ICML
International Conference
On Machine Learning

上海人工智能实验室
Shanghai Artificial Intelligence Laboratory

Liger: Linearizing Large Language Models to Gated Recurrent Structures

Disen Lan[12], Weigao Sun[1†], Jiaxi Hu[3], Jusen Du[14], Yu Cheng[5†]

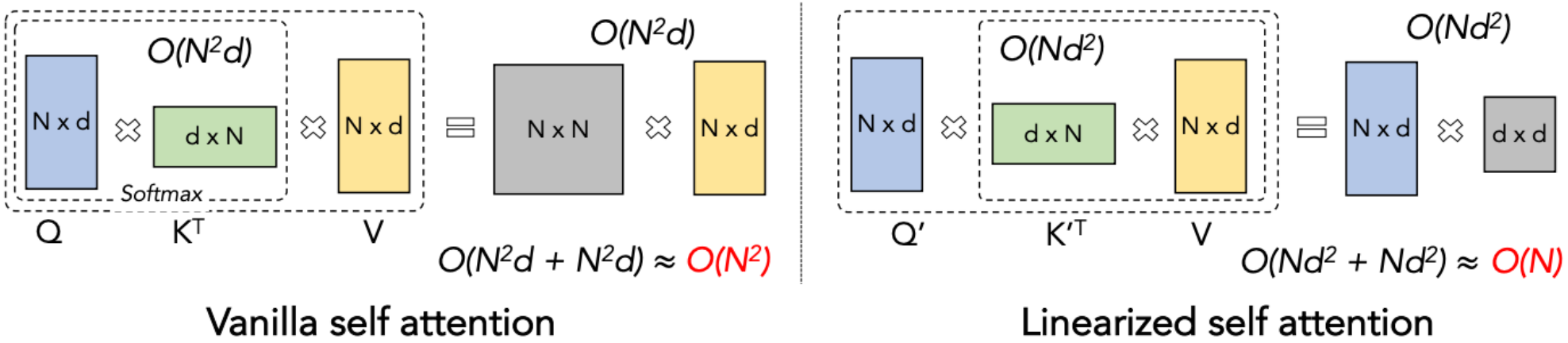
[1]Shanghai AI Laboratory, [2]South China University of Technology,

[3]The Hong Kong University of Science and Technology (Guangzhou)

[4]Nanjing University, [5]The Chinese University of Hong Kong

Introduction

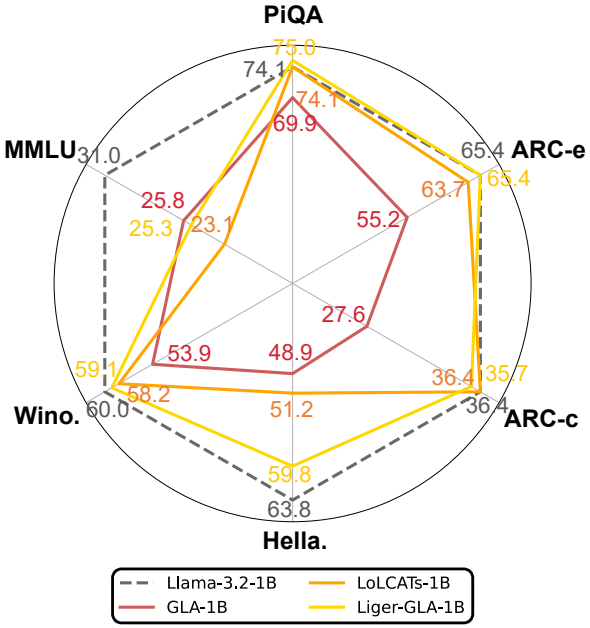
- **Transformers**
  - Quadratic Complexity
  - Glowing Memory Usage (KV-cache)
- **Linear Recurrent Models (Linear Attention, RNNs, SSMs):**
  - Linear-time Complexity
  - Constant Memory Usage

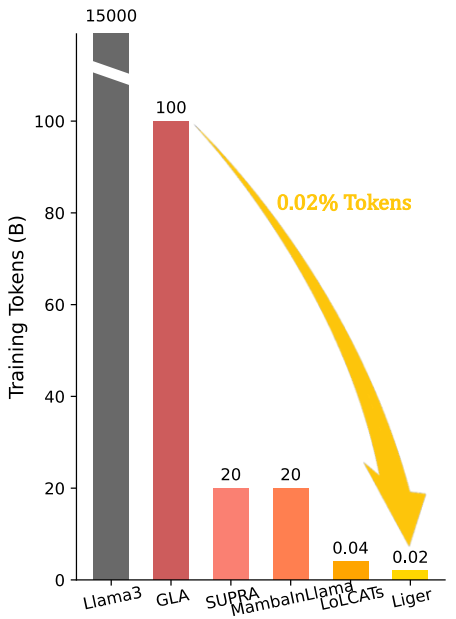


Zhen Qin, et al. cosFormer: Rethinking Softmax in Attention. ICLR, 2022.

Introduction

- **Pretrain LLMs from scratch: *Extremely* high resource requirement** ☹️
  - Data: ~15T tokens
  - GPU: 48K × H100
  - Time: ~6.4M GPU hours
- **Linearization: *Existing pretrained* Transformer → Linear Recurrent Model** 🤩
  - Efficient, effective, lightweight, low cost



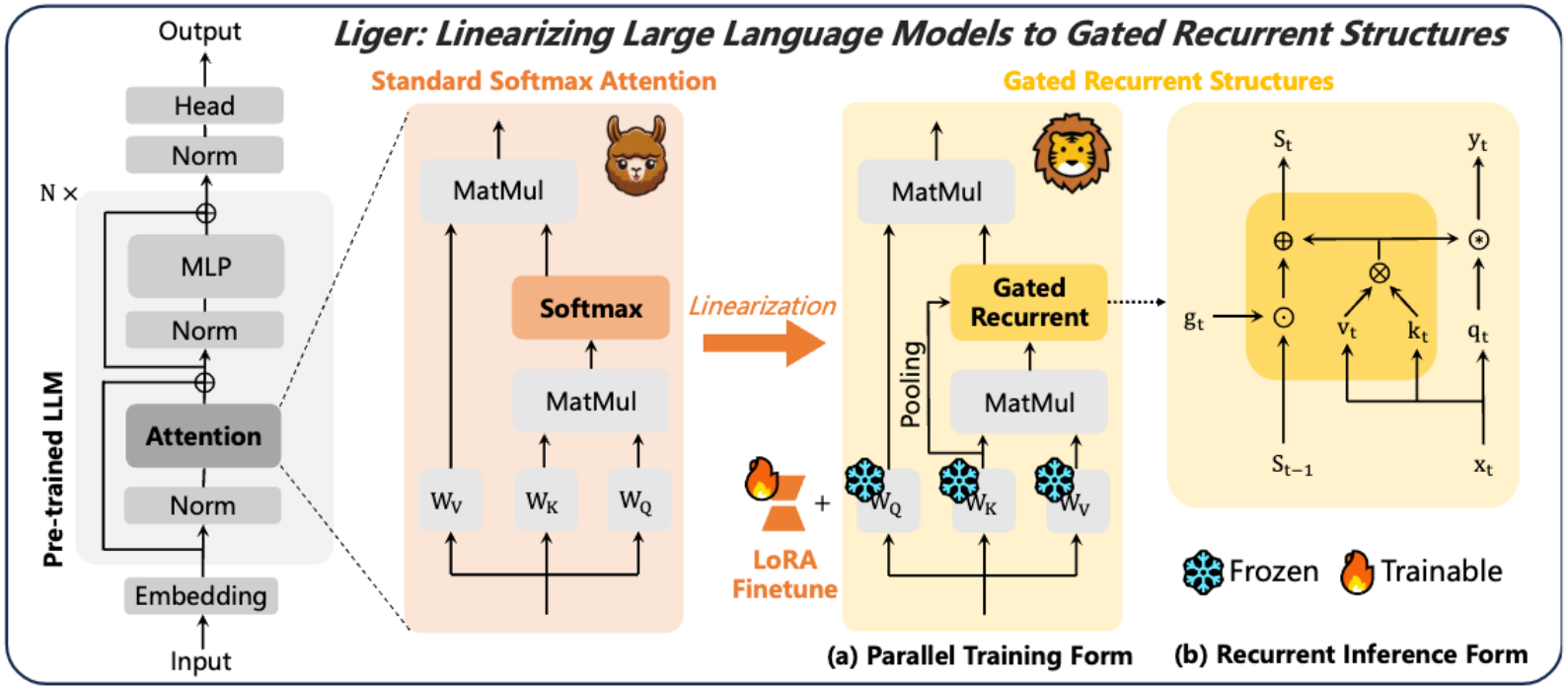


Motivation

- **How to incorporate *additional gate module* into linearization?**
  - Need to trained from scratch → Increase linearization process complexity & cost
  - Larger architectural divergence from Transformer-based LLMs → Limited performance
- **What is the *suitable* form of (Linear) Attention for linearization?**
- **How to *simplify* the linearization process?**
  - Multi-stage training → Higher linearization process cost
  - Do we really rely on distillation?

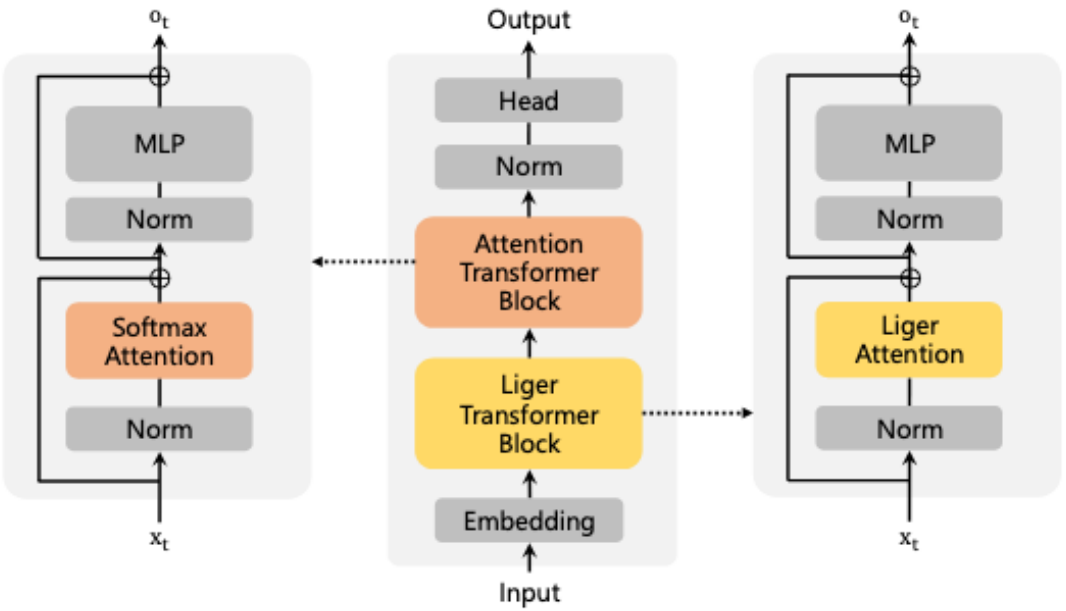
Methodology

- **Liger: Linearizing LLMs to gated recurrent structures**
  - Pooling for gate construction: Fully reuse pretrained LLM weights
  - Normalized feature mapping with the non-parametric way
  - LoRA Finetune: Significantly reduce linearization cost & Works w/o distillation



Methodology

- **Liger Attention**: *Intra-layer Hybrid* Attention
  - Gated Recurrent Modeling + Sliding Window Attention
  - Linear-time complexity: $O(TD^2 + TWD)$
- **Liger Architecture**: Support *Inter-layer Hybrid* Linearization



Main Results

- Liger achieves *better* performance with *less* training tokens compared with SOTA linearization methods

Model	Training Tokens (B)	PiQA	ARC-e	ARC-c	Hella.	Wino.	MMLU	Avg.	Avg.
		acc ↑	acc ↑	acc_norm ↑	acc_norm ↑	acc ↑	acc (5-shot) ↑	↑	(no MMLU) ↑
Mistral-7B	8000	80.6	80.7	53.9	81.1	74.3	62.6	72.2	74.1
SUPRA-Mistral-7B	100	80.4	75.9	45.8	77.1	70.3	34.2	64.0	69.9
LoLCATs-Mistral-7B Attn. Trf.	0.02	79.8	79.3	51.7	48.3	74.2	23.0	59.4	66.7
LoLCATs-Mistral-7B LoRA	0.02	77.3	74.9	45.1	40.9	67.9	23.0	54.8	61.2
LoLCATs-Mistral-7B	0.04	79.7	78.4	47.4	58.4	71.0	23.7	59.8	67.0
Liger-GLA-Mistral-7B (Ours)	0.02	80.1	78.7	49.3	76.3	70.1	36.3	65.1	70.9
Llama-3-8B	15000	79.4	80.1	53.2	79.2	72.9	65.3	71.7	73.0
SUPRA-Llama-3-8B	20	78.9	75.1	46.5	71.7	65.8	40.9	63.2	67.6
Mamba2-Llama-3-8B	20	76.8	74.1	48.0	70.8	58.6	43.2	61.9	65.6
Mamba2-Llama-3-8B 50% Attn.	20	81.5	78.8	58.2	79.5	71.5	56.7	71.0	73.9
LoLCATs-Llama-3-8B Attn. Trf.	0.02	78.4	79.3	51.9	51.6	73.4	23.5	59.7	66.9
LoLCATs-Llama-3-8B LoRA	0.02	72.4	72.6	44.3	34.6	68.0	23.0	52.5	58.4
LoLCATs-Llama-3-8B	0.04	80.1	80.4	53.5	63.4	72.9	42.1	65.4	70.0
Liger-GLA-Llama-3-8B (Ours)	0.02	80.3	81.1	52.5	76.3	72.0	43.4	67.6	72.4

Table 2. **Linearized LLMs Comparison.** Liger outperforms other linearization method on language modeling and understanding tasks with less training tokens across Mistral-7B and Llama-3-8B LLM architectures.

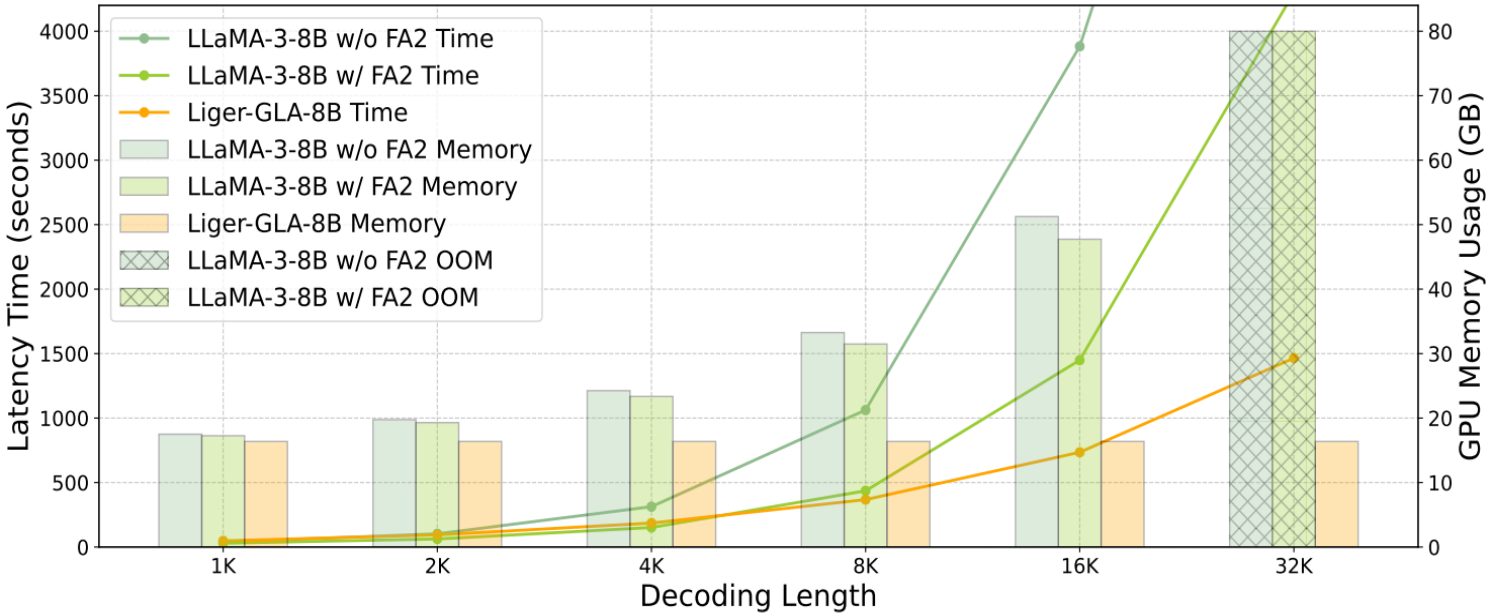
Main Results

- Liger achieves *competitive* performance with *significantly less* training tokens compared with pretrained LLMs

Model	Training Tokens (B)	PiQA	ARC-e	ARC-c	Hella.	Wino.	MMLU	Avg.	Avg.
		acc ↑	acc↑	acc_norm ↑	acc_norm ↑	acc ↑	acc (5-shot) ↑	↑	(no MMLU) ↑
(Transformer)									
Mistral-7B	8000	80.6	80.7	53.9	81.1	74.3	62.6	72.2	74.1
Llama-3-8B	15000	79.4	80.1	53.2	79.2	72.9	65.3	71.7	73.0
(Linear/Subquadratic)									
Mamba-7B	1200	81.0	77.5	46.7	77.9	71.8	33.3	64.7	71.0
RWKV-6-World-7B	1420	78.7	76.8	46.3	75.1	70.0	-	69.4	69.4
TransNormerLLM-7B	1400	80.1	75.4	44.4	75.2	66.1	43.1	64.1	68.2
Hawk-7B	300	80.0	74.4	45.9	77.6	69.9	35.0	63.8	69.6
Griffin-7B	300	81.0	75.4	47.9	78.6	72.6	39.3	65.8	71.1
(Hybrid)									
StripedHyena-Nous-7B	-	78.8	77.2	40.0	76.4	66.4	26.0	60.8	67.8
Zamba-7B	1000	81.4	74.5	46.6	80.2	76.4	57.7	69.5	71.8
Zamba2-7B	2100	81.0	80.3	56.4	81.5	77.2	64.8	73.5	75.3
(Linearized)									
Liger-GLA-Llama-3-8B (Ours)	0.02	80.3	81.1	52.5	76.3	72.0	43.4	67.6	72.4
Liger-GLA-Llama-3-8B-H (Ours)	0.02	80.6	80.7	52.7	76.9	71.4	44.4	67.8	72.5

Table 3. **Performance Comparison of Pre-trained and Linearized LLMs on Common-sense Reasoning and Knowledge Benchmarks.**s Results span Transformer-based (Mistral-7B, Llama-3-8B), linear/subquadratic (Mamba, RWKV), hybrid (Zamba), and our linearized Liger-GLA variants on language modeling and understanding tasks. Our Linearized Liger models achieve competitive performance with only 0.02B training tokens, demonstrating efficient adaptation to gated linear recurrent architectures.

Liger is Efficient and Scalable



Model	Avg. ↑	Avg. (no MMLU) ↑
Llama-3.2-1B	55.1	59.9
GLA-1B	46.9	51.1
LoLCATs-Llama-3.2-1B	51.1	56.7
Liger-GLA-Llama-3.2-1B	52.9	59.0
Llama-3.2-3B	66.1	68.1
GLA-3B	49.1	53.8
LoLCATs-Llama-3.2-3B	55.6	62.0
Liger-GLA-Llama-3.2-3B	60.7	66.5
Llama-3-8B	71.7	73.0
LoLCATs-Llama-3-8B	62.2	70.0
Liger-GLA-Llama-3-8B (Ours)	67.6	72.4

Liger can be used in various Linear Models

Model	Gate Parameterization	Pooling for Gate Construction
Gated Linear Attention (Yang et al., 2023)	$\mathbf{G}_t = \boldsymbol{\alpha}_t^\top \mathbf{1}$	$\boldsymbol{\alpha}_t = \sigma(\text{Pooling}(\boldsymbol{k}_t))$
Mamba2 (Dao & Gu, 2024)	$\mathbf{G}_t = \alpha_t \mathbf{1}^\top \mathbf{1}$	$\alpha_t = \exp(-\text{softplus}(\text{Pooling}(\boldsymbol{k}_t)))$
mLSTM (Beck et al., 2024)	$\mathbf{G}_t = \alpha_t \mathbf{1}^\top \mathbf{1}$	$\alpha_t = \sigma(\text{Pooling}(\boldsymbol{k}_t))$
Gated Retention (Sun et al., 2024b)	$\mathbf{G}_t = \alpha_t \mathbf{1}^\top \mathbf{1}$	$\alpha_t = \sigma(\text{Pooling}(\boldsymbol{k}_t))$
HGRN2 (Qin et al., 2024c)	$\mathbf{G}_t = \boldsymbol{\alpha}_t^\top \mathbf{1}$	$\boldsymbol{\alpha}_t = \gamma + (1-\gamma)\sigma(\text{Pooling}(\boldsymbol{k}_t))$
RWKV6 (Peng et al., 2024)	$\mathbf{G}_t = \boldsymbol{\alpha}_t^\top \mathbf{1}$	$\boldsymbol{\alpha}_t = \exp(-\exp(\text{Pooling}(\boldsymbol{k}_t)))$
Gated Slot Attention (Zhang et al., 2024c)	$\mathbf{G}_t = \boldsymbol{\alpha}_t^\top \mathbf{1}$	$\boldsymbol{\alpha}_t = \sigma(\text{Pooling}(\boldsymbol{k}_t))$

Gated Linear Recurrent Variants	Gated Memory Formulation	Output Formulation	Form of Gate G	Avg.	MMLU
				0-shot	5-shot
Liger-GLA	$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \boldsymbol{k}_t^\top \boldsymbol{v}_t$	$\boldsymbol{o}_t = \boldsymbol{q}_t \mathbf{S}_t$	$\mathbf{G}_t \in \mathbb{R}^D$	72.4	43.4
Liger-HGRN2	$\mathbf{S}_t = \mathbf{G}_t \mathbf{S}_{t-1} + (1-\mathbf{G}_t)^\top \boldsymbol{v}_t$	$\boldsymbol{o}_t = \boldsymbol{q}_t \mathbf{S}_t$	$\mathbf{G}_t \in \mathbb{R}^D$	69.5	36.2
Liger-GSA	$\begin{cases} \tilde{\mathbf{K}}_t = \mathbf{G}_t \tilde{\mathbf{K}}_{t-1} + (1-\mathbf{G}_t)^\top \boldsymbol{k}_t \\ \tilde{\mathbf{V}}_t = \mathbf{G}_t \tilde{\mathbf{V}}_{t-1} + (1-\mathbf{G}_t)^\top \boldsymbol{v}_t \end{cases}$	$\boldsymbol{o}_t = \tilde{\mathbf{V}}_t \,\text{Softmax}(\tilde{\mathbf{K}}_t^\top \boldsymbol{q}_t)$	$\mathbf{G}_t \in \mathbb{R}^M$	70.5	41.2

Thank you!