北京大学
PEKING UNIVERSITY

面向显存优化的大模型训练新方法

袁 坤

北京大学MELON课题组

2025年8月19日

课题组简介

北京大学**机器学习与最优化**(**M**achin**E** **L**earning and **O**ptimizatio**N**, **MELON**)课题组

MELON

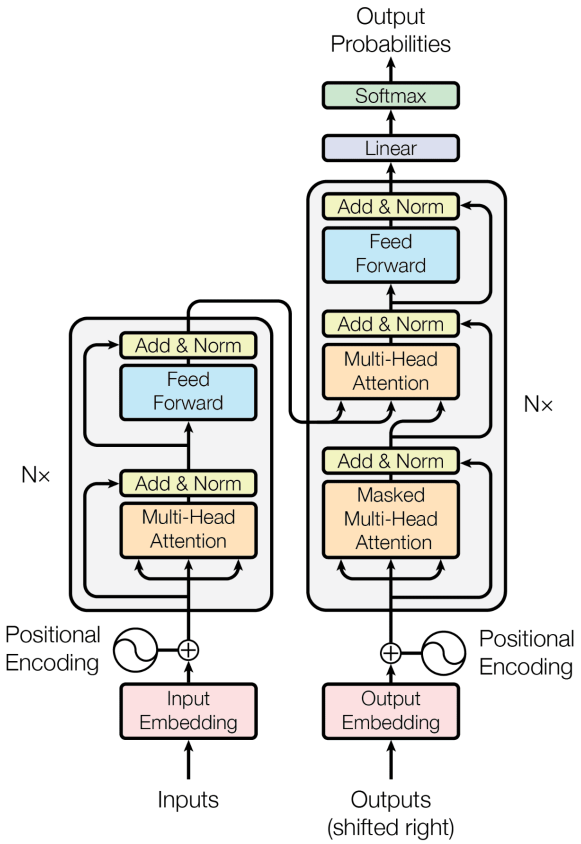
Machine Learning and Optimization

北京大学
PEKING UNIVERSITY

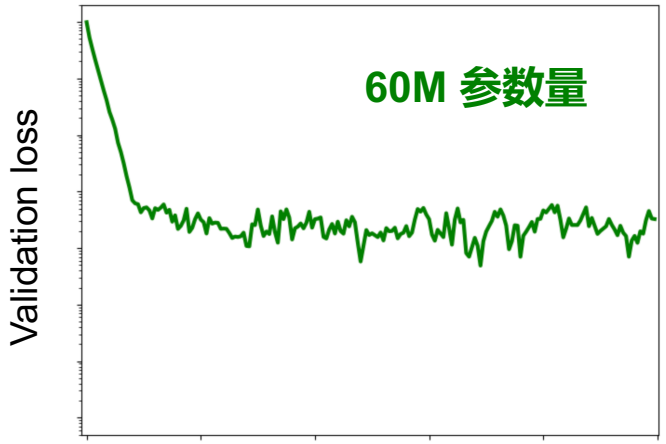
PART 00

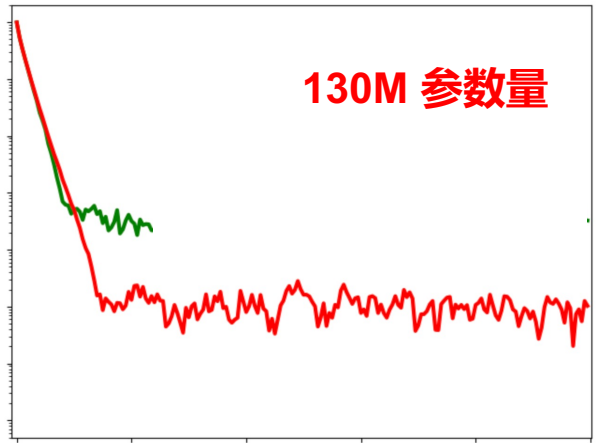
基本背景

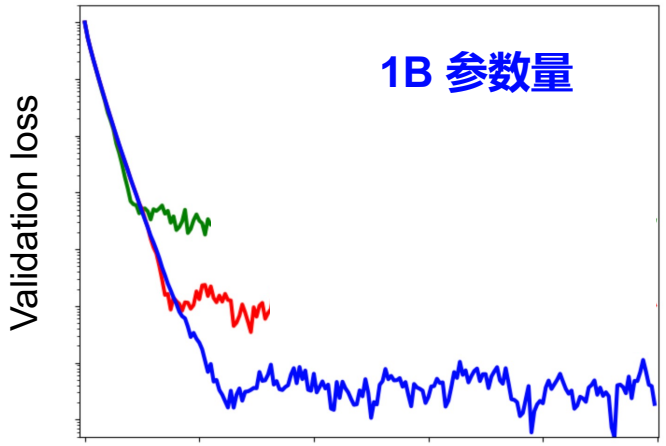
大模型的幂率法则

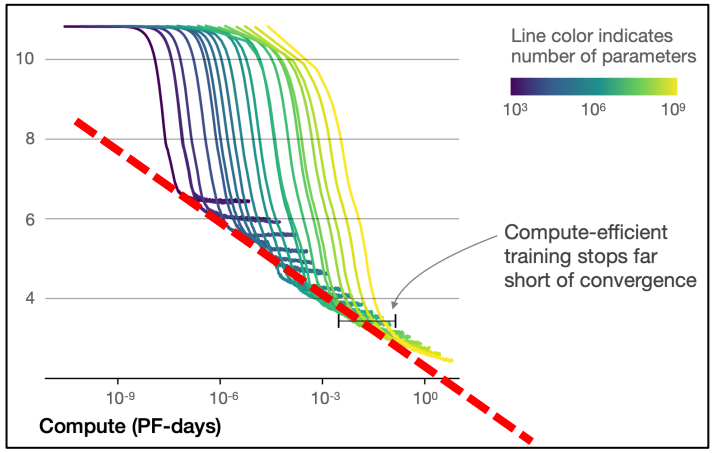


基于transformer架构的大模型性能随参数量的变化情况







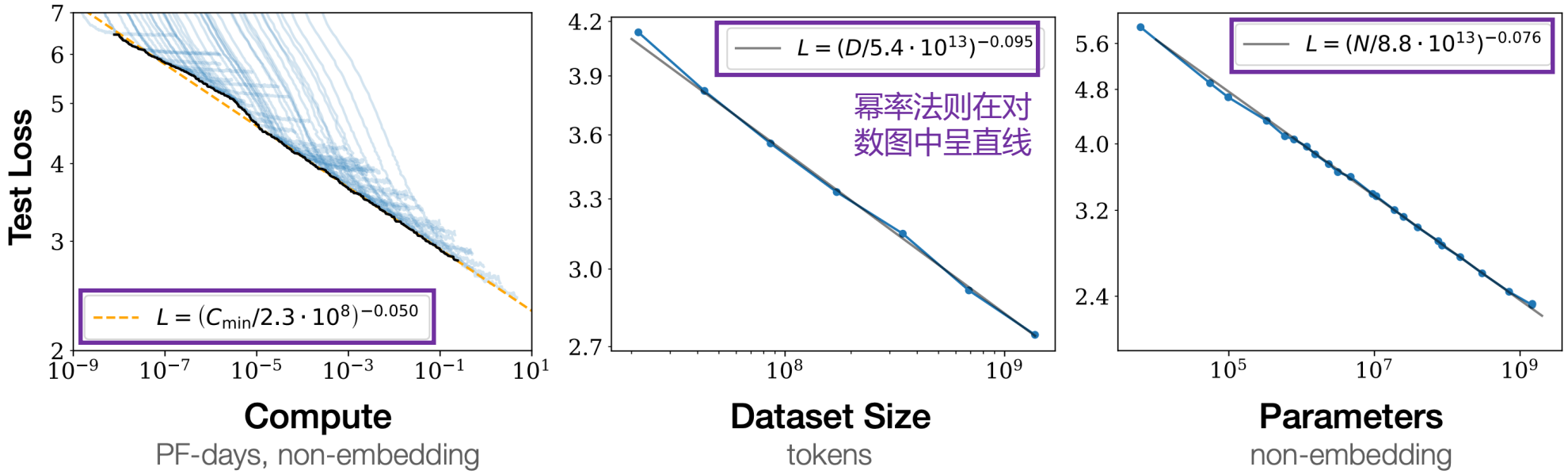


测试大量不同规格的模型

模型性能随着参数量的增长而不断改善

大模型的幂率法则

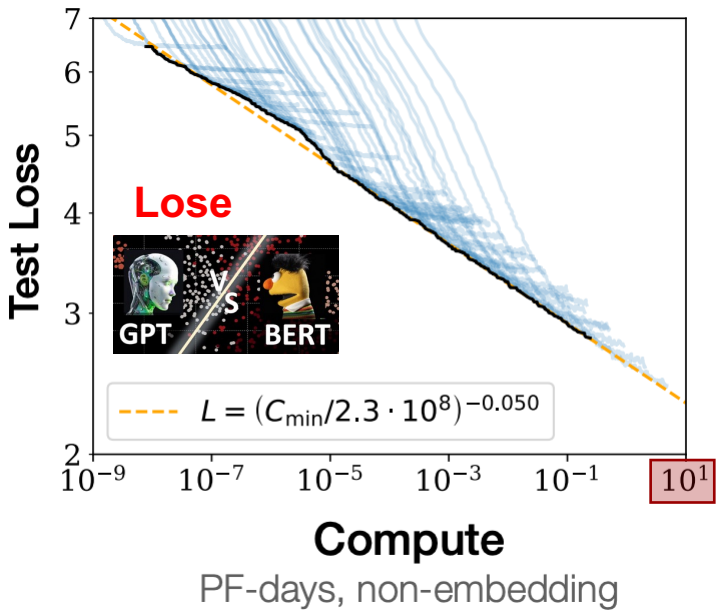
[OpenAI, Scaling Laws for Neural Language Models, 2020]



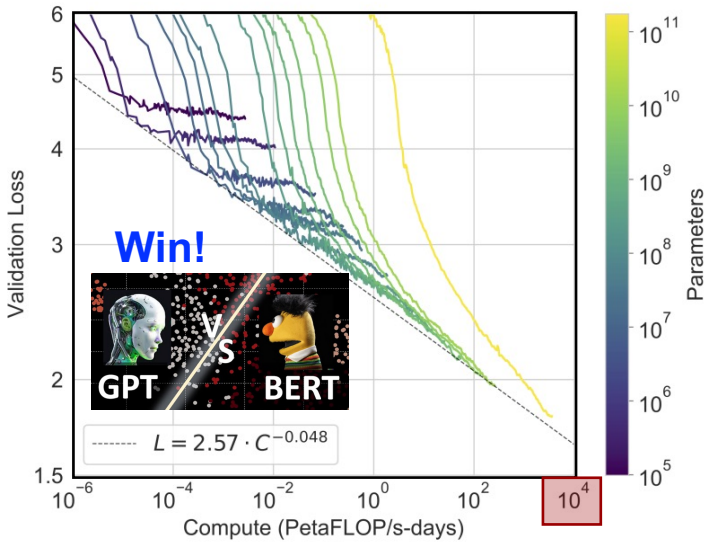
- 模型性能并不依赖于模型具体的架构或算法细节
- 模型性能只随着**数据量**、**参数量**、**计算量**的增大而不断改善
- 模型性能与数据量、参数量、计算量的数学关系可以被拟和成**幂率法则**

幂率法则是ChatGPT成功的基石之一

GPT-2,**1.5B**参数量,2019



GPT-3,**175B**参数量,2020

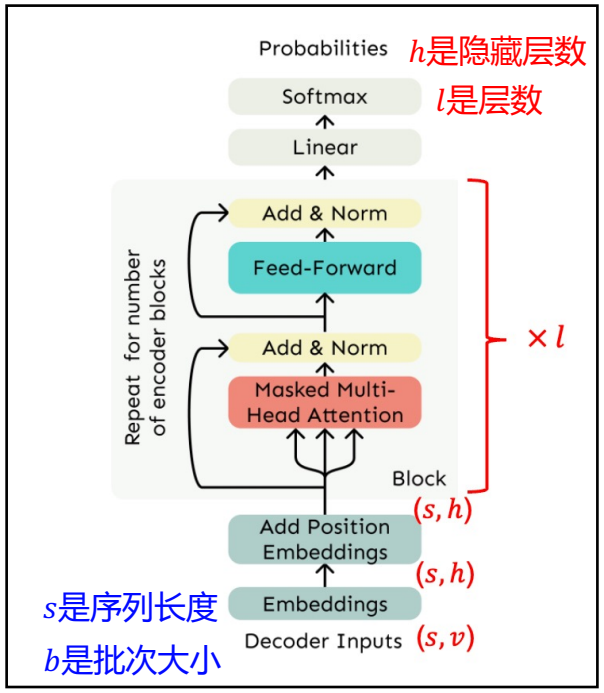


- GPT-2在2019年出现，虽然参数量远超Bert, 但实际性能却远不及Bert, 没有引起广泛关注
- OpenAI大力押注基于幂率法则，持续规模化参数，成功研制175B GPT-3模型，开启大模型时代
- 幂率法则已成为共识：**大模型** + **大数据** + **大算力** = **高智能**

幂率法则使大模型内存需求急剧增长

- 随着数据量D和参数量P的增大，大模型内存开销急剧增长
- 换言之，大模型性能提升的代价之一就是内存上的巨大开销

大模型内存 = **模型权重** + **梯度** + **优化器状态** + **激活值**

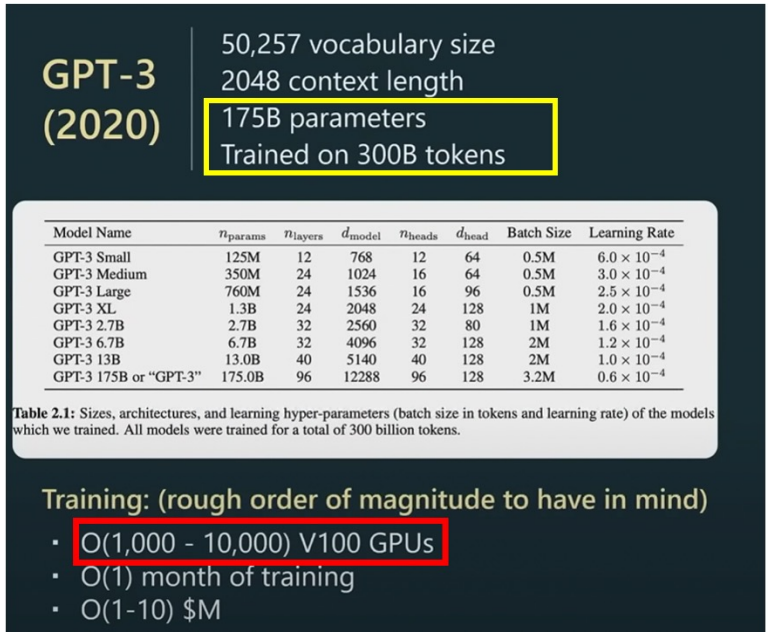


模型 + 梯度 + 优化器状态的内存开销：$\theta(lh^2) \sim \theta(P)$

- 与参数大小严格成正比
- **参数越多内存开销越大**

激活值的内存开销：$\Omega(bslh + bls^2a) \sim \Omega(D)$

- 至少与数据量成正比增长
- $D = bs$; **数据越多内存开销越大**



Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

幂率法则使大模型内存需求急剧增长

训练和推理成本显著上升

成本换算

内存急剧增长；超大集群联合训练

内存换算

模型参数量增大数据量增多

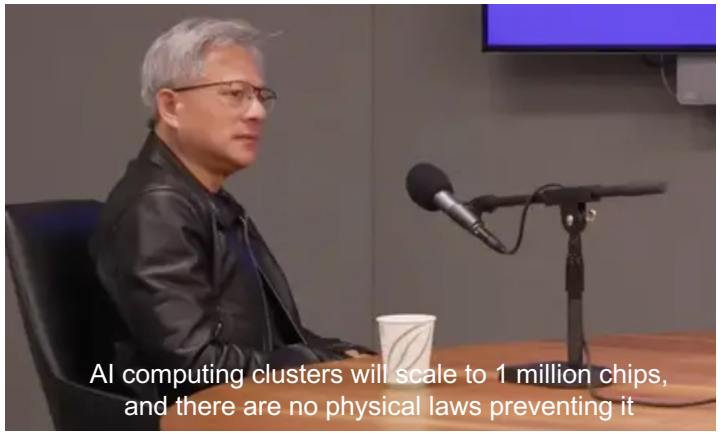
幂率法则

大模型性能提升的需求不断增长

LLaMA-3 2万卡集群

To train our largest Llama 3 models, we combined three types of parallelization: data parallelization, model parallelization, and pipeline parallelization. Our most efficient implementation achieves a compute utilization of over 400 TFLOPS per GPU when trained on 16K GPUs simultaneously. We performed training runs on two custom-built 24K GPU clusters. To maximize GPU uptime, we developed an advanced new training stack that automates error detection, handling, and maintenance. We also greatly improved our

XAI 搭建十万卡集群



黄仁勋：AI算力集群会扩展到100万芯片

关键问题：如何节省内存?

幂率法则使大模型内存需求急剧增长

训练和推理成本显著上升

成本换算

内存急剧增长；超大集群联合训练

内存换算

模型参数量增大数据量增多

幂率法则

大模型性能提升的需求不断增长

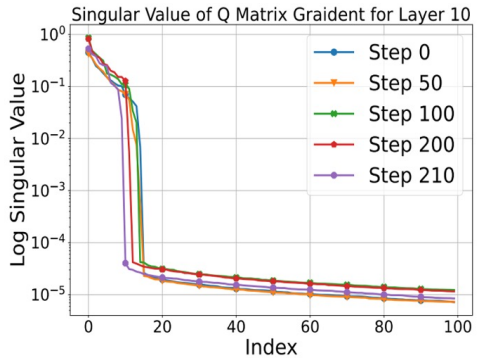
路线一：设计大模型架构，寻找全新的幂率准则

路线二：发展新硬件(如超节点)降低训推成本

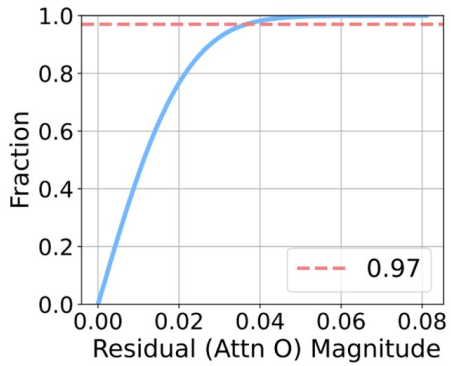
路线三：大模型结构驱动的内存高效训练新方法

洞察：随着数量的持续增加，大模型中存在大量冗余结构

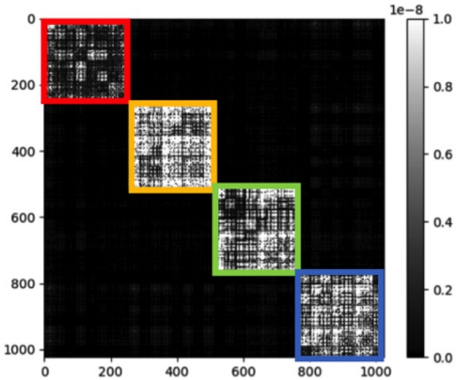
传统训练方法忽略模型特殊结构，导致内存开销巨大



大模型梯度具有低秩性
(适合低秩子空间存算)



FFN激活值具有稀疏性
(适合稀疏化Mask存算)



Hessian近似块状对角
(适合局部/全局解耦存算)

报告内容：解析大模型特殊结构，挖掘内存节省的红利

本次报告的主题：**如何挖掘大模型的特殊结构，设计内存节省的训练新方法**

❶	大模型梯度呈现低秩性	发展基于低秩性质的子空间投影训练算法 节省优化器状态内存，适用于稠密模型/MoE模型	(He-Yuan, ICML 2025; Chen-Yuan, ICML 2025;Chen-Yuan, ICLR 2025)	**优化器状态**
❷	前馈网络层呈现稀疏性	发展基于FFN稀疏特性的重要性采样训练算法 节省激活值内存，适用于稠密模型/MoE模型	(Song-Yuan, ICML 2025; Zhu-Yuan, NeurIPS 2024;He-Yuan, ICML 2024;)	**激活值**
❸	激活层呈现层间低秩性	发展基于层间低秩性的大模型参数高效训练算法 节省模型与梯度内存，适用于稠密模型/MoE模型	(Kong-Yuan, 2025; Wu-Yuan, 2025)	**模型和梯度**
❹	线性激活层呈现均匀性	发展基于参数分布特性的FP8/INT8混合精度训练算法	(华为稼先社区)	

北京大学
PEKING UNIVERSITY

PART 01

基于梯度低秩结构的子空间训练方法

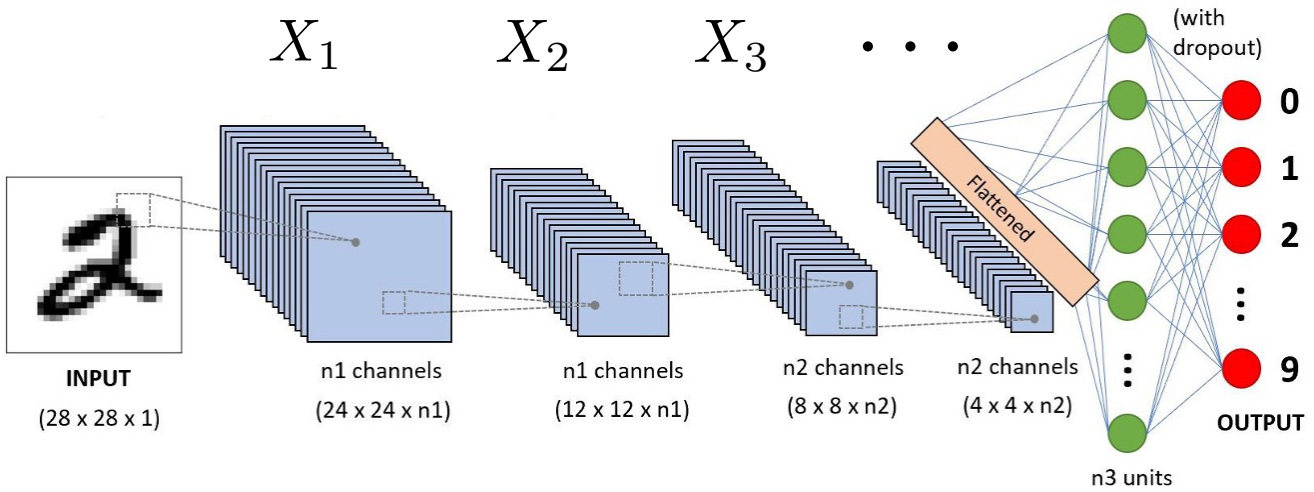
何雨桐 陈一鸣 张源 李澎瑞 胡奕鹏 陈楚岩

Y. He, P. Li, Y. Hu, C. Chen, K. Yuan, *Subspace Optimization for Large Language Models with Convergence Guarantees*, ICML 2025.

Y. Chen, Y. Zhang, Y. Liu, K. Yuan, Z. Wen, *A Memory Efficient Randomized Subspace Optimization Method for Training Large Language Models*, ICML 2025

大模型预训练的损失函数

- 神经网络的模型权重可以建模为一系列矩阵变量 $\boldsymbol{X} = \{X_\ell\}_{\ell=1}^{L}$



- 将语言模型建模为 $h(\boldsymbol{X};\xi)$；其中，ξ 是输入的token，$\hat{y} = h(\boldsymbol{X};\xi)$ 是预测出的token

大语言模型的数学建模：

$$\boldsymbol{X}^\star = \arg\min_{\boldsymbol{X}} \left\{ \mathbb{E}_{\xi\sim\mathcal{D}} \left[L(h(\boldsymbol{X};\xi), y) \right] \right\}$$

交叉熵

数据分布　预测token　实际token

预训练大模型的本质是求解随机优化问题

- 引入如下定义: $\boldsymbol{\xi} = (\xi, y)$, $F(\boldsymbol{X}; \boldsymbol{\xi}) = L(h(\boldsymbol{X}; \xi), y)$, 则大模型预训练可以建模为随机优化

带有矩阵变量的随机优化问题: $\boldsymbol{X}^\star = \arg\min_{\boldsymbol{X}} \left\{ \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{D}} \left[F(\boldsymbol{X}; \boldsymbol{\xi}) \right] \right\}$

其中，优化变量$\boldsymbol{X}$是矩阵变量。矩阵变量相比于向量变量具有更好的结构。

- Adam是预训练大模型的主流优化器

优化器状态

$$\boldsymbol{G}_t = \nabla F(\boldsymbol{X}_t; \boldsymbol{\xi}_t) \quad \text{(随机梯度)}$$

$$\boldsymbol{M}_t = (1-\beta_1)\boldsymbol{M}_{t-1} + \beta_1 \boldsymbol{G}_t \quad \text{(一阶动量)}$$

$$\boldsymbol{V}_t = (1-\beta_2)\boldsymbol{V}_{t-1} + \beta_2 \boldsymbol{G}_t \odot \boldsymbol{G}_t \quad \text{(二阶动量)}$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \frac{\gamma}{\sqrt{\boldsymbol{V}_t} + \epsilon} \odot \boldsymbol{M}_t \quad \text{(自适应SGD)}$$

大模型训练内存构成

大模型内存 = **模型参数** + **梯度** + **优化器状态**+ **激活值**

- 参数量为 **P** 的**模型**: **梯度**参数量为 **P**, **优化器状态**占用 **2P** 参数量; 合计 **4P 参数量.**

P $$\boldsymbol{G}_t = \nabla F(\boldsymbol{X}_t; \boldsymbol{\xi}_t)$$

2P
$$\boldsymbol{M}_t = (1-\beta_1)\boldsymbol{M}_{t-1} + \beta_1 \boldsymbol{G}_t$$
$$\boldsymbol{V}_t = (1-\beta_2)\boldsymbol{V}_{t-1} + \beta_2 \boldsymbol{G}_t \odot \boldsymbol{G}_t$$

P $$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \frac{\gamma}{\sqrt{\boldsymbol{V}_t} + \epsilon} \odot \boldsymbol{M}_t$$

优化器状态带来显著内存占用

大模型训练内存构成

大模型内存 = **模型参数** + **梯度** + **优化器状态**+ **激活值**

- 激活值是计算梯度时需要存储的辅助变量

考虑如下的线性神经网络

$$z_i = X_i z_{i-1}, \forall\, i = 1, \cdots, L$$
$$f = \mathcal{L}(z_L; y)$$

根据链式法则，梯度计算方式如下

$$\frac{\partial f}{\partial X_i} = \frac{\partial f}{\partial z_i} z_{i-1}^\top$$

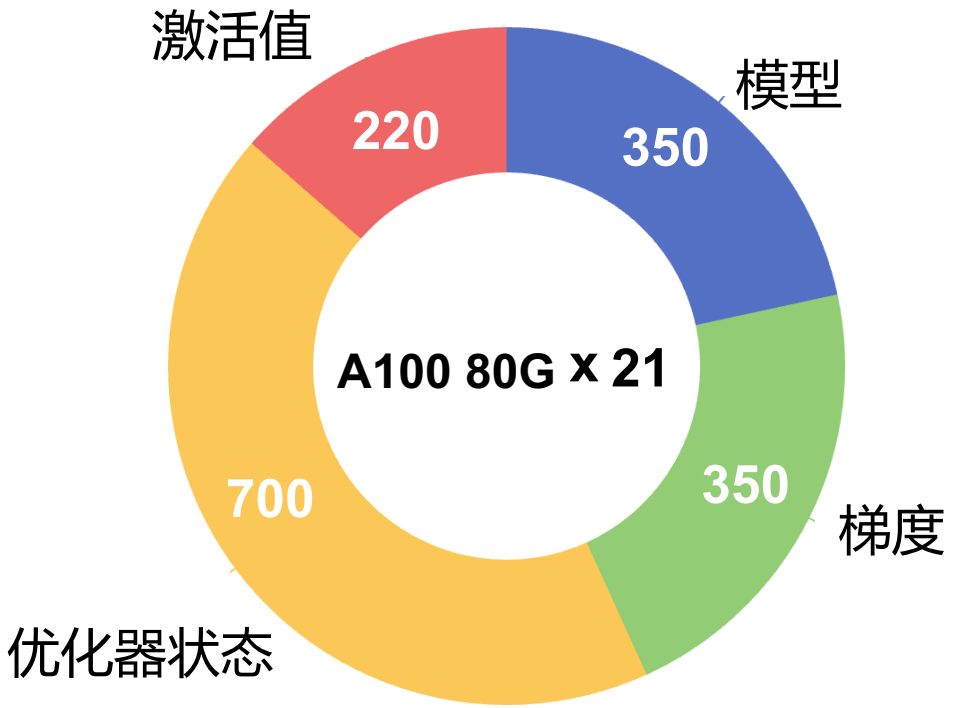
需要存储辅助变量 $z_1, z_2, \cdots, z_L$

- 激活值大小依赖于序列长度(sequence length)和批大小(batch size)

训练GPT-3的最低内存开销

- 使用BF16精度格式预训练GPT-3模型的最低内存开销如下(批次大小设为1)

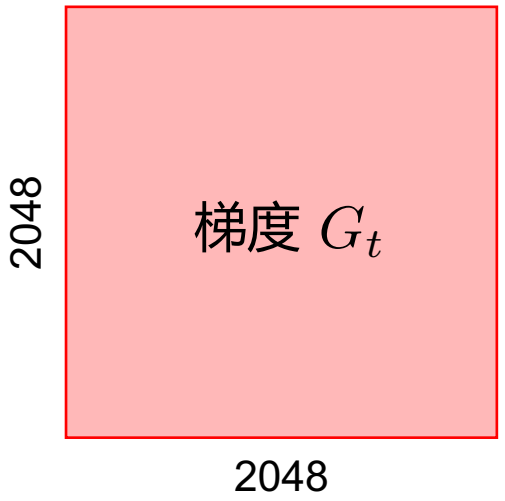
  - 参数量: 175B
  - 模型内存: 175B * 2 Bytes = 350 GB
  - 梯度内存: 350 GB
  - 优化器状态内存: 700 GB　(使用Adam优化器)
  - 激活值内存: ~220 GB
  - 共计: **1620 GB**

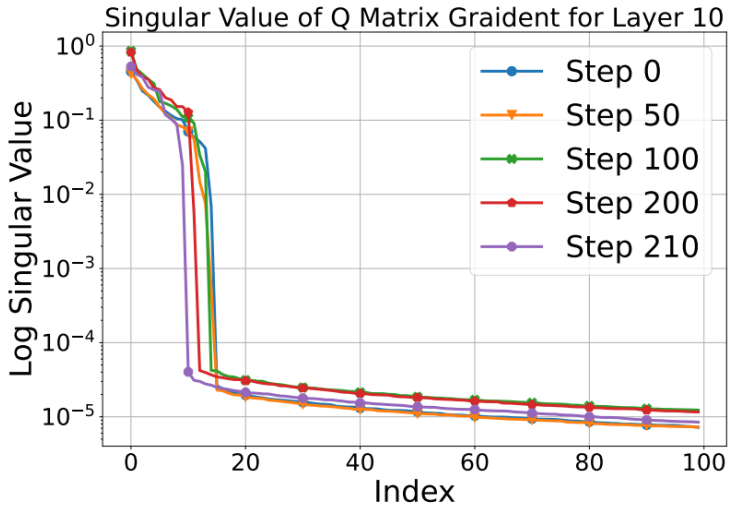


如何节省优化器状态内存?

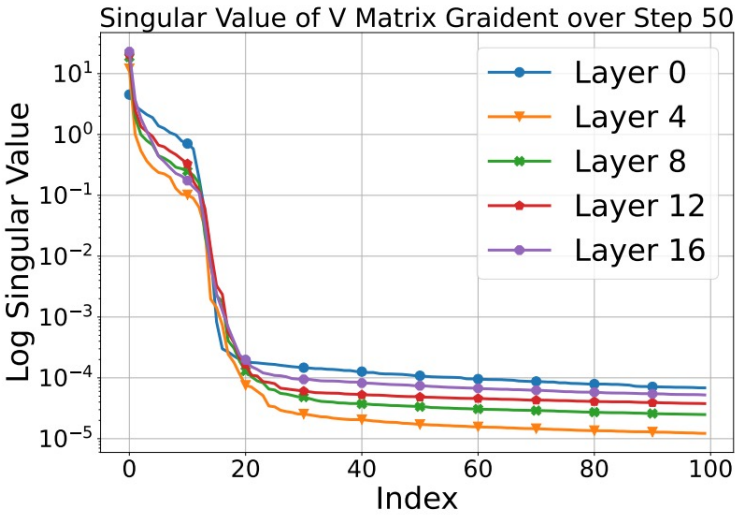
大模型结构：梯度低秩性

- 重要观察：训练过程中大模型的梯度矩阵具有**低秩性**[1]





不同训练步数保持低秩性



不同层保持低秩性

- 给定一个2048×2048维的梯度矩阵, 约**前10个**奇异值占主导；**梯度矩阵的秩约为10**
- 如何利用梯度矩阵的低秩结构?

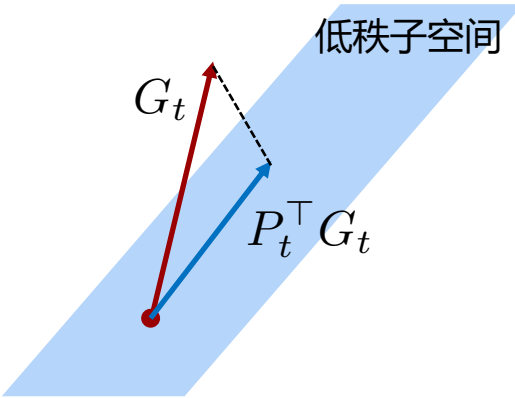
[1] Yiming Chen, Kun Yuan, et. al., *Enhancing Zeroth-Order Fine-tuning for Language Models with Low-Rank Structures*, ICLR 2025

GaLore: 梯度低秩投影训练算法

- 核心思想: 将梯度投影至低秩子空间[1]
- 给定梯度 $G_t \in \mathbb{R}^{m\times n}$ 和投影矩阵 $P_t \in \mathbb{R}^{m\times r}$ ，我们将高维梯度投影至低秩子空间

$$G_t \in \mathbb{R}^{m\times n} \quad P_t^\top G_t \quad g_t = P_t^\top G_t \in \mathbb{R}^{r\times n}$$

低秩投影



- 当子空间的秩 $r \ll m$, 低秩梯度的维度 g_t 远远小于原始梯度 G_t
- 实际情况中，m的量级是千或万，r的量级是十或百；当梯度 G_t 低秩时，g_t 是很好的近似梯度

[1] J. Zhao, et. al., *Galore: Memory-efficient LLM training by gradient low-rank projection,* ICML 2024

利用低秩投影梯度更新优化器状态

- 由于低秩梯度是原始梯度的有效近似，可以利用低秩梯度更新优化器状态:

$$
\begin{aligned}
\boldsymbol{g}_t &= \boldsymbol{P}_t^\top \boldsymbol{G}_t && \triangleright \ \text{r x n 维} \\
\boldsymbol{m}_t &= (1-\beta_1)\boldsymbol{m}_{t-1} + \beta_1 \boldsymbol{g}_t && \triangleright \ \text{r x n 维} \\
\boldsymbol{v}_t &= (1-\beta_2)\boldsymbol{v}_{t-1} + \beta_2 \boldsymbol{g}_t \odot \boldsymbol{g}_t && \triangleright \ \text{r x n 维} \\
\boldsymbol{\delta}_t &= \frac{\gamma}{\sqrt{\boldsymbol{v}_t} + \epsilon} \odot \boldsymbol{m}_t && \triangleright \ \text{r x n 维}
\end{aligned}
$$

简记为

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + \boldsymbol{P}_t \boldsymbol{\rho}(\boldsymbol{P}_t^\top \boldsymbol{G}_t)$$

- 参数更新:

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \boldsymbol{P}_t \boldsymbol{\delta}_t \qquad \triangleright \ \text{m x n 维}$$

- 内存开销: 模型参数 $\boldsymbol{X}$，梯度 $\boldsymbol{G}$，投影矩阵 $\boldsymbol{P}$，优化器状态 $\boldsymbol{m}, \boldsymbol{v}$ 以及激活值

内存开销小，将优化器状态从mn降低为mr

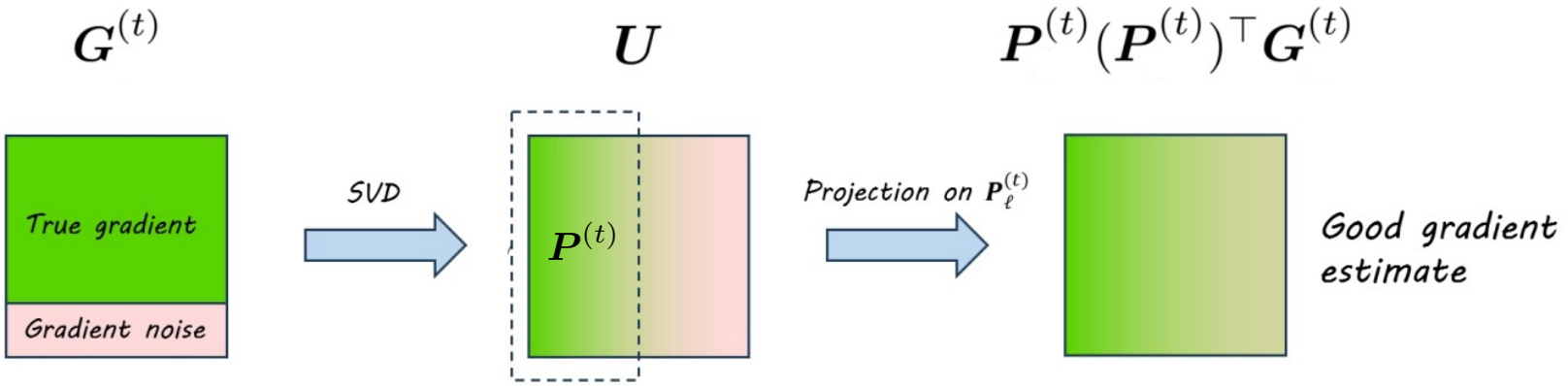
J. Zhao, et. al., *Galore: Memory-efficient LLM training by gradient low-rank projection,* ICML 2024

利用低秩投影更新优化器状态

- 低秩投影训练算法: $\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + \boldsymbol{P}_t \rho(\boldsymbol{P}_t^\top \boldsymbol{G}_t)$
- 如何得到投影矩阵? **SVD分解!**

$$G_t = U\Sigma V^\top \longrightarrow P_t = \boxed{U[:, :r]} \in \mathbb{R}^{m\times r}$$

选择起主导作用的前r列



Yutong He, Kun Yuan, et. al., *Subspace Optimization for Large Language Models with Convergence Guarantees*, ICML 2025

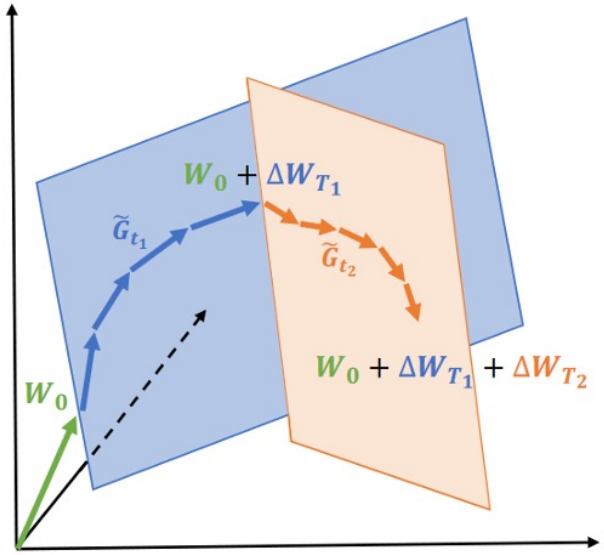
利用低秩投影更新优化器状态

- SVD的计算非常昂贵，难以每次迭代时均使用SVD分解
- 惰性SVD分解: 每τ次迭代中，更新一次SVD；其余迭代使用相同的投影矩阵[1]

基于惰性SVD分解的低秩投影训练算法

$$\begin{cases} \boldsymbol{P}_t \leftarrow \mathrm{SVD}(\boldsymbol{G}_t) & \text{if } t \bmod \tau = 0 \\ \boldsymbol{P}_t \leftarrow \boldsymbol{P}_{t-1} & \text{otherwise} \end{cases}$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + \boldsymbol{P}_t \boldsymbol{\rho}(\boldsymbol{P}_t^\top \boldsymbol{G}_t)$$



- 由于每 τ 步做一次SVD分解，计算开销被均摊，从而使得SVD的计算量变小

J. Zhao, et. al., *Galore: Memory-efficient LLM training by gradient low-rank projection,* ICML 2024

GaLore: 梯度低秩投影训练算法

Pretraining LLaMA on C4 dataset [1]

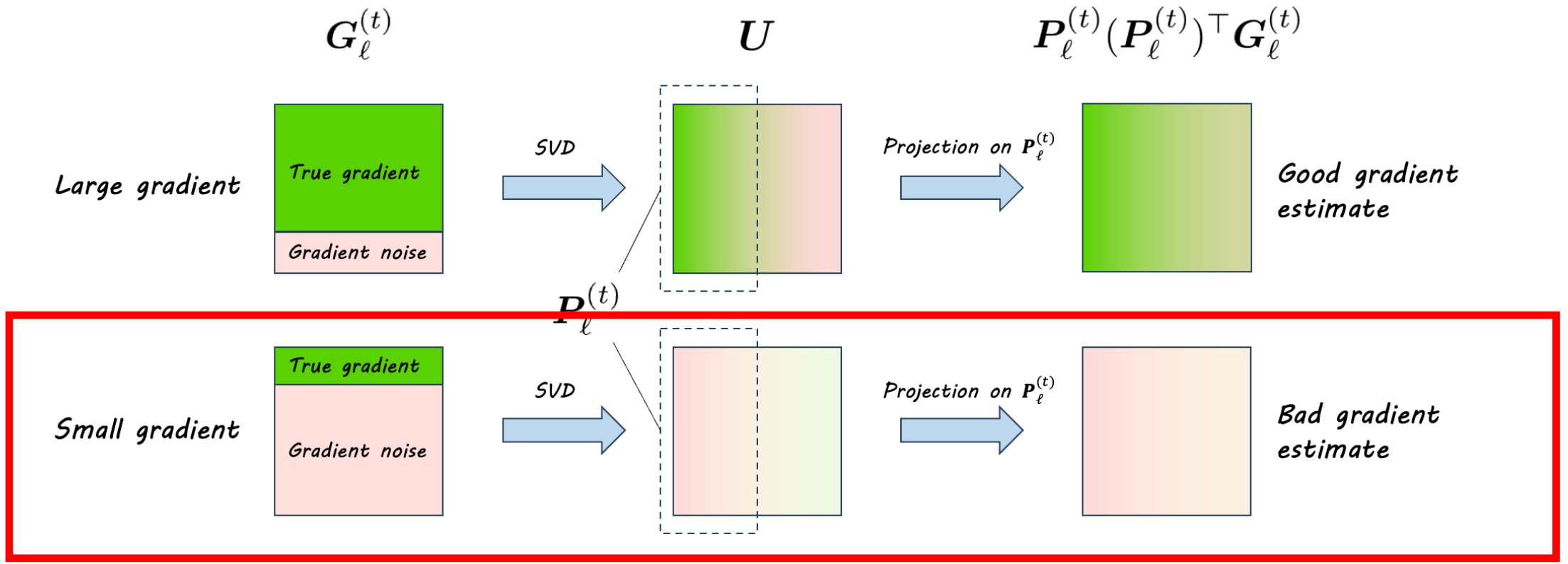
	60M	130M	350M	1B
Full-Rank	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56 (7.80G)
GaLore	**34.88** (0.24G)	**25.36** (0.52G)	**18.95** (1.22G)	**15.64** (4.38G)
Low-Rank	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53 (3.57G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21 (6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33 (6.17G)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

[1] J. Zhao, et. al., *Galore: Memory-efficient LLM training by gradient low-rank projection,* ICML 2024

GaLore算法在0.5%的性能损失以内，实现了内存节省

GaLore算法是否具有理论保证？是否总是能收敛至局部最优值(或者是稳定点)?

GaLore并非总是收敛！SVD投影带来算法收敛的隐患

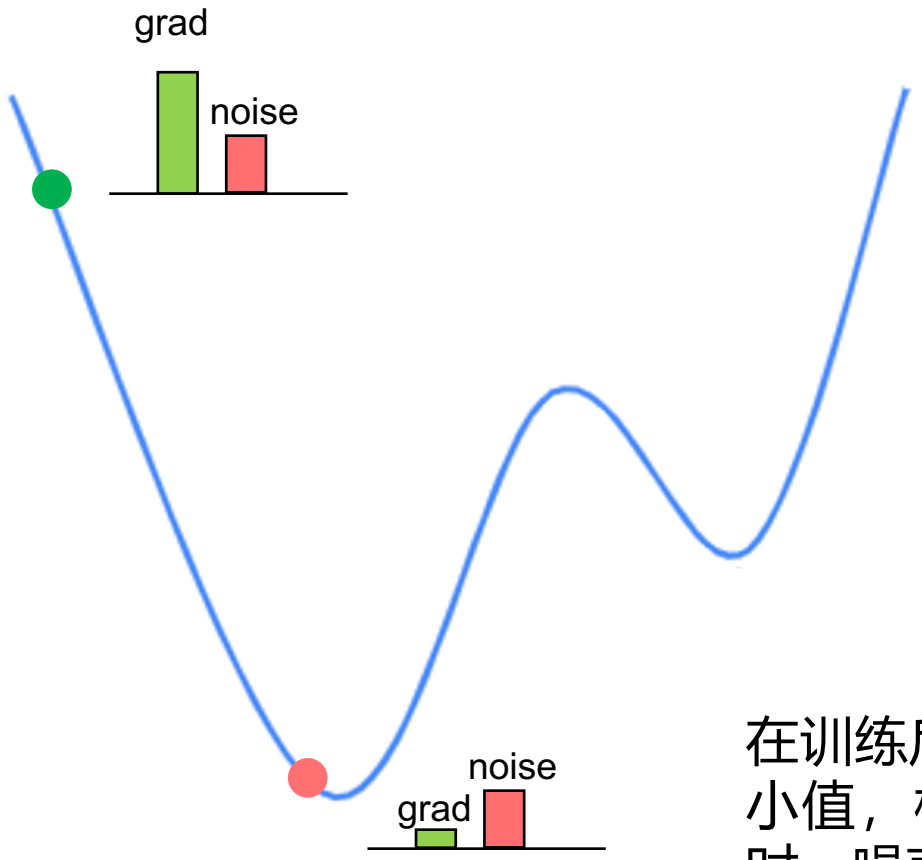


当梯度噪声占主导时，SVD分解得到的投影矩阵只能落在噪声主导的子空间

所有的梯度信息此时将会丢失，这说明低秩投影训练具有不收敛的风险

噪声占主导的情形是否经常出现？是的！

在训练初期，梯度将占主导



在训练后期，由于非常接近局部极小值，梯度信息非常接近于0。此时，噪声将占主导

SVD 投影的理论缺陷：反例构造

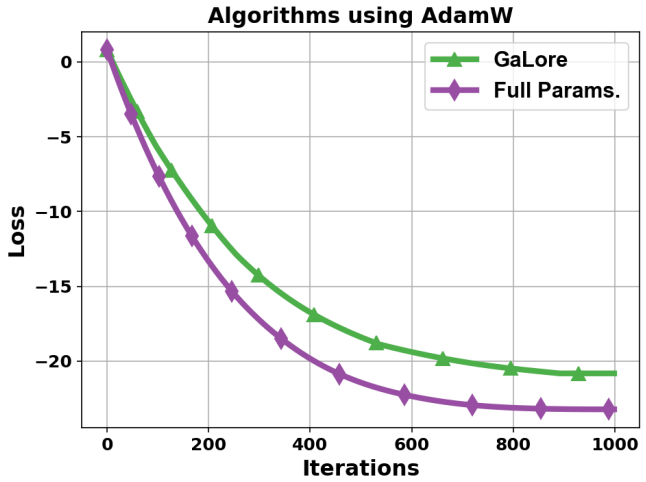
Counter-Example. We consider the following quadratic problem with gradient noise:

$$f(\boldsymbol{X}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{X}\|_F^2 + \langle \boldsymbol{B}, \boldsymbol{X}\rangle_F, \quad \nabla F(\boldsymbol{X};\xi) = \nabla f(\boldsymbol{X}) + \xi\sigma\boldsymbol{C}, \tag{1}$$

where $\boldsymbol{A} = (\boldsymbol{I}_{n-r} \quad 0) \in \mathbb{R}^{(n-r)\times n}$, $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{D} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{D} \in \mathbb{R}^{(n-r)\times(n-r)}$ generated randomly, $\boldsymbol{C} = \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{I}_r \end{pmatrix} \in \mathbb{R}^{n\times n}$, ξ is a random variable uniformly sampled from $\{1, -1\}$ per iteration, and σ is used to control the gradient noise.

Theorem (Non-convergence of GaLore): There exists an objective function $f : \mathbb{R}^d \to \mathbb{R}$ satisfying Assumptions 1 and 2, a stochastic gradient oracle (F, D) satisfying Assumption 3, an initial point $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$, and a constant $\epsilon_0 > 0$ such that for any rank $r_\ell < \min\{m_\ell, n_\ell\}$, subspace changing frequency τ, any optimizer ρ that inputs a subspace gradient of shape $r_\ell \times n_\ell$ and outputs a subspace update direction of the same shape, and for any $t > 0$, it holds that

$$\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2 \geq \epsilon_0.$$



GaLore**无法**精确收敛

[1] Yutong He, Kun Yuan et al., *Subspace Optimization for Large Language Models with Convergence Guarantees*, ICML 2025

GoLore:基于随机均匀投影的子空间训练算法

- Gradient random Low-rank projection **(GoLore)** 将梯度投影至随机

$$P_t \sim \mathcal{U}(\mathrm{St}_{m,r})$$

Lemma 5 (Error of GoLore's projection). *Let* $\boldsymbol{P} \sim \mathcal{U}(\mathrm{St}_{m,r})$, $\boldsymbol{Q} \sim \mathcal{U}(\mathrm{St}_{n,r})$, *it holds for all* $\boldsymbol{G} \in \mathbb{R}^{m\times n}$ *that*

$$\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^\top] = \frac{r}{m}\cdot \boldsymbol{I}, \quad \mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^\top] = \frac{r}{n}\cdot \boldsymbol{I},$$

and

$$\mathbb{E}[\|\boldsymbol{P}\boldsymbol{P}^\top \boldsymbol{G} - \boldsymbol{G}\|_F^2] = \left(1-\frac{r}{m}\right)\|\boldsymbol{G}\|_F^2, \quad \mathbb{E}[\|\boldsymbol{G}\boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{G}\|_F^2] = \left(1-\frac{r}{n}\right)\|\boldsymbol{G}\|_F^2.$$

$$\mathbb{E}[PP^\top G] = \mathbb{E}[PP^\top]\cdot\mathbb{E}[G] = \frac{r}{m}\nabla F(X)$$

低秩投影后的梯度在期望意义下就是真正的梯度；即便在大噪声下梯度信息也不会丢失

GoLore算法的收敛保证

Theorem (Convergence rate of GoLore): Under Assumptions 1-3, for any $T \geq 2 + 128/(3\underline{\delta}) + (128\sigma)^2/(9\sqrt{\underline{\delta}}L\Delta)$, GoLore using small-batch stochastic gradients and MSGD with MP converges as

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2\right] = \mathcal{O}\left(\frac{L\Delta}{\underline{\delta}^{5/2}T} + \sqrt{\frac{L\Delta\sigma^2}{\underline{\delta}^{7/2}T}}\right),$$

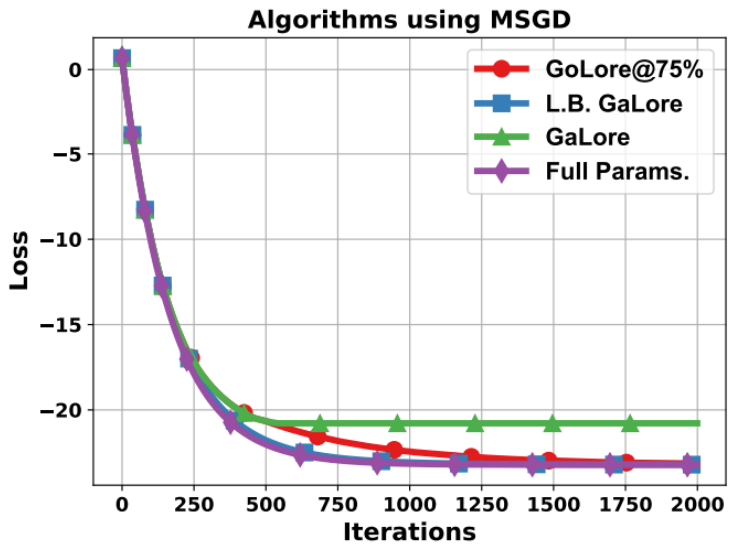
where $\Delta = f(\boldsymbol{x}^{(0)}) - \inf_{\boldsymbol{x}} f(\boldsymbol{x})$ and $\underline{\delta} := \min_\ell \frac{r_\ell}{\min\{m_\ell, n_\ell\}}$.

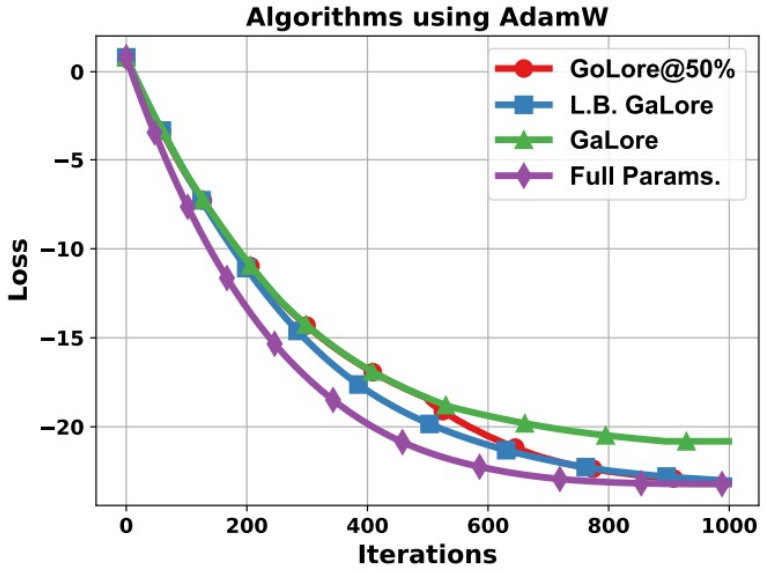
- GoLore可以保证收敛，其收敛速度是$\mathcal{O}(1/\sqrt{T})$.
- Adam的收敛速度也是 $\mathcal{O}(1/\sqrt{T})$ ，这说明梯度低秩投影不会影响收敛速度的阶次

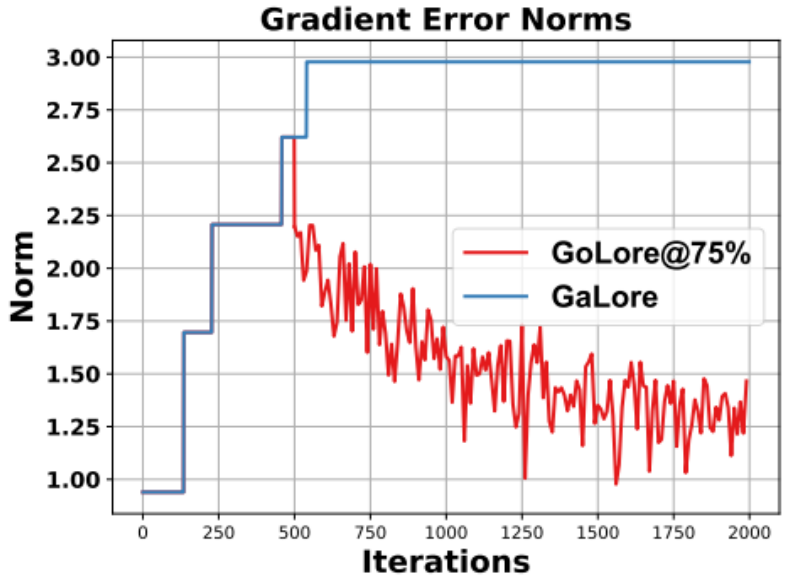
随机投影与SVD投影的混合使用

- 在算法收敛前期，梯度信息起主导作用。此时SVD投影可以有效捕捉重要的子空间
- 在算法收敛后期，噪声起主导作用。此时用随机均匀投影可以有效捕捉梯度，避免湮没在噪声中

GoLore@x% = GaLore (first (100-x)% iters) + GoLore (last x% iters)

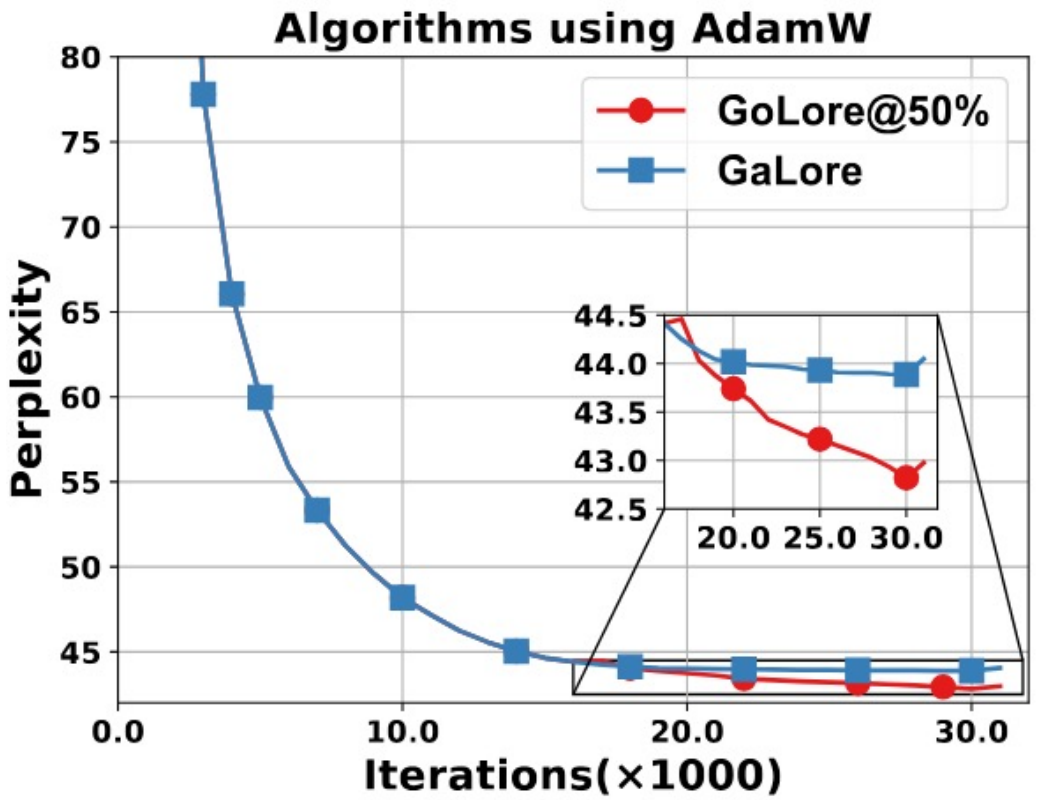




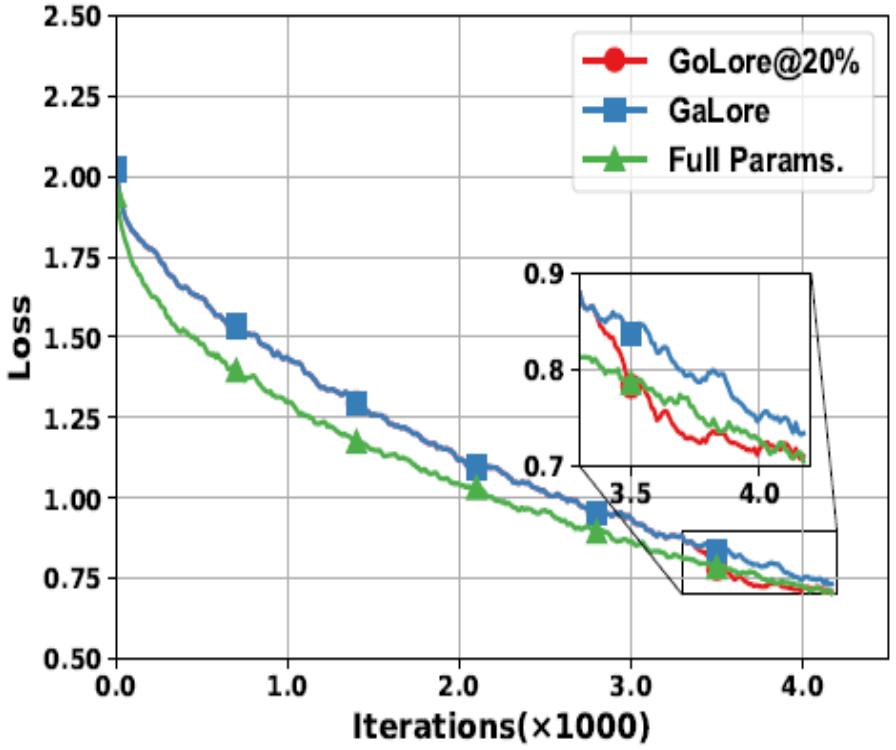


随机投影显著抑制收敛误差

Pre-train LLaM2-60M on C4



Fine-tuning LLaMA2-7B on WinoGrande:



GoLore在GLUE任务集中的benchmark

北京大学 PEKING UNIVERSITY

- 在GLUE任务集中微调RoBERTa-BASE模型:

Algorithm	CoLA	STS-B	MRPC	RTE	SST2	MNLI	QNLI	QQP	Avg
Full Params.	62.07	90.18	92.25	78.34	94.38	87.59	92.46	91.90	86.15
GaLore	61.32	90.24	92.55	77.62	**94.61**	86.92	92.06	90.84	85.77
GoLore@20%	**61.66**	**90.55**	**92.93**	**78.34**	**94.61**	**87.02**	**92.20**	**90.91**	**86.03**

- 由于在后20%使用了随机投影，避免了梯度信息的丢失，GoLore算法在所有任务中均呈现持平甚至更加的性能表现

GoLore/RSO在Pretrain中的benchmark

北京大学 PEKING UNIVERSITY

LLaMA Pretrain

Algorithm	60M	130M	350M	1B
Adam*	34.06 (0.22G)	25.08 (0.50G)	18.80 (1.37G)	15.56 (4.99G)
GaLore*	34.88 (0.14G)	25.36 (0.27G)	18.95 (0.49G)	**15.64** (1.46G)
LoRA*	34.99 (0.16G)	33.92 (0.35G)	25.58 (0.69G)	19.21 (2.27G)
ReLoRA*	37.04 (0.16G)	29.37 (0.35G)	29.08 (0.69G)	18.33 (2.27G)
RSO/GoLore	**34.55**(0.14G)	**25.34** (0.27G)	**18.86** (0.49G)	15.68 (1.46G)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens (B)	1.1	2.2	6.4	13.1

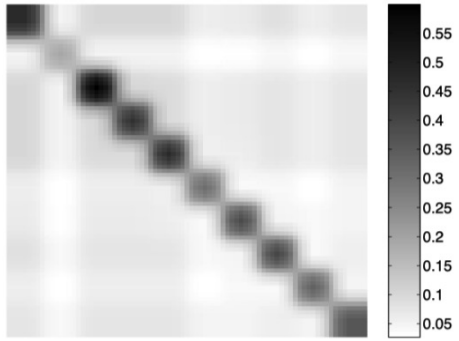
PPL损失0.77%　　优化器节省70.7%

[1] Yiming Chen, Kun Yuan et al., *A Memory Efficient Randomized Subspace Optimization Method For Training Large Language Models*, ICML 2015

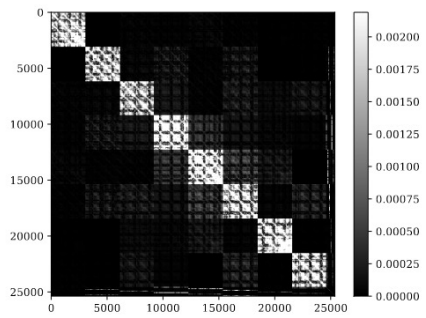
一些最新进展: Adam-mini[1]

$$G_t = \nabla F(X_t; \xi_t) \qquad V_t = (1-\beta_2)V_{t-1} + \beta_2 G_t \odot G_t$$

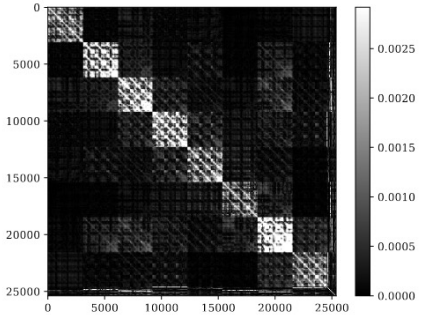
$$M_t = (1-\beta_1)M_{t-1} + \beta_1 G_t \qquad X_{t+1} = X_t - \frac{\gamma}{\sqrt{V_t}+\epsilon} \odot M_t$$



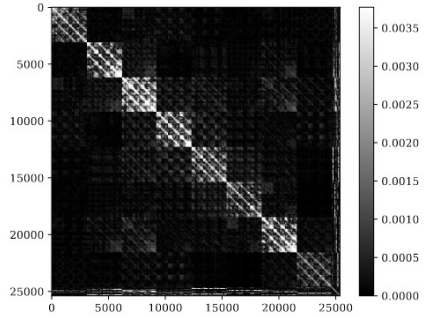
(a) Hessian of a MLP (Collobert, 2004)



(b) Hessian of a MLP at initialization

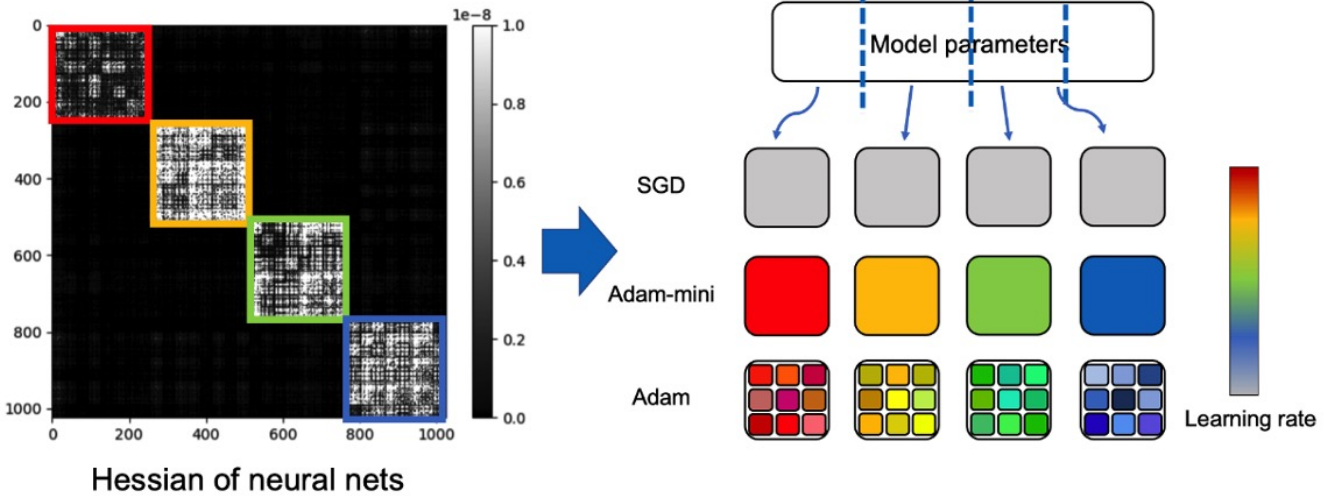


(c) Hessian of a MLP at 50% step



(d) Hessian of a MLP at 100% step

大模型的Hessian矩阵呈现**近似分块块对角**结构: 块内参数联系紧密，块之间参数相对独立



Hessian of neural nets

Adam: 每个**参数**都有一个学习率

Adam-mini: 每个**参数块**共享一个学习率

$$v = (1-\beta_2) * \mathrm{mean}(g \circ g) + \beta_2 * v$$

显著节省二阶动量的内存

[1] Yushun Zhang et. al., ADAM-MINI: USE FEWER LEARNING RATES TO GAIN MORE, ICLR 2025

一些最新进展: Adam-mini

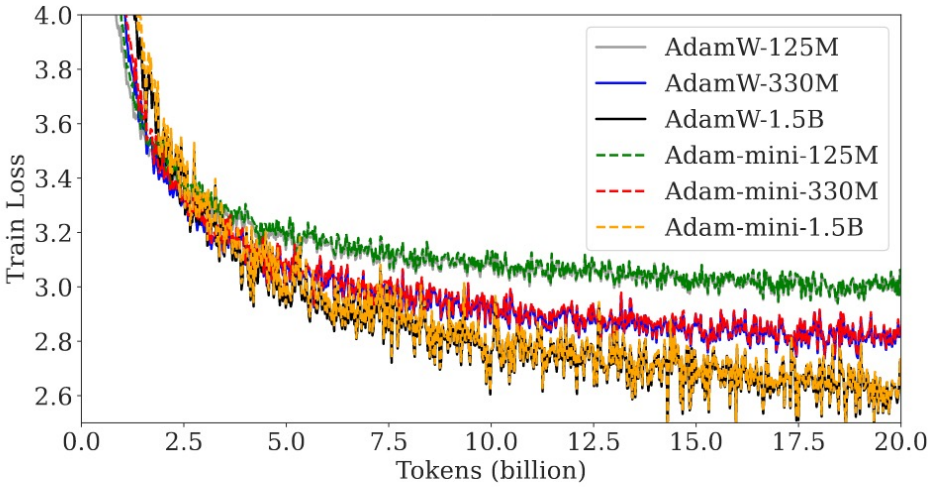
Algorithm 1 Adam-mini (General form)

1: Input weight-decay coefficient λ and current step t
2: Partition params into `param_blocks` by **Principle 1** in Section 2.3
3: **for** `param` in `param_blocks` **do**
4: $\quad g = \texttt{param.grad}$
5: $\quad \texttt{param} = \texttt{param} - \eta_t * \lambda * \texttt{param}$
6: $\quad m = (1-\beta_1) * g + \beta_1 * m$
7: $\quad \hat{m} = \frac{m}{1-\beta_1^t}$
8: $\quad v = (1-\beta_2) * \texttt{mean}(g \odot g) + \beta_2 * v$
9: $\quad \hat{v} = \frac{v}{1-\beta_2^t}$
10: $\quad \texttt{param} = \texttt{param} - \eta_t * \frac{\hat{m}}{\sqrt{\hat{v}}+\epsilon}$
11: **end for**

- For Adam: $u_{\text{Adam}} = \left(\frac{\eta}{\sqrt{v_1}}, \frac{\eta}{\sqrt{v_2}}, \frac{\eta}{\sqrt{v_3}}, \frac{\eta}{\sqrt{v_4}}, \frac{\eta}{\sqrt{v_5}}\right)$.
- For Adam-mini: suppose the partition is $(1,2,3)$ and $(4,5)$ then

$$u_{\text{mini}} = \left(\frac{\eta}{\sqrt{(v_1+v_2+v_3)/3}}, \frac{\eta}{\sqrt{(v_1+v_2+v_3)/3}}, \frac{\eta}{\sqrt{(v_1+v_2+v_3)/3}}, \frac{\eta}{\sqrt{(v_4+v_5)/2}}, \frac{\eta}{\sqrt{(v_4+v_5)/2}}\right).$$

Model	Optimizer	Memory (GB)
GPT-2-1.5B	AdamW	12.48
GPT-2-1.5B	Adam-mini	6.24 (**50%** ↓)
Llama 2-1B	AdamW	8.80
Llama 2-1B	Adam-mini	4.40 (**50%** ↓)
Llama 2-7B	AdamW	53.92
Llama 2-7B	Adam-mini	26.96 (**50%** ↓)
Llama 3-8B	AdamW	64.24
Llama 3-8B	Adam-mini	32.12 (**50%** ↓)
Llama 2-13B	AdamW	104.16
Llama 2-13B	Adam-mini	52.08 (**50%** ↓)



(a) GPT-2 series

一些最新进展: Apollo[1]

北京大学 PEKING UNIVERSITY

- Apollo : SGD-like Memory, AdamW-level Performance
- APOLLO 列共享学习率 / APOLLO-Mini 矩阵共享学习率： $\boldsymbol{S} = \mathrm{diag}(s_0, s_1, \cdots, s_n)$ $\left\{ s_j = \frac{\|\tilde{\boldsymbol{g}}_t[:, j]\|_2}{\|\boldsymbol{g}_t[:, j]\|_2} \right\}$

$$\tilde{\boldsymbol{G}}_t = \boldsymbol{G}_t \boldsymbol{S} \qquad \boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta \cdot \tilde{\boldsymbol{G}}_t$$

(无需依赖优化器状态)

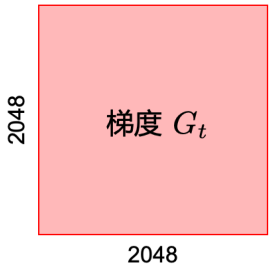
Pre-trian LLaMA on C4

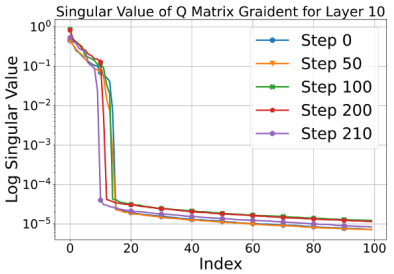
Methods	60M		130M		350M		1B	
	Perplexity	Memory	Perplexity	Memory	Perplexity	Memory	Perplexity	Memory
AdamW♠	34.06	0.36G	25.08	0.76G	18.80	2.06G	15.56	7.80G
Low-Rank♠	78.18	0.26G	45.51	0.54G	37.41	1.08G	142.53	3.57G
LoRA♠	34.99	0.36G	33.92	0.80G	25.58	1.76G	19.21	6.17G
ReLoRA♠	37.04	0.36G	29.37	0.80G	29.08	1.76G	18.33	6.17G
GaLore♠	34.88	0.24G	25.36	0.52G	18.95	1.22G	15.64	4.38G
Fira	**31.06**	0.24G	**22.73**	0.52G	17.03	1.22G	14.31	4.38G
APOLLO *w.* SVD	31.26	0.24G	22.84	0.52G	**16.67**	1.22G	**14.10**	4.38G
APOLLO	31.55	0.24G	22.94	0.52G	**16.85**	1.22G	**14.20**	4.38G
APOLLO †	31.26	0.18G	23.18	0.39G	**16.98**	0.95G	**14.25**	3.49G
APOLLO-Mini	31.93	0.12G	23.84	0.25G	17.18	0.69G	**14.17**	2.60G

[1] Hanqing Zhu, Jinwon Lee et al., *APOLLO: SGD-like Memory, AdamW-level Performance*, MLSys 2025

单元小结

- 大模型内存由参数、梯度、**优化器状态**、激活值四部分组成
- 大模型梯度有显著的低秩结构
- 新算法核心思想：利用低秩梯度来计算优化器状态，从而节省优化器状态
- 低秩子空间投影方法：随机Stiefel流形投影
- 收益总结：在1%性能损失以内，可以节省70%左右的优化器状态内存

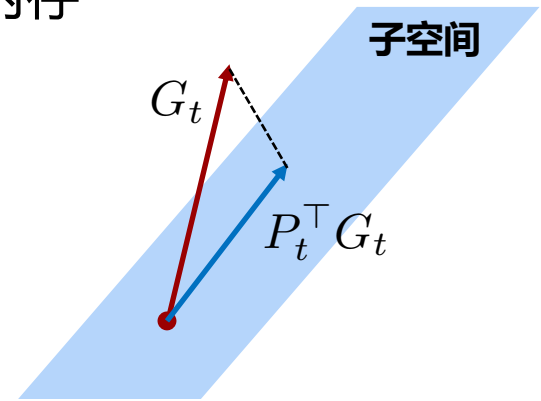




不同训练步数保持低秩性

$$G_t \in \mathbb{R}^{m\times n} \quad P_t^\top G_t \quad g_t = P_t^\top G_t \in \mathbb{R}^{r\times n}$$

低秩投影



北京大学
PEKING UNIVERSITY

PART 02

基于FFN稀疏特性的重要性采样训练方法

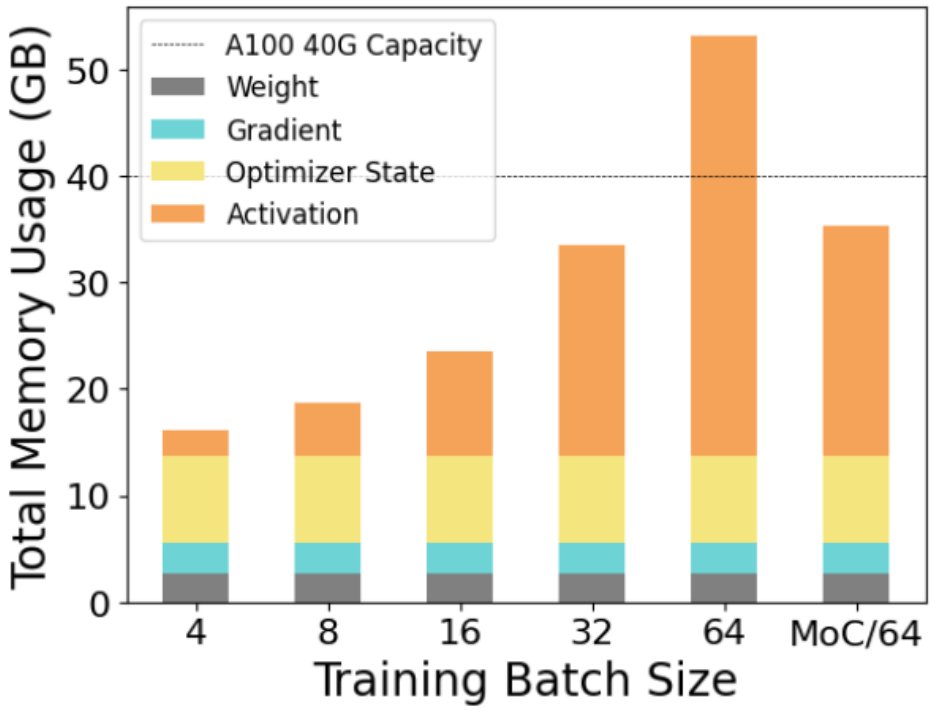
吴童 何雨桐

T. Wu, Y. He, B. Wang and K. Yuan, *Mixture-of-Channels: Exploiting Sparse FFNs for Efficient LLMs Pre-Training and Inference,* Submitted to NeurIPS 2025

大模型激活值内存

大模型内存 = **模型参数** + **梯度** + **优化器状态**+ **激活值**

LLaMA-2的内存占用(文本长度256)



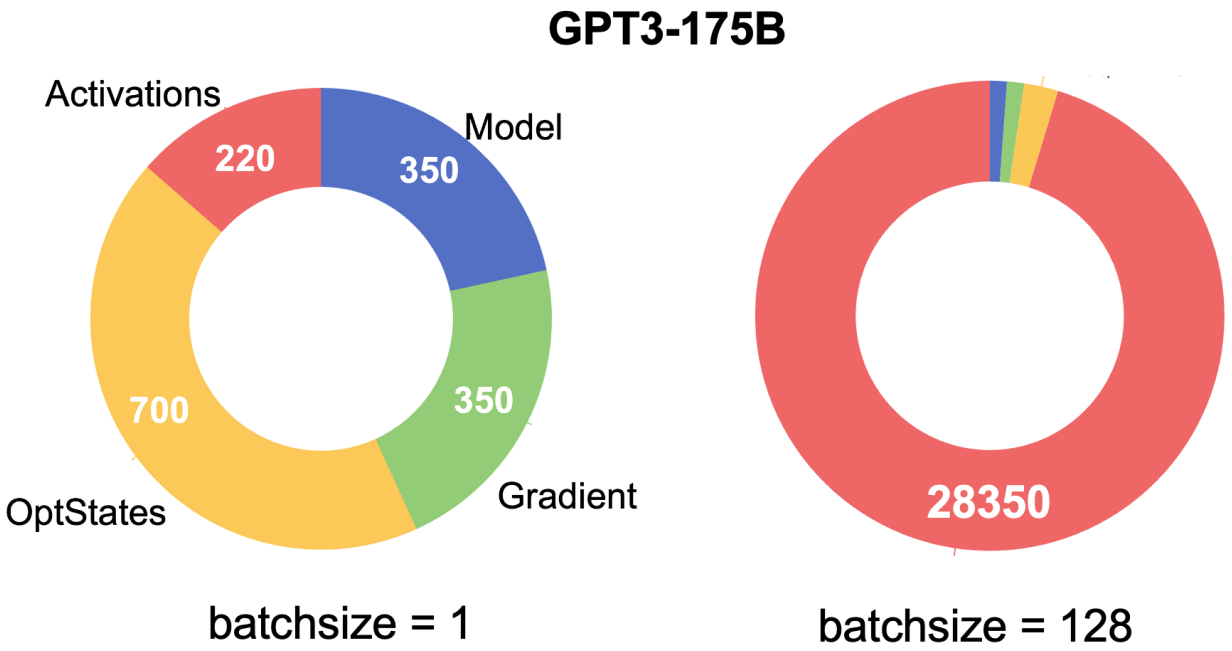
激活值是计算梯度时需要存储的辅助变量

激活值内存与批量大小(batch-size)成正比

[1] T. Wu, Y. He, B. Wang and K. Yuan, *Mixture-of-Channels: Exploiting Sparse FFNs for Efficient LLMs Pre-Training and Inference,* Submitted to NeurIPS 2025

大模型激活值内存

大模型内存 = **模型参数** + **梯度** + **优化器状态**+ **激活值**

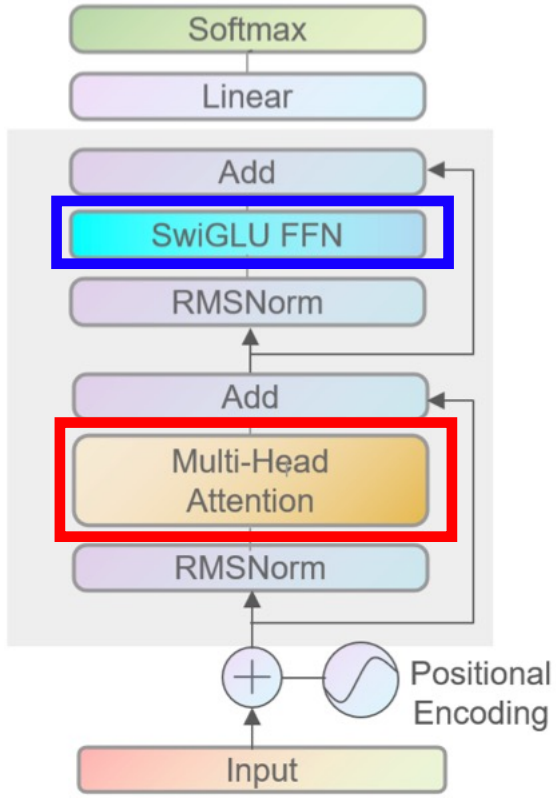


当批大小较大时，激活值所需内存急剧增大，占主导作用

激活值内存节省在大批量情形下更重要

本单元探讨如何节省激活值

激活值内存分解



批量大小: b　文本长度: s

隐藏层维度: d

Multi-Head Self-Attention (MHSA): 对于第i个头，需要

$$Q_i = XW_Q^i \in \mathbb{R}^{s\times d_h},\ K_i = XW_K^i \in \mathbb{R}^{s\times d_h},\ V_i = XW_V^i \in \mathbb{R}^{s\times d_h}$$

$$A_i = \mathrm{FlashAttention}(Q_i, K_i, V_i) \in \mathbb{R}^{s\times d_h}$$

$$A = [A_1; \cdots; A_h] \in \mathbb{R}^{s\times d},\quad O = AW_o \in \mathbb{R}^{s\times d}$$

需要存储Q, K, V, A, O共计$5sd$参数；当批大小为b时，共计$\mathbf{5bsd}$参数

SwiGLU FFN: 需要计算

$$G = XW_{\text{gate}} \in \mathbb{R}^{s\times d_{\text{ffn}}},\quad U = XW_{\text{up}} \in \mathbb{R}^{s\times d_{\text{ffn}}}$$

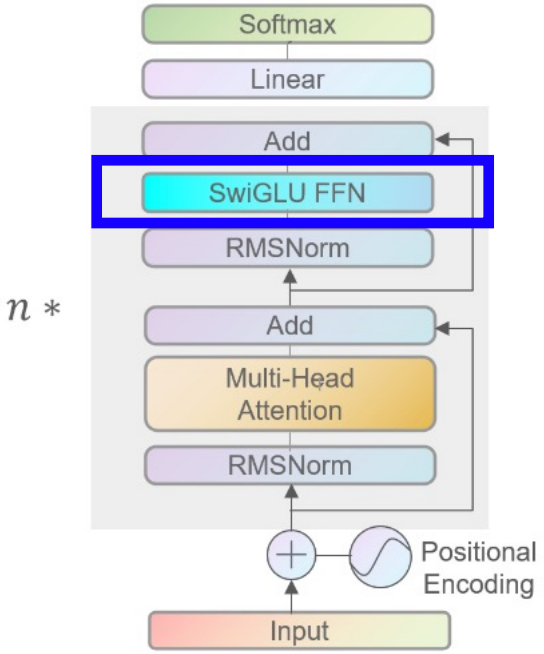
$$S = \mathrm{SiLU}(G) \in \mathbb{R}^{s\times d_{\text{ffn}}},\ Z = S \odot U \in \mathbb{R}^{s\times d_{\text{ffn}}},\ D = ZW_{\text{down}} \in \mathbb{R}^{s\times d}.$$

需要存储G, U, S, Z, D共计$\boldsymbol{b(d + 4d_{\text{ffn}})s}$ 参数 (其中$d_{\text{ffn}} = \frac{8}{3}d$)

激活值内存分解

- 大模型激活值有四个来源: FlashAttention (FA) + FFN + RMSNorm + Residual

$$Layer\ Activation = Attention\ 5bsd + FFN\ 11.67bsd + RMSNorm\ 2bsd + Residual\ 2bsd$$



激活值实际测试结果

		LLaMA (350M)	LLaMA (1.3B)
Per-Layer (×24)	Attention	177M	336M
	FFN	400M	791M (18.54G)
	Others	68M	134M
LLM head		2.16G	2.16G
Total		17.64G	32.4G

11.67/5 = 2.33

FFN激活值是FA的**2.314**倍

节省FFN激活值是节省内存的关键

Tong Wu, Kun Yuan, et. al. *Mixture-of-Channels: Exploiting Sparse FFNs for Efficient LLMs Pre-Training and Inference*, submitted to NeurIPS 2025

大模型FFN的结构：SwiGLU

- SwiGLU在大模型FFN中广泛使用，如LLaMA-2/3, Qwen2等

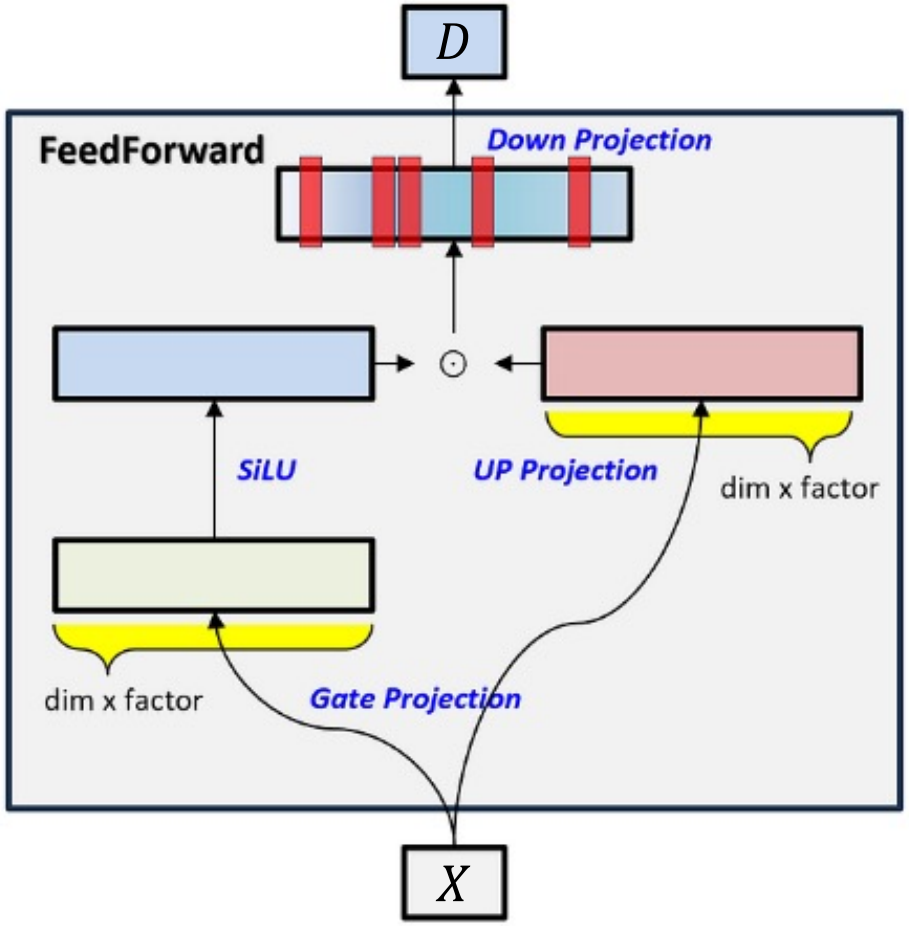
FFN. For each input $X \in \mathbb{R}^{s\times d}$ to the FFN module, we first compute and store:

$$G = XW_{\text{gate}} \in \mathbb{R}^{s\times d_{\text{ffn}}}, \quad U = XW_{\text{up}} \in \mathbb{R}^{s\times d_{\text{ffn}}},$$

where $W_{\text{gate}}, W_{\text{up}} \in \mathbb{R}^{d\times d_{\text{ffn}}}$ are the weights corresponding to the gating and up-sampling branches in the SwiGLU activation.

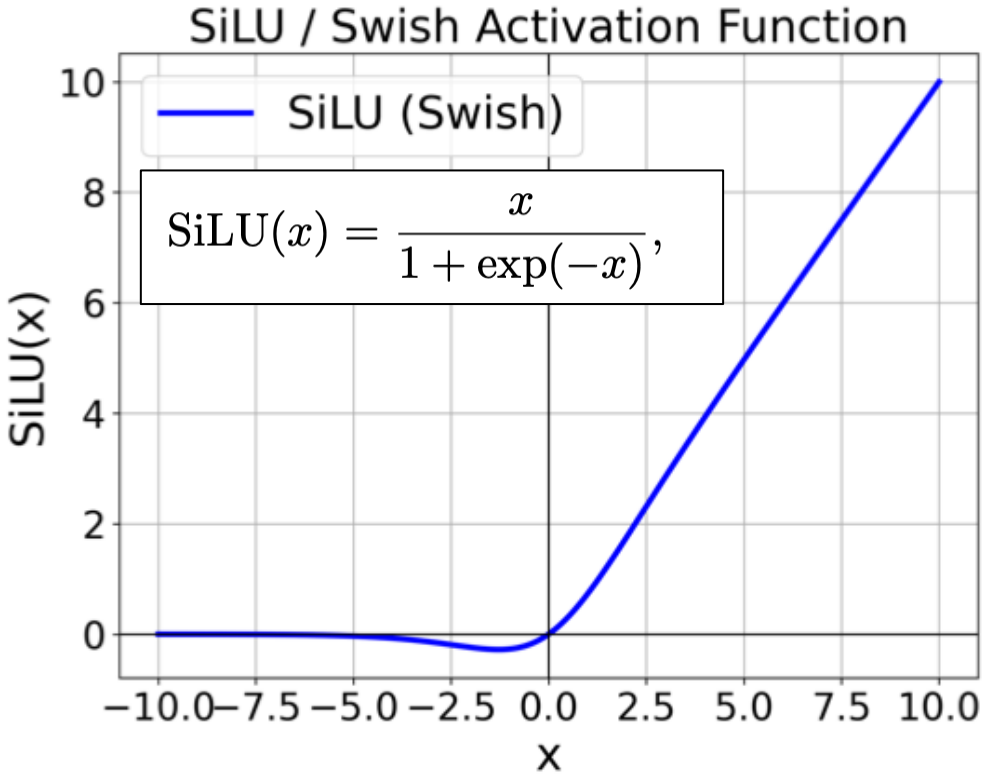
$$S = \text{SiLU}(G) \in \mathbb{R}^{s\times d_{\text{ffn}}}$$
$$Z = S \odot U \in \mathbb{R}^{s\times d_{\text{ffn}}}$$
$$D = ZW_{\text{down}} \in \mathbb{R}^{s\times d},$$

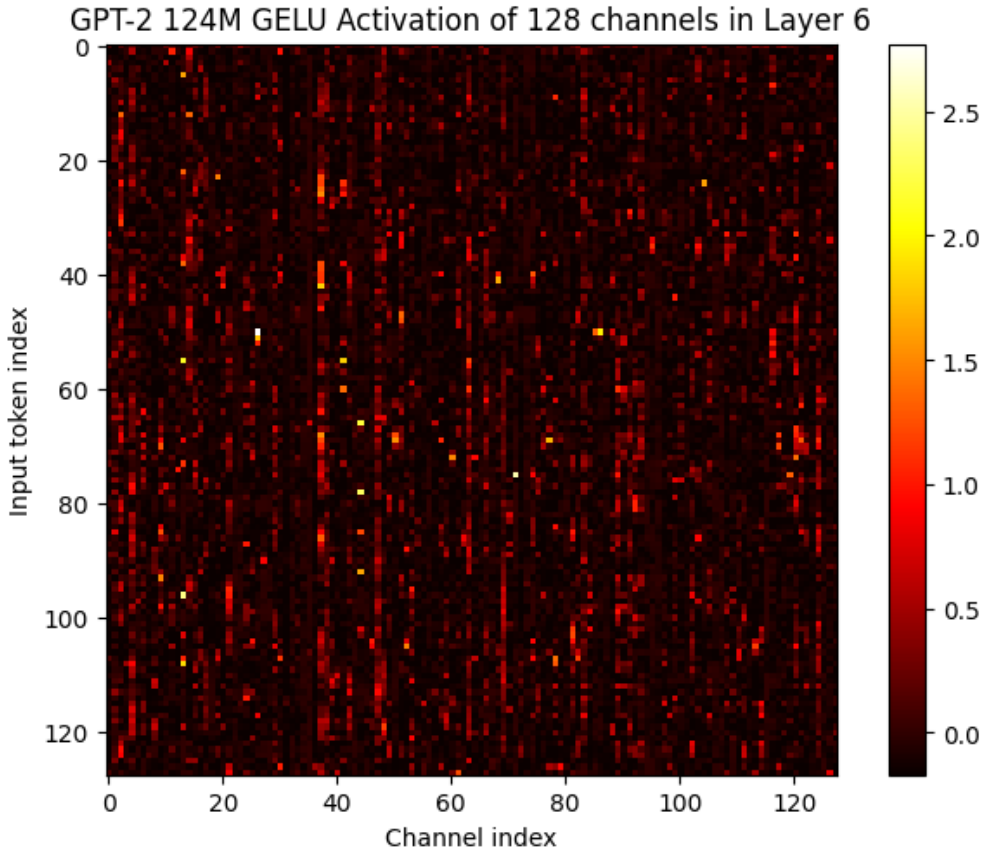
where $W_{\text{down}} \in \mathbb{R}^{d_{\text{ffn}}\times d}$ is the down-sampling weight.



大模型FFN的结构：SwiGLU

Tong Wu, Kun Yuan, et. al. *Mixture-of-Channels: Exploiting Sparse FFNs for Efficient LLMs Pre-Training and Inference*, submitted to NeurIPS 2025





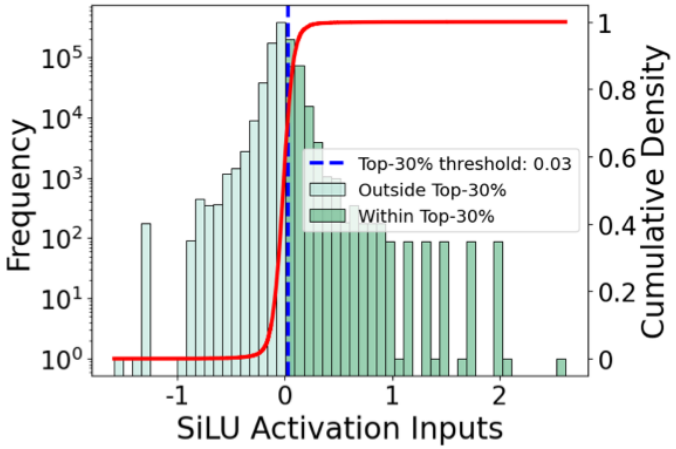
- 当$x \geq 0$, SiLU进行强激活；
- 当$x < 0$, SiLU将对输入进行抑制；

>70% are negative or near 0 in each row

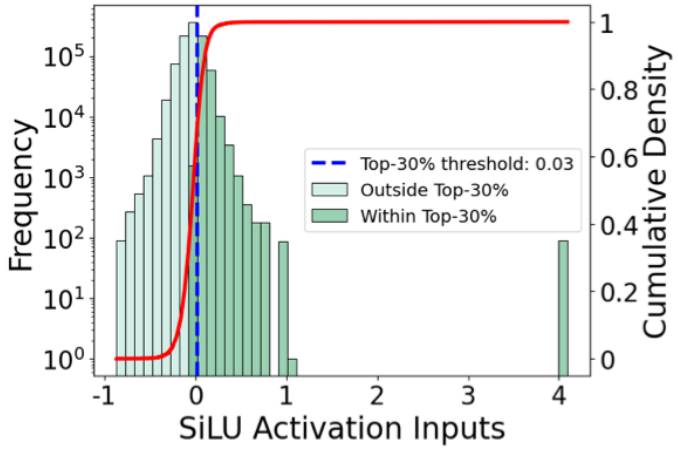
大模型FFN的结构：SwiGLU

北京大学 PEKING UNIVERSITY

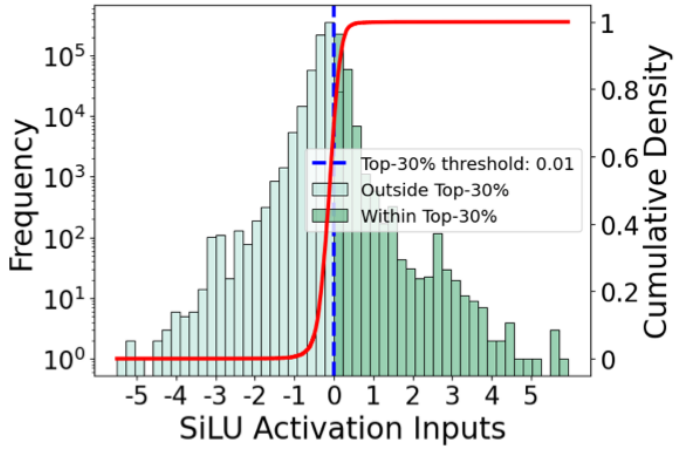
接近70%的SiLU输入小于等于0



(a) LLaMA-2 Layer 0.

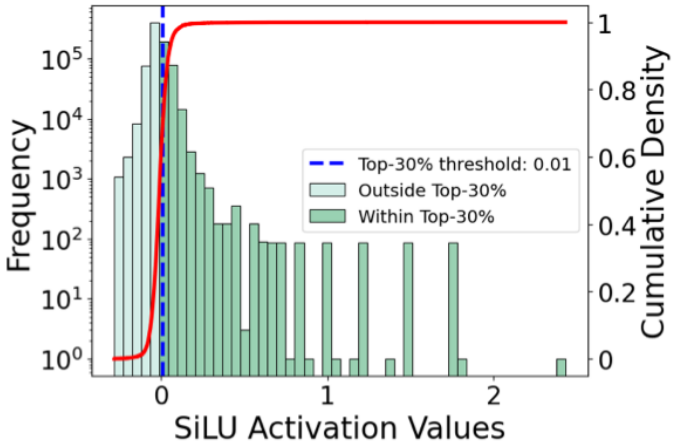


(b) LLaMA-2 Layer 16.

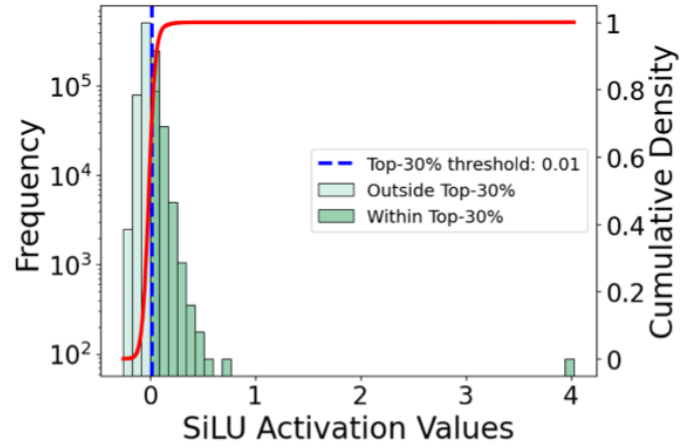


(c) LLaMA-2 Layer 31.

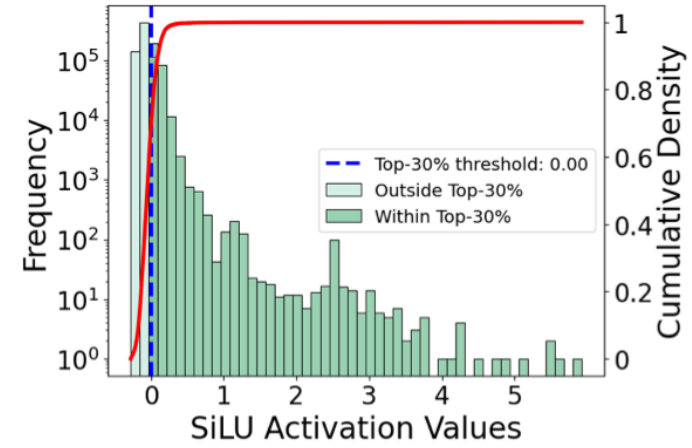
接近70%的SiLU输出被抑制，只有剩余30%被激活



(d) LLaMA-2 Layer 0.



(e) LLaMA-2 Layer 16.



(f) LLaMA-2 Layer 31.

基于稀疏特征的FFN新架构: Mixture of Channels

核心想法：对于每个input token，动态选择最相关的Top-K channels

传统FFN架构: SwiGLU

FFN. For each input $X \in \mathbb{R}^{s\times d}$ to the FFN module, we first compute and store:

$$G = XW_{\text{gate}} \in \mathbb{R}^{s\times d_{\text{ffn}}}, \quad U = XW_{\text{up}} \in \mathbb{R}^{s\times d_{\text{ffn}}},$$

where $W_{\text{gate}}, W_{\text{up}} \in \mathbb{R}^{d\times d_{\text{ffn}}}$ are the weights corresponding to the gating and up-sampling branches in the SwiGLU activation.

$$S = \text{SiLU}(G) \in \mathbb{R}^{s\times d_{\text{ffn}}}$$
$$Z = S \odot U \in \mathbb{R}^{s\times d_{\text{ffn}}}$$
$$D = ZW_{\text{down}} \in \mathbb{R}^{s\times d},$$

where $W_{\text{down}} \in \mathbb{R}^{d_{\text{ffn}}\times d}$ is the down-sampling weight.

基于稀疏特征的FFN新架构: Mixture of Channels (MoC)

MoC. For each input $X \in \mathbb{R}^{s\times d}$ to the FFN module, we first compute and store:

$$G = XW_{\text{gate}} \in \mathbb{R}^{s\times d_{\text{ffn}}}, \quad U = XW_{\text{up}} \in \mathbb{R}^{s\times d_{\text{ffn}}},$$

where $W_{\text{gate}}, W_{\text{up}} \in \mathbb{R}^{d\times d_{\text{ffn}}}$ are the weights corresponding to the gating and up-sampling branches in the SwiGLU activation.

$$S = \text{SiLU}(G), \quad S' = S \odot M,$$
$$Z' = S' \odot U,$$
$$D = Z'W_{\text{down}},$$

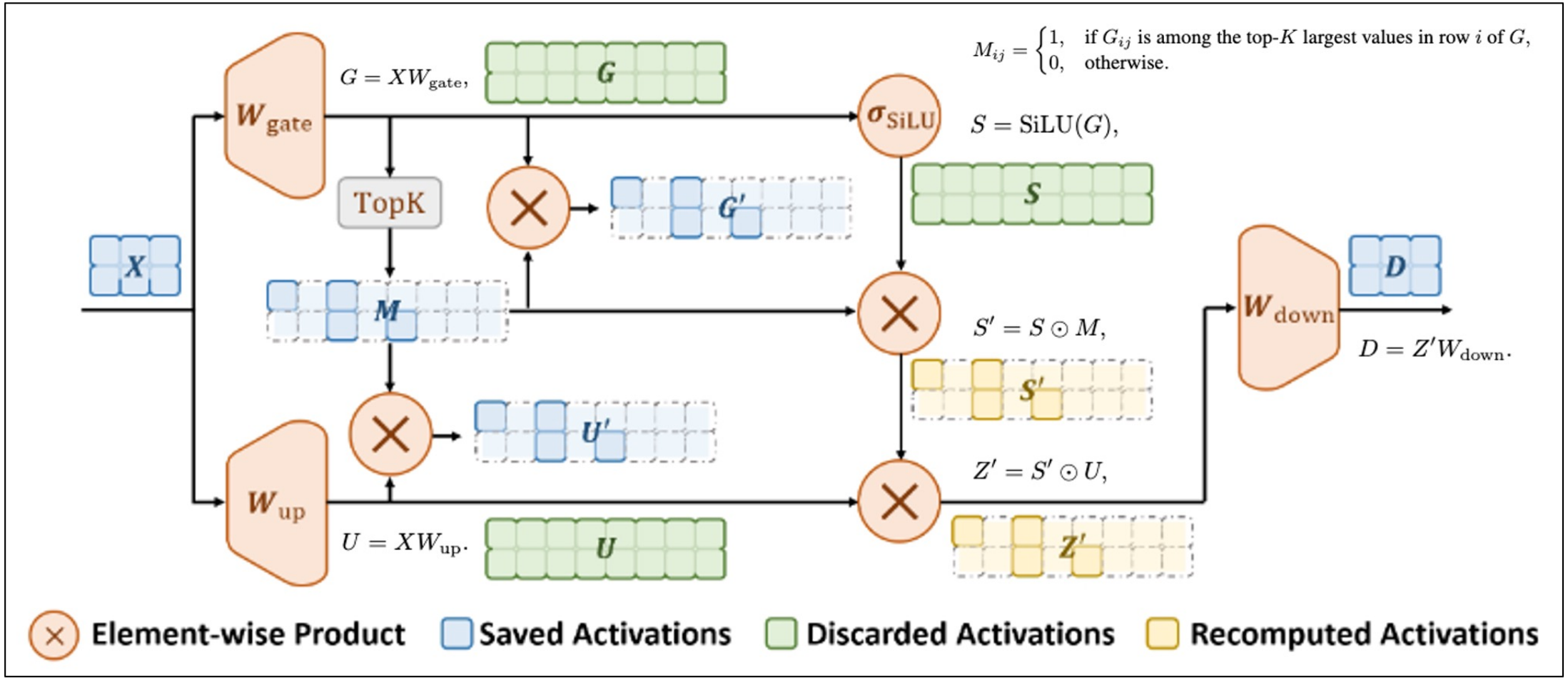
where W_{down} is the down-sampling weight, and M is defined as

$$M_{ij} = \begin{cases} 1, & \text{if } G_{ij} \text{ is among the top-}K \text{ largest values in row } i \text{ of } G, \\ 0, & \text{otherwise.} \end{cases}$$

由于我们只激活部分channel进行前向运算，我们称该类架构为混合Channel模型(Mixture of Channel, MoC)

混合Channel模型(Mixture of Channel, MoC)

- MoC架构示意图(实验中一般选择top-20%的Channel)



混合Channel模型的高效训练

北京大学 PEKING UNIVERSITY

MoC模型的前向传播

$$G = XW_{\text{gate}}, \qquad U = XW_{\text{up}}.$$
$$M = \text{TopK}(G)$$
$$S = \text{SiLU}(G), \quad S' = S \odot M,$$
$$Z' = S' \odot U, \quad D = Z'W_{\text{down}}.$$

MoC模型的反向传播

$$\nabla W_{\text{down}} = (Z')^\top \nabla D, \qquad \nabla Z' = \nabla D W_{\text{down}}^\top,$$
$$\nabla S' = (U \odot M) \odot \nabla Z', \qquad \nabla U = S' \odot \nabla Z',$$
$$\nabla S = \nabla S', \qquad \nabla G = \nabla S' \odot (\nabla \text{SiLU})(G),$$
$$\nabla W_{\text{gate}} = X^\top \nabla G, \qquad \nabla W_{\text{up}} = X^\top \nabla U,$$
$$\nabla X = \nabla G W_{\text{gate}}^\top + \nabla U W_{\text{up}}^\top,$$

稀疏存储

- 存储稀疏激活值$Z' = Z \odot M$, $U' = U \odot M$ 以及 $S' = S \odot M$
- 由于 $\nabla S' = \nabla S = U \odot M \odot \nabla Z'$, 且∇SiLU是逐元素操作，所以只需存储$G' = G \odot M$
- 与之对比，传统FFN架构的激活值需要存储稠密的Z, U, S, G

混合Channel模型的高效训练

MoC模型的激活值计算

	MoC	MoC+GCP	FFN	FFN+GCP
Stored Activations	$G \odot M$ $U \odot M$ $S \odot M$ $Z \odot M$ M and D	$G \odot M$ $U \odot M$ – – M and D	G U S Z D	G U – – D
Memory Cost	$bsd_{\text{ffn}} + bsd$ $(3.67bsd)$	$0.6bsd_{\text{ffn}} + bsd$ $(2.6bsd)$	$4bsd_{\text{ffn}} + bsd$ $(11.67bsd)$	$2bsd_{\text{ffn}} + bsd$ $(6.33bsd)$

- 相比FFN激活值，MoC只存储**稀疏激活值**
- MoC节省约**68%**的激活值内存
- S与Z的计算非常简单，可以通过重计算来得到，无需存储

MoC系统级算子优化与内存节省

- 无结构的稀疏性无法带来计算的节省，同时top-K会带来额外计算开销，需要优化
- 基于RAFT库和Triton对MoC算子进行系统级优化，使得MoC的端到端计算时间与FFN相同
- 2:8结构稀疏算子可以带来计算的节省

MoC没有带来整体计算时间的显著overhead

	Standard FFN (ms)	MoC using optimized kernels (ms)
Forward	20.2	21.1
Backward	21.6	22.8
Total	41.8	43.9

Pretrain LLaMA on C4

	60M	130M	350M	1B
FFN-based LLM + AdamW	30.44 (38.3G)	23.92 (54.1G)	18.26 (52.5G)	15.56 (56.6G)
GaLore	34.88 (38.1G)	25.36 (53.9G)	18.95 (51.7G)	15.64 (52.5G)
SLTrain	34.15 (33.6G)	26.04 (59.4G)	19.42 (69.1G)	16.14 (70.0G)
MoC	**30.59** (21.8G)	**24.02** (41.7G)	**18.57** (34.6G)	15.80 (41.9G)
$\text{MoC}_{2:8}$	31.02 (22.1G)	24.12 (42.3G)	18.68(36.4G)	**15.62** (42.7G)
batch size per GPU	256	256	128	64
K/d_{model}	128 / 256	384 / 768	512 / 1024	1024 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

Loss损失**0.4%**的前提下端到端整体内存节省**24.6%**(所有内存均计算在内)

MoC与常见大模型架构的兼容性

北京大学 PEKING UNIVERSITY

MoC与MoE架构的兼容性

	160M	530M
Vanilla Mixtral	**23.77** (48.3G)	**18.65** (41.7G)
Mixtral+MoC	**24.44** (38.2G)	**18.88** (30.0G)
batch size per GPU	256	128
K/d_{model}	256 / 512	512 / 1024

MoC与GQA架构的兼容性

Model	Structure	Ppl.	Memory
LLaMA-130M	GQA	24.02	53.9G
LLaMA-130M	GQA+MoC	24.26	**34.4G**
LLaMA-350M	GQA	18.51	52.9G
LLaMA-350M	GQA+MoC	18.69	**36.9G**

MoC与Qwen架构的兼容性

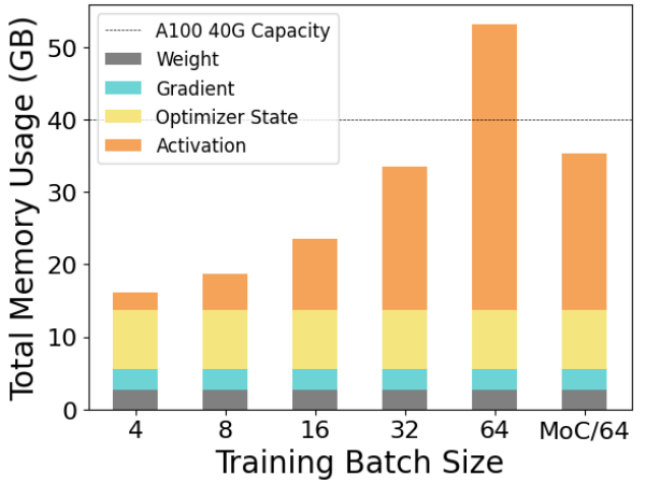
Model	Ppl.
Qwen3-300M	18.52
Qwen3-300M+MoC	18.59

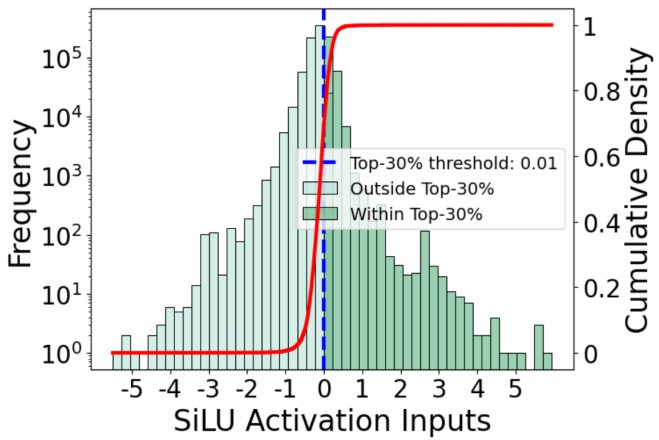
MoC与LLaMA-7B的兼容性

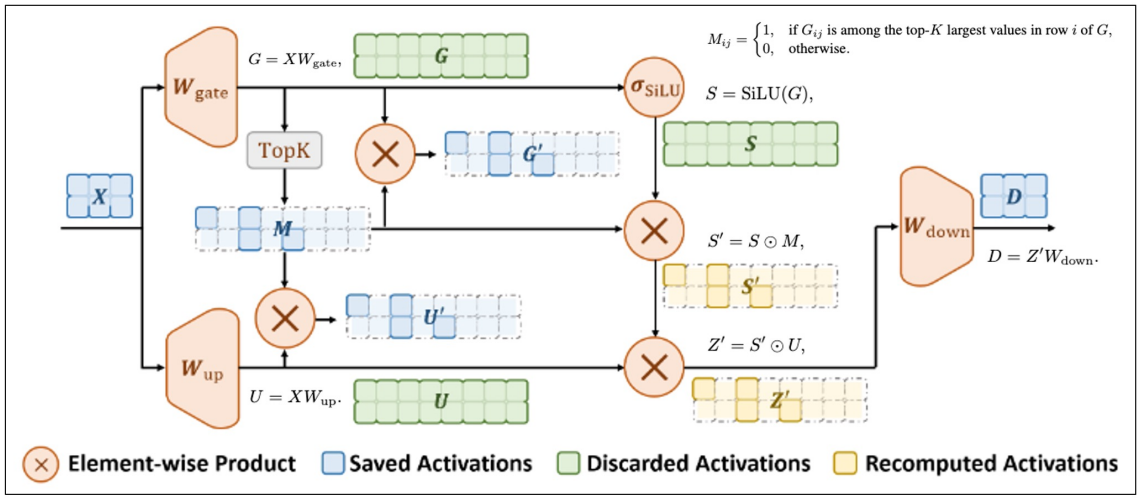
Model	Ppl.
LLaMA-7B	26.14
LLaMA-7B+MoC	26.47

单元小结

- 大模型内存中**激活值**占主导；激活值中FFN的内存占用占主导
- FFN结构：SiLU激活函数具有稀疏结构; 多数Channel被抑制
- 新算法核心思想：利用SiLU稀疏性结构设计混合Channel模型，节省训练内存，提升推理速度
- 收益总结：在1%性能损失以内，可以节省68%左右的激活值内存，以及25%左右的端到端内存；推理速度端到端提升10%以上







北京大学
PEKING UNIVERSITY

PART 03

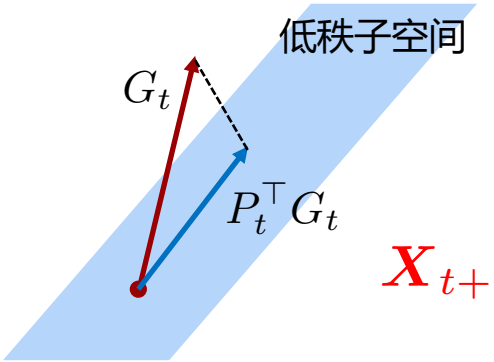
基于激活值层间低秩结构的参数高效训练方法

孔博傲 梁钧著 刘禹希 邓人嘉

大模型训练内存构成

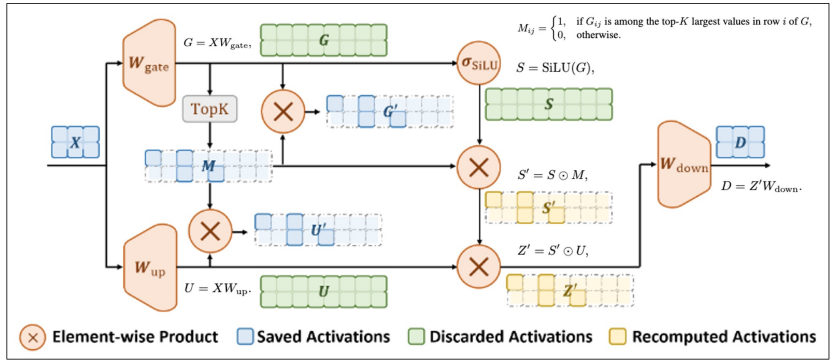
大模型内存 = **模型参数** + **梯度** + **优化器状态** + **激活值**

- 梯度低秩投影方法仅节省**优化器状态**内存



$$X_{t+1} = X_t + P_t \rho(P_t^\top G_t)$$

- 激活值稀疏方法（MoC）仅节省**激活值**内存



$$M = \text{TopK}(G)$$
$$S = \text{SiLU}(G),$$
$$S' = S \odot M,$$
$$Z' = S' \odot U,$$

- 如何节省**模型参数**和**梯度**？**参数高效的训练方法！**
- 由于模型参数、梯度、优化器状态都与参数量有关，参数高效的训练方法将同时节省这三个部分

LoRA是参数高效的微调方法，无法应用于预训练

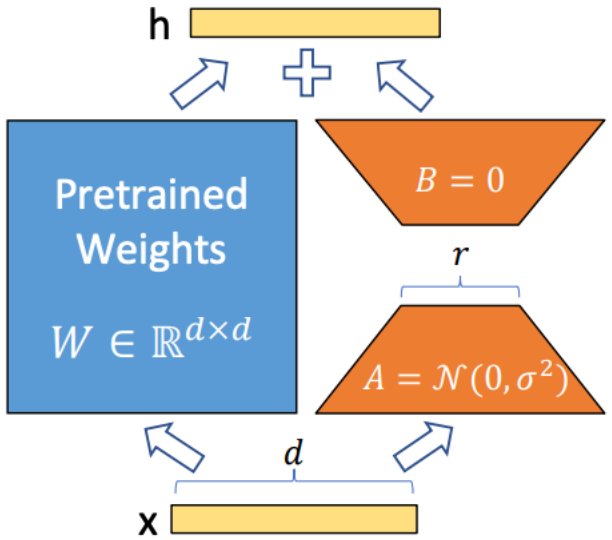
- LoRA是最流行的参数高效微调方法，但其无法应用于预训练

$$\min_{W \in \mathbb{R}^{p \times q}} \quad \mathbb{E}_{\xi \sim \mathcal{D}}[F(W;\xi)] \qquad \text{(预训练loss函数)}$$

$$\min_{A \in \mathbb{R}^{p \times r}, B \in \mathbb{R}^{r \times q}} \quad \mathbb{E}_{\xi \sim \mathcal{D}}[F(W + AB;\xi)] \qquad \text{(LoRA微调loss函数)}$$

只更新A和B

不更新X



Pre-trian LLaMA on C4

Methods	60M		130M		350M		1B	
	Perplexity	Memory	Perplexity	Memory	Perplexity	Memory	Perplexity	Memory
AdamW♠	34.06	0.36G	25.08	0.76G	18.80	2.06G	15.56	7.80G
Low-Rank♠	78.18	0.26G	45.51	0.54G	37.41	1.08G	142.53	3.57G
LoRA♠	34.99	0.36G	33.92	0.80G	25.58	1.76G	19.21	6.17G

- LoRA的预训练PPL显著增大
- 需要探索适用于预训练的参数高效算法

大模型结构：相邻层激活值残差具有低秩结构

- 大模型参数本身不具有低秩性，直接对参数进行低秩逼近会引入较大误差

$$Q_\ell = XW_\ell^Q \approx XA_\ell^Q B_\ell^Q$$

激活值　输入　参数矩阵　低秩参数矩阵

- 核心想法：需要使用**上一层的激活值**对参数低秩逼近误差进行**补偿** [1]

$$Q_\ell = XW_\ell^Q \approx Q_{\ell-1} + XA_\ell^Q B_\ell^Q$$

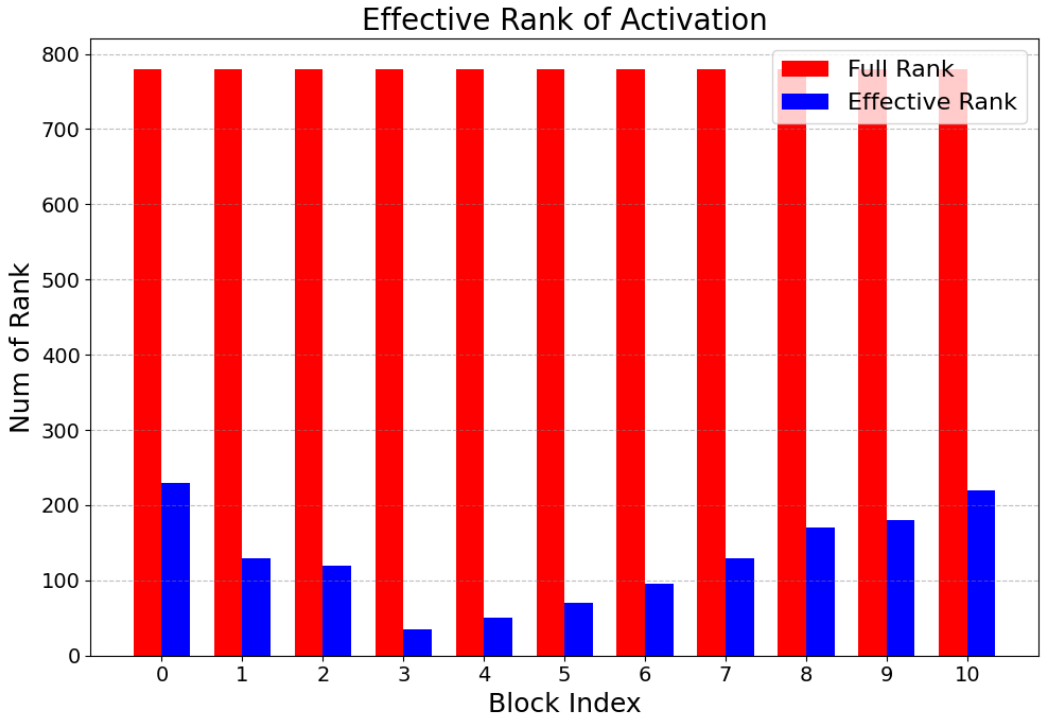
l层激活值　　$l-1$层激活值

换言之，相邻层激活值残差 $Q_\ell - Q_{\ell-1}$ 具有低秩结构

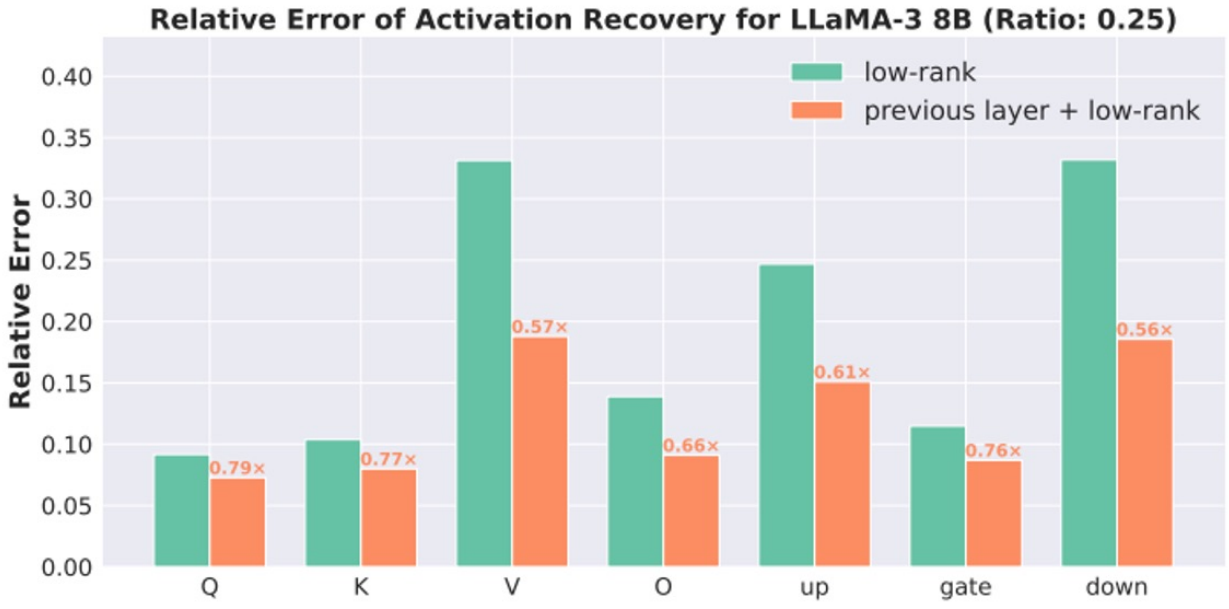
[1] Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

大模型结构：相邻层激活值残差具有低秩结构

$$Q_\ell = XW_\ell^Q \approx XA_\ell^Q B_\ell^Q$$
$$Q_\ell = XW_\ell^Q \approx Q_{\ell-1} + XA_\ell^Q B_\ell^Q$$



激活值层间残差具有低秩性

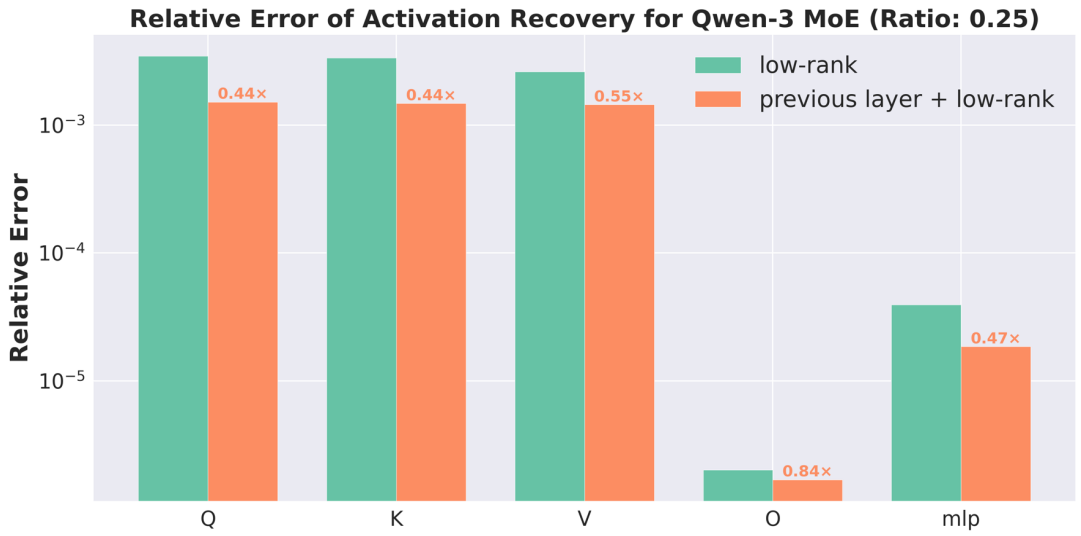


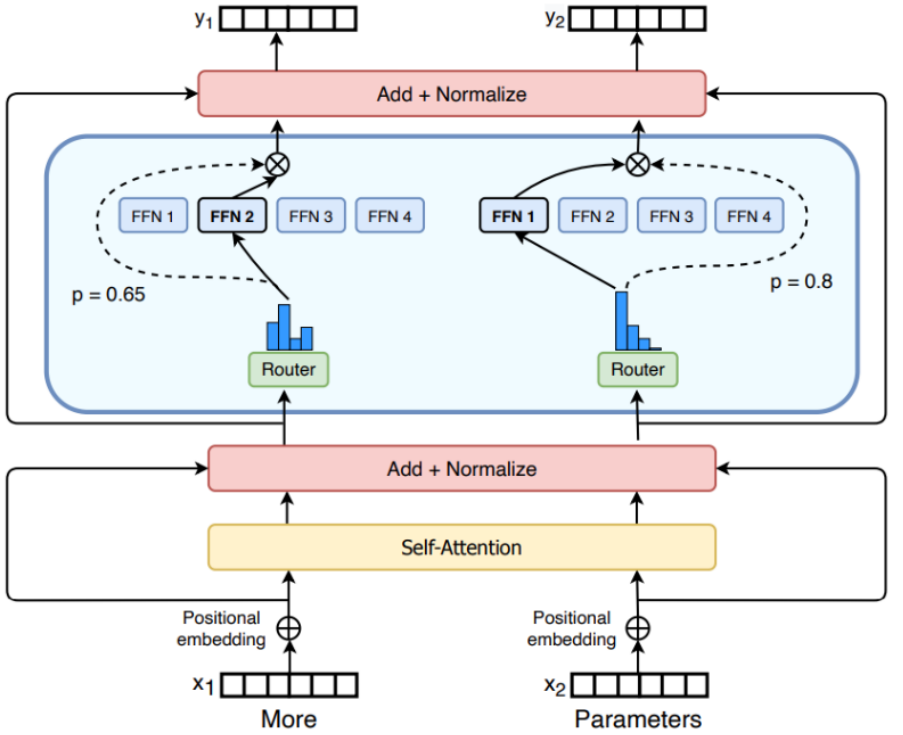
误差补偿后，恢复误差减少**9%~54%**

[1] Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

大模型结构：相邻层间激活值残差具有低秩结构

- 在MoE架构中可以发现类似的层间低秩结构



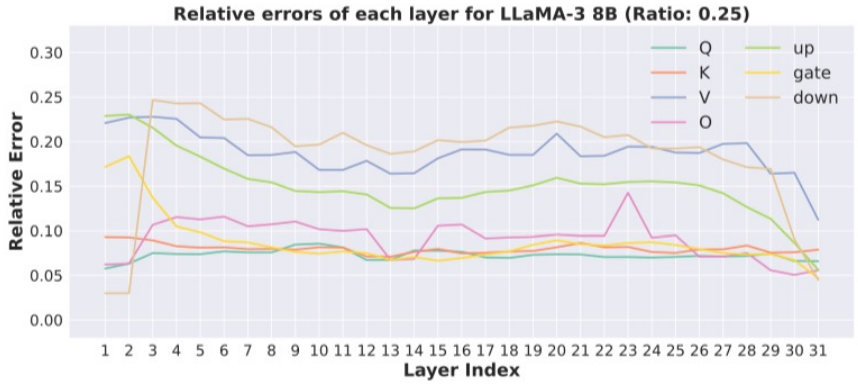


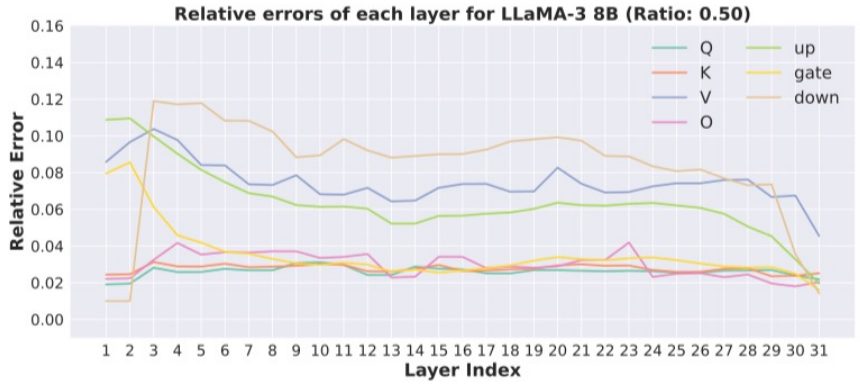
- 在相同秩的选取下，利用上一层激活值进行补偿能减小**16%~56%**的误差

[1] Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

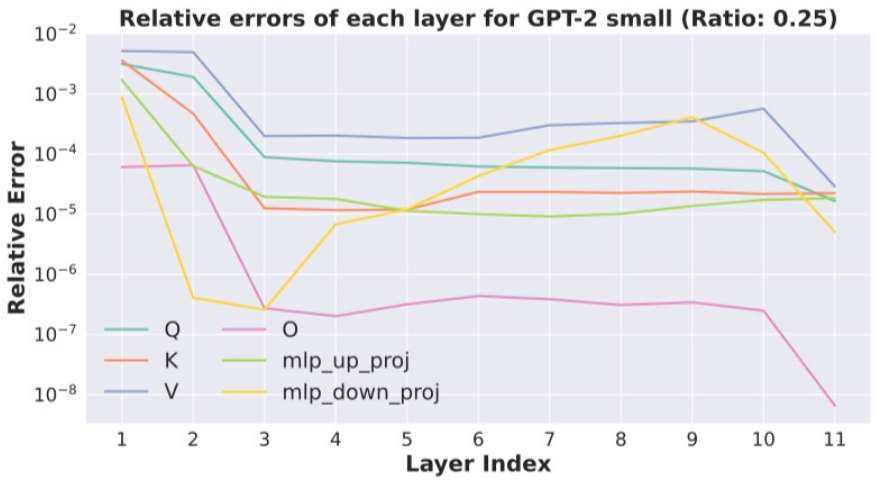
大模型结构：相邻层间激活值残差具有低秩结构

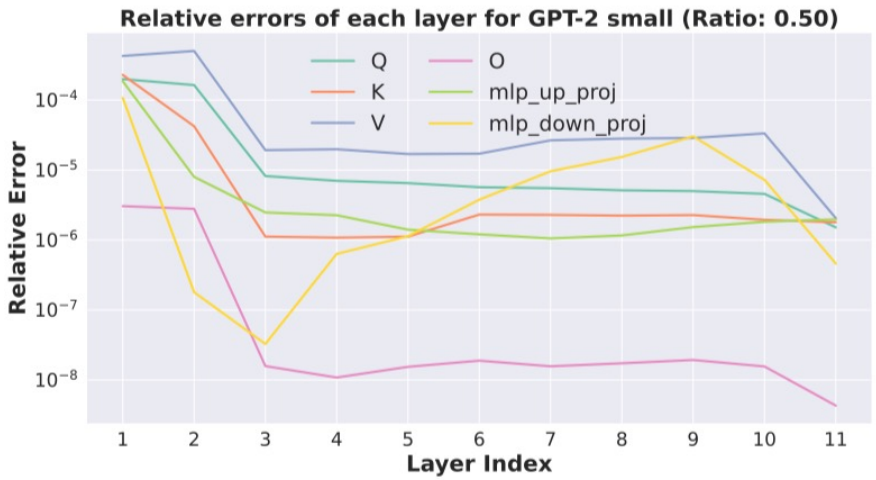
LLaMA-3 8B





GPT-2





低秩激活恢复的平均相对误差

激活值层间低秩结构的理论解释

- 相邻层激活值之间的相似性（高余弦相似度）使得层间激活值之差具有低秩性

Assumption 1 (Cosine similarity of adjacent attentions). *For $l = 2, 3, \cdots, L$, let $Y_l^P \in \mathbb{R}^{d\times d}$ as the activation of the linear of position P for the l-th layer. There exists a constant $\epsilon \in (0, 1)$ such that:*

$$\frac{\langle Y_l^P, Y_{l-1}^P \rangle_F}{\|Y_l^P\|_F \cdot \|Y_{l-1}^P\|_F} \geq 1 - \epsilon,$$

where $\langle \cdot, \cdot \rangle_F$ denotes the inner production of matrices induced by Frobenius norm.

Theorem 1. *Suppose Assumption 1 holds. Then there exists $r_0 > 0$ such that the approximation $\widetilde{Y}_{l,\beta}^P$ obtained by Eq. (3) has a lower error than the direct low-rank approximation $LR_r(Y_l^P)$ by a properly-selected β if $r < r_0$. Specifically, it holds that:*

$$\left\| Y_l^P - \widetilde{Y}_{l,\beta}^P \right\|_F^2 \leq \left\| Y_l^P - LR_r(Y_l^P) \right\|_F^2 .$$

带有误差补偿的低秩近似　　直接低秩近似

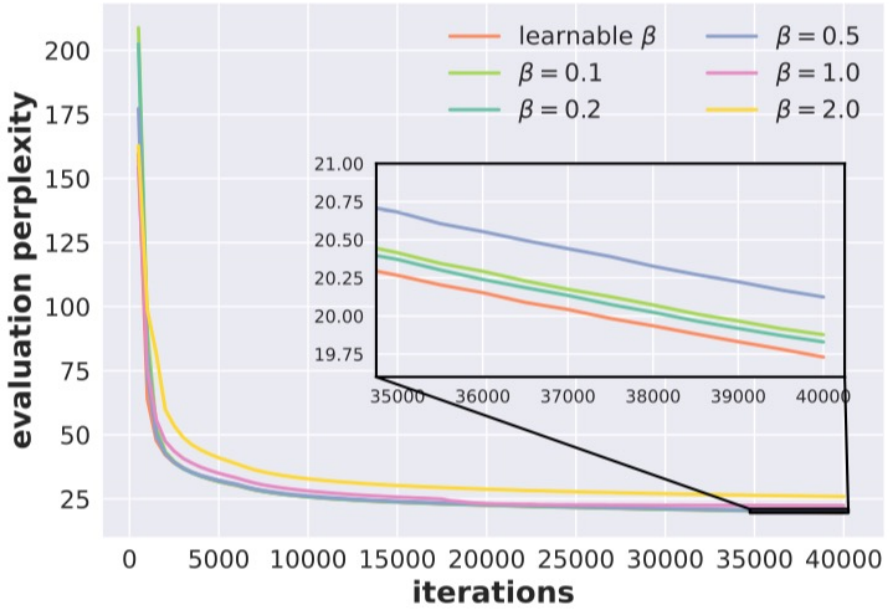
层间低秩结构引入的误差小于对激活值直接进行低秩近似

缩放系数的选择

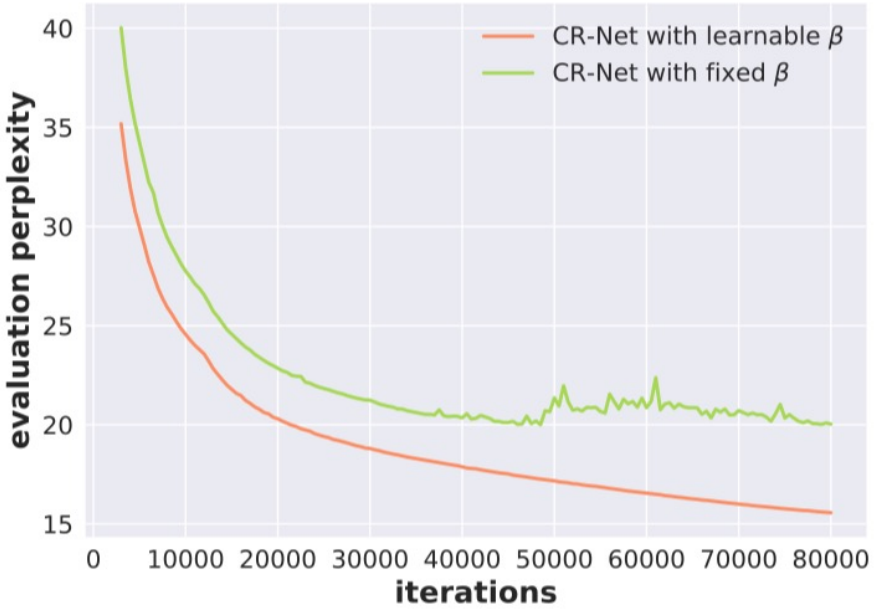
$$Y_\ell^{\mathrm{P}} = \beta_\ell^{\mathrm{P}} Y_{\ell-1}^{\mathrm{P}} + X_\ell^{\mathrm{P}} A_\ell^{\mathrm{P}} B_\ell^{\mathrm{P}}$$

- β_0 和 β_l是经验优化的缩放系数，用于平衡前一层激活和跨层差异低秩估计的贡献
- 将β_l设为可学习的缩放因子，允许模型动态调整历史激活（即前一层激活）和低秩输出在当前激活计算中的影响

LLaMA-2 350M



LLaMA-2 1B



Cross-layer Low-Rank Residual Network (CR-Net): 模型架构

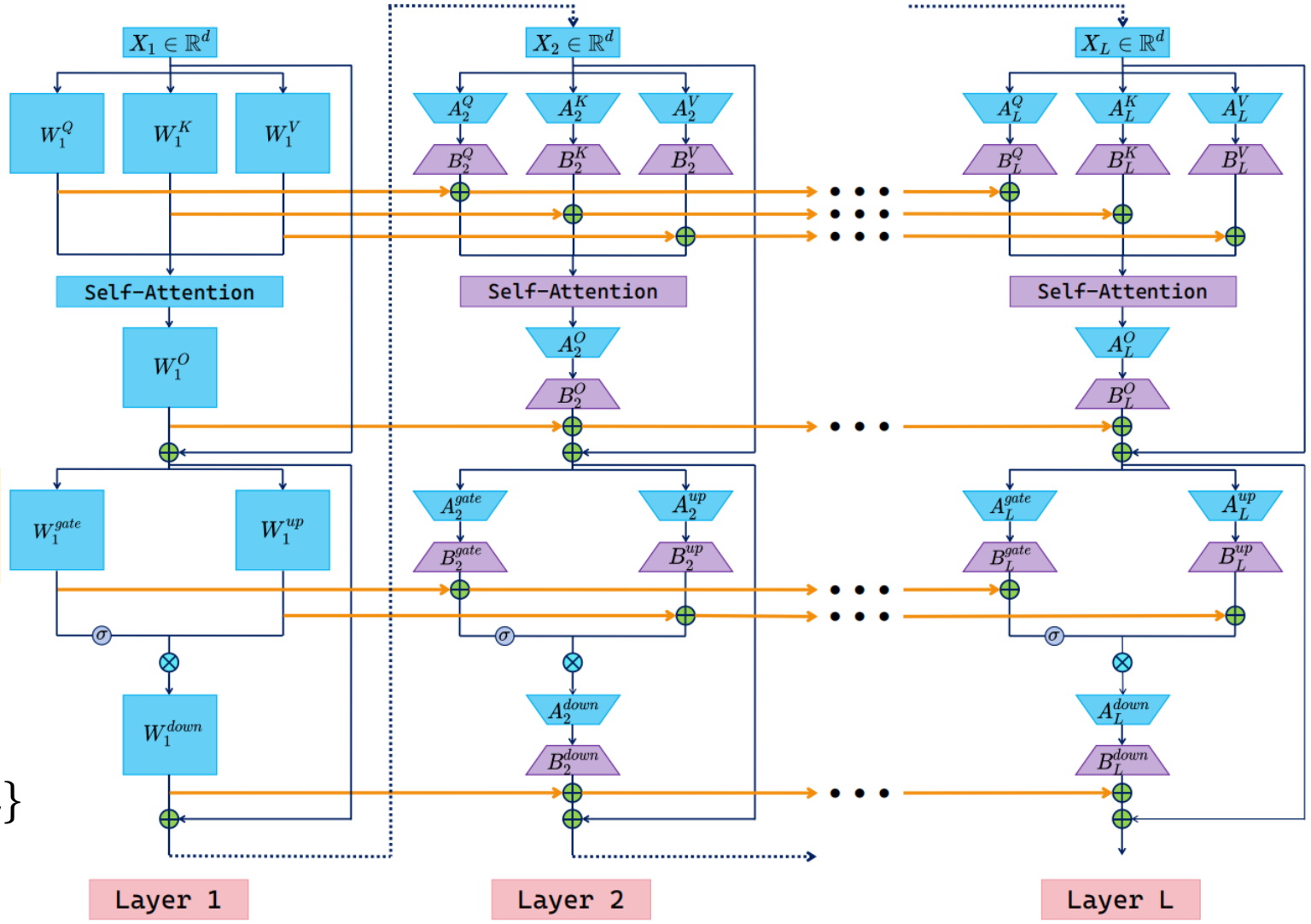
- 第一个transformer层，采用全尺寸参数矩阵
- 对 $\ell = 2, 3, \cdots, L$ 层

$$Y_\ell^{\mathrm{P}} = \beta_\ell^{\mathrm{P}} Y_{\ell-1}^{\mathrm{P}} + X_\ell^{\mathrm{P}} A_\ell^{\mathrm{P}} B_\ell^{\mathrm{P}}$$

可学习的缩放系数

第l层第P模块的激活值

$$\mathrm{P} \in \{Q, K, V, O, gate, up, down\}$$



Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

Cross-layer Low-Rank Residual Network (CR-Net): 模型架构

- 全参数预训练

$$Y_\ell^{\mathrm{P}} = X_\ell^{\mathrm{P}} W_\ell^{\mathrm{P}} \quad \text{(m, n)}$$

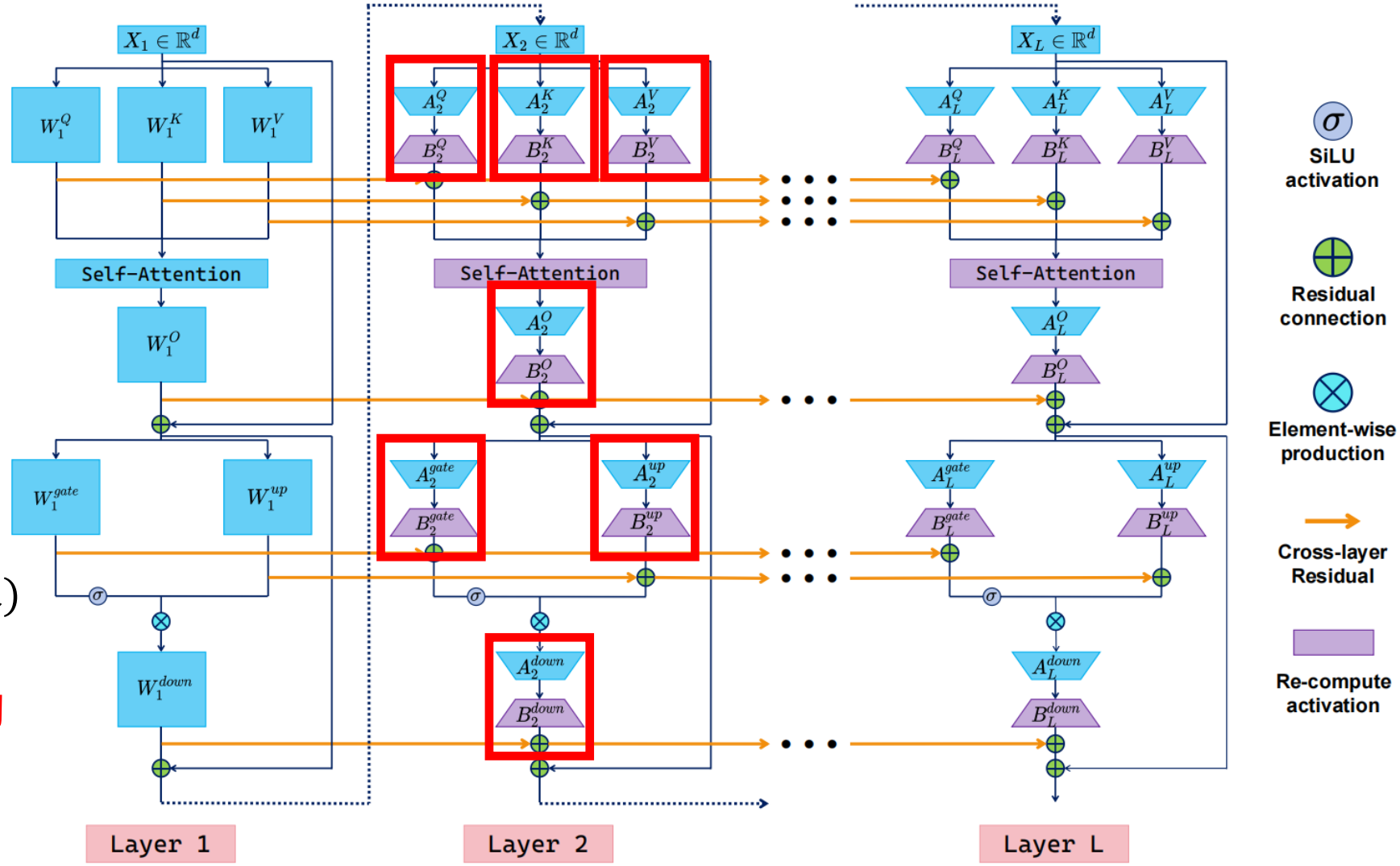
- 参数高效的预训练算法

$$Y_\ell^{\mathrm{P}} = \beta_\ell^{\mathrm{P}} Y_{\ell-1}^{\mathrm{P}} + X_\ell^{\mathrm{P}} A_\ell^{\mathrm{P}} B_\ell^{\mathrm{P}}$$

(m, r), (r, n)

- 参数量从mn降低为$r(m+n)$

- **更少的参数量意味着更小的模型、梯度和优化器状态**



Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

CR-Net：参数节省的实验验证

Pretrain LLaMA on C4

Approach	60M			130M			350M			1B		
	1.1B tokens			2.2B tokens			6.4B tokens			13.1B tokens		
	PPL	Para	Mem	PPL	Para	Mem	PPL	Para	Mem	PPL	Para	Mem
Full-rank	34.06	58	0.43	24.36	134	1.00	**18.80**	368	2.74	15.56	1339	9.98
LoRA	34.99	58	0.37	33.92	134	0.86	25.58	368	1.94	19.21	1339	6.79
ReLoRA	37.04	58	0.37	29.37	134	0.86	29.08	368	1.94	18.33	1339	6.79
SLTrain	34.15	44	0.32	26.04	97	0.72	19.42	194	1.45	16.14	646	4.81
CoLA	34.04	43	0.32	24.48	94	0.70	19.40	185	1.38	15.52	609	4.54
CR-Net ◇	**32.76**	43	0.32	**24.31**	90	0.67	18.95	183	1.36	**15.28**	583	4.35
GaLore	34.88	58	0.36	25.36	134	0.79	18.95	368	1.90	15.64	1339	6.60
RSO	34.55	58	0.36	25.34	134	0.79	18.87	368	1.90	15.86	1339	6.60
Apollo	**31.55**	58	0.36	**22.94**	134	0.79	**16.85**	368	1.90	14.20	1339	6.60
CR-Net †	32.76	43	0.32	23.74	106	0.79	17.08	250	1.86	**14.05**	870	6.48

- 相比参数高效方法，CR-Net能用更小的参数量（~43.6% in LLaMA-2 1B）实现更低的损失
- 相比利用梯度低秩性的方法，当对齐内存开销时，CR-Net在**更大的模型规模**下可实现更低的损失

CR-Net的重计算: 节省激活值

- CR-Net不节省激活值内存

(s, m) x (m, n) = (s, n) : Lsn

Vanilla: $Y_\ell^{\mathrm{P}} = X_\ell^{\mathrm{P}} W_\ell^{\mathrm{P}}$

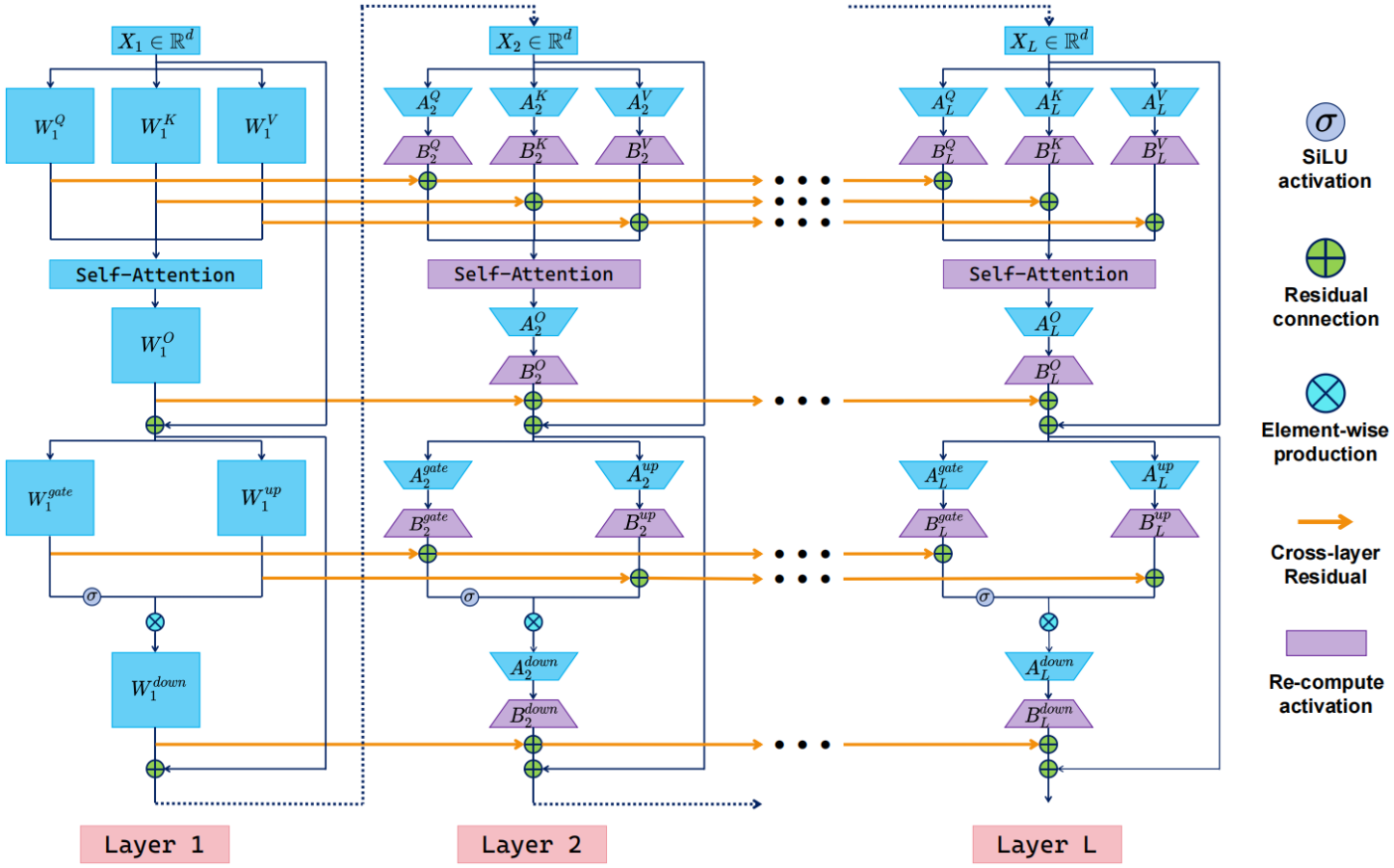
CR-Net: $Y_\ell^{\mathrm{P}} = \beta_\ell^{\mathrm{P}} Y_{\ell-1}^{\mathrm{P}} + X_\ell^{\mathrm{P}} A_\ell^{\mathrm{P}} B_\ell^{\mathrm{P}}$

(s, m) x (m, r) x (r, n) = (s, n): Lsn

- CR-Net + 重计算:

$$Y_\ell^{\mathrm{P}} = \frac{1}{\beta_\ell^{\mathrm{P}}}\left(Y_{\ell+1}^{\mathrm{P}} - X_{\ell+1}^{\mathrm{P}} A_{\ell+1}^{\mathrm{P}} B_{\ell+1}^{\mathrm{P}}\right)$$

只需存1份Y和L份$X_i A_i$: sn + Lsr



Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

CR-Net：内存节省

北京大学 PEKING UNIVERSITY

- 基于LLaMA2-7B模型预训练C4-en数据集
- 对于CoLA-M，r取1024；对于CR-Net，r取896

Table 4: Comparsion of validation perplexity (↓) and memory (↓) of different approaches in LLaMA-7B pre-training tasks. The results of compared methods are referred from [38, 57].

	Memory (GB)	10K	40K	65K	80K
8-bit Adam	72.59	N.A.	18.09	N.A.	15.47
8-bit GaLore	65.16	26.87	17.94	N.A.	15.39
Apollo	N.A.	N.A.	17.55	N.A.	14.39
CoLA-M	28.82	**22.76**	16.21	14.59	13.82
CR-Net w. re-computation	27.60	23.11	**16.01**	**14.47**	**13.72**
Training tokens (B)		1.3	5.2	8.5	10.5

- CR-Net能实现**62%**的内存节省，且模型性能优于其余baseline。

CR-Net：计算节省

Vanilla: $Y_\ell^{\mathrm{P}} = X_\ell^{\mathrm{P}} W_\ell^{\mathrm{P}}$ (smn)

(s, m) x (m, n) = (s, n) : smn

CR-Net: $Y_\ell^{\mathrm{P}} = \beta_\ell^{\mathrm{P}} Y_{\ell-1}^{\mathrm{P}} + X_\ell^{\mathrm{P}} A_\ell^{\mathrm{P}} B_\ell^{\mathrm{P}}$

(s, n) + (s, m) x (m, r) x (r, n) = sn + s(m+n)r

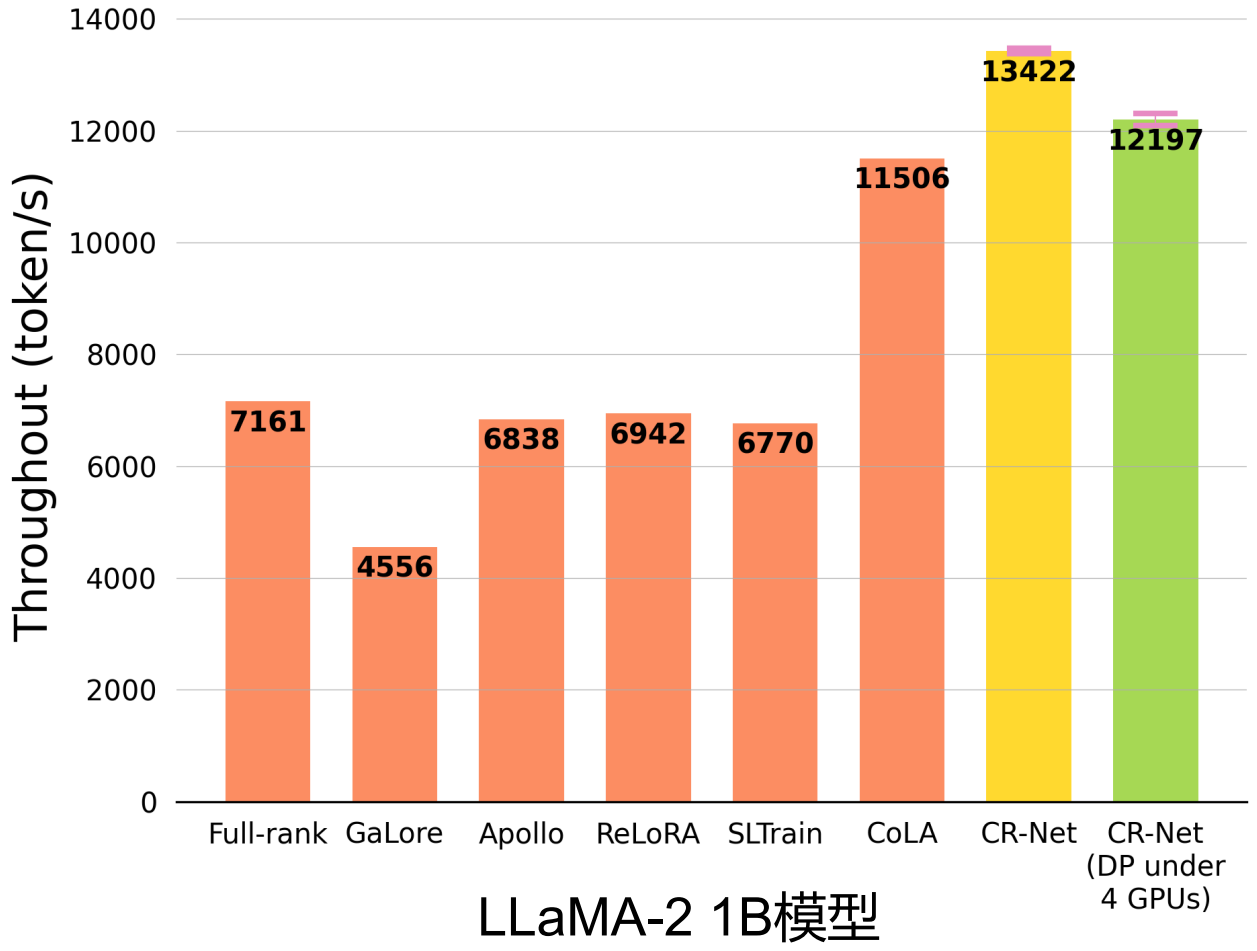
- s：序列长度；h：隐藏层维度；L：transformer层数；取序列长度s=256，batch size为1
- 对其他方法，r=512；对CR-Net，r=448，可保证CR-Net实现更好的验证集困惑度

Approach	FLOPs	LLaMA-2 1B
Full-rank	$L(24sh^2 + 12s^2h + 18shh_{\mathrm{ff}})$	2.422×10^{12} (1.000×)
(Re)LoRA	$L(40sh^2 + 24s^2h + 30shh_{\mathrm{ff}})$	4.054×10^{12} (1.674×)
SLTrain	$L(24sh^2 + 12s^2h + 18shh_{\mathrm{ff}} + 24h^2r + 18hh_{\mathrm{ff}}r)$	7.164×10^{12} (2.958×)
GaLore	$L(24sh^2 + 12s^2h + 18shh_{\mathrm{ff}} + 16h^2r + 12hh_{\mathrm{ff}}r)$	5.583×10^{12} (2.305×)
CoLA	$L(48shr + 12s^2h + 18sr(h + h_{\mathrm{ff}}))$	1.005×10^{12} (0.415×)
CR-Net	$24sh^2 + 12s^2h + 18shh_{\mathrm{ff}} + (L-1)(48shr + 12s^2h + 18sr(h + h_{\mathrm{ff}}))$	0.934×10^{12} (0.385×)

- 相比标准LLaMA-2 1B网络，CR-Net仅需**38.5%**的计算开销

CR-Net：吞吐量

- CR-Net整体上能实现较优的吞吐量。对比标准模型提升**87%**
- 即使额外考虑多卡数据并行下的通信时间，CR-Net在吞吐量上仍优于其余所有baseline，对比标准模型提升约**66%**



CR-Net + GCP v.s. 标准架构 + GCP

- 假设序列长度s=256，考虑LLaMA-2 1B模型在**重计算策略**下，单步梯度估计的**全过程计算量**和**全局存储量**

- 对其他方法，r=512；对CR-Net，r=448，可保证CR-Net实现更好的验证集困惑度

Algorithms	Memory (GB)	FLOPs ($\times 10^{14}$)	
Vanilla GCP + Full-rank	11.98 (1.000×)	2.067 (1.000×)	
CoLA-M	12.04 (1.005×)	0.764 (0.370×)	
CR-Net‡	11.81(0.986×)	0.692 (0.334×)	每8层存储一套完整激活
CR-Net♯	9.94(0.830×)	0.694 (0.335×)	每32层存储一套完整激活

- 相比标准LLaMA-2 1B使用标准GCP，CR-Net在对齐内存开销的前提下计算速度提升67%

CR-Net: 内存和计算双高效

- 基于 LLaMA-2 1B 和 LLaMA-2 7B 模型，在 BF16 精度下，不同方法的总体理论内存和计算复杂度
- 对于 CR-Net，符号 b 代表重计算时存储的全套激活层数，其中 b = 4 (标记为 ‡) 和 b = 1 (标记为 ♯)
- CoLA-M 的秩选择基于CoLA文献，CR-Net的秩 r 选择可保证其具有更优验证集困惑度

Algorithms	LLaMA-2 1B		LLaMA-2 7B	
	Memory (GB)	FLOPs ($\times 10^{14}$)	Memory (GB)	FLOPs ($\times 10^{15}$)
Full-rank + Vanilla GCP	11.98 (1.000×)	2.133 (1.000×)	51.22 (1.000×)	2.119 (1.000×)
CoLA-M	12.04 (1.005×)	0.764 (0.358×)	24.78 (0.484×)	0.752 (0.355×)
CR-Net‡	11.81 (0.986×)	0.703 (0.330×)	23.35 (0.456×)	0.692 (0.326×)
CR-Net♯	9.94 (0.830×)	0.713 (0.334×)	22.42 (0.438×)	0.702 (0.331×)
Full-rank *w.o.* GCP	51.31 (4.283×)	0.764 (0.370×)	70.97 (1.386×)	1.608 (0.759×)

CR-Net: 扩展至GQA与MoE架构

- 基于LLaMA-3架构的1B规模模型，具有GQA结构的对比结果

	40K	80K	100K	110K
GaLore	19.29	16.89	16.47	16.40
CR-Net	18.29	16.05	15.70	15.65
Training tokens (B)	5.2	10.5	13.1	14.4

性能表现显著优于GaLore

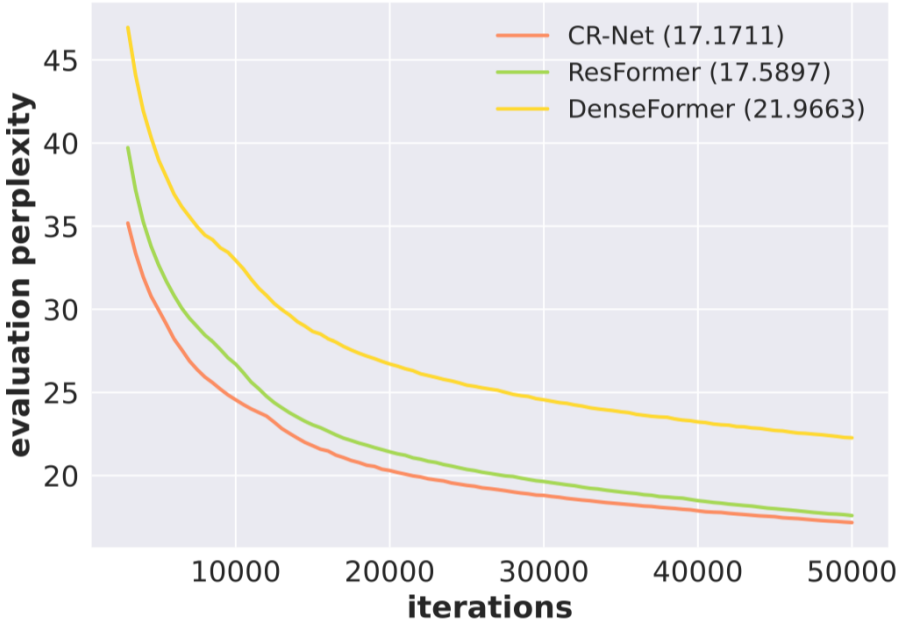
- 基于Qwen-3 MoE架构，具有1.8B参数规模和650M激活参数的对比结果

	10K	20K	30K	40K
Full-rank	24.31	20.53	18.84	17.85
CR-Net	25.99	21.73	19.32	18.12
Training tokens (B)	1.3	2.6	3.9	5.2

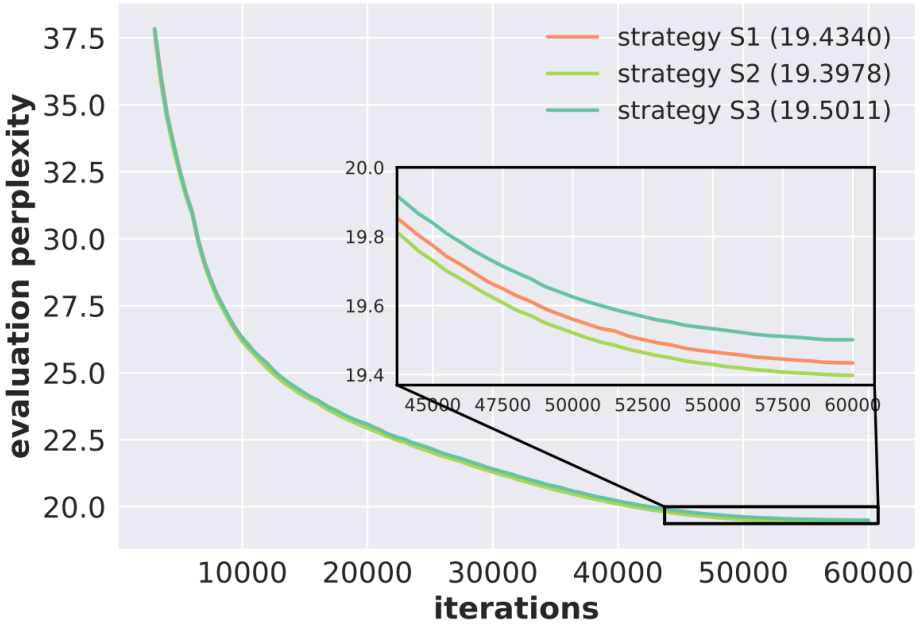
相比全秩参数训练，性能损失在1.5%以内

- CR-Net在GQA与MoE架构的模型中，依旧具有良好的性能表现

CR-Net: 消融实验



与ResFormer和DenseFormer的对比



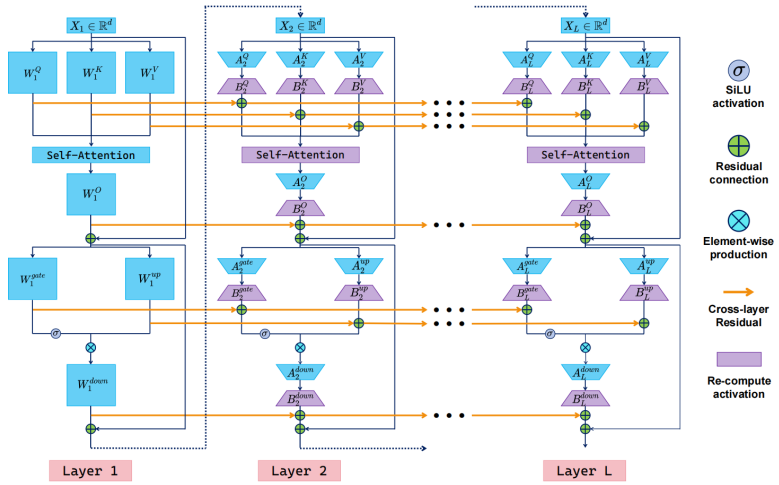
秩的不同选择

R. Tian, et. al., ResFormer: Scaling ViTs with Multi-Resolution Training, CVPR 2023

Matteo Pagliardini, et. al., DenseFormer: Enhancing Information Flow in Transformers via Depth Weighted Averaging, NeurIPS 2024

单元小结

- 参数矩阵不具有低秩结构，直接进行低秩近似会引入较大误差
- **核心思想**：利用前一层激活值进行误差补偿 (激活值层间残差具有低秩结构)；基于该思想设计出CR-Net的参数高效架构。参数高效可以节省模型、梯度和优化器状态；还能节省计算
- **重计算策略**：设计针对性重计算方法，有效节省激活值内存开销。在对齐全局内存的情况下节省66.6%的计算量



- **收益总结**：做到了内存与计算协同优化。在LLaMA-2 1B模型中，利用43.6%的参数量实现更好的模型性能，且吞吐量提升66%。在LLaMA-2 7B模型中，利用38.5%的内存实现更好的模型性能

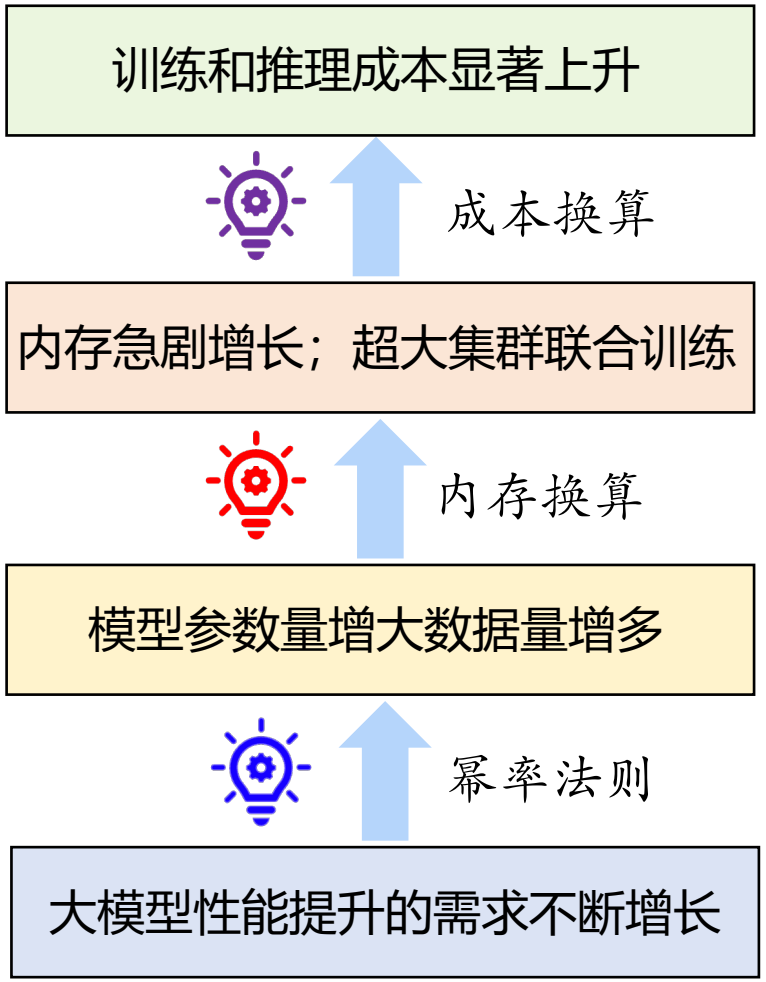
Boao Kong, Kun Yuan et al. *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure*, submitted to NeurIPS 2025

北京大学
PEKING UNIVERSITY

PART 05

总结与展望

报告总结



路线一：设计大模型架构，寻找全新的幂率准则

路线二：发展新硬件(如超节点)降低训推成本

路线三：大模型结构驱动的内存高效训练新方法

洞察：随着数量的持续增加，大模型中存在大量冗余结构
传统训练方法忽略模型特殊结构，导致内存开销巨大

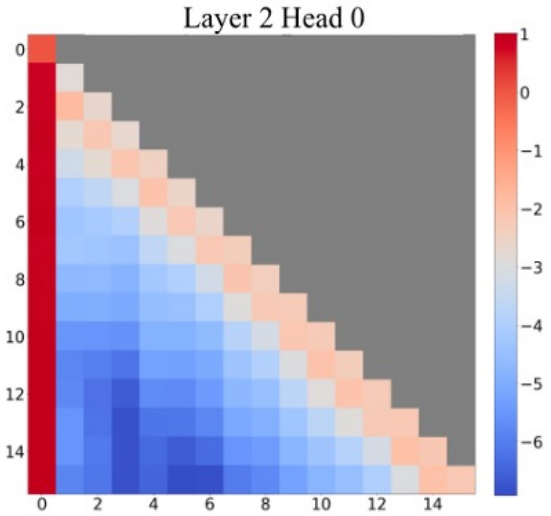
- **梯度低秩性：子空间训练方法**(优化器状态节省75%以上)
- **激活值稀疏性：重要性采样方法**(FFN激活值内存节省68%以上)
- **激活值层间低秩性： 参数高效方法**(参数和梯度内存节省56%)
- **参数分布均匀性：混合精度训练**(面向盘古模型正在进行性能测试)

模型结构挖掘与利用 + 软硬件协同设计 = 高效训练新方法

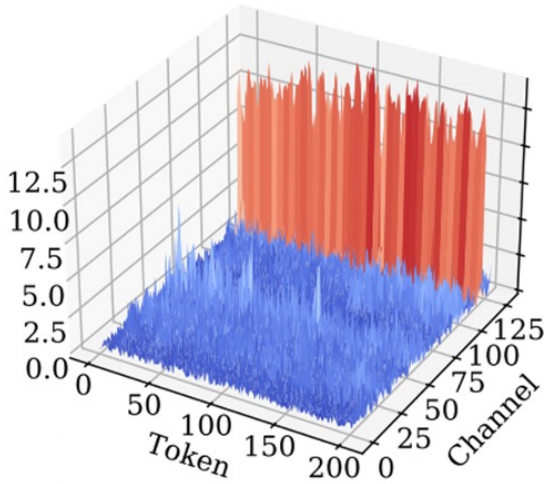
展望一：更多结构特征等待挖掘

大模型内存 = **模型参数** + **梯度** + **优化器状态**+ **激活值**

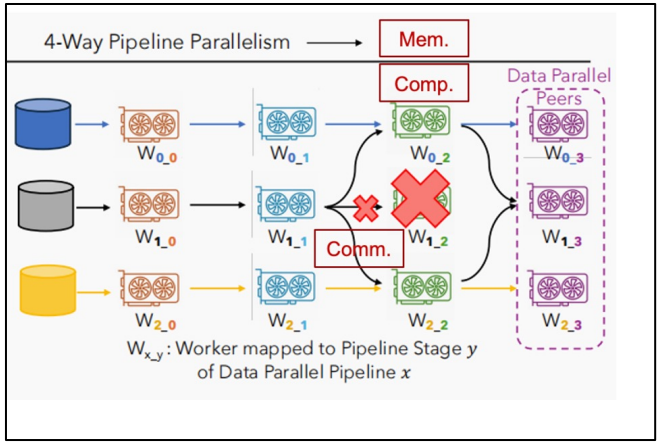
除了梯度低秩性，激活层稀疏性、均匀性以外，大量隐藏结构等待挖掘



局部稀疏 + Sink 结构
(适合节省Attention内存)



参数分布特性
(适合更优混精策略的设计)



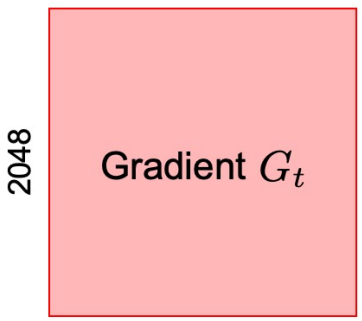
3D并行参数具有冗余性
(适合鲁棒容错训练)

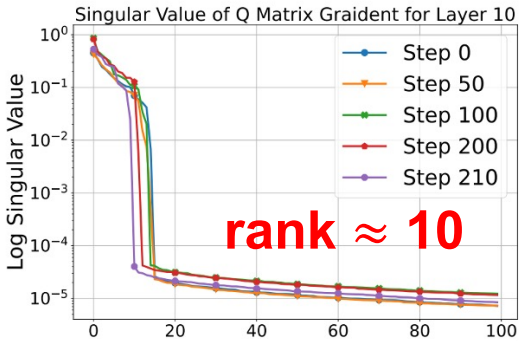
更多模型结构特征将会被挖掘，进而帮助设计更高效的训练方法

展望二：利用结构特征来节省通信和计算(不局限于节省内存)

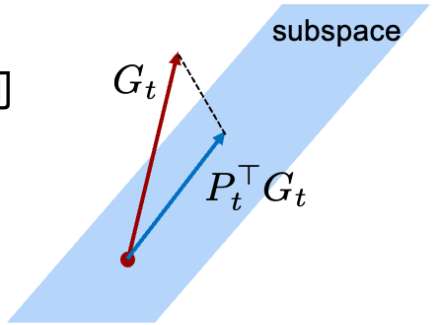
- 在大模型当中，除了内存外，计算与通信也是非常宝贵的资源
- 与节省内存类似，基于模型结构特征的新算法也可以节省通信与计算

大模型的梯度具有低秩性。将梯度进行低秩投影，在低维空间进行通信和动量更新；显著**节省通信**





投影至低维空间

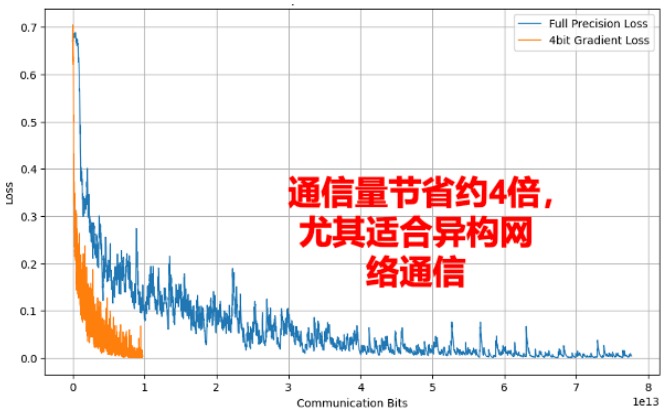


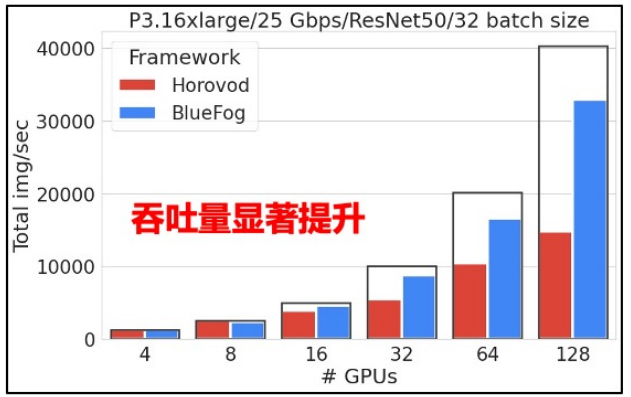
G_t (m,n) → $P_t^\top G_t$ → g_t (r,n)

$$
\begin{aligned}
\boldsymbol{g}_t &= \boldsymbol{P}_t^\top \boldsymbol{G}_t \qquad \boldsymbol{g}_t \leftarrow (1/n)\textstyle\sum_{i=1}^n \boldsymbol{g}_{t,i} \\
\boldsymbol{m}_t &= (1-\beta_1)\boldsymbol{m}_{t-1} + \beta_1 \boldsymbol{g}_t \\
\boldsymbol{v}_t &= (1-\beta_2)\boldsymbol{v}_{t-1} + \beta_2 \boldsymbol{g}_t \odot \boldsymbol{g}_t \\
\boldsymbol{\delta}_t &= \frac{\gamma}{\sqrt{\boldsymbol{v}_t}+\epsilon} \odot \boldsymbol{m}_t \\
\boldsymbol{X}_{t+1} &= \boldsymbol{X}_t - \boldsymbol{P}_t \boldsymbol{\delta}_t
\end{aligned}
$$

低维空间变量
通信占用很少

低维空间变量
显存占用很少





Chuyan Chen, Kun Yuan, et. al., Greedy Low-Rank Gradient Compression for Distributed Learning with Convergence Guarantees, arXiv: 2507.08784, 2025

展望二：利用结构特征来节省通信和计算(不局限于节省内存)

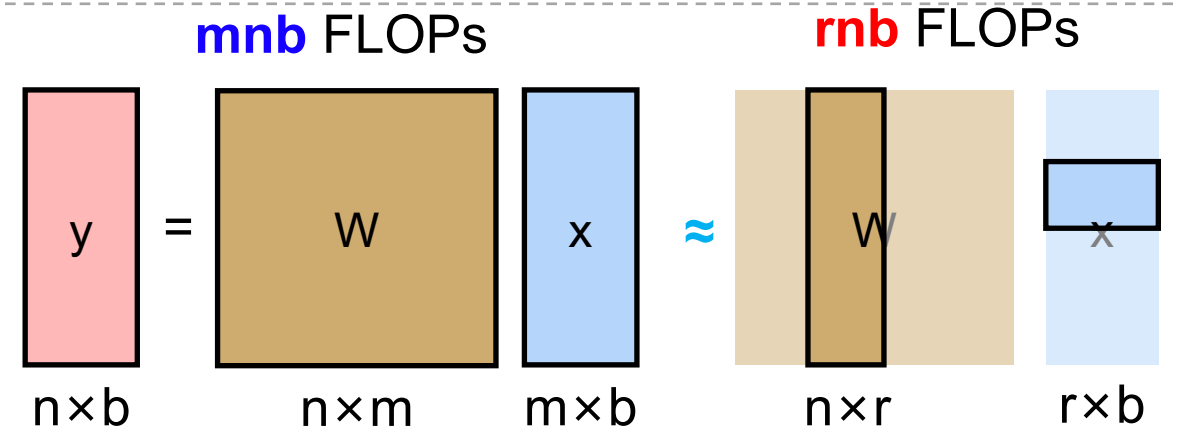
研究动机：当前的内存高效预训练/微调算法无法**节省**大模型的**计算量**；它们只关注内存

算法设计：**随机矩阵采样**；避免大矩阵与大矩阵的相乘；采矩阵的重要列与重要行进行相乘

正向传播： $y = xW_0 + xBA = x(\overline{W} + B_0A_0) + xBA$

反向传播： $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\overline{W}^T + \frac{\partial L}{\partial y}(B_0A_0)^T + \frac{\partial L}{\partial y}(BA)^T$

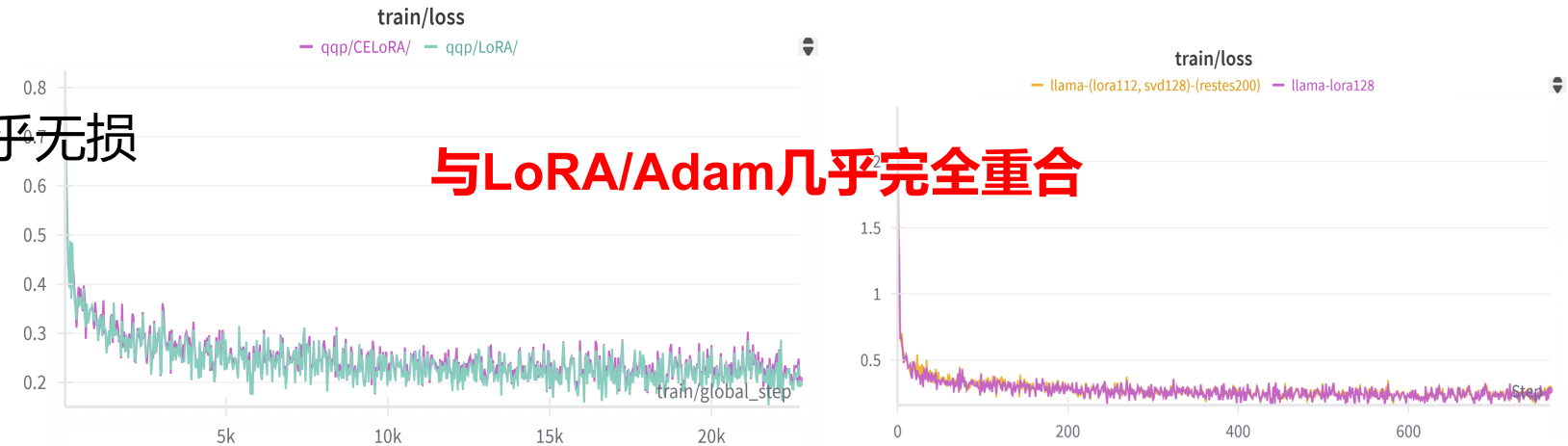
矩阵采样： $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}[:,i]\overline{W}^T[i,:] + \frac{\partial L}{\partial y}(B_0A_0)^T + \frac{\partial L}{\partial y}(BA)^T$



MLP采样90%, attention采样40%

合计节省计算量**38.57%** 精度几乎无损

未来展望：在预训练时，计算量节省30%以上且精度几乎无损



与LoRA/Adam几乎完全重合

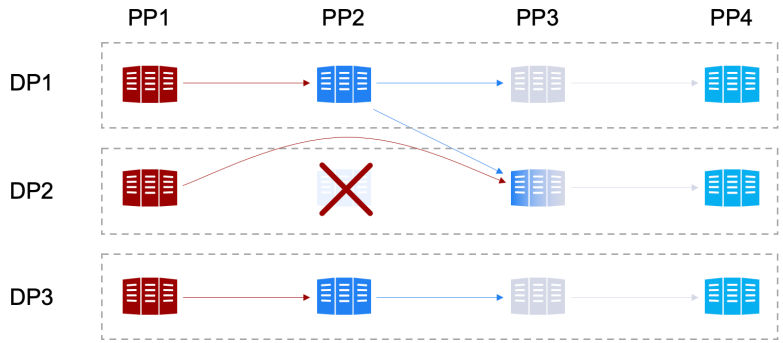
G Chen, Y He, Y Hu, K Yuan, B Yuan, CE-LoRA: Computation-Efficient LoRA Fine-Tuning for Language Models, arXiv 2502.01378, 2025

展望二：利用结构特征来节省通信和计算(不局限于节省内存)

研究动机：25年集群规模会由万卡扩增至10万卡，涉及器件百万甚至千万级别，单个器件的故障可能导致整个训练任务中断，集群可用度存在大的挑战。

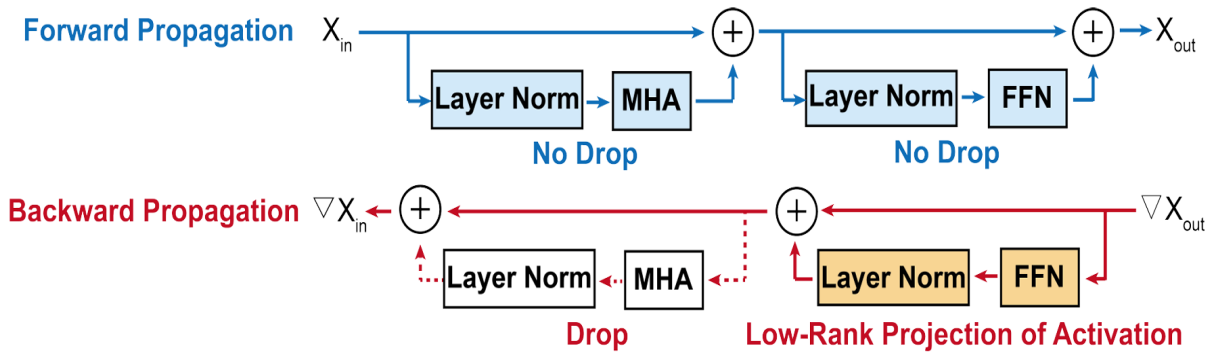
厂商	模型	MTBF	可用度	十万卡可用度估测
字节	175B LLM	8~15小时 故障100+次	~90%	~60%
Meta	Llama3.1 405B	~4小时 故障419次	~90%	

算法提出：高效弹性训练方案设计，支持掉卡场景下的无停滞连续训练，引入**混合精度中参数冗余**的结构，从邻居拉取参数备份。



邻居节点需要同时处理两份流水线任务，导致内存压力和计算瓶颈

内存和计算平衡: MHA层的激活值占据了显存使用的主要部分，**丢弃激活值**可大幅节省GPU显存



Center of Machine Learning Research

R. Hu, K. Yuan, et. al., MeCeFO: Enhancing LLM Training Robustness via Fault-Tolerant Optimization, Submitted to NeurIPS 2025, 2025

展望二：利用结构特征来节省通信和计算(不局限于节省内存)

北京大学 PEKING UNIVERSITY

Table 2: Throughput Performance and Degradation under Different Fault Frequencies

Model	System	Throughput (tokens/s)				Throughput Drop (%)		
		No Fault	Low Freq.	Mid Freq.	High Freq.	Low Freq.	Mid Freq.	High Freq.
LLaMA-350M	Bamboo	438.06k	428.90k	421.45k	407.22k	2.09	3.79	7.04
	Oobleck	703.73k	674.15k	662.93k	632.40k	4.20	5.80	10.14
	MeCeFO	**1199.23k**	**1197.39k**	**1193.25k**	**1186.35k**	**0.15**	**0.50**	**1.07**
LLaMA-1B	Bamboo	153.75k	146.91k	144.66k	141.13k	4.45	5.91	8.21
	Oobleck	291.05k	276.05k	268.29k	250.68k	5.16	7.82	13.87
	MeCeFO	**471.19k**	**464.79k**	**461.23k**	**457.13k**	**1.36**	**2.11**	**2.98**
LLaMA-7B	Bamboo	12.41k	11.45k	10.74k	9.82k	7.73	13.42	20.84
	Oobleck	66.95k	57.05k	51.63k	48.14k	14.78	22.87	28.09
	MeCeFO	**111.12k**	**108.15k**	**107.70k**	**106.47k**	**2.67**	**3.08**	**4.18**

MeCeFO在各故障频率场景下训练相同Iteration，LLama模型在验证集的PPL

Model	No Fault	Low-frequency Fault	Medium-frequency Fault	High-frequency Fault
Llama-350M	18.74	18.75	18.88	19.04
Llama-1B	15.49	15.51	15.61	15.83
Llama-7B	14.92	14.97	15.04	15.16

R. Hu, K. Yuan, et. al., MeCeFO: Enhancing LLM Training Robustness via Fault-Tolerant Optimization, Submitted to NeurIPS 2025, 2025

展望三：具有更优幂率准则的模型架构设计

北京大学 PEKING UNIVERSITY

训练和推理成本显著上升

成本换算

内存急剧增长；超大集群联合训练

内存换算

模型参数量增大数据量增多

幂率法则

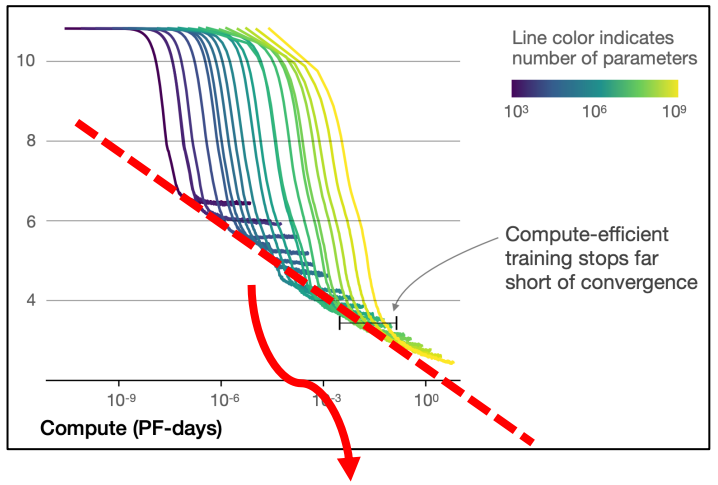
大模型性能提升的需求不断增长

路线一：设计大模型架构，寻找全新的幂率准则

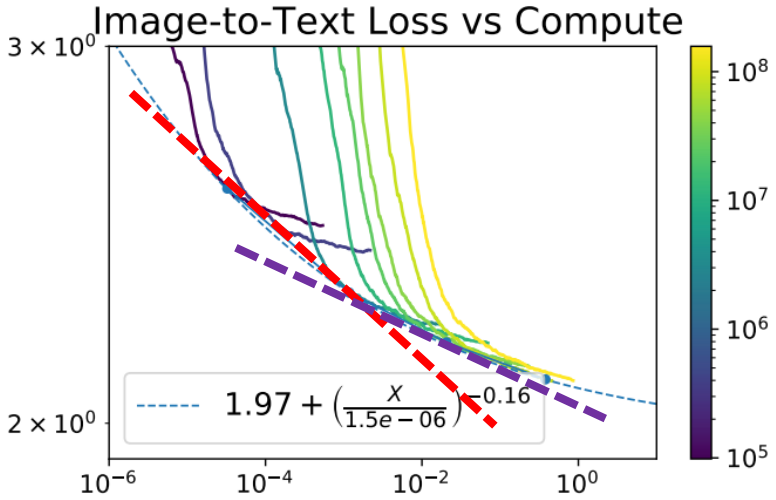
路线二：发展新硬件(如超节点)降低训推成本

路线三：大模型结构驱动的内存高效训练新方法

改进幂率法则是提升大模型训练效率更为本质的研究方向



无法跨越的Transformer性能墙



斜率甚至越来越低

展望三：具有更优幂率准则的模型架构设计

训练和推理成本显著上升

成本换算

内存急剧增长；超大集群联合训练

内存换算

模型参数量增大数据量增多

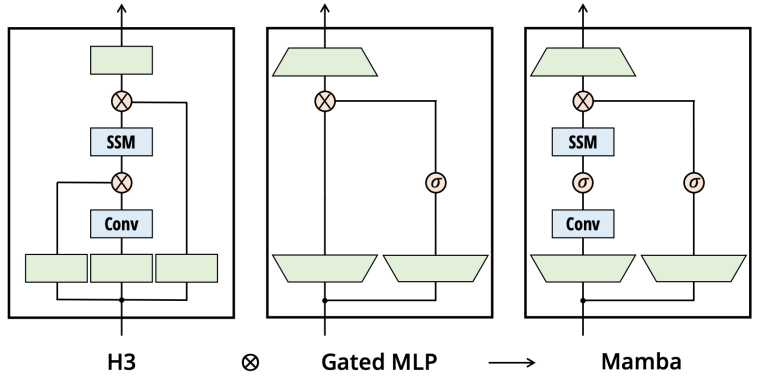
幂率法则

大模型性能提升的需求不断增长

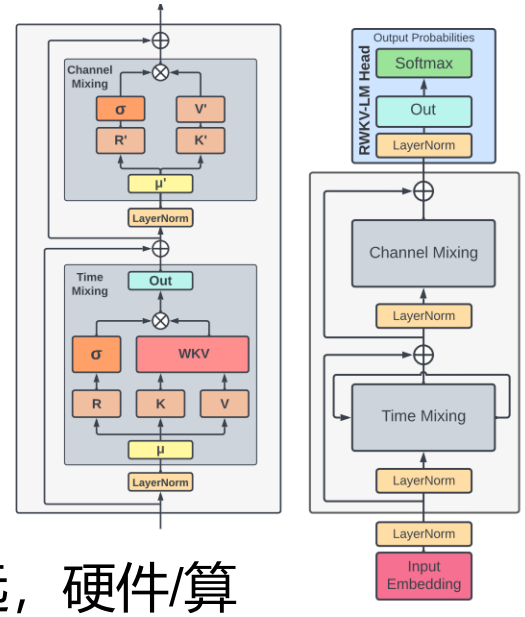
路线一：设计大模型架构，寻找全新的幂率准则

未来5~10年大模型架构需要进行本质性的变革

Mamba?



RWKV?

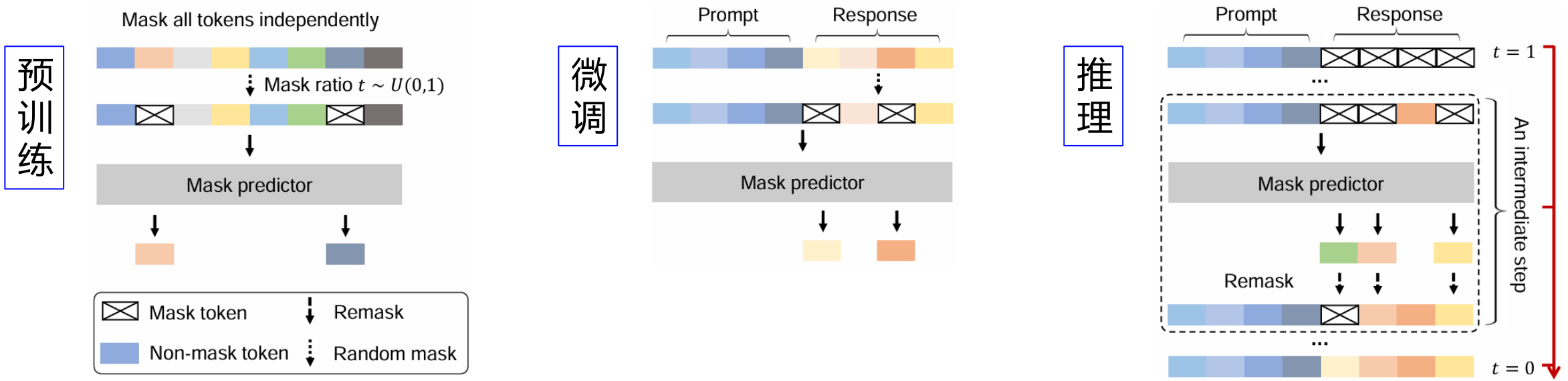


我们已经在Transformer架构上走得太远，硬件/算法/理论都深度绑定Transformer架构

要有勇气探索一条全新的道路

大模型结构设计：扩散语言模型

- 使用**diffusion**代替传统自回归模式[1]：

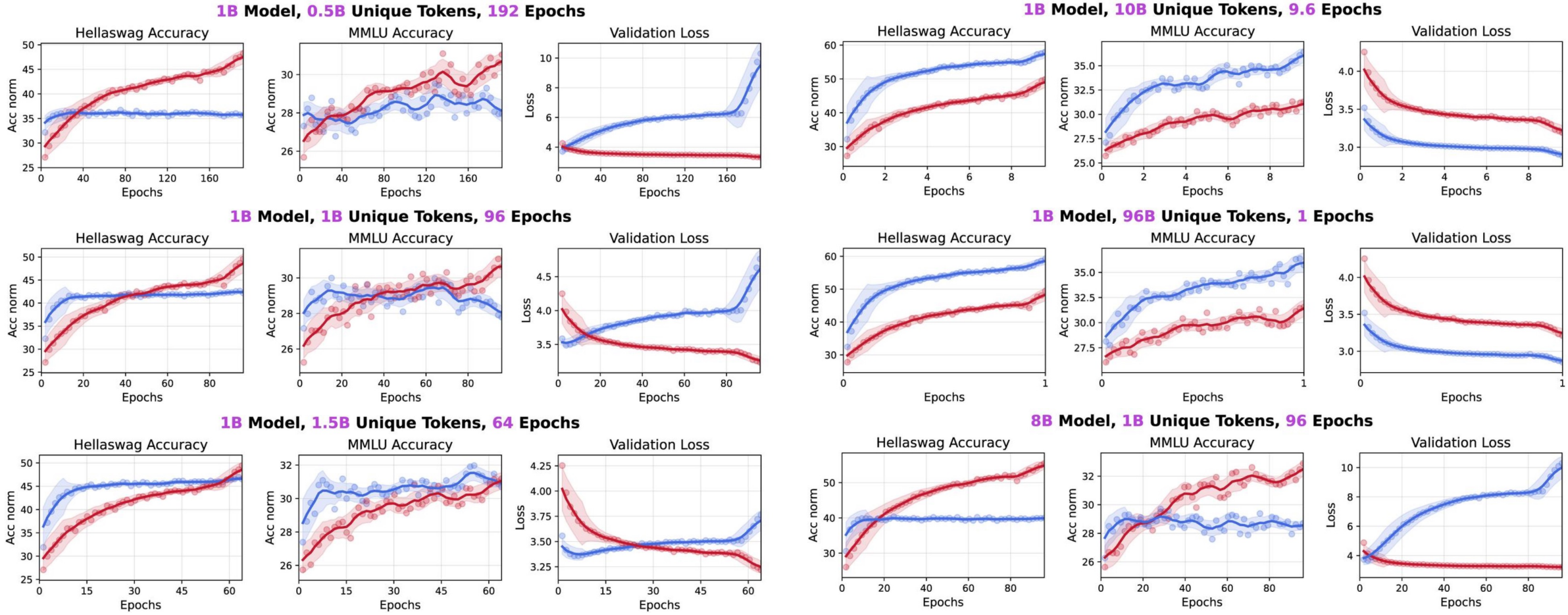


- 多个mask token同时采样，利好**可并行性**与**推理速度**
- 同一条数据多种 mask 方式，**数据利用率**高

[1] Shen Nie et al., *Large Language Diffusion Models*, ICLR 2025

大模型结构设计：扩散语言模型

- **同等数据**下，训练足够多epoch时，**Diffusion 模型总能超过自回归模型**[1]



[1] Jinjie Ni et al., *Diffusion Language Models are Super Data Learners*, https://jinjieni.notion.site/Diffusion-Language-Models-are-Super-Data-Learners-239d8f03a866800ab196e49928c019ac

大模型结构设计：线性核注意力

北京大学 PEKING UNIVERSITY

Transformer 模型

- ✔ 训练高效：支持并行
- ✘ 推理低效：KVcache 占用$O(s)$内存

RNN 模型

- ✘ 训练低效：不支持并行
- ✔ 推理高效：$O(1)$内存占用

Transformer 训练模式

如何设计核函数，达到近似exp的表达能力?

RNN 推理模式

Exp 激活

核函数激活

$\exp(QK^\top)$

$\phi(Q)\phi(K)^\top$

Kernelized Linear Attention[1]

- ✔ 训练高效：与Transformer一致
- ✔ 推理高效：可无损转化为RNN

$$\frac{\sum \phi(q)\phi(k_i)^\top v_i}{\sum \phi(q)\phi(k_i)^\top}$$

$$\frac{\phi(q)[\sum \phi(k_i)^\top v_i]}{\phi(q)[\sum \phi(k_i)^\top]}$$

训练形式

推理形式

[1] Angelos Katharopoulos et al., *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*, ICML 2020

大模型结构设计：线性核注意力

北京大学 PEKING UNIVERSITY

- **同等规模**下，RNN类结构具有比肩 Transformer、RWKV、Mamba 等结构的能力[1]

	Model	#Params M	SlimPajama (300B) ppl ↓	LAMBADA ppl ↓	LAMBADA acc ↑	HellaSwag acc ↑	PIQA acc ↑	ARC-E acc ↑	ARC-C acc ↑	WinoGrande acc ↑	Average acc ↑
125M	RWKV-4	169.4	16.66	54.72	23.77	34.03	66.00	47.94	24.06	50.91	41.12
	Llama	162.2	15.89	39.21	31.54	34.09	65.45	45.33	23.63	50.67	41.78
	Mamba	167.8	15.08	27.76	34.14	36.47	66.76	**48.86**	24.40	51.14	43.63
	xLSTM[1:0]	163.8	14.63	**25.98**	**36.52**	36.74	65.61	47.81	24.83	**51.85**	43.89
	xLSTM[7:1]	163.7	**14.60**	26.59	36.08	**36.75**	**66.87**	48.32	**25.26**	51.70	**44.16**
350M	RWKV-4	430.5	12.62	21.57	36.62	42.47	69.42	54.46	25.43	51.22	46.60
	Llama	406.6	12.19	15.73	44.19	44.45	69.15	52.23	26.28	53.59	48.32
	Mamba	423.1	11.64	12.83	46.24	47.55	69.70	55.47	27.56	54.30	50.14
	xLSTM[1:0]	409.3	**11.31**	**11.49**	**49.33**	**48.06**	69.59	55.72	26.62	**54.38**	50.62
	xLSTM[7:1]	408.4	11.37	12.11	47.74	47.89	**71.16**	**56.61**	27.82	53.28	**50.75**
760M	RWKV-4	891.0	10.55	10.98	47.43	52.29	72.69	58.84	28.84	55.41	52.58
	Llama	834.1	10.60	9.90	51.41	52.16	70.95	56.48	28.75	56.67	52.74
	Mamba	870.5	10.24	9.24	50.84	53.97	71.16	60.44	29.78	56.99	53.86
	xLSTM[1:0]	840.4	**9.86**	8.09	54.78	55.72	72.69	**62.75**	**32.59**	**58.17**	**56.12**
	xLSTM[7:1]	839.7	9.91	**8.07**	**55.27**	**56.12**	**72.74**	61.36	29.61	56.43	55.26
1.3B	RWKV-4	1515.2	9.83	9.84	49.78	56.20	74.70	61.83	30.63	55.56	54.78
	Llama	1420.4	9.44	7.23	57.44	57.81	73.12	62.79	31.74	59.04	56.99
	Mamba	1475.3	9.14	7.41	55.64	60.45	74.43	**66.12**	**33.70**	60.14	58.41
	xLSTM[1:0]	1422.6	**8.89**	**6.86**	**57.83**	**60.91**	74.59	64.31	32.59	**60.62**	**58.48**
	xLSTM[7:1]	1420.1	9.00	7.04	56.69	60.26	**74.92**	65.11	32.34	59.27	58.10

[1] Maximilian Beck et al., *xLSTM: Extended Long Short-Term Memory*, NeurIPS 2024

相关文献

[ICML 2025] Y. He, P. Li, Y. Hu, C. Chen, and **K. Yuan***, “Subspace Optimization for Large Language Models with Convergence Guarantees”, International Conference on Machine Learning (ICML), 2025.

[ICML 2025] Y. Chen, Y. Zhang, Y. Liu, **K. Yuan***, and Z. Wen, “A Memory Efficient Randomized Subspace Optimization Method for Training Large Language Models”, International Conference on Machine Learning (ICML), 2025.

[ICML 2025] Y. Song, P. Li, B. Gao, and **K. Yuan***, “Distributed Retraction-Free and Communication-Efficient Optimization on the Stiefel Manifold”, International Conference on Machine Learning (ICML), 2025.

[ICML 2025] L. Liang, G. Luo, X. Chen, and **K. Yuan***, “Achieving Linear Speedup and Optimal Complexity for Decentralized Optimization over Row-stochastic Networks”, International Conference on Machine Learning (ICML), 2025.

[ICML 2025] L. Chen, Q. Xiao, E. H. Fukuda, X. Chen, **K. Yuan**, and T. Chen, “Efficient Multi-Objective Learning under Preference Guidance: A First-Order Penalty Approach”, International Conference on Machine Learning (ICML), 2025.

[ICLR 2025] Y. Chen, Y. Zhang, L. Cao, **K. Yuan***, and Z. Wen, “Enhancing Zeroth-Order Fine-Tuning for Language Models with Low-Rank Structures”, International Conference on Learning Representations (ICLR), 2025.

[NeurIPS 2024] S. Zhu, B. Kong, S. Lu, X. Huang, and **K. Yuan***, “SPARKLE: A Unified Single-Loop Primal- Dual Framework for Decentralized Bilevel Optimization”, Advances in Neural Information Processing Systems (NeurIPS), 2024

[ICML 2024] Y. He, J. Hu, X. Huang, S. Lu, B. Wang, and **K. Yuan***, “Distributed Bilevel Optimization with Communication Compression”, International Conference on Machine Learning (ICML), 2024.

相关文献

[NeurIPS 2023] Y. He, X. Huang, and **K. Yuan***. “Unbiased Compression Saves Communication in Distributed Optimization: When and How Much?”, Advances in Neural Information Processing Systems (NeurIPS), 2023.

[ICML 2023] L. Ding, K. Jin, B. Ying, **K. Yuan**, and W. Yin. “DSGD-CECA: Decentralized SGD with Communication-Optimal Exact Consensus Algorithm”, The International Conference on Machine Learning (ICML), 2023.

[NeurIPS 2022] X. Huang, Y. Chen, W. Yin, and **K. Yuan***, “Lower Bounds and Nearly Optimal Algorithms in Distributed Learning with Communication Compression”, Neural Information Processing Systems (NeurIPS), 2022.

[NeurIPS 2022] Z. Song, W. Li, K. Jin, L. Shi, M. Yan, W. Yin, and **K. Yuan***, “Communication-Efficient Topologies for Decentralized Learning with O(1) Consensus Rate”, Neural Information Processing Systems (NeurIPS), 2022.

[NeurIPS 2022] **K. Yuan*,** X. Huang, Y. Chen, X. Zhang, Y. Zhang, and P. Pan, “Revisiting Optimal Convergence Rate for Smooth and Non-Convex Stochastic Decentralized Optimization”, Neural Information Processing Systems (NeurIPS), 2022

Preprint and New Submissions:

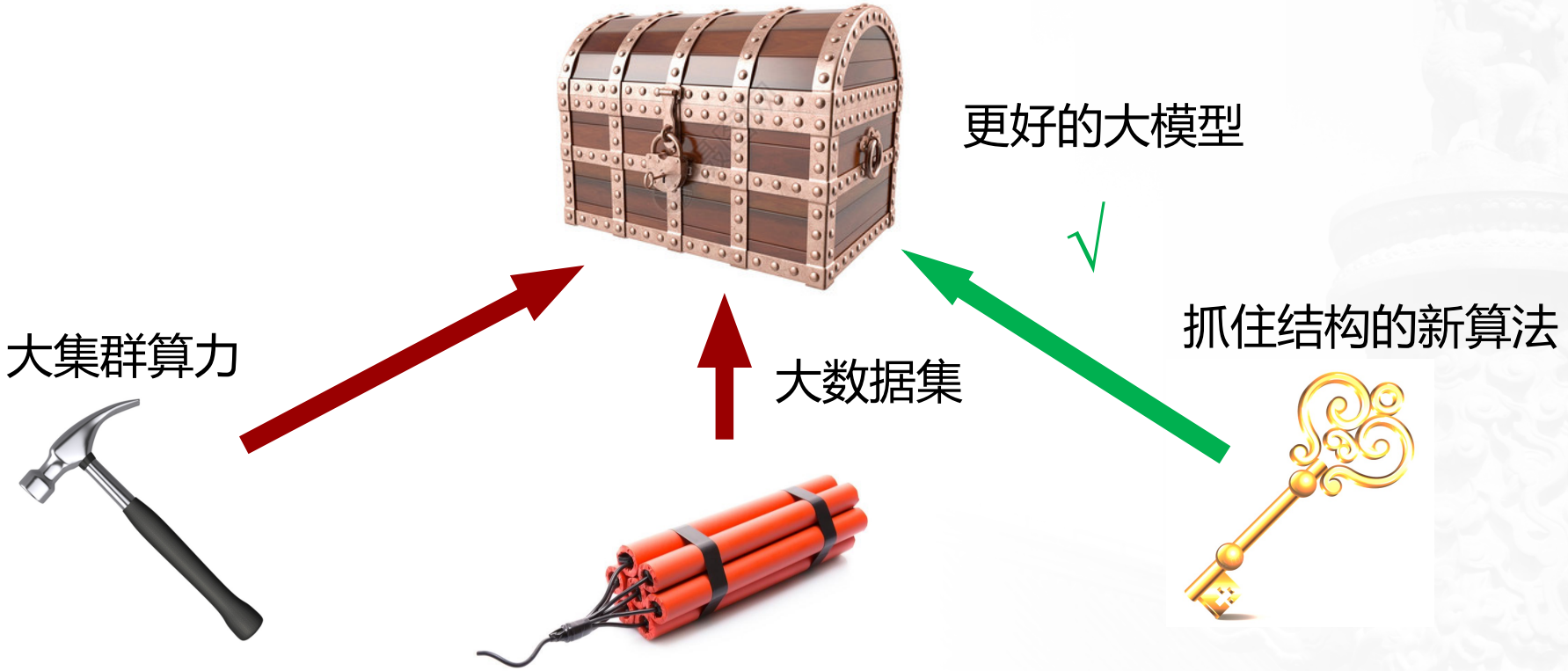
T. Wu, Y. He, B. Wang, and **K. Yuan***, *Mixture-of-Channels: Exploiting Sparse FFNs for Efficient LLMs Pre-Training and Inference, 2025*

B. Kong, J. Liang, Y. Liu, R. Deng, and **K. Yuan***, *CR-Net: Scaling Parameter-Efficient Training with Cross-Layer Low-Rank Structure, 2025*

B. Kong, X. Huang, Y. Xu, Y. Liang, B. Wang, and **K. Yuan***, *Clapping: Removing Per-sample Storage for Pipeline Parallel Learning with Communication Compression, 2025*

R. Hu, Y. He, R. Yan, M. Sun, B. Yuan, **K. Yuan***, *MeCeFO: Enhancing LLM Training Robustness via Fault-Tolerant Optimization, 2025*

C Chen, Y He, P Li, W Jia, **K Yuan***, Greedy Low-Rank Gradient Compression for Distributed Learning with Convergence Guarantees, arXiv:2507.08784, 2025



谢谢！

Kun Yuan homepage: https://kunyuan827.github.io/