Large Language Models

Advanced Attention Mechanisms - I

ELL881 · AIL821

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

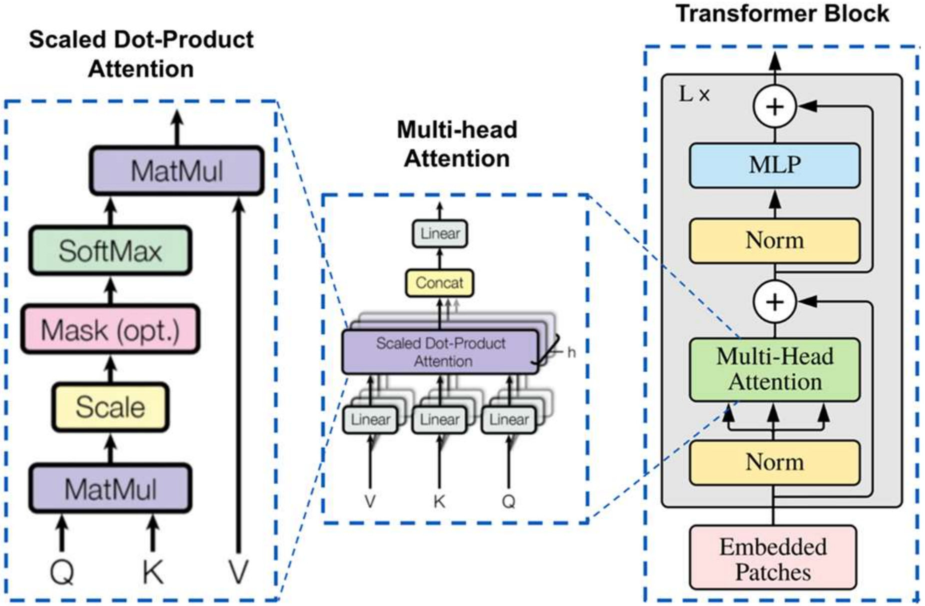
Semester 1, 2024-2025

Year: 2017, NeurIPS

Google Brain Team

Make machines intelligent. Improve people's lives.

Self Attention



similarity.

$$\frac{Q_{(6,\,4096)} \times K^T_{(4096,\,6)}}{\sqrt{4096}} =$$

contextual embedding

$$Attention(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	THE	CAT	IS	ON	A	CHAIR
THE	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
ON	0.210	0.128	0.206	0.212	0.119	0.125
A	0.146	0.158	0.152	0.143	0.227	0.174
CHAIR	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

softmax(loves. John K)

% ? % %

John loves Ice made in NY v.

v v v v v

Q % % v

v: values.

Year: 2017, NeurIPS

Google Brain Team

Make machines intelligent. Improve people's lives.

Causal (Forward Masked) Attention

N: sequence length.

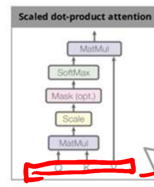
d: dimension.

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\frac{\underset{(6,\ 4096)}{Q} \times \underset{(4096,\ 6)}{K^T}}{\sqrt{4096}} =$$

	THE	CAT	IS	ON	A	CHAIR
THE	0.268	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
CAT	0.124	0.278	$-\infty$	$-\infty$	$-\infty$	$-\infty$
IS	0.147	0.132	0.262	$-\infty$	$-\infty$	$-\infty$
ON	0.210	0.128	0.206	0.212	$-\infty$	$-\infty$
A	0.146	0.158	0.152	0.143	0.227	$-\infty$
CHAIR	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)



similarity (dot product)

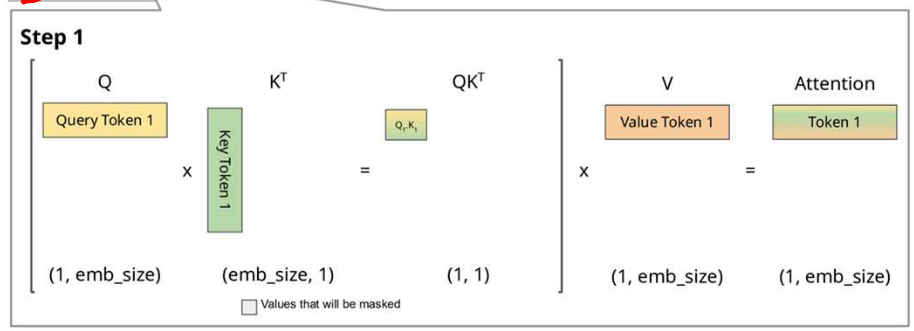
softmax

$\times V$

Time: $\underbrace{O(N^2*d)}_{}$ + $O(N^2)$ + $O(N^2*d)$ = $O(N^2*d)$

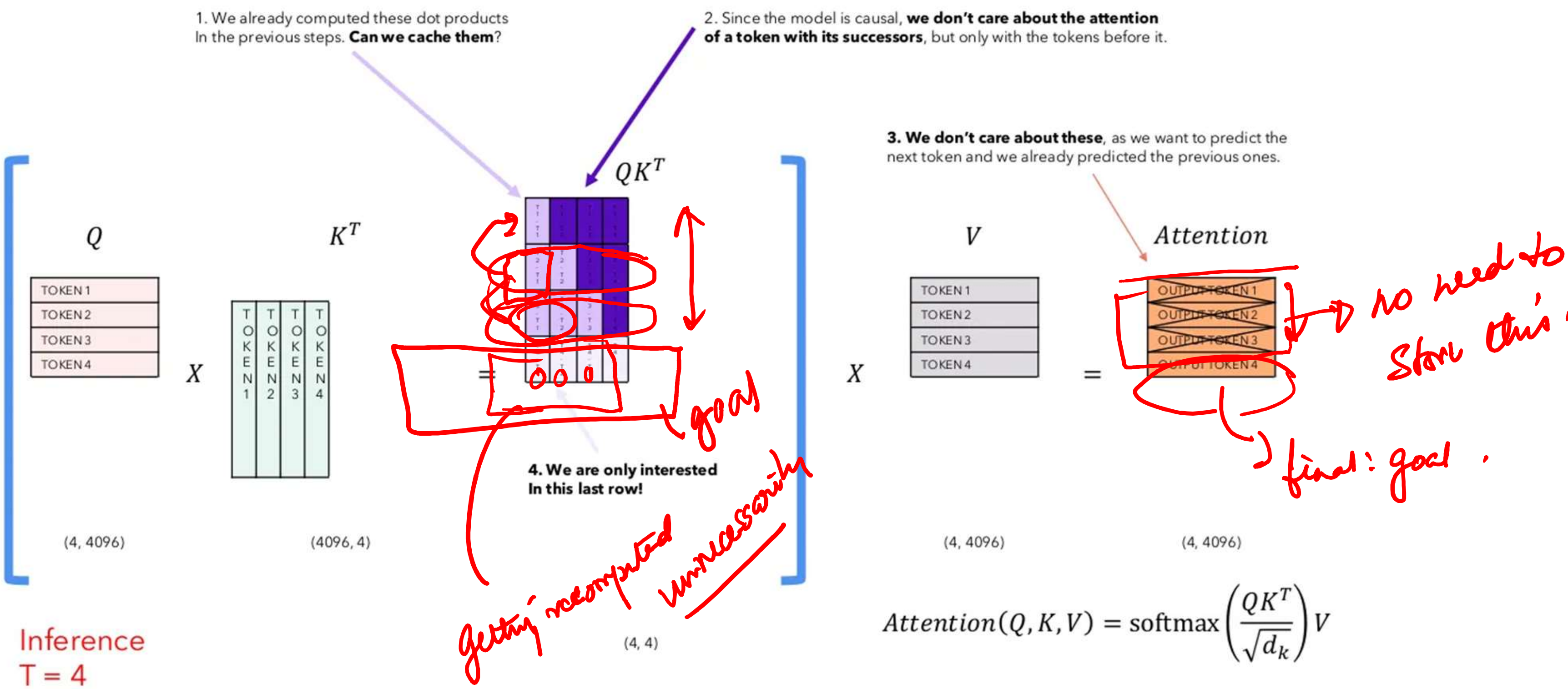
Space: $O(3*N*d)$ + $O(N^2)$ + $O(N*d)$ = $O(N^2 + N*d)$

$= (3 \times N \times d + N^2) \times$ Size of a float



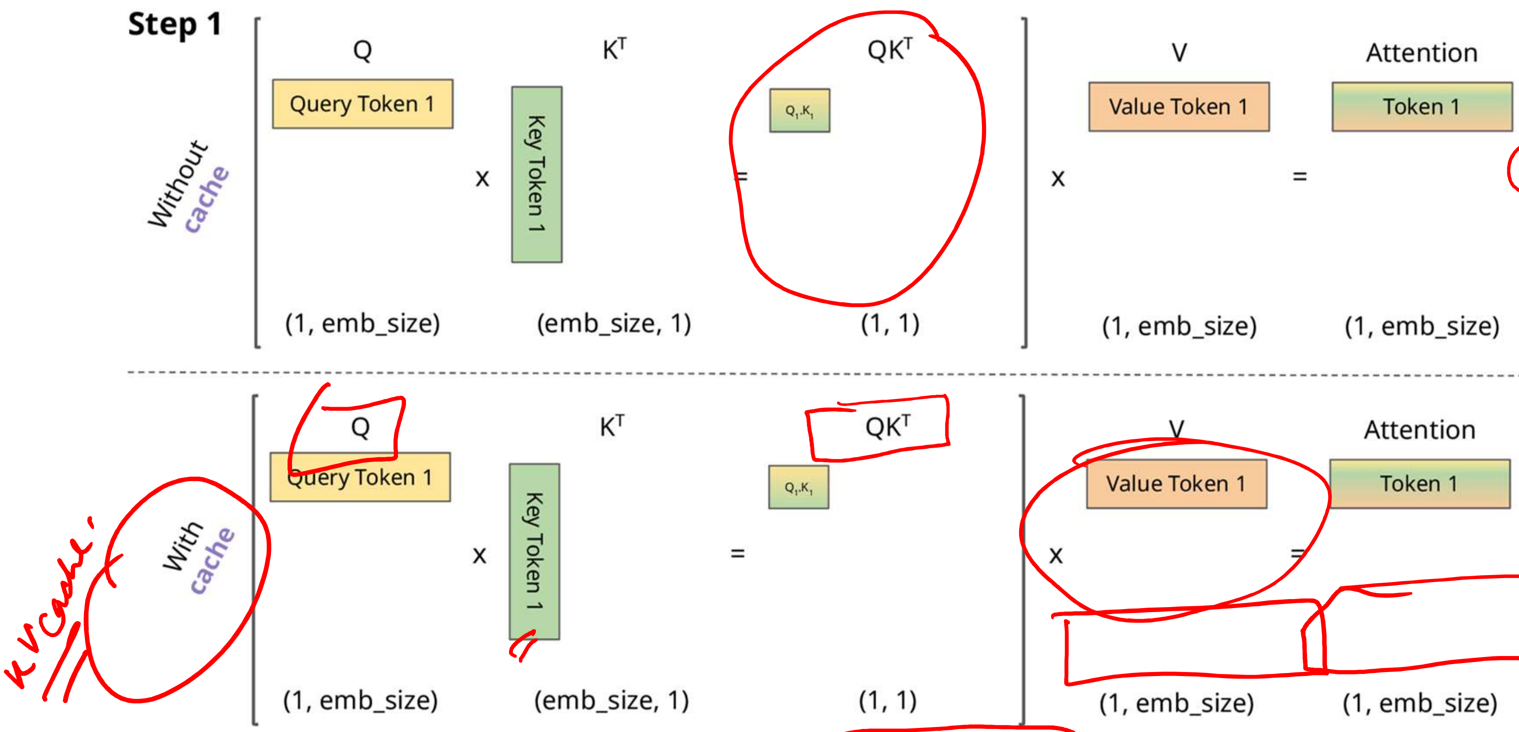
Zoom-in! (simplified without Scale and Softmax)

Why do we need to do better?



$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

KV Cache based (Forward Masked) Attention



KV Cache Storage = O(N*d)

$(2 \times N \times d) \times$ *Size of a float*

vs.

$(3 \times N \times d + N^2) \times$ *Size of a float*

Source: https://medium.com/@joaolages/kv-caching-explained-276520203249

Year: 2020, Arxiv

Ai2 Allen Institute for AI

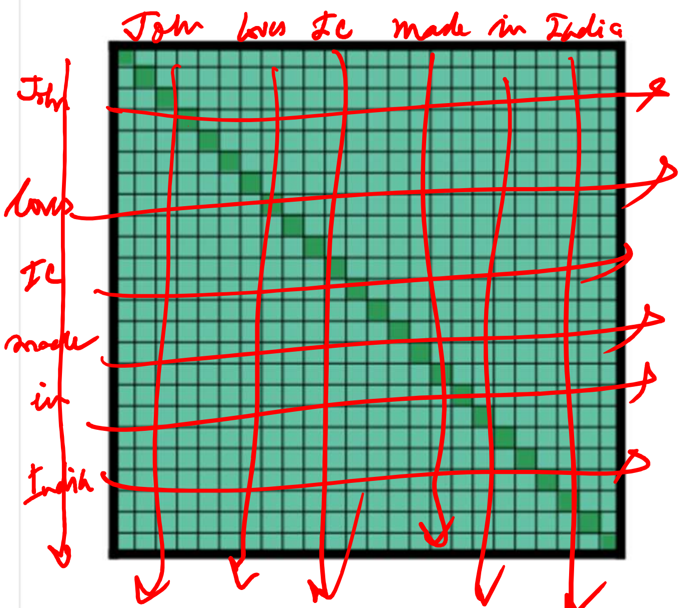
Sliding Window Attention

→ Any LM (not just causal)

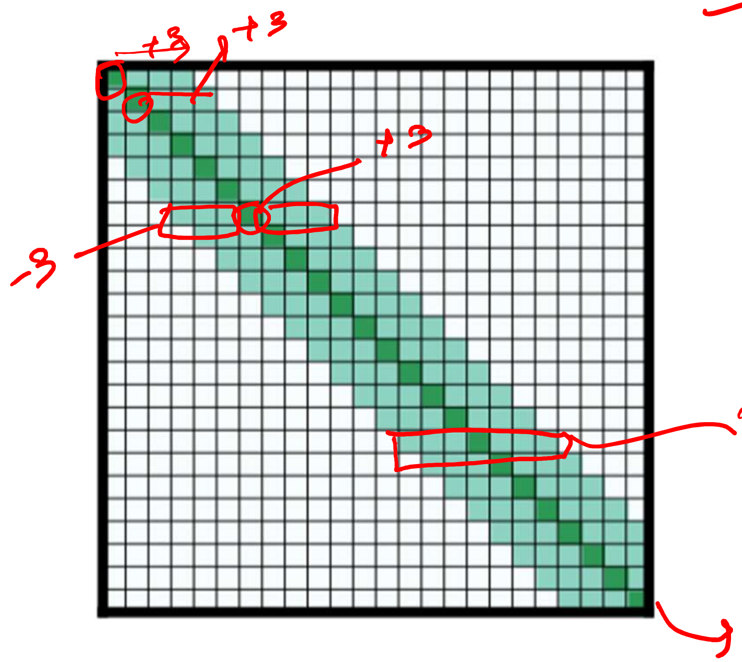
window size = w (4, 5, 6)

↓

context window.



(a) Full n^2 attention



(b) Sliding window attention

w = 7

-3 x 2 + $

X apply for causal LM.

→ get information from the future.

↳ attention.

Year: 2020, Arxiv

Allen Institute for AI

Sliding Window Attention

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left(\frac{\underset{(6,\,4096)}{Q} \times \underset{(4096,\,6)}{K^T}}{\sqrt{4096}}\right) =$$

	THE	CAT	IS	ON	A	CHAIR
THE	1.0	0	0	0	0	0
CAT	0.461	0.538	0	0	0	0
IS	0.3219	0.317	0.361	0	0	0
ON	0	0.316	0.341	0.343	0	0
A	0	0	0.326	0.323	0.351	0
CHAIR	0	0	0	0.313	0.331	0.356

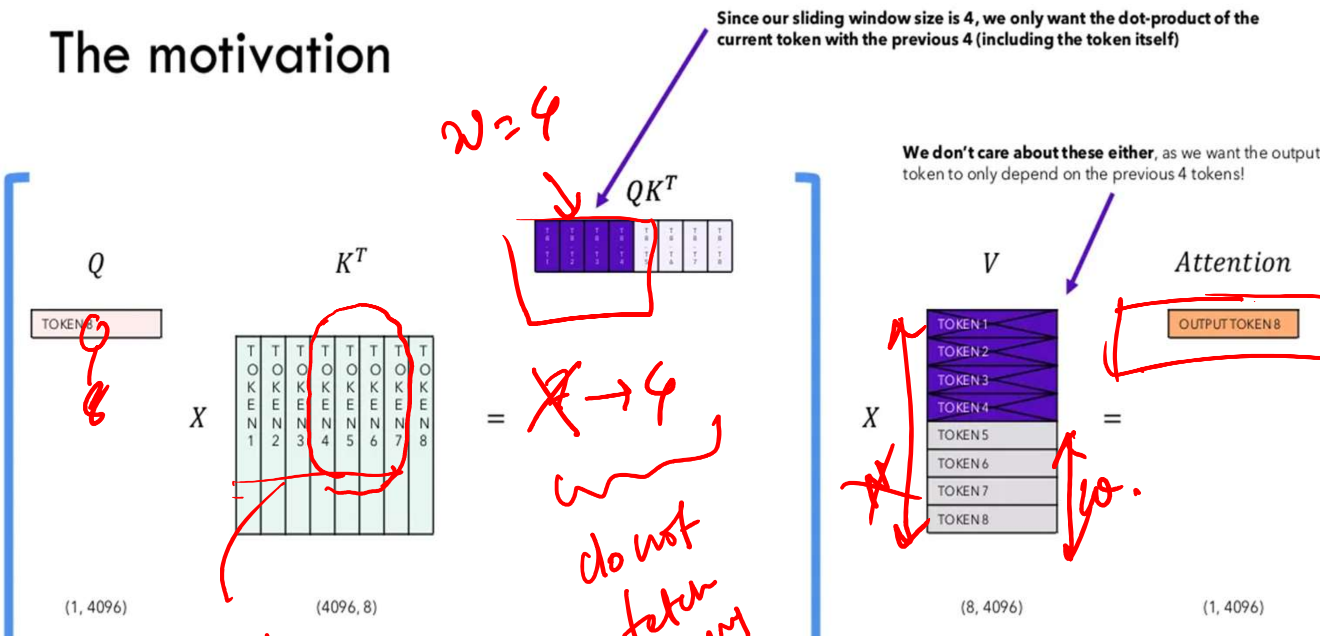
$w = \underline{\underline{3}}.$

Year: 2020, Arxiv

Allen Institute for AI

What happens to the KV Cache?

The motivation



$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Time: O(~~N~~^{2}*d) + O(N^2) + O(N^2*d) = O(N*w*d)

Space: O(3*N*d) + O(N^2) + O(N*d) = O(N*w + N*d)

(N*w)*d

(2×**w**×d)×Size of a float

vs.

(2×N×d)×Size of a float

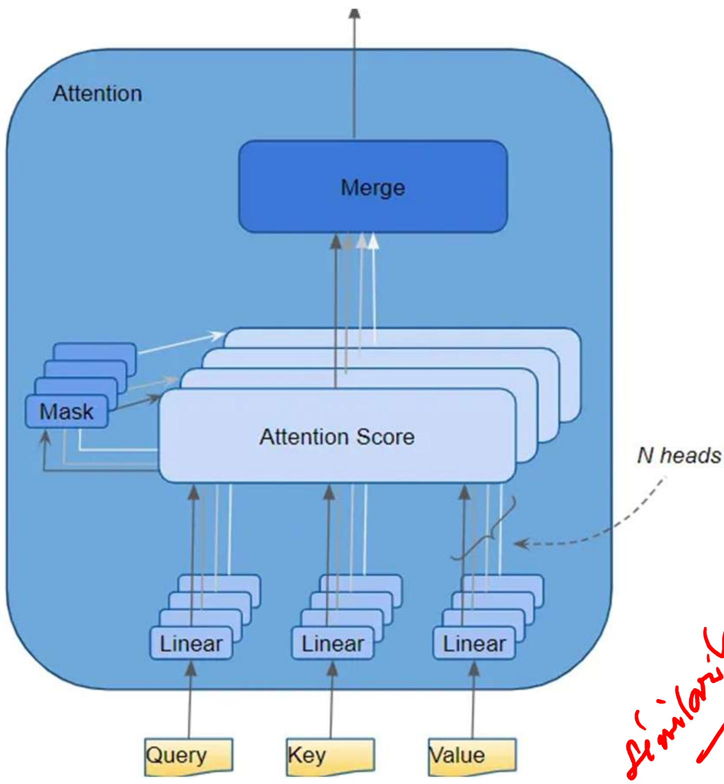
Going back to year: 2017, NeurIPS

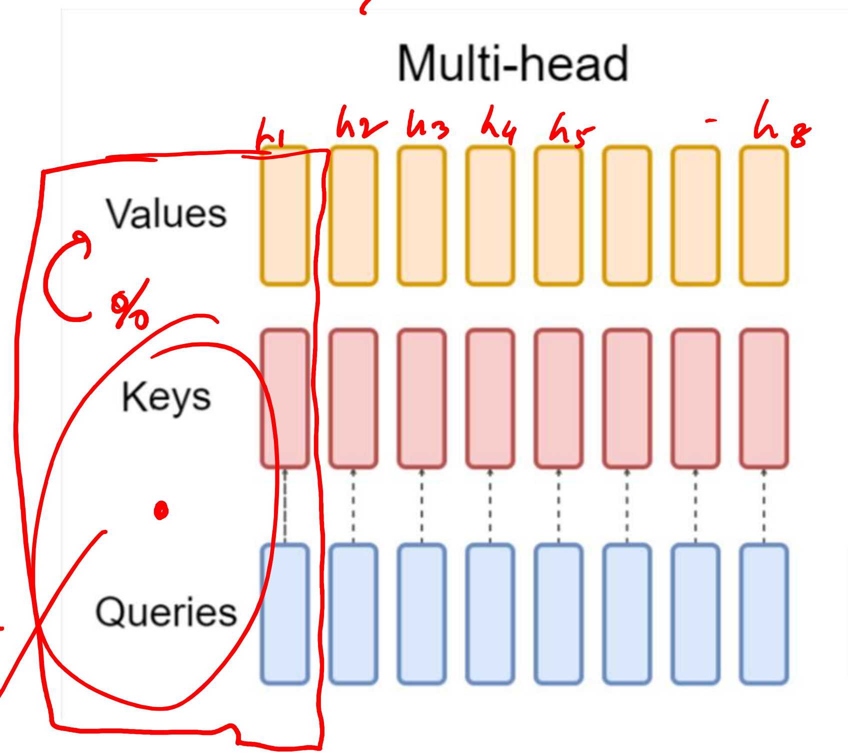
Google Brain Team

Make machines intelligent. Improve people's lives.

Multi-Head Self Attention

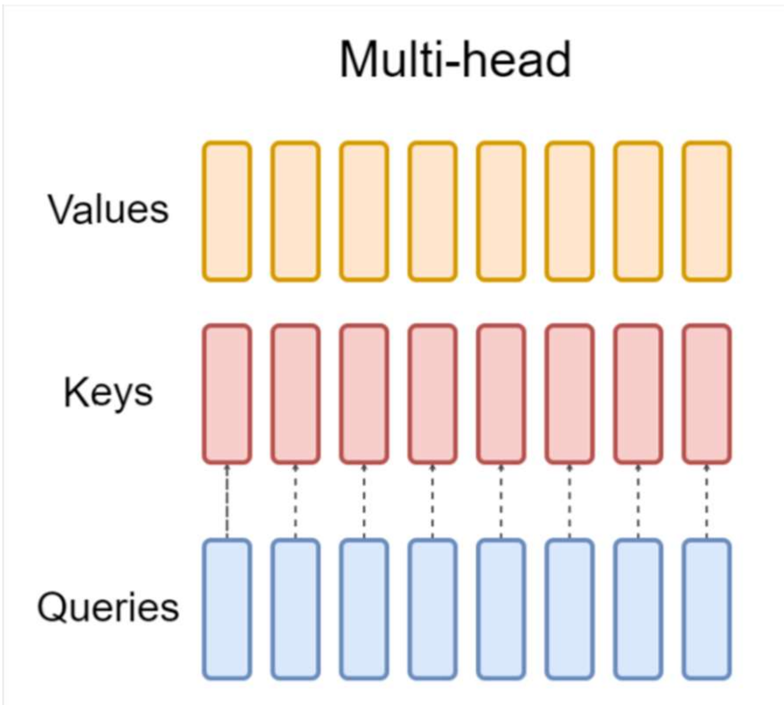
heads: h

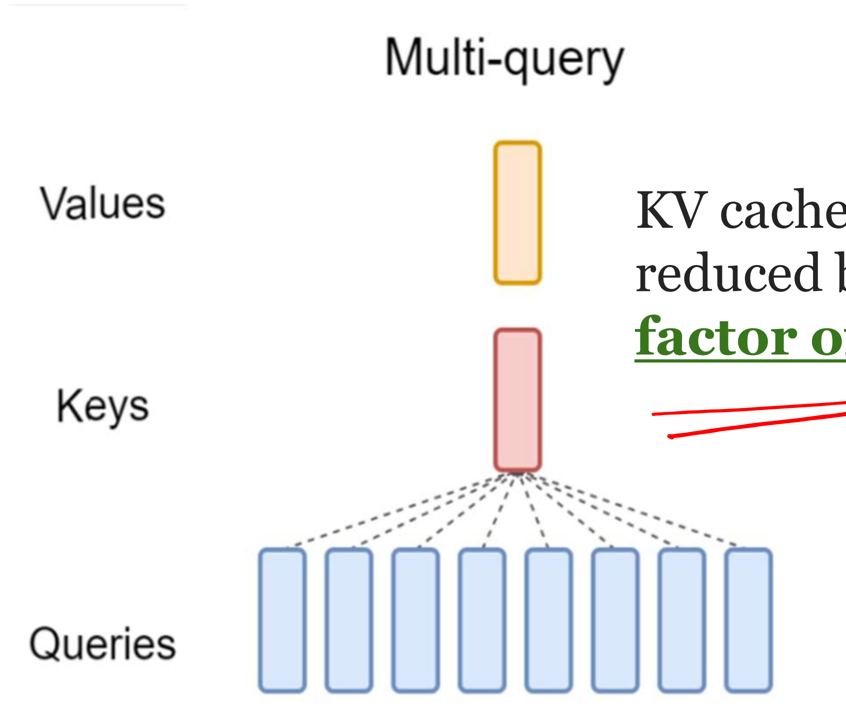




Year: 2019, arxiv Google

Multi-Query Attention (MQA)





KV cache size is reduced by a **factor of *h***

Do we lose out on something?

- Decline in performance quality
- Training instability

Year: 2023; ICLR

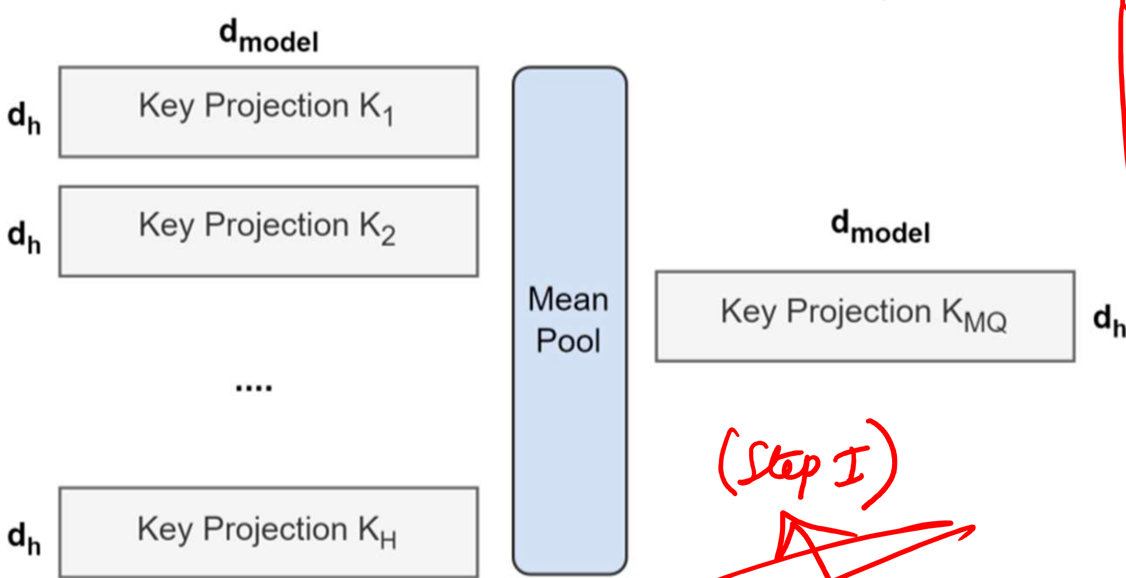
Uptraining: Converting MHA to MQA

getting trained } K epochs.

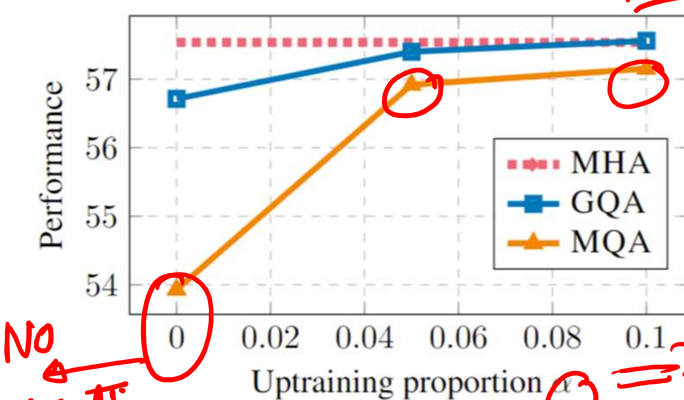
"further"

$\boxed{\text{Step 2}}$. pre-training

$K' \rightarrow K$



(Step I)



NO Step II.

Stop the train @ K' epoch.

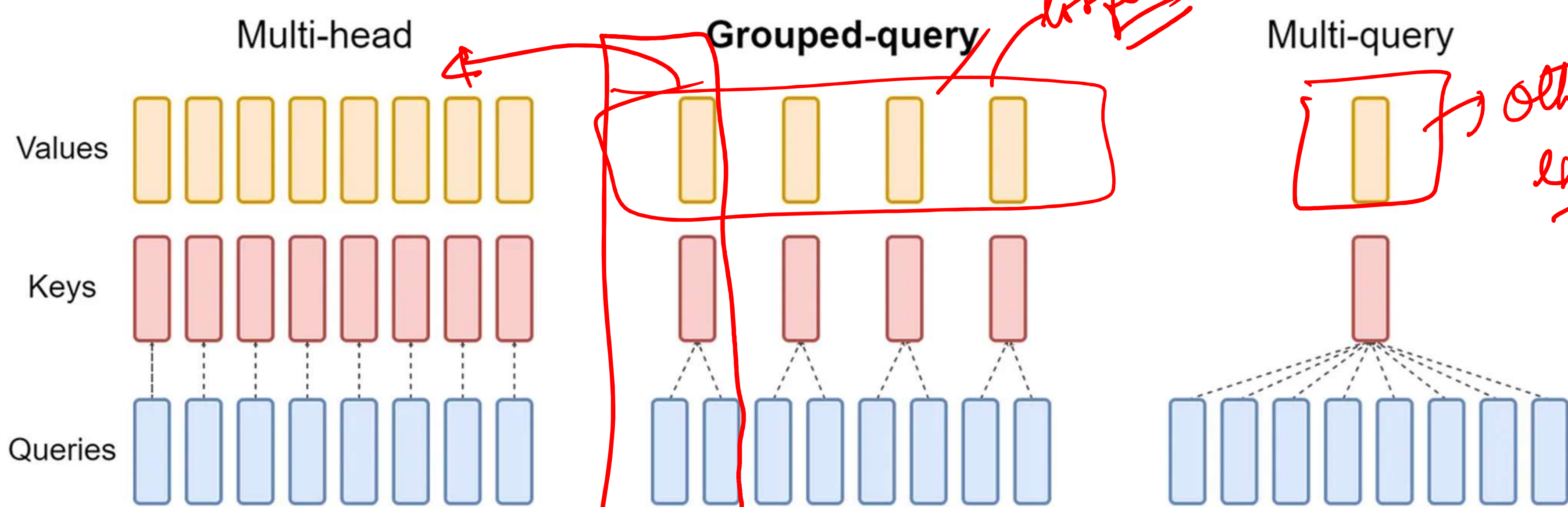
$\alpha = \frac{K-K'}{K}$

What can still go wrong?

- Decline in performance quality
- ~~Training instability~~

Year: 2023; EMNLP

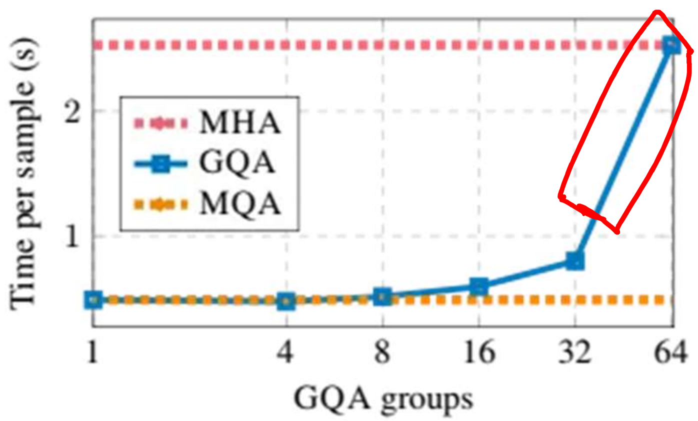
Grouped Query Attention



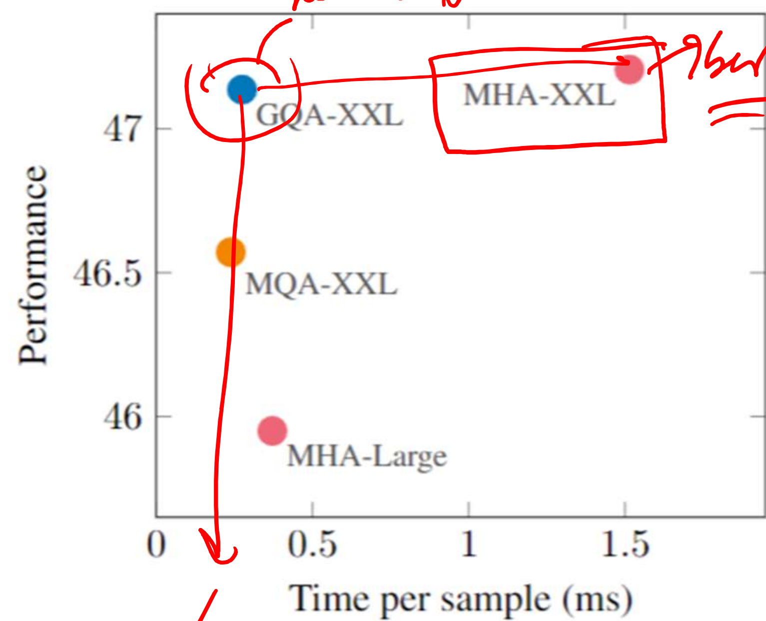
What did we gain?

inference Time

Model	T_{infer}	Average	CNN	arXiv	PubMed	MediaSum	MultiNews	WMT	TriviaQA
	s		R_1	R_1	R_1	R_1	R_1	BLEU	F1
MHA-Large	0.37	46.0	42.9	44.6	46.2	35.5	46.6	27.7	78.2
MHA-XXL	1.51	47.2	43.8	45.6	47.5	36.4	46.9	28.4	81.9
MQA-XXL	0.24	46.6	43.0	45.0	46.9	36.1	46.5	28.5	81.3
GQA-8-XXL	$\boxed{0.28}$	47.1	43.5	45.4	47.7	36.3	47.2	28.4	81.6



near to the best



best.

lot less time.

So are we all set? Key

- GQA/MQA Aim: *To reduce the need for storing a large amount of KV cache*

- LLM server can handle more requests, larger batch sizes and increased throughput

- *Cannot significantly reduce the computational load* BUT.

- *Quality degradation remains*

Large Language Models

Advanced Attention Mechanisms - II

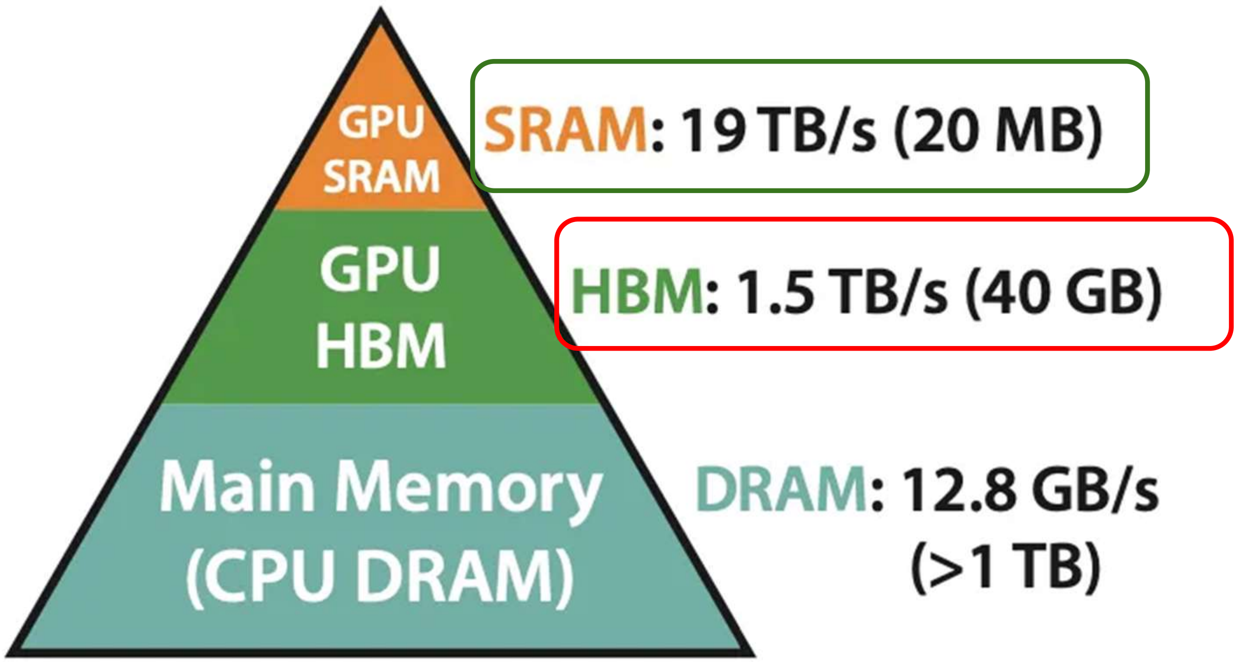
ELL881 · AIL821

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

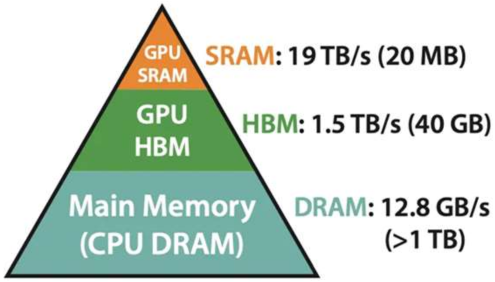
Semester 1, 2024-2025

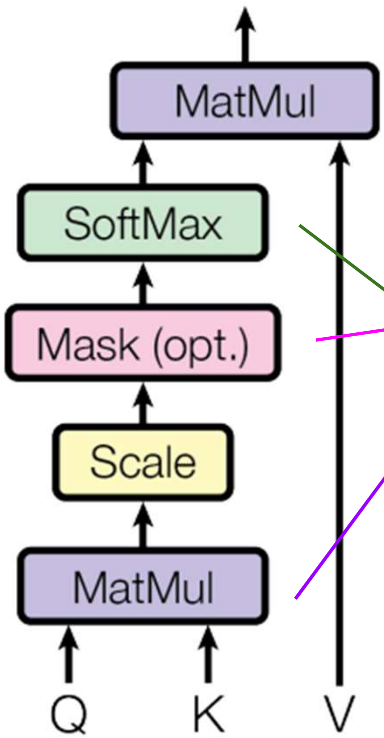
Can we optimize without performance degradation?

A bit more about the GPU



What was happening so far:

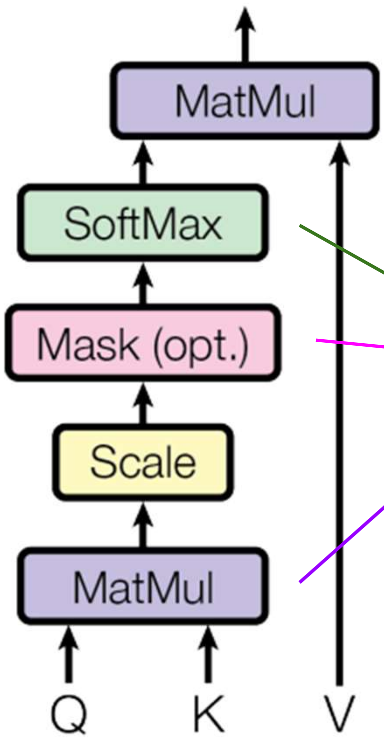




1. Matmul_op (Q,K)
    a. Read Q,K to SRAM (read-op)
    b. Compute matmul A=QxK (compute-op)
    c. Write A to HBM (write-op)
2. Mask_op
    a. Read A to SRAM (read-op)
    b. Mask A into A' (compute-op)
    c. Write A' to HBM (write-op)
3. Softmax_op
    a. Read A' to SRAM (read-op)
    b. Softmax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

Year: 2022; NeurIPS

The magic: Fused Kernel (GPU Operations)!

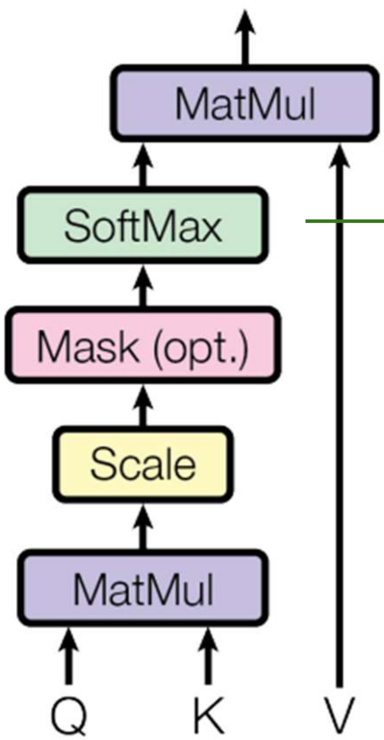


Flash Attention

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

Year: 2022; NeurIPS

The magic does not end here! More optimization

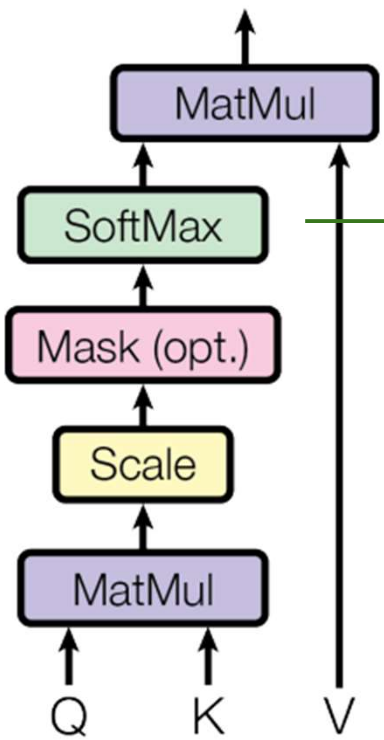


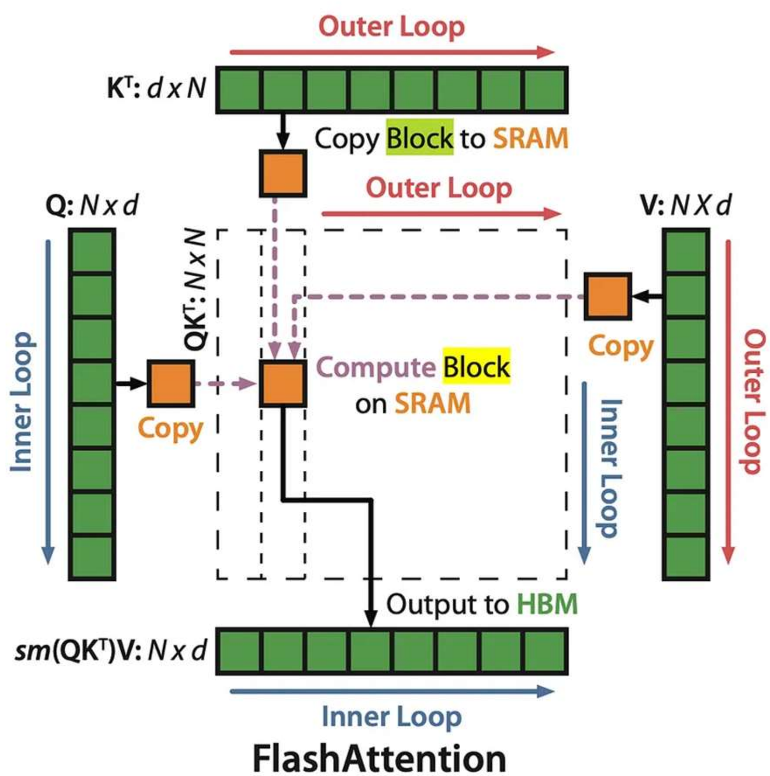
$$s(x_i) = \frac{e^{x_i}}{\boxed{\sum_{j=1}^{N} e^{x_j}}}$$

Gets computed for every row - *Problem!*

Year: 2022; NeurIPS

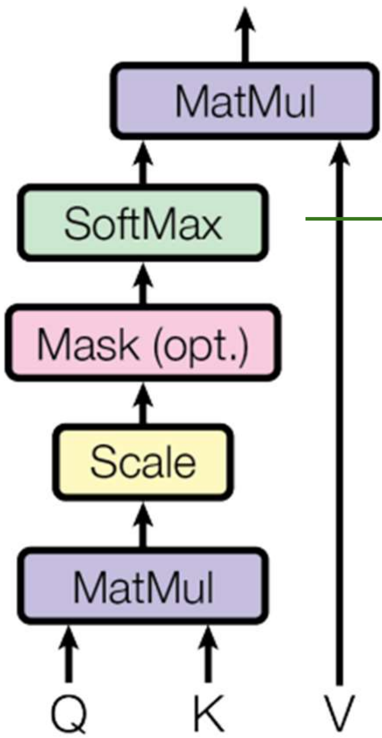
The magic does not end here! *Tiling*



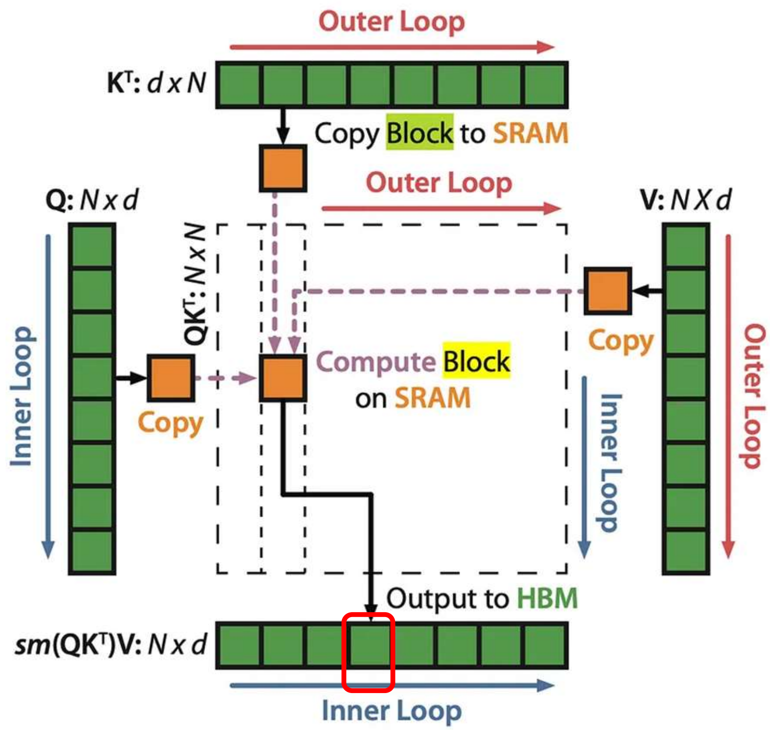


Year: 2022; NeurIPS

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



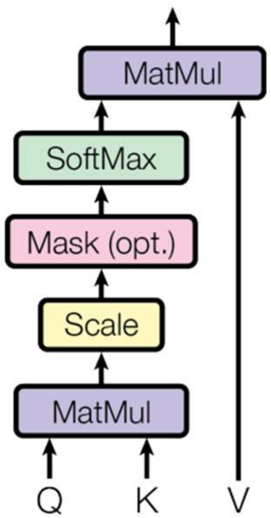
$$s(x_i) = \frac{e^{x_i}}{\boxed{\sum}_{j=1}^{N} e^{x_j}}$$



FlashAttention

Year: 2022; NeurIPS

The softmax denominator problem



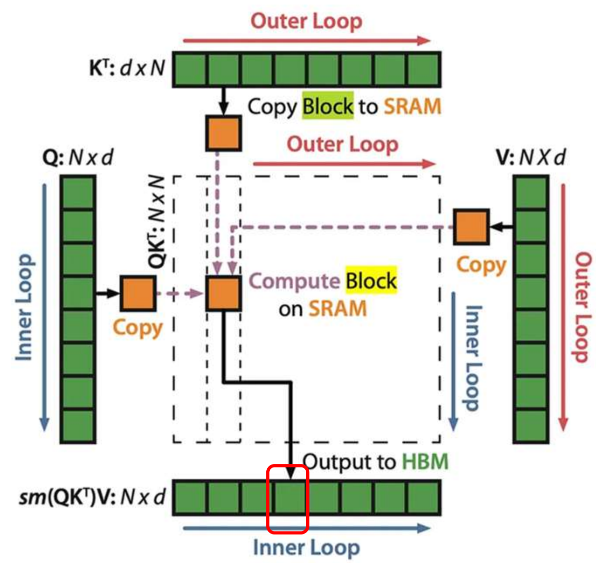
$$Q = [1] \qquad K = [1, 2, 3] \qquad V = [2, 4, 8]$$

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

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

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

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



FlashAttention

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

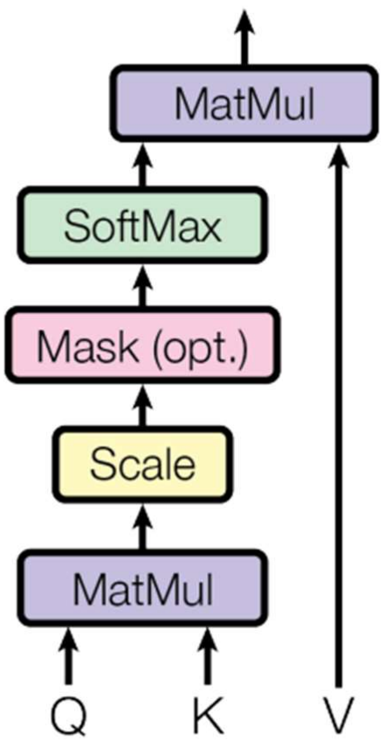
$$D_b = e^1$$

$$N_b = 2e^1$$

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

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 $i = 0$

$D_b = e^1$

$N_b = 2e^1$

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

$D = e^1$

$D * O$

At $i = 1$

$D_b = e^2$

$N_b = 4e^2$

$O = \dfrac{1}{e^1 + e^2}[\boxed{2e^1} + 4e^2]$

$D = e^1 + e^2$

$D * O$

At $i = 2$

$D_b = e^3$

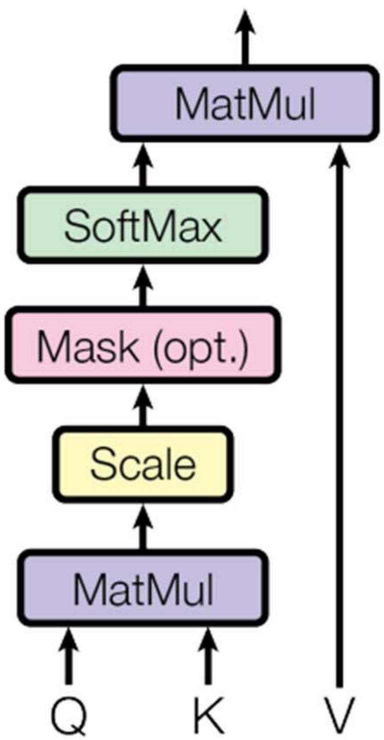
$N_b = 8e^3$

$O = \dfrac{1}{e^1 + e^2 + e^3}[\boxed{2e^1 + 4e^2} + 8e^3]$

$D = e^1 + e^2 + e^3$

Year: 2022; NeurIPS

Summary Statistics - the final touch!



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

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

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

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

$D = 0, O = 0$

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

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

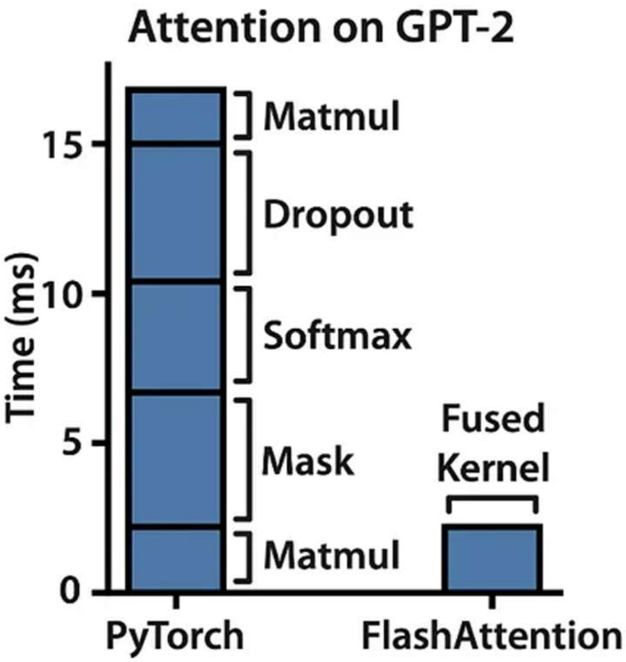
$N_b = V[i] * exp(Q[i] \times K[i])$

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

$D = D + D_b$

Year: 2022; NeurIPS

How well did they do?

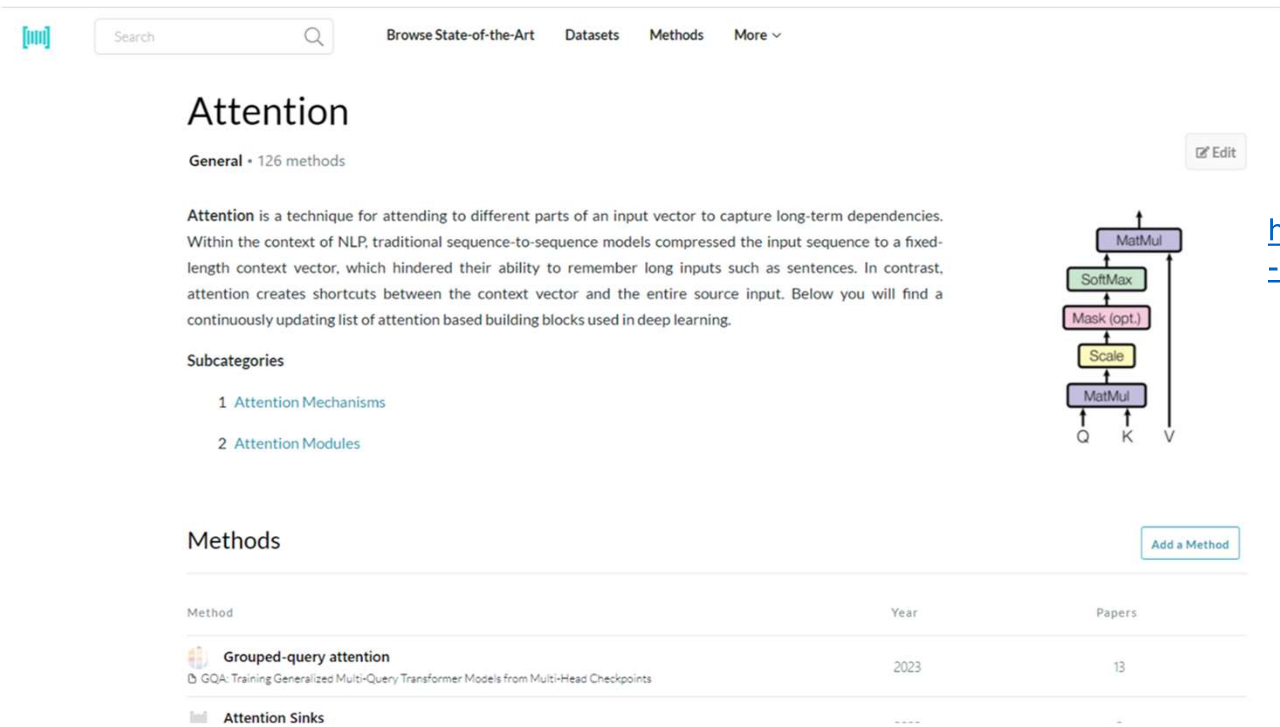


BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FlashAttention (ours)	$\mathbf{17.4} \pm 1.4$

Key Takeaways

- Avoid unnecessary HBM writes
- 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