稀疏神经网络

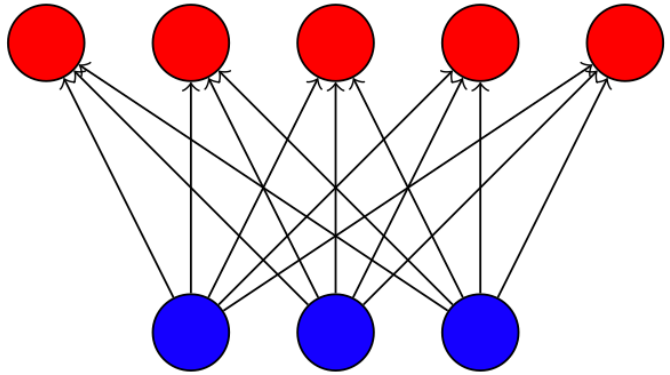
清华大学计算机系 陈键飞

《大模型计算》课程团队

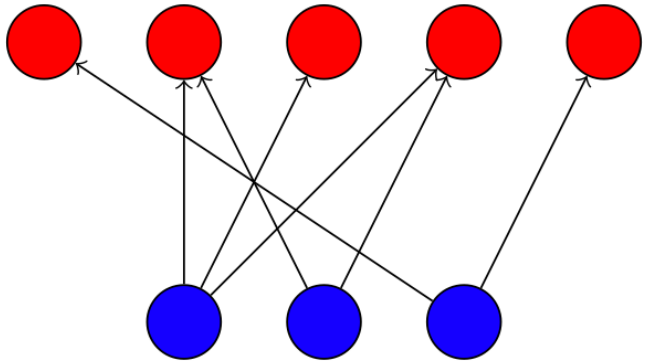
稀疏神经网络

- 考虑线性层 $\mathbf{Y} = \mathbf{X}\mathbf{W}^T$
- 直觉：神经元未必需要看到所有输入
- 人脑神经元：860亿
- 度数：1000～10000

- 结果：
- 网络更小
- 计算所需的flops更少

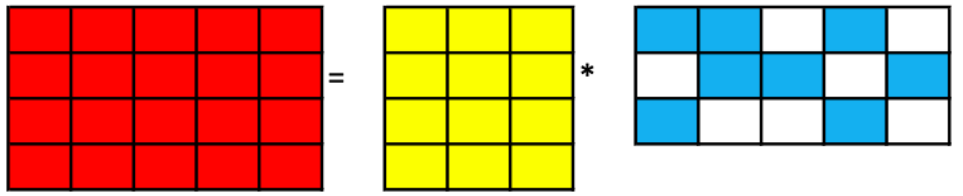


稠密网络



稀疏网络

$$\mathbf{Y}_{dense} = \mathbf{X}_{dense}\mathbf{W}_{sparse}^{\mathrm{T}}$$



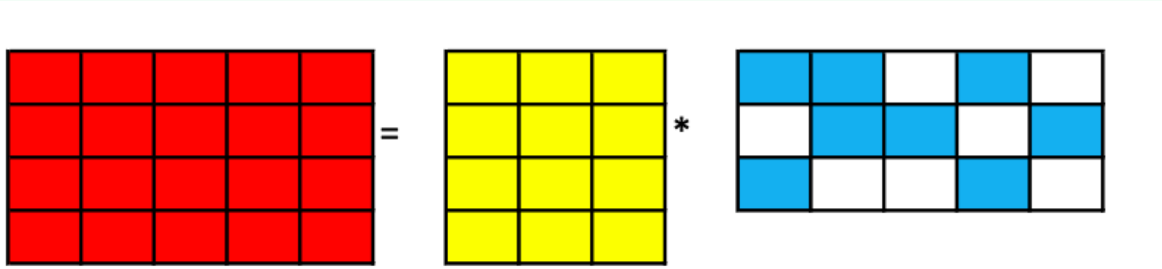
稀疏矩阵乘法

- 稀疏率（sparsity / sparsity ratio）
- 零的占比
- 50%稀疏率 = 2倍加速
- 90%稀疏率 = 10倍加速

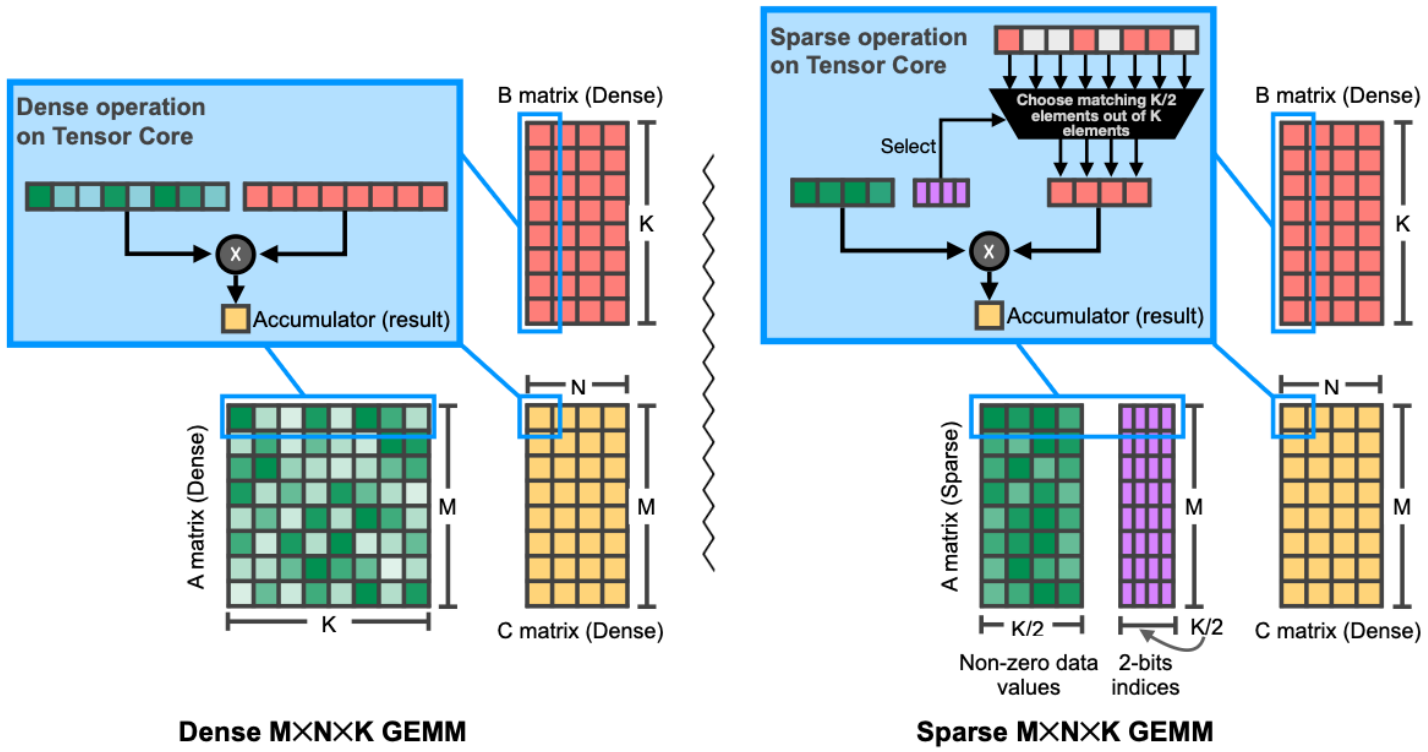
- 零的模式
- 非结构化稀疏
- 半结构化稀疏（2:4稀疏）
- 结构化稀疏（块稀疏/整行整列稀疏）

- SpMM

$$C_{dense} = A_{dense} B_{sparse}$$

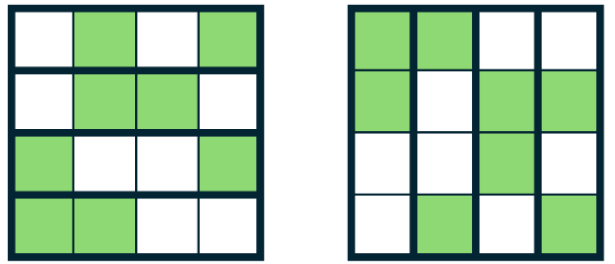


2:4稀疏



Form Factor	H100 SXM
FP64	30 teraFLOPS
FP64 Tensor Core	60 teraFLOPS
FP32	60 teraFLOPS
TF32 Tensor Core	1,000 teraFLOPS* | 500 teraFLOPS
BFLOAT16 Tensor Core	2,000 teraFLOPS* | 1,000 teraFLOPS
FP16 Tensor Core	2,000 teraFLOPS* | 1,000 teraFLOPS
FP8 Tensor Core	4,000 teraFLOPS* | ,000 teraFLOPS

Row-wise 2:4 Col-wise 2:4



PyTorch原生支持: torch.sparse

神经网络剪枝

- 如何把稠密神经网络变成稀疏神经网络，使得精度损失尽量小?
- 思路1：让稠密权重W和稀疏权重$S(W)$尽量接近

$$\min_{S(W)} \|W - S(W)\|^2$$

- 解析解：贪心地保留幅值最大的元素

$$S(W)_i = W_i \times M_i$$

$$M_i = \mathbb{I}(W_i \ is \ topk) \in \{0,1\} \longrightarrow \text{掩码mask}$$

- 免训练剪枝算法，基于幅值的剪枝（magnitude-based pruning）
- 问题：权重最小不代表总体损失最小

幅值剪枝+微调

❖ 思路2：基于一次性计算得到的掩码，微调保留下的权重

$$\min_{W} \mathcal{L}(W \circ M)$$

掩码M为根据W的幅值一次性计算得到

❖ “一次性”剪枝（one-shot pruning / single-shot pruning）

❖ 直观理解：在部分突触受到损伤时，其他突触会进行“代偿”

❖ 问题：初始选定的M不一定是最优的？

❖ 例如：两个突触权重均很高，但功能重合

❖ 更理想的方式是同时学习W和M

训练掩码：通过正则化引入稀疏性

- Lasso回归
- 问题：找到最少的线性因子，解释预测结果

$$\min_{\beta} \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta^T x_i)^2 + \lambda \|\beta\|_1$$

- 完全是线性回归，只是增加了正则化项$\|\beta\|_1 = \sum_{i=1}^{D} |\beta_i|$
- 为什么这样就能稀疏?
- 考虑梯度：$\mathcal{L}'_{\beta_i} = f(\beta)'_{\beta_i} + \lambda$
- 梯度下降：每维固定减λ
- 作用小的维度无力跑赢正则化

作为对比，L2正则化的梯度是：

- 考虑梯度：$\mathcal{L}'_{\beta_i} = f(\beta)'_{\beta_i} + \lambda\beta_i$
- 梯度下降：小的减得少，大的减得多

训练稀疏神经网络-L1正则化

- 分类损失$f(W)$
- L1正则化项：$\lambda \|W\|_1$
- 问题：
- 比较适合非结构化稀疏，半结构化稀疏/结构化稀疏需要额外的trick
- 稀疏度为动态决定，不容易控制
- 正则化项与分类损失存在竞争
- 正则化与Adam联用时需要特别的处理

- 但都可以解决，解决以后L1正则化事实上能取得SOTA的性能

结果：神经网络剪枝

❖ 训练40B token

TABLE IV: Few-shot Performance on More Challenging Tasks for LLaMA models.

Model	Weight Pattern	MMLU	MATH	TriviaQA	NQ	BoolQ	CSQA	SciQ	Mean
LLaMA2-7B	Dense	45.74	5.56	**64.15**	**26.34**	78.90	56.01	**97.00**	53.39
	Sparse	**52.34**	**7.64**	62.32	25.90	**79.94**	**64.62**	**97.00**	**55.68**
LLaMA2-13B	Dense	55.13	6.78	**70.45**	**30.69**	83.21	67.57	**97.50**	58.76
	Sparse	**56.07**	**8.12**	67.65	28.83	**84.31**	**71.42**	97.30	**59.10**
LLaMA3-8B	Dense	**65.43**	13.42	**71.69**	**29.39**	81.93	**73.62**	97.60	**61.87**
	Sparse	62.15	**13.44**	65.84	27.53	**82.66**	73.38	**97.80**	60.40

结果：神经网络剪枝

Methods	Tokens Trained	HellaS.	RACE	PIQA	WinoG.	ARC-e	ARC-c	OBQA	Average	Wikitext PPL
LLaMA2-7B	2T	57.03	44.11	78.07	69.38	75.38	42.92	33.20	57.16	5.12
CAST†	40B	56.13	40.86	77.58	69.53	77.78	47.18	33.60	57.52	5.21
Wanda	✗	41.05	35.02	70.78	62.67	61.99	27.56	22.80	45.98	11.29
MaskLLM	2B	50.91	**40.77**	74.92	64.48	69.57	36.00	28.00	52.09	6.72
Naive Retraining	10B	53.90	38.28	76.61	68.27	75.21	41.21	29.60	54.73	5.78
SR-STE	10B	54.02	39.02	76.88	**68.35**	75.58	41.46	29.60	54.99	5.74
CAST	7.5B	**54.50**	40.48	**77.09**	68.27	**76.52**	**43.68**	**30.80**	**55.91**	**5.58**
LLaMA2-13B	2T	60.15	44.59	79.27	72.45	78.93	47.18	34.60	59.60	4.57
CAST†	15B	58.01	41.24	77.42	72.45	79.50	49.06	34.80	58.93	4.71
Wanda	✗	46.96	38.09	74.05	66.69	68.64	34.81	25.00	50.61	8.47
MaskLLM	2B	55.09	**41.24**	77.69	67.80	73.15	40.44	30.00	55.06	5.85
Naive Retraining	10B	58.08	39.43	78.07	71.35	77.23	47.77	33.60	57.93	5.08
SR-STE	10B	**58.27**	40.38	**78.32**	71.11	78.29	**47.79**	34.80	58.42	5.04
CAST	7.5B	58.14	40.67	78.13	**72.30**	**78.83**	**47.79**	**35.40**	**58.75**	**4.91**
LLaMA3-8B	15T	60.10	40.00	79.43	73.56	80.26	50.00	34.80	59.74	5.76
CAST†	40B	57.90	40.77	79.27	71.67	81.40	50.43	34.60	59.43	6.33
Wanda	✗	37.09	32.63	67.41	59.75	56.48	25.85	18.00	42.46	22.97
MaskLLM	2B	53.89	37.79	77.86	67.88	71.88	38.73	30.00	54.00	8.50
Naive Retraining	10B	56.07	39.43	78.56	70.48	78.78	46.41	32.20	57.42	7.26
SR-STE	10B	56.18	**39.81**	78.89	70.64	78.57	47.18	31.80	57.58	7.22
CAST	7.5B	**56.41**	**39.81**	**79.05**	**71.74**	**79.54**	**48.89**	**33.20**	**58.38**	**6.85**

稀疏训练

- 利用L1正则化，在训练的过程中网络未必稀疏
- 在训练全过程中网络均稀疏的训练方法，称为稀疏训练（sparse training）

$$\min_{W} \mathcal{L}(\widetilde{W}), \qquad \widetilde{W} = W \circ M$$

- 事实上，掩码也可以动态更新
- 例如说，动态地定义$M(W)$为W的topk。每次前向传播时：
  - 根据W计算M
  - 计算$\widetilde{W}$
  - 计算网络损失（只利用稀疏的$\widetilde{W}$）
- 这样，当某维权重增大时，就会从不保留变为保留
- 稀疏度控制比较容易。只需调整topk的k

稀疏训练：案例

$$\min_{W} \mathcal{L}(\widetilde{W}), \qquad \widetilde{W} = W \circ M$$

- 非结构化剪枝：把M设置成整个tensor的topk
- 半结构化剪枝：把M设置成每个4元组的top 2
- 结构化剪枝：把M设置成所有neuron的topk neuron

稀疏训练

- 问题是：如何计算梯度?

$$\min_W \mathcal{L}(\widetilde{W}), \qquad \widetilde{W} = W \circ M(W)$$

- 方案1:直接通过autograd

$$\mathrm{d}W = \mathrm{d}\widetilde{W} \circ M(W)$$

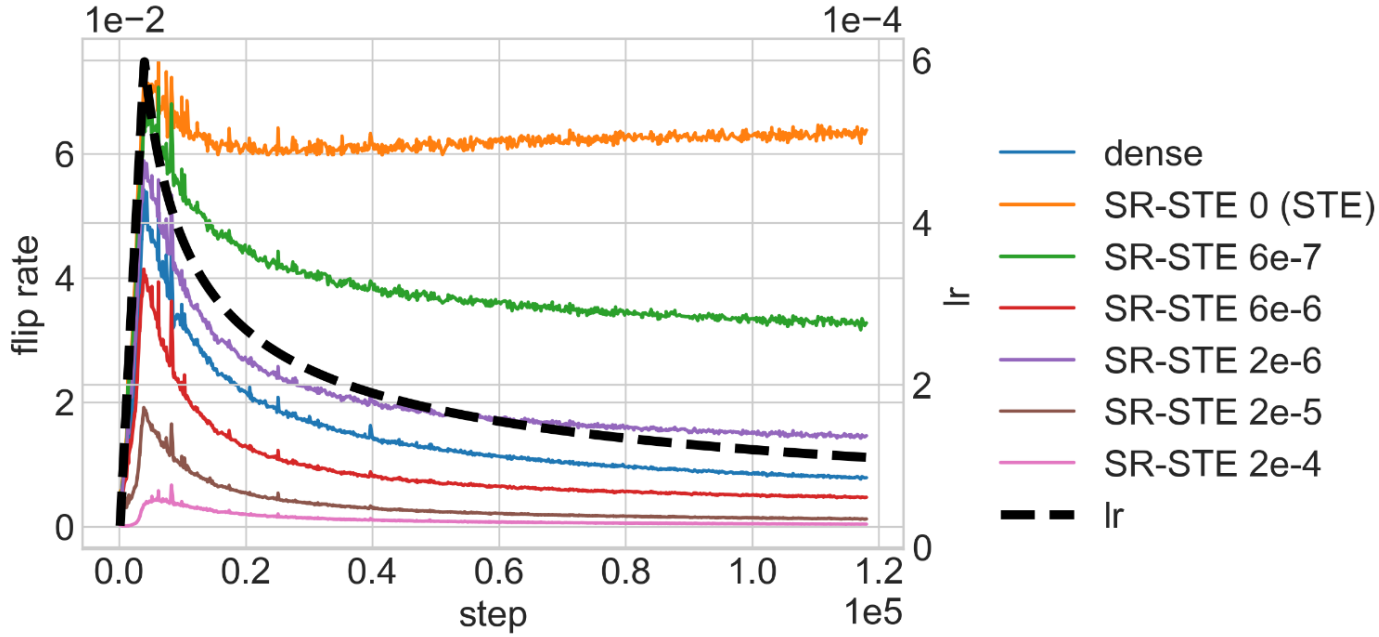
- 缺点：如果某维未被选中，就不会收到梯度
- 方案2:利用直通估计器

$$\mathrm{d}W = \mathrm{d}\widetilde{W} \circ \left(M(W) + W \circ M'(W)\right) \approx \mathrm{d}\widetilde{W} \circ (M(W) + W)$$

- 例如说，某维初始W=0.001，未进入topk，M(W)=0。而梯度$-\mathrm{d}\widetilde{W} = 10$
- 则该维仍会收到10*0.001=0.01的更新，积累足够多时仍可能成为topk

稀疏训练

- 问题：震荡又来了
- 每次迭代，竟有6%的掩码反转



- 解决方案：进行正则化
- SR-STE

$$+\lambda\left\|\left(1-M(W)\right)\circ W\right\|^2$$

- 惩罚M(W)=0的维度的权重
- “让逝者安息”

- 有效抑制震荡，但也同时抑制了可能性

彩票假说（Lottery Ticket Hypothesis）

❖ **一个随机初始化的稠密前馈网络包含一个子网络（“中奖彩票”），当该子网络被独立训练时，在相似的迭代次数内，可以达到与原始网络相当甚至更优的准确率。**

❖ **甚至有可能存在一些子网络，无需训练就能达到很高准确率?**

❖ **剪枝即训练? “突触修剪”**

突触修剪包括轴突和树突的完全退化和消失，是在包括人类在内的许多哺乳动物的幼年期至进入青春期之间发生的突触消除过程。修剪从出生时开始，一直持续到25岁左右。传统上认为，突触修剪在性成熟时完成，但这一观点受到核磁共振成像研究的挑战。婴儿大脑的尺寸到成年时将增长至原来的5倍，最终达到约860（ ± 80）亿个神经元。导致这一增长的因素有两个：神经元之间的突触连接的增长，以及神经纤维的髓鞘形成。不过，神经元的总数保持不变。修剪受环境因素影响，被普巴认为表征了学习。青春期之后，突触连接的体积由于突触修剪再次减少。

第一阶段：突触的过量产生（Synaptogenesis）

- 时间窗口：　这个阶段从孕期最后三个月开始，一直持续到童年早期（大约5-7岁左右达到峰值）。
- 过程：　在这个阶段，大脑神经元之间会以惊人的速度建立新的连接，称为“突触”。
- 在生命的最初几年，大脑每秒可以形成超过100万个新的神经连接。
- 到2岁时，幼儿大脑中的突触数量甚至能超过成年人大脑。
- 目的：　这种“过量生产”是为了让大脑为任何可能的环境和文化做好准备。
- 它赋予了婴幼儿巨大的学习潜力和可塑性，使他们能够轻松地吸收语言、技能和各种信息。
- 大脑此时就像一块海绵，尽可能多地吸收一切。

第二阶段：突触修剪（Synaptic Pruning）

- 时间窗口： 这个过程从童年早期开始，并一直持续到青春期晚期，甚至到20岁出头。
- 修剪的高峰期通常出现在青春期前后。
- 过程： 大脑并不会保留所有在早期形成的连接。
- 根据“用进废退”的原则，开始系统地清除那些不常被使用的、弱化的神经连接。
- 经常被刺激、被使用的连接（经常练习的技能、使用的语言、重复思维模式）得到加强和巩固
- 目的：
- 提高效率：减少“无效连接”的干扰，让神经信号的传递路径更专一、更迅速
- 功能特化：大脑的不同区域变得更加专业化
- 塑造自我：经历、学习和环境决定了哪些连接被保留，哪些被修剪

https://zh.wikipedia.org/wiki/突触修剪

小结

- 如果要做：优先尝试one-shot方法
- 训练W比不训练W好很多
- 训练M比不训练M好一些，但需要非常仔细地调试
- 比量化难得多，需要训练很久
- 训练数据集质量非常重要

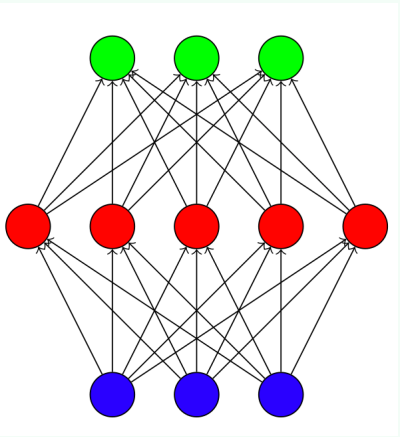
- 减少参数量，因此也减少了知识容量，难度取决于data / parameter
- 为什么有用?
- 可能性1:神经元不需要看全部输入（非结构化/半结构化剪枝）
- 可能性2:网络过参数化是为了方便优化，多余的参数在训练完毕后可以去除（所有剪枝方法）

稀疏混合专家

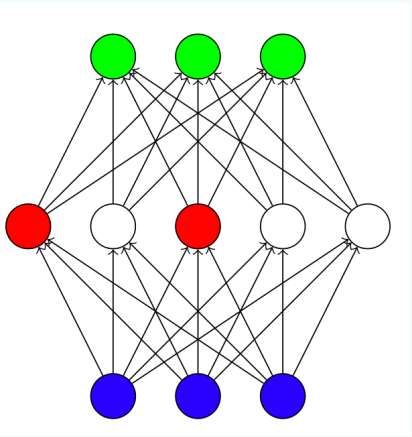
清华大学计算机系 陈键飞
《大模型计算》课程团队

激活稀疏

❖ 对于给定的输入，不是所有的神经元都需要激活



稠密激活



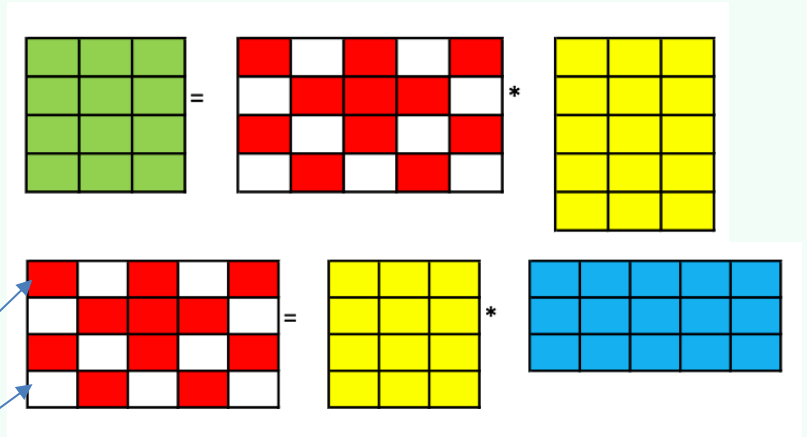
稀疏激活

FFN2

SPMM

FFN1

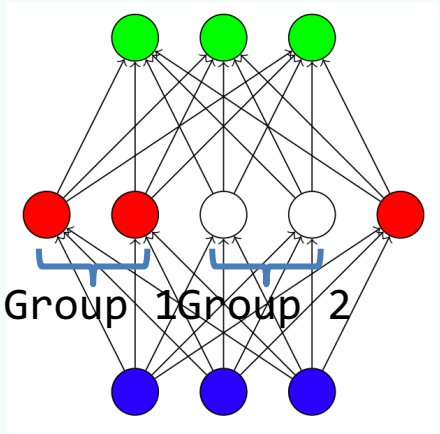
SDDMM



不同token激活的神经元不同

❖ 问题：对硬件不友好

稀疏混合专家



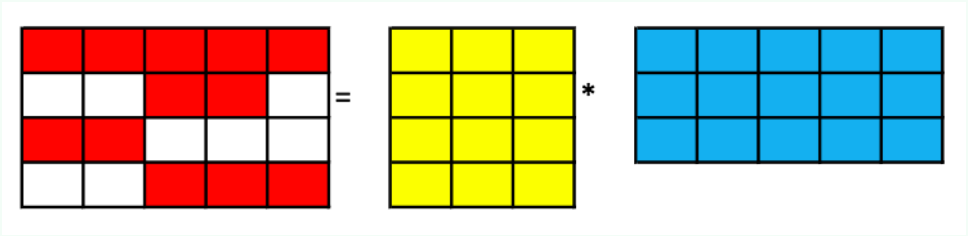
- 思路：将神经元分组，要么一起激活，要么一起不激活
- 稀疏矩阵乘法可以通过GroupGEMM高效实现
- 问题：如何预先知道每个token需要计算哪些专家?
- 通过路由器（router）

$$Z_i(x) = m_i(x) \times f_i(x)$$

稀疏掩码预测

即*router*

分块稀疏激活



$$m(x) = \mathbf{topk}(\mathrm{softmax}(g(x)))$$

举例：DeepSeekv3

- 257个专家，1个总激活，剩余的256选8
- 61层（其中4~61层是MoE），Hidden dim=7168，Expert dim=2048
- MoE部分总参数量 = 58 * 7168 * 2048 * 3 * 257 = 656B
- MoE部分激活参数量 = 58 * 7168 * 2048 * 3 * 9 = 23B
- 稠密FFN激活参数量 = 3 * 7168 * 18432 * 3 = 1.2B
- 注意力部分激活参数量约为11.4B（之后讲解）
- LM Head = 0.9B
- 路由参数 = 58 * 7168 * 256 = 0.1B
- 共计36.6B
- 直觉：MoE部分是一个巨大的知识库，根据输入查找对应的知识

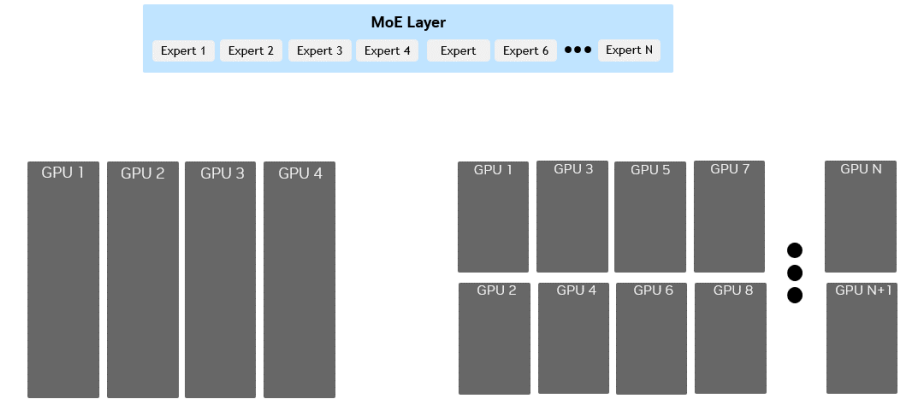
负载均衡

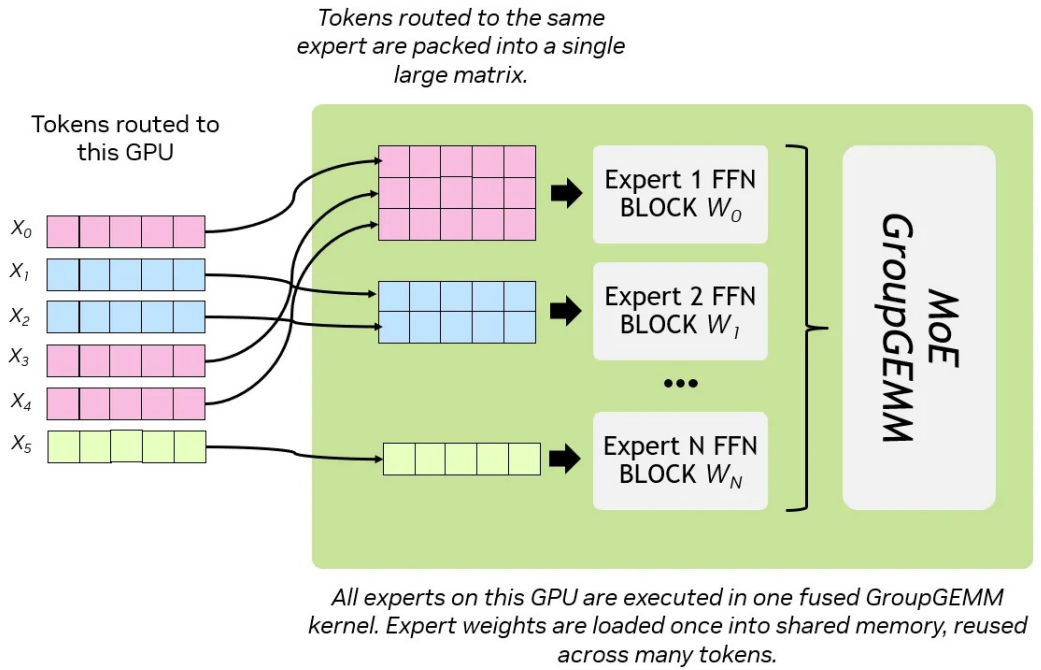
❖ 通过负载均衡损失（load balancing loss）

- **专家负载：** $f_e = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}(\text{Token } t \text{ selects Expert } e)$
  - 对于E选k的MoE，我们有 $\sum_{e=1}^{E} f_e = \frac{k}{E}$
- **优化目标：** $\min \sum_{e=1}^{E} f_e^2$
  - 然而，f_e不可导，该函数无法传梯度
  - 解决方案：将其中一个f_e通过可导的平均路由得分近似（平均分数越高，选择该专家的token数往往越多）
  - 平均路由得分：$P_e = \frac{1}{T}\sum_{t=1}^{T} s_{e,t}$，其中 $s_{e,t}$ 为 Token t 对 Expert e 的路由得分
- **损失函数：** $\boxed{\mathcal{L}_{lbl} = \sum_{e=1}^{E} f_e P_e}$
- **作用：**
  - 不加负载均衡loss，会导致routing collapse（路由塌缩），即模型只倾向于选择少数几个专家，剩下的专家会失活
  - 负载均衡loss能让专家分配的token数接近，有利于专家并行

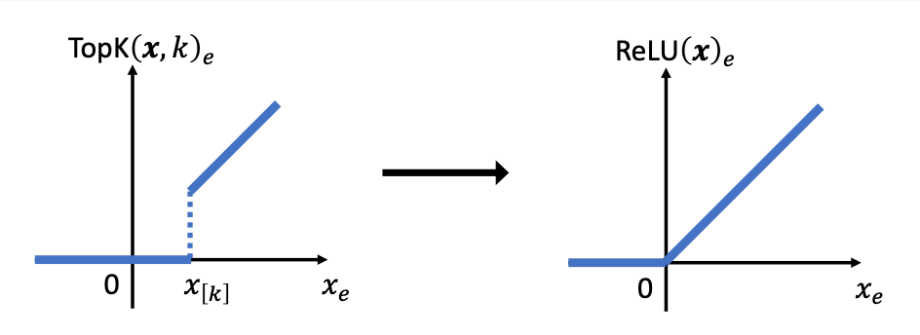
实现

- 计算逻辑：为每个Expert找到所有路由到的Token，拼成稠密矩阵
- 实现细节：专家并行 + 分组矩阵乘
  - 专家并行（Expert Parallel，EP）
    - 将不同的专家分到不同的卡上，各张卡上的Token找到合适的专家进行通信，并行计算
  - 分组矩阵乘（Group General Matrix Multiply，GroupGEMM）
    - 对于同一张卡的不同专家，同时计算矩阵乘法，避免等待上一个专家算完后才开始计算下一个专家





不连续性



离散Topk路由

$$\mathrm{m(x)} = \mathbf{topk}(\mathrm{softmax}(\mathrm{g}(x)))$$

连续ReLU路由

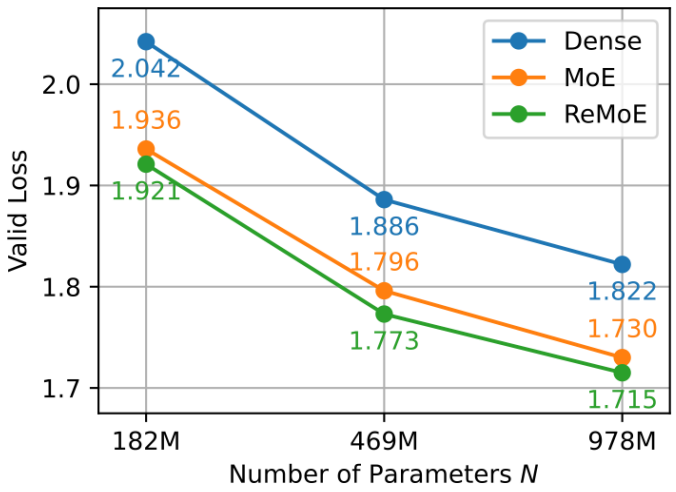
$$\mathrm{m(x)} = \mathbf{ReLU}(\mathrm{g}(x))$$

✅ **稀疏**

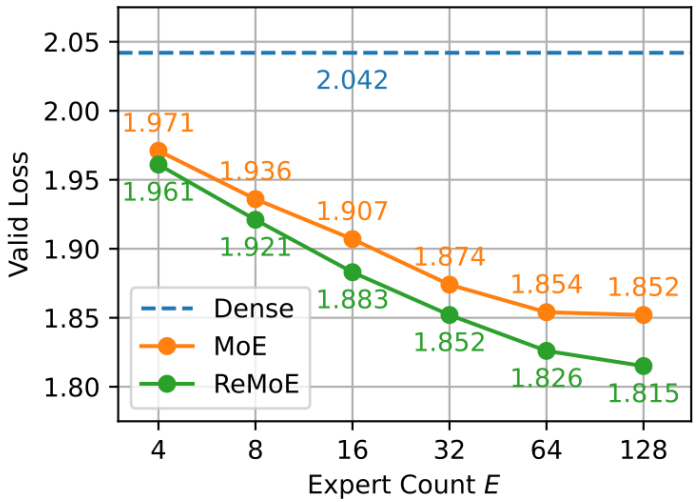
✅ **连续**

✅ **可微**

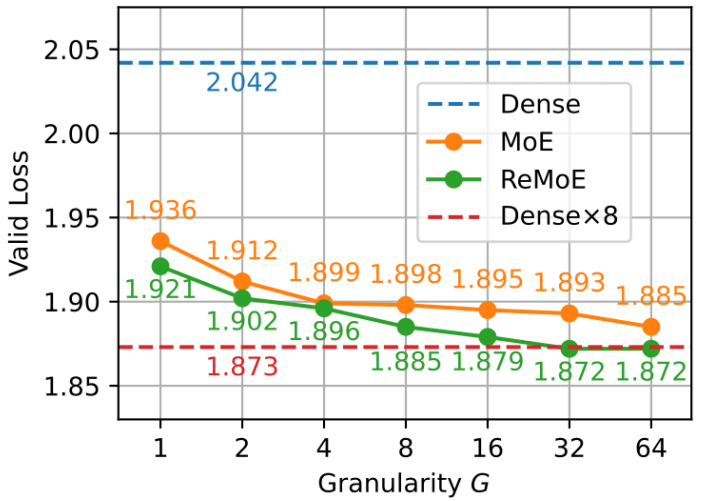
结果



(a) Scaling in N

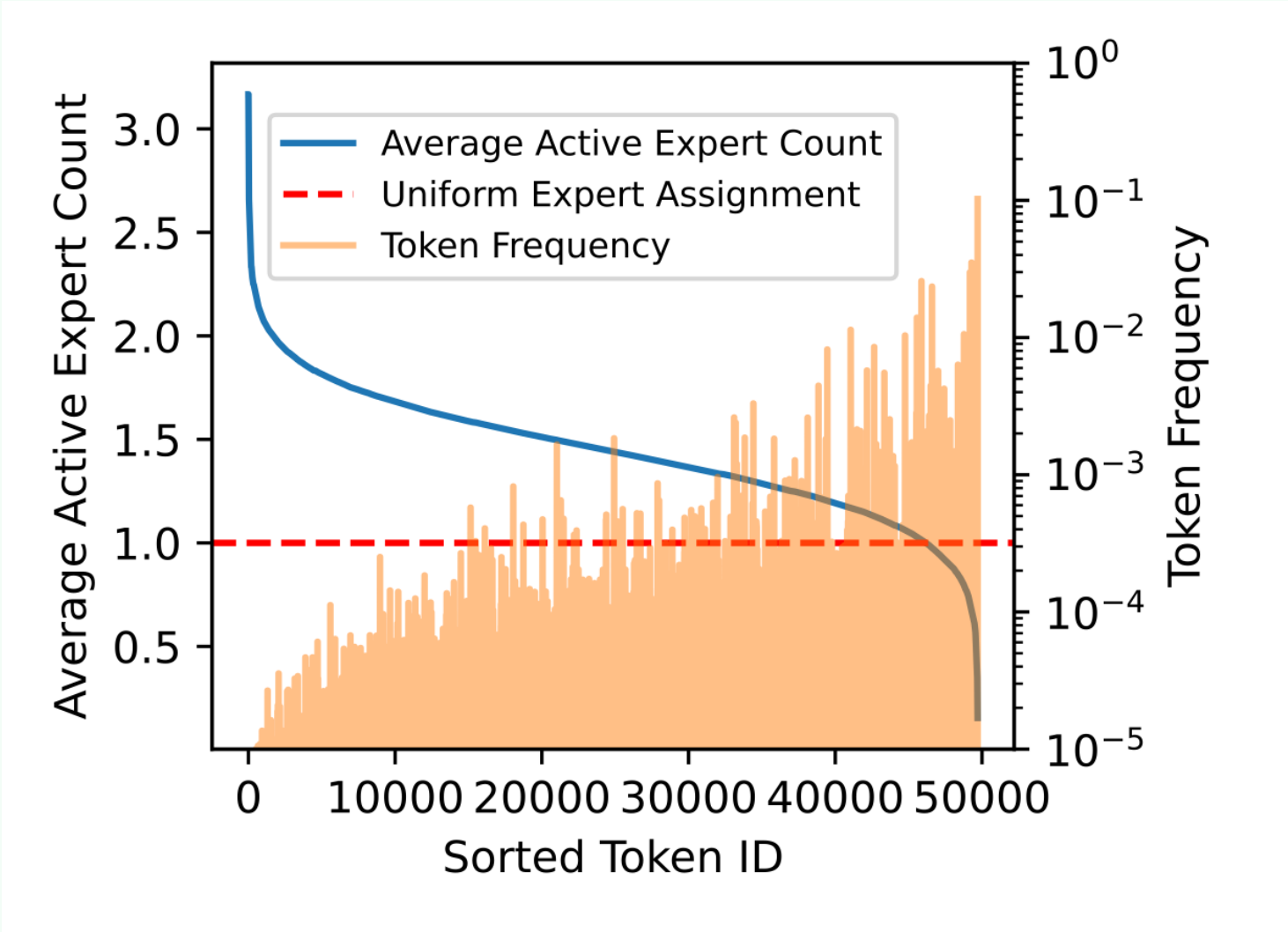


(b) Scaling in E



(c) Scaling in G

动态计算量分配


Average Active Expert Count
Uniform Expert Assignment
Token Frequency
Average Active Expert Count
Token Frequency
Sorted Token ID
3.0
2.5
2.0
1.5
1.0
0.5
0
10000
20000
30000
40000
50000
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}

小结：MoE

- Routing策略上有多种变体
- 其他改进：细粒度专家，级连专家，etc.
- 目前认为37B active MoE约等于100B+的dense model
- 为什么不等于600B的?
  - 深度不够
  - Hidden不够
  - 模型越大边际效益递减

谢谢

清华大学计算机系 陈键飞

《大模型计算》课程团队