2023-11-17

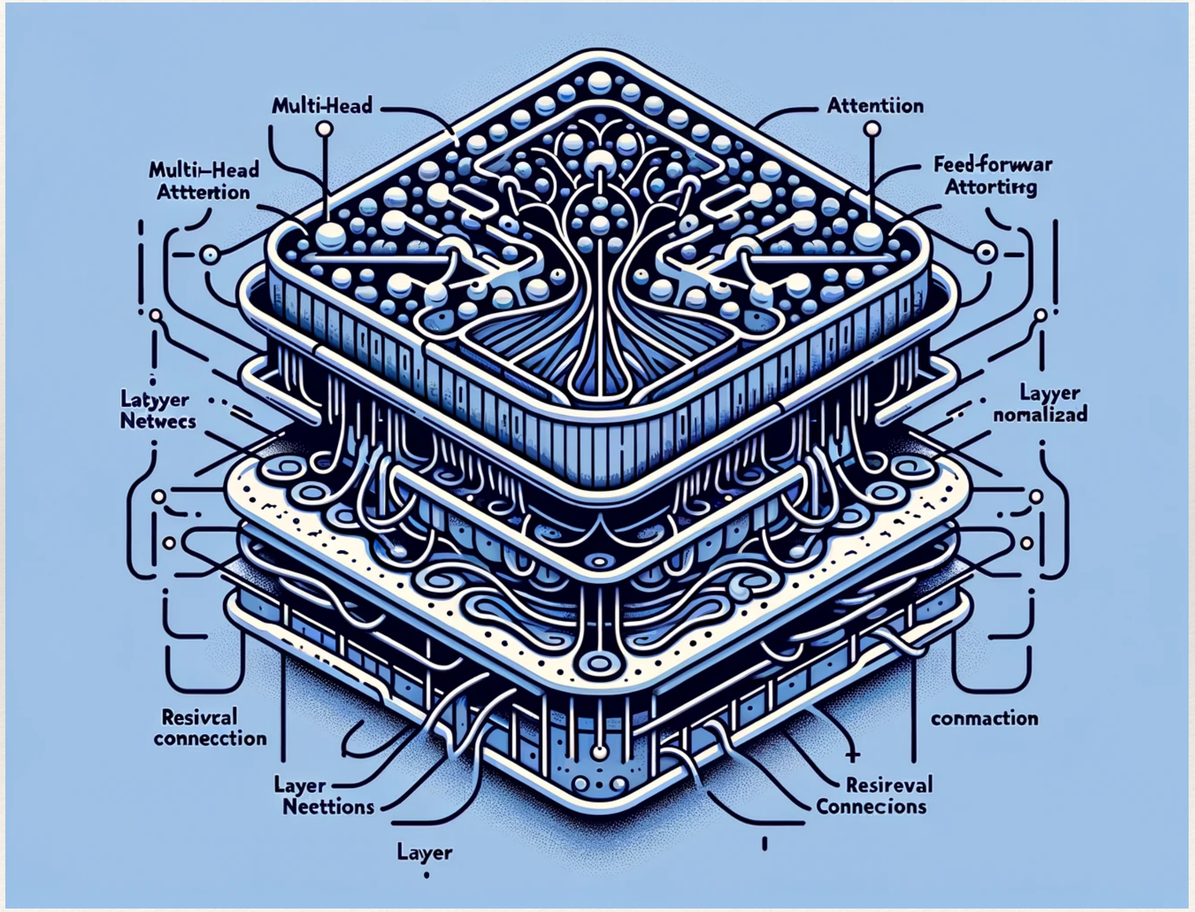
Retentive Networks

Gordon Erlebacher
Department of Scientific Computing

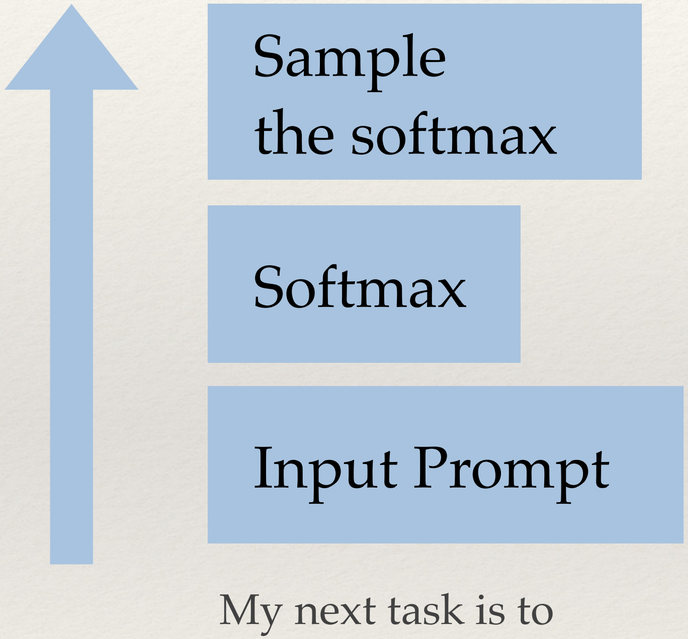
Outline

- Inference with a decoder-only transformer
- Memory and computational cost of the transformer
- Operation reduction
- Retentive network
- Some results

Transformer Review

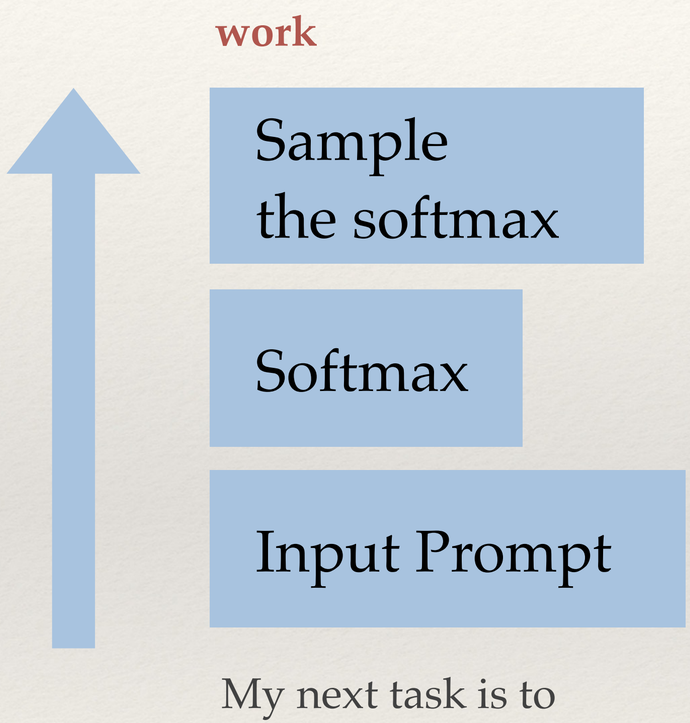


Inference: Top-Down



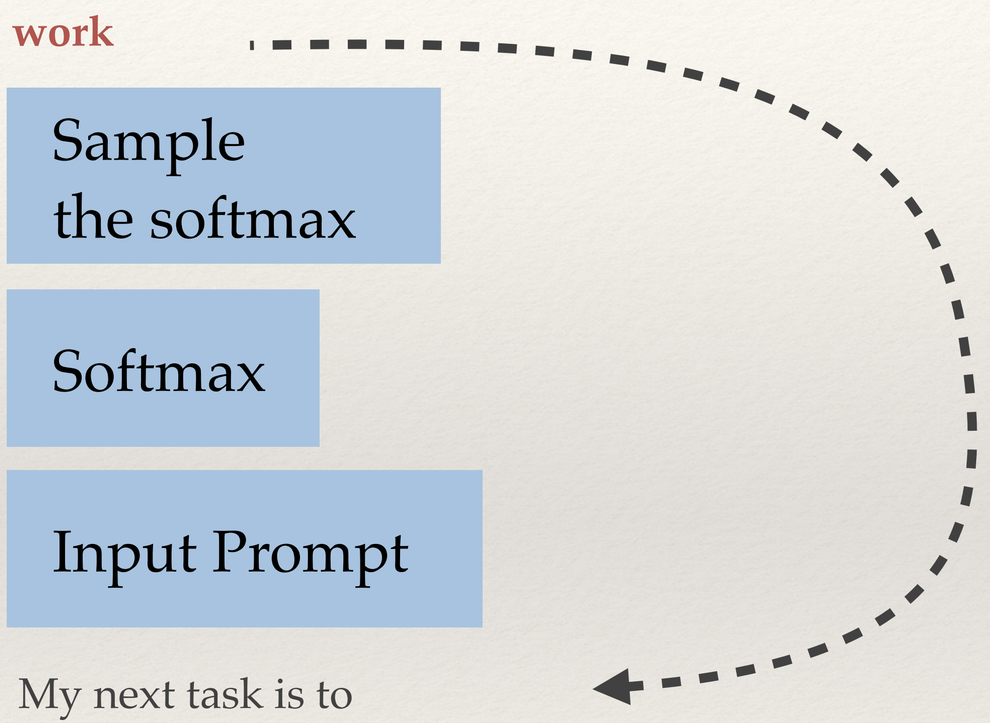
Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down

Sample
the softmax

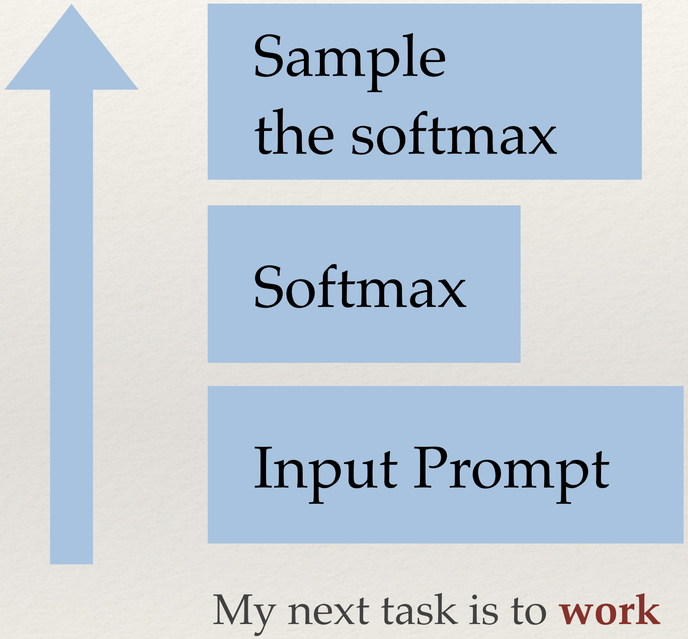
Softmax

Input Prompt

My next task is to **work**

Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down

on

Sample
the softmax

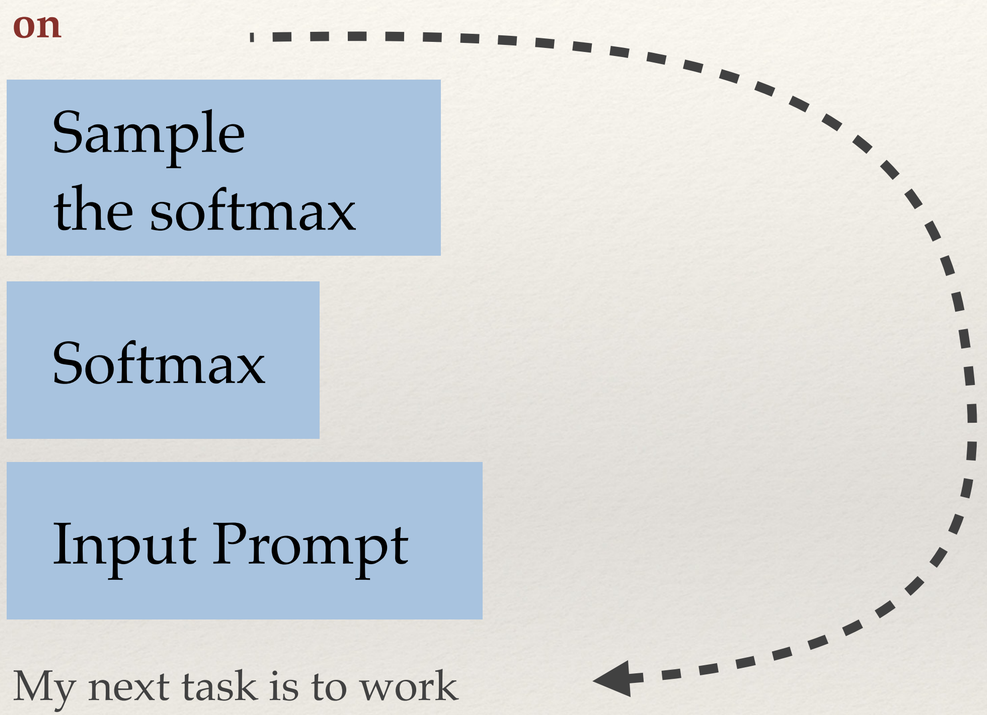
Softmax

Input Prompt

My next task is to work

Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down

Sample
the softmax

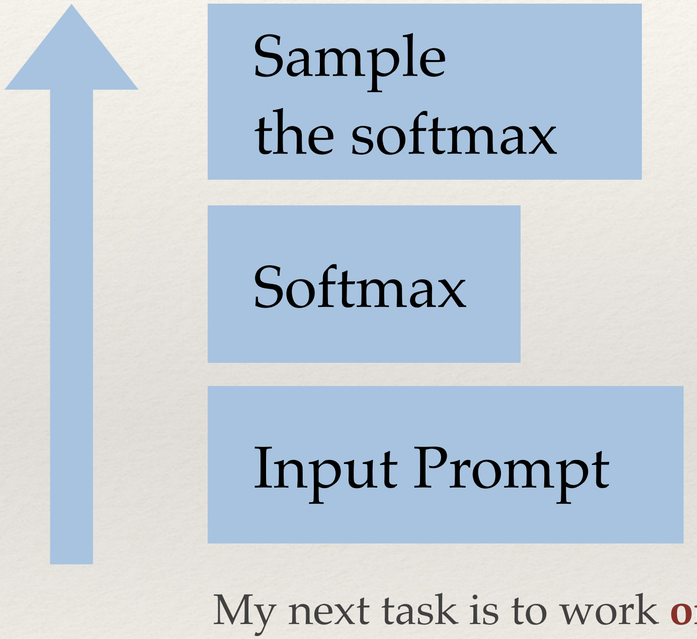
Softmax

Input Prompt

My next task is to work **on**

Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down

my

Sample
the softmax

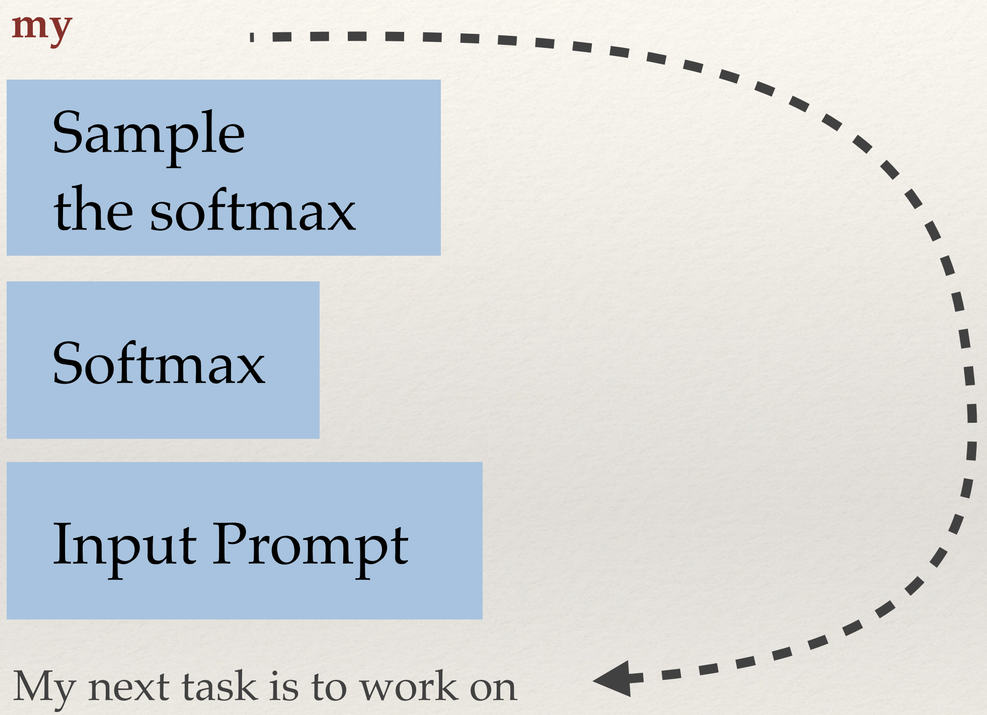
Softmax

Input Prompt

My next task is to work

Transformer Networks
Retentive Networks

Inference: Top-Down



Transformer Networks
Retentive Networks

Inference: Top-Down

Sample
the softmax

Softmax

Input Prompt

My next task is to work on **my**

Transformer Networks
Retentive Networks

Inference: Top-Down

Sample
the softmax

Softmax

Input Prompt

My next task is to work on **my**

$\mathbb{R}^{c \times d}$

Transformer Networks
Retentive Networks

4	3	2	7	4
2	7	2	-1	3

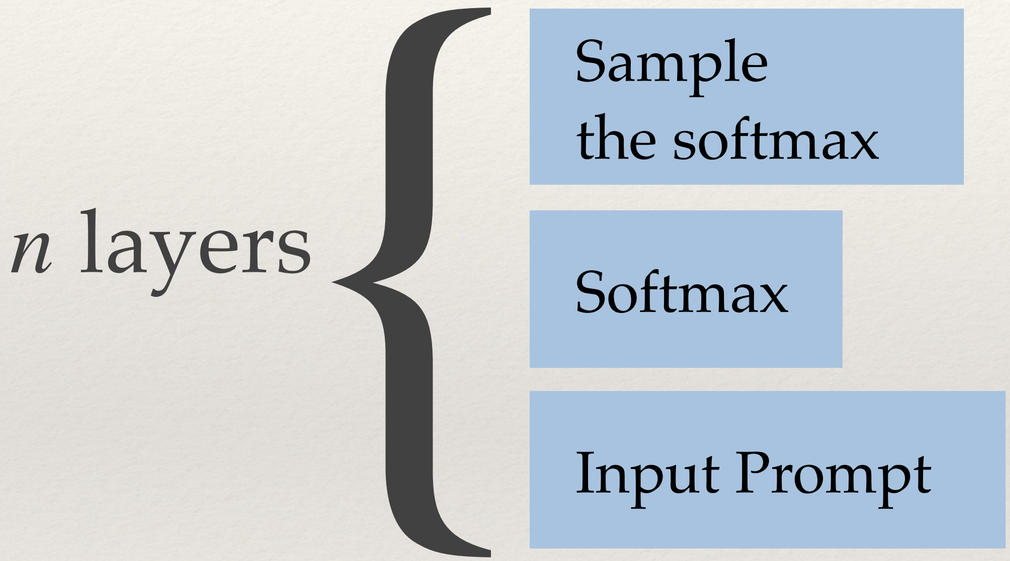
Context length: 2 words

Embedding dimension: 5

c : Context length

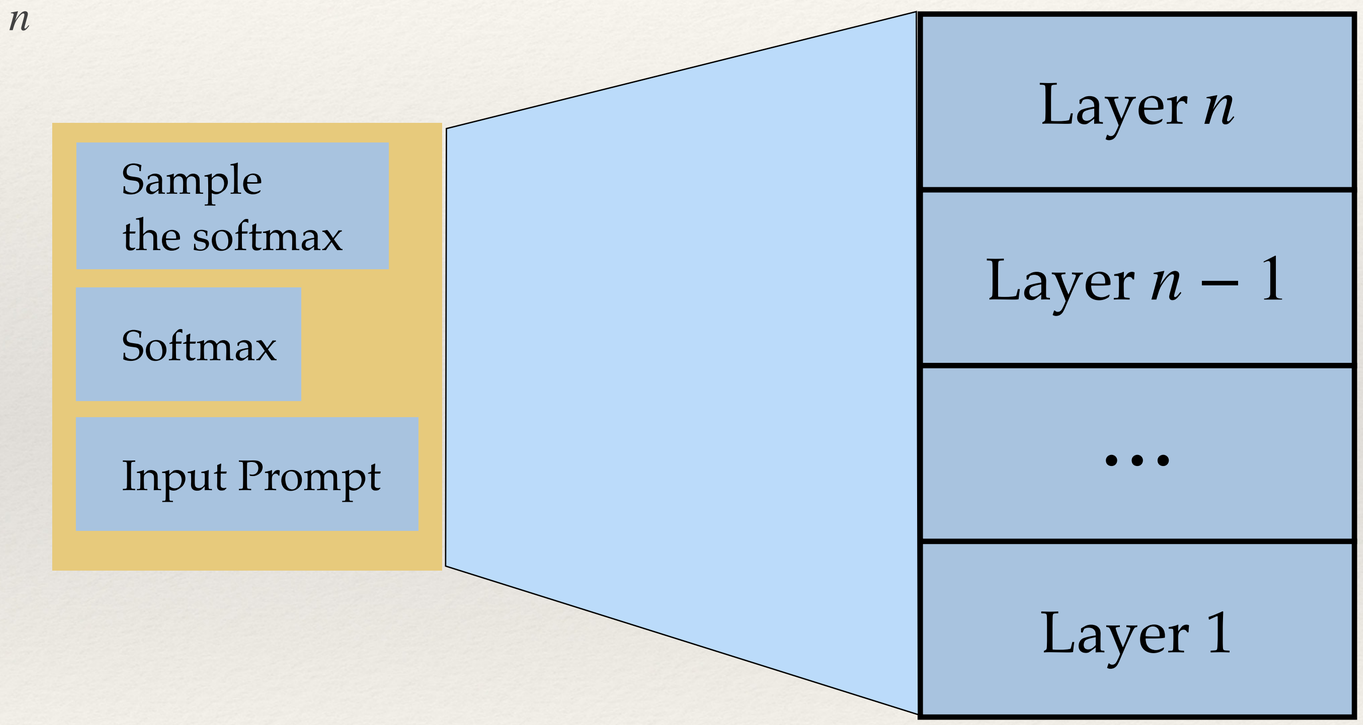
d : Embedding dimension

Inference: Top-Down

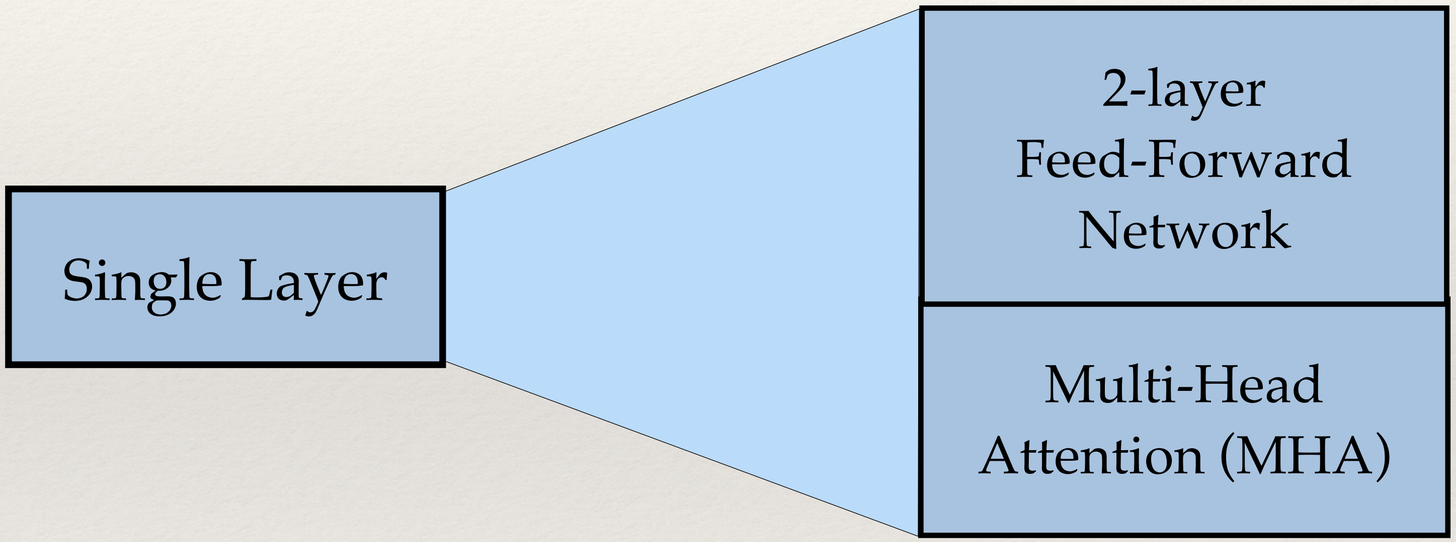


Transformer Networks
Retentive Networks

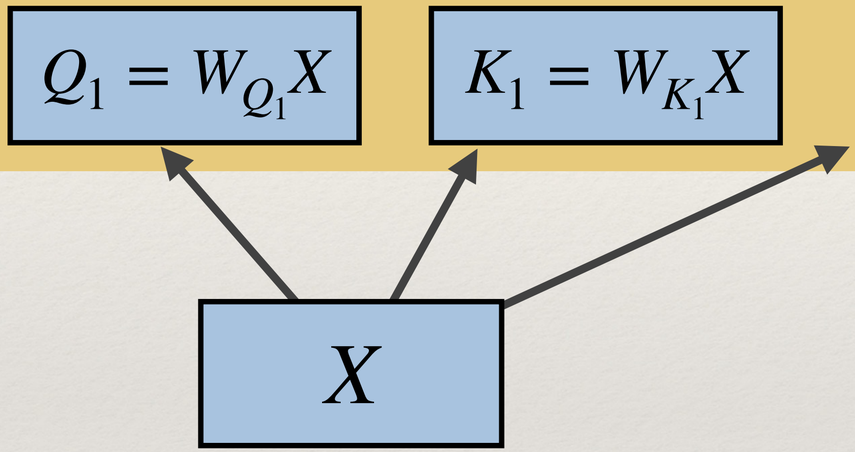
The Network has n layers



Single Transformer Layer



Single-Head Attention



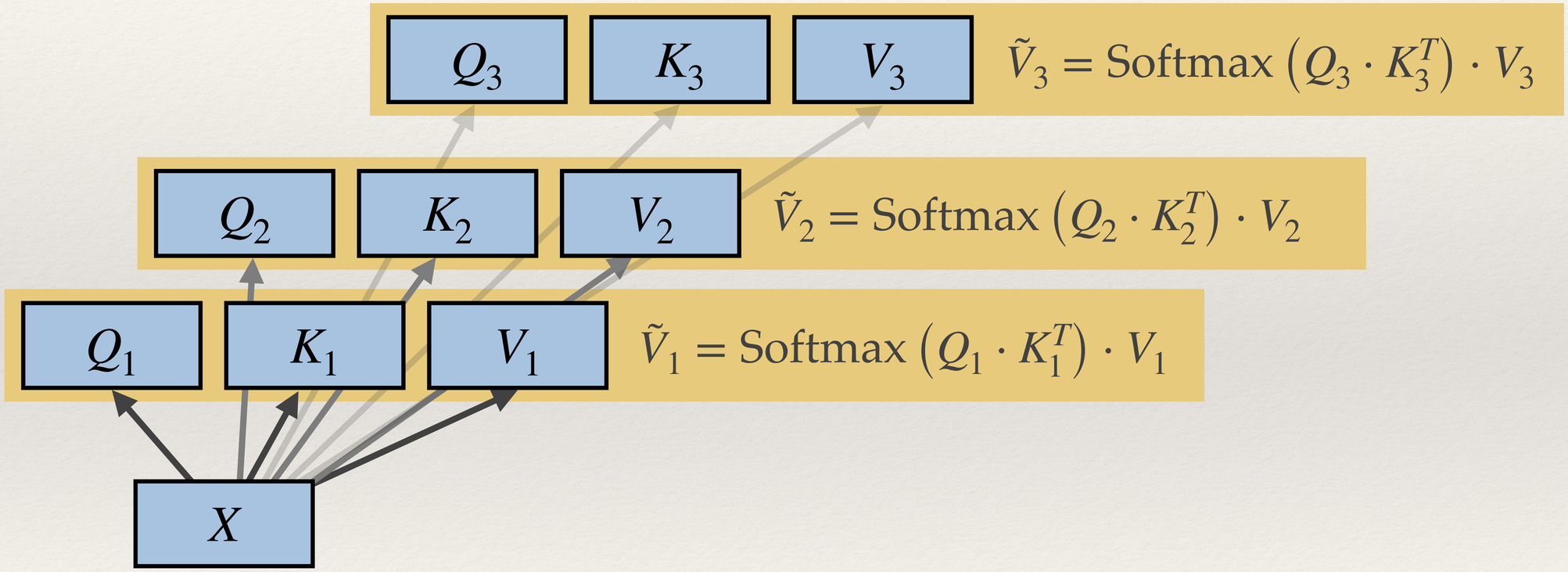
$$\tilde{V}_1 = \text{Softmax}\left(Q_1 \cdot K_1^T\right) \cdot V_1$$

The embedded context: $X \in \mathbb{R}^{c \times d}$

$W_{Q,K,V} \in R^{d \times d}$: Weights that transform from x to attention space

Multi-Head Attention

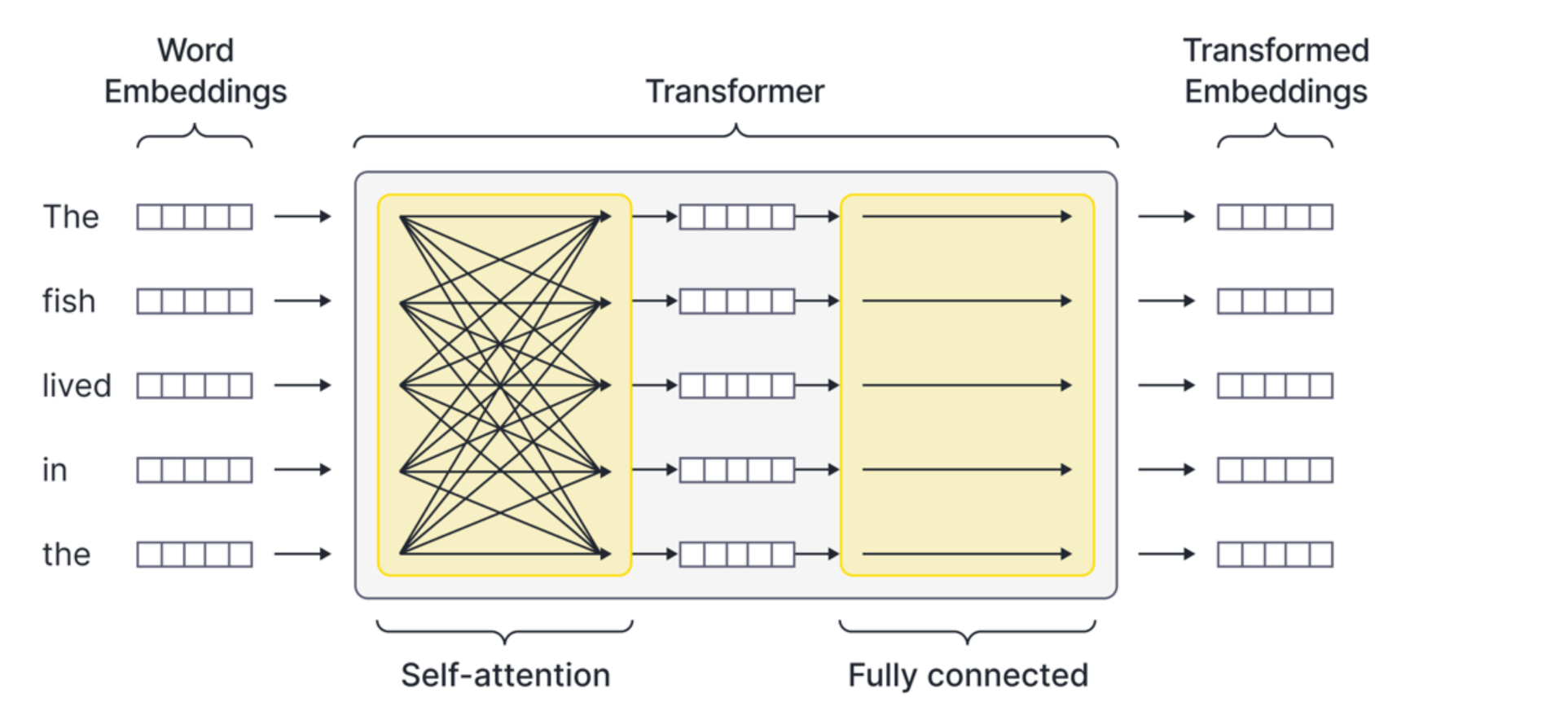
#heads: n_h



Operation Count and Memory Utilization

- Context Length: c
- Embedding dimension: d
- Input to a layer: $X \in \mathbb{R}^{c \times d}$
- Projection into attention space:
  Attention weights: $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$
- Projection out of attention space: $W_O \in \mathbb{R}^{d \times d}$
- For Multi-head attention: $W_{Q_i}, W_{K_i}, W_{V_i} \in \mathbb{R}^{d \times \frac{d}{n_h}}$
  Total memory used is n_h * memory($W_{Q_1}, W_{K_1}, W_{V_1}$)
  Independent of the number of heads

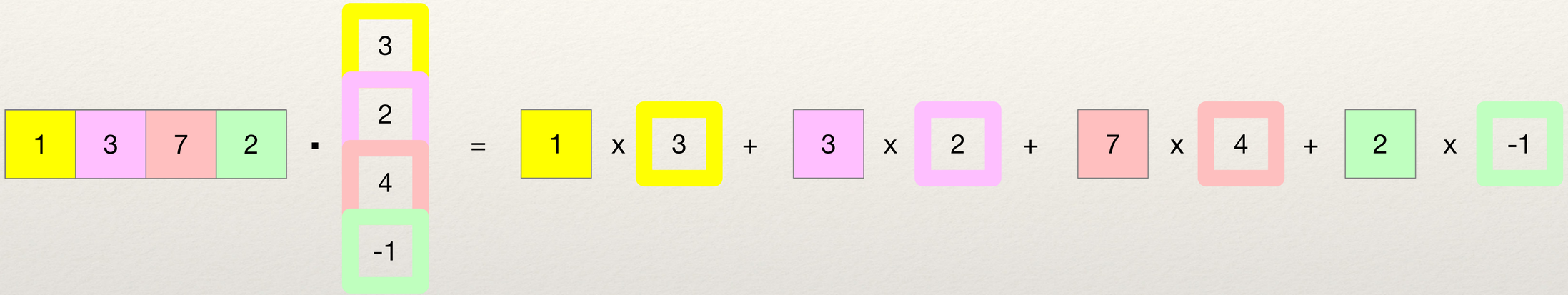
Overall Picture of Single Transformer Layer



Source: https://www.borealisai.com/research-blogs/speeding-up-inference-in-transformers/

Operation Counts

Dot Product



Two vector of size 4: 4 multiplications + (4-1) additions

Two vector of size n: n multiplications + (n-1) additions

$$= 2n - 1 \text{ ops}$$

$$= O(n) \text{ ops}$$

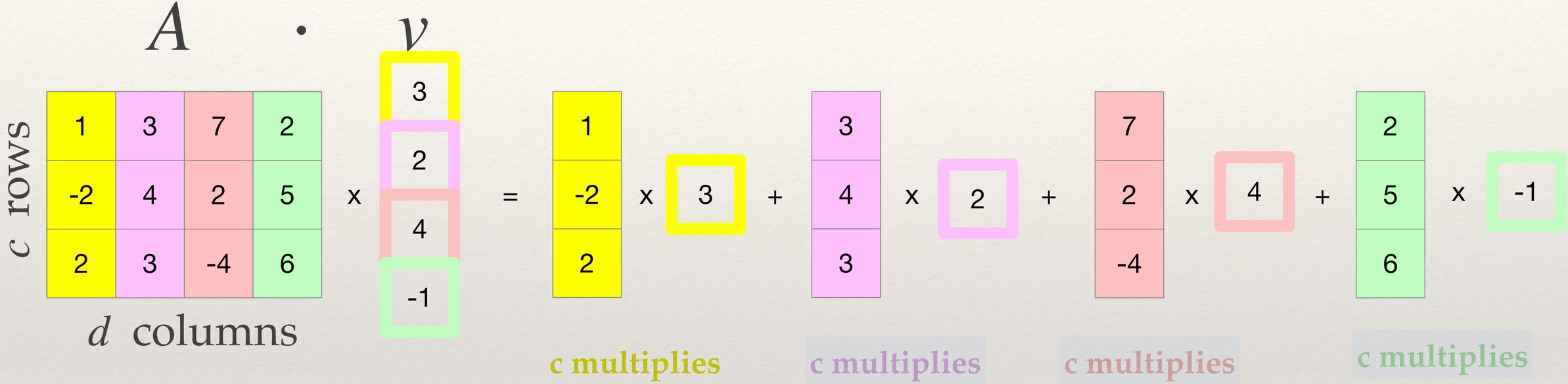
Matrix -Vector Multiplication

- Consider matrix $A \in \mathbb{R}^{c \times d}$ and a vector $V \in \mathbb{R}^{d}$
- Written out:

$$
\begin{aligned}
[A \,.\, v]_i &= A_{i,1} v_2 + A_{i,2} v_2 + \cdots + A_{i,d} v_d \\
&= \sum_{j=1}^{d} A_{i,j} v_j \\
&= A_{i,j} v_j \qquad \text{(Einstein summation convention)}
\end{aligned}
$$

$c(d-1)$ additions, cd multiplications
#Flops: $2cd - c$
Asymptotic complexity: $O(2cd) = O(cd)$ for large c and d

Matrix-Vector Product $A \cdot v$



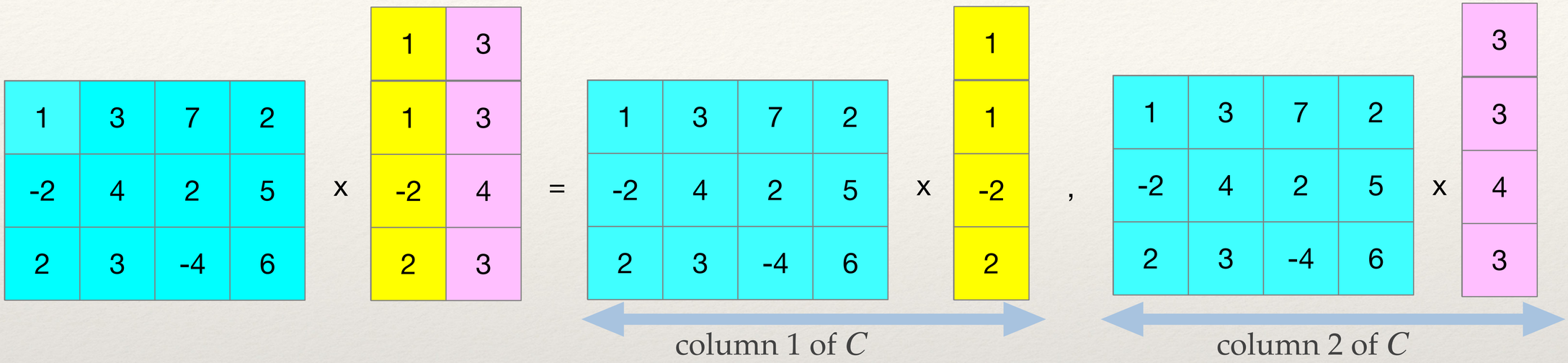
$A \,.\, v \in \mathbb{R}^{c \times d} \mathbb{R}^{d \times 1} = \mathbb{R}^{c \times 1}$

#multiplications: #cols(A) * #rows$(A) = d * c$

#additions: $d - 1$

Total ops: $dc + d - 1 = O(cd)$

Matrix-Matrix Product $C = AB$



- $A \cdot B = \mathbb{R}^{c \times d} \mathbb{R}^{d \times e} = \mathbb{R}^{c \times e}$
- Multiply A by each column of B (e columns)
- Ops: $O(cde)$

Cost of embedding

- Array to store the embeddings: $\mathbb{R}^{|V| \times d}$
- Embedding: $X \longrightarrow X_E \in \mathbb{R}^{c \times d}$
  - memory: $O(cd)$

Transformer: Feed-Forward Network

- $X^{(c \times d)} W^{(d \times d)}$
  - memory: $O(d^2) + O(cd)$
  - flops: $O(cd^2)$
- Feed-Forward Networks
  - Each token is passed through the network independently
  - Feed-forward: $4d$ nodes
  - memory: $O(d^2)$
  - flops: $X^{c \times d} W^{d \times d} = O(cd^2)$
  - memory: $O(d^2)$, one token is passed through FFN at a time

Cost of Single Layer

- **Attention**
  - **Total memory:** $O(c^2 + c^2 + c^2 d) = O(c^2 d)$
  - **Total Flops:** $O(cd^2) + O(c^2) + O\left(c^2 d\right) = O\left(cd(c+d)\right)$
- **Feed Forward Network**
  - **Total memory:** $O(d^2)$
  - **Total Flops:** $O(cd^2)$
- **Full Layer**
  - **Total memory:** $O(c^2 d + d^2) = O(c^2 d)$ if $c \gg d$
  - **Total Flops:** $O(cd(c+d)) = O(c^2 d)$ if $c \gg d$

More Nuance:
Evaluate the cost of one pass through the layer

Standard Transformers
Pass c tokens on each forward pass in one shot.
Store full Softmax in memory $O(c^2)$

Retentive Network
Cut down the cost of attention. Save a factor c on inference in both memory and flops. Inference cost becomes:
Total Memory: $O(cd + d^2) = O(cd)$ if $c \gg d$
Total Flops: $O\left(d(c+d) + d^2\right) = O(cd)$ if $c \gg d$

Cost of a single Full Layer of Transformer

$$\text{Input}: x \in \mathbb{R}^{c \times d}$$

$$\text{Output}: y \in \mathbb{R}^{c \times d}$$

$$\text{Weight}: W \in \mathbb{R}^{c \times c}$$

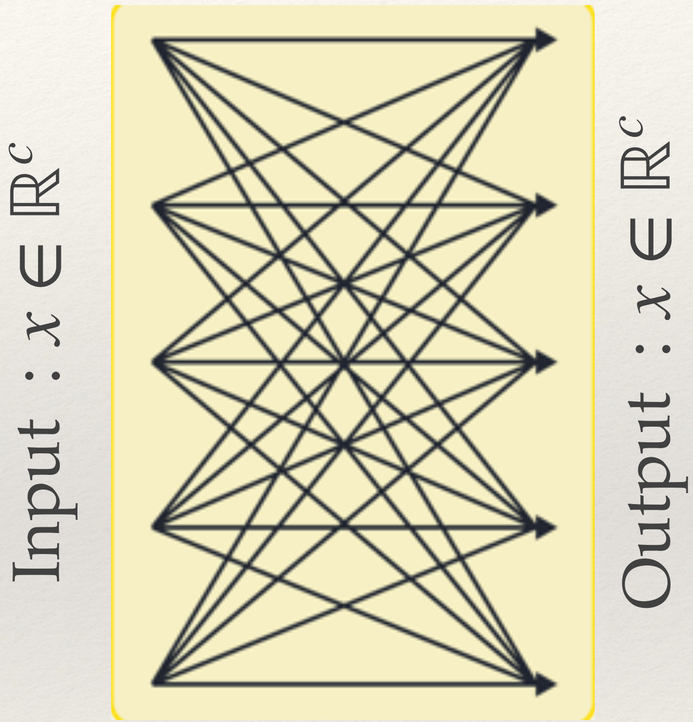
$$\text{Activation function}: y = f(x), \quad \mathbb{R}^c \longrightarrow \mathbb{R}^c$$

$$y = f(x \cdot W + b)$$

Dominant cost: $x \cdot W$

Ops: $O(c^2 d)$

Memory: $O(c^2)$



Backward Pass

- Several works have demonstrated the the cost of the backward pass is between 1x and 3x the cost of the forward pass.
- 2x is the accepted average value across architectures and hardware

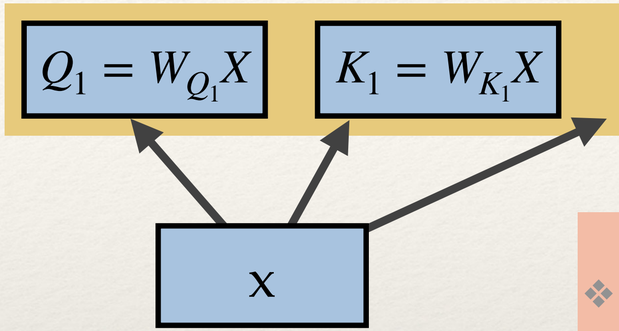
Cost of Attention (single head)

- Compute: $A = \frac{1}{\sqrt{d}}\text{Softmax}(Q \cdot K^T) \cdot V$
- Compute $Q^{c\times d} \underbrace{\cdot}_{\Sigma_d} (K^T)^{d\times c}$
  - Memory: $O(c^2)$
  - Flops: $O(c^2 d)$
- Softmax: $\left[S(x)\right]_i = \frac{\exp x_i}{\sum_{j=1}^{c} \exp x_j}$ for each query
  - Memory: $O(c)$, for all queries: $O(c^2)$
  - Flops: $O(c)$, for all queries: $O(c^2)$
- $A = S^{c\times c} \underbrace{\cdot}_{\Sigma_c} V^{c\times d}$
  - Memory: $O(c^2 d)$
  - Flops: $O(c^2 d)$

Total memory: $O(c^2 + c^2 + c^2 d) = O(c^2 d)$
Total Flops: $O(c^2 d) + O(c^2) + O\left(c^2 d\right) = O(c^2 d)$

Cost of Attention, Single Head

- $X \to XW_Q$
  - memory: $O(d^2)$ for W_Q
  - ops: $O(cd^2)$



$$\tilde{V}_1 = \text{Softmax}\left(Q_1 \cdot K_1^T\right) \cdot V_1$$

- $Q_1 \cdot K_1^T$

  Alternatively:

  $$(Q_1 K_1^T)_{i,j} = \sum_{k=1}^{d} (Q_1)_{i,k} (K_1^T)_{k,j}$$

- $Q_1 \cdot K_1^T$
- Alternatively:

  $$(Q_1 K_1^T)_{i,j} = \sum_{k=1}^{d} (Q_1)_{i,k} (K_1^T)_{k,j}$$

  - memory: $O(c^2)$
  - ops: $O(cd^2)$

Cost of Softmax

- Given a vector of size c, generating the cost max costs $O(c)$
- However, $QK^T \in \mathbb{R}^{c \times c}$. To compute c softmaxes costs $O(c^2)$
- Memory cost of the softmax: $O(c^2)$

Finally: Softmax $(Q \cdot K^T) \cdot V$

- $Q \cdot K^T \in \mathbb{R}^{c \times c}$
- $V \in \mathbb{R}^c$
- The dot product is over the context length c
- Cost to compute the dot product: $O(c^2)$

Attention: Ops and Memory costs

Memory

- QK^T : $O(c^2)$: Dominates
- Softmax: $O(1)$
- **Total:** $O(c^2)$

Ops

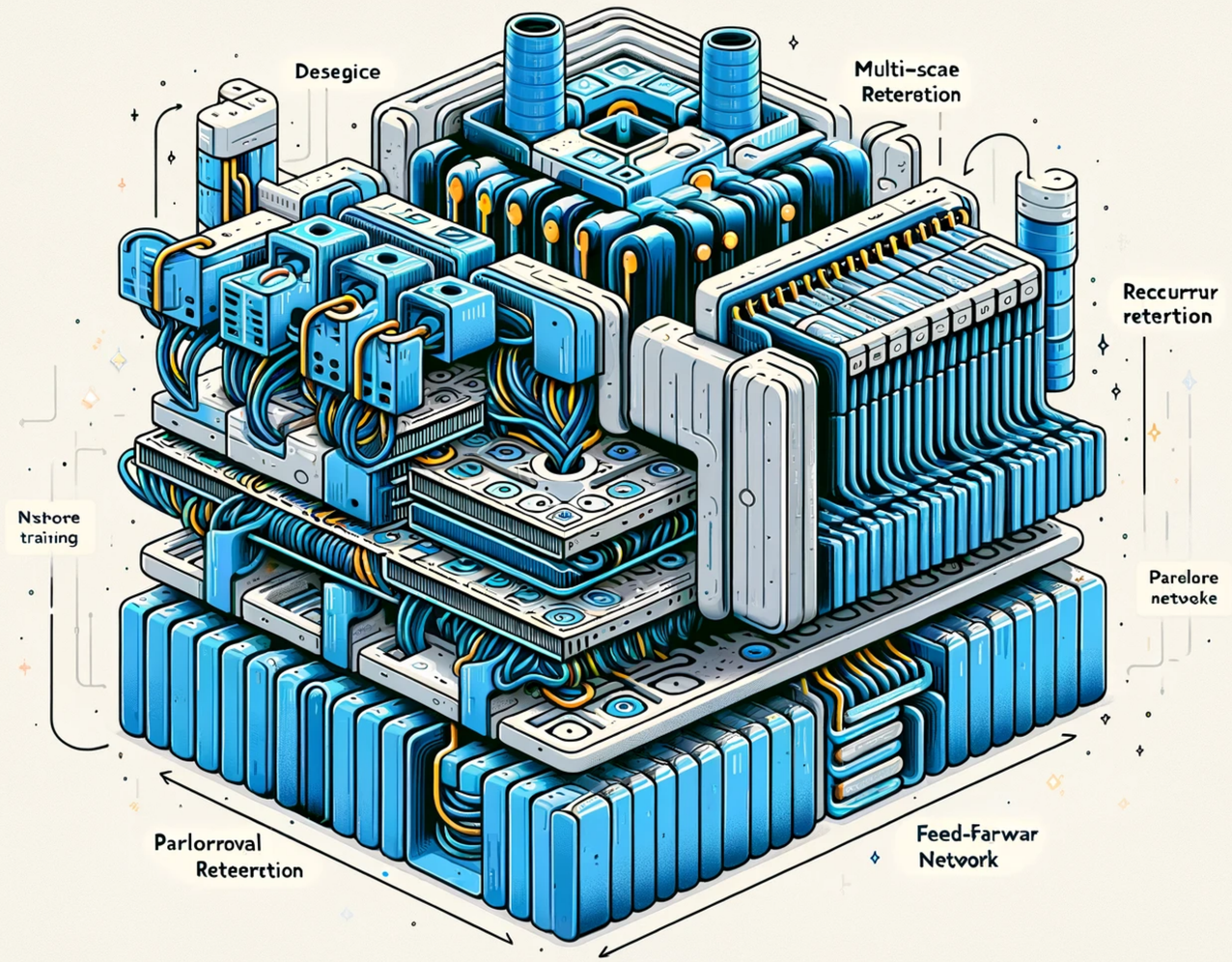
- QK^T : $O(c^2 d)$
- Softmax computation: $O(c^2)$
- $V^* = \text{Softmax} \cdot V$: $O(c^2 d)$
- $V \cdot W^O$: $O(cd^2)$
- **Total:** $O(c^2)$

Transformer Costs

c: context length

	Training	Inference Ops (per token)	Memory
Transformers	$O(c)$	$O(c)$	$O(c^2)$

Retentive Network



Parallel to Recursion

- Goals for per-token inference
  - Memory: $O(c^2) \longrightarrow O(c)$
  - Ops: $O(c) \longrightarrow O(1)$

- Approach
  - Reformulate parallel attention into a recursive algorithm

From Transformers to Retention

- Step 1: remove the softmax
- Step 2: autoregressive masking matrix D

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ \dots & \dots & \dots & \dots \end{pmatrix} \in \mathbb{R}^{c \times c}$$

- Revised attention:

$$V^* = \left(Q \cdot (K^T \odot D)\right) \cdot V \ \in \mathbb{R}^{c \times d}$$

- Rewrite the formula query by query

Vector vs Index Notation

- Dot Product : $V \cdot X = \sum_i v_i x_i \qquad x_i \in \mathbb{R}$
- Vector-Matrix : $X \cdot W = \sum_i x_i w_{i,j} \qquad x_i , w_{i,j} \in \mathbb{R}$
- Matrix Matrix : $A \cdot B = \sum_k a_{i,k} b_{k,j} \qquad a_{i,k} , b_{k,j} \in \mathbb{R}$

Breakdown of $V^* = \left(Q \cdot (K^T \odot D)\right) \cdot V$

- $(Q \cdot K^T)_{i_c, i'_c} = \sum_{i_d} Q_{i_c, i_d} K_{i_d, i'_c}$ Scalar product
- $(Q \cdot K_T \odot D)_{i_c, i'_c} = (Q \cdot K_T)_{i_c, i'_c} D_{i_c, i'_c}$ Hadamard product
- $\left[\left(Q \cdot (K^T \odot D)\right) \cdot V\right]_{i_c, i_d} = \sum_{i'_c} \left(Q \cdot (K^T \odot D)\right)_{i_c, i'_c} V_{i'_c, i_d}$ Scalar product

$$\left[\left(Q \cdot (K^T \odot D)\right) \cdot V\right]_{i_c, i_d} = \sum_{i'_c} \sum_{i'_d} Q_{i_c, i'_d} (K^T_{i'_d, i'_c} D_{i_c, i'_c})_{i_c, i'_c} V_{i'_c, i_d}$$

First Query (i.e., token)

$$V^* = \left(Q \cdot (K^T \odot D)\right) \cdot V \in \mathbb{R}^{c \times d}$$

i_c, i'_c: Context indices
i_d, i'_d: Embedding indices

- First row of V^* :

$$V^*_{1,d} = \left(\sum_{i'_c} \left[\sum_{i_d} Q_{1,i_d} K_{i_d,i'_c} \right] D_{1,i'_c} V_{i'_c,d} \right)$$

- Note that $D_{1,i'_c} = 0 \quad \text{for } c' > 1$
- Therefore,

$$V^*_{1,d} = \left[\sum_{i_d} Q_{1,i_d} K_{i_d,1} \right] D_{1,1} V_{1,d}$$

$$V^*_1 = (Q_1 \cdot K_1^T)\, V_1$$

Second Query (i.e., token)

$$V^* = \left(Q \cdot (K^T \odot D)\right) \cdot V \in \mathbb{R}^{c \times d}$$

i_c, i'_c: Context index
i_d, i'_d: Embedding index

- $D_{2,i'_c} = 0 \quad i'_c > 2$
- Second row of V^*: V^*_2
- Note that the first term of V^*_2 is already calculated!

$$V^*_{2,i_d} = \left(\sum_{i'_c} \left[\sum_{i'_d} Q_{2,i'_d} K_{i'_d,i'_c} \right] D_{2,i'_c} V_{i'_c,i_d} \right)$$

$$V^*_{2,i_d} = \left[\sum_{i'_d} Q_{2,i'_d} K_{i'_d,1} \right] D_{2,1} V_{1,i_d} + \left[\sum_{i'_d} Q_{2,i'_d} K_{i'_d,2} \right] D_{2,2} V_{2,i_d}$$

$$V^*_{2,i_d} = \left[\sum_{i'_d} Q_{2,i'_d} K_{i'_d,1} \right] V_{1,d} + \left[\sum_{i'_d} Q_{2,i'_d} K_{i'_d,2} \right] V_{2,d}$$

$$V^*_2 = Q_2 \cdot K_1^T V_1 + Q_2 \cdot K_2^T V_2 = Q_2 \cdot (K_1^T V_1 + Q_2 \cdot K_2^T V_2)$$

Generalization

$$V^* = \left(Q \cdot (K^T \odot D)\right) \cdot V \in \mathbb{R}^{c \times d}$$

i_c, i'_c: Context index

i_d, i'_d: Embedding index

$$
\begin{aligned}
V_1^* &= Q_1 \cdot K_1^T V_1 \\
&= Q_1 \cdot S_1 \\
V_2^* &= Q_2 \cdot (K_1^T V_1 + K_2^T V_2) \\
&= Q_2 \cdot (S_1 + K_2^T V_2) \\
&= Q_2 \cdot S_2 \\
\cdots &= \cdots \\
V_n^* &= Q_n \cdot (K_{n-1}^T V_{n-1} + K_n^T V_n) \\
&= Q_n \cdot (S_{n-1} + K_n^T V_n) \\
&= Q_n \cdot S_n
\end{aligned}
$$

$$
\begin{aligned}
S_1 &= K_1^T V_1 \\
V_1^* &= Q_1 \cdot S_1 \\
S_2 &= S_1 + K_2^T V_2 \\
V_2^* &= Q_2 \cdot S_2 \\
\cdots &= \cdots \\
S^c &= S_{c-1} + K_c^T V_c \\
V_c^* &= Q_c \cdot S_c
\end{aligned}
$$

Memory per token : S_i : $O(d^2)$

Ops per token: $V^* i$: $O(c)$

Recursive Form

$S_0 = 0$

for $i \in [1,2,\ldots,c]$:

$S_i = S_{i-1} + K_i^T V_i$

$V_i^* = Q_i \cdot S_i$

$Q_i, K_i, V_i \in \mathbb{R}^d$

$S_i, K_i^T V_i \in \mathbb{R}^{d \times d}$

Transformer $\longrightarrow$ Retention

	Training ops (per token)	Inference Ops (per token)	Memory
Transformers	$O(c)$	$O(c)$	$O(c^2)$
Retention	$O(c)$	$O(1)$	$O(c)$

Additional Features

- Removal of Softmax
  - removes a nonlinear interaction
  - add some nonlinearity via Group Normalization
- Decay: replace masking matrix by

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \gamma & 1 & 0 & 0 \\ \gamma & \gamma & 1 & 0 \\ \dots & \dots & \dots & \dots \end{pmatrix} \in \mathbb{R}^{c \times c}$$

γ is a decay factor < 1

Decay Parameter γ

$$S_0 = 0$$

$$\text{for } i \in [1,2,\ldots,c]:$$

$$S_i = \gamma S_{i-1} + K_i^T V_i$$

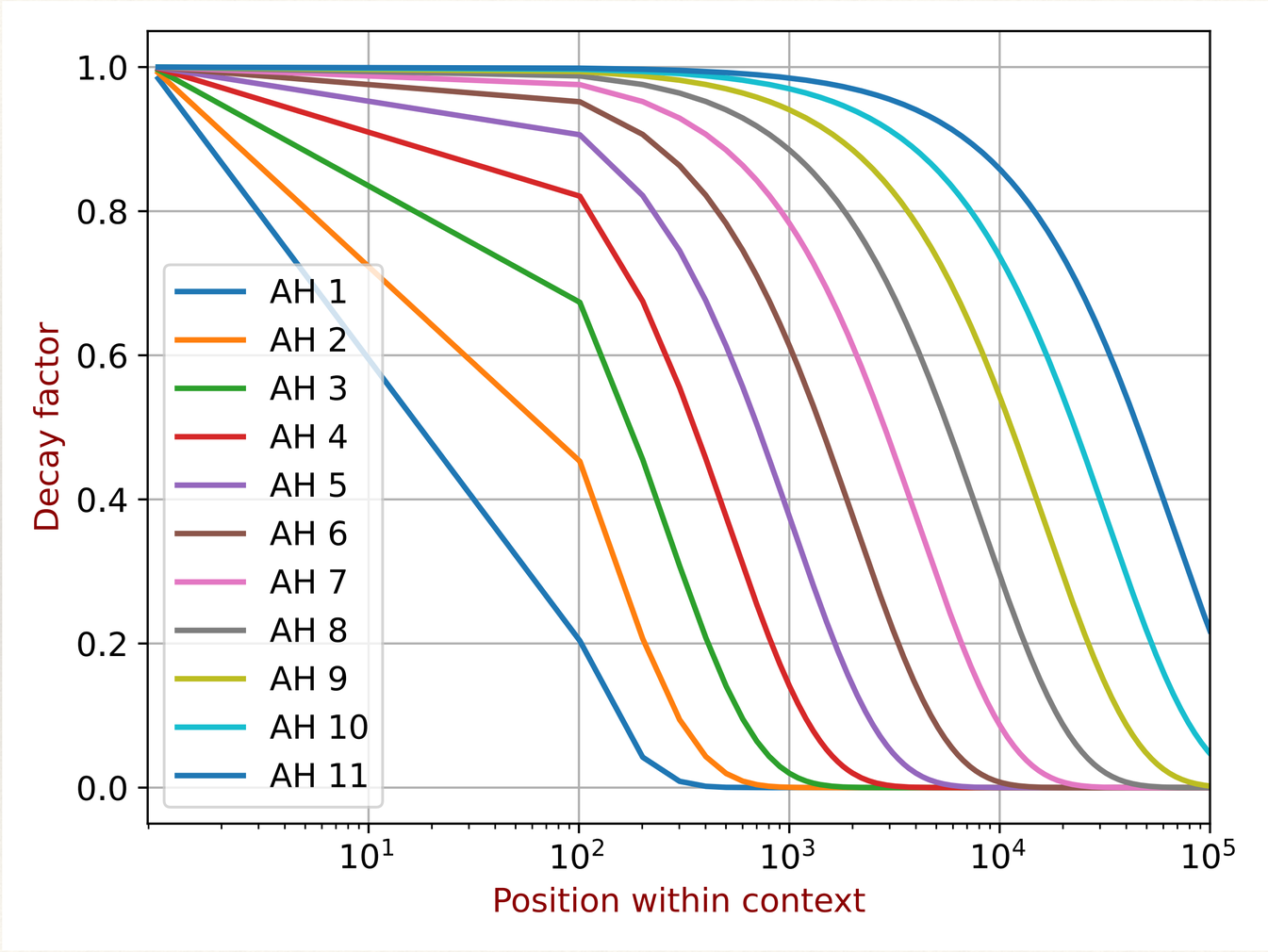
$$V_i^* = Q_i \cdot S_i$$

- Elements in the context with great separation have low interaction
- If $\gamma = 1 - 2^{-5}$

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \gamma & 1 & 0 & 0 \\ \gamma & \gamma & 1 & 0 \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix} \in \mathbb{R}^{c \times c}$$

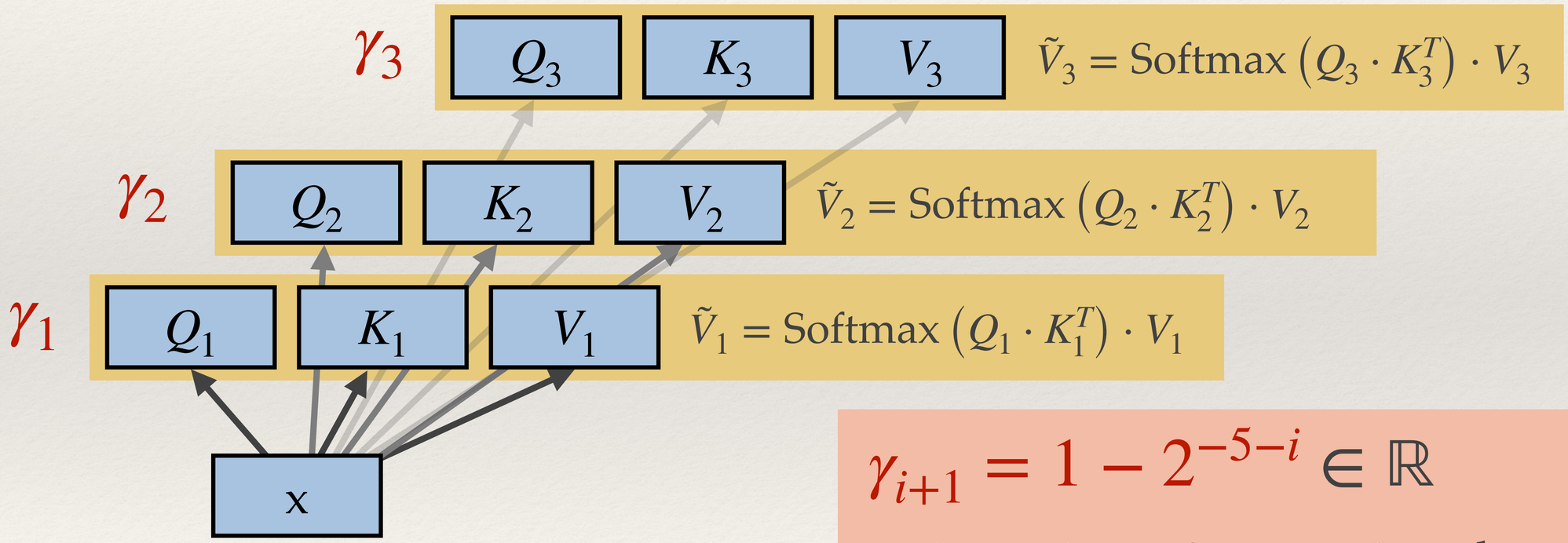
c	γ^c
1000	10^{-14}
10000	10^{-133}

Effect of γ Decay



Multiscale Retention

- Set different decay factor γ for each head of Multihead Attention



$$\gamma_{i+1} = 1 - 2^{-5-i} \in \mathbb{R}$$

independent of context length c!

Relative Position Encoding

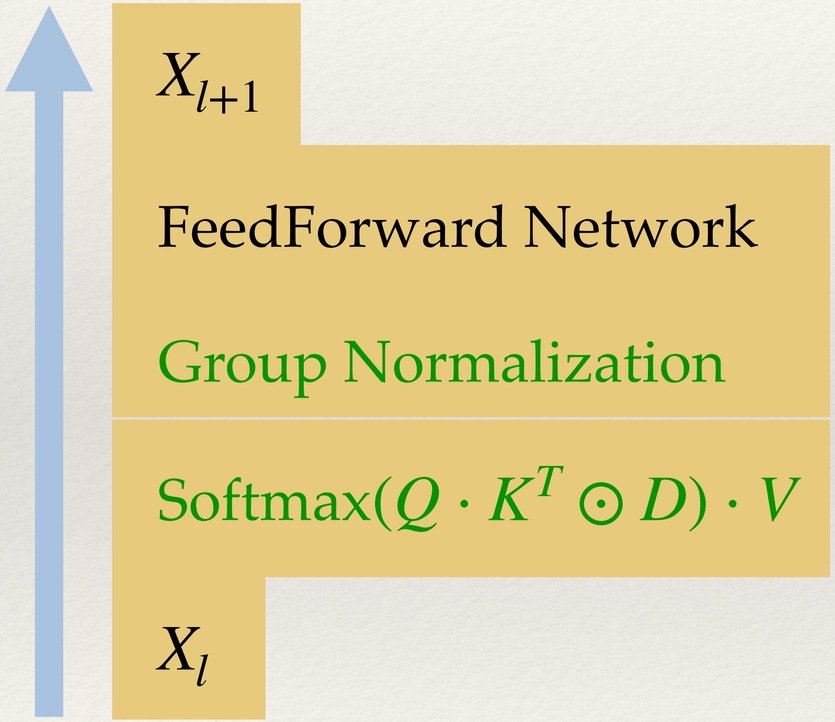
- See paper

Group Normalization

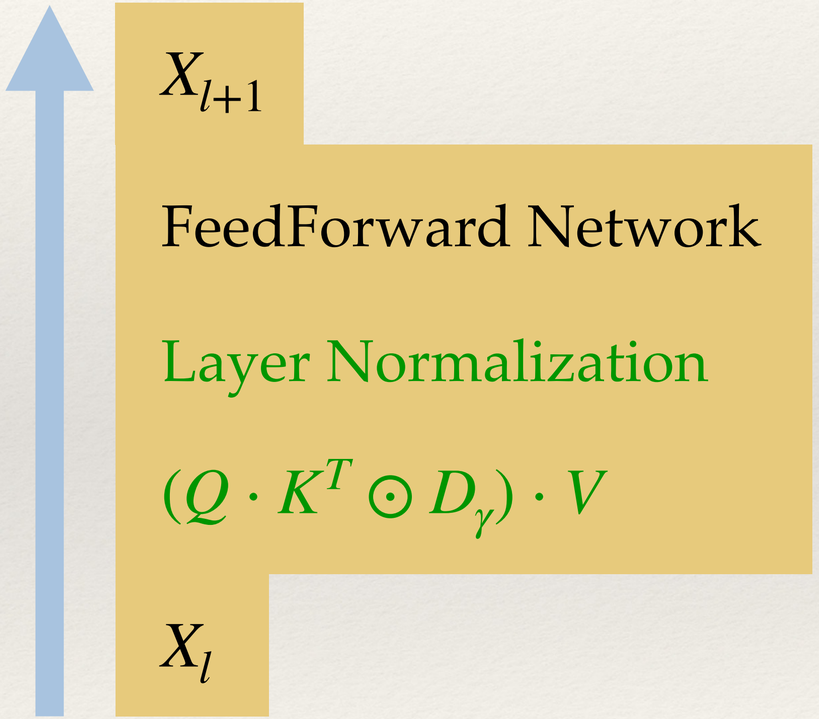
- 2018
- Similar to layer normalization
  - divides features R^d into different groups
  - layer normalization in each group
- Each group is one attention channel

Single Layer

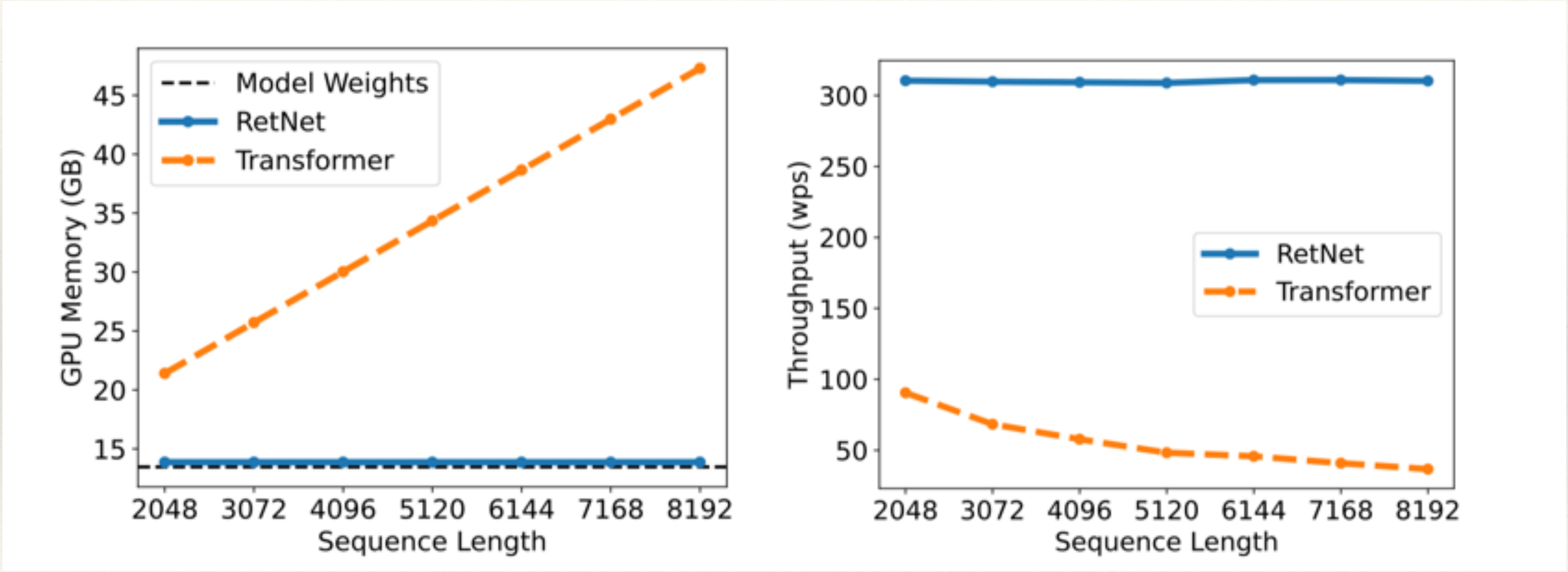
Transformer



Retentive

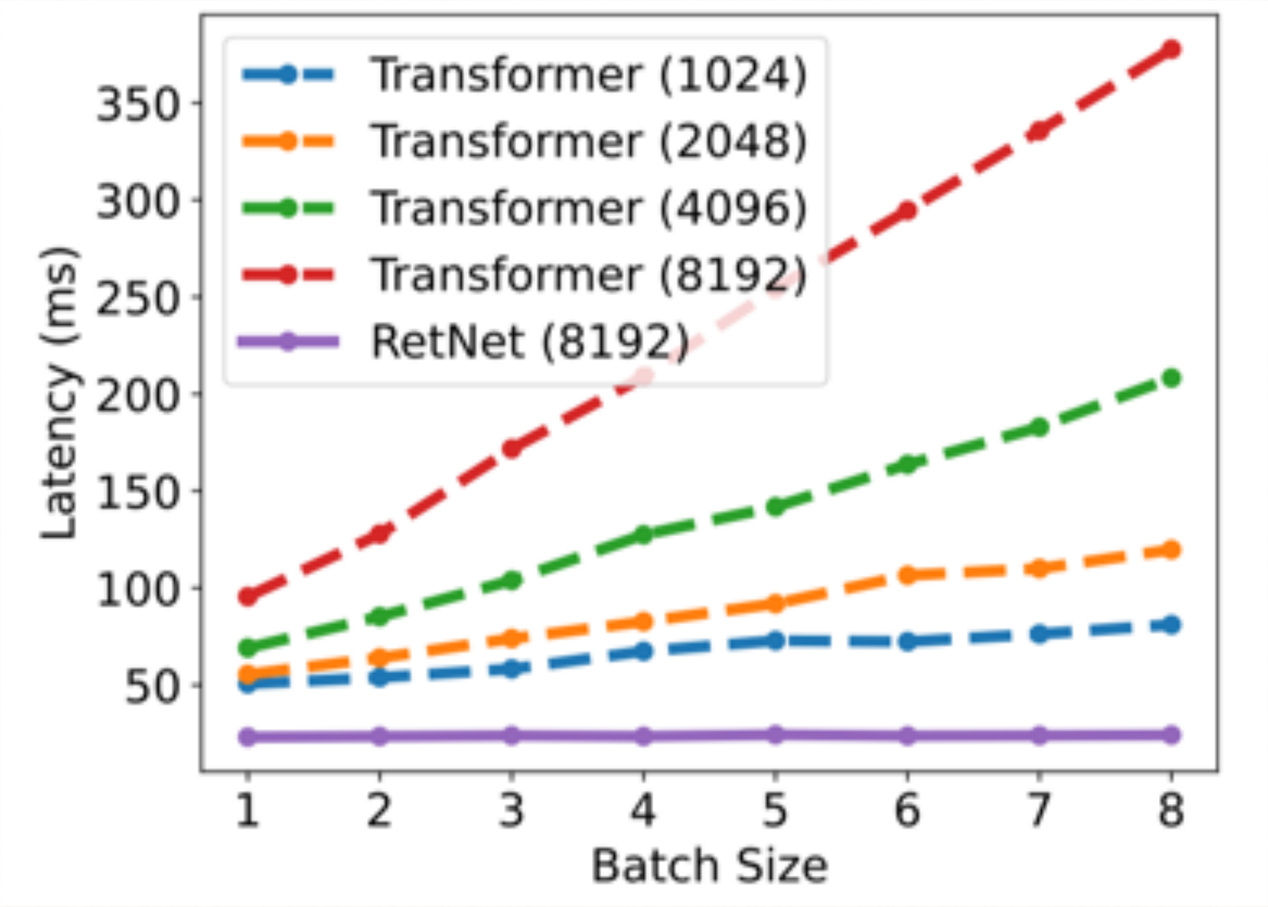


Memory and Throughput



Source: Retentive Network: A Successor to Transformer for Large Language Models, 2023, Sun metal

Inference Latency vs Batch Size



Source: Retentive Network: A Successor to Transformer for Large Language Models, 2023, Sun metal

Perplexity of Different Methods

Method	In-Domain	PG22	QMSum	GovReport	SummScreen
RWKV	30.92	51.41	28.17	19.80	25.78
H3	29.97	49.17	24.29	19.19	25.11
Hyena	32.08	52.75	28.18	20.55	26.51
Linear Transformer	40.24	63.86	28.45	25.33	32.02
RetNet	**26.05**	**45.27**	**21.33**	**16.52**	**22.48**

Source: Retentive Network: A Successor to Transformer for Large Language Models, 2023, Sun metal

Ablation Study

Method	In-Domain	PG22	QMSum	GovReport	SummScreen
RetNet	**26.05**	**45.27**	**21.33**	**16.52**	**22.48**
– swish gate	27.84	49.44	22.52	17.45	23.72
– GroupNorm	27.54	46.95	22.61	17.59	23.73
– γ decay	27.86	47.85	21.99	17.49	23.70
– multi-scale decay	27.02	47.18	22.08	17.17	23.38
Reduce head dimension	27.68	47.72	23.09	17.46	23.41

Source: Retentive Network: A Successor to Transformer for Large Language Models, 2023, Sun metal

Questions?

Questions?

Questions?

Questions?