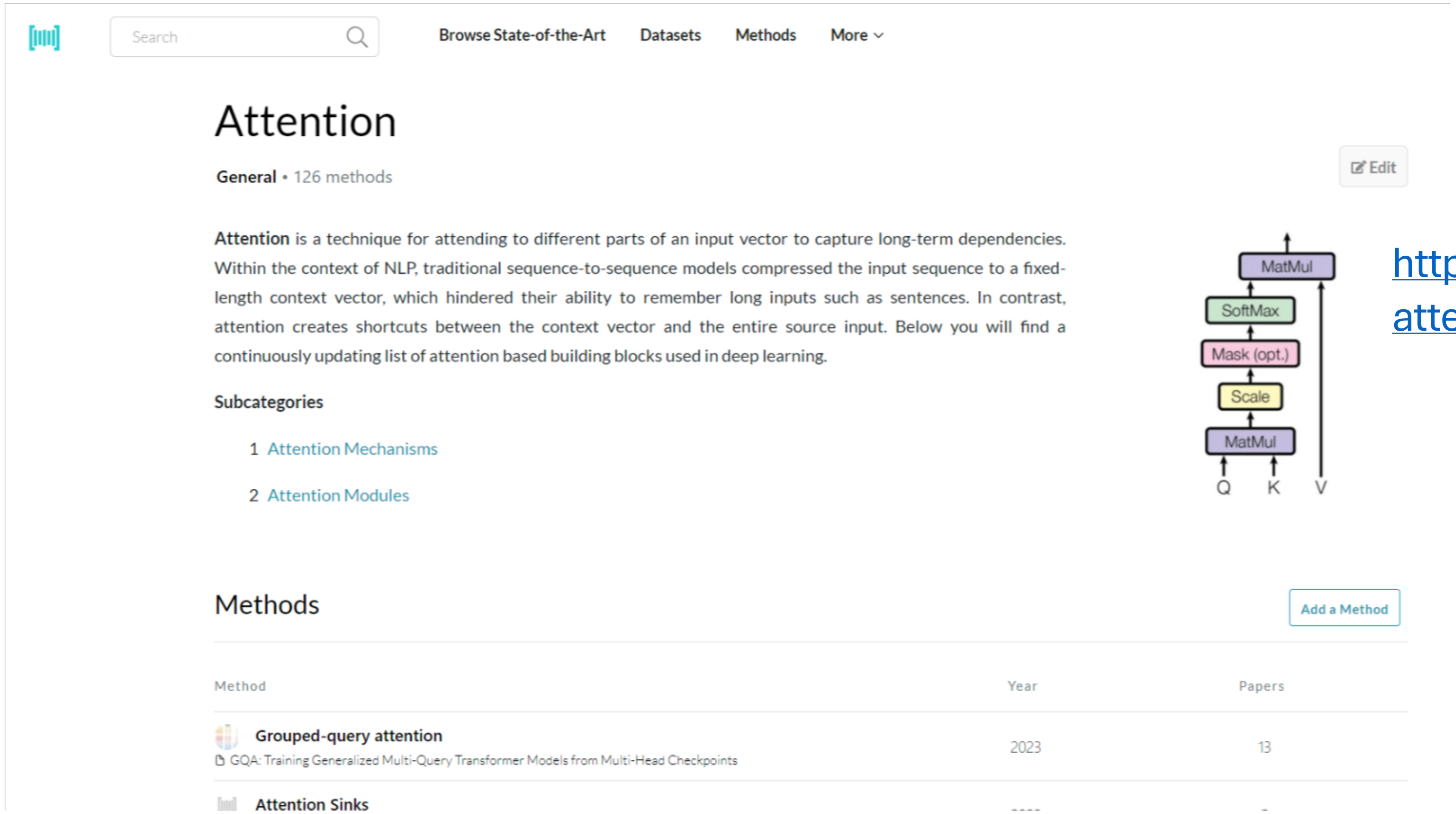
So have we finally solved the attention hurdle?

- *Does your GPU come with?*
  - CUDA (or have to be re-written in ROCm (AMD) or SYCL (Intel))
  - Fast shared GPU memory (SRAM)
  - Tensor cores (specifically dedicated to matrix operations)
- Too much pro-NVIDIA (Ampere, Volta, etc.)
- A new attention on the block? Have to rewrite the fused kernel

Key Takeaways

- Avoid unnecessary HBM read/write
- Maximize SRAM computation

Want more? Follow:



Search

Browse State-of-the-Art Datasets Methods More

Attention

General • 126 methods

Edit

Attention is a technique for attending to different parts of an input vector to capture long-term dependencies. Within the context of NLP, traditional sequence-to-sequence models compressed the input sequence to a fixed-length context vector, which hindered their ability to remember long inputs such as sentences. In contrast, attention creates shortcuts between the context vector and the entire source input. Below you will find a continuously updating list of attention based building blocks used in deep learning.

Subcategories

1 Attention Mechanisms

2 Attention Modules

Methods

Add a Method

Method	Year	Papers
Grouped-query attention GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints	2023	13
Attention Sinks		

https://paperswithcode.com/methods/category/attention-mechanisms