Retentive Network: A Successor to Transformer for Large Language Models

Yutao Sun - Li Dong - Shaohan Huang - Shuming Ma - Yuqing Xia - Jilong Xue - Jianyong Wang - Furu Wei
Microsoft Research, Tsinghua University

Presented by: Amirhossein Alimohammadi

Retentive Network

Ret-Net: A foundation architecture for large language models, replacing transformers.

Simultaneously, achieving 3 attributes:

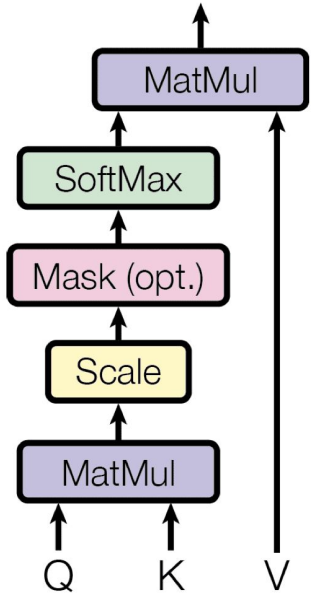
1. Training Parallelism
2. Low-Cost Inference
3. Strong Performance

What is the problem of transformers that need to be fixed?

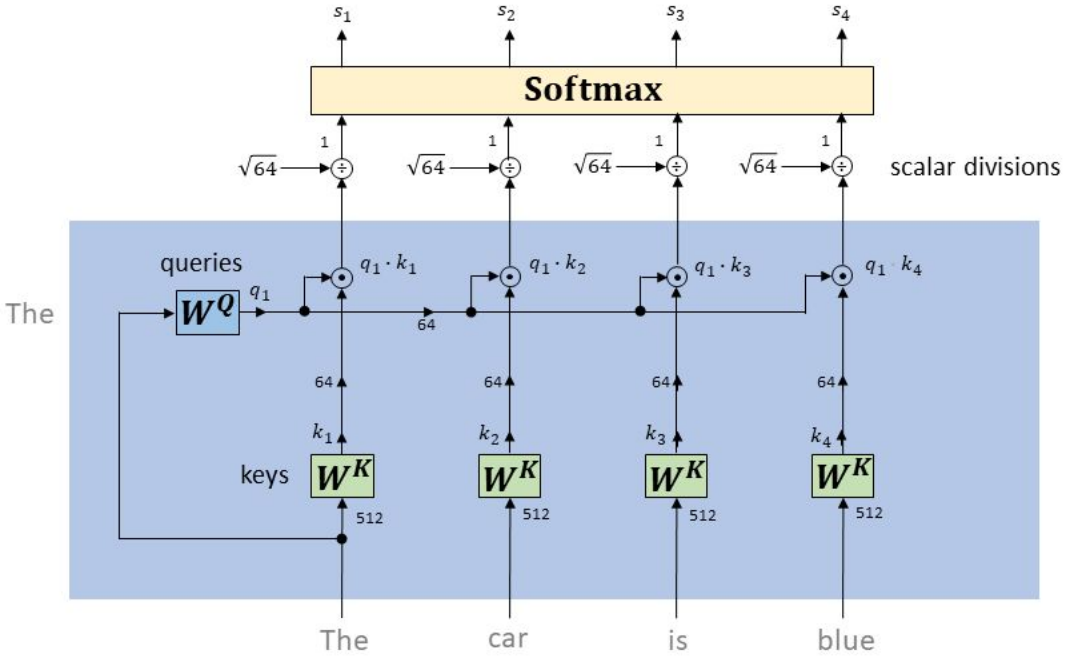
Training Memory Order: O(n^2)

Inference Order: O(n)

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

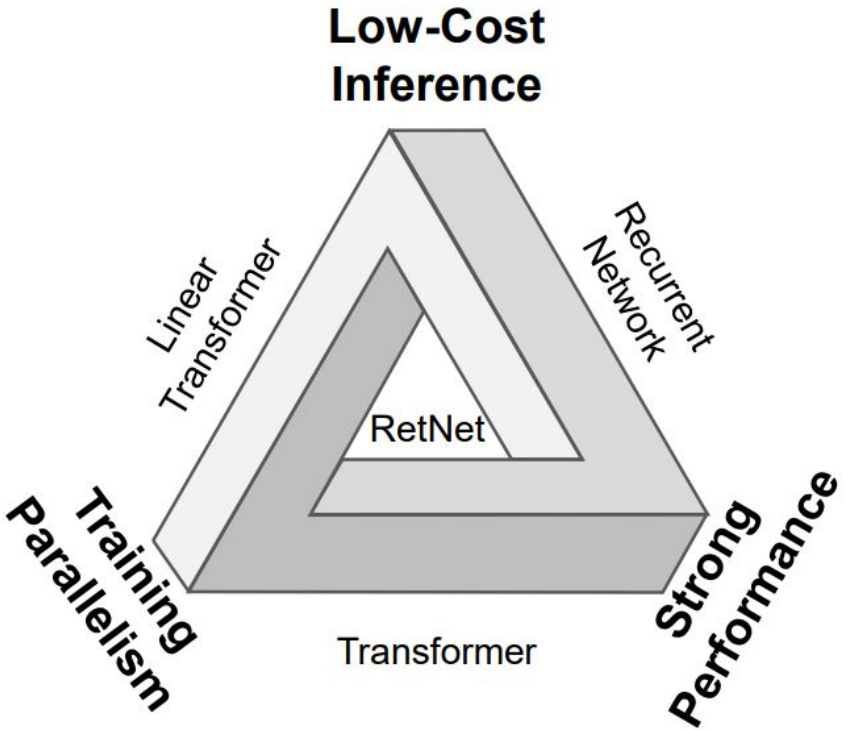


How this happens?



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

Impossible Triangle

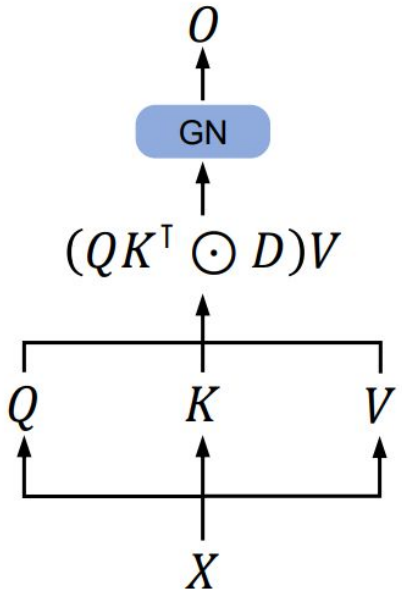


Retention

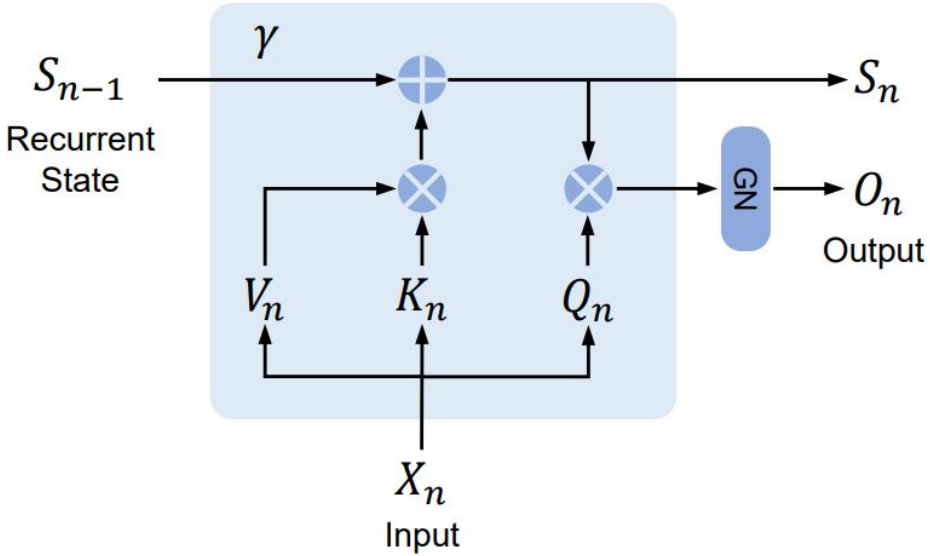
$$Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \overline{\Theta}, \quad V = XW_V$$

$$\Theta_n = e^{in\theta}, \quad D_{nm} = \begin{cases} \gamma^{n-m}, & n \geq m \\ 0, & n < m \end{cases}$$

$$\text{Retention}(X) = (QK^\intercal \odot D)V$$



How does it solve the problem?



The Chunkwise Recurrent Representation of Retention

$$Q_{[i]} = Q_{Bi:B(i+1)}, \quad K_{[i]} = K_{Bi:B(i+1)}, \quad V_{[i]} = V_{Bi:B(i+1)}$$

$$R_i = K_{[i]}^{\intercal}(V_{[i]} \odot \zeta) + \gamma^B R_{i-1}, \quad \zeta_{ij} = \gamma^{B-i-1}$$

$$\text{Retention}(X_{[i]}) = \underbrace{(Q_{[i]}K_{[i]}^{\intercal} \odot D)V_{[i]}}_{\text{Inner-Chunk}} + \underbrace{(Q_{[i]}R_{i-1}) \odot \xi}_{\text{Cross-Chunk}}, \quad \xi_{ij} = \gamma^{i+1}$$

Gated Multi-Scale Retention

Multi-Scale Retention (MSR) assigns different γ for each head.

$$
\begin{aligned}
\gamma &= 1 - 2^{-5-\text{arange}(0,h)} \in \mathbb{R}^h \\
\text{head}_i &= \text{Retention}(X, \gamma_i) \\
Y &= \text{GroupNorm}_h(\text{Concat}(\text{head}_1, \cdots, \text{head}_h)) \\
\text{MSR}(X) &= (\text{swish}(XW_G) \odot Y)W_O
\end{aligned}
$$

Model Comparison

Architectures	Training Parallelization	Inference Cost	Long-Sequence Memory Complexity	Performance
Transformer	✔	$O(N)$	$O(N^2)$	✔✔
Linear Transformer	✔	$O(1)$	$O(N)$	✘
Recurrent NN	✘	$O(1)$	$O(N)$	✘
RWKV	✘	$O(1)$	$O(N)$	✔
H3/S4	✔	$O(1)$	$O(N \log N)$	✔
Hyena	✔	$O(N)$	$O(N \log N)$	✔
RetNet	✔	$O(1)$	$O(N)$	✔✔

Comparison to Linear Attention

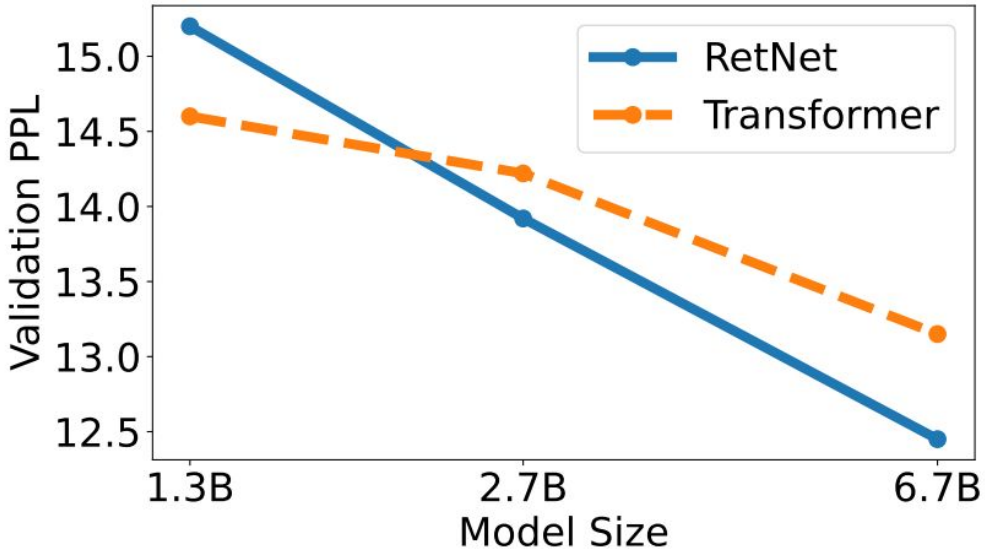
In Linear Attention, we observe various kernels, replacing softmax:

$$\phi(q_i)\phi(k_j) \Big/ \sum_{n=1}^{|x|} \phi(q_i)\phi(k_n)$$

However, it suffers from encoding the position information effectively which leads to weaker performance.

Experiments

Size	Hidden Dim.	#Layers	Batch Size	# Tokens	Learning Rate
1.3B	2048	24	4M	100B	6×10^{-4}
2.7B	2560	32	4M	100B	3×10^{-4}
6.7B	4096	32	4M	100B	3×10^{-4}



More Detail

Formulate the mapping in a recurrent manner:

$$\boldsymbol{s}_n = A\boldsymbol{s}_{n-1} + K_n^\intercal v_n, \qquad A \in \mathbb{R}^{d\times d}, K_n \in \mathbb{R}^{1\times d}$$

$$o_n = Q_n \boldsymbol{s}_n = \sum_{m=1}^{n} Q_n A^{n-m} K_m^\intercal v_m, \qquad Q_n \in \mathbb{R}^{1\times d}$$

Diagonalizing the matrix

By diagonalizing the matrix in this way:

$$A = \Lambda(\gamma e^{i\theta})\Lambda^{-1}$$

The formula in the last slide change to:

$$o_n = \sum_{m=1}^{n} Q_n(\gamma e^{i\theta})^{n-m} K_m^{\intercal} v_m$$

$$= \sum_{m=1}^{n} (Q_n(\gamma e^{i\theta})^n)(K_m(\gamma e^{i\theta})^{-m})^{\intercal} v_m$$

It can be simplified to:

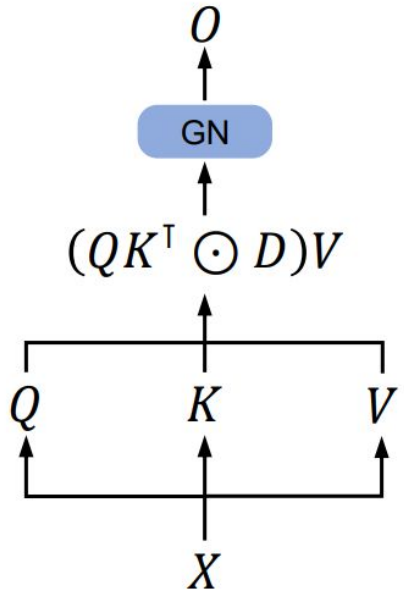
$$o_n = \sum_{m=1}^{n} \gamma^{n-m}(Q_n e^{in\theta})(K_m e^{im\theta})^{\dagger} v_m$$

Retention

$$Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \overline{\Theta}, \quad V = XW_V$$

$$\Theta_n = e^{in\theta}, \quad D_{nm} = \begin{cases} \gamma^{n-m}, & n \geq m \\ 0, & n < m \end{cases}$$

$$\text{Retention}(X) = (QK^\intercal \odot D)V$$



Comparisons with Transformer

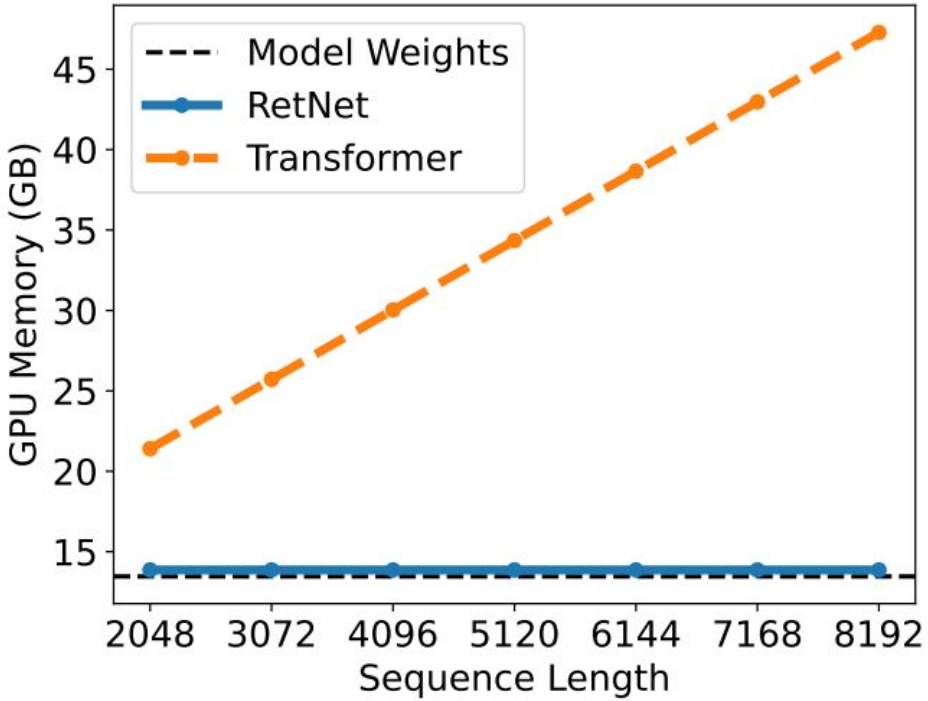
	HS	BoolQ	COPA	PIQA	Winograd	Winogrande	SC	Avg
Zero-Shot								
Transformer	55.9	62.0	69.0	74.6	69.5	56.5	75.0	66.07
RetNet	**60.7**	**62.2**	**77.0**	**75.4**	**77.2**	**58.1**	**76.0**	**69.51**
4-Shot								
Transformer	55.8	58.7	71.0	75.0	71.9	57.3	75.4	66.44
RetNet	**60.5**	**60.1**	**78.0**	**76.0**	**77.9**	**59.9**	**75.9**	**69.76**

Comparisons with Transformer

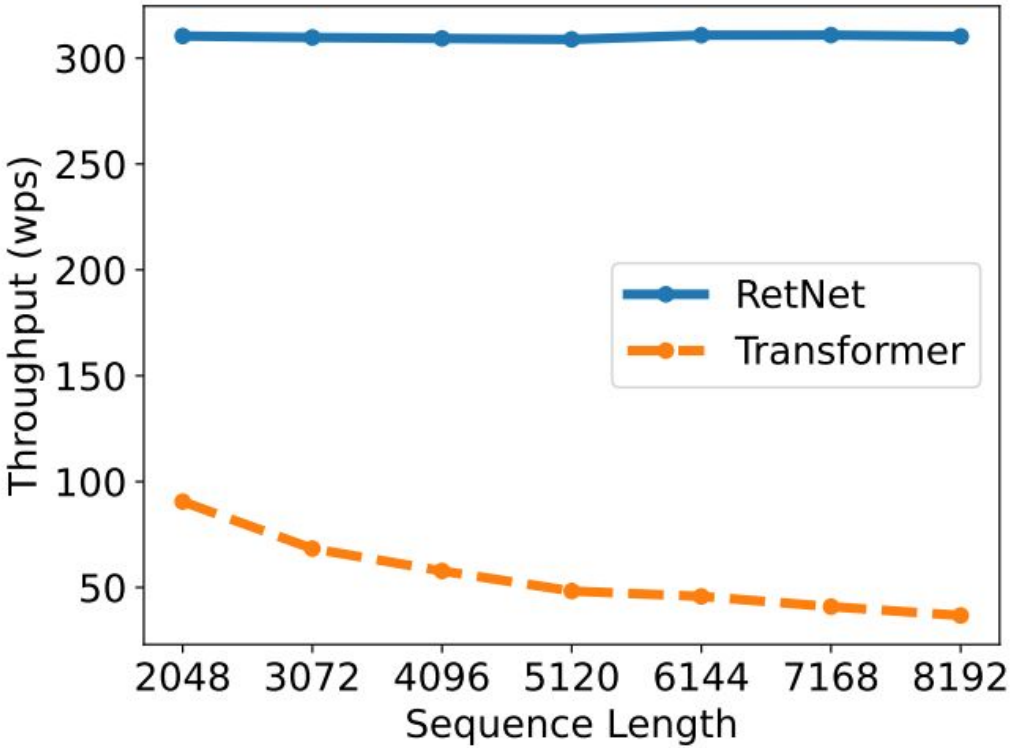
Model Size	Memory (GB) ↓			Throughput (wps) ↑		
	Trm	Trm+FlashAttn	RetNet	Trm	Trm+FlashAttn	RetNet
1.3B	74.8	38.8	34.5	10832.4	63965.2	73344.8
2.7B	69.6	42.1	42.0	5186.0	34990.2	38921.2
6.7B	69.0	51.4	48.0	2754.4	16230.1	17458.6
13B	61.4	46.3	45.9	1208.9	7945.1	8642.2

Training cost of Transformer (Trm), Transformer with FlashAttention (Trm+FlashAttn), and RetNet. Memory consumption and training throughput are reported (word per second; wps).

GPU Memory Cost



Throughput


Throughput (wps)
300
250
200
150
100
50
2048
3072
4096
5120
6144
7168
8192
Sequence Length
RetNet
Transformer

Perplexity Result

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**

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

Thank you!

- Thank you for your attention!
- I appreciate your time and interest.
- If you have any questions, please feel free to ask.
- Contact information: alimohammadiamirhossein@gmail.com