天津大学
Tianjin University

大模型轻量化技术

张 鹏

2024.8.24

目录

语言模型背景

- **自然语言处理**是国家重大战略需求

国务院新一代人工智能发展规划

8. 自然语言处理技术。研究短文本的计算与分析技术，跨语言文本挖掘技术和面向能的语义理解技术，多媒体信息理解的人机对话系统。

专栏2 关键共性技术
自然语言处理技术

自然语言处理技术。重点突破自然语言的语法逻辑、字符概念表征和深度语义交互，实现多风格多语言多领域的自然语言智能理解和自动生成。

重点突破**自然语言**的语法逻辑、字符概念表征和深度语义分析的核心技术

- **语言模型**是自然语言处理任务中的核心技术，大语言模型的发展取得突破性进展

自然语言处理

信息检索

Chat GPT

数字媒宣

多模态内容理解

算力资源消耗大

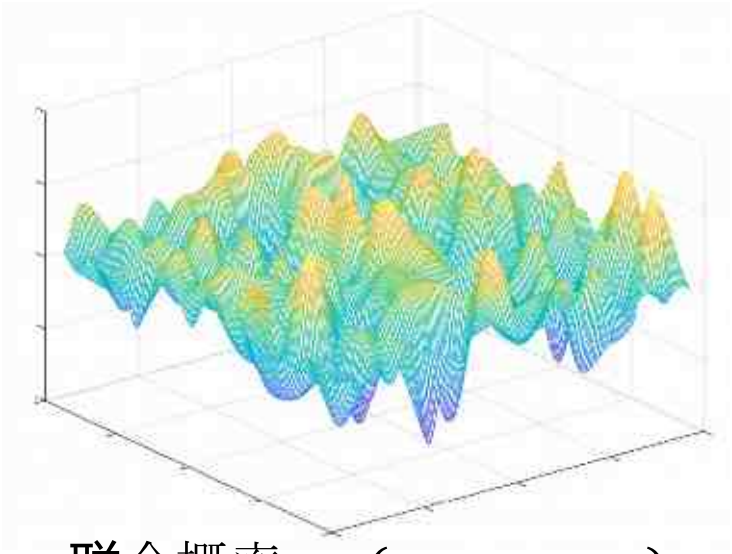
可解释性差

……

语言模型的发展历史和关键难题

天津大学 Tianjin University

如何构建
语义概率空间



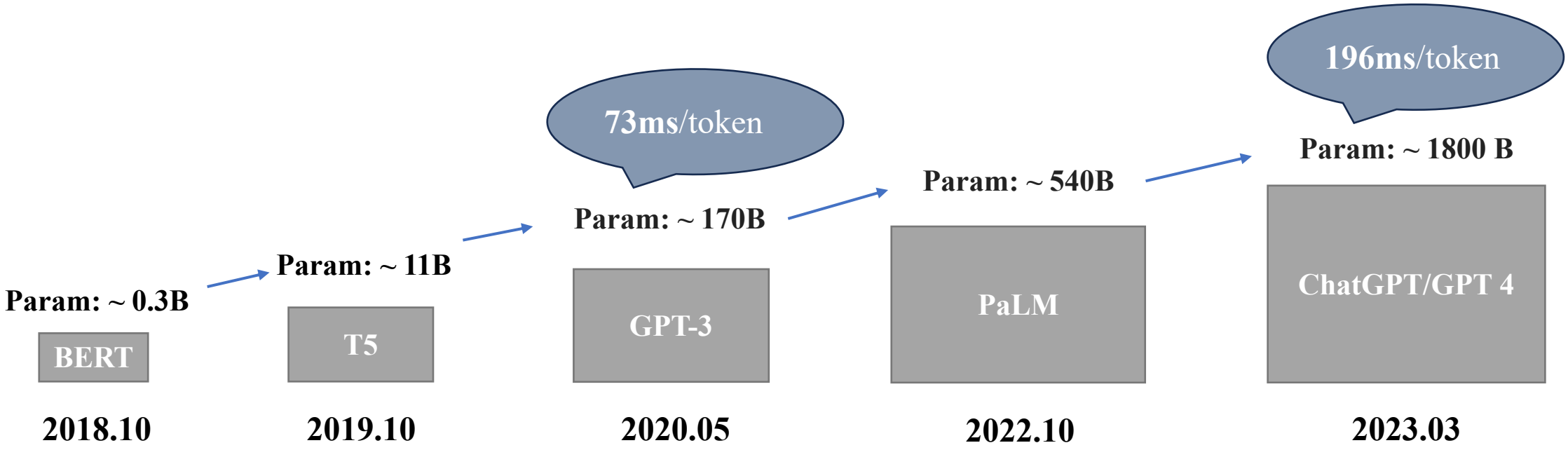
联合概率：$p(w_1, w_2 \ldots w_i)$

条件概率：$p(w_i | w_1 \ldots w_{i-1})$

发展阶段	关键难题
统计语言模型 (N-gram)	语义计算 维数灾难
神经语言模型 (CNN、RNN)	模型设计 不可解释
大型预训练语言模型 (GPT)	应用部署 参数量大

总体思路：用轻量化的方式解决大模型实际应用部署过程中遇到的问题

大模型参数规模



大语言模型**涌现**

但是

高存储成本和计算成本　　**推理速度受限**

大模型轻量化目标

预训练语言模型

轻量化技术

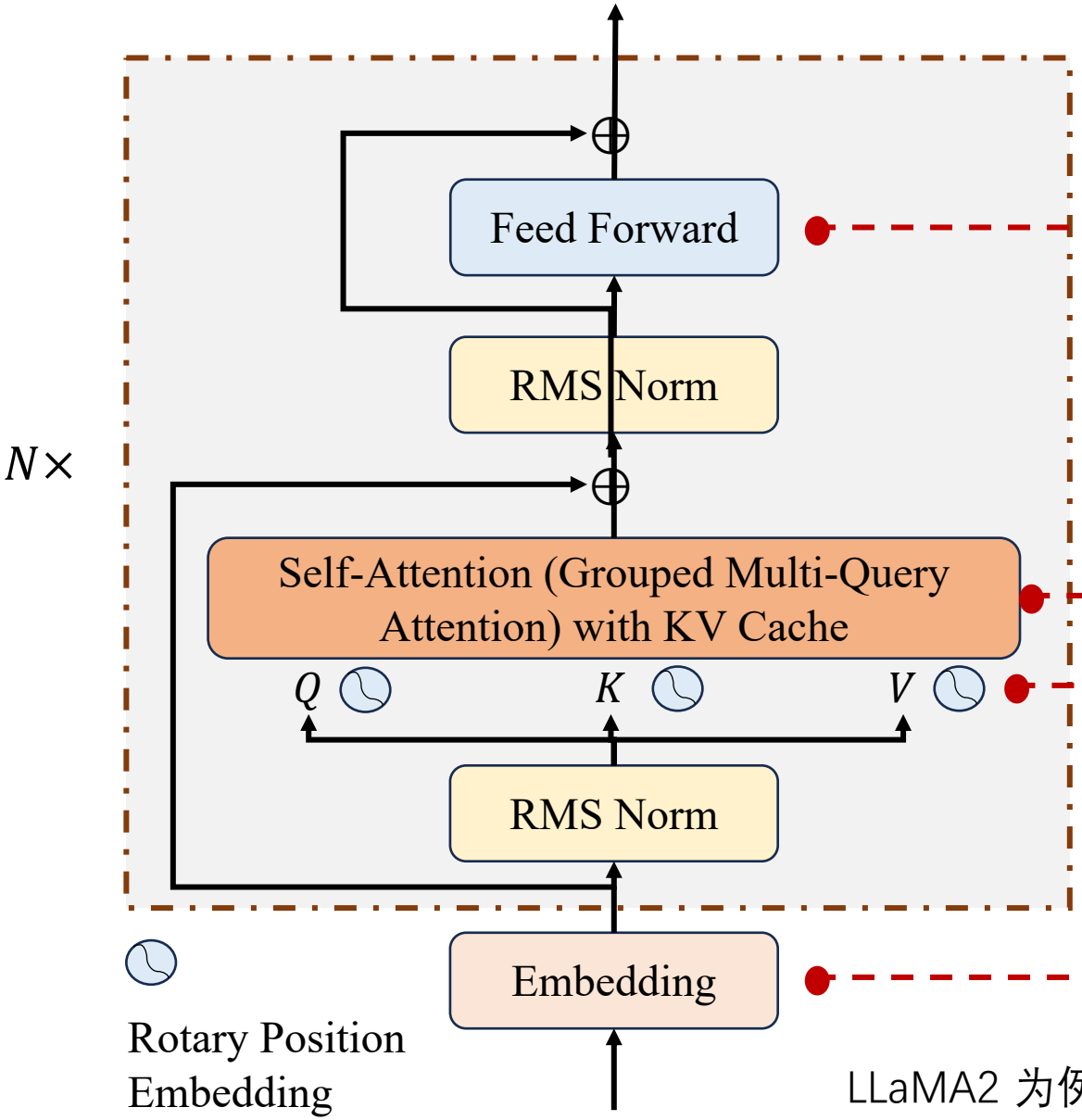
压缩

压缩后的预训练语言模型

体积更小

跑的更快

大模型轻量化的细粒度解析



LLaMA2 为例

- **参数量占比较大**，对存储及显存计算造成压力
- 多头注意力计算造成大量的计算成本，**影响计算速度，参数量占比较大**。此外，KV Cache部分使用空间换取时间，造成缓存压力。
- QKV作为中间表示存于内存中，也会**对存储造成压力**
- Embedding层，语义表示的初始化，**影响效果，占据一定的参数量**

大模型轻量化技术的意义

大模型轻量化技术为模型在实际应用和发展中带来更多便利和机遇

手机端侧大模型

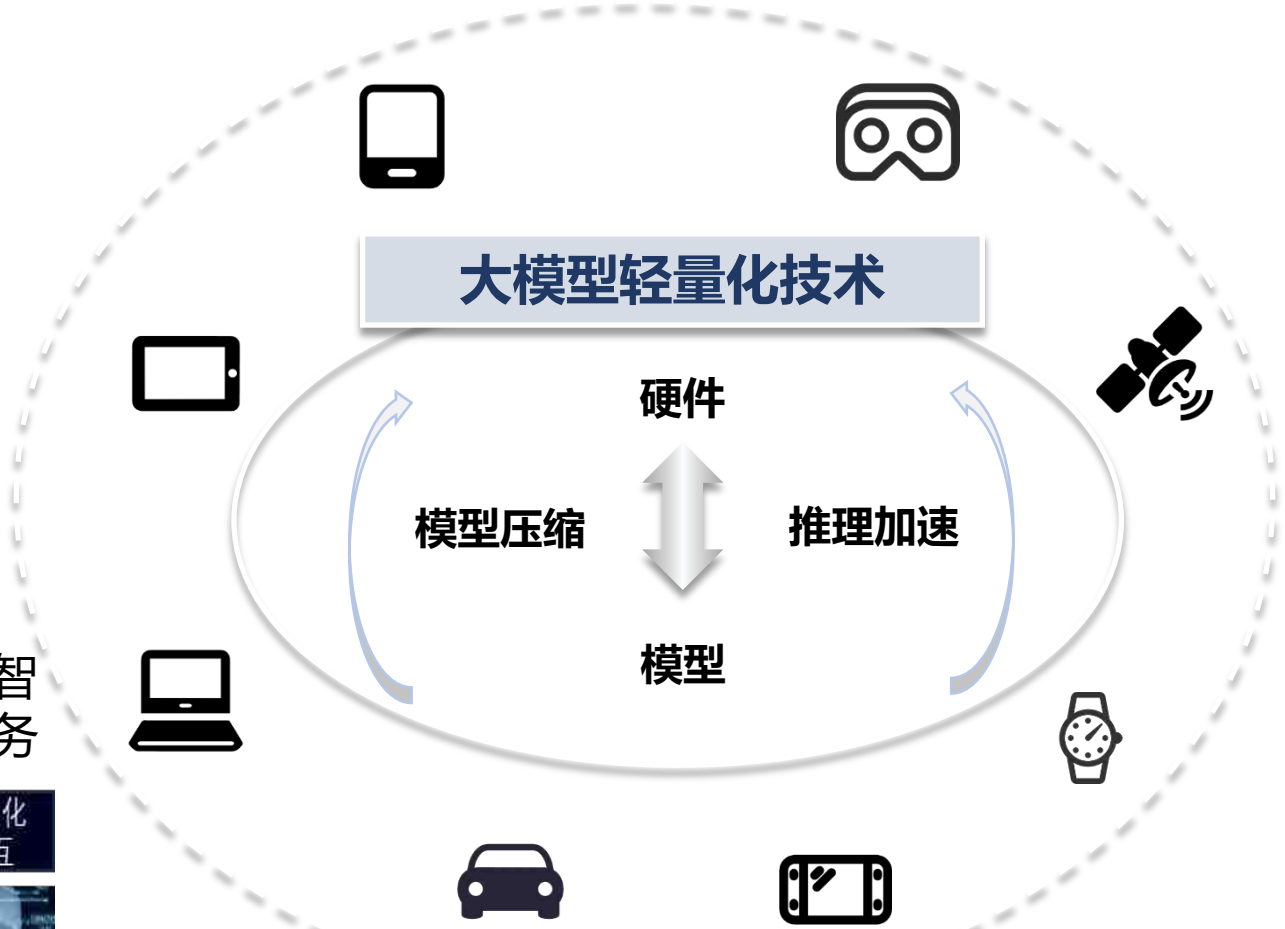
应用：将大模型应用于移动端，进行家居控制


智能家居


手机应用

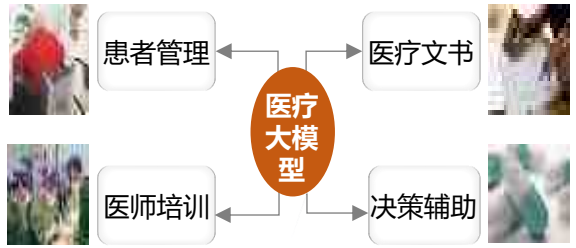
智能驾驶舱

应用：将大模型应用于智能车仓，提升个性化服务



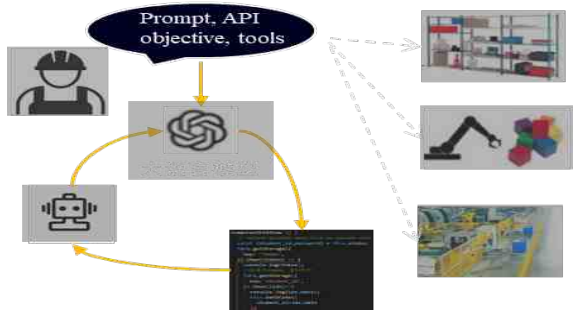
医疗大模型

应用：辅助医疗



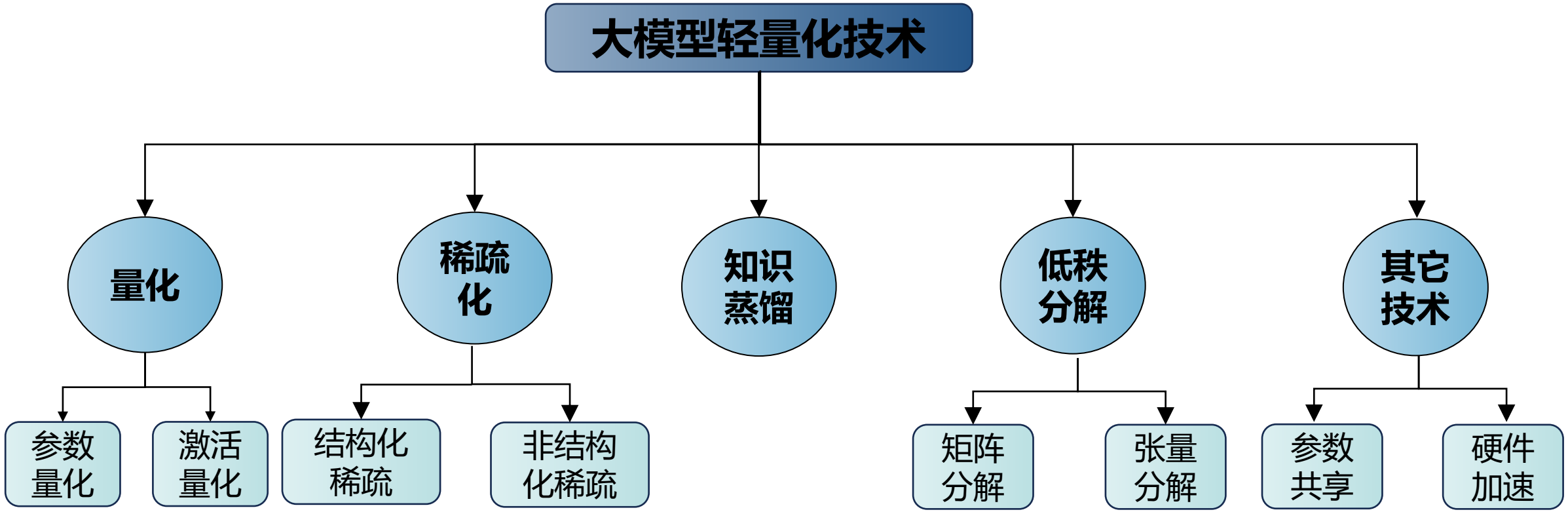
工业大模型

应用：解决生产效率问题等



目录

轻量化技术总览



目标	减少计算量	减少计算量	保留泛化能力	减少模型参数量	高效训练推理加速

减少模型计算复杂度和内存占用，同时尽可能保持性能和泛化能力

轻量化相关理论

◆ 轻量化的优化目标

模型	参数数量	模型大小
LLaMA3-8B	8B	>16GB
LLaMA3-70B	70B	>145GB

- **降低参数数量**

  更多的参数数量通常意味着模型更复杂

  通过提高参数压缩比，可以降低存储和计算需求

- **减少占用存储空间大小**

  模型参数越多，模型文件需要的存储空间越大

  压缩存储空间可以降低部署成本，提高模型在存储设备上的传输效率

- **降低浮点运算数（FLOPs）**

  模型参数越多，通常意味着在一次前向传播中所需的浮点运算数量越多

  降低FLOPs可以为模型带来更快的推理速度

轻量化相关理论

◆ 轻量化模型减轻硬件压力

- **显存（GPU Memory）**

  用于存储训练、推理中的模型参数、梯度和激活值

  减少显存占用可降低对显卡设备的要求，增加训练批次大小，减少训练时间。

- **带宽（Bandwidth）**

  代表数据在处理器和内存之间的传输速度

  降低带宽占用可以减少因数据传输带来的延迟，提高计算速度。

- **内存（RAM）**

  用于存储训练数据、模型参数和中间计算结果

  降低内存空间需求可以减少磁盘交换操作，提升训练效率。

性能有限设备上LLM难以部署

	RTX4080m 12G
架构	Ada Lovelace
核心代号	AD104
制程	台积电4nm
晶体管数量	358亿
芯片面积	295mm²
SM数量	58
CUDA数量	7424
ROPS	80
Tensor Core数量	232
RT Core数量	58
L2缓存	48MB
游戏频率	2410MHz
显存容量和位宽	12GB 192bits
显存频率	16Gbps
显存带宽	384GB/s
TGP	150W

轻量化相关理论

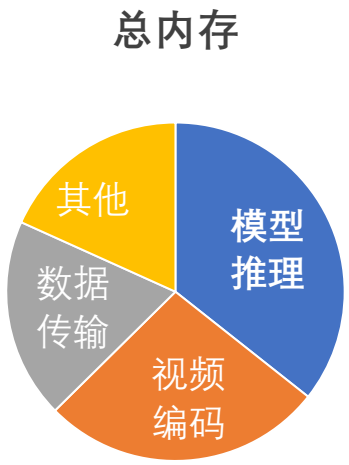
◆ 轻量化模型评估指标

参数压缩比（Compression Rate）：轻量化后模型的参数占原始参数的比例

➢ 内存占用（Memory Footprint）

模型在运行过程中占用的内存大小。

较小的内存占用有助于在内存受限的设备上高效运行模型。



➢ 吞吐量（Throughput）

单位时间内模型输出token的数量

高吞吐量表示模型能够更高效地处理大批量数据,适用于需要高处理能力的应用。

$$sample_{throughput} = \frac{V_u * R}{T}$$

虚拟用户个数

每个虚拟用户请求生成的Token数

服务所用的总时间

轻量化相关理论

◆ 轻量化模型评估指标

- 推理速度（Inference Speed）

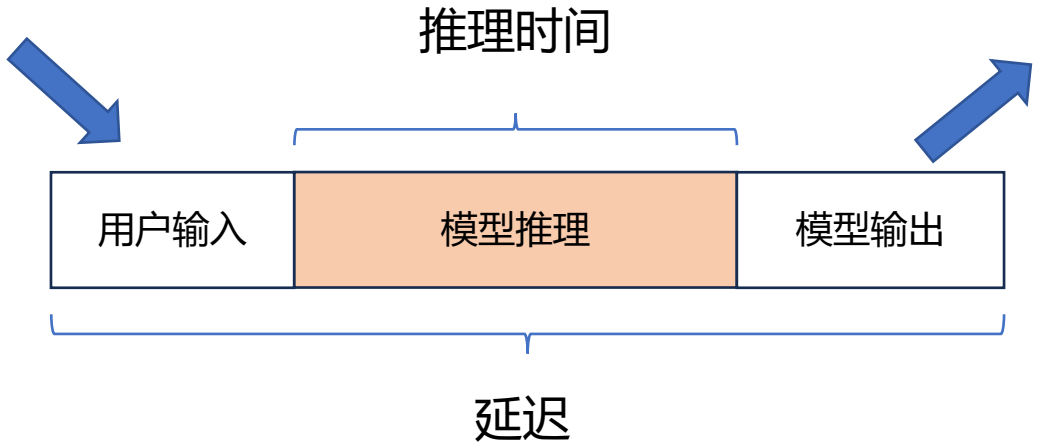
模型每次推理所需的时间，通常以毫秒（ms）为单位。高推理速度对于实时应用和用户体验非常重要。

- 延迟（Latency）

模型从接收到输入到输出结果所需的时间。

低延迟对于实时应用（如语音识别、自动驾驶）尤为重要。在LLM推理中，计算公式如下：

$$T = T_{\text{io}} + T_{\text{attention}} + T_{\text{ffn}}$$



轻量化相关理论

◆ 轻量化模型评估指标

- 推理效果（performance）

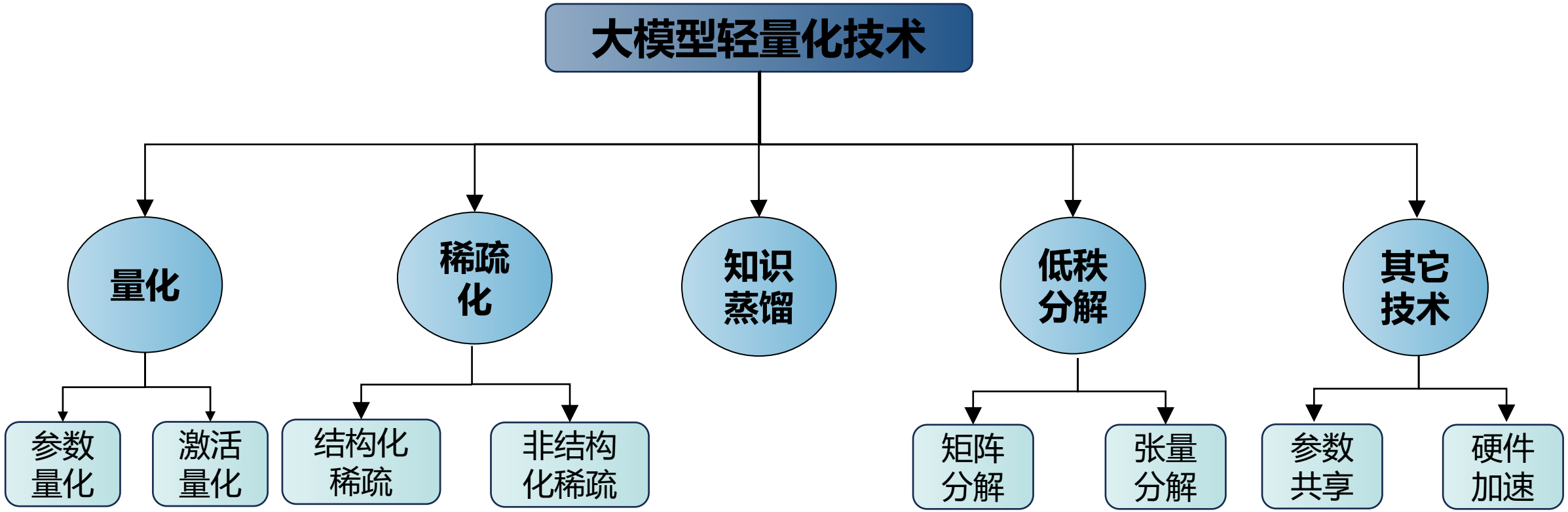
  压缩后模型在各类任务上的表现，如精准度（ACC），困惑度（PPL），BLEU值等。

维持压缩后模型的推理效果是轻量化的重要的目标之一。

指标	适用任务	说明
ACC	分类任务	准确率（Accuracy），衡量模型正确预测的样本占总样本的比例
PPL	生成任务	困惑度（Perplexity），衡量语言模型预测下一个词的不确定性，值越低表示模型预测能力越强。
BLEU	机器翻译、文本生成	双语评估的不确定性（Bilingual Evaluation Understudy），用于评估机器翻译或文本生成任务的质量。
F1	分类任务	F1分数是精确率和召回率的调和平均数，用于衡量不平衡数据集上的分类性能。
EM	信息抽取	精确匹配（Exact Match），用于评估信息抽取任务中模型输出与真实标签的完全一致性。
…	…	……

目录

轻量化技术总览



目标	减少计算量	减少计算量	保留泛化能力	减少模型参数量	高效训练推理加速

减少模型计算复杂度和内存占用，同时尽可能保持性能和泛化能力

量化技术

◆ 量化基本理论

大模型量化是一种将深度学习模型的参数从高精度（16位浮点数，FP16）转换为低精度（如8位整数，INT8）的方法。

➢ 量化过程：

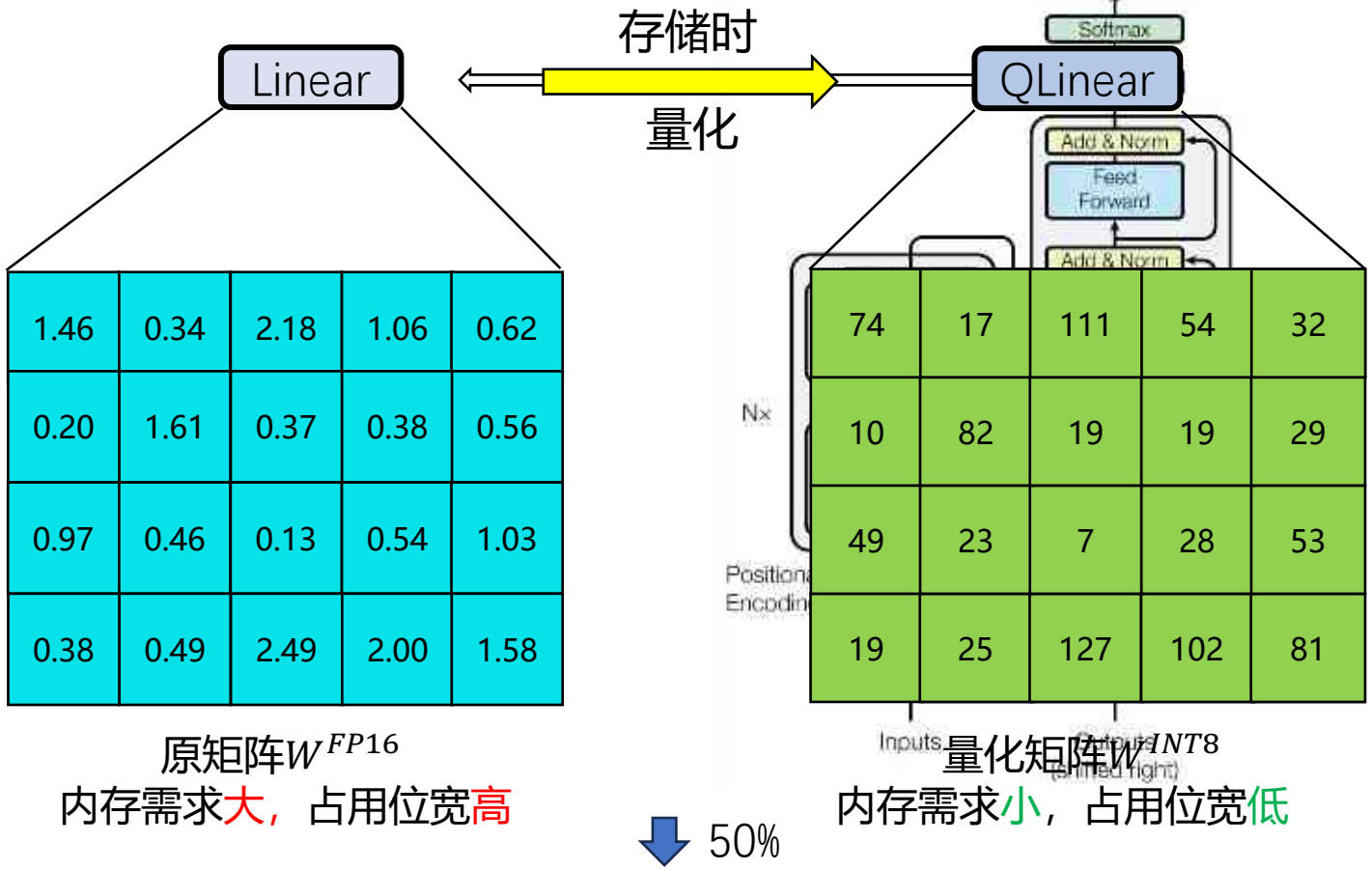
以INT8对称量化为例，在存储权重参数时,将16位浮点数组成的矩阵W^{FP16}经量化存储为8位整数矩阵为W^{INT8}：

$$W^{INT8} = round(\frac{W^{FP16}}{scale})$$

其中，$round()$为近似取整函数，scale为缩放因子：

$$scale = \frac{\max_{i,j}\{|w_{ij}|\}}{127}$$

W^{INT8}内所有值均为$[-127,127]$内的整数。



原矩阵W^{FP16}

内存需求大，占用位宽高

量化矩阵W^{INT8}

内存需求小，占用位宽低

量化技术

◆ 量化基本理论

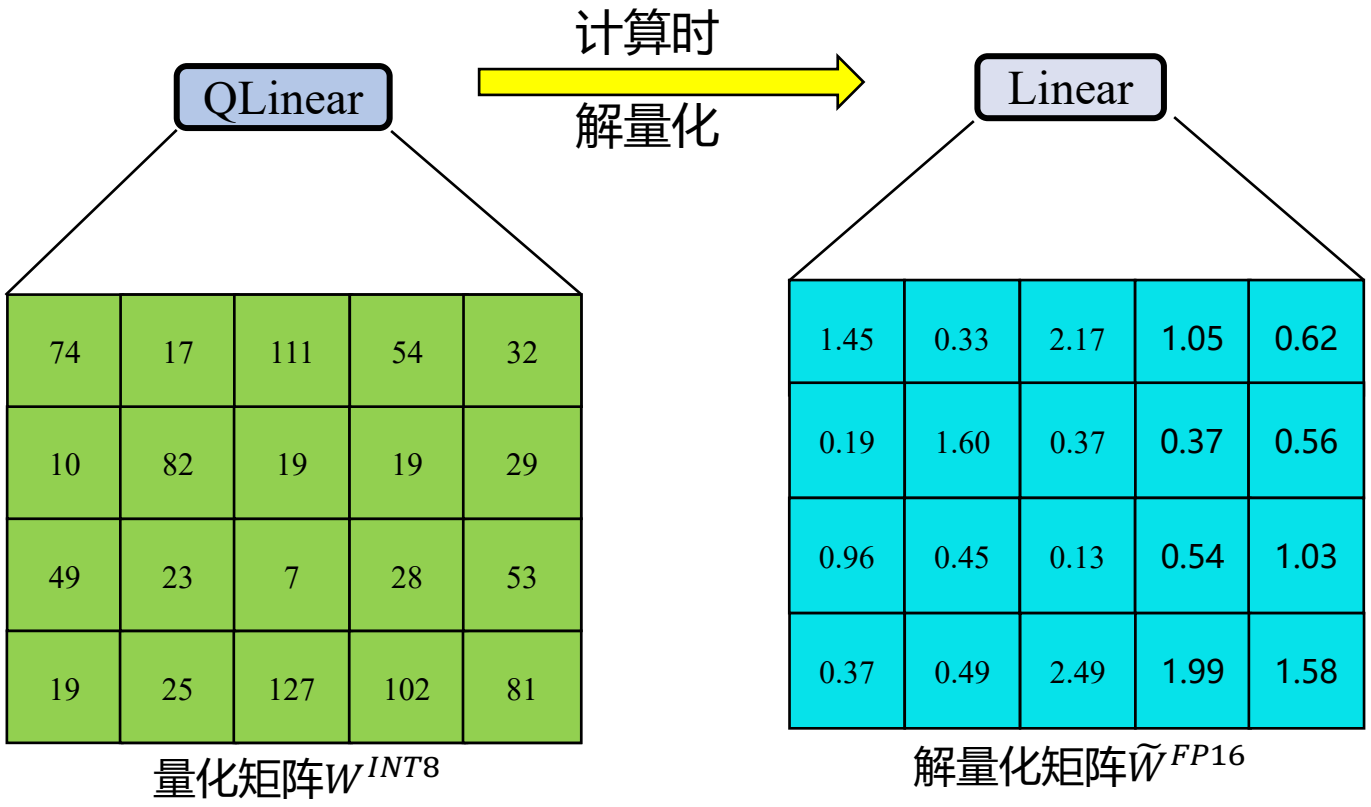
大模型量化是一种将深度学习模型的参数从高精度（16位浮点数，FP16）转换为低精度（如8位整数，INT8）的方法。

➢ 解量化过程

在推理时，为了恢复近似的原始浮点数值，需要进行解量化，即将量化后的整数矩阵W^{INT8}映射回浮点数矩阵$\widetilde{W}^{FP16}$：

$$\widetilde{W}^{FP16} = W^{INT8} \cdot scale$$

解量化后的矩阵$\widetilde{W}^{FP16}$相对于原矩阵W^{FP16}有一定的误差，使用的比特数（bits）越多，误差越小。



量化技术

◆低比特量化的难点

异常值

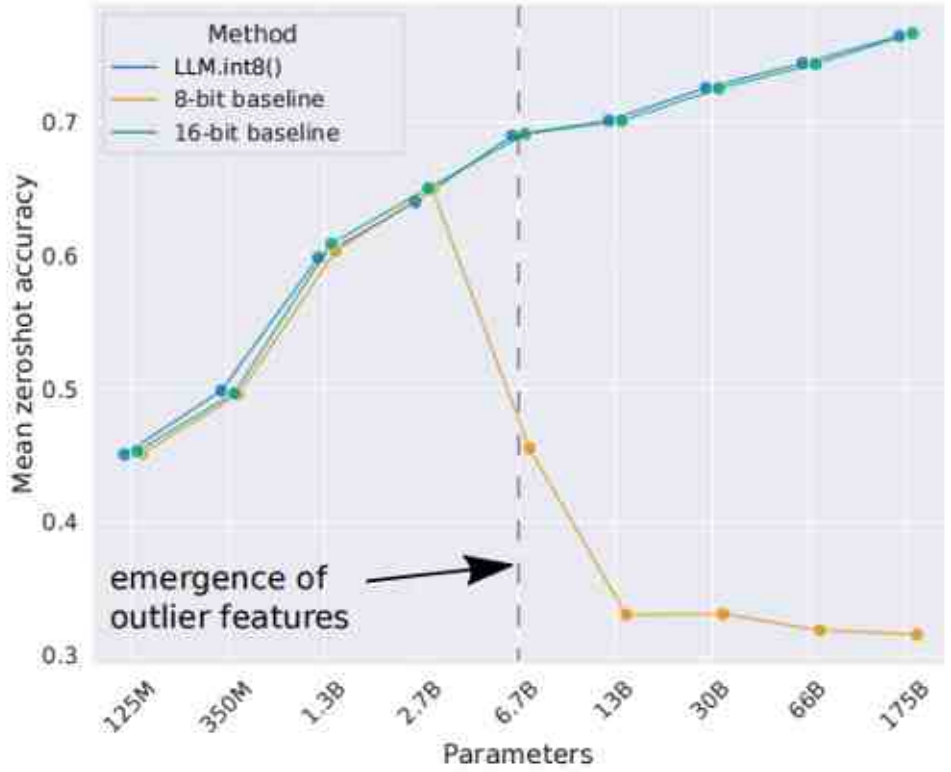
0.09	43.4	-0.1	65.4
0.15	58.7	0.0	57.7
-0.2	-68.3	1.1	42.1
0.1	54.8	-0.25	87.5
0.9	61.9	-0.8	47.9

原矩阵X^{FP16}

量化+
解量化

0.0	43.4	0.0	65.4
0.0	58.7	0.0	57.7
0.0	-68.3	1.0	41.1
0.0	55.1	0.0	87.5
0.9	61.9	-0.9	47.7

解量化矩阵$\tilde{X}^{FP16}$



当大模型参数量大于6.7B时，经激活层生成的矩阵X存在占总参数量0.1%的异常值（outlier），这些异常值导致量化时矩阵一部分正常值被量化为零（如中间示例图标红部分），严重影响量化大模型的性能。

Dettmers T, Lewis M, Belkada Y, et al. LLM. int8 (): 8-bit Matrix Multiplication for Transformers at Scale[C]. NeurIPS 2022

量化技术

◆ LLM.int8()解决方案

LLM.int8() 采用混合精度量化的方法，保持 X矩阵内的异常值为16位浮点数（FP16）且不参与量化，X其余的参数正常量化，取得了很好的效果。

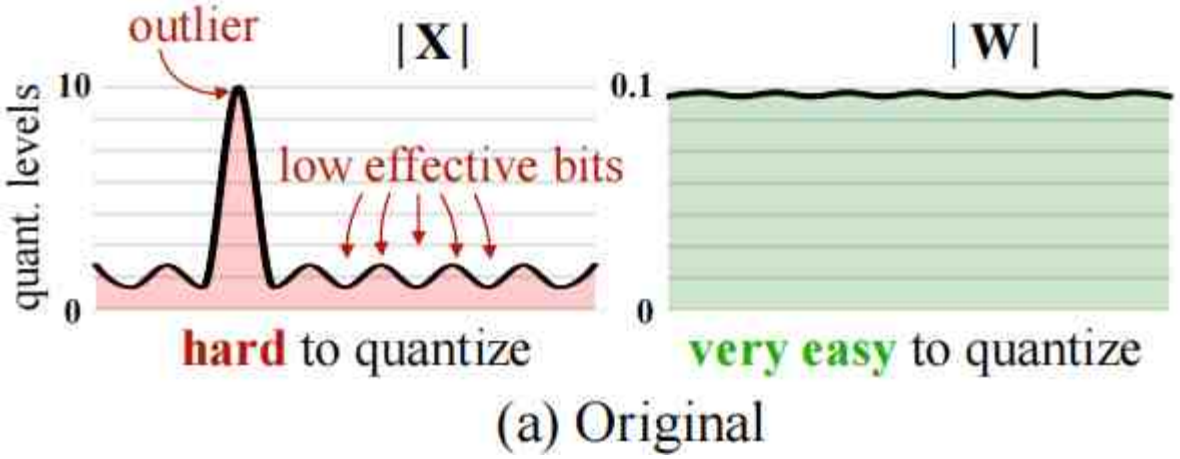
Parameters	125M	1.3B	2.7B	6.7B	13B
浮点类型→ 32-bit Float	25.65	15.91	14.43	13.30	12.45
Int8 absmax	87.76	16.55	15.11	14.59	19.08
Int8 zeropoint	56.66	16.24	14.76	13.49	13.94
Int8 absmax row-wise	30.93	17.08	15.24	14.13	16.49
Int8 absmax vector-wise	35.84	16.82	14.98	14.13	16.48
Int8 zeropoint vector-wise	25.72	15.94	14.36	13.38	13.47
Int8 absmax row-wise + decomposition	30.76	16.19	14.65	13.25	12.46
混合精度→ Absmax LLM.int8() (vector-wise + decomp)	25.83	15.93	14.44	**13.24**	**12.45**
Zeropoint LLM.int8() (vector-wise + decomp)	**25.69**	**15.92**	**14.43**	**13.24**	**12.45**

问题：混合精度在实际应用中难以部署

Dettmers T, Lewis M, Belkada Y, et al. LLM. int8 (): 8-bit Matrix Multiplication for Transformers at Scale[C]. NeurIPS 2022

量化技术

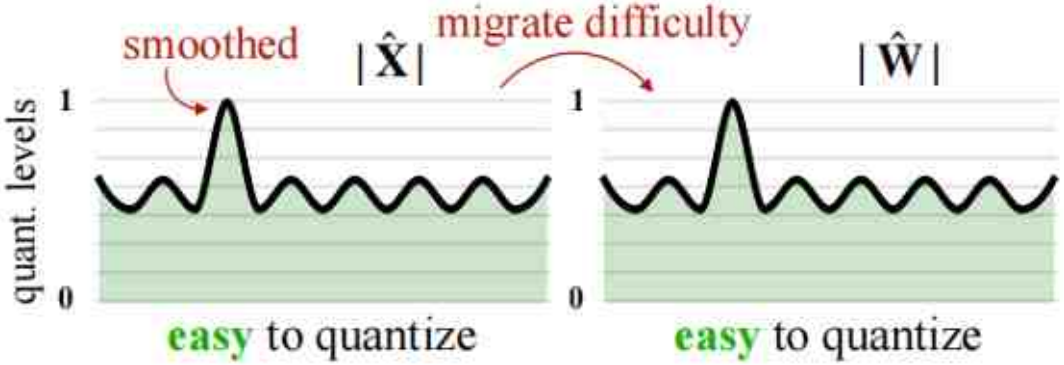
◆ SmoothQuant: 缓解异常值的影响



(a) Original

普通量化方法为直接对W和X分别做量化，由于异常值的存在，X难以被高精度量化：

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{W}$$



(b) SmoothQuant

SmoothQuant方法利用W矩阵的参数分布均匀，无异常值的特点，从数学等价的角度出发，令W 矩阵“代偿”一部分异常值的影响，实现了W和X的高精度INT8量化：

$$\mathbf{Y} = (\mathbf{X}\mathrm{diag}(\mathbf{s})^{-1}) \cdot (\mathrm{diag}(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha}$$

Xiao G, Lin J, Seznec M, et al. Smoothquant: Accurate and efficient post-training quantization for large language models[C]. ICML 2023

量化技术

- 业界常用的量化工具

通用

- 训练后量化 • SmoothQuant • AWQ • OmniQuant • Squeeze LLM …
- 量化感知训练 • LLM-QAT • QLoRA • TensorRT-LLM …

端侧

• TinyChat • GPTQ • llama.cpp …

Lin J, Tang J, Tang H, et al. AWQ: Activation-ware weight quantization for on-device llm compression and acceleration[C]. MLSys 2024

Frantar E, Ashkboos S, Hoefler T, et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers[C]. ICLR 2023

Shao W, Chen M, Zhang Z, et al. Omniquant: Omnidirectionally calibrated quantization for large language models[C]. ICLR 2024

Kim S, Hooper C, Gholami A, et al. SqueezerLLM: Dense-and-Sparse Quantization[C]. ICML 2024

稀疏化技术

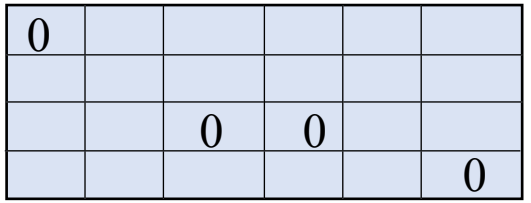
◆ 参数稀疏化

背景

随着模型参数量的增大，训练一个巨大的生成式模型，需要很大的GPU内存，并且产生巨大的计算量。大模型稀疏化通过减少参数的密集度来**加快计算速度**和**减少存储成本**。

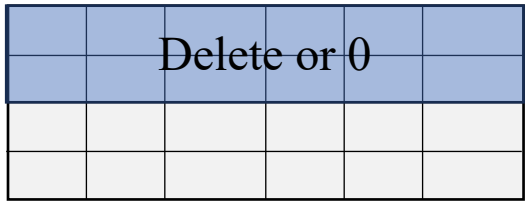
稀疏化的基本思想

(1) **非结构化稀疏**
寻找一种方法来确定模型中**哪些参数对模型的输出贡献较小或不重要**，然后将这些参数设置为零或进行其他　形式的删减。这样可以在保持模型性能的前提下，大幅减少模型的参数数量。



直接移除权重矩阵中最不重要的权重值，使得它们变为零

(2) **结构化稀疏**
基于结构式的稀疏策略对参数进行剪枝或置零，以**充分利用参数的稀疏性来加速计算过程**。例如，在矩阵乘法等运算中，跳过零值参数的计算，从而提高计算效率。

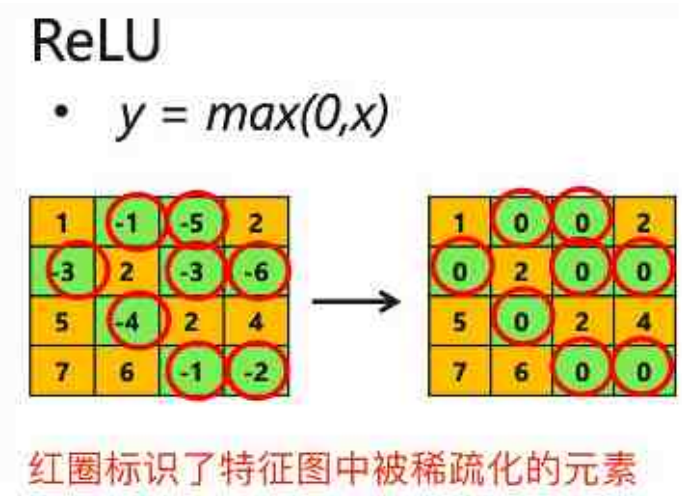


移除整个行、列、卷积核或者神经元等结构单元

稀疏化技术

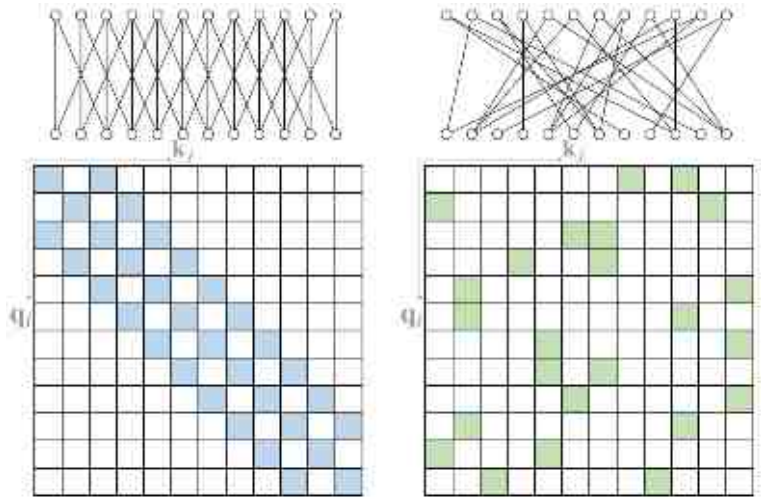
◆示例

➢ 稀疏激活：



可以在激活函数层面上引入稀疏性，例如使用ReLU激活函数自然产生的零值。

➢ 稀疏注意力机制：



稀疏注意力机制通过限制注意力计算的范围，减少了不必要的计算。例如，使用局部注意力或分块稀疏注意力来降低计算量。

稀疏化技术

◆非结构化稀疏

为什么使用非结构化稀疏?

结构化稀疏由于限制了剪枝元素的选择自由，会导致模型准确率的大幅下降，**而采用非结构化稀疏则可以较好的保留模型的准确率。**

非结构化稀疏产生的问题

由于GPU中的张量核专门设计用于**加速稠密矩阵乘法**计算的专用单元，对非结构化稀疏矩阵乘法计算的效率较低，**因此会造成模型推理速度的大幅下降。**

问题：如何在使用非结构化稀疏的同时，保持较高的模型推理速度呢?

Xia H, Zheng Z, Li Y, et al. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. VLDB 2023.

稀疏化技术

◆非结构化稀疏

针对非结构化稀疏矩阵乘法较慢的问题，Flash-LLM提出将稀疏矩阵转化为密集矩阵，**每次进行计算前都将稀疏矩阵转化为这种稠密格式**

Tiled-CSL密集存储格式

T1	T2	T3
T4	T5	T6
…	…	…

将稀疏矩阵分成多个大小固定的Tiles

10	16	21
28	34	39
…	…	…

用数组TileOffsets存储每个Tile的非零元素的数量

N1	N2	N3
N4	N5	N6
…	…	…

用数组NonZeros依次存储每个Tile的非零元素

NonZeros中的每个N都存储着非零元素值与其位置

Xia H, Zheng Z, Li Y, et al. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. VLDB 2023.

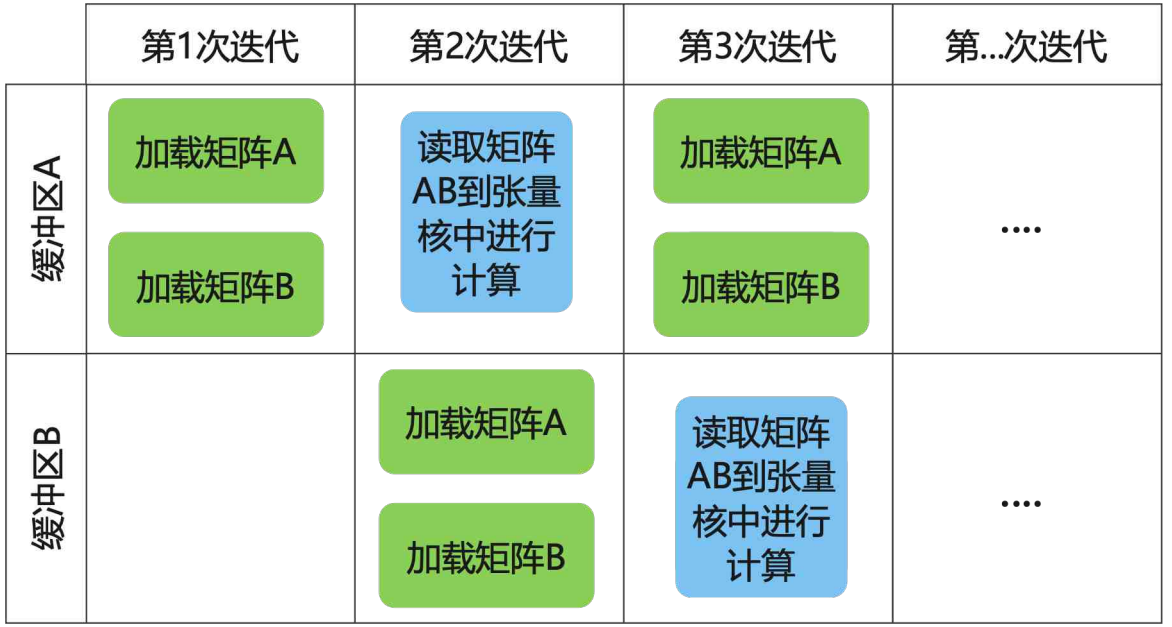
稀疏化技术

◆非结构化稀疏

问题：张量核进行计算前，需要进行矩阵数据加载，而加载时张量核空闲，造成了核使用率低

解决方案：Flash-LLM提出了一种双缓冲计算重叠的计算流水线。

优势：采用这种新的流水线进行计算能够减少GPU的空闲时间。有效提升了模型推理的效率。

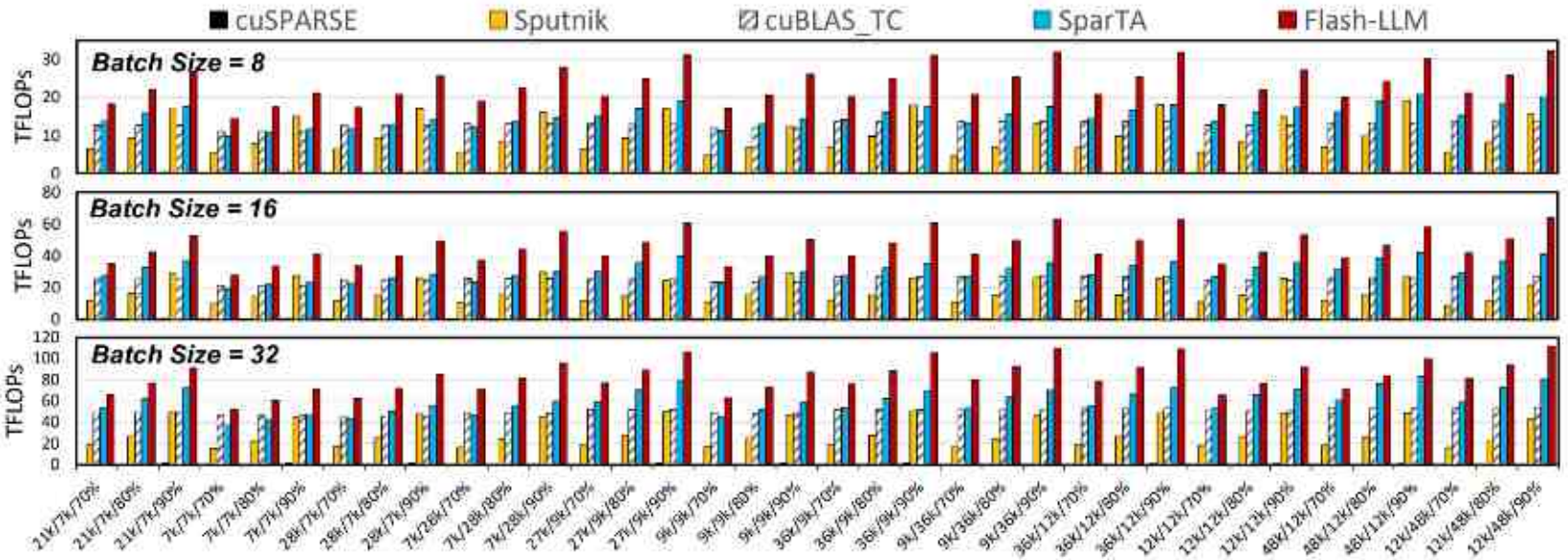


双缓冲计算重叠的计算流水线

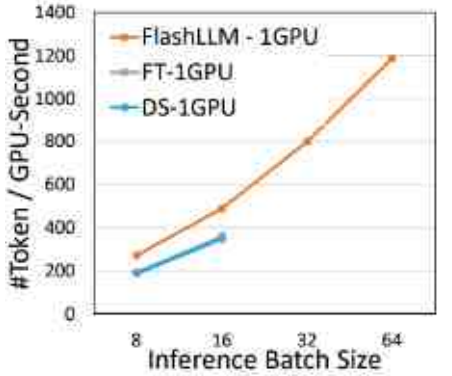
计算重叠：可以看出每次迭代时，都会在一个缓冲区加载数据，另一个缓冲区计算矩阵乘法。

Xia H, Zheng Z, Li Y, et al. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. VLDB 2023.

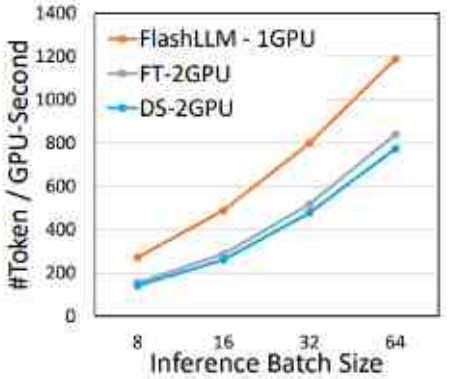
稀疏化技术



Flash-LLM与多个baselines计算性能的对比结果



1个GPU的模型吞吐量对比



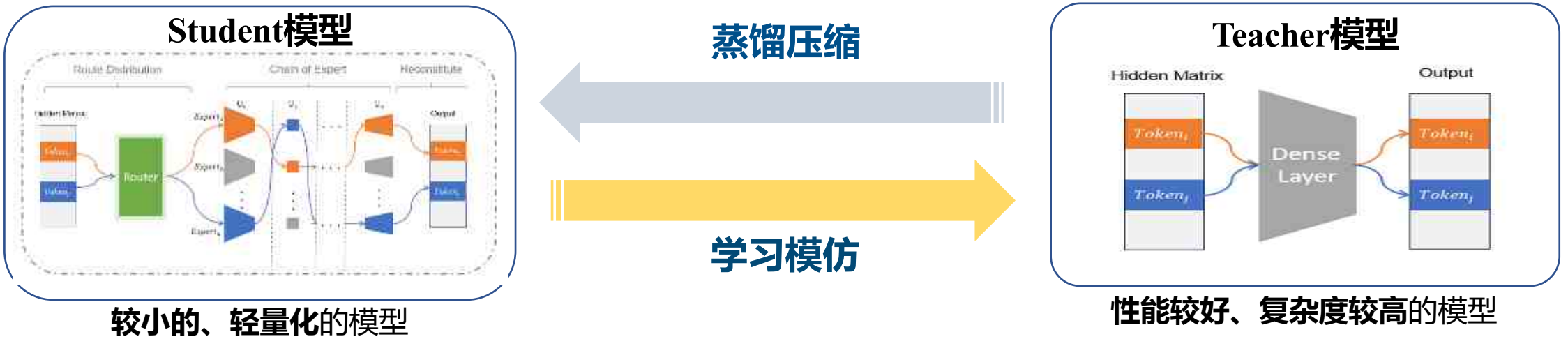
2个GPU的模型吞吐量对比

上述结果表明，Flash-LLM在非结构化稀疏矩阵乘法的性能方面具有显著优势，能够更好地支持大规模生成模型的推理。

Xia H, Zheng Z, Li Y, et al. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. VLDB 2023.

知识蒸馏

◆知识蒸馏基础理论



知识蒸馏（Knowledge Distillation）旨在将知识从大型复杂模型（教师模型）转移到更小更简单的模型（学生模型），使得学生模型能够在性能上接近教师模型，同时具有较少的计算资源需求，从而实现模型压缩。

知识蒸馏的核心公式为蒸馏损失函数： $L = \alpha L_{\mathrm{CE}} + (1-\alpha) L_{\mathrm{KD}}$

其中L_{CE}是学生模型的交叉熵损失， L_{KD}是学生模型与教师模型软标签之间的蒸馏损失。

知识蒸馏

◆大语言模型的知识蒸馏

大语言模型上的知识蒸馏工作可以划分为两类，**黑盒知识蒸馏**和**白盒知识蒸馏**。

黑盒知识蒸馏

黑盒（Black-Box）知识蒸馏中，学生模型只能访问教师模型的输出（**闭源大模型**），而无法直接访问教师模型的内部结构、参数或中间层的激活值。

其中黑盒知识蒸馏又分为“思维链蒸馏”、“上下文学习蒸馏”以及“指令遵循蒸馏”三种方法。

白盒知识蒸馏

白盒（White-Box）知识蒸馏中，学生模型不仅可以访问教师模型的输出，还可以访问**教师模型**的内部结构、参数和中间层的激活值（**开源大模型**）。

学生模型可以直接学习教师模型的中间层特征或特定参数，从而获得更丰富的知识。

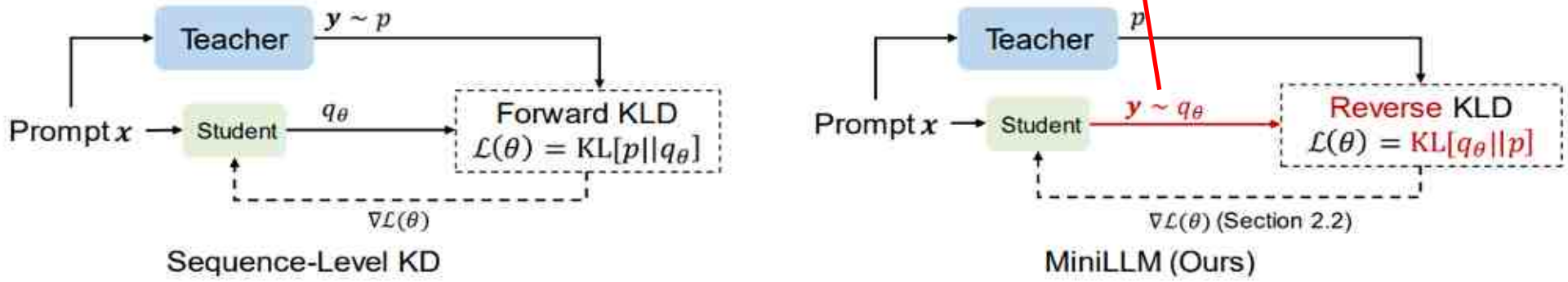
Gu Y, Dong L, Wei F, et al. MiniLLM: Knowledge distillation of large language models[C]. ICLR 2024.

知识蒸馏

◆ 大语言模型的知识蒸馏

标准知识蒸馏中的**前向KL散度**（Forward KLD）会迫使学生模型试图覆盖教师模型的**所有细节信息**，即使那些细节对任务并不重要，这在大模型知识蒸馏中是不实际的，因为小模型的能力有限，这会**导致资源浪费**的同时，使得学生模型**在真正重要的部分表现不佳。**

反向KL散度（Reverse KLD）选择从学生模型中采样学习样本 y ，允许学生模型可以结合自身学习能力的同时，**从教师模型中学习对于学生模型最重要的知识**，以**避免资源的浪费**，从而**在关键任务上表现更好。**



Gu Y, Dong L, Wei F, et al. MiniLLM: Knowledge distillation of large language models[C]. ICLR 2024.

知识蒸馏

天津大学 Tianjin University

◆ 大语言模型的知识蒸馏

在前面损失函数基础上，采用多种优化算法进一步改进学习：

单步分解：这是将每步的生成质量从损失的梯度中单独提出来，以减少训练时的方差并加速收敛，提升单步生成质量。

教师指导的采样：在采样 $\boldsymbol{y}$ 时混合教师和学生模型的分布。

长度正则化：当前的损失容易导致蒸馏后的模型产生较短的序列，因此增加了一个正则化到损失函数中，以避免KL散度的累积值过小。

$$\nabla \mathcal{L}(\theta) = -\underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}}_{\boldsymbol{y}\sim\widetilde{p}(\cdot|\boldsymbol{x})}\left[\sum_{t=1}^{T} w_t \left[\underbrace{\nabla \sum_{y'\in V} q_\theta(y'|\boldsymbol{y}_{<t},\boldsymbol{x}) \log \frac{p(y'|\boldsymbol{y}_{<t},\boldsymbol{x})}{q_\theta(y'|\boldsymbol{y}_{<t},\boldsymbol{x})}}_{(\nabla\mathcal{L})_{\text{Single}}\text{part}} + \underbrace{R_{t+1}^{\text{Norm}} \frac{\nabla q_\theta(y_t|\boldsymbol{y}_{<t},\boldsymbol{x})}{q_\theta(y_t|\boldsymbol{y}_{<t},\boldsymbol{x})}}_{(\nabla\mathcal{L})_{\text{Long}}^{\text{Noom}}\text{part}}\right]\right]$$

Gu Y, Dong L, Wei F, et al. MiniLLM: Knowledge distillation of large language models[C]. ICLR 2024.

知识蒸馏

MiniLLM方法在不同体量大模型上的实验结果

与其他知识蒸馏方法相比，MiniLLM方法学到了三种教师大模型**更多的知识**，**性能表现更优**。

MiniLLM方法在各种小规模的学生模型上达到超越原教师模型的性能。

*表示学生模型性能超越了教师模型。

可以用一半的参数达到原本的性能，实验效果优于其他蒸馏方法。

Model	#Params	Method	DollyEval GPT4	DollyEval R-L	SelfInst GPT4	SelfInst R-L	VicunaEval GPT4	VicunaEval R-L	S-NI R-L	UnNI R-L
GPT-2	1.5B	Teacher	58.4	27.6	42.9	14.3	48.6	16.3	27.6	31.8
	120M	SFT w/o KD	38.6	23.3	26.3	10.0	32.8	14.7	16.3	18.5
		KD	40.3	22.8	27.8	10.8	31.9	13.4	19.7	22.0
		SeqKD	41.2	22.7	26.2	10.1	31.0	14.3	16.4	18.8
		MiniLLM	**44.7**	**24.6**	**29.2**	**13.2**	**34.1**	**16.9***	**25.3**	**26.6**
	340M	SFT w/o KD	51.9	**25.5**	39.6	13.0	42.3	16.0	25.1	32.0
		KD	51.6	25.0	39.2	12.0	42.8	15.4	23.7	31.0
		SeqKD	50.5	25.3	39.0	12.6	**43.0**	16.9*	22.9	30.2
		MiniLLM	**52.2**	25.4	**40.5**	**15.6**	42.6	**17.7***	**27.4**	**34.5**
	760M	SFT w/o KD	50.7	25.4	38.3	12.4	43.1	16.1	21.5	27.1
		KD	53.4	25.9	40.4	13.4	43.4	16.9*	25.3	31.7
		SeqKD	52.0	25.6	38.9	14.0	42.4	15.9	26.1	32.9
		MiniLLM	**54.7**	**26.4**	**44.6***	**15.9**	**45.7**	**18.3***	**29.3***	**37.7***
OPT	13B	Teacher	70.3	29.2	56.1	18.4	58.0	17.8	30.4	36.1
	1.3B	SFT w/o KD	52.6	26.0	37.7	11.4	40.5	15.6	23.1	28.4
		KD	52.7	25.4	36.0	12.2	40.8	14.9	21.9	27.0
		SeqKD	51.0	26.1	36.6	12.7	42.6	16.6	21.4	28.2
		MiniLLM	**60.7**	**26.7**	**47.0**	**14.8**	**50.6**	**17.9***	**28.6**	**33.4**
	2.7B	SFT w/o KD	55.4	27.1	38.9	13.9	44.8	16.6	24.9	32.3
		KD	60.5	25.9	48.6	13.8	51.3	16.7	26.3	30.2
		SeqKD	57.6	27.5	40.5	13.3	44.5	16.5	25.3	32.3
		MiniLLM	**63.2**	**27.4**	**52.7**	**17.2**	**55.9**	**19.1***	**30.7***	**35.1**
	6.7B	SFT w/o KD	67.9	27.6	56.4	16.4	57.3	17.8	30.3	28.6
		KD	68.6	28.3	58.0	17.0	57.0	17.5	30.7*	26.7
		SeqKD	69.6	28.5	54.0	17.0	57.6	17.9*	30.4	28.2
		MiniLLM	**70.8***	**29.0**	**58.5***	**17.5**	**60.1***	**18.7***	**32.5***	**36.7***
LLaMA	13B	Teacher	79.0	29.7	75.5	23.4	65.1	19.4	35.8	38.5
	7B	SFT w/o KD	73.0	26.3	69.2	20.8	61.6	17.5	32.4	35.8
		KD	73.7	27.4	70.5	20.2	62.7	18.4	33.7	37.9
		SeqKD	73.6	27.5	71.5	20.8	62.6	18.1	33.7	37.6
		MiniLLM	**76.4**	**29.0**	**73.1**	**23.2**	**64.1**	**20.7***	**35.5**	**40.2***

Gu Y, Dong L, Wei F, et al. MiniLLM: Knowledge distillation of large language models[C]. ICLR 2024.

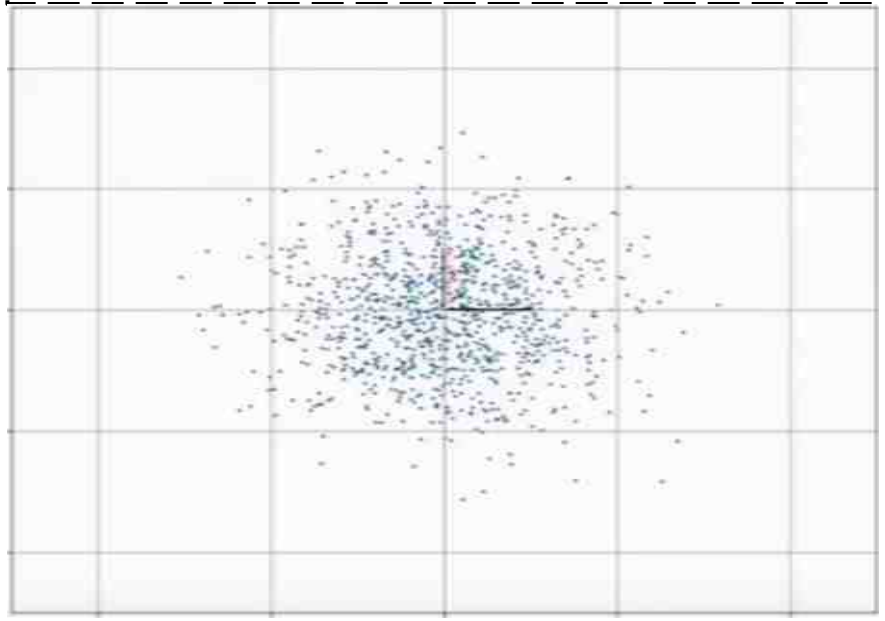
低秩分解

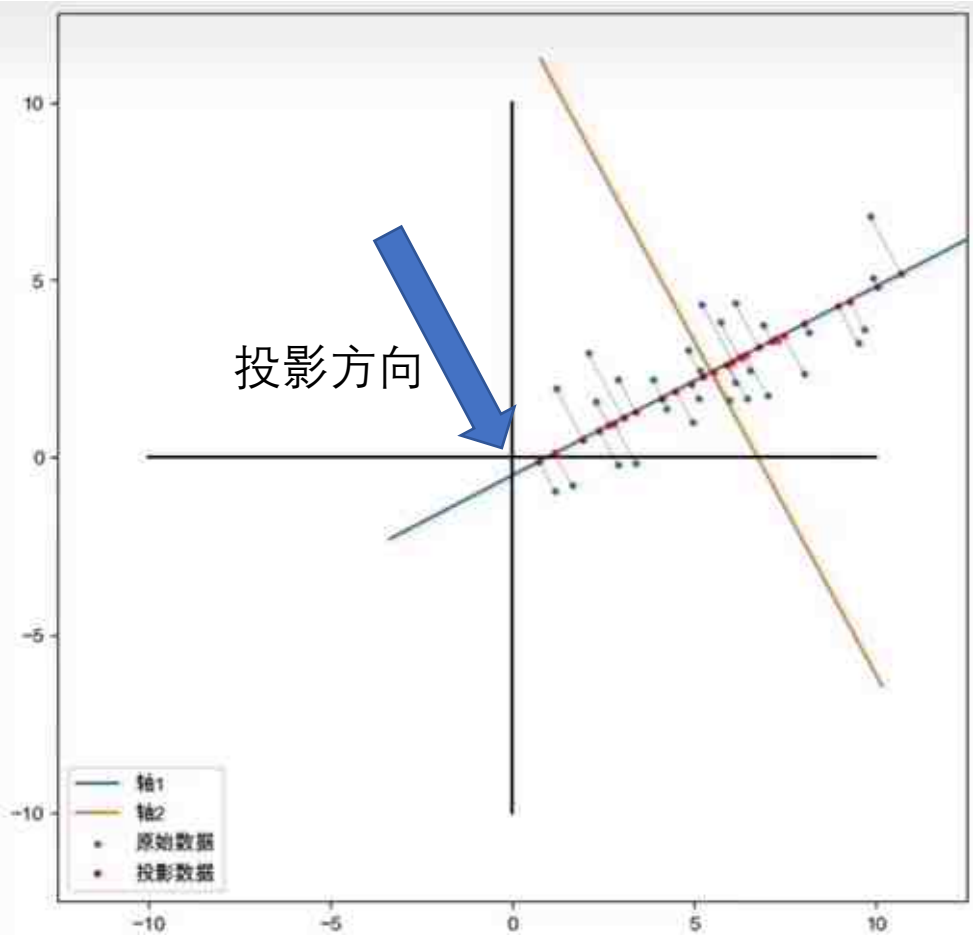
◆ 基本理论

经典分解理论：PCA分解

将数据向方差最大的方向投影
从而得到最具代表性的特征

原始数据可能有极多的维度，难以储存与使用
我们希望实现数据压缩，只保留原数据最主要的信息，去除冗余信息





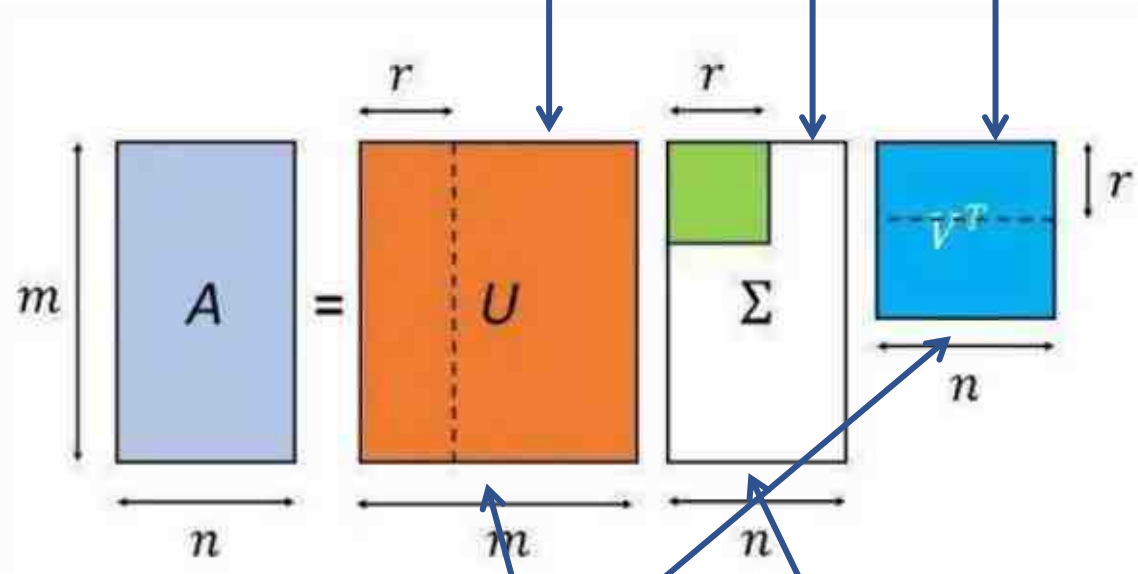
通过这种方式可以实现数据的压缩

例如左图中，将二维数据降为一维

低秩分解

◆ 基本理论

经典分解理论：SVD分解



将原始矩阵分解为**左右奇异矩阵**与**特征值矩阵**
左右**奇异矩阵**的行列代表原矩阵中的**成分**
对应的**特征值大小**则代表相应成分的**信息量**

删除奇异矩阵中不重要的成分
实现数据的压缩

例如左图中，仅维度为r的部分被保留

其中 **r** 被称为分解矩阵的**秩**，它代表了原矩阵中被保留的成分多少。

通过将模型参数转为低秩形式，我们可以有效压缩模型参数，例如，在Llama3-8B中，保留**50%**的矩阵秩，即可压缩超过**20亿**的模型参数，大大减少部署成本。

低秩分解

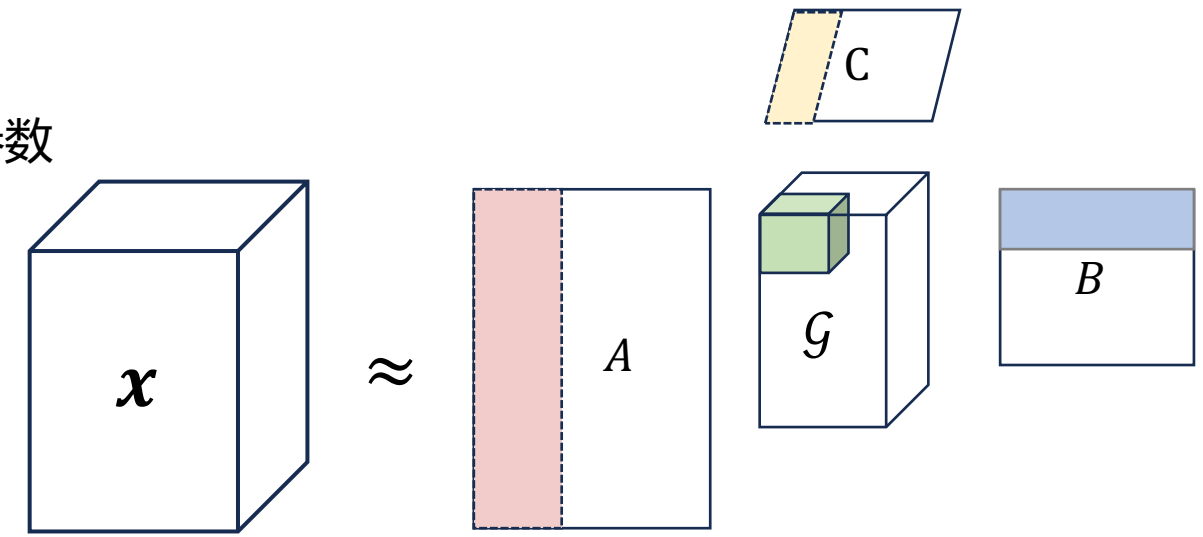
◆ 基本理论

适应大模型：张量分解技术

——分解结构更为复杂的大模型参数

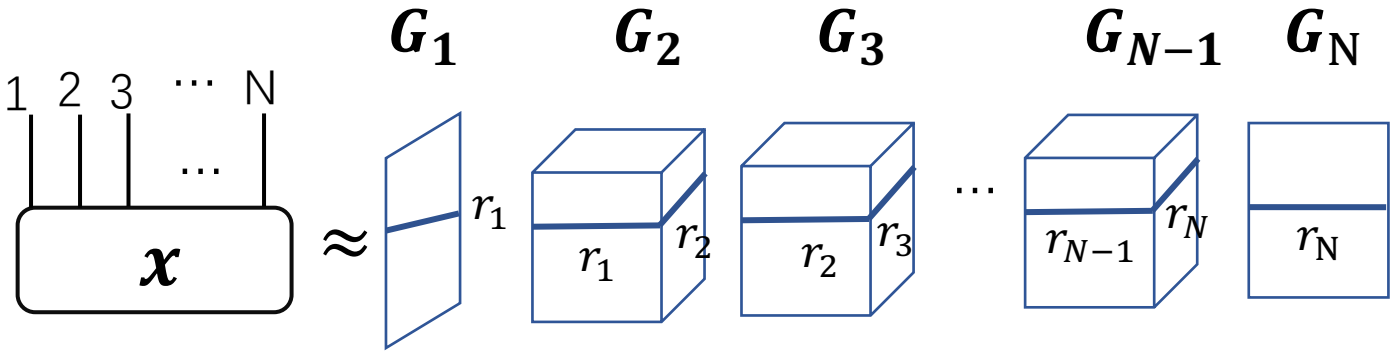
Tucker分解

可以被视作一种高阶PCA. 将张量分解为核心张量在每个mode上与矩阵的乘积



Tensor Train分解

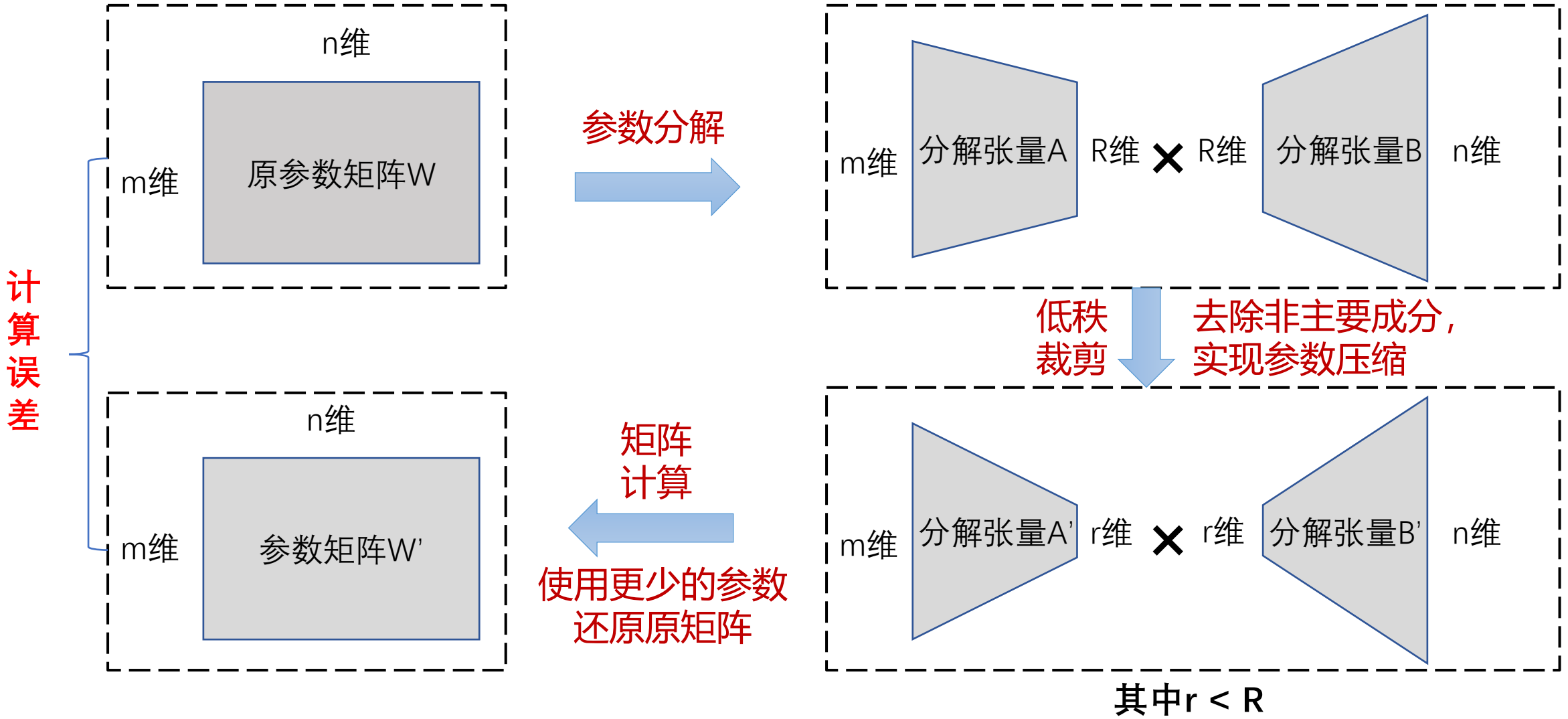
将一个N阶张量分解成了2个二阶张量和N-2个三阶张量的乘积,



低秩分解

◆基本理论

矩阵分解技术

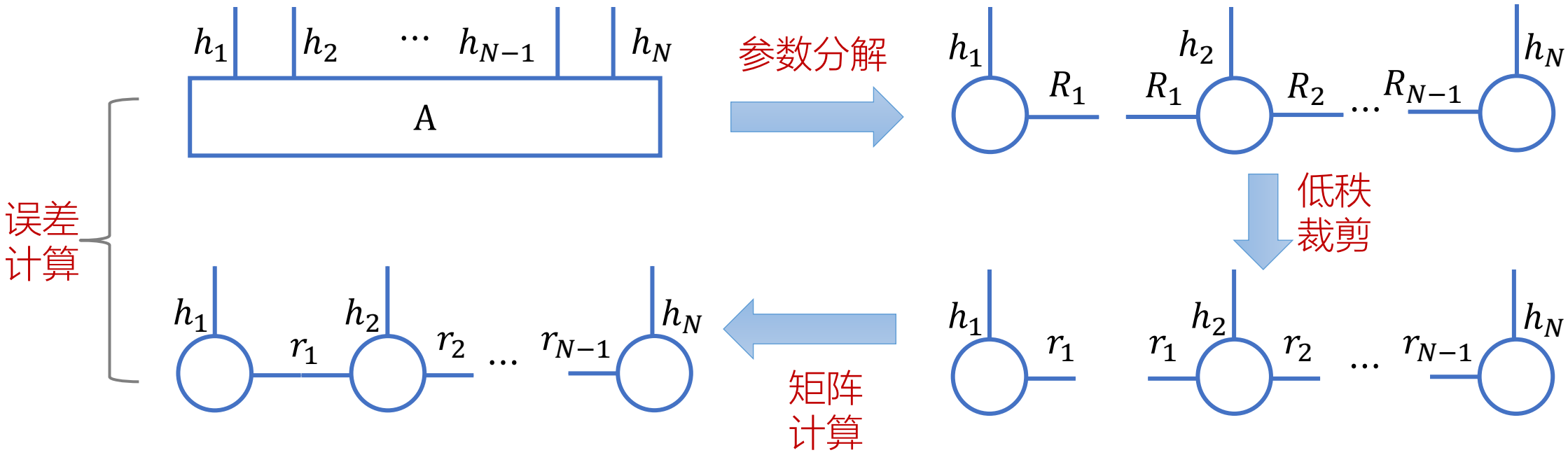


其中r < R

低秩分解

◆基本理论

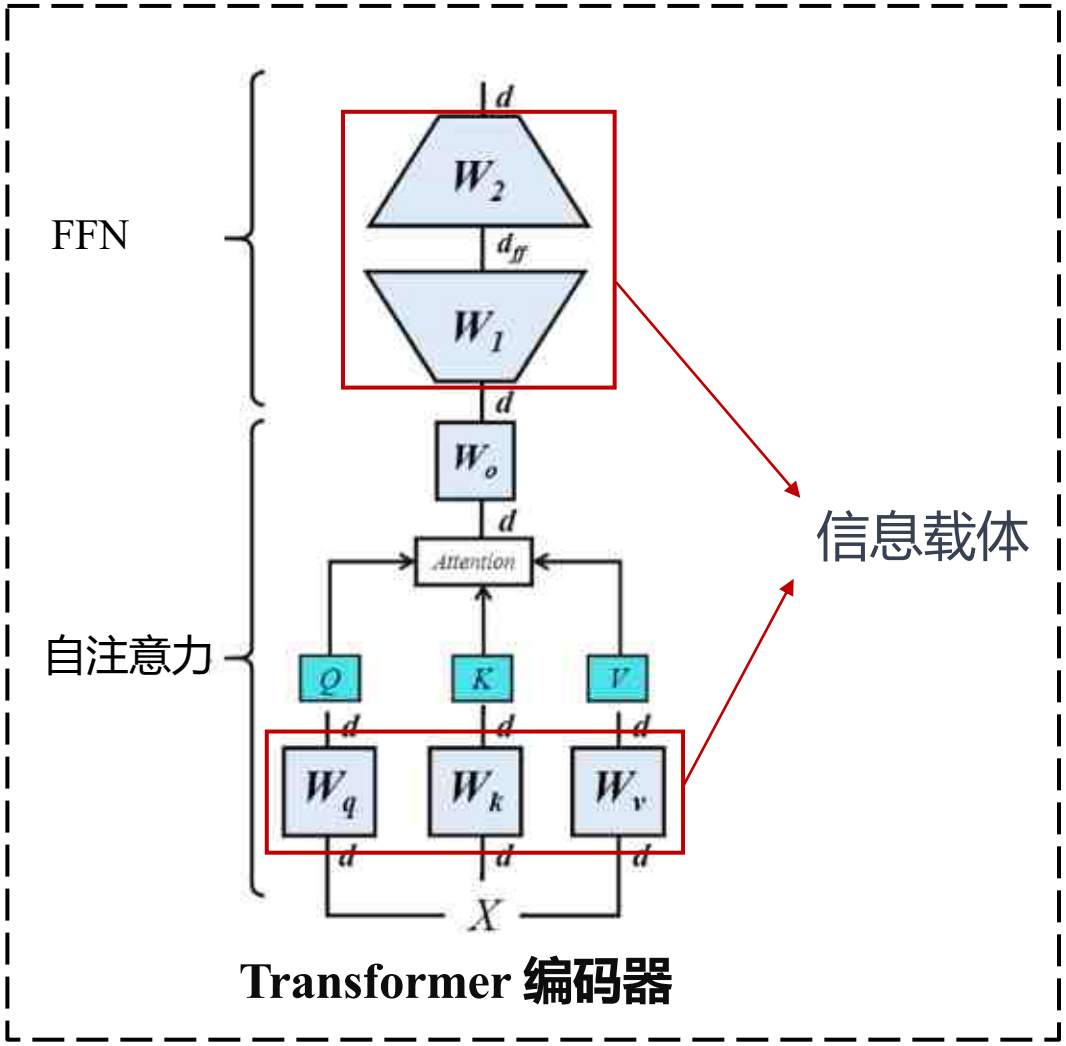
张量网络

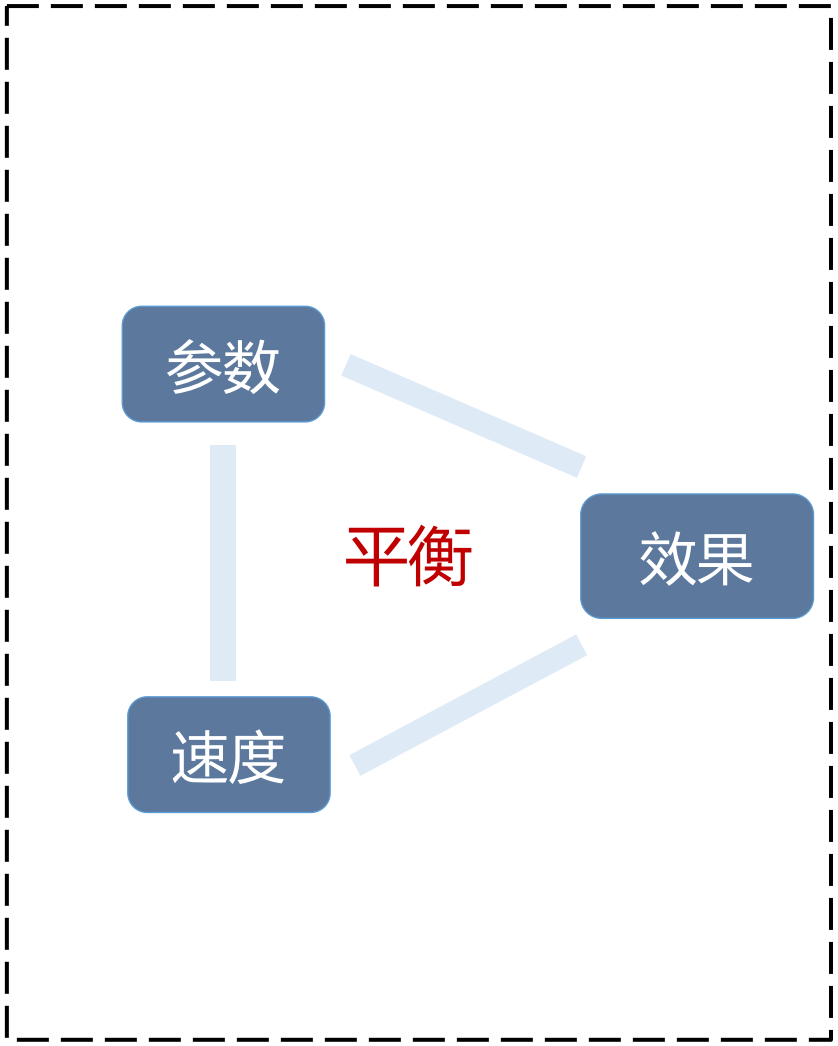


在大模型中，张量分解转为以可训练参数的形式存在，形成**张量网络**，让端到端训练成为可能

混合张量分解技术

单独的低秩分解技术存在模型参数规模、计算速度、以及预测效果平衡问题

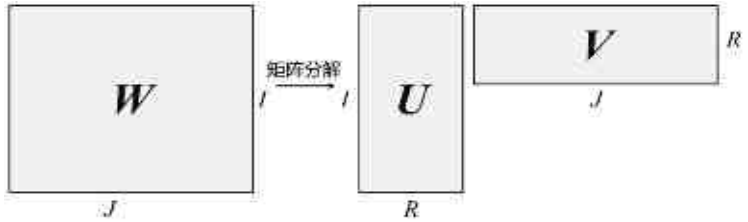




Li S, Zhang P, Gan G, et al. Hypoformer: Hybrid decomposition transformer for edge-friendly neural machine translation[C]. EMNLP 2022.

混合张量分解技术

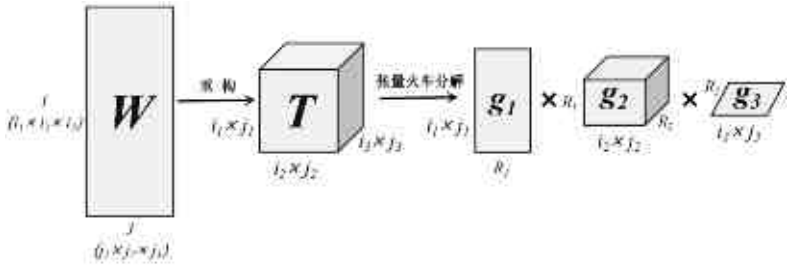
通过将矩阵分解和张量火车分解（Tensor Train, TT）结合，平衡Transformer模型推理速度，预测效果和参数规模的平衡问题



（1）矩阵分解

优势：简单实现，计算速度快。

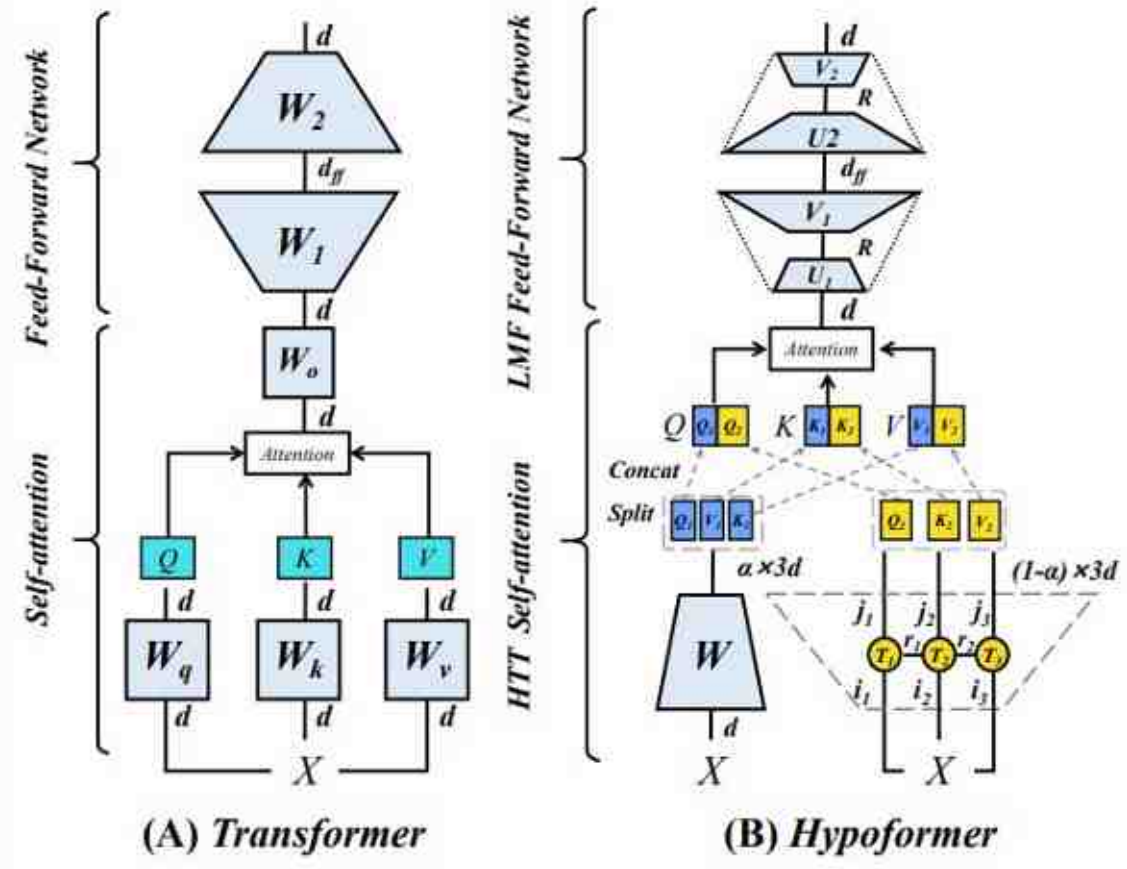
缺点：高秩情况下，参数压缩效果有限。低秩场景中，效果无法保证。



（2）TT分解

优势：具有强大的参数压缩能力。

缺点：在高秩情况下，复杂度较高，影响速度。在低秩场景中，速度快但难以适应，影响效果。

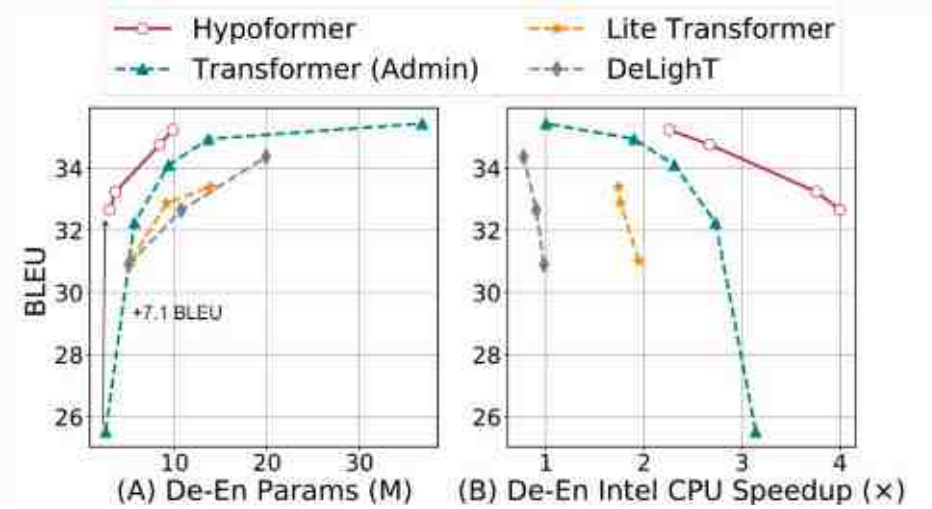


利用小型稠密矩阵来补充低秩张量列车分解，提高效果而不显著增加参数和计算复杂性。

Li S, Zhang P, Gan G, et al. Hypoformer: Hybrid decomposition transformer for edge-friendly neural machine translation[C]. EMNLP 2022.

混合张量分解技术

Hypoformer方法在Transformer模型上推理速度、预测效果以及参数规模的实验结果分析



在不同的压缩倍数下，它在准确率和推理速度上都具有明显的优势。

$Ratio$是参数压缩比，$S_{(Pi)}$为树莓派设备上速度提升倍数，$S_{(\mathrm{Intel})}$为CPU设备上速度提升倍数。

Model	IWSLT'14 De-En					WMT'14 En-De				
	Params	**Ratio**	$S_{(Pi)}$	$S_{(Intel)}$	**BLEU**	**Params**	**Ratio**	$S_{(Pi)}$	$S_{(Intel)}$	**BLEU**
Transformer	36.8 M	1.0×	1.0×	1.0×	34.5	61.0 M	1.0×	1.0×	1.0×	27.3
Lite Transformer	13.9 M	2.7×	1.5×	1.5×	33.6	33.6 M	1.8×	0.8×	1.1×	26.5
HAT Transformer	35.2 M	1.1×	1.8×	1.7×	34.5	46.2 M	1.3×	1.5×	1.7×	26.9
DeLighT	19.9 M	1.9×	1.0×	0.8×	34.4	23.3 M	2.6×	1.3×	1.2×	26.7
Subformer	-	-	-	-	-	38.0 M	1.6×	-	-	27.7
Hypoformer 12-2	**8.4 M**	**4.4×**	3.5×	2.7×	**34.8**	**21.2 M**	**2.9×**	1.5×	1.6×	**27.5**
Hypoformer 12-1	8.6 M	4.3×	**3.9×**	**3.3×**	34.4	23.6 M	2.6×	**1.8×**	**2.8×**	27.4

参数压缩，速度提升，保持性能。

Li S, Zhang P, Gan G, et al. Hypoformer: Hybrid decomposition transformer for edge-friendly neural machine translation[C]. EMNLP 2022.

语素增强的低秩近似技术

通过**语素词嵌入**的**低秩近似**解决原始词向量矩阵的**参数量巨大**问题

Transformer语言模型词向量参数量分析

单词量（大）：$|V|$ 可达到数万甚至数十万

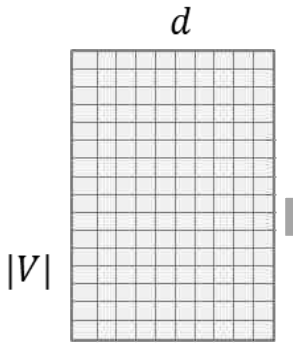
维度（大）：d 通常为512、768、1024

词嵌入矩阵（大）：$|V|\times d$

通过**形态素**分割和**张量积**实现的单词嵌入压缩

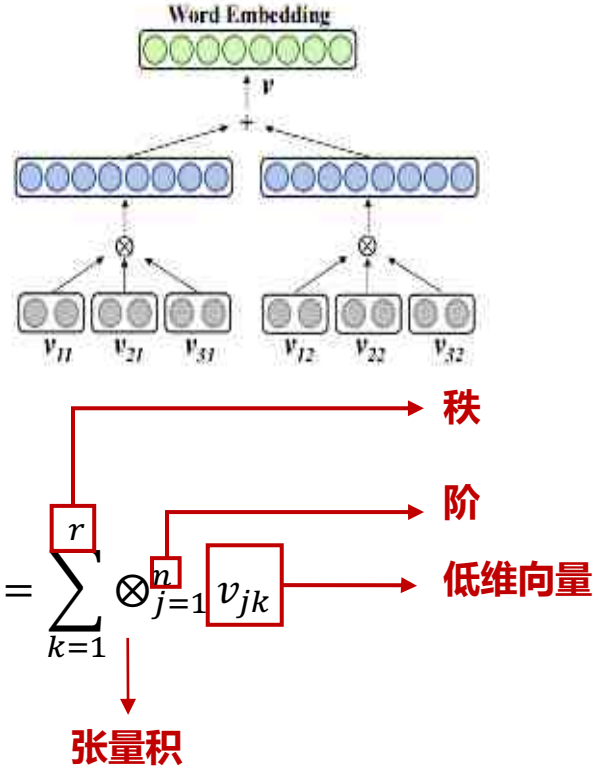
单词维度 >> 形态素维度

单词数量 >> 形态素数量



占Transformer模型参数总量的
20% ~ 80%

传统基于张量积进行embedding压缩



$$v = \sum_{k=1}^{r} \bigotimes_{j=1}^{n} v_{jk}$$

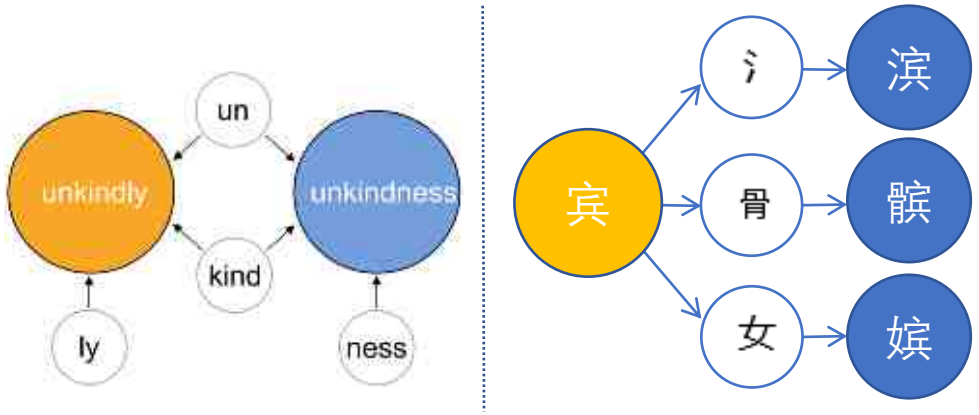
Gan G., **Zhang P.,** Li S., et al. MorphTE: Injecting Morphology in Tensorized Embeddings[J]. NeurIPS 2022.

Kim Y., Kim K. M, S K. Adaptive compression of word embeddings[J]. ACL 2020.

语素增强的低秩近似技术

MorphTE方法在词嵌入矩阵模块上的计算与实验分析

词语的形态组成与语言现象



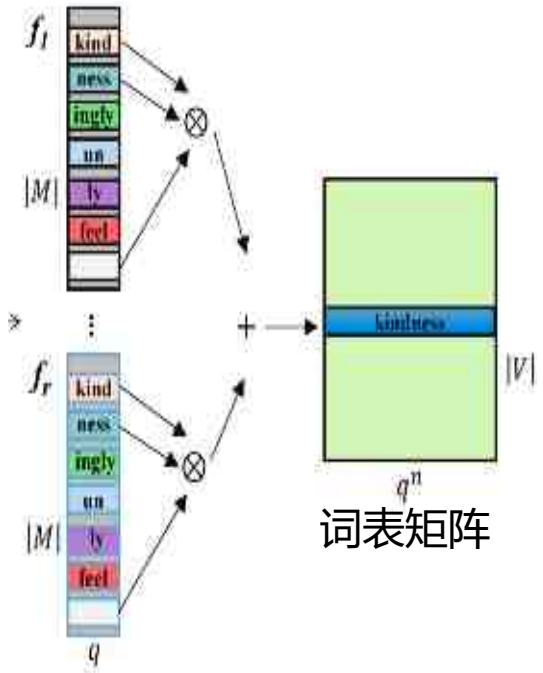
Morpheme：构成一个词的基本单位

为形态素赋予意义，引入先验知识

低维向量：张量积单词嵌入的基本单元

根据形态素分割单词、组合低维张量

形态素矩阵



实验结果

Method	De-En		En-It	
Original	34.5 (1.0×)	-	**32.9** (1.0×)	-
Matrix Factor.	32.7 (19×)	22.8 (40×)	31.0 (20×)	23.2 (42×)
Tensor Train	34.3 (20×)	33.4 (43×)	32.4 (21×)	**32.1** (43×)
Word2ketXs	34.3 (21×)	33.7 (42×)	32.6 (21×)	31.5 (43×)
Word2ket	34.2 (21×)	-	32.3 (21×)	-
MorphTE	**34.9** (21×)	**34.1** (43×)	**32.9** (21×)	**32.1** (45×)

保持原模型的有效性

参数压缩比例超过**20倍**

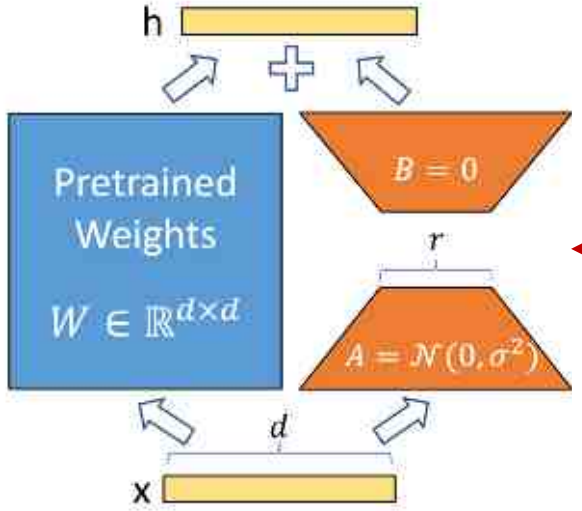
通过少数量的、低维的语素向量**替代**原始的词向量表示矩阵，保持了模型性能，从而**减少模型参数**

Gan G, **Zhang P**, Li S, et al. MorphTE: Injecting Morphology in Tensorized Embeddings[J]. NeurIPS 2022.

语素增强的低秩近似技术

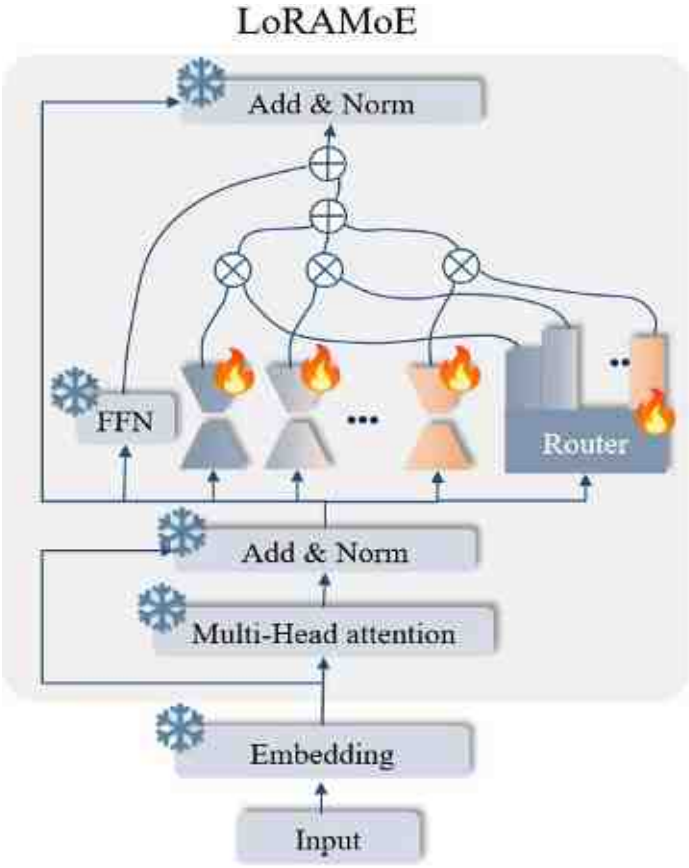
LORA: LOW-RANK ADAPTATION

冻结模型原参数，仅使用可训练的低秩分解矩阵进行模型高效微调



原模型参数规模过大，微调成本高

A、B为可训练矩阵
用于在微调中学习权重变化



LoRA已经成为大模型时代最常用的模型微调方式，有充分的研究价值。

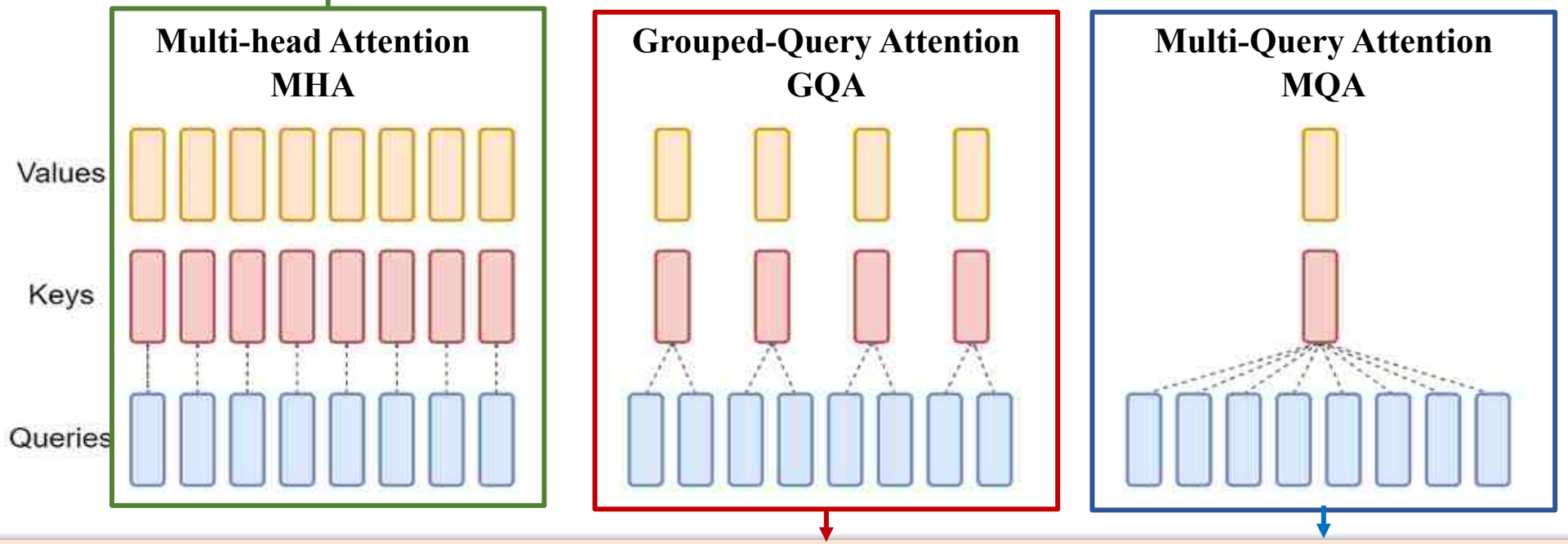
例如，近期的研究将LoRA与MoE架构结合，使一部分 LoRA 专注于利用世界知识来解决下游任务，以减轻世界知识边缘遗忘。

Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models[J]. arXiv preprint arXiv:2106.09685, 2021.

Dou S, Zhou E, Liu Y, et al. LoRAMoE: Alleviating World Knowledge Forgetting in Large Language Models via MoE-Style Plugin[C]. ACL 2024.

参数共享

问题：MHA中，每个“头”都需要独立工作，这就需要很多资源（计算量和内存）。
当头数很多时，这会变得很麻烦，就像请了很多朋友参加聚会，每个人都要吃饭，费用自然很高。



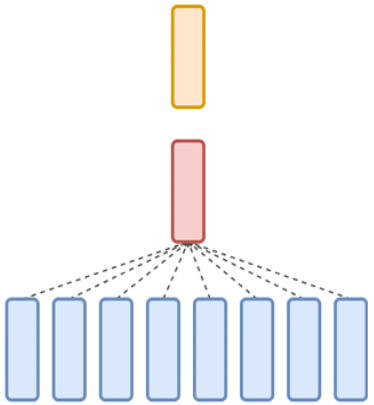
解决方案：多个查询头（Query）共享相同的键（Key）和值（Value）矩阵。
就像让几个朋友共同用同一个资源，不用每个人都从头开始找。
这种共享资源的方式大大减少了需要处理的内容，从而节省了资源。

Ainslie J, Lee-Thorp J, de Jong M, et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints[C]. EMNLP 2023.

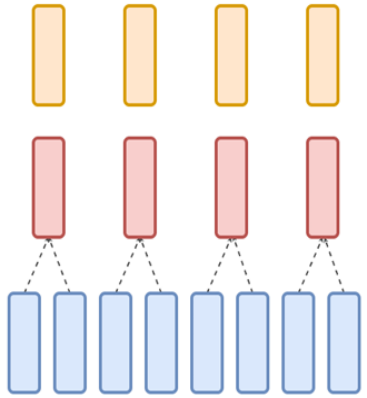
Shazeer N. Fast Transformer Decoding: One Write-Head is All You Need[J]. arXiv preprint arXiv:1911.02150, 2019.

参数共享

多查询注意力（**Multi-Query Attention, MQA**）让**所有的头之间共享同一份 Key 和 Value 矩阵**，每个头只单独保留了一份 Query 参数，从而大大减少 Key 和 Value 矩阵的参数量。



分组查询注意力（**Grouped-Query Attention, GQA**）将查询头分成 G 组，**每个组共享一个 Key 和 Value 矩阵**。GQA-G 是指具有 G 组的 Grouped-query Attention。



KV-cache

核心思想：将之前计算的键和值存储起来，当处理新的输入时，可以直接利用这些已缓存的键和值，而不是重新计算整个序列的键和值。

优势：

效率提升：减少重复计算，特别是在处理长序列时，可以显著提高处理速度；

实时性增强：适用于实时更新的场景，如在线学习或流式处理，可快速响应新数据。

优点：减少了计算的复杂度和内存的占用，可以让模型运行得更快，占用更少的资源。

Ainslie J, Lee-Thorp J, de Jong M, et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints[C]. EMNLP 2023.

Shazeer N. Fast Transformer Decoding: One Write-Head is All You Need[J]. arXiv preprint arXiv:1911.02150, 2019.

参数共享

MQA和GQA在不同数据集上推理速度、预测效果的实验结果分析

Model	T_{infer}	Average	CNN	arXiv	PubMed	MediaSum	MultiNews	WMT	TriviaQA
	s		R_1	R_1	R_1	R_1	R_1	BLEU	F1
MHA-Large	0.37	46.0	42.9	44.6	46.2	35.5	46.6	27.7	78.2
MHA-XXL	1.51	47.2	43.8	45.6	47.5	36.4	46.9	28.4	81.9
MQA-XXL	0.24	46.6	43.0	45.0	46.9	36.1	46.5	28.5	81.3
GQA-8-XXL	0.28	47.1	43.5	45.4	47.7	36.3	47.2	28.4	81.6

与同体量的MHA大模型相比，MQA的平均推理时间加速了约6.29倍，但在多个数据集上出现性能的小幅衰减。

采用MQA和GQA两种注意力后模型的平均推理时间缩短了5-6倍，同时模型的平均性能几乎不变。

与同体量的MHA大模型相比，GQA的平均推理时间加速了约5.39倍，在多个数据集上出现性能的轻微衰减。与MQA相比，GQA保持了更高的性能表现。

Ainslie J, Lee-Thorp J, de Jong M, et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints[C]. EMNLP 2023.

Shazeer N. Fast Transformer Decoding: One Write-Head is All You Need[J]. arXiv preprint arXiv:1911.02150, 2019.

结合硬件特点的技术

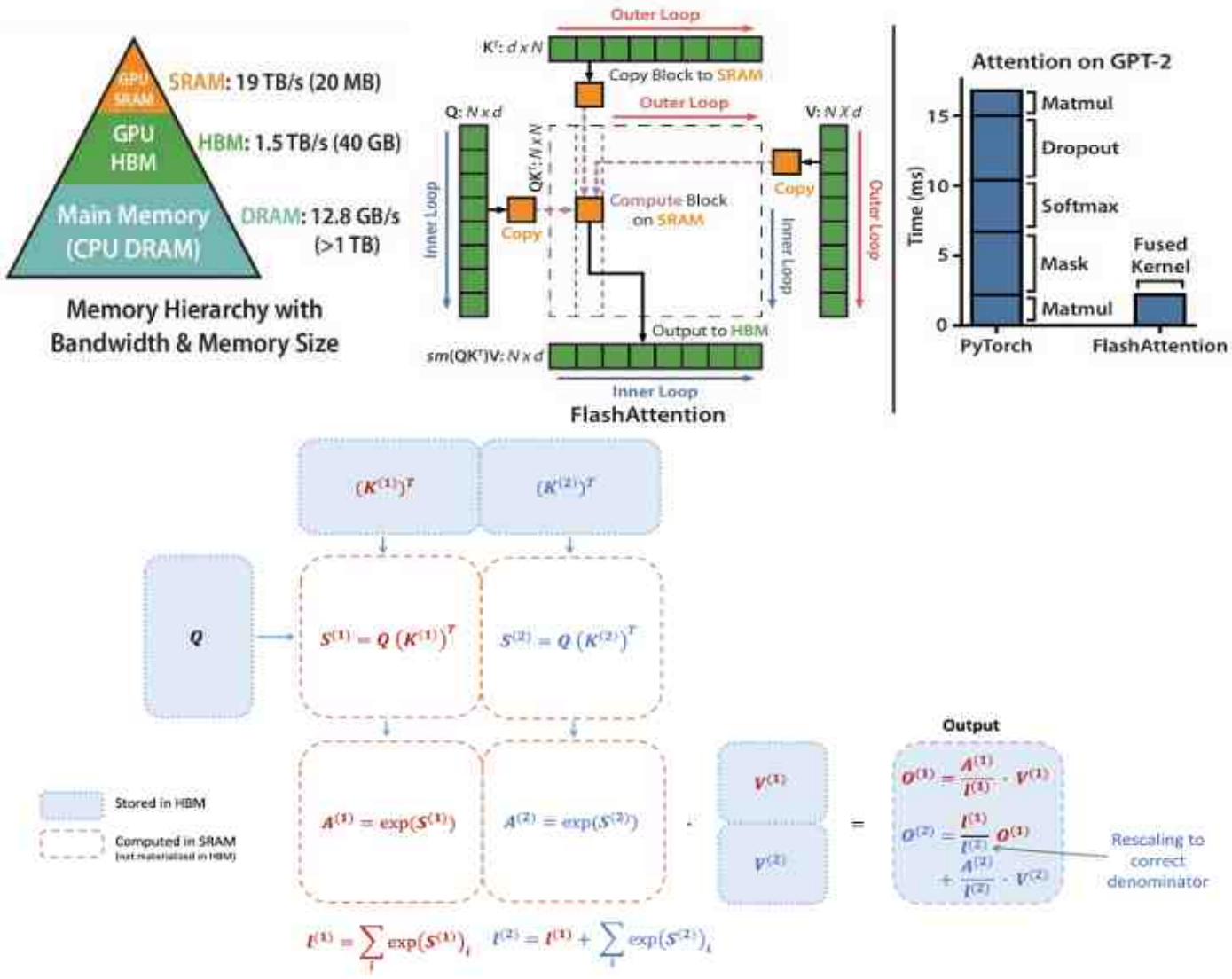
◆ Flash Attention

➢ **减少存取操作次数：**

减少语言模型自回归计算过程对HBM(内存)访问次数，充分利用SRAM内存，通过融合计算，实现一次存取，加速运算；

➢ **分块优化计算：**

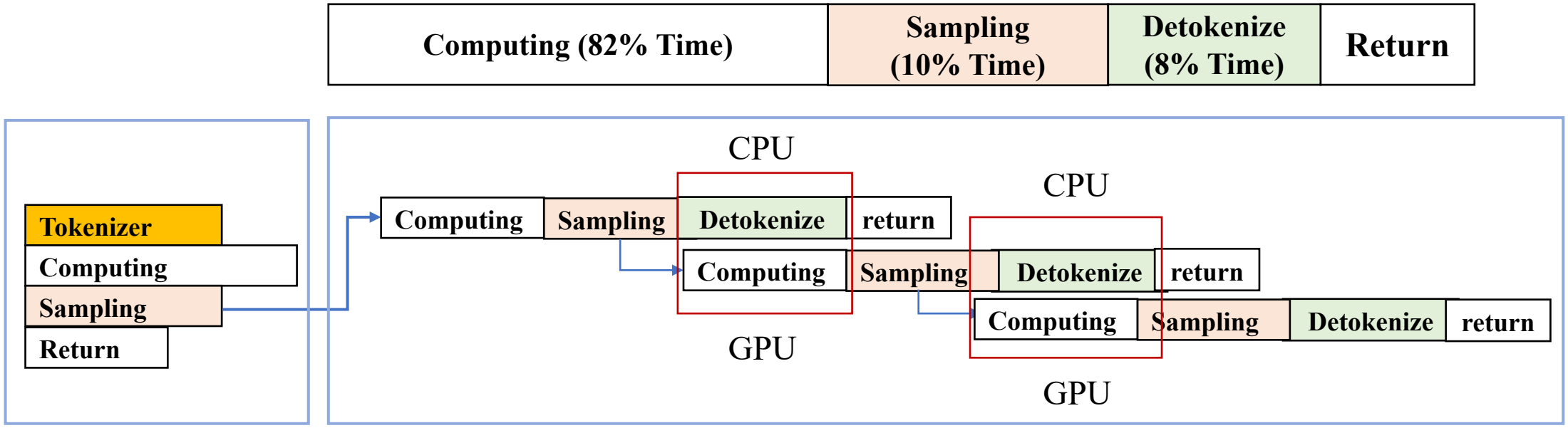
优化Softmax归一化计算步骤，通过对注意力计算过程中的K和V矩阵进行分块，解决超出SRAM缓存问题



结合硬件特点的技术

并行解码策略可**减少7.2%**的推理时间，**提升吞吐量**，不影响模型效果

Prefill（10%）　　Decoding（90%）



递归解码阶段，可以将**Detokenize**和下一个token的**Computing**计算在CPU和GPU上**并行**计算，**掩盖**掉前面生成单词的Detokenize的时间

各类轻量化方法总结

从以下多个角度进行评价

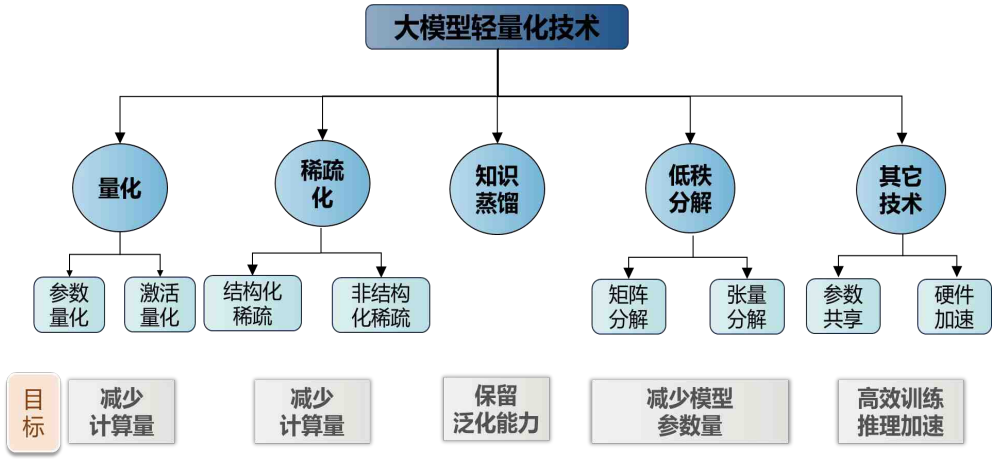
压缩方案	最高压缩率	是否需要额外训练	可否自由控制压缩比例	可优化结构	可否加速	模型效果	可否联合使用
量化	32倍	通常不需要	否	全部参数	是	位宽低时显著变差	是
稀疏化	自适应	是	是	全部参数	是	稀疏率变大时显著变差	是
知识蒸馏	自适应	是	是	全部参数	是	属于辅助增强算法	是
参数共享	有限	通常不需要	是	层级结构 块状结构	否	多层共享效果显著变差	是
低秩近似	自适应	是	是	全部参数	一些低阶的分解方案可加速	效果保持能力较强	是

目录

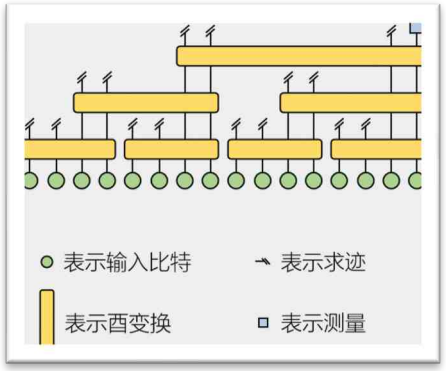
量子计算

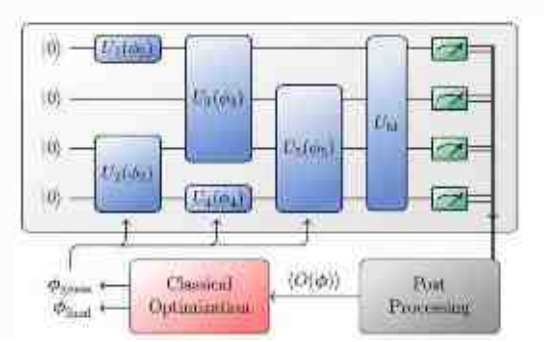
◆ 大模型轻量化的新研究路径

- **经典计算机架构**上的**大模型轻量化技术**
  - 量化
  - 稀疏化
  - 知识蒸馏
  - 低秩分解
  - 其他



- 量子计算架构上的**轻量化技术**
  - 量子变分线路





万能（通用）近似性定理

Theorem (万能 (通用) 近似定理 (Universal Approximation Theorem))

令 $\phi(\cdot)$ 是一个非常数、有界、单调递增的连续函数，I_d 是一个 d 维的单位超立方体 $[0,1]^d$，$C(I_d)$ 是定义在 I_d 上的连续函数集合。对于任何一个函数 $f \in C(I_d)$，存在一个整数 m，和一组实数 $v_i, b_i \in \mathbb{R}$ 以及实数向量 $w_i \in \mathbb{R}^d, i = 1, \cdots, m$，以至于我们可以定义函数

$$F(x) = \sum_{i=1}^{m} v_i \phi(w_i^T x + b_i), \tag{1}$$

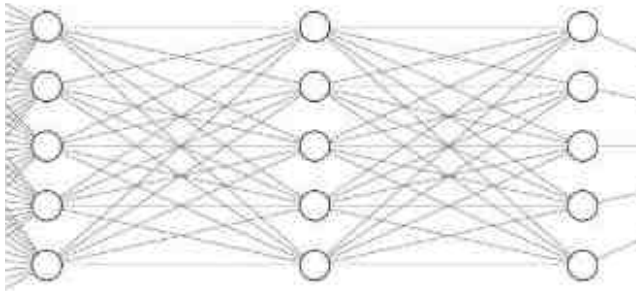
作为函数 f 的近似实现，即

$$|F(x) - f(x)| < \epsilon, \forall x \in I_d. \tag{2}$$

其中 $\epsilon > 0$ 是一个很小的正数。

神经网络

神经网络能够逼近任意复杂度的连续函数



Theorem (量子模型的万能 (通用) 近似定理 (Universal Approximation Theorem))

设 $\{H_m\}$ 是一个通用哈密顿量族，$\{f_m\}$ 是通过以下公式定义的关联量子模型族：

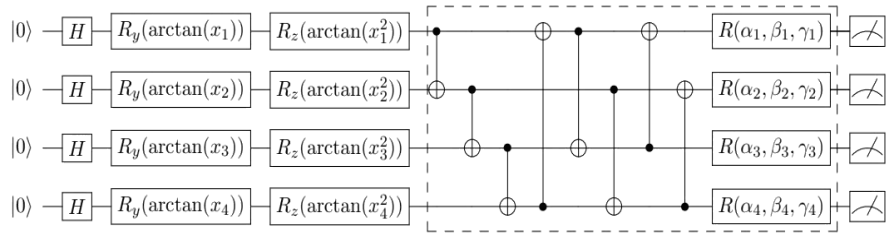
$$f_m(x) = \langle \Gamma | S_{H_m}^{\dagger}(x) M S_{H_m}(x) | \Gamma \rangle,$$

对于所有函数 $g \in L_2([0, 2\pi]^N)$，对于所有 $\epsilon > 0$，存在某个 $m' \in \mathbb{N}$，某个状态 $|\Gamma\rangle \in \mathbb{C}^{d^{m'}}$，以及某个可观测量 M，使得

$$\|f_{m'} - g\|_2 \leq \epsilon.$$

量子模型

可以实现任何可能的傅里叶系数集



Schuld, M., Sweke, R., and Meyer, J. J. Effect of data encoding on the expressive power of variational quantum machine-learning models. Physical Review A, 103(3): 032430, 2021.

量子隐式神经表征

天津大学 Tianjin University

探索数据重上传量子线路的指数级增长的傅里叶序列拟合能力

经典隐式神经表示

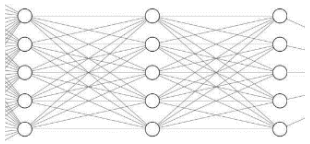
引入随机傅里叶特征

使用三角函数激活

$$g(\mathbf{x}) = \sum_{\mathbf{n}\in\mathbb{Z}^d, \|\mathbf{n}\|_\infty \le N} c_{\mathbf{n}} e^{2\pi i \frac{\mathbf{n}}{T}\cdot\mathbf{x}},$$

线性长的频谱

随神经网络参数**线性增长**的傅里叶序列表表达能力



大量的参数和有限的表示精度

量子隐式神经表示

使用**变分量子线路激活**神经元

$$f(\mathbf{h}) = \sum_{\mathbf{K},\mathbf{J}} a_{\mathbf{K},\mathbf{J}} e^{i(\mathbf{\Lambda}_\mathbf{K} - \mathbf{\Lambda}_\mathbf{J})\cdot\mathbf{h}},$$

指数长的频谱

量子优势

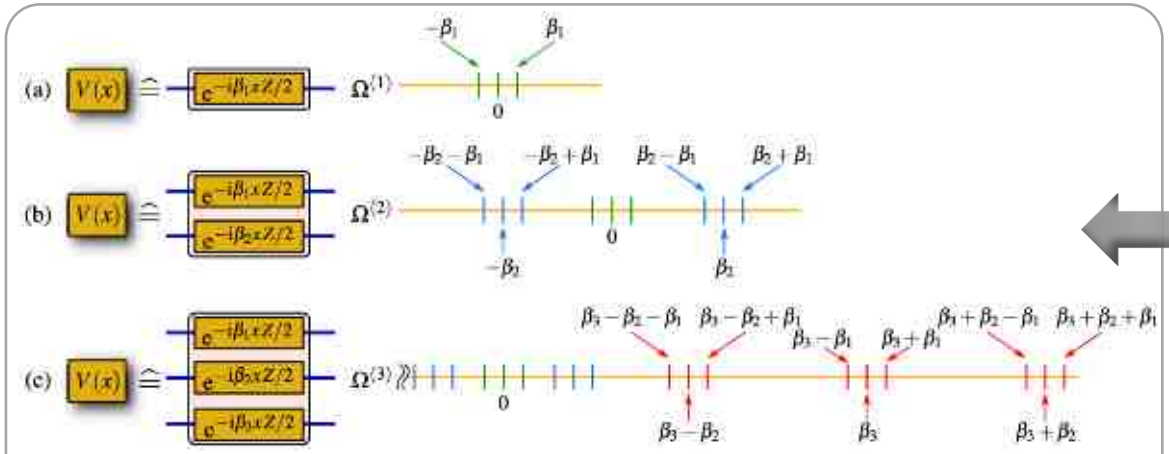
随量子比特数量**指数增长**的傅里叶序列表达能力

更少的参数和更精确的表示

Jiaming Zhao, Wenbo Qiao, **Peng Zhang***, et al. Quantum Implicit Neural Representations. **ICML, 2024 (CCF-A)**

量子隐式神经表征

在最佳条件下，数据重上传量子电路表示傅立叶级数的能力随着电路的大小呈指数增长



量子层扩展频谱

Step 1: $[U(x)|0\rangle]_i = \sum_{j_1\cdots j_L=1}^{d} e^{-i(\lambda_{j_1}+\cdots+\lambda_{j_L})x} \times W^{(L+1)}_{ij_L} \cdots W^{(2)}_{j_2 j_1} W^{(1)}_{j_1 1}$

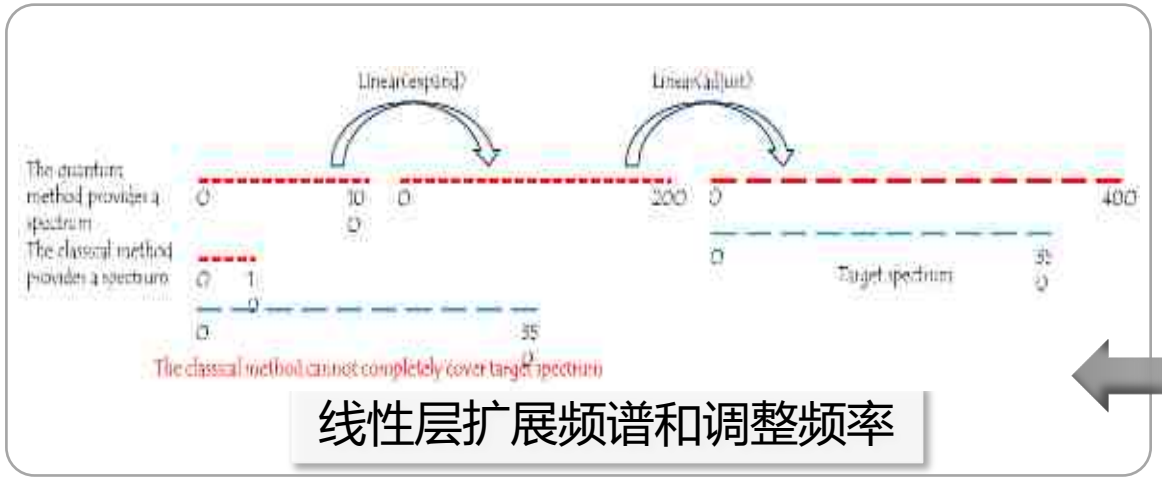
Step 2: $[U(x)|0\rangle]_i = \sum_{j\in[d]^L} e^{i\Lambda_j x} W^{(L+1)}_{ij_L} \cdots W^{(2)}_{j_2 j_1} W^{(1)}_{j_1 1}$

Step 3: $f(x) = \sum_{k,j\in[d]^L} e^{i(\Lambda_k-\Lambda_j)x} a_{k,j}$

$a_{k,j} = \sum_{i,i'} (W^*)^{(1)}_{1k_1} (W^*)^{(2)}_{j_1 j_2} \cdots (W^*)^{(L+1)}_{j_L i'} M_{i,i'} \times W^{(L+1)}_{ij_L} \cdots W^{(2)}_{j_2 j_1} W^{(1)}_{j_1 1}.$

推导

分析数据重上传电路的频谱



线性层扩展频谱和调整频率

结论

1) 数据重上传量子线路的本质是傅里叶级数：$f(x) = \sum_{K,J} a_{K,J} e^{i(\Lambda_K-\Lambda_J)\cdot x}$

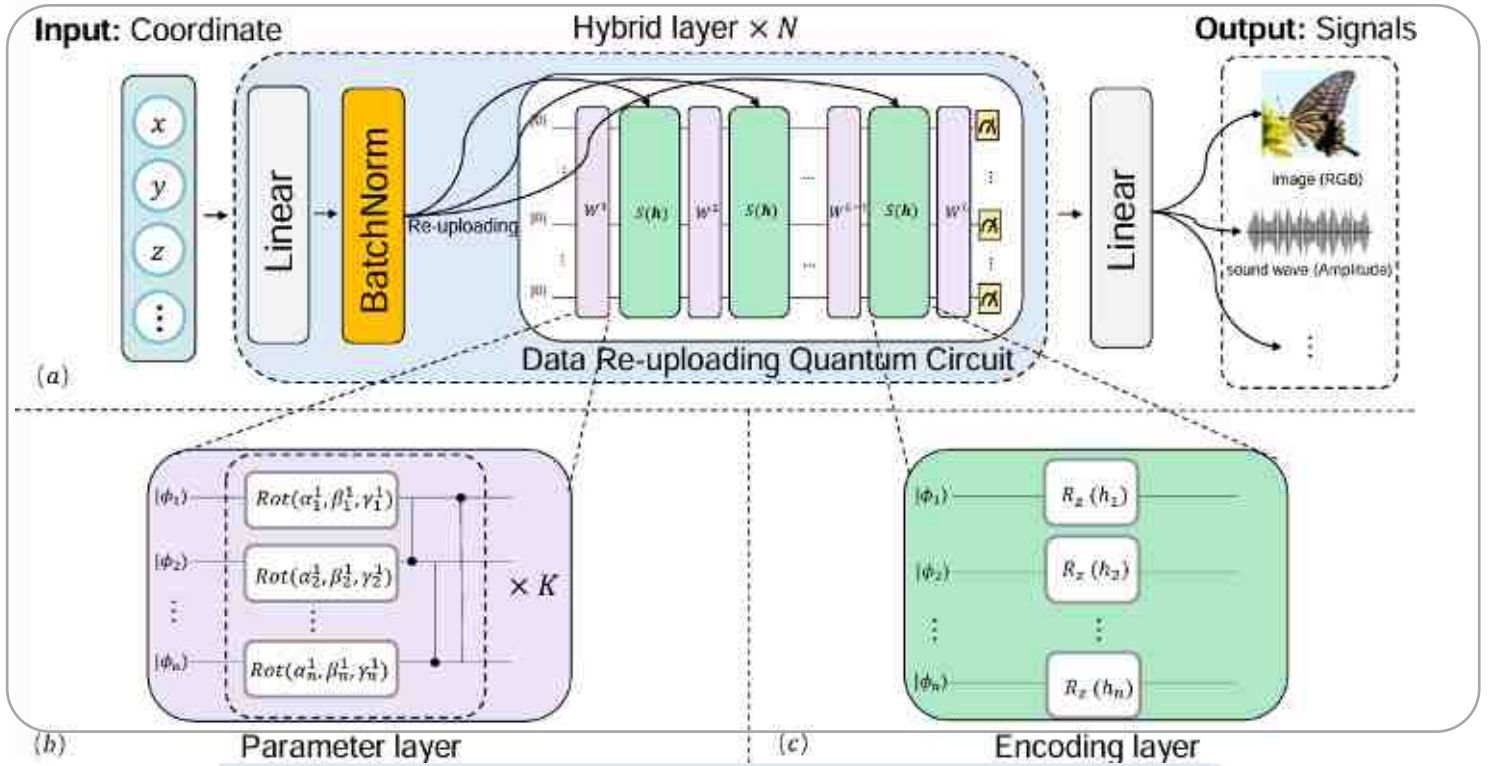
2) 对于一个大小为$d\times d_x$个qubits的数据重上传量子线路，**它能表征的傅里叶级数的的频谱大小为**：$\{\Lambda_K - \Lambda_J\}_{K,J} = \{dL \ldots dL\}^{d_x}$

3)**在线性层的帮助下，频谱可以进一步扩展**，从$(2dL+1)^{d_x}$扩展到$((3^d-1)L+1)^{d_x}$

量子隐式神经表征

结果

- 从理论上揭示了某种**量子线路具有指数级增长的傅里叶序列拟合能力**
- 量子机器学习从**理论到实践的一次跨越，为人工智能提供了量子视角的轻量化方案**



变分量子线路作为激活函数插入每层网络

高频拟合能力

更少的训练参数

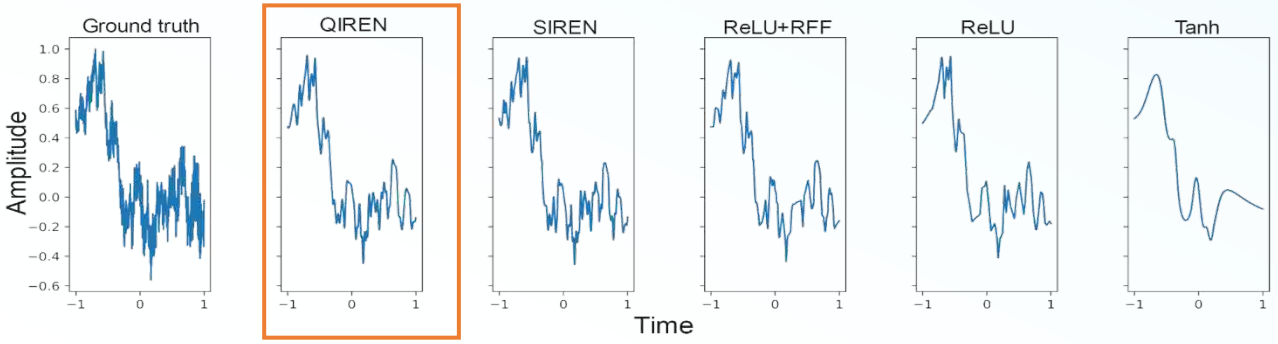
精细化表示

量子隐式神经表征

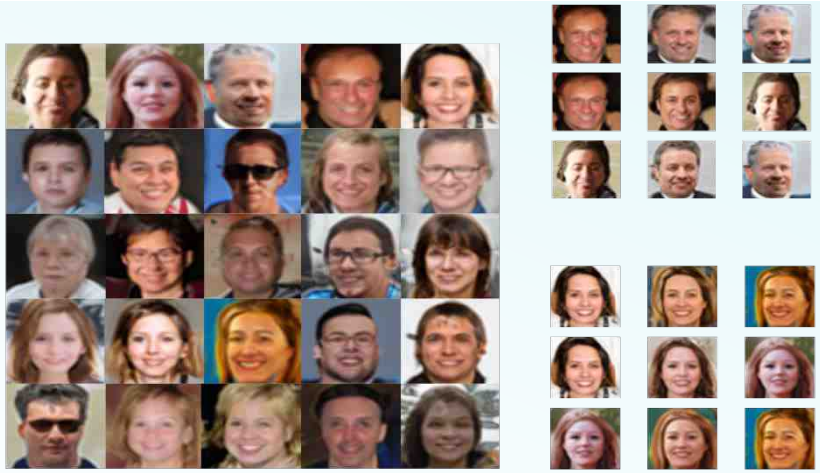
ICML International Conference On Machine Learning

在信号表征、超分辨率和图像生成等众多任务中展现出精度和参数优势

Method	Sound Representation Cello	#params	#mem(%)	Image Representation Astronaut	Camera	Coffee	#params	#mem(%)	Image Superresolution Astronaut	Camera	Coffee
Nearest	-	-	-	-	-	-	-	-	26.6	10.4	13.6
Bilinear	-	-	-	-	-	-	-	-	25.2	9.2	12.2
ReLU	6.8	831	16.9	9.9	2.7	4.2	841	17.9	33.8	11.3	13.5
Tanh	14.0	831	16.9	20.7	5.8	14.8	841	17.9	47.8	15.2	26.7
ReLU+RFF*	6.0	791	20.9	5.1	1.9	4.9	791	22.8	39.9	13.3	23.9
SIREN*	**5.5**	691	30.9	9.0	1.5	2.3	701	31.5	77.0	26.3	15.7
QIREN (ours)	**5.5**	**649**	**35.1**	**4.0**	**1.1**	**1.5**	**657**	**35.8**	**24.3**	**7.9**	**9.4**



Method	FFHQ	CelebA-HQ	#params
Tanh	26.98	25.17	1.16M
ReLU	84.94	110.81	1.16M
ReLU+RFF*	15.01	13.91	1.14M
SIREN*	22.31	20.97	1.16M
QIREN (ours)	**11.53**	**11.78**	**1.13M**



Jiaming Zhao, Wenbo Qiao, **Peng Zhang***, et al. Quantum Implicit Neural Representations. **ICML, 2024 (CCF-A)**

未来展望

人工智能也许也会像通用计算机的发展历程一样不断被轻量化，其中量子机器学习有望扮演重要角色

通用计算机

20世纪第一台通用计算机

体型庞大

晶体管

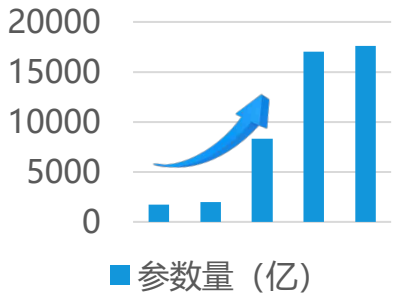
集成电路

小型化

微型个人电脑

?

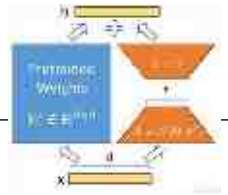
通用人工智能

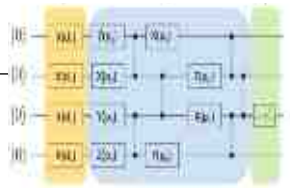


大语言模型

体型庞大

大模型轻量化

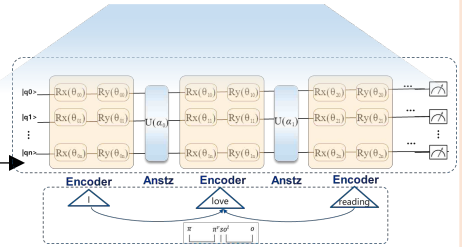




量子计算

量子化

量子机器学习



量子AI模型

天津大学

Tianjin University

请批评指正

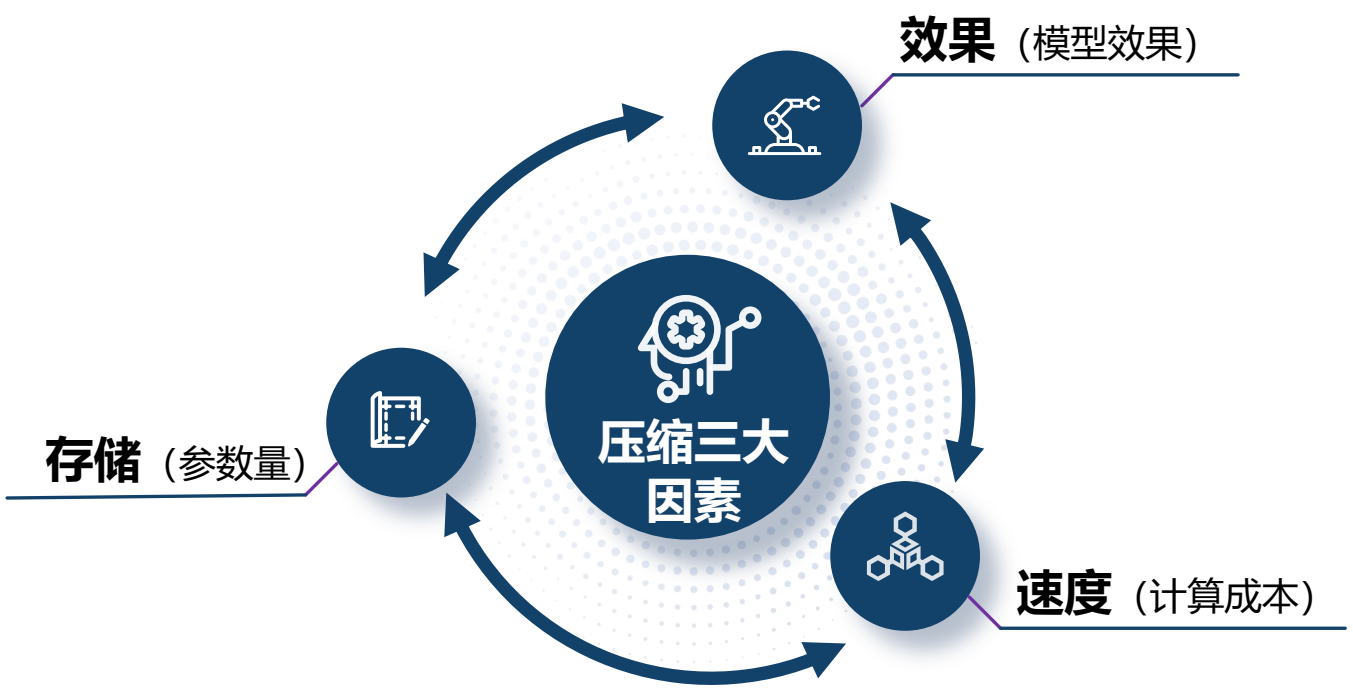
天津大学
Tianjin University

LLM稀疏轻量化技术

张静

2024.8.24

为什么考虑稀疏化



- 在轻量化三大要素中，**低秩近似方法**尤其是高阶的方法，可实现**较高的压缩率及较优的模型效果**，然而计算速度方面的优势并不凸显。
- **稀疏化技术**具备较强**降低计算成本与速度的能力**。
- 因此，通过对稀疏化技术的系统性研究，探索**低秩近似与稀疏化补充结合**的可行性。

目录

天津大学 Tianjin University

稀疏轻量化背景

- **Transformer时代前的稀疏化技术发展**

➢ 20世纪八九十年代

- 在神经网络中，LeCun等人提出了Optimal Brain Damage方法，该方法**通过剪枝不重要的权重**来优化网络结构（LeCun et al., 1989）。
- Hassibi和Stork提出了Optimal Brain Surgeon方法，通过**更精确的权重修剪进一步提高了模型的稀疏性和效率**（Hassibi & Stork, 1993）。这些早期探索拉开了后续稀疏轻量化技术发展的大门。

稀疏轻量化背景

- **Transformer时代前的稀疏化技术发展**

  - 20世纪末
    - 在1990年代，L0正则化被引入**神经网络稀疏化**，通过**惩罚非零权重**，促使模型**自发减少不必要的参数**（Tibshirani, 1996）。
    - Olshausen和Field的研究表明，**稀疏编码通过学习稀疏表示可以有效压缩信息**，使其在神经科学和计算机视觉中得到了广泛应用（Olshausen & Field, 1997）。这些技术**推动了稀疏轻量化技术的广泛应用。**

稀疏轻量化背景

- **Transformer时代前的稀疏化技术发展**

  ➢ 21世纪初

  - 2000年代，压缩感知理论的提出推动了稀疏轻量化的发展。研究表明，通过**较少采样重构信号**，可以在**不显著损失信息**的情况下**大幅降低计算量**（Donoho, 2006）。
  - **模型剪枝技术**被提出，通过移除冗余参数，在**保持模型性能的同时显著降低了计算复杂度**（Han et al., 2015）。

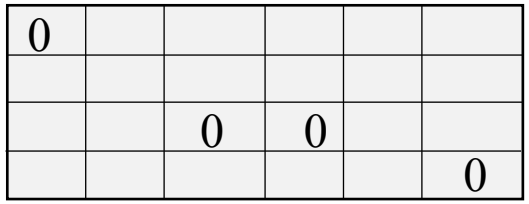
以上发展为Transformer网络的稀疏化奠定了基础

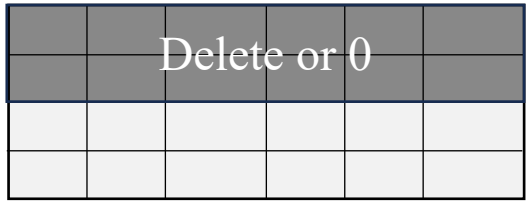
稀疏轻量化背景

- **稀疏化基本类型——参数稀疏**

大模型稀疏化通过**减少参数的密集度**来减少计算成本和存储成本。主要分为结构稀疏化和非稀疏化两种。

➢ 非结构化稀疏与结构化稀疏：





➢ 稀疏化三大要点：

1. **稀疏化评估：**使用某种标准（如权重的绝对值、梯度等）评估各参数的重要性。
2. **稀疏化：**删除或置零评估为不重要的参数。
3. **微调：**对稀疏化后的模型进行再训练，以恢复和提升模型性能。

稀疏轻量化背景

- **稀疏化基本类型——参数稀疏**

  - 结构化与非结构化的对比

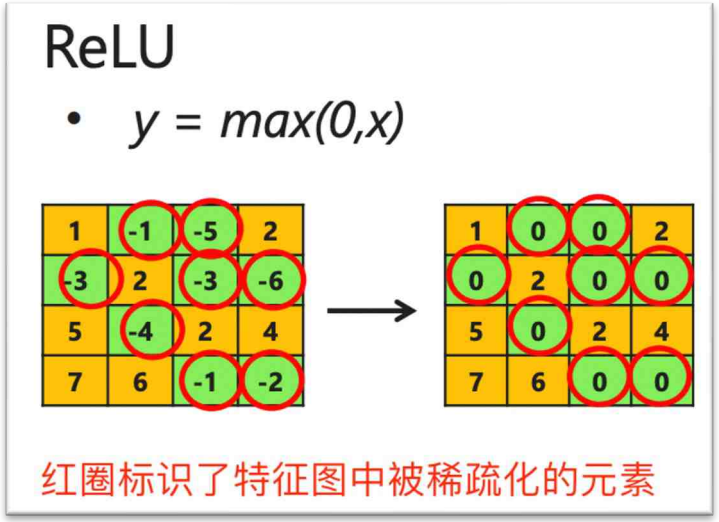
    - **结构化**：结构具有规则性，**并行计算架构中可被高效利用**；由于剪枝是成块进行的，可能会移除更多有用的信息，因此**精度损失可能较大。**
    - **非结构化：非结构化稀疏在保持模型精度方面往往优于结构化稀疏；剪枝后的稀疏性没有规则性，使得硬件难以高效利用，可能导致不规则的内存访问，影响计算效率。**

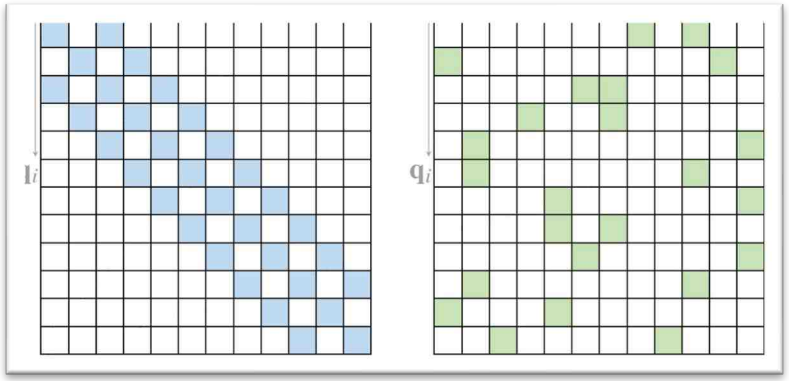
稀疏轻量化背景

- **稀疏化基本类型——中间表示稀疏**

➢ 中间表示的稀疏化

- 通过**引入激活函数等方式**。使模型的中间表示**部分元素稀疏化为0**，减少计算成本。
- 采用**如聚类等方式**，直接**减少中间表示的尺寸**，从而降低计算成本。





稀疏轻量化背景

- **稀疏化基本类型——结合自适应计算策略**

大模型的自适应计算轻量化旨在通过动态分配计算资源以优化模型性能和效率。根据输入数据的复杂度和模型的推理需求，自适应地调整计算路径，从而在保证精度的前提下减少计算开销。

- **自适应激活策略：** $g_i(x)$为门控函数，用于选择性激活对应的子网络 $f_i(x)$ 。

$$y = \sum_{i=1}^{n} g_i(x) \cdot f_i(x)$$

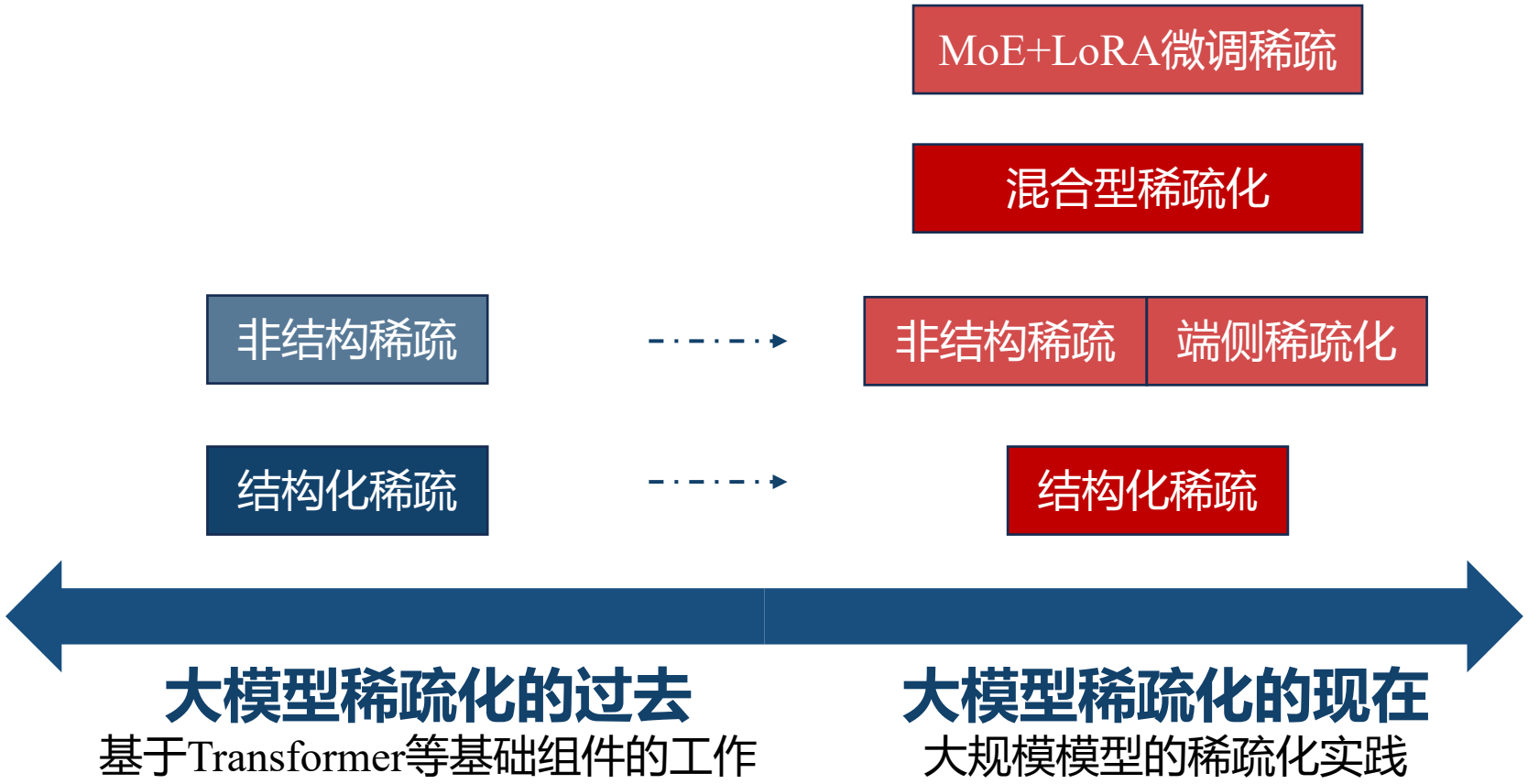
- **早停策略：** 根据输入数据的复杂度动态决定网络层的深度，在满足一定条件时提前终止计算。

稀疏轻量化背景

- **大模型稀疏化的过去和现在——关注点**

  - Transformer稀疏化的更关注什么？
    - 主要关注**预训练阶段的稀疏化**
    - 更关注**参数与训练速度**
  - 生成式大模型稀疏化的更关注什么？
    - 主要普遍行业更关注**微调与推理的计算成本**
    - 更关注**吞吐量、带宽及显存等受限的成本**

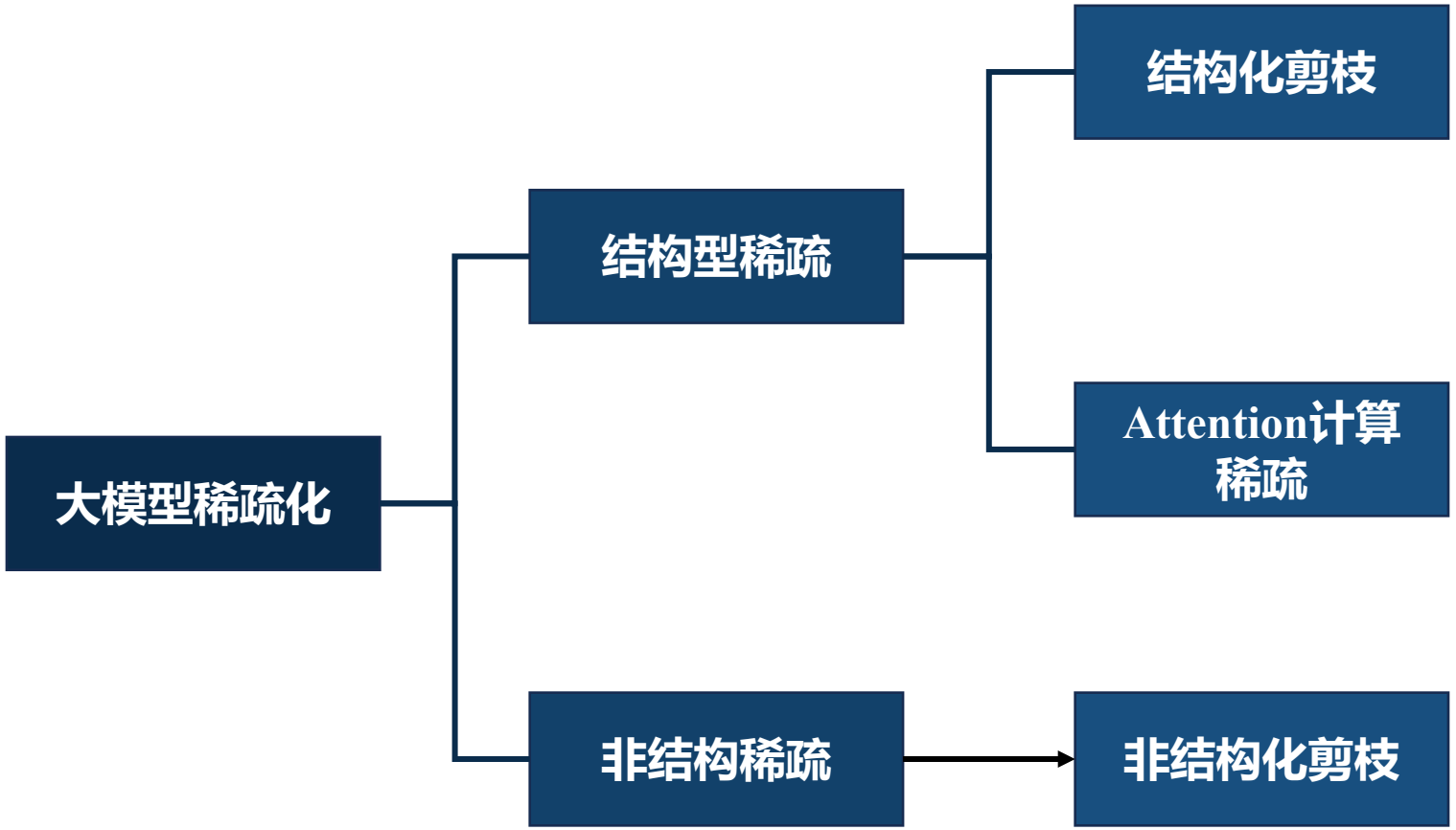
稀疏轻量化背景


MoE+LoRA微调稀疏
混合型稀疏化
非结构稀疏
非结构稀疏
端侧稀疏化
结构化稀疏
结构化稀疏
大模型稀疏化的过去
基于Transformer等基础组件的工作
大模型稀疏化的现在
大规模模型的稀疏化实践

目录

天津大学 Tianjin University

本章脉络



大模型稀疏化的过去——Transformer时代

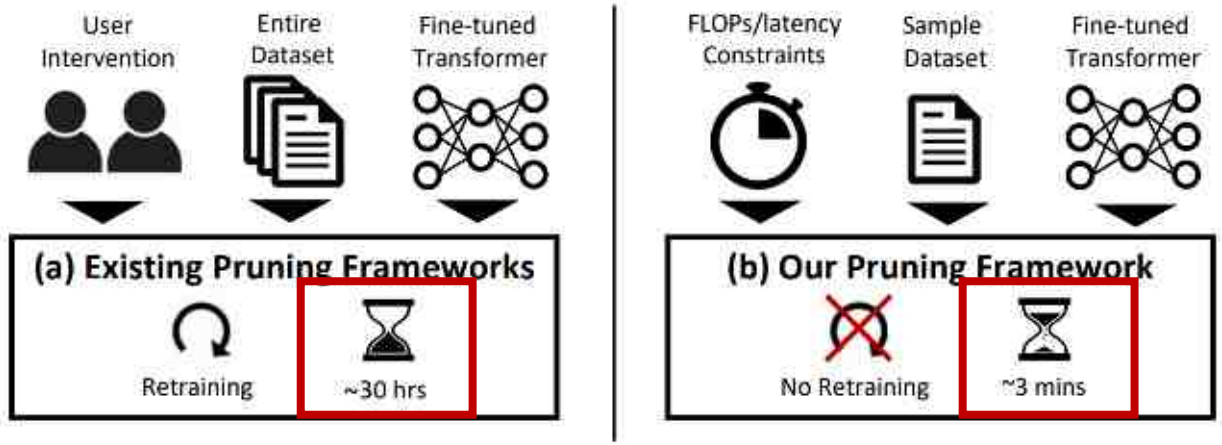
- **结构化剪枝——挑战**

  - 结构性稀疏一般存在两个问题
    - 结构化稀疏由于**限制了剪枝元素的选择自由**，会导致**模型效果的下降**；
    - 进行结构化稀疏的**最后一步需要进行微调**，**微调**全模型会**产生大量的计算成本**。

Kwon W, Kim S, Mahoney M W, et al. A fast post-training pruning framework for transformers[J]. NeurIPS 2022.

大模型稀疏化的过去——Transformer时代

- 结构化剪枝



1. 基于Fisher信息矩阵对角化的掩码搜索，以确定各层修剪比例；
2. 基于层内相互作用的掩码重排列以确定各层的修剪位置，找到最优二值掩码；
3. 基于线性最小二乘的掩码微调，以找到最优实值掩码，尽可能恢复模型性能。

掩码量<<模型参数量

Kwon W, Kim S, Mahoney M W, et al. A fast post-training pruning framework for transformers[J]. NeurIPS 2022.

大模型稀疏化的过去——Transformer时代

$$\arg\min_{\mathrm{m}} \mathcal{L}(\mathrm{m}) \quad \text{s.t.} \quad \mathrm{Cost}(\mathrm{m}) \le C \quad \Rightarrow \quad \arg\min_{\mathrm{m}} \mathcal{L}(\mathrm{m}) \approx \arg\min_{\mathrm{m}} (\mathbb{1}-\mathrm{m})^{\intercal}\mathrm{H}(\mathbb{1}-\mathrm{m}).$$

- **确定修剪比例：**针对Cost优化问题，由于Hessian矩阵难以精确构建，因此将其近似为Fisher信息矩阵对角化。

$$\mathcal{I} := \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \left(\frac{\partial}{\partial \mathrm{m}} \mathcal{L}(x, y; \mathbb{1})\right) \left(\frac{\partial}{\partial \mathrm{m}} \mathcal{L}(x, y; \mathbb{1})\right)^{\intercal},$$

Hessian矩阵是什么，其复杂度是多少？

$$H = \begin{bmatrix} \frac{\partial^2 l}{\partial w_1^2} & \frac{\partial^2 l}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 l}{\partial w_1 \partial w_n} \\ \frac{\partial^2 l}{\partial w_2 \partial w_1} & \frac{\partial^2 l}{\partial w_2^2} & \cdots & \frac{\partial^2 l}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 l}{\partial w_n \partial w_1} & \frac{\partial^2 l}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 l}{\partial w_n^2} \end{bmatrix}$$

帮助评估每个参数块对损失函数的二阶敏感性

Fisher信息矩阵是什么？为什么与Hessian近似

$$I(\mathbf{w}) = \mathbb{E}\left[\left(\frac{\partial \log p(\mathbf{x};\mathbf{w})}{\partial \mathbf{w}}\right)\left(\frac{\partial \log p(\mathbf{x};\mathbf{w})}{\partial \mathbf{w}}\right)^{\top}\right]$$

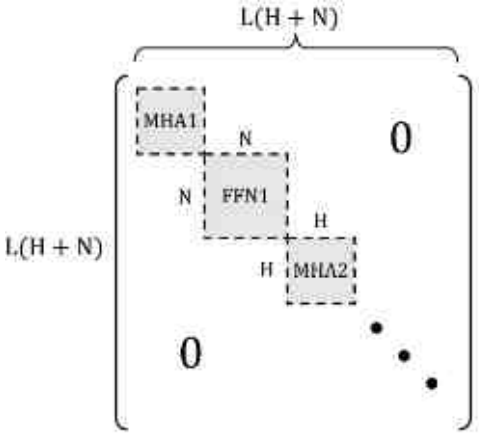
当损失函数为负对数似然时

$$H \approx \mathbb{E}\left[\left(\frac{\partial \log p(\mathbf{x};\mathbf{w})}{\partial \mathbf{w}}\right)\left(\frac{\partial \log p(\mathbf{x};\mathbf{w})}{\partial \mathbf{w}}\right)^{\top}\right] = I(\mathbf{w})$$

Kwon W, Kim S, Mahoney M W, et al. A fast post-training pruning framework for transformers[J]. NeurIPS 2022.

大模型稀疏化的过去——Transformer时代

- **最优二值掩码：** Fisher块对角近似与热启动贪婪搜索相结合，以避免由于不同掩码变量间存在的相互关系导致的性能下降。



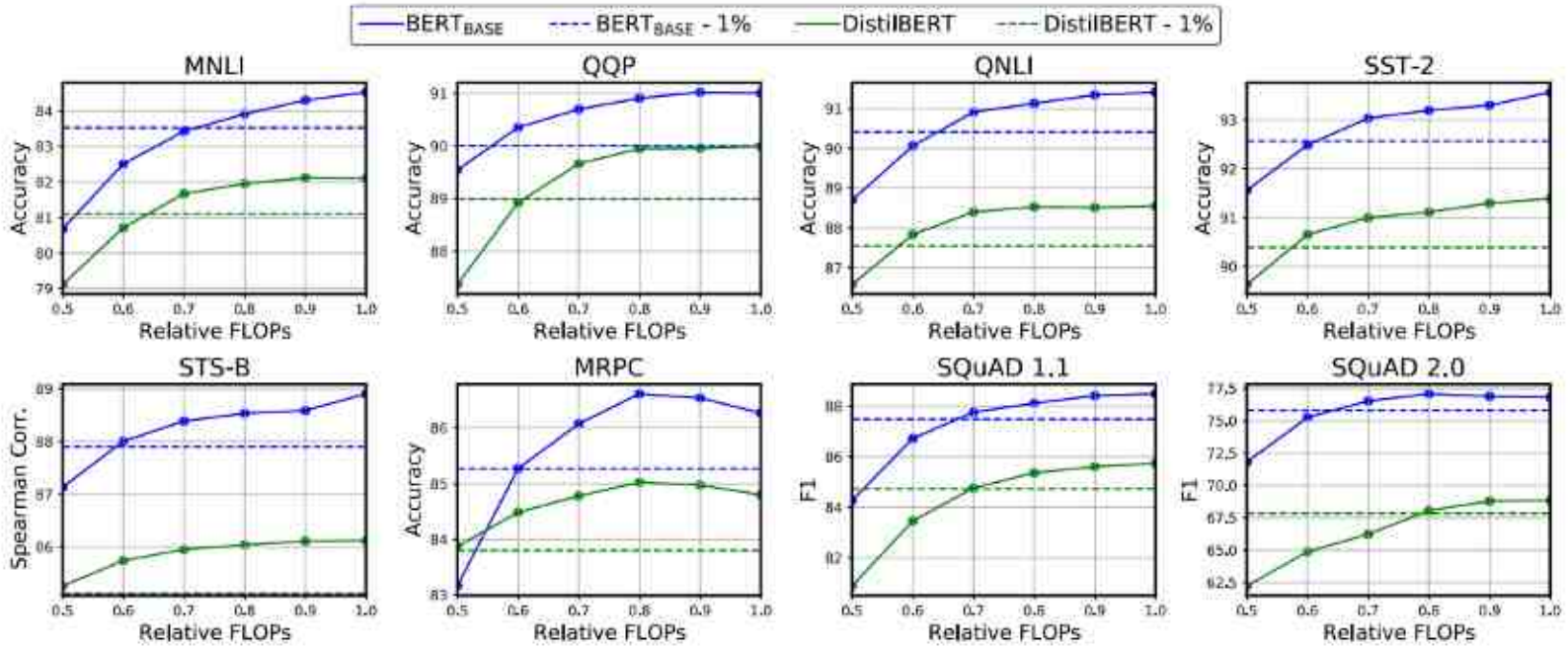
- **掩码微调：** 非零变量被调整为任意实数值，通过线性最小二乘法进行逐层重构，以使剪枝后的模型恢复其准确性。

$$\arg\min_{\mathbf{m}_l} ||\mathbf{x} + \text{layer}(\mathbf{x}; \mathbf{m}_l) - \left(\mathbf{x}' + \text{layer}(\mathbf{x}'; \mathbb{1})\right)||_2^2,$$

每层的输出

Kwon W, Kim S, Mahoney M W, et al. A fast post-training pruning framework for transformers[J]. NeurIPS 2022.

大模型稀疏化的过去——Transformer时代



	# Epochs	E2E time (hr)
DynaBERT [25]	4	12
EBERT [49]	6	5
BMP [38]	20	17
CoFi [88]	40	33
Ours	**0**	**0.01**

与其他蒸馏方法的时间比较

应用于具有不同 FLOPs 约束的 BERTBASE 和 DistilBERT 的准确性。

Batch size	MNLI	QQP	QNLI	SST-2	STS-B	MRPC	SQuAD$_{1.1}$	SQuAD$_{2.0}$	Geo. mean
32	1.27×	1.42×	1.42×	1.23×	1.34×	1.36×	1.33×	1.37×	1.34×
256	1.34×	1.54×	1.53×	1.56×	1.54×	1.55×	1.34×	1.40×	1.47×

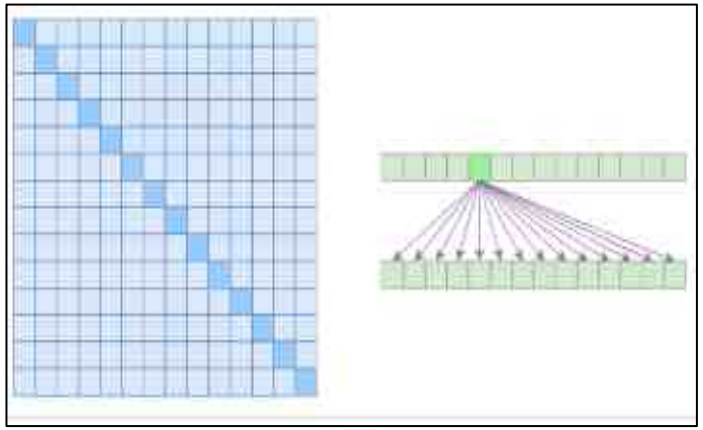
在仅降低1%效果的约束下，满足约束条件的最大延迟加速

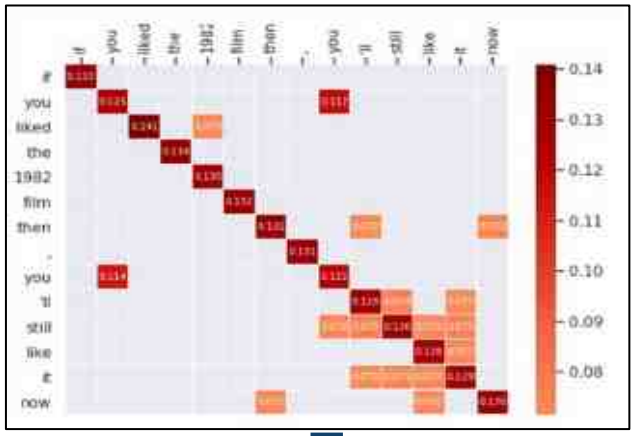
Kwon W, Kim S, Mahoney M W, et al. A fast post-training pruning framework for transformers[J]. NeurIPS 2022.

大模型稀疏化的过去——Transformer时代

自注意机制：需要计算输入文本序列中任意两个单词之间的关联。

注意力机制加速





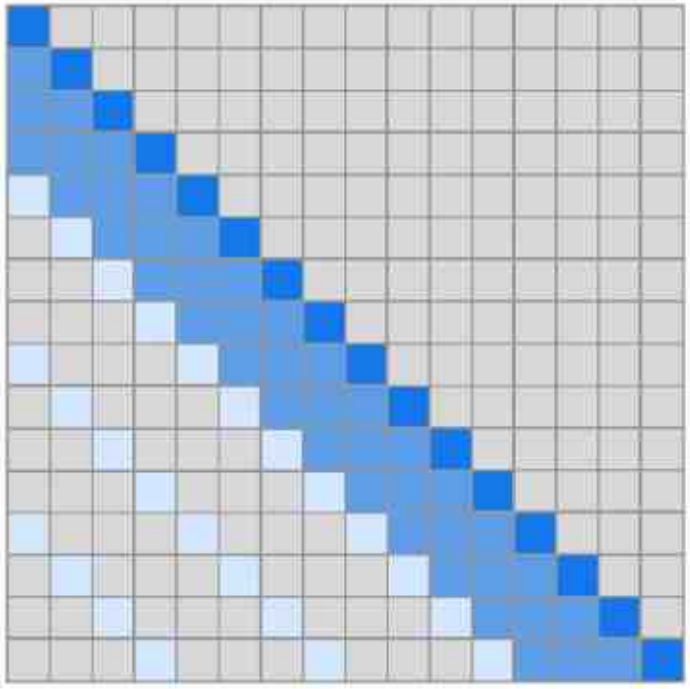
二次复杂度

稀疏

利用稀疏化技术进行低秩逼近，减少不必要的计算以加速模型。

大模型稀疏化的过去——Transformer时代

- 稀疏化Transformer——结构化稀疏



稀疏因式分解结合空洞自注意力

$$O(N^2) \rightarrow O(N\sqrt{N})$$

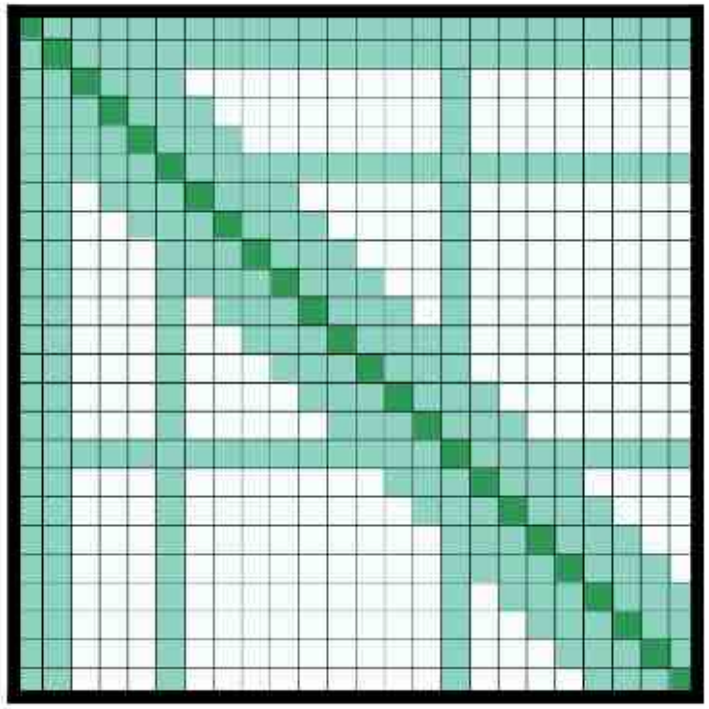
Model	Bits per byte
CIFAR-10	
PixelCNN (Oord et al., 2016)	3.03
PixelCNN++ (Salimans et al., 2017)	2.92
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2017)	2.85
Sparse Transformer 59M (strided)	**2.80**
Enwik8	
Deeper Self-Attention (Al-Rfou et al., 2018)	1.06
Transformer-XL 88M (Dai et al., 2018)	1.03
Transformer-XL 277M (Dai et al., 2018)	**0.99**
Sparse Transformer 95M (fixed)	**0.99**
ImageNet 64x64	
PixelCNN (Oord et al., 2016)	3.57
Parallel Multiscale (Reed et al., 2017)	3.7
Glow (Kingma & Dhariwal, 2018)	3.81
SPN 150M (Menick & Kalchbrenner, 2018)	3.52
Sparse Transformer 152M (strided)	**3.44**
Classical music, 5 seconds at 12 kHz	
Sparse Transformer 152M (strided)	**1.97**

在图像，文本和语音上Bits per byte指标都达到最低

Child R, Gray S, Radford A, et al. Generating long sequences with sparse transformers. arXiv:1904.10509

大模型稀疏化的过去——Transformer时代

- 局部稀疏化——结构化稀疏



局部窗口稀疏化+部分全局注意力

$$O(N^2) \rightarrow O(n\times(w+g))$$

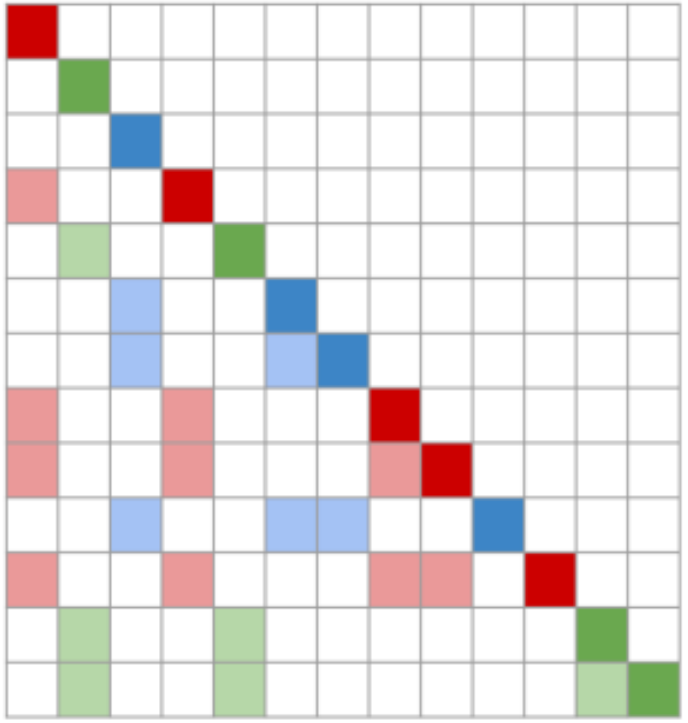
Model	#Param	Dev	Test
Dataset `text8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.18
Adaptive (Sukhbaatar et al., 2019)	38M	1.05	1.11
BP-Transformer (Ye et al., 2019)	39M	-	1.11
Our Longformer	41M	1.04	**1.10**
Dataset `enwik8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.11
Transformer-XL (Dai et al., 2019)	41M	-	1.06
Reformer (Kitaev et al., 2020)	-	-	1.05
Adaptive (Sukhbaatar et al., 2019)	39M	1.04	1.02
BP-Transformer (Ye et al., 2019)	38M	-	1.02
Our Longformer	41M	1.02	**1.00**

在text8和enwik8的实验显示同等参数下获得到更好的效果

Beltagy I, Peters M E, Cohan A. Longformer: The long-document transformer. arXiv: 2004.05150

大模型稀疏化的过去——Transformer时代

- 动态路由算法——结构化稀疏



动态路由稀疏化

$O(N^2) \rightarrow O(nlogn)$ or $O(n)$

Model	Layers	Heads	Perplexity
LSTMs (Grave et al., 2017)	-	-	40.8
QRNNs (Merity et al., 2018)	-	-	33.0
Adaptive Transformer (Sukhbaatar et al., 2019)	36	8	20.6
Local Transformer	16	16	19.8
Adaptive Input (Baevski and Auli, 2019)	16	16	18.7
TransformerXL (Dai et al., 2019)	18	16	18.3
Routing Transformer	10	16	**15.8**

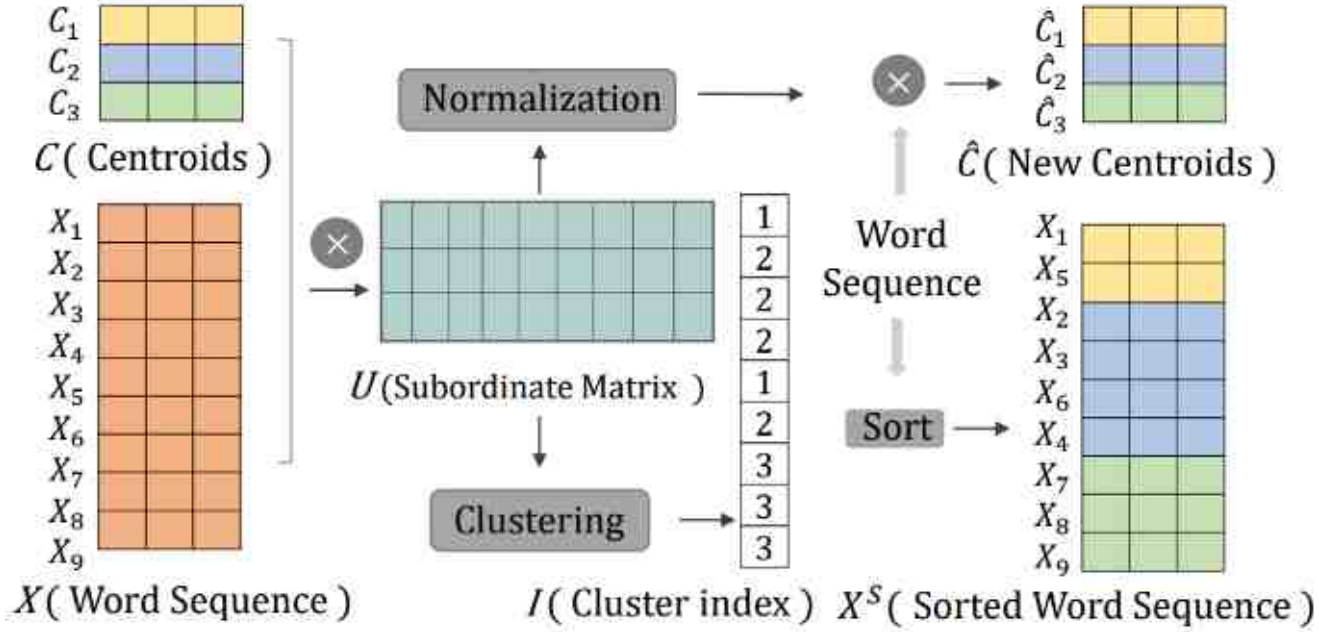
以更少的计算成本获得到更好的效果

Roy A, Saffar M, Vaswani A, et al. Efficient content-based sparse attention with routing transformers[J]. TACL, 2021.

大模型稀疏化的过去——Transformer时代

- **神经聚类方法——结构化稀疏**

神经聚类方法：学习聚类中心的表示，并在输入单词序列上进行端到端的聚类。



输入序列　　**神经聚类过程**

关系矩阵

$$U_{ij} = \frac{exp\left(\emptyset\left(C_i, X_j W^c\right)\right)}{\sum_{j=1}^{N} exp\left(\emptyset\left(C_i, X_j W^c\right)\right)}$$

聚类中心更新

$$\widehat{C_i} = Noramlize\left(\left(\sum_{j=1}^{N} U_{ij} X_j\right) + C\right)$$

聚类中心矩阵

$$I'_j = Argmax(U'_{:j})$$

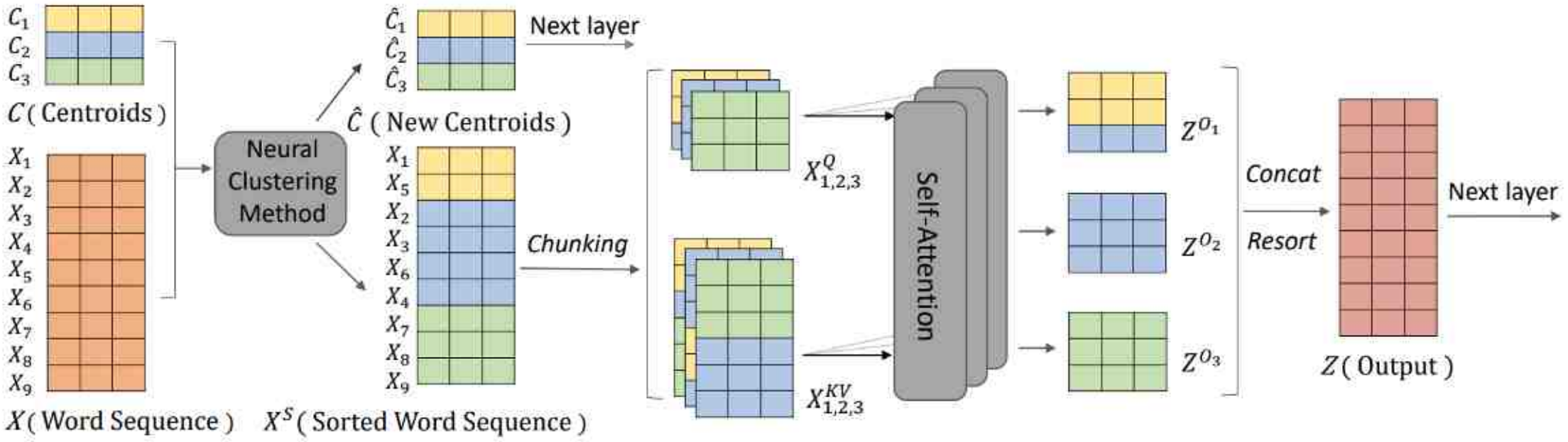
排序后的表示

$$X^S, I' = Sort(X, I)$$

Wang N., Gan G., **Zhang P.**, et al. ClusterFormer: Neural Clustering Attention for Efficient and Effective Transformer [J]. ACL, 2022, 1: 2390–2402.

大模型稀疏化的过去——Transformer时代

神经聚类注意力：对每组查询（Query）、键（Key）和值（Value）块进行并行的注意力机制计算。



$$Z^{O_i} = Attention(Q^{O_i}, K^{O_i}, V^{O_i})$$

$$Z^{O} = BlockConcat(Z^{O_1}, \dots, Z^{O_k})$$

Model Complexity： $O(N^2) \rightarrow O(N\sqrt{N})$

Wang N., Gan G., **Zhang P.**, et al. ClusterFormer: Neural Clustering Attention for Efficient and Effective Transformer [J]. ACL 2022, 1: 2390–2402.

大模型稀疏化的过去——Transformer时代

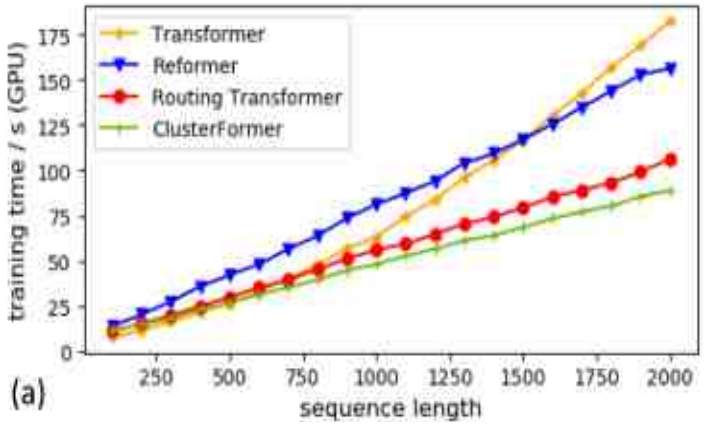
(1) 翻译任务

Model	Test BLEU	
	IWSLT14(De-En)	WMT14(En-De)
Variational Attention	33.1	–
Random Feature Attention	34.6	–
Transformer	34.4	27.3/26.4
Reformer	34.0	26.3/25.4
Routing Transformer	32.5	24.3/23.6
ClusterFormer	**34.9**	**27.4/26.5**

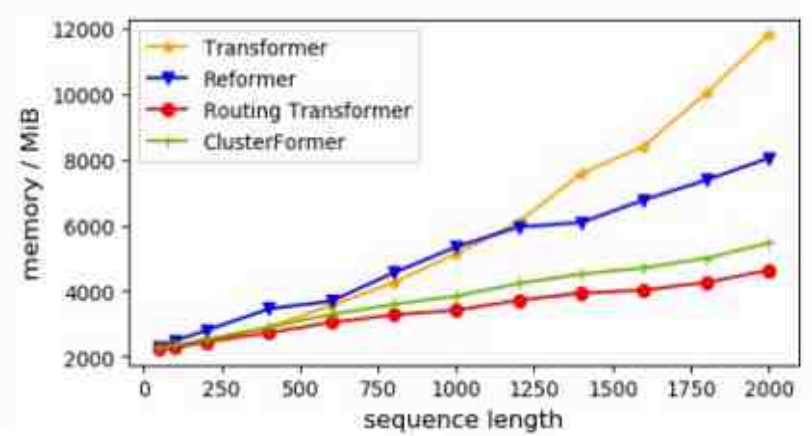
(2分类任务

Model	CR	MR	SUBJ	MPQA	20NEWS	Average
DiSAN (Shen et al., 2018)	84.8	–	94.2	90.1	–	–
MPSAN (Dai et al., 2020)	85.4	–	94.6	**90.4**	–	–
Transformer† (Vaswani et al., 2017)	86.2	81.8	95.4	89.9	83.6	87.38
Reformer† (Kitaev et al., 2020)	83.0	79.7	94.7	88.6	81.7	85.54
Routing Transformer† (Roy et al., 2021)	80.1	78.8	94.3	81.2	81.3	83.14
ClusterFormer	**88.1**	**82.7**	**96.2**	**90.4**	**83.8**	**88.24**

(3) 时间测试



(4) 显存测试



The longer the sequence, the more noticeable the efficiency improvement

Wang N., Gan G., **Zhang P.**, et al. ClusterFormer: Neural Clustering Attention for Efficient and Effective Transformer [J]. ACL, 2022, 1: 2390–2402.

大模型稀疏化的过去——Transformer时代

- **FFN与Attention稀疏策略——非结构化稀疏**

Transformer模型的问题：训练和微调计算成本高昂；

在FFN层和QKV分别采用了不同的稀疏化策略：

- 在FFN层中，每个块**只允许一个浮点数非零**，论文提出了一个**基于低秩分解思想**的控制器，最终输出一个独热编码：

$$\text{Controller}(x) = \arg\max(\text{Reshape}(xC_1C_2, (-1, N)))$$

$$Y_{\text{sparse}} = \max(0, xW_1 + b_1) \odot \text{Controller}(x)$$
$$\text{SparseFFN}(x) = Y_{\text{sparse}}W_2 + b_2$$

- 在Attention层中，该工作使用了两种策略，首先是**乘法密集层**将$d \times d$的权重矩阵变成$d \times (M + S)$的矩阵，其中M$\times S = d$，并通过$y_{s,m} = \sum_i x_i D_{i,s} E_{i,m}$的方式计算输出。同时可以结合**二维卷积层**来降低模型的参数量。

	Params	Dec. time	Dec. time per block
baseline Transf.	800M	0.160s	5.9ms
+ Sparse FF	-	0.093s	3.1ms
+ Sparse QKV	-	0.152s	6.2ms
+ Sparse FF+QKV	-	0.061s	1.9ms
Speedup		2.62x	3.05x
baseline Transf.	17B	3.690s	0.581s
+Sparse FF	-	1.595s	0.259s
+Sparse QKV	-	3.154s	0.554s
+Sparse FF+QKV	-	0.183s	0.014s
Speedup		20.0x	42.5x

解码时间加速20倍

Jaszczur S, Chowdhery A, Mohiuddin A, et al. Sparse is enough in scaling transformers[J]. NeurIPS 2021.

大模型稀疏化的过去——Transformer时代

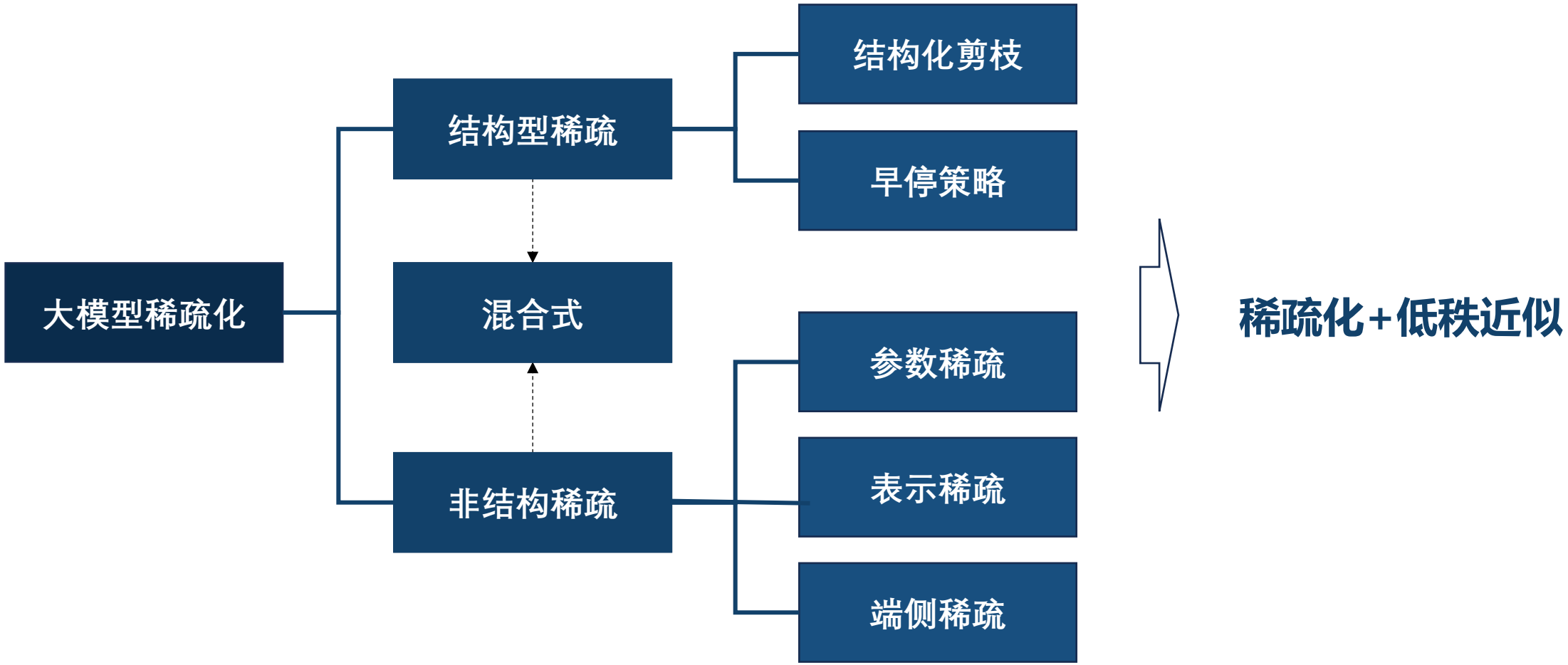
- **Some thoughts**

  - 可以看到在Transformer组件上的稀疏化工作大多是需要再次进行训练，在超大模型规模下，这种稀疏成本是难以让人接受的。
  - 如何将现有方法与大模型解耦，类似于控制器的学习改装为LoRA的形式，减少稀疏化技术实现本身的时间成本，也是大模型稀疏化主要关注的问题。

目录

天津大学 Tianjin University

本章脉络

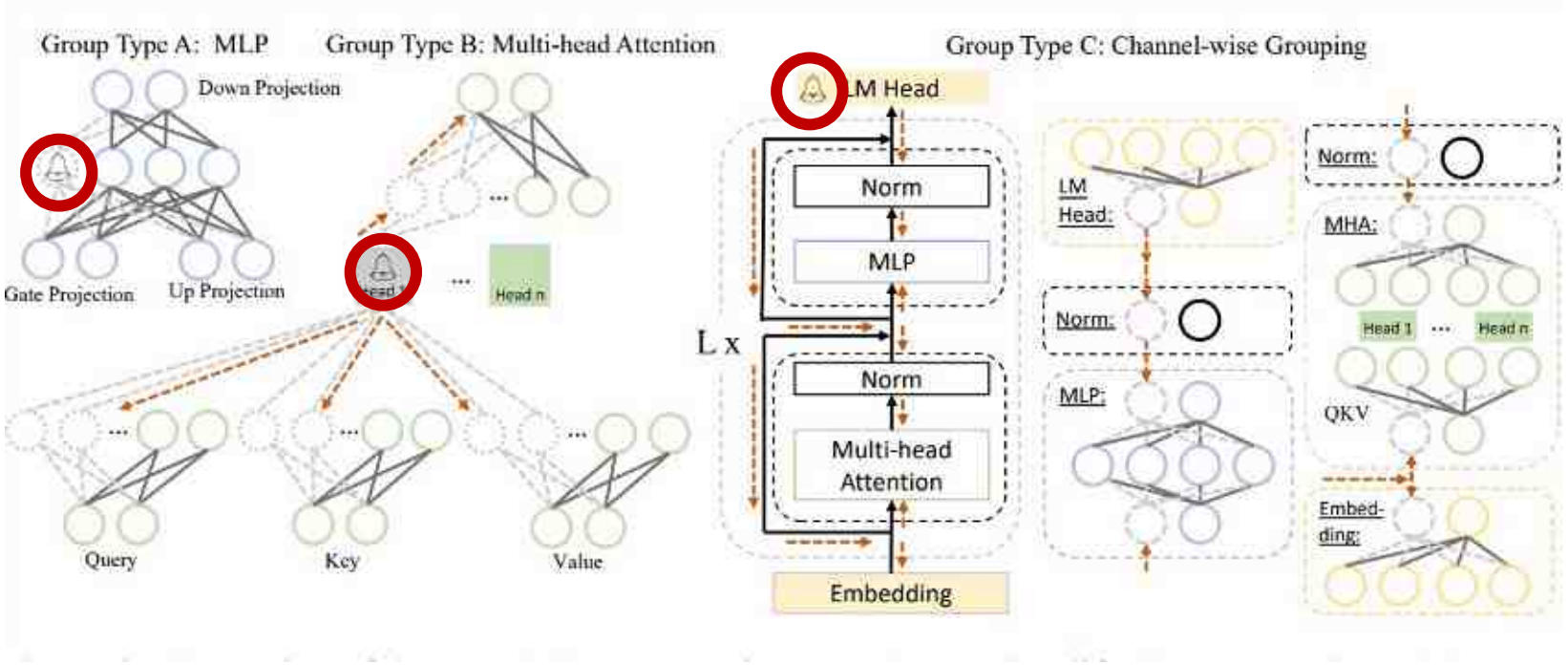


LLM稀疏化技术

- **LLM剪枝——结构稀疏化**

挑战

大语言模型的参数量更多，不同参数剪枝元素间存在大量的依赖性关系，贸然剪枝会造成模型效果下降。



参数块间存在依赖性关系

- MLP型分组
- Attention型分组
- 层型分组

Ma X, Fang G, Wang X. Llm-pruner: On the structural pruning of large language models[J]. NeurIPS 2023.

LLM稀疏化技术

重要性估计 → 剪枝 → LoRA微调

- 重要性估计方面：利用损失的偏差来度量来移除对模型预测影响最小的组，公式计算了当某个参数 被置为零时，损失变化的影响：

$$I_{W_i^k} = |\mathcal{L}_{W_i^k}(\mathcal{D}) - \mathcal{L}_{W_i^k=0}(\mathcal{D})| \approx |\frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i^k} W_i^k - \frac{1}{2} \sum_{j=1}^{N} \left(\frac{\partial \mathcal{L}(\mathcal{D}_j)}{\partial W_i^k} W_i^k \right)^2 + \mathcal{O}\left(\|W_i^k\|^3\right)|$$

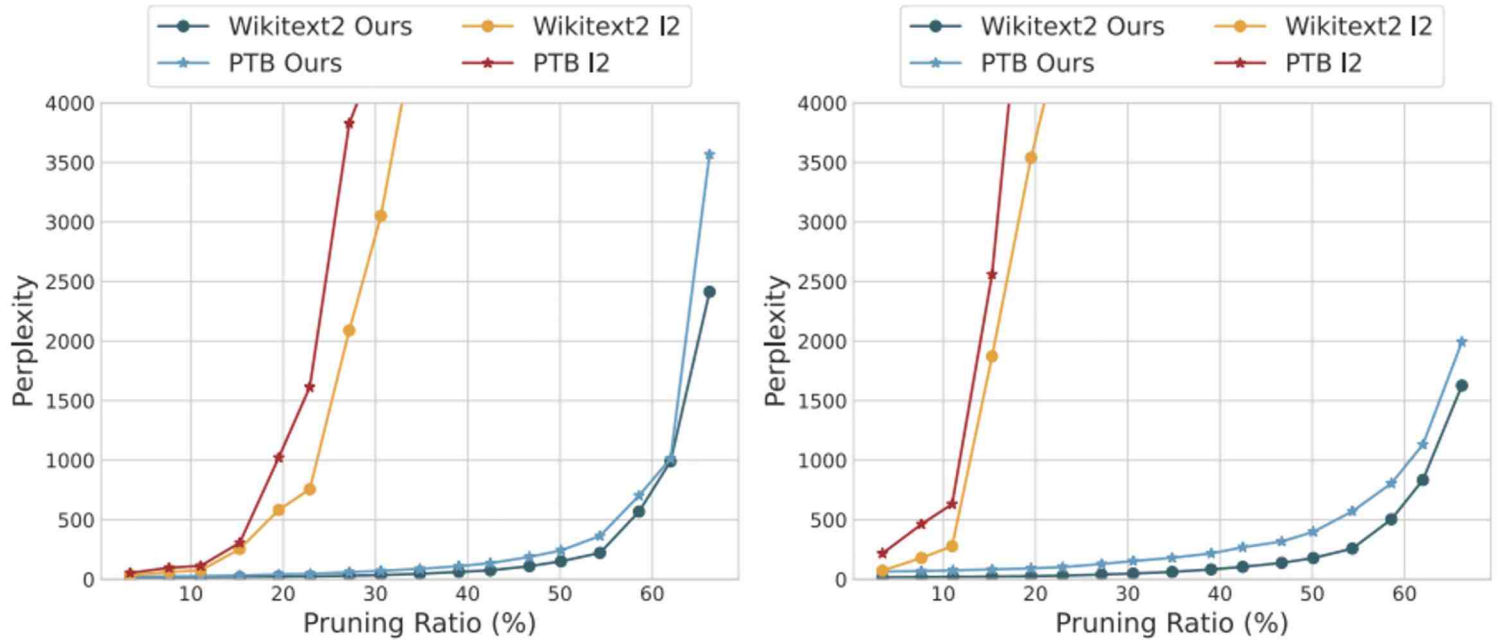
- 剪枝：对分组进行聚合，按照相加、乘或门控等操作，聚合出分组的重要性得分，并进行剪枝；
- 利用LoRA的方式对剪枝后的模型进行微调：

$$f(x) = (W + \Delta W)X + b = (WX + b) + (PQ)X$$

Ma X, Fang G, Wang X. Llm-pruner: On the structural pruning of large language models[J]. NeurIPS 2023.

LLM稀疏化技术

Model	Strategy	Ratio	#Params	#MACs	Memory	Latency
LLaMA-7B Vicuna-7B	-	-	6.74B	424.02G	12884.5MiB	69.32s
	Channel	20%	5.39B	339.36G	10363.6MiB	61.50s
	Block	20%	5.42B	339.60G	10375.5MiB	58.55s
	Channel	50%	3.37B	212.58G	6556.3MiB	40.11s
	Block	50%	3.35B	206.59G	6533.9MiB	37.54s

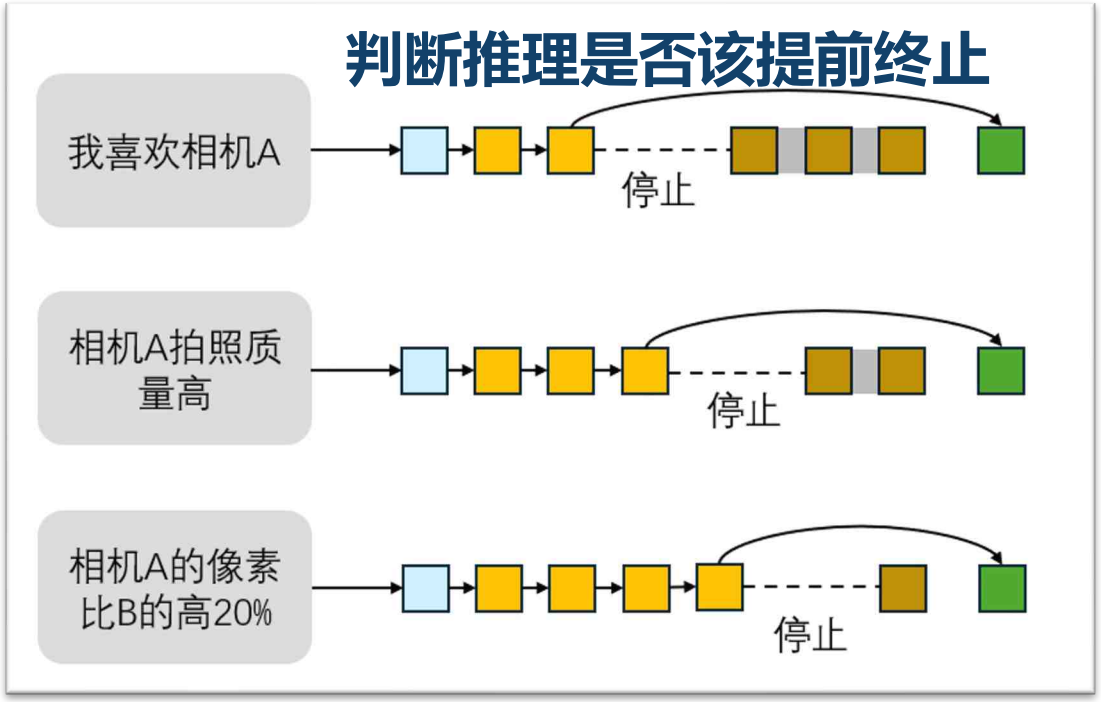


- 在**参数量、MACs及延迟**等多个轻量化性能指标上都展现了有益的效果；
- 传统的剪枝方法在较小稀疏率下，效果出现显著下降，而**LLM-Pruner能进行更高倍压缩**；
- 效果随参数量的增大而上升，**符合LLM scaling law**。

Ma X, Fang G, Wang X. Llm-pruner: On the structural pruning of large language models[J]. NeurIPS 2023.

LLM稀疏化技术

- **早停策略——结构稀疏化**



	轻量化后		LLaMA2-13B
性能	-3%	+0%	92.58
计算量	-40%	-20%	100

在GLUE Benchmark数据集上，**减少了40%的计算量**，并**实现了效果的稳定**。

- 这是一种**不需要牺牲带宽的LLM加速方法**；
- 在LLM推理过程中，可以**对简单实例使用浅层，对困难实例使用深层**；
- 对LLM的特征进行统计分析并选择logits构建特征，采用高效的SVM和CRF等方法来促进提前退出策略；

Fan S, Jiang X, Li X, et al. Not all layers of llms are necessary during inference[J].

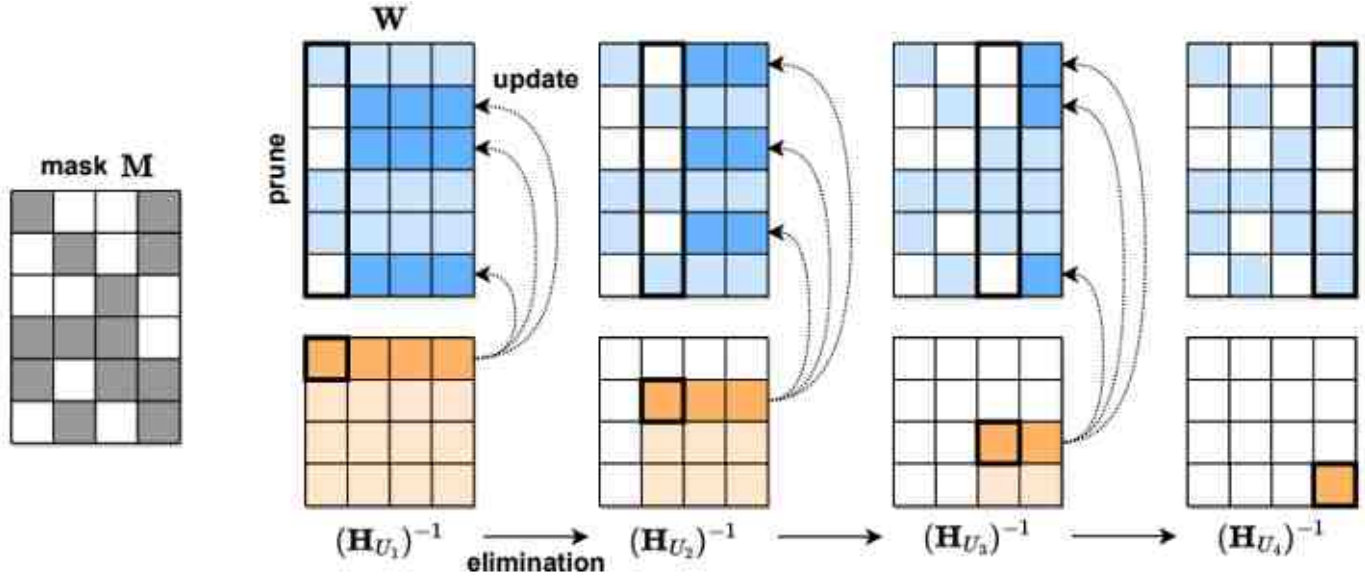
- **Some thoughts**

  - 早停思想本身是基于一种普适性的启发，具备不同复杂度的输入所需要的层数是有差别的，现有的方法通过引入外部“控制器”的方式实现早停判断
  - 是否可以通过扩散模型一些自适应控制计算的方式（Cheng Lu, 2022），结合大模型本身的结构特点，通过内部误差计算的方式实现早停?

LLM稀疏化技术

- **SparseGPT——非结构性稀疏**

➢ 挑战：在巨型LLM上，一次性剪枝方法通常需要花费大量的时间来计算完整的最优解。



计算复杂度：$O(d_{hidden}^4) \rightarrow O(d_{hidden}^3)$

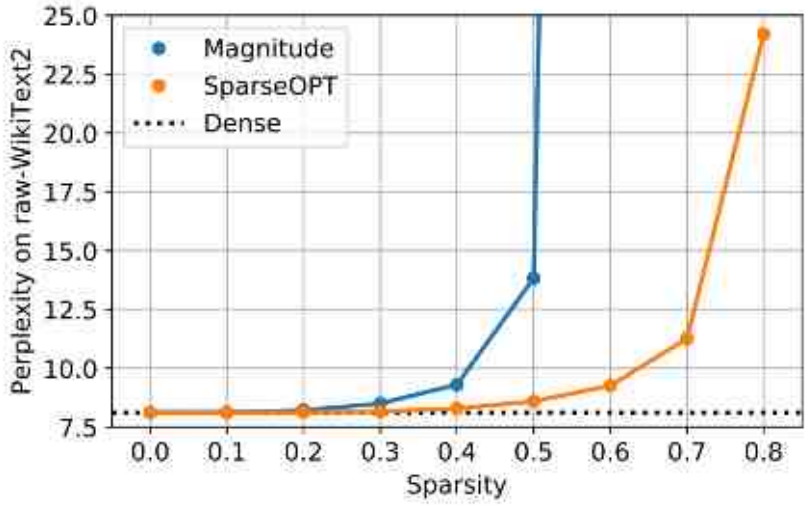
➢ 基于OBS更新方法的理论，当修剪某一参数时，**此时调整其他column对应的权重，并且局部更新Hessian矩阵**，将会弥补剪枝该参数造成的误差。

Frantar, Elias, and Dan Alistarh. "Sparsegpt: Massive language models can be accurately pruned in one-shot." ICML 2023.

LLM稀疏化技术

- **SparseGPT——非结构性稀疏**

图一：BLOOM 176B上的稀疏率实验



相较于其他方法，在较高稀疏率下能保持模型的效果

表一：在WikiText2上的OPT模型PPL结果

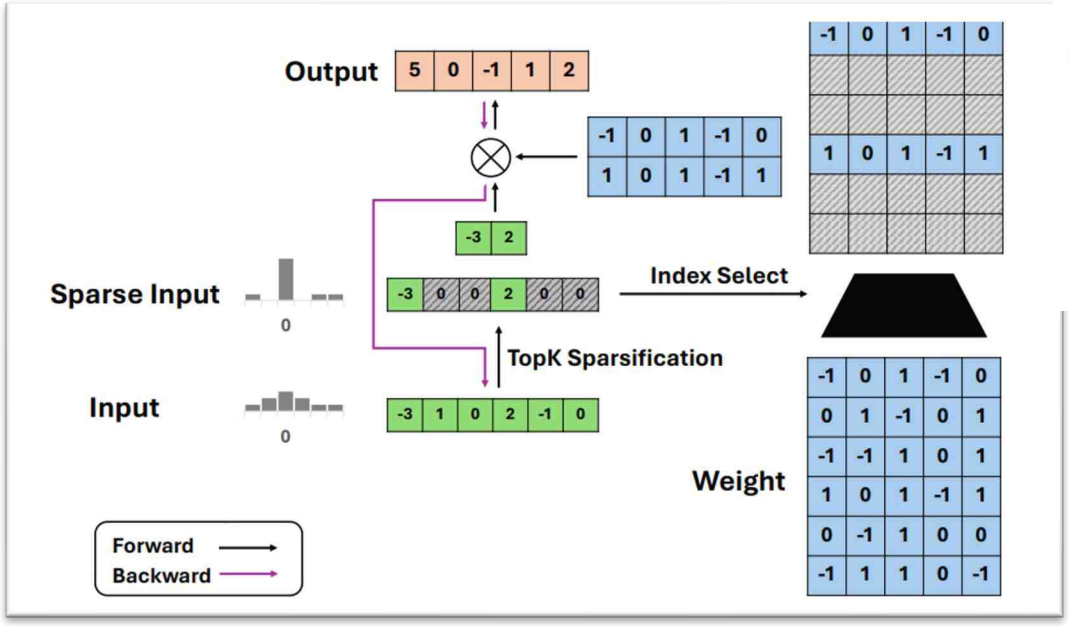
OPT	Sparsity	2.7B	6.7B	13B	30B	66B	175B
Dense	0%	12.47	10.86	10.13	9.56	9.34	8.35
Magnitude	50%	265.	969.	1.2e4	168.	4.2e3	4.3e4
SparseGPT	50%	**13.48**	**11.55**	**11.17**	**9.79**	**9.32**	**8.21**
SparseGPT	4:8	14.98	12.56	11.77	10.30	9.65	8.45
SparseGPT	2:4	17.18	14.20	12.96	10.90	10.09	8.74

50%的稀疏率下仍能保持较优的效果

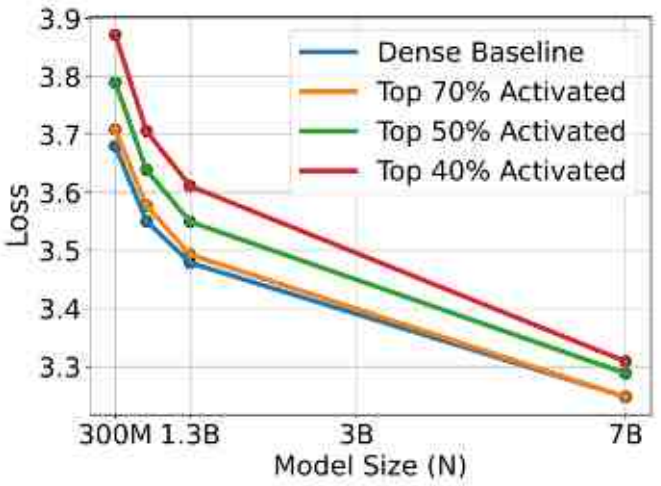
Frantar, Elias, and Dan Alistarh. "Sparsegpt: Massive language models can be accurately pruned in one-shot." ICML 2023.

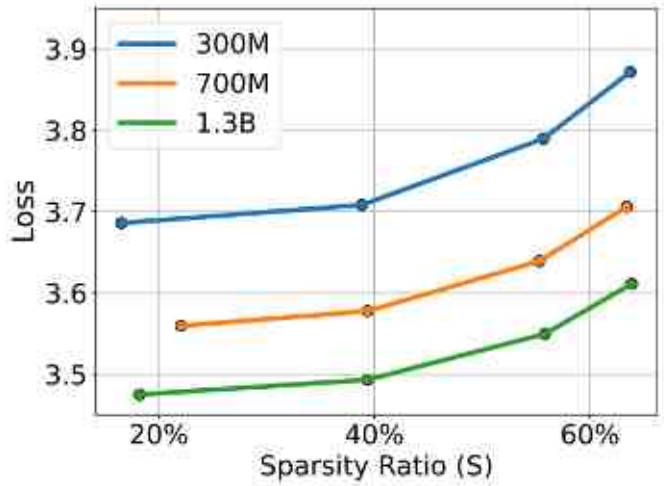
LLM稀疏化技术

- 表示稀疏化——非结构性稀疏



替代ReLU，用Top-K函数实现稀疏化





比密集模型更出色的推理最优缩放律

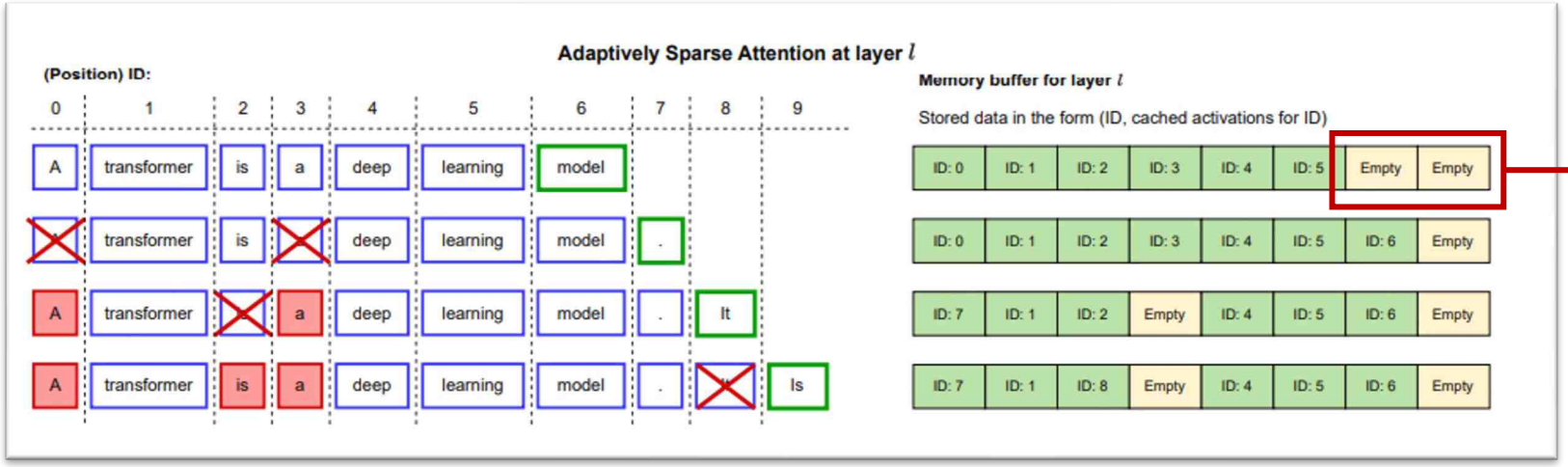
Models	Activated	ARC	HS	MMLU	WG	TQA	Avg.
Dense Baseline	7.0B	61.8	81.4	59.8	77.5	42.7	64.6
ReLUfication [MAM+23]	5.0B	57.2	78.8	54.7	74.7	38.8	60.8
dReLU Sparsification [SXZ+24]	5.4B	59.2	78.0	54.0	75.8	38.3	61.0
Q-Sparse (this work)	2.9B	59.0	79.0	55.6	74.0	41.0	61.7
	3.8B	60.5	80.7	58.0	75.9	43.5	63.7

在激活率50%左右时，能达到与源模型相近的结果

Wang H, Ma S, Wang R, et al. Q-Sparse: All Large Language Models can be Fully Sparsely-Activated[J].

LLM稀疏化技术

- KV表示稀疏化——非结构性稀疏



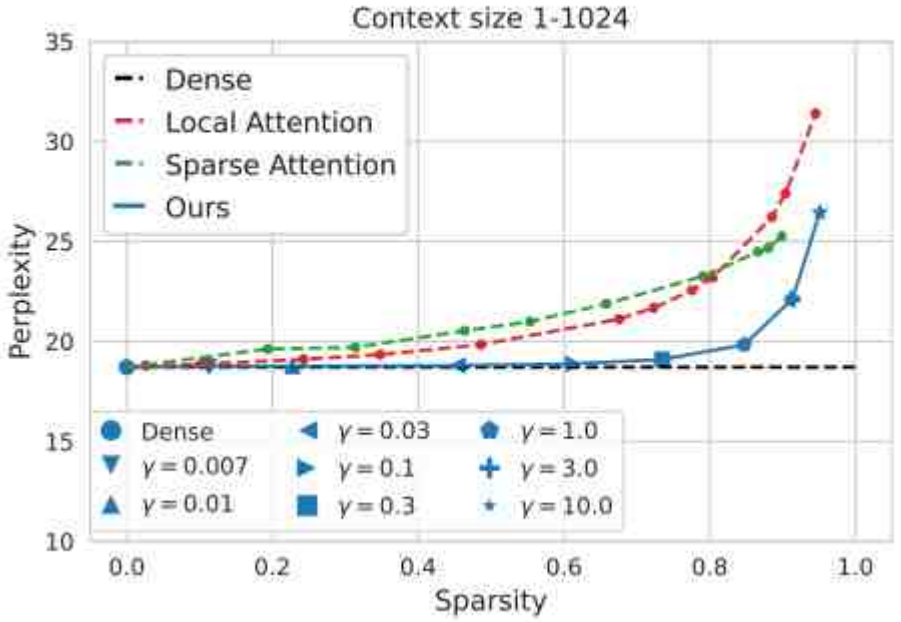
缓存区的激活状态被清除

如下公式所示，$I_{n,j}^{l}$为“重要性”累积值，一旦其等于0，此标记被丢弃，其效果是不可逆的，因为它将对所有后续标记以及随后的生成过程保持丢弃状态

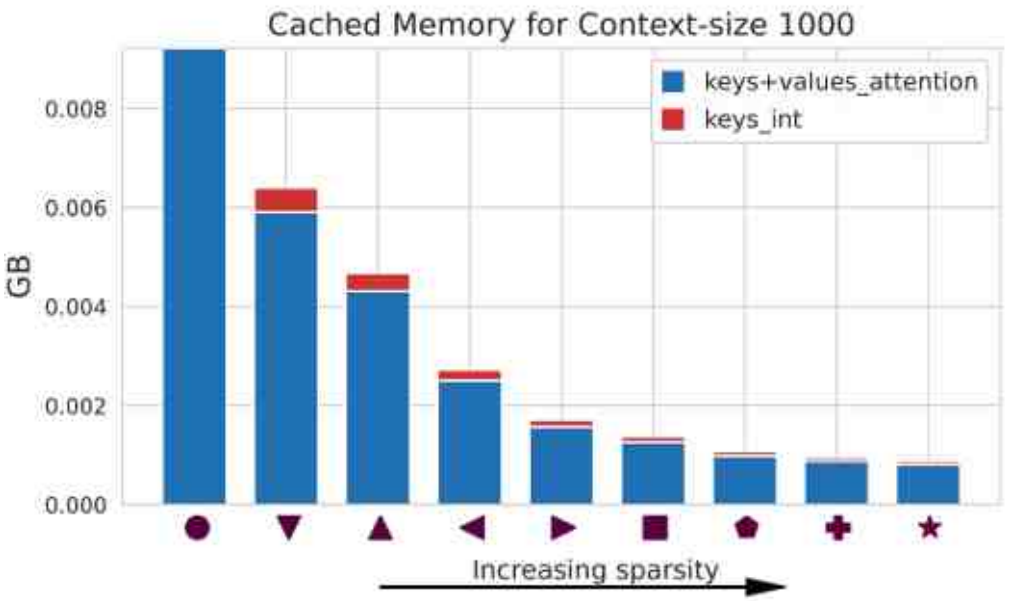
$$\mathbf{I}_{k,j}^{\ell} = \begin{cases} \prod_{n=j+1}^{k} \bar{\mathbf{I}}_{n,j}^{\ell} & \text{and } \bar{\mathbf{I}}_{n,j}^{\ell} = \sigma\left(\frac{(\mathbf{Q}_{\text{int}}^{\ell})_n^{\top}(\mathbf{K}_{\text{int}}^{\ell})_j}{\sqrt{r}} + \beta^{\ell}\right), \text{if } j < k \\ 1, \text{if } j = k, \\ 0, \text{if } j > k, \end{cases}$$

Anagnostidis S, Pavllo D, Biggio L, et al. Dynamic context pruning for efficient and interpretable autoregressive transformers[J]. NeurIPS 2023.

LLM稀疏化技术



随稀疏率变化，PPL呈现Scaling Law的趋势



随着稀疏率的增加Cache Memory逐渐降低

Anagnostidis S, Pavllo D, Biggio L, et al. Dynamic context pruning for efficient and interpretable autoregressive transformers[J]. NeurIPS 2023.

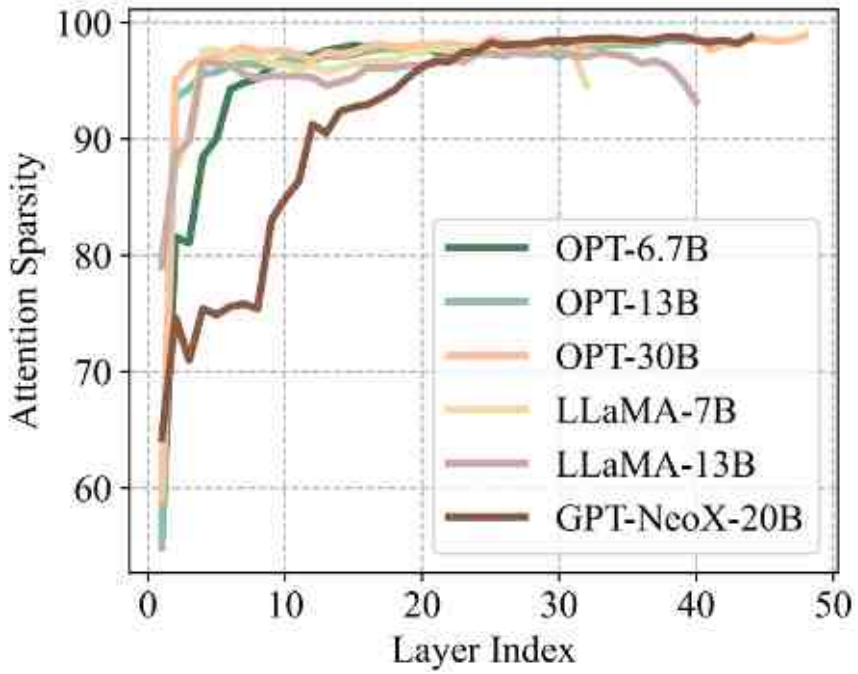
- **Some thoughts**

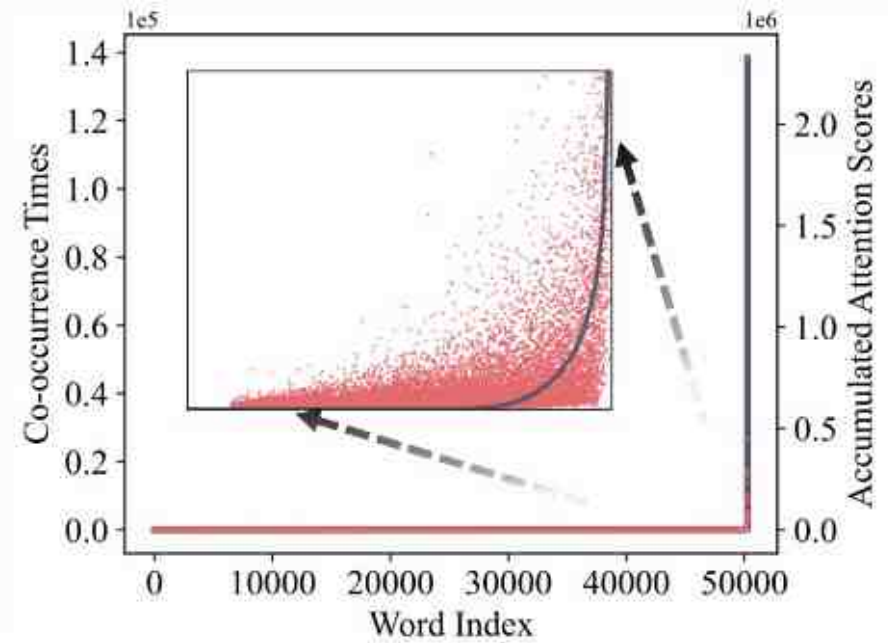
  - 以上方法倾向于选择值较小的元素进行稀疏化操作，进一步能否通过观察数据分布，通过数据分布的特点提出更合理的稀疏化策略也是一种有效的思路。

LLM稀疏化技术

• KV表示稀疏化——非结构性稀疏

➢ 观察：在计算注意力得分时，仅有**一小部分标记对结果的贡献最大**。（1）这些标记往往与其他标记有较强的关联性，它们频繁地与其他标记一起出现；（2）移除它们会导致显著的性能下降。





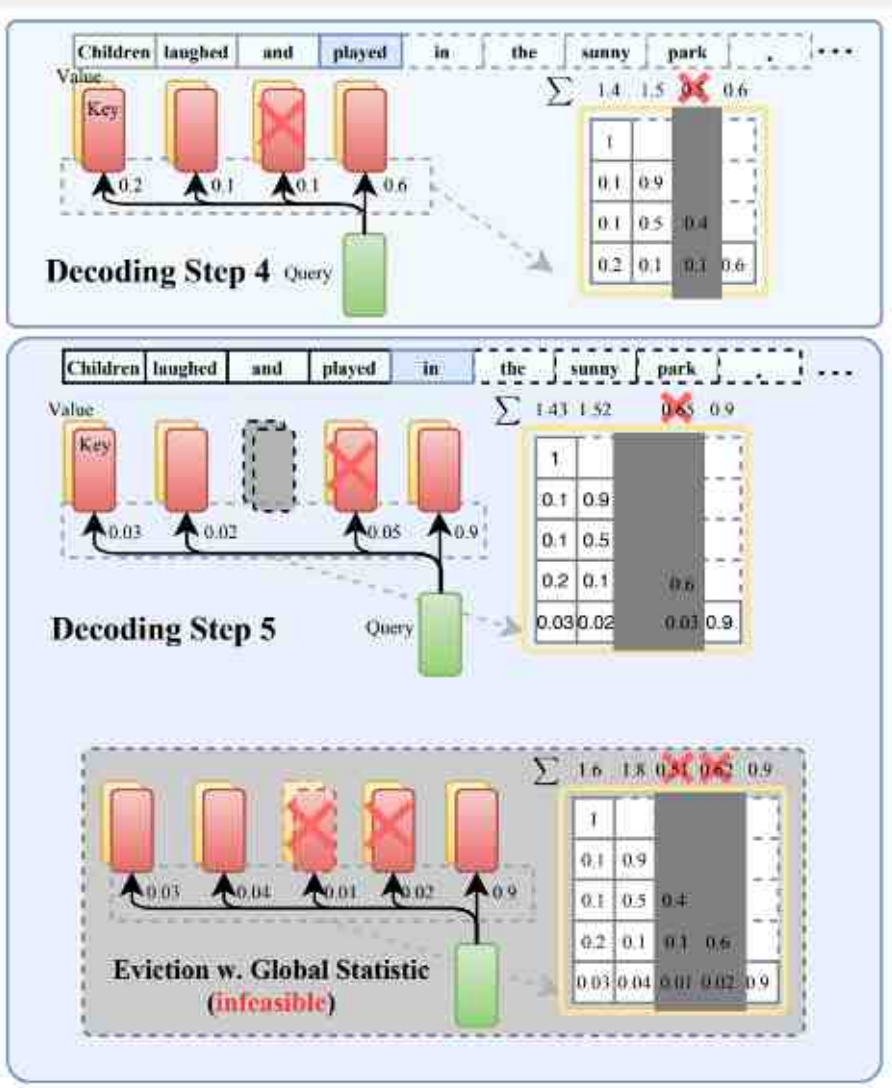
Zhang, Zhenyu, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. NeurIPS 2023.

LLM稀疏化技术

- **KV表示稀疏化——非结构性稀疏**

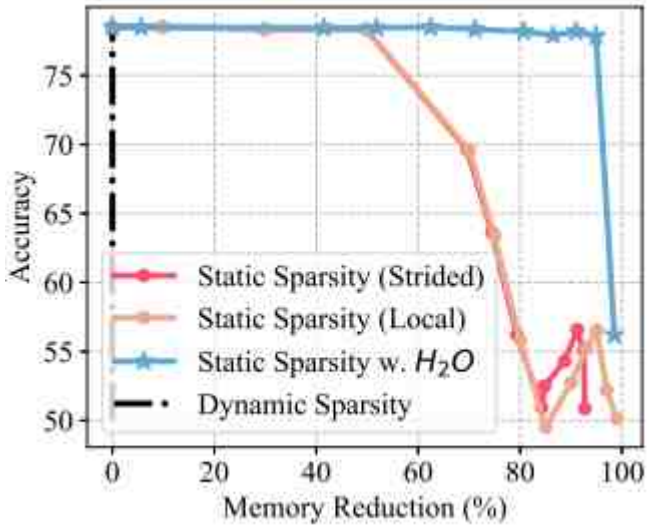
  - 提出了Heavy-Hitters Oracle（H2O），是一种动态子模优化算法，能够**动态平衡最近的标记和**Heavy-Hitters**标记**。
  - 具体而言，其提出了一种KV cache驱逐策略，每一步都增加最新的token及驱逐一个token。
  - 该方法被定义为了一个动态子模量的问题，经理论推导验证这种贪婪驱逐策略得到的集合理论上是接近最理想集合状态的：

$$f(S_{e_i}) \geq (1-\alpha)(1-1/e)\max_{|S|=k} f(S) - \beta$$



Zhang, Zhenyu, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. NeurIPS 2023.

LLM稀疏化技术



Seq. length	512+32		512+512		512+1024	
Model size	6.7B	30B	6.7B	30B	6.7B	30B
Accelerate	20.4 (2, G)	0.6 (8, C)	15.5 (1, G)	0.6 (8, C)	5.6 (16, C)	0.6 (8, C)
DeepSpeed	10.2 (16, C)	0.6 (4, C)	9.6 (16, C)	0.6 (4, C)	10.1 (16, C)	0.6 (4, C)
FlexGen	20.2 (2, G)	8.1 (144, C)	16.8 (1, G)	8.5 (80, C)	16.9 (1, G)	7.1 (48, C)
H_2O (20%)	35.1 (4, G)	12.7 (728, C)	51.7 (4, G)	18.83 (416, C)	52.1 (4, G)	13.82 (264, C)

- 在接近100%（95%）时的稀疏率下，H2O算法的效果才出现显著下降。
- 将三大主流推理系统DeepSpeed Zero-Inference、Hugging Face Accelerate和FlexGen的**吞吐量提升了最多29倍、29倍和3倍**。在相同的批量大小下，H2O可以将延迟减少最多1.9倍。

Zhang, Zhenyu, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. NeurIPS 2023.

- **Some thoughts**

  - 关于Heavy-Hitters的观测可能与矩阵的秩有关。
  - 能否建模更高阶的动态子模优化算法，设计KV cache驱逐策略来进一步提升模型效果。

LLM稀疏化技术

- **端侧稀疏化技术——非结构化稀疏**

为什么使用将模型参数存储在闪存中?

由于大模型的参数量巨大，**端侧的DRAM容量有限**，为了将大模型部署在端侧，只能将部分模型参数存储在闪存中。在模型推理时，如若需要使用相关参数，则从闪存中读取参数并使用。

参数存储在闪存中产生的问题

在模型推理时，**频繁地从闪存中读取对应的参数会花费大量时间，造成模型推理速度的下降。**

问题：在将参数存储在闪存的基础上，如何减少模型参数读取的时间?

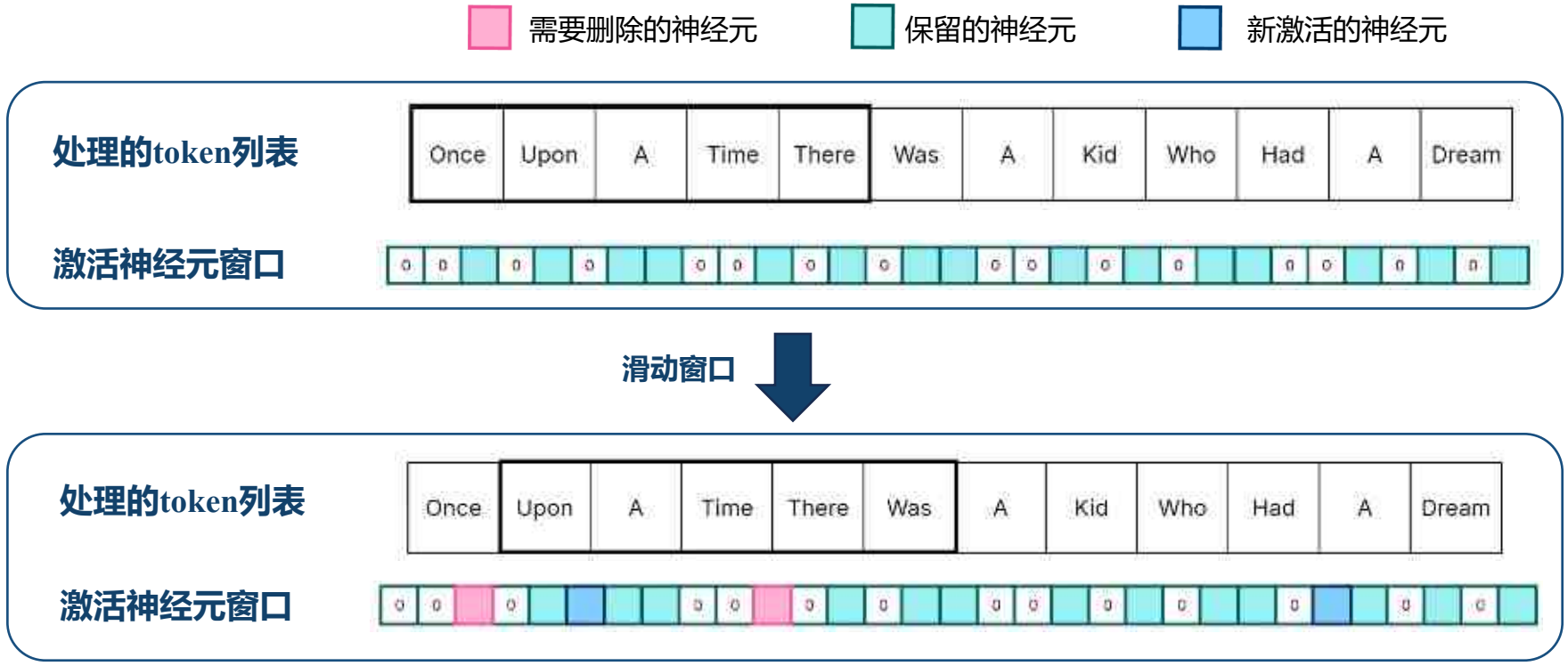
Alizadeh K, Mirzadeh I, Belenko D, et al. Llm in a flash: Efficient large language model inference with limited memory[J]. ACL 2024.

LLM稀疏化技术

- **端侧稀疏化技术——非结构化稀疏**

解决方案一：减少参数的重复读取，通过“窗口化”保留下次迭代时需要的参数

窗口化实现原理：窗口化技术通过设定一个“窗口”，将**推理过程中一部分神经元的激活状态保留在DRAM中**，而不是每次都从闪存中重新加载这些神经元的数据。显著减少了与闪存之间的数据传输。



Alizadeh K, Mirzadeh I, Belenko D, et al. Llm in a flash: Efficient large language model inference with limited memory[J]. ACL 2024.

LLM稀疏化技术

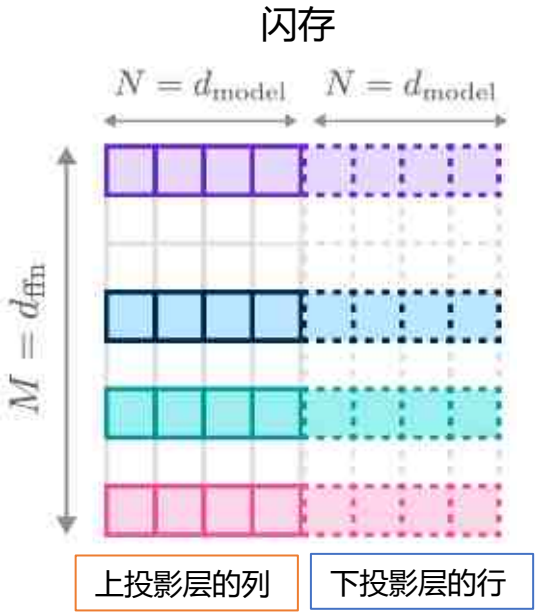
- **端侧稀疏化技术——非结构化稀疏**

解决方案二：采用“行列捆绑”技术，加快参数读取速度

参数读取特点：在读取数据量相同的情况下，读取连续存储在一起的参数的速度会远远快于读取分散存储的参数。

FFN神经元激活特点：在FFN中，第i个神经元的激活与上投影层的第i列和下投影层的第i行是相关的。当其激活时，则需要连续对上下投影层的i行i列进行读取。

行列捆绑技术：将FFN中上下投影层的i行i列捆绑存储在同一内存中，方便连续读取。



行列捆绑示意图

Alizadeh K, Mirzadeh I, Belenko D, et al. Llm in a flash: Efficient large language model inference with limited memory[J]. ACL 2024.

LLM稀疏化技术

Configuration				Performance Metrics			
Hybrid	Predictor	Windowing	Bundling	DRAM (GB)	Flash→ DRAM (GB)	Throughput (GB/s)	I/O Latency (ms)
✗	✗	✗	✗	0	13.4 GB	6.10 GB/s	2196 ms
✓	✗	✗	✗	6.7	6.7 GB	6.10 GB/s	1090 ms
✓	✓	✗	✗	4.8	0.9 GB	1.25 GB/s	738 ms
✓	✓	✓	✗	6.5	0.2 GB	1.25 GB/s	164 ms
✓	✓	✓	✓	6.5	0.2 GB	2.25 GB/s	87 ms

吞吐量提高一倍，将整体延迟降低一半

			Inference Latency (ms)			
Model	Method	Backend	I/O	Mem	Compute	Total
OPT 6.7B	Naive	CPU	2196	0	986	3182
OPT 6.7B	All	CPU	**105**	58	506	**669**
OPT 6.7B	Naive	Metal M1	2196	0	193	2389
OPT 6.7B	All	Metal M1	**92**	35	438	**565**
OPT 6.7B	Naive	Metal M2	2145	0	125	2270
OPT 6.7B	All	Metal M2	**26**	8	271	**305**
OPT 6.7B	Naive	GPU	2196	0	22	2218
OPT 6.7B	All	GPU	**30**	34	20	84
OPT 6.7B	Speculative	GPU	38.5	9.5	12	**60**
Falcon 7B	Naive	CPU	2295	0	800	3095
Falcon 7B	Hybrid	CPU	1147	0	800	1947
Falcon 7B	All	CPU	**161**	92	453	**706**
Persimmon 8B	Naive	CPU	2622	0	1184	3806
Persimmon 8B	Hybrid	CPU	1311	0	1184	2495
Persimmon 8B	All	CPU	**283**	98	660	**1041**
Phi-2 2.7B	Naive	CPU	885	0	402	1287
Phi-2 2.7B	Hybrid	CPU	309	0	402	711
Phi-2 2.7B	All	CPU	**211**	76	259	**546**
Llama 2 7B	Naive	CPU	2166	0	929	3095
Llama 2 7B	Hybrid	CPU	974	0	929	1903
Llama 2 7B	All	CPU	**279**	152	563	**994**

显著减少不同设置下的端到端延迟

Alizadeh K, Mirzadeh I, Belenko D, et al. Llm in a flash: Efficient large language model inference with limited memory[J]. ACL 2024.

LLM稀疏化技术

- **端侧稀疏化技术——非结构化稀疏**

智能手机中部署LLM的问题

智能手机的内存容量有限，模型参数存储在闪存中。而**单一命令队列**无法支持并发访问，因为智能手机功能较弱，异构硬件和存储设备带宽较低，这使得I/O活动成为移动设备上LLM推理的常见瓶颈。限制了LLM的推理速度。

问题：在智能手机基础上，提升LLM的推理速度?

Xue Z, Song Y, Mi Z, et al. PowerInfer-2: Fast Large Language Model Inference on a Smartphone[J]

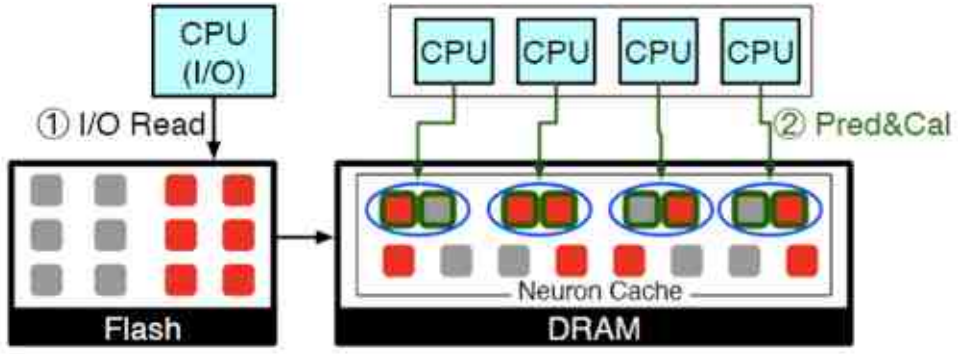
LLM稀疏化技术

- **端侧稀疏化技术——非结构化稀疏**

解决方案一：基于神经元粒度去存储模型权重，以智能手机的异构硬件环境，计算和I/O开销

神经元权重：
参考LLM in Flash，PowerInfer-2 抛弃了矩阵结构，进而采用神经元为单位存储模型权重。

神经元粒度的推理：
以神经元簇的粒度进行计算和I/O操作，神经元簇可以在计算过程中动态地由多个激活的神经元组成，神经元的数量由计算单元的计算能力决定。以此可以减少神经元权重的读取次数。



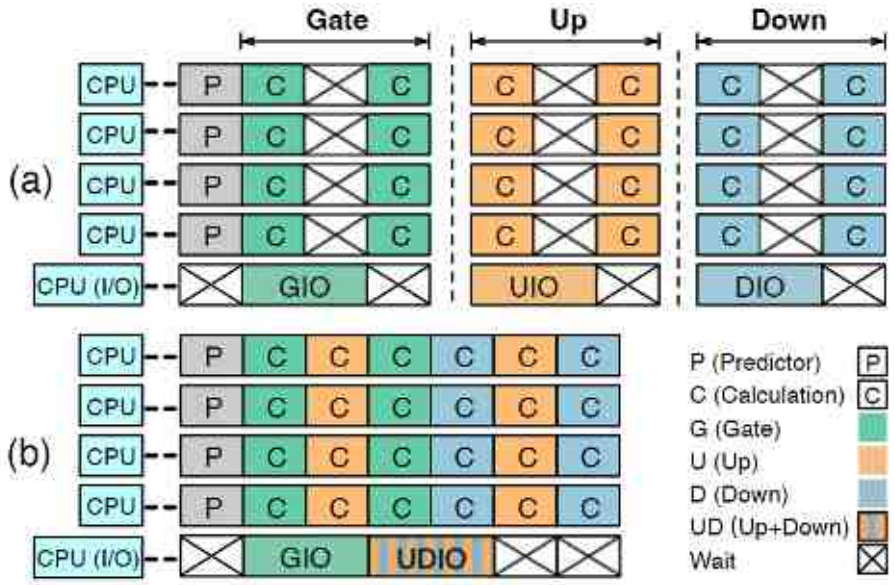
CPU动态使用闪存和DARM的神经元

Xue Z, Song Y, Mi Z, et al. PowerInfer-2: Fast Large Language Model Inference on a Smartphone[J]

解决方案二：将计算和 I/O 操作同步进行，隐藏I/O操作带来的延迟

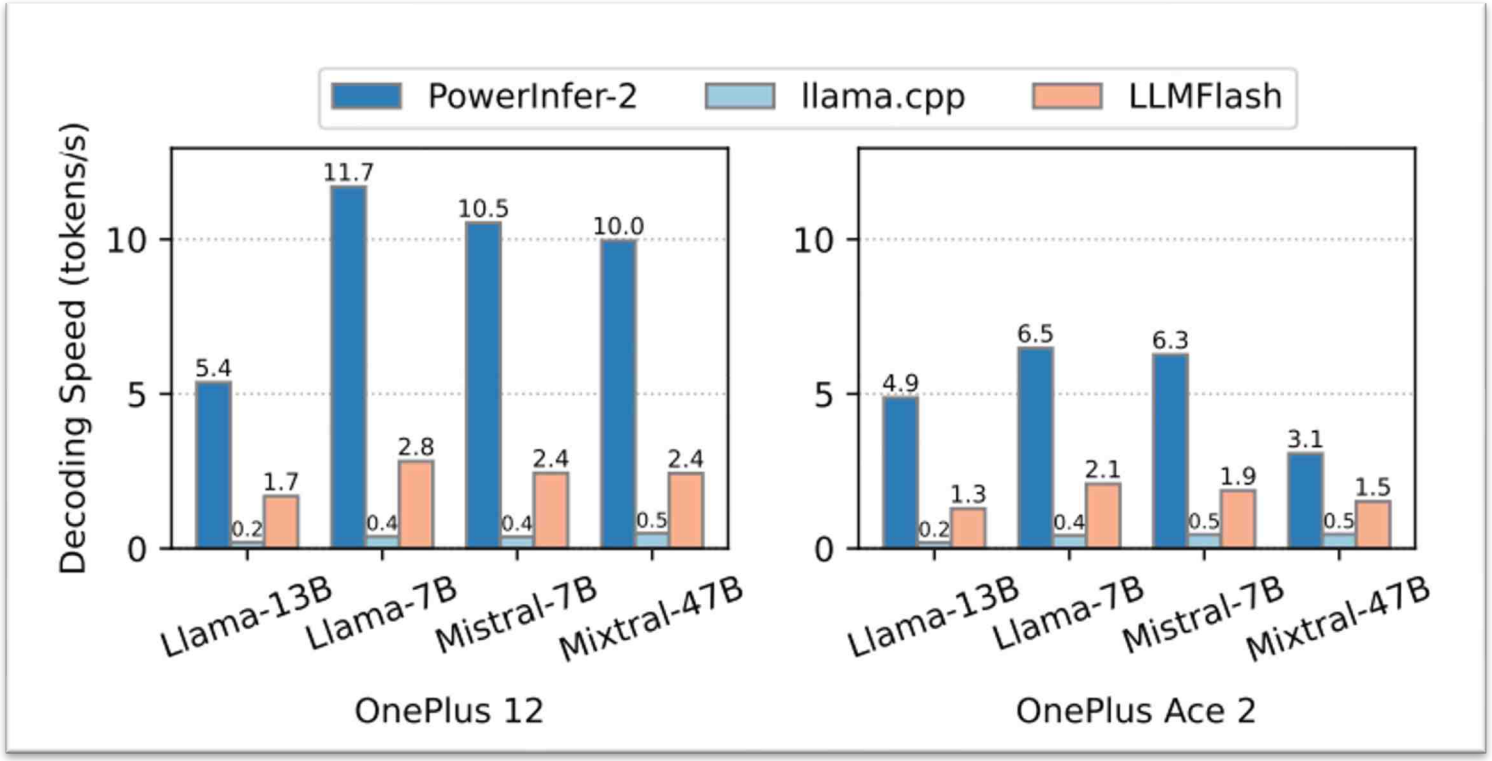
神经元集群级的I/O流水线：
通一个CPU负责判断神经元的激活情况（Pred），然后把信息传递给计算线程。计算线程中的一些CPU负责从存储中读取 Gate 矩阵的权重（GIO），另一些CPU同时计算 Gate 矩阵和输入向量的乘积（GC）。还有CPU负责读取 Up 和 Down 矩阵的权重（UDIO），并计算它们与输入向量的乘积（UDC）。

效果：PowerInfer - 2 中的这些计算和 I/O 操作也可以同时进行，大大提高了效率，减少了等待时间，从而实现了更快的 LLM 推理速度。



普通I/O管道(a)和PowerInfer-2(b)对比图

LLM稀疏化技术

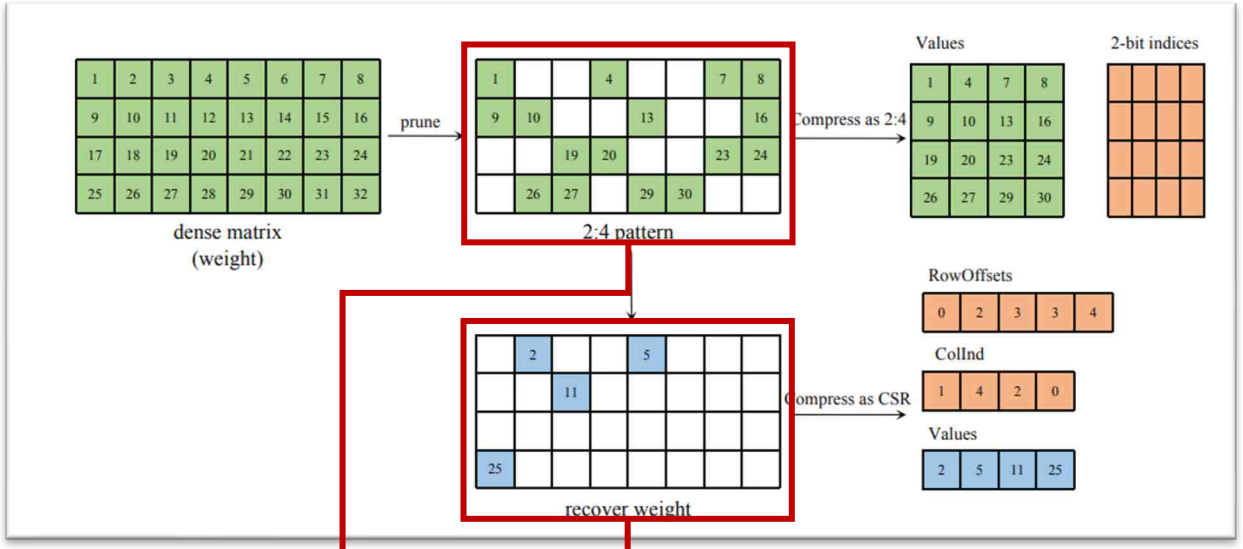


- 相较于其他方案，明显提高了解码速度
- 能够在移动端进行Mixtral-47B的推理

Xue Z, Song Y, Mi Z, et al. PowerInfer-2: Fast Large Language Model Inference on a Smartphone. arXiv:2406.06282

LLM稀疏化技术

- **LLM剪枝——融合非结构化与结构化优势的2:4模式**



Size	Method	HellaSwag	PIQA	WinoGrande	OpenBookQA	RTE	Average
7B	Dense	75.9	79.0	69.1	44.0	63.2	66.2
	SparseGPT(50%)	70.0	77.3	69.7	42.4	53.4	62.6
	Wanda(50%)	70.7	76.7	67.0	42.0	54.2	62.1
	SparseGPT(2:4)	56.7	70.8	64.5	35.4	**55.2**	56.5
	Wanda(2:4)	54.5	70.9	61.9	37.4	53.4	55.6
	WRP(α=0.25)	**63.1**	**74.2**	**66.1**	**38.8**	54.2	**59.3**
13B	Dense	79.4	80.6	72.2	45.4	65.0	68.5
	SparseGPT(50%)	74.4	78.1	72.5	43.2	62.8	66.2
	Wanda(50%)	76.1	78.7	70.9	44.4	60.6	66.1
	SparseGPT(2:4)	62.7	73.8	**69.6**	36.6	58.8	60.3
	Wanda(2:4)	62.1	74.0	65.7	35.6	57.0	58.9
	WRP(α=0.25)	**69.9**	**77.3**	69.1	**41.2**	**61.0**	**63.7**

- 为了减少索引成本，**NVIDIA提出了2:4模式剪枝**，在**每组四个连续的权重中，只保留两个非零值**，并且以固定的方式进行剪枝。这种结构化稀疏能够更好地**适应硬件加速器的计算架构**。
- 提出了权重恢复剪枝方法。通过**恢复一小部分关键权重，提高模型性能，同时保持压缩的效率**。

Tan Z, Zhang X, Wei Z. WRP: Weight Recover Prune for Structured Sparsity[C]. ACL 2024.

➢ Some thoughts

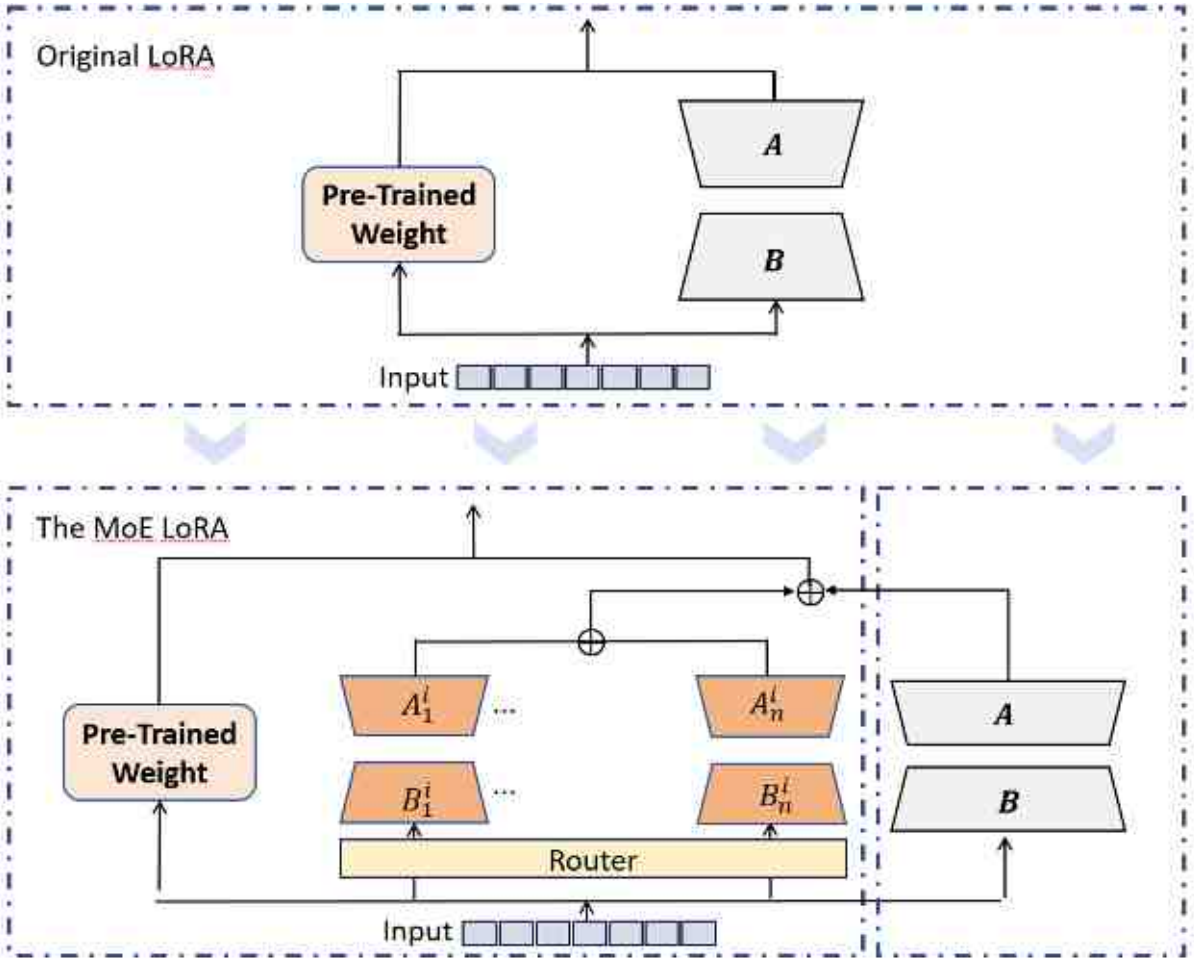
- 2:4模式是一种结合结构化稀疏与非结构化稀疏的方法，使其平衡模型效果的同时适应硬件计算架构；
- 作为一种有益的思想启发，能否迁移到如量化等技术上，解决类似的问题如结构式量化与非结构式量化的平衡问题？

➢ 稀疏化+低秩近似的思路

- 基于以上研究，我们发现无论是“控制器”的低秩参数块，还是利用LoRA技术微调稀疏模块，都是稀疏化+低秩近似结合的线索；
- 轻量化的思路：利用低秩近似来补充稀疏化在参数量和效果方面的不足，利用稀疏化来补充低秩近似在计算成本方面的不足；
- 近期的LoRA+MoE技术，是一种显式的稀疏+低秩近似的结合样例。

稀疏化技术

- **LoRA+MoE——微调稀疏化**



挑战

如何提升大模型参数微调的效率也是近两年业界关注的重要问题。

- 以轻量化微调见长的LoRA，其相关参数矩阵仍然是**稠密且全部使用的。**
- 受启发与全量参数微调与MoE相结合的思想，提出了**极端提升参数效率**的MoLoRA算法。

Zadouri T, Üstün A, Ahmadian A, et al. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning[J].

稀疏化技术

Zero-shot Results at 3B Scale

	Model	% Params.	ANLI	CB	RTE	WSC	WIC	Copa	WNG	HS	Average
Full-FT	T0-3B (Sanh et al., 2022)	100%	33.46	50.0	64.08	64.42	50.39	74.92	50.51	27.51	51.91
	T0-3B (our replication)	100%	41.08	80.36	76.17	53.37	53.92	88.94	57.46	29.19	60.06
PEFT	$(IA)^3$	0.018%	34.08	50.0	66.43	56.25	55.41	79.08	52.09	29.91	52.90
	LORA	0.3%	37.5	75.57	73.53	61.02	51.25	83.6	54.33	25.32	57.51
Our Method	MoV-10	0.32%	38.92	75.0	78.88	62.5	52.19	85.77	55.96	30.24	59.93
	MoV-30	0.68%	38.7	78.57	80.87	63.46	51.1	87.25	56.27	28.63	60.61
	MoV-60	1.22%	38.83	76.79	74.55	60.1	52.66	89.79	55.49	30.47	59.83
	MoLORA-10	3.18%	38.5	78.57	78.16	63.46	50.86	86.5	55.41	26.72	59.77
	MoLORA-15	4.69%	40.0	80.36	80.51	62.98	50.86	89.0	55.33	27.3	60.79

相较于全量微调，使用尽少量的参数，达到了相近的实验效果

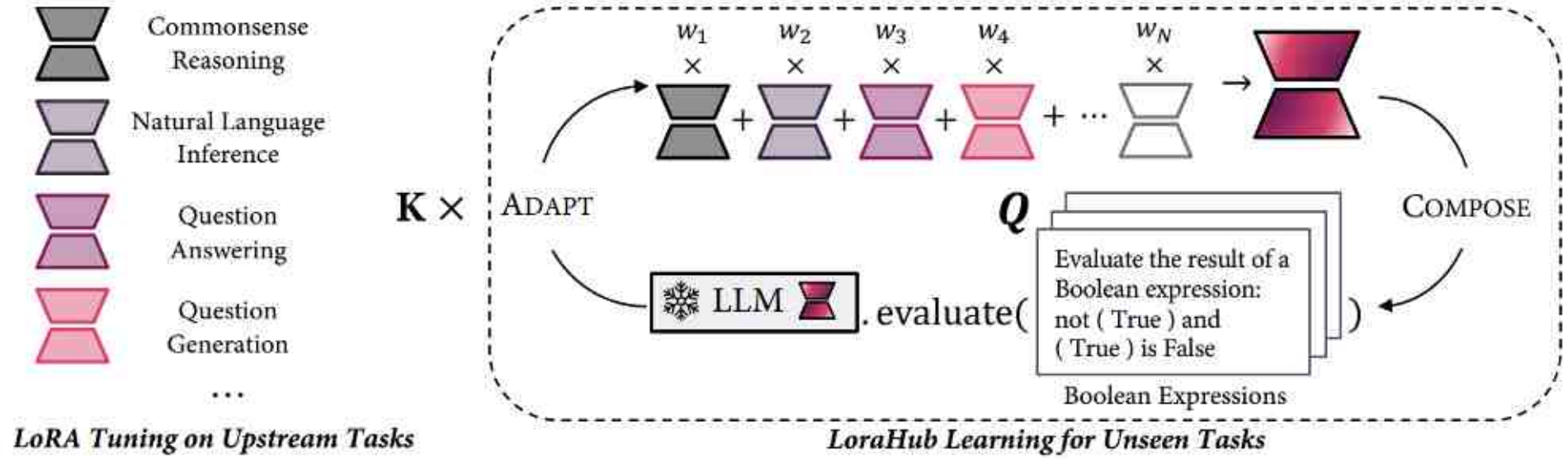
Zadouri T, Üstün A, Ahmadian A, et al. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning[J].

稀疏化技术

- **LoRA+MoE——微调稀疏化**

现有的 LoRA 模块被集成到一个统一的模块中，采用一组系数。

应用无梯度算法来优化 w，根据来自未见任务的几个示例进行评估。执行 K 次迭代后，生成一个高度适应的组合 LoRA 模块，可将其与 LLM 结合以执行预期任务。



Chengsong Huang, et al. LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition[J]. COLM 2024.

稀疏化技术

Task	Zero	ICL_{avg}	$IA3_{avg}$	$LoRA_{avg}$	FFT_{avg}	$LoraHub_{avg}$
Boolean Expressions	54.0	59.6	56.2	56.0	62.2	55.5
Causal Judgement	57.5	59.4	60.2	55.6	57.5	54.3
Date Understanding	15.3	20.4	20.0	35.8	59.3	32.9
Disambiguation	0.0	69.1	0.0	68.0	68.2	45.2
Dyck Languages	1.3	0.9	4.2	22.2	19.5	1.0
Formal Fallacies	51.3	55.3	51.5	53.6	54.0	52.8
Geometric Shapes	6.7	19.6	14.7	24	31.1	7.4
Hyperbaton	6.7	71.8	49.3	55.3	77.3	62.8
Logical Deduction§ (five objects)	21.3	39.1	32.7	40.0	42.2	36.1
Logical Deduction§ (seven objects)	12.7	40.7	33.8	37.3	44.9	36.8
Logical Deduction§ (three objects)	0.0	51.6	8.5	53.6	52.9	45.7

在多种任务上进行了评估，综合取得了最好的实验效果

Chengsong Huang, et al. LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition[J]. COLM 2024.

➢ Some thoughts

- （改进思路）通常情况下，高阶的低秩近似参数可以拆分成多个小参数模块，相较于原始LoRA的MF分解形式，是否可以与稀疏机制更灵活的结合。
- （改进思路）在大模型上实行稀疏化技术（如MoE和激活稀疏等）+低秩近似技术，实现效果、计算成本与参数量三者的平衡。

目录

展望与期待

天津大学 Tianjin University

展望

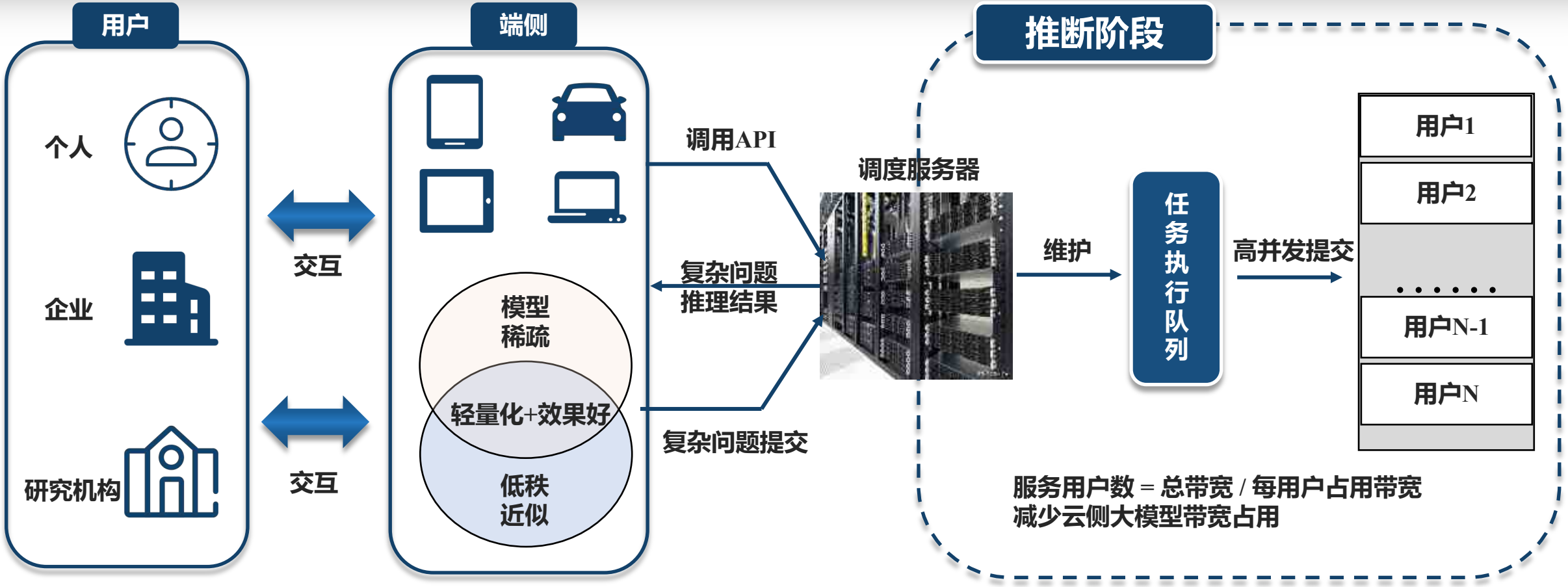
- **如何将稀疏化技术与硬件存储特性更好结合?**
  - 寻求低效高存储与高效低存储组件的协同计算，寻找动态平衡；
- **如何在较高稀疏率的情况下，保持模型的效果?**
  - 在高稀疏率的状态下，结合其他的轻量化算法，如量化与低秩近似等，保证模型效果。
- **如何在端侧达到存储、速度与效果的平衡?**
  - 在量化技术打底的情况下，实行稀疏+低秩的融合，并结合密集存储结构及硬件计算的特点，实现端侧轻量化目标。

期待

- **稀疏率与效果的平衡**
  - 在Scaling law的指导下，快速配置效果与成本平衡的大模型推理体系；
- **实时在线微调**
  - 结合稀疏化技术的微调手段，实现快速在线微调，助益大模型进化；
- **端云高效推理体系建成**
  - 端侧稀疏避免显存与带宽资源受限的问题，云侧稀疏助益吞吐量提升。端云协同助力大模型广泛布局。

展望——经典端云协同架构

云侧大模型提供高并发服务，端侧大模型在极低资源条件下进行高效准确的响应，实现端云协同轻量化框架。



天津大学

Tianjin University

请批评指正