Retentive Network: A Successor to Transformer for Large Language Models

Sun, Yutao, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei

Microsoft Research Tsinghua University

November 17, 2023

Presented by Boyao Li

Overview

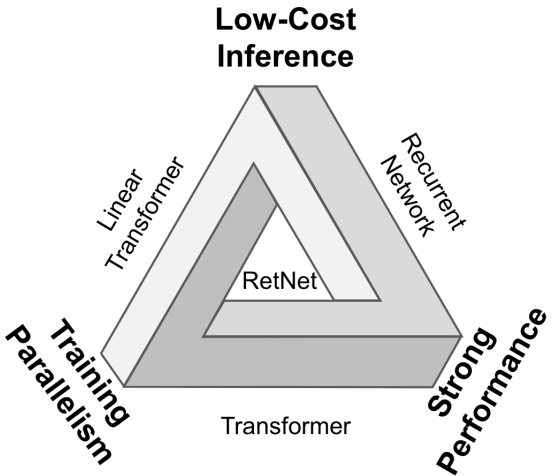
- Introduction
- RetNet
  - Retention
  - Gated Multi-Scale Retention
- Experiments
  - Comparison with Transformer
  - Comparison with Transformer Variants

Introduction

This paper proposes **retentive network** (RetNet) as a foundation architecture for large language models (LLMs), simultaneously achieving better inference efficiency (in terms of memory, speed, and latency), favorable training parallelization, and competitive performance compared with Transformers.

- Derive the connection between recurrence and attention.
- Propose the retention mechanism for sequence modeling that supports parallel, recurrent, and chunkwise recurrent representations.
  - Parallel $\Longrightarrow$ training parallelism
  - Recurrent $\Longrightarrow$ low-cost inference
  - Chunkwise recurrent $\Longrightarrow$ efficient long-sequence modeling with linear complexity

"Impossible Triangle"



RetNet Architecture

- L identical blocks with residual connection and pre-LayerNorm as in Transformers.
- Each block contains a **multi-scale retention** (MSR) module and a feed-forward network (FFN) module.
- Input sequence: $\{\boldsymbol{x}_i\}_{i=1}^{|x|}$
- Packed embeddings (after a word embedding layer): $X^0 = [\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{|x|}] \in \mathbb{R}^{|x| \times d_{\text{model}}}$
- Calculate X^L:

$$Y^l = \text{MSR}(\text{LN}(X^l)) + X^l$$
$$X^{l+1} = \text{FFN}(\text{LN}(Y^l)) + Y^l$$

where $\text{LN}(\cdot)$ means LayerNorm, and $\text{FFN}(X) = \text{GELU}(XW_1)W_2$. W_1, W_2 are parameter matrices.

Review: Attention

Scaled Dot-Product Attention

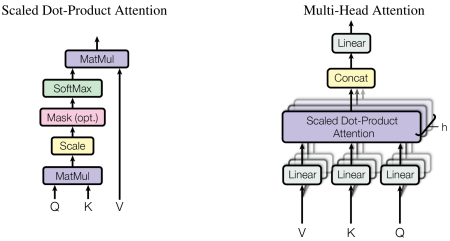
- Queries(d_k), keys(d_k) and values(d_v)
- $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V$

Multi-Head Attention

- $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$, where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

- $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$

In vanilla Transformer,
$d_{model} = 512, h = 8, d_k = d_v = d_{model}/h = 64$.



Retention

- For a single word $\boldsymbol{x}_n$, we want to map its value v_n to output o_n.
- Formulate the mapping $v_n \mapsto o_n$ in a recurrent manner with state s_n:

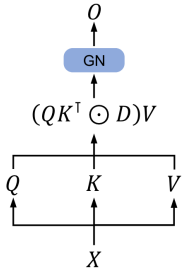
$$s_n = As_{n-1} + K_n^\intercal v_n$$
$$o_n = Q_n s_n = \sum_{m=1}^{n} Q_n A_{n-m} K_m^\intercal v_m$$

  where $A \in \mathbb{R}^{d \times d}$, $Q_n, K_n \in \mathbb{R}^{1 \times d}$.
- Diagonalize matrix $A = \Lambda(\gamma e^{i\theta})\Lambda^{-1}$ $(\gamma, \theta \in \mathbb{R}^d)$ and rewrite the equation:

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

Parallel Representation of Retention

- $Q = (XW_Q) \odot \Theta, K = (XW_K) \odot \bar{\Theta}, V = XW_V$, where $\Theta = (\theta_1, \theta_2, ..., \theta_{|x|})$ and $\theta_n = e^{in\theta}$.
- $D \in \mathbb{R}^{|x| \times |x|}$, where $D_{nm} = \gamma^{n-m} \mathbb{1}_{n \geq m}$. D combines causal masking and exponential decay along relative distance.
- $\text{Retention}(X) = (QK^\intercal \odot D)V$
- $\text{GN}(\cdot)$ means GroupNorm.

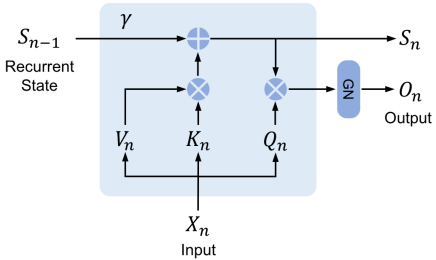


(a) Parallel representation.

Recurrent Representation of Retention

For $n-$th time-step, We can obtain the output recurrently:

$$S_n = \gamma S_{n-1} + K_n^{\intercal} V_n$$
$$\text{Retention}(X_n) = Q_n S_n, \quad n = 1, 2, ..., |x|$$



(b) Recurrent representation.

Chunkwise Recurrent Representation of Retention

- Hybrid form of parallel representation and recurrent representation
- Inner-chunk information follows parallel representation, while cross-chunk information follows recurrent representation.
- Let B denote the chunk length, $i-$th chunk $\boldsymbol{x}_{[i]} = [\boldsymbol{x}_{(i-1)B+1}, ..., \boldsymbol{x}_{iB}]$
- Retention output of $\boldsymbol{x}_{[i]}$:

$$
\begin{aligned}
&Q_{[i]} = Q_{iB:(i+1)B}, \quad K_{[i]} = K_{iB:(i+1)B}, \quad V_{[i]} = V_{iB:(i+1)B} \\
&R_i = K_{[i]}(V_{[i]} \odot \zeta) + \gamma^B R_{i-1}, \quad \zeta_{ij} = \gamma^{B-i-1} \\
&\text{Retention}(X_{[i]}) = \underbrace{(Q_{[i]} K_{[i]}^{\intercal} \odot D) V_{[i]}}_{\text{Inner-Chunk}} + \underbrace{(Q_{[i]} R_{i-1}) \odot \xi}_{\text{Cross-Chunk}}, \quad \xi_{ij} = \gamma^{i+1}
\end{aligned}
$$

Gated Multi-Scale Retention

- Use $h = d_{\text{model}}/d$ retention heads in each layer, where d is the head dimension. Parameter matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$.
- Assign different γ for each head.
- Add a swish gate ($\text{swish}(x) = x \cdot \text{sigmoid}(x)$) increase the non-linearity of retention layers.
- Given Input X,

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

Retention Score Normalization

- GroupNorm is scale-invariant, i.e., $\text{GroupNorm}(\alpha \cdot \text{head}_i) = \text{GroupNorm}(\text{head}_i)$.
- $QK^\intercal \longrightarrow QK^\intercal / \sqrt{d}$
- $D_{nm} \longrightarrow D_{nm} / \sqrt{\sum_{i=1}^{|x|} D_{ni}}$
- Retention scores R, $R_{nm} \longrightarrow R_{nm} / \max(|\sum_{i=1}^{|x|} R_{ni}|, 1)$
- Stabilize the numerical flow of both forward and backward passes.

Comparison with Previous Methods

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	✔	$O(1)$	$O(N)$	✔✔

Setup for Experiments

- Parameter reallocation for MSR and FFN (here $d = d_{\text{model}}$)
  - MSR: $W_Q, W_K \in \mathbb{R}^{d \times d}$, $W_V, W_G \in \mathbb{R}^{d \times 2d}$, $W_O \in \mathbb{R}^{2d \times d}$.
  - FFN: set intermediate dimension to $2d$
  - Set $d = 256$.
  - Keep γ identical among different model sizes, where $\gamma = 1 - e^{\text{linspace}(\log 1/32, \log 1/512, h)} \in \mathbb{R}^h$.
- Language Model Training
  - Various model sizes (i.e., 1.3B, 2.7B and 6.7B)
  - Append [bos] token to indicate the start of a sequence.
  - Use AdamW optimizer.
  - Implement based on TorchScale.

Setup for Experiments

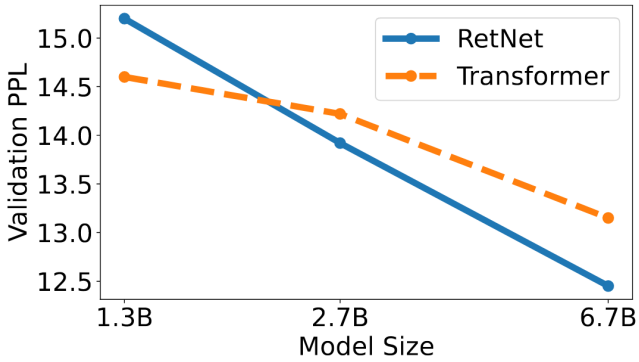
Sizes and hyper-parameters in experiments

Hyperparameters	**1.3B**	**2.7B**	**6.7B**
Layers	24	32	32
Hidden size	2048	2560	4096
FFN size	4096	5120	8192
Heads	8	10	16
Learning rate	6×10^{-4}	3×10^{-4}	3×10^{-4}
LR scheduler	Polynomial decay		
Warm-up steps	375		
Tokens per batch	4M		
Adam β	(0.9, 0.98)		
Training steps	25,000		
Gradient clipping	2.0		
Dropout	0.1		
Weight decay	0.01		

Language Modeling

Perplexity(PPL): Given a tokenized sequence $X = (x_0, ..., x_t)$,

$$\text{PPL}(x) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}$$



Zero-shot and Few-shot Evaluation on Downstream Tasks

	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**

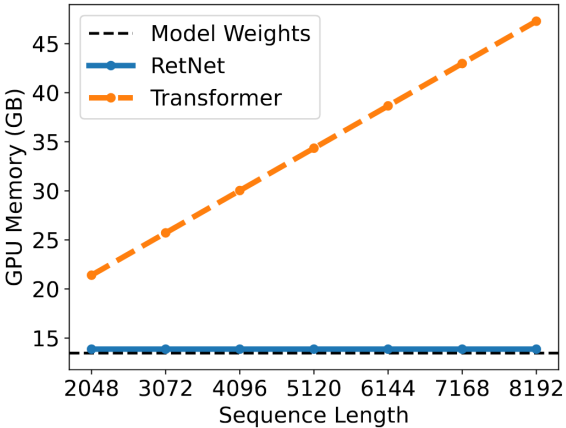
Table 3: Zero-shot and few-shot learning with Transformer and RetNet. The model size is 6.7B.

Training Cost

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

- wps=word per second
- Training sequence length=8192
- Implement RetNet with vanilla PyTorch code, without using kernel fusion or FlashAttention-like acceleration

Inference Cost



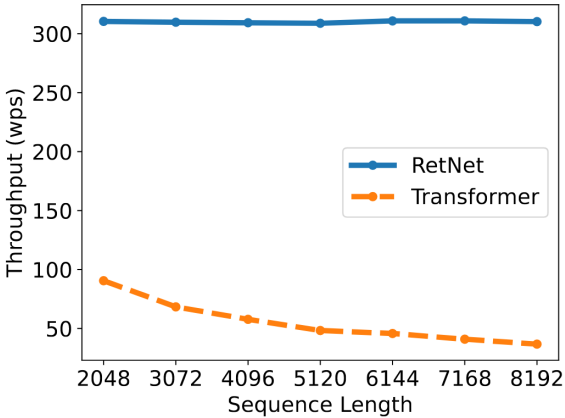
(a) GPU memory cost of Transformer and RetNet.

Memory

- Model size is 6.7B.
- Transformer reuses KV caches of previously decoded tokens.
- RetNet uses the recurrent representation.

Inference Cost

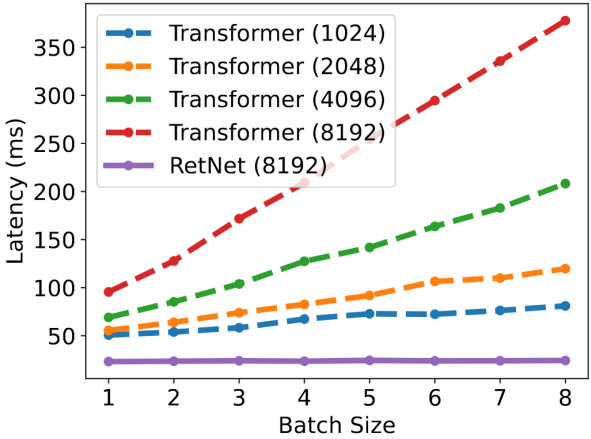
Throughput



(b) Throughput of Transformer and RetNet.

Inference Cost

Latency



(c) Inference latency with different batch sizes.

Comparison with Transformer Variants

All models have 200M parameters with 16 layers and a hidden dimension of 1024.

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**

Table 5: Perplexity results on language modeling. RetNet outperforms other architectures on both the in-domain evaluation set and various out-of-domain corpora.

Ablation Studies

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

Table 6: Ablation results on in-domain and out-of-domain corpora.