低精度推理

清华大学计算机系 陈键飞

《大模型计算》课程团队

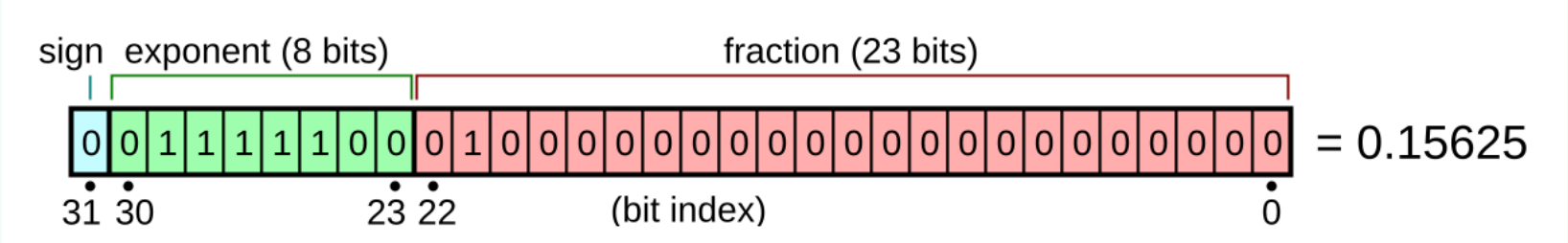
动机

	A100	H100	B200	RTX 4090	RTX 5090
BF16	312	1000	2250	165	209.5
FP8	N/A	2000 (2x)	4500 (2x)	330 (2x)	419 (2x)
INT8	624 (2x)	2000 (2x)	4500 (2x)	660 (4x)	838 (4x)
INT4	1248 (4x)	N/A	N/A	1320 (8x)	N/A
FP4	N/A	N/A	9000 (4x)	N/A	1676 (8x)

- BF16→FP4/INT4: 4～8倍加速
- FP16→INT1: 16倍存储空间节省

低精度数值格式

❖ IEEE 754



❖ **数值表示：**

$$(-1)^{s}\times(1.M)\times 2^{(E-B)}\text{(规格化数：E非全零)}$$

$$(-1)^{s}\times(0.M)\times 2^{(1-B)}\text{(非规格化数：E全零)}$$

❖ **对于FP32：** $B=127$

❖ **例中：** $(01111100)_2=124$，$(1.01)_2=1.25$，$1.25\times 2^{-3}=0.15625$

❖ **最小数值：** $2^{-23}\times 2^{-126}=2^{-149}\approx 1.4\times 10^{-45}$

❖ **最大数值：** $(2-2^{-23})\times 2^{127}\approx 3.4\times 10^{38}$

数值类型

数值	指数	尾数	最小值	最大值
FP32	8	22	1.2e-38	3.4e38
FP16	5	10	6e-8	65504
BF16	8	7	1.2e-38	3.4e38
FP8 (E5M2)	5	2	1.2e-38	5.7e4
FP8 (E4M3)	4	3	6.1e-5	448
FP4	2	1	0.5	6
INT8	0	7	-128 ~ 127	
INT4	0	3	-8 ~ 7	

低精度张量

❖ 高精度矩阵如何用低精度矩阵近似?

方法1: $\mathbf{X} \approx \mathrm{int}(\mathbf{X})$

❖ 问题：不同矩阵数值差异很大

❖ 矩阵A：[-1e-3，+1e-3]　　# 全零（下溢）

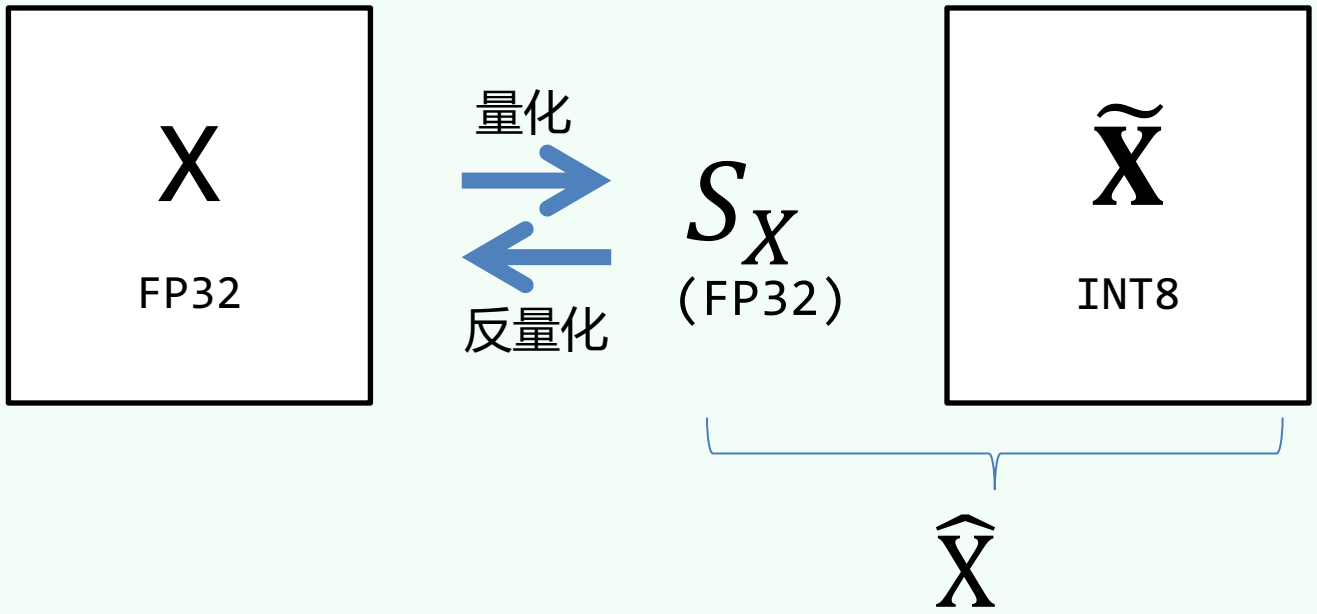
❖ 矩阵B：[-1e+4，+1e+4]　　# 全-128或+127（上溢）

❖ 解决方案：利用缩放因子（scaling factor）捕捉张量的数值范围

$$Q(\mathbf{X}) = s_x \times \mathrm{round}\left(\frac{\mathbf{X}}{s_x}\right) \approx \mathbf{X}$$

❖ $s_x = \max_{ij} |X_{ij}| / 127,\ \frac{X}{s_x} \in [-127, +127]$

量化与反量化




```
def quantize(X):
    s_x = max_ij |X_ij| / 127
    X̃ = round(X / s_x)
    return s_x, X̃
```

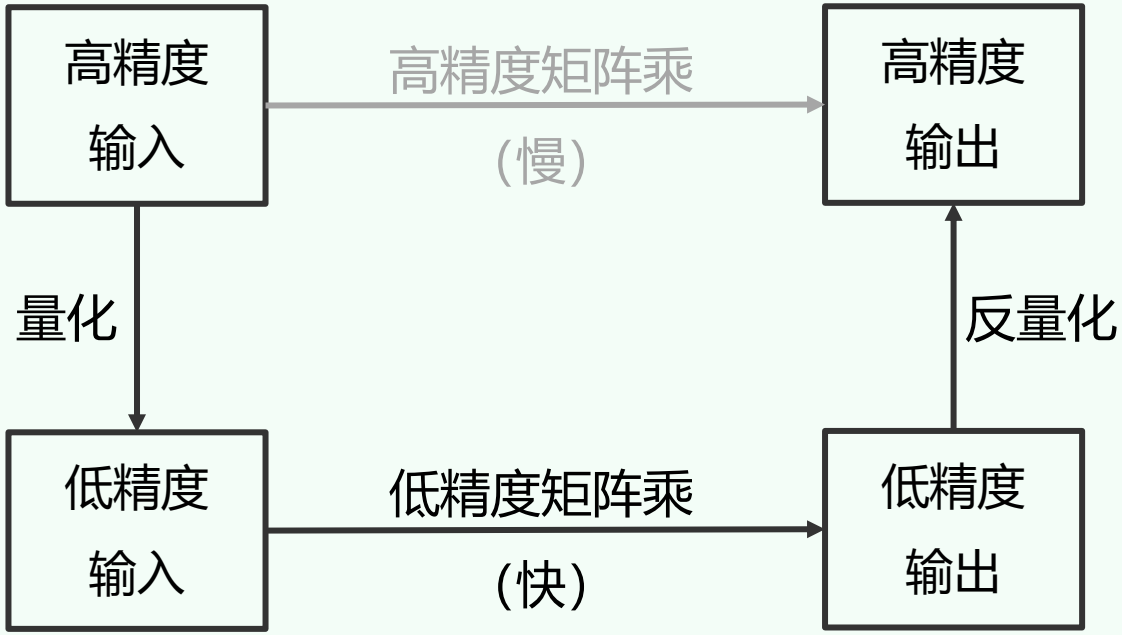
```
def dequantize(s_x, X̃):
    return s_x × X̃
```

低精度矩阵乘

$$\mathbf{X} \approx s_x \widetilde{\mathbf{X}}, \qquad \mathbf{Y} \approx s_y \widetilde{\mathbf{Y}}$$

$$\mathrm{XY} \approx (s_x s_y)\widetilde{\mathbf{X}}\widetilde{\mathbf{Y}}$$

```
def QGEMM(X, Y):
    s_x, X̃ = quantize(X)
    s_y, Ỹ = quantize(Y)
    Z̃ = X̃Ỹ        低精度矩阵乘（高效）
    return dequantize(s_x s_y, Z̃)
```



累加器

❖ $\tilde{Z}$的数值精度是什么?

$$\tilde{Z}_{ij} = \sum_{k=1}^{K} \tilde{X}_{ik}\tilde{Y}_{kj}$$

```
def QGEMM(X, Y):
    s_x, X̃ = quantize(X)
    s_y, Ỹ = quantize(Y)
    Z̃ = X̃Ỹ        低精度矩阵乘（高效）
    return dequantize(s_x s_y, Z̃)
```

```
for each i do
    for each j do
        z[i, j] = 0 ——→ 累加器 (accumulator)
        for each k do
            z[i, j] += x[i, k] * y[k, j]
                       └──────┘
                        INT16
```

❖ 对于INT输入，累加器均为INT32

❖ 对于FP输入，累加器均为FP32

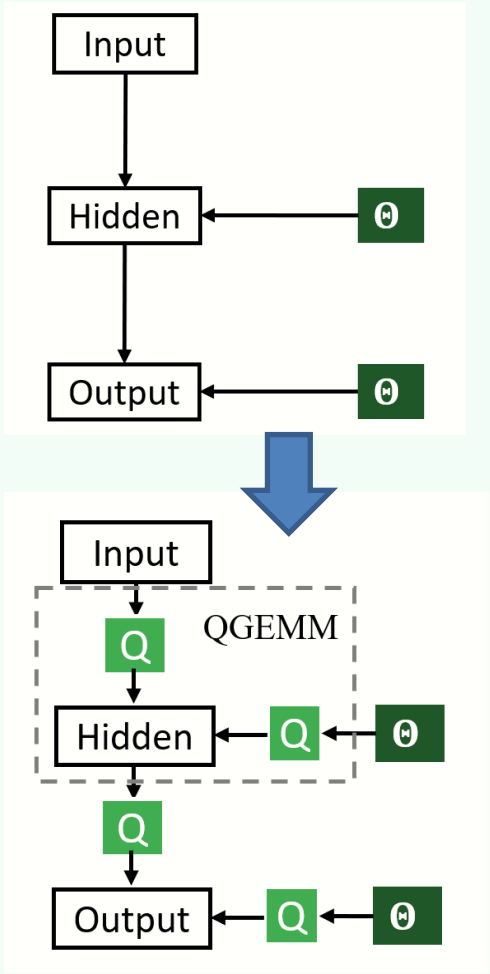
大模型中的量化

- 高度取决于要实现什么功能？对于每个场景，能达到的精度完全不同……

- 加速（串行）解码：权重量化、KV量化
- 加速（并行）解码/预填充：权重+激活量化，QKV量化
- 加速训练：权重+激活+梯度量化
- 降低KV缓存显存占用：KV量化

- 降低权重显存占用/模型存储空间：权重/优化器状态量化
- 降低KV缓存显存占用：KV量化
- 降低通信量：梯度量化

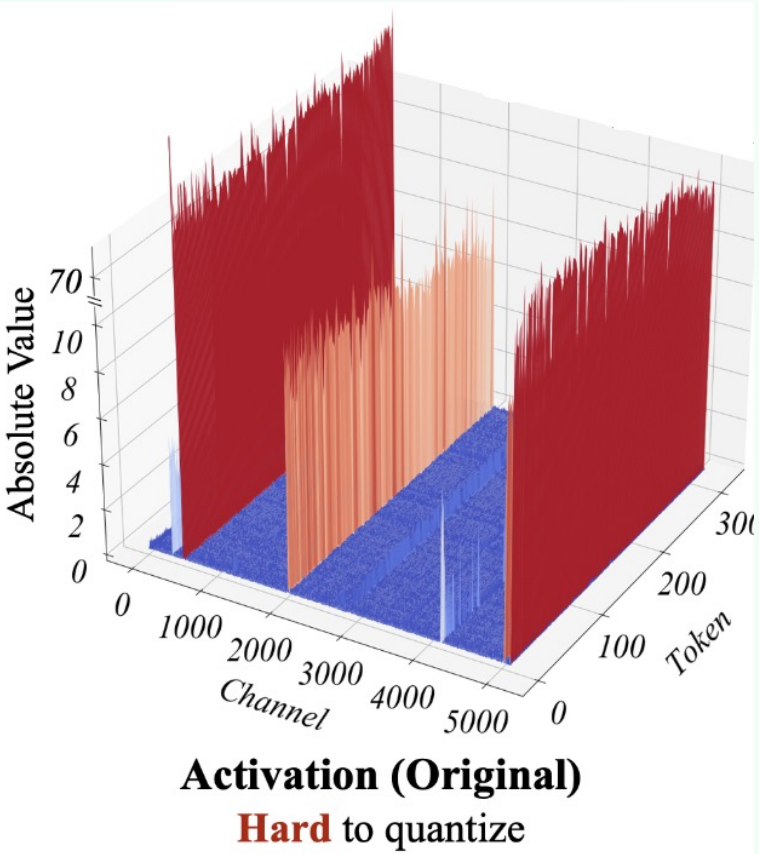
训练后量化（post-training quantization）



- 直接在训练完毕的模型中将 权重/激活 替换为量化后的版本
- 优势：方便快捷
- 劣势：准确率损失严重（俗称“掉点”）
  - 第一层输入/权重不准 造成 第一层输出不准
  - 因此第二层输入更加不准
  - 误差不断累积……
- 如何尽量不损失精度？两个线索
  - 让$Q(\mathrm{X}) - \mathrm{X}$尽量小
  - 让$Q(\mathrm{X})Q(\mathrm{Y}) - \mathrm{XY}$尽量小

离群值

- 实际的张量内数值差异巨大
- 最大值有时可达平均绝对值的10000倍



channel

token

0.01	1.02	0.16	100	10
0.03	0.31	2.42	300	5
0.32	1.23	1.63	200	6
0.16	1.53	0.32	400	15
0.24	2.13	1.32	1000	8

Round to zero
Huge information loss

0	0	0	104	0
0	0	0	304	8
0	0	0	200	8
0	0	0	400	16
0	0	0	1000	8

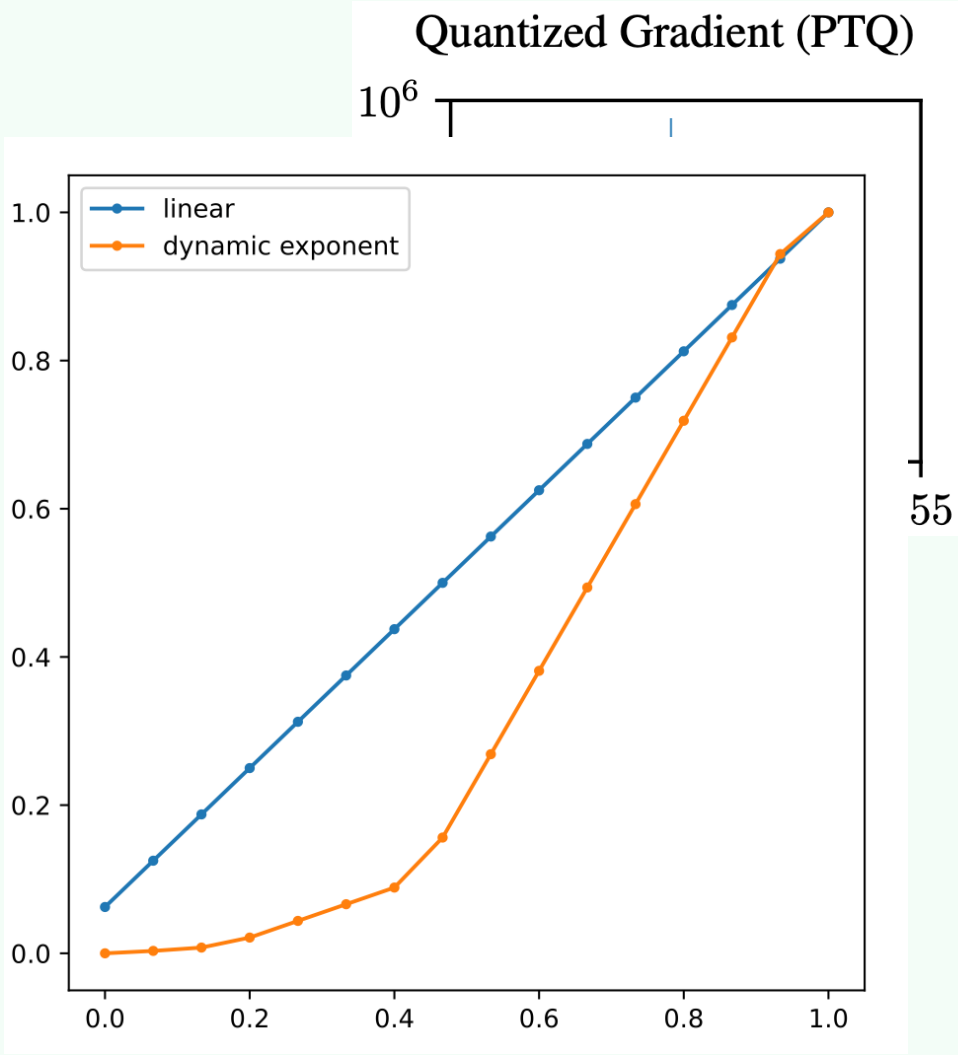
INT8 quantization: scales the largest element (1000) to +127

Resolution is $\frac{1000}{127} \approx 8$

解决方案1：非均匀量化

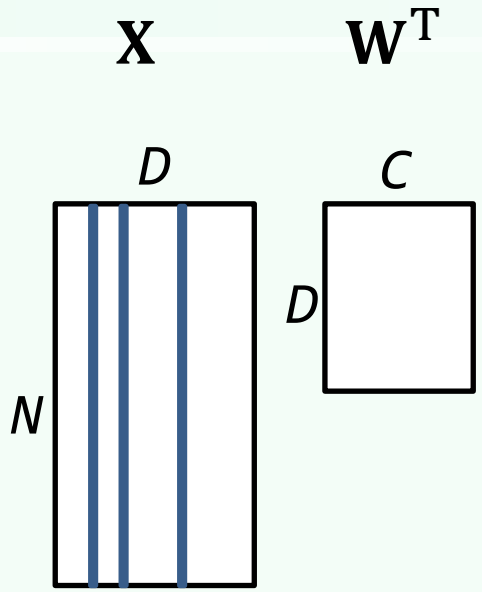
- 小数更多，大数更少
- 思路：接近0处量化点更多，远离0处量化点更少
- 整数：均匀
- 浮点数：非均匀

- 没有绝对的优劣，要看**数值格式**与**数值分布**是否一致
- 若数值分布非均匀：浮点更好
- 若数值分布均匀：指数更好

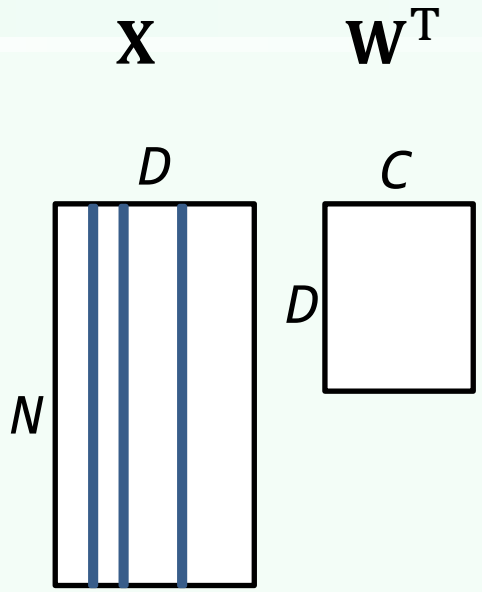


解决方案2：细粒度缩放

- 离群值模式
- $\mathbf{Y} = \mathbf{X}\mathbf{W}^{\mathrm{T}}$
- X：N词元 * D维度
- W：C通道 * D维度



解决方案2：细粒度缩放



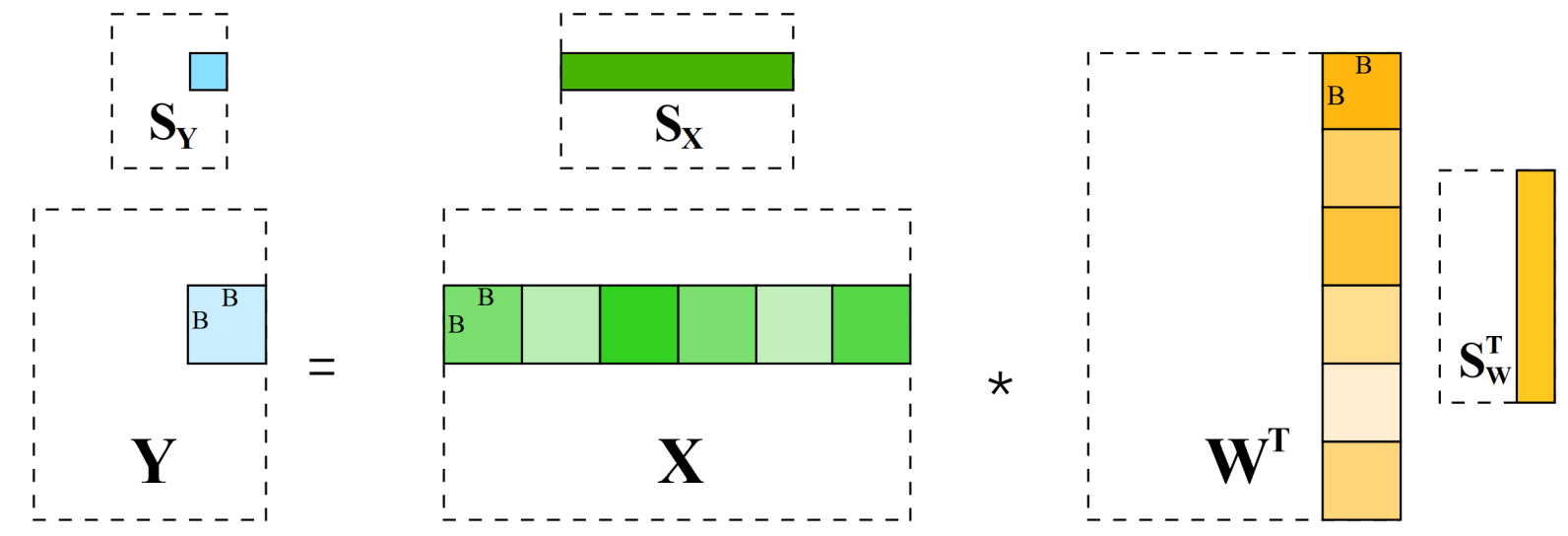
- **逐词元缩放：** $Q(\mathbf{X}) = \mathrm{diag}(\boldsymbol{s_x})\widetilde{\mathbf{X}}$
  - **缩放因子** $\boldsymbol{s_x} \in \mathbb{R}^N$
- **逐通道缩放：** $Q(\mathbf{W}) = \mathrm{diag}(\boldsymbol{s_w})\widetilde{\mathbf{W}}$
  - **缩放因子** $\boldsymbol{s_w} \in \mathbb{R}^C$
- **矩阵乘法** $Q(\mathbf{X})Q(\boldsymbol{W}) = \mathrm{diag}(\boldsymbol{s_x})\left(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}}^T\right)\mathrm{diag}(\boldsymbol{s_w})$
- **思考：逐词元缩放并不能处理X的列状离群值，为什么不用逐维度缩放来完全排除离群值影响?**
- $Q(\mathbf{X}) = \widetilde{\mathbf{X}}\mathrm{diag}(\boldsymbol{s_x}), Q(\mathbf{X})Q(\boldsymbol{W}) = \left(\widetilde{\mathbf{X}}\mathrm{diag}(\boldsymbol{s_x})\widetilde{\mathbf{W}}^T\right)\mathrm{diag}(\boldsymbol{s_w})$
- $\left(\widetilde{\mathbf{X}}\mathrm{diag}(\boldsymbol{s_x})\widetilde{\mathbf{W}}^T\right)_{ij} = \sum_{k=1}^{D} X_{ik}W_{jk}(s_x)_k$
- **取决于目的：如果只是为了压缩X，可以；如果是为了加速XW^T，不行**

解决方案2：细粒度缩放—分块缩放

- X分成大小为(n，d)的块，每块为X_{ik}，缩放因子为s_{ik}
- W^T分成大小为(d，c)的块，每块为W_{kj}^T，缩放因子为t_{kj}
- 则$Y_{ij} = \sum_{k=1}^{K} (s_{ik} t_{kj}) (X_{ik} W_{kj}^T)$
- 计算流程：全局累加器 += 反量化（张量内核累加器）

Jetfire: n=d=c=32

DeepSeekv3: n=1, d=c=128



解决方案2：细粒度缩放—微缩放（microscaling）

MXFP数值格式：n=c=1，d=32

缩放因子为UE8M0

- MXFP8
- MXFP6（速度同FP8）
- MXFP4（速度同FP8）

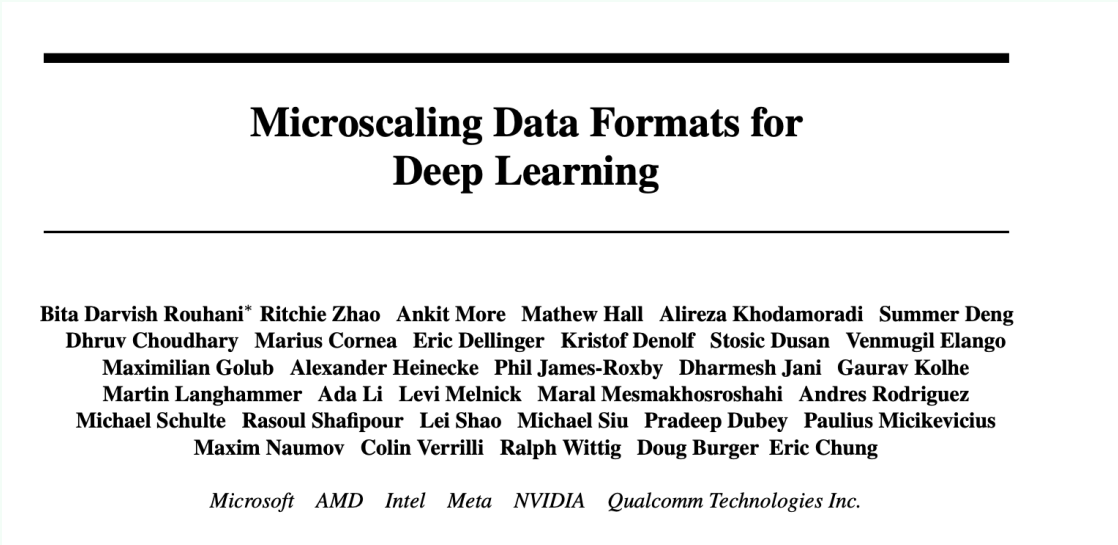
NVFP数值格式：n=c=1，d=16

缩放因子为E4M3

- NVFP6（速度同FP8）
- NVFP4（2～4x快于FP8）

Nvidia Blackwell起支持

几乎完全解决了离群值的问题，且自带反量化，计算很快



Microscaling Data Formats for Deep Learning

Bita Darvish Rouhani* Ritchie Zhao Ankit More Mathew Hall Alireza Khodamoradi Summer Deng
Dhruv Choudhary Marius Cornea Eric Dellinger Kristof Denolf Stosic Dusan Venmugil Elango
Maximilian Golub Alexander Heinecke Phil James-Roxby Dharmesh Jani Gaurav Kolhe
Martin Langhammer Ada Li Levi Melnick Maral Mesmakhosroshahi Andres Rodriguez
Michael Schulte Rasoul Shafipour Lei Shao Michael Siu Pradeep Dubey Paulius Micikevicius
Maxim Naumov Colin Verrilli Ralph Wittig Doug Burger Eric Chung

Microsoft AMD Intel Meta NVIDIA Qualcomm Technologies Inc.

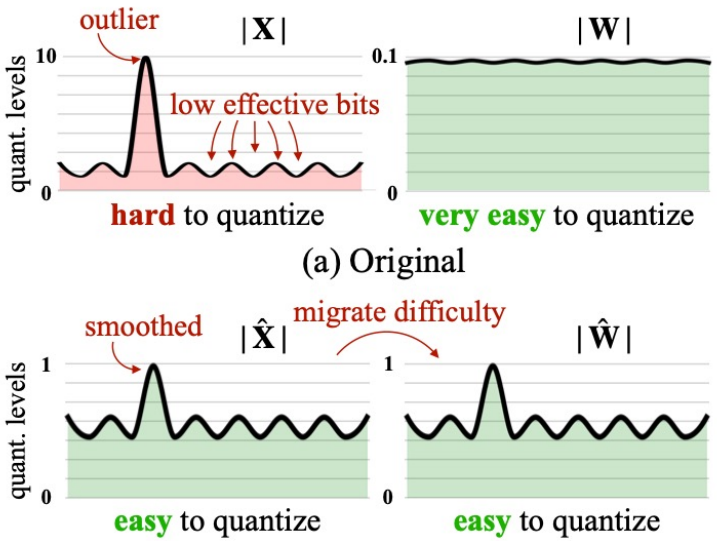
解决方案3：旋转—SmoothQuant

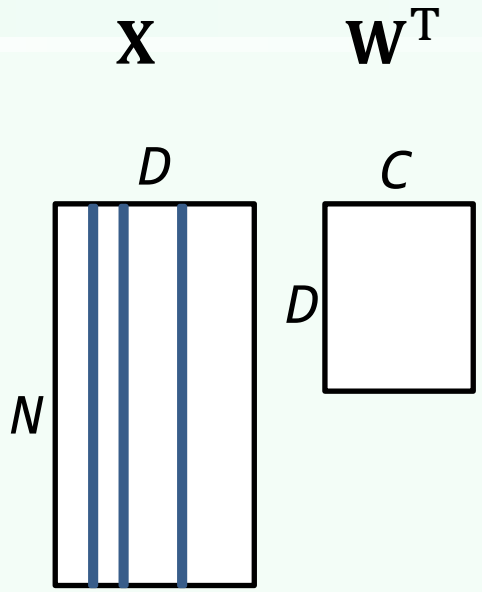
- 主要发现：

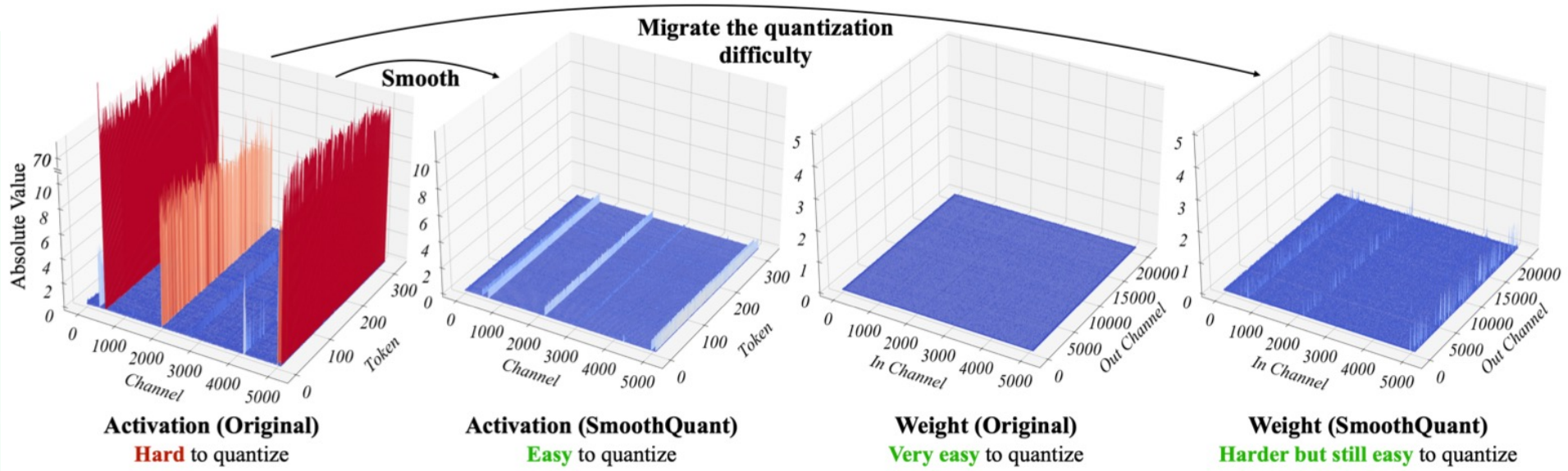
$$\mathrm{X}\mathrm{W}^{T} = \mathrm{X}PP^{T}\mathrm{W}^{T} \approx Q(XP)Q(P^{T}\mathrm{W}^{T})$$

- 可以在其他子空间做量化
- 量化难度可以在X和W间分配

解决方案3：旋转—SmoothQuant







解决方案3：旋转—哈达马变换

- 考虑特例：X矩阵（NxD）只有一列很大（0～10000），其他列均很小（0～1）
- 量化误差正比于缩放因子
- 直接量化：误差为O(ND)
- 逐列量化：误差为O(N)—离群值列不会“牵连”其他列
- 但是绝大部分的比特都用来存储非离群值，而这些数都接近零！
- 宝贵的比特浪费在存储无信息量的零上面！ # 离群值究竟是否更有信息量？为什么？
- 哈达马变换：将离群列的信息均摊到其他列上
- 误差为O(N/D)

0.01	1.02	0.16	100	10
0.03	0.31	2.42	300	5
0.32	1.23	1.63	200	6
0.16	1.53	0.32	400	15
0.24	2.13	1.32	1000	8

- 所谓高效深度学习，是要理解神经网络如何工作，并简化其计算，在不失功能的情况下提升效率

哈达马(Hadamard)矩阵

$$H_1 = [1],$$

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},$$

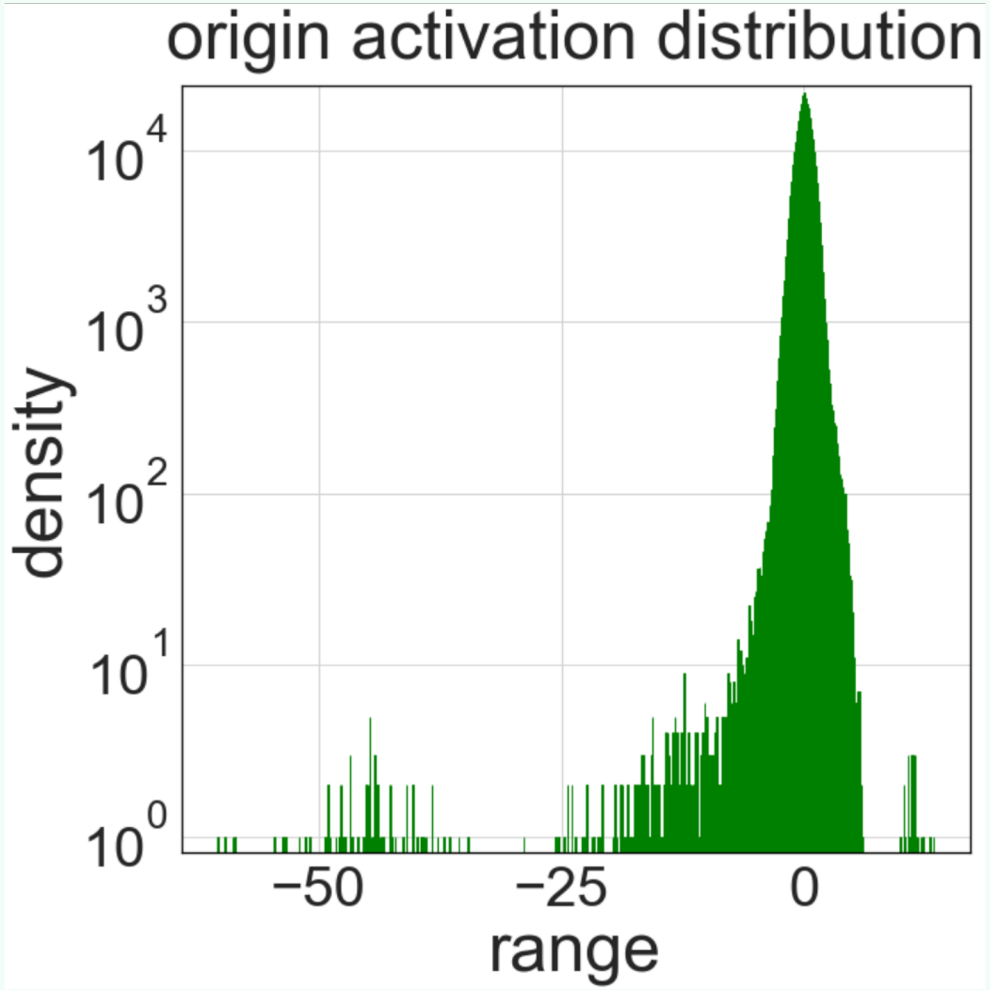
$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}$$

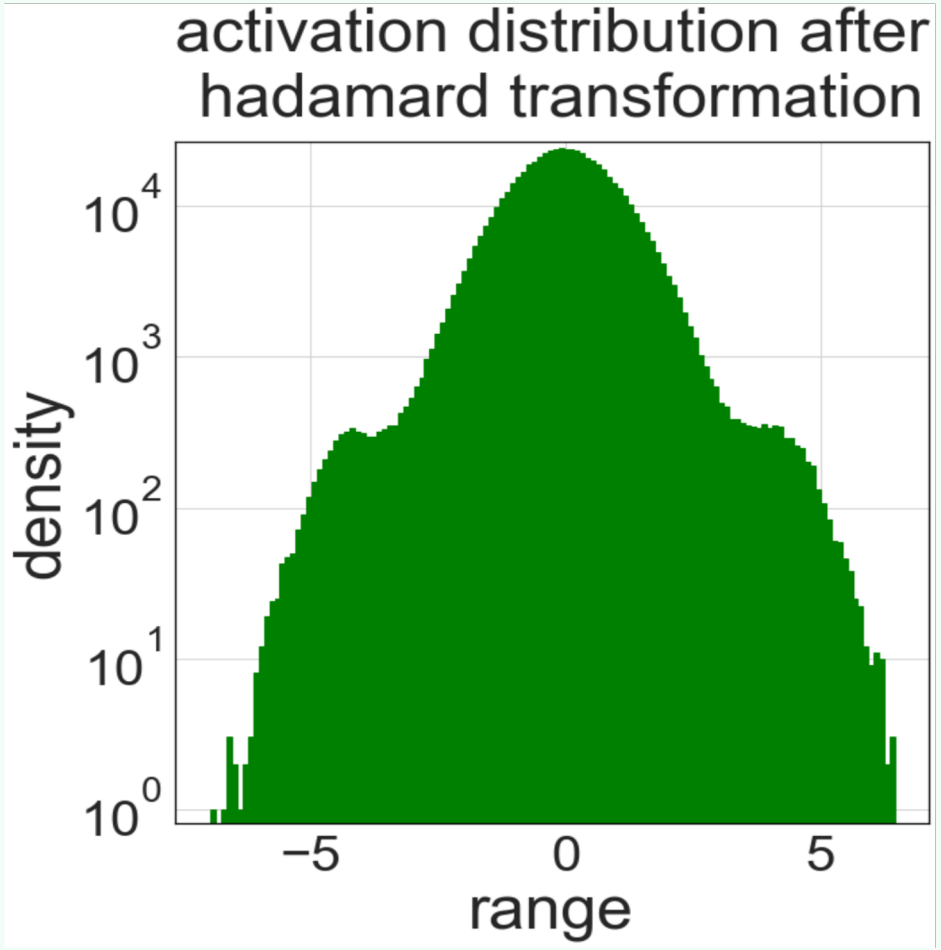
- 性质1：哈达马矩阵为对称正交阵：$HH = \mathbf{I}$
- 性质2：哈达马矩阵将单位向量转为全一向量，反之亦然

$$H_N e_i = (\pm 1, \pm 1, \ldots, \pm 1)/\sqrt{N}$$

- 因此$\boldsymbol{X}H_N$将是一个理想的，无离群值的矩阵
- 将代表示的离群值均为分布到了各个维度，每维贡献一点
- 量化$Q(\mathbf{X}) = \text{round}(\boldsymbol{X}H)H$
- 矩阵乘法$\mathbf{XW} = \mathbf{XHHW} \approx Q(\boldsymbol{XH})Q(\boldsymbol{HW})$
- 恰好处理了逐列的离群值
- 哈达马变换仅用于预处理，几乎无实现开销

解决方案3：旋转—哈达马变换

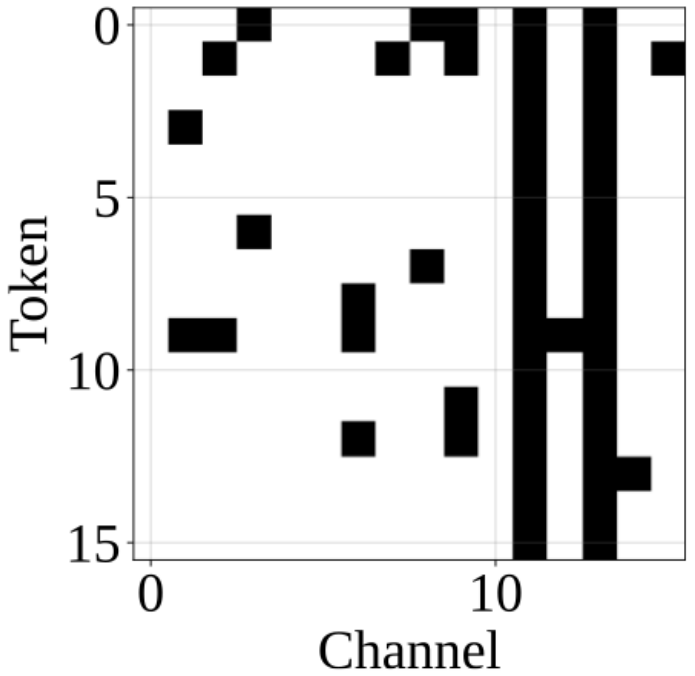




解决方案4：混合精度

- 这一列这么重要，为什么不干脆设成更高精度?
- 误差为$O(N2^{-D})$

0.01	1.02	0.16	100	10
0.03	0.31	2.42	300	5
0.32	1.23	1.63	200	6
0.16	1.53	0.32	400	15
0.24	2.13	1.32	1000	8



目前是什么状态?

- 采用了NVFP4量化以后，对神经网络中的张量进行量化，损失和对随机张量量化差不多

- 可能已逼近信息论极限，各种技术均不会再有太大效果

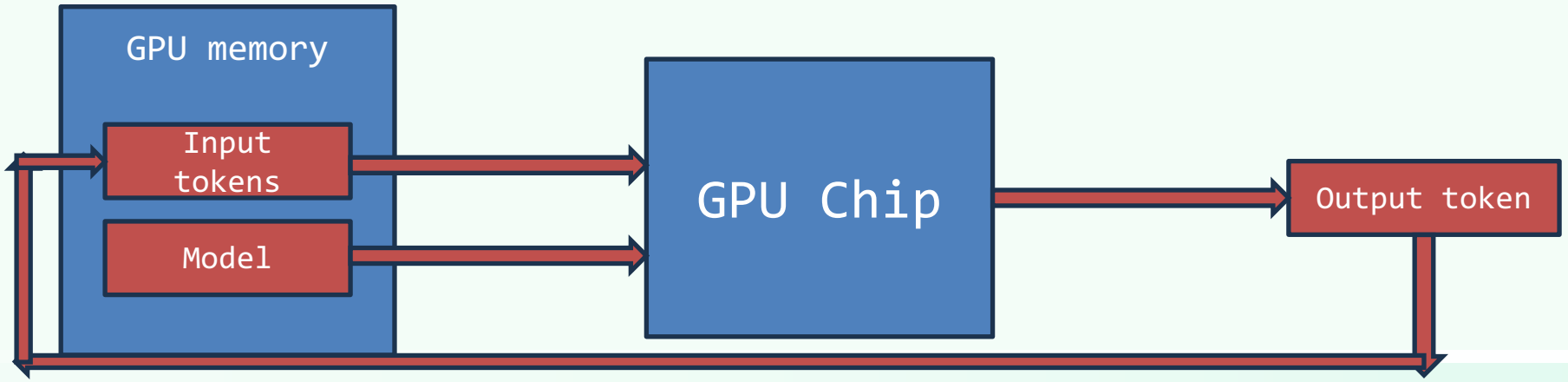
大模型量化实践

清华大学计算机系 陈键飞

《大模型计算》课程团队

仅权重量化

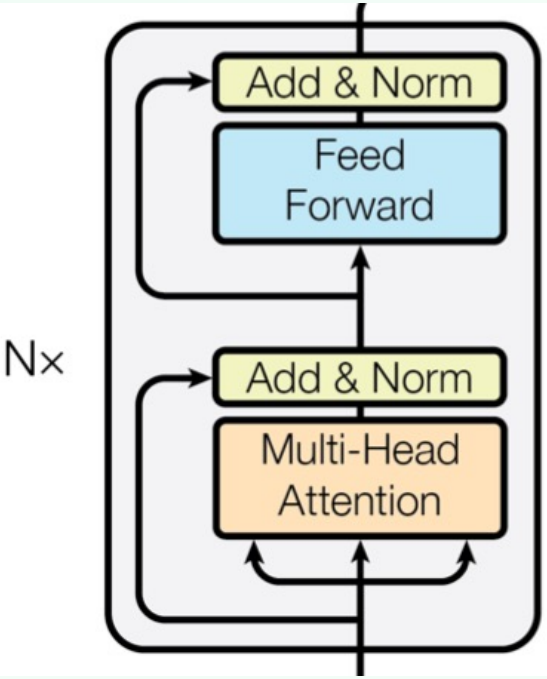
- $\mathbf{X}\mathbf{W}^{\mathrm{T}} \approx X Q(\mathbf{W}^{\mathrm{T}})$
- 之前非均匀量化/旋转/混合精度等各种trick被用来提点，现在直接转NVFP4通常损失已可接受
- 模型越大，掉点越少（为什么？）
- 访存共计15GB / token（简易算法：每个参数2字节）
- 理想情况下，如果打满带宽，解码速度为1008 / 15 = 67 token/s
- 采用4比特量化，访存除4，速度乘4



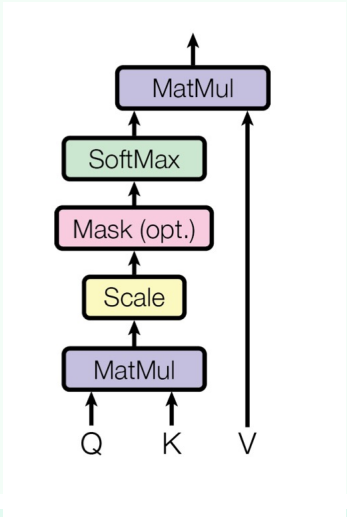
权重+激活量化

- SmoothQuant可以实现pertensor的WINT8 AINT8量化
- 直接转NVFP4损失稍大
- SVDQuant可能可以实现NVFP4量化，有待尝试

Low-precision Attention

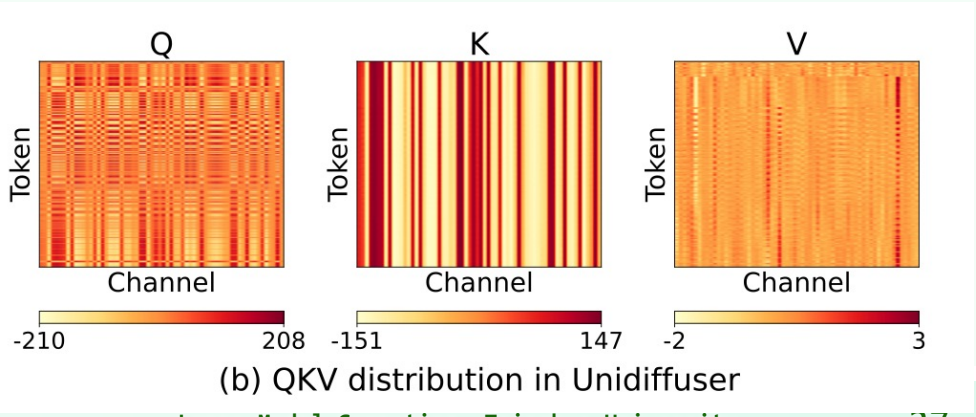


- Attention mostly remain in FP/BF16
- Long context tasks: text summarization / video generation
- MM1: $P = softmax(QK^T /sqrt(d))$
- MM2: O = PV
- Goal: quantize Q, K, P, V



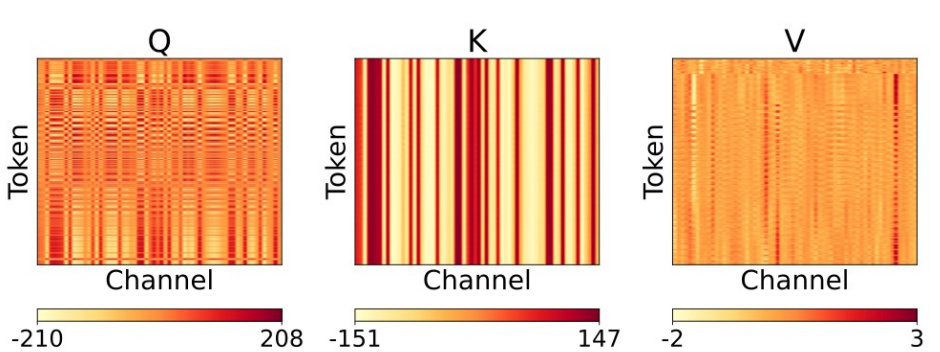
Challenges

- Problem 1: QK has large bias
- Problem 2: the logits qK^T = large constant + (small per-dim variations)
- Problem 3: the probability P has many small quantities, but the overall contribution is large
- Problem 4: Ada and Hopper's FP8 tensor core only has 22-bit accumulator
- C(FP22) = C(FP22) + A(FP8)B(FP8)



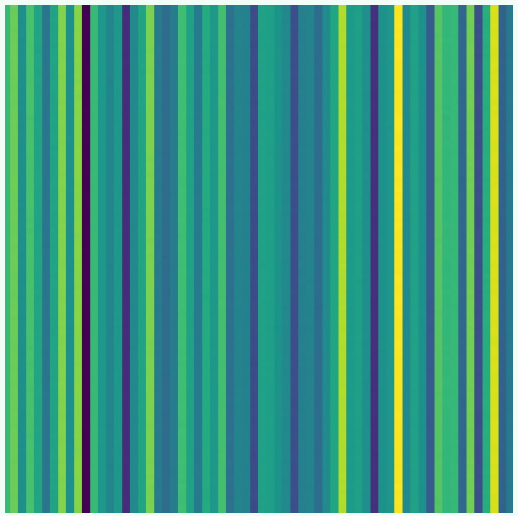
(b) QKV distribution in Unidiffuser

SageAttention

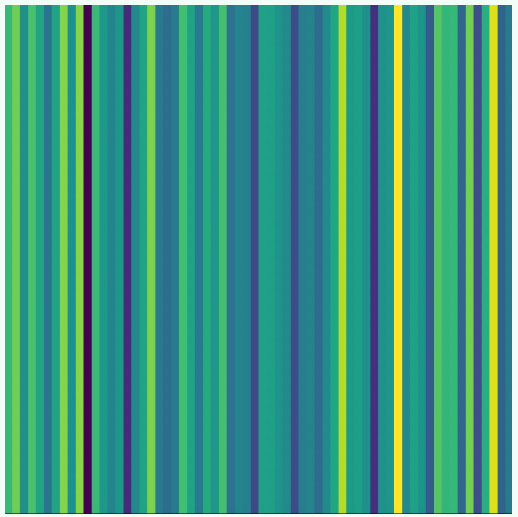


(b) QKV distribution in Unidiffuser

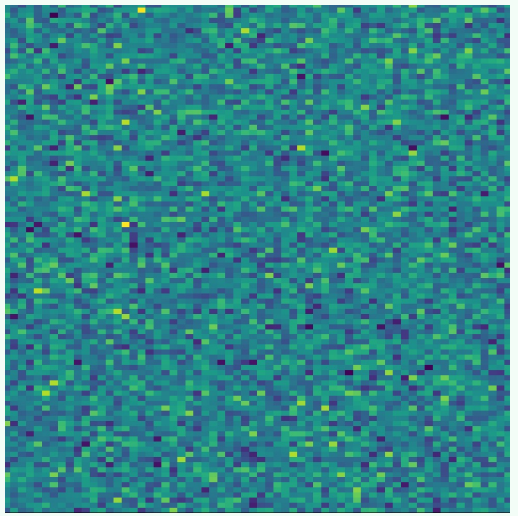
$$K = k + \Delta K$$



$=$



$+ 0.01*$



$$\text{softmax}(qK^T) = \text{softmax}(qk^T + q\Delta K^T) = \text{softmax}(q\Delta K^T)$$

SageAttention

Table 2: **Average accuracy** using different data types across all layers of real models.

Q, K	$\widetilde{P}, V$	Cos Sim ↑	Relative L1 ↓	RMSE ↓
INT8	E4M3	99.94%	0.0345	3.53e-3
	E5M2	99.81%	0.0572	6.11e-3
	INT8	99.70%	0.1035	6.82e-3
E4M3	E4M3	99.81%	0.0607	5.93e-3
	E5M2	99.68%	0.0769	7.72e-3
	INT8	99.58%	0.1199	8.31e-3
E5M2	E4M3	99.37%	0.1107	1.09e-2
	E5M2	99.22%	0.1213	1.20e-2
	INT8	99.13%	0.1583	1.24e-2

❖ **SageAttention1**

➢ Surpress K outlier by substracting mean

$$\gamma(K) = K - \text{mean}(K)$$

$$\sigma(q(K - \text{mean}(K)^\top)) = \sigma(q\hat{K}^\top - q \cdot \text{mean}(K)) = \sigma(q\acute{K}^\top)$$

❖ **SageAttention2**

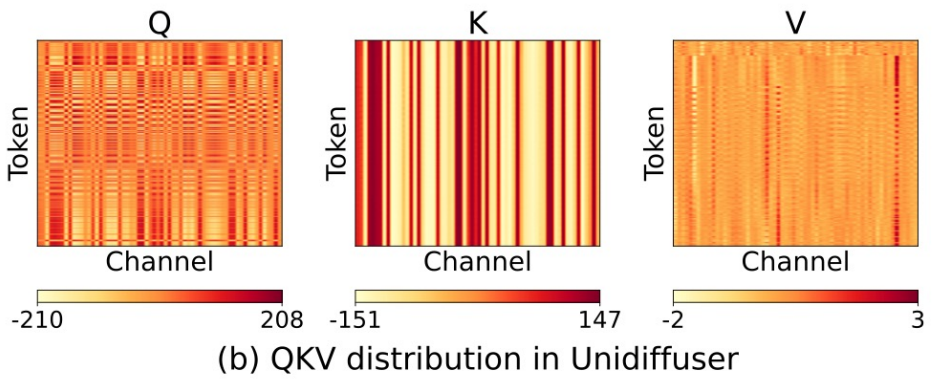
➢ Solution: further smooth K and V by subtracting the mean

❖ **SageAttention3**

➢ Utilize NVFP4 for QKPV with two-level scaling

❖ **Fine-grained dynamic quantization**

❖ **Use INT rather than FP for QK**



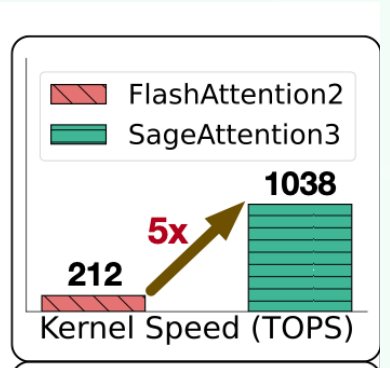
(b) QKV distribution in Unidiffuser

Zhang, Jintao, et al. "SageAttention: Accurate 8-Bit Attention for Plug-and-play Inference Acceleration." *ICLR 2025.*

Zhang, Jintao, et al. "SageAttention2 Technical Report: Accurate 4 Bit Attention for Plug-and-play Inference Acceleration." *ICML 2025.*

Results

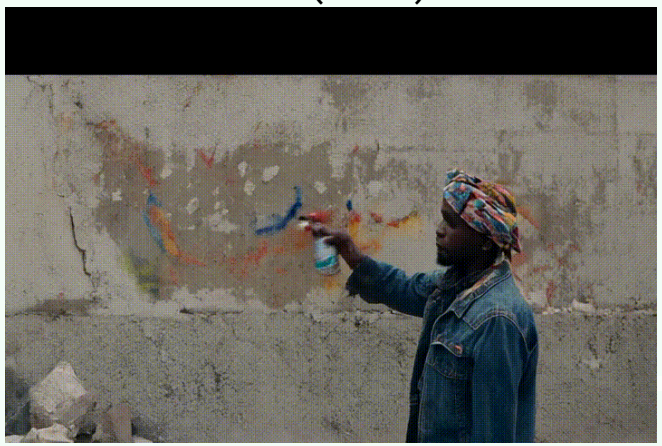
- FlashAttention2: FP16 attention (~208 Tflops on RTX 4090)
- SageAttention1 recipe: (~473 Tflops)
  - INT8 QK (4x speedup)
  - FP16 PV with FP16 accumulator (2x speedup)
- SageAttention2 recipe: (~537 Tflops)
  - INT4 QK (8x speedup)
  - FP8 PV with FP22 accumulator (2x speedup)
- SageAttention3 recipe: (~1022 Tflops)



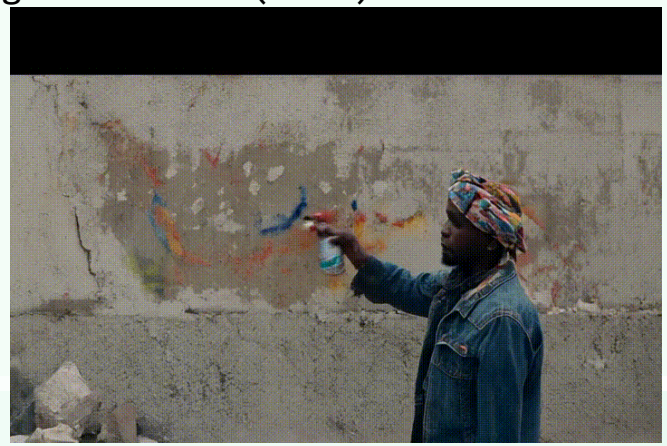
HunyuanVideo

540p, 4s

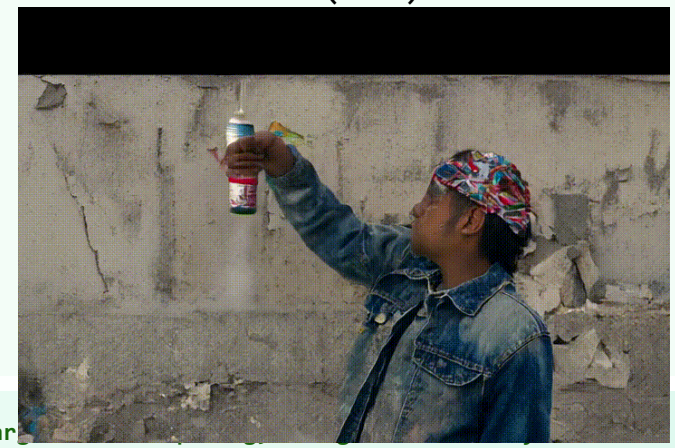
FlashAttention 2 (FP16)



SageAttention (INT8)



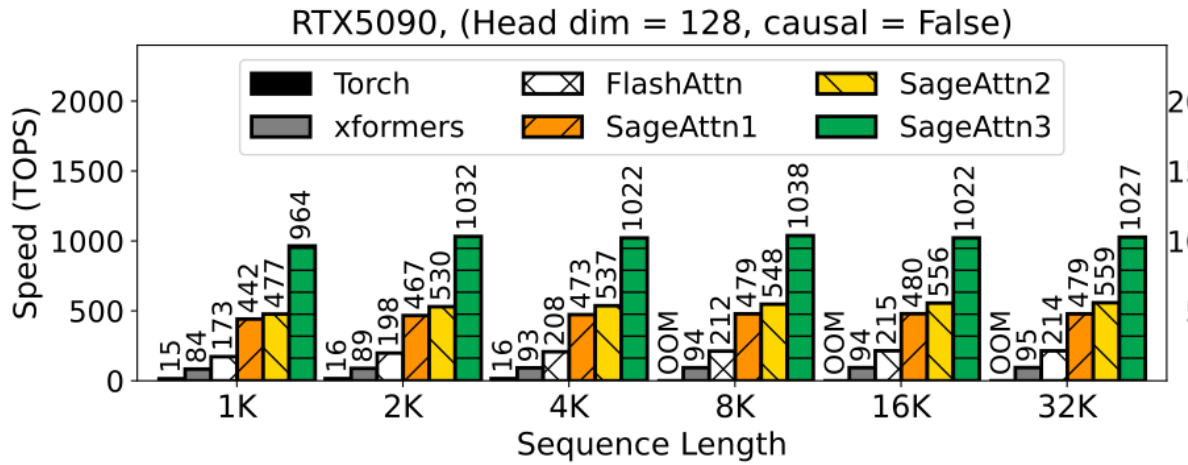
FlashAttention 3 (FP8)

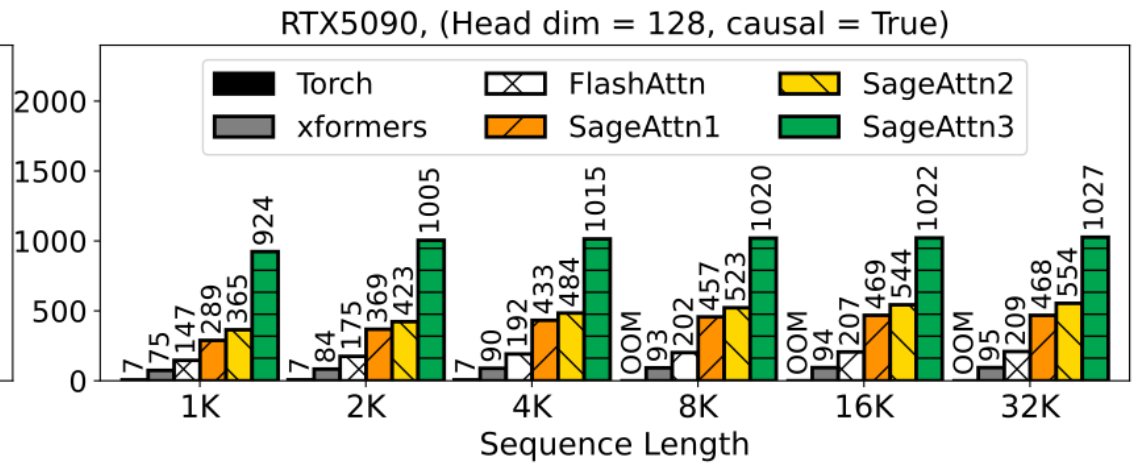


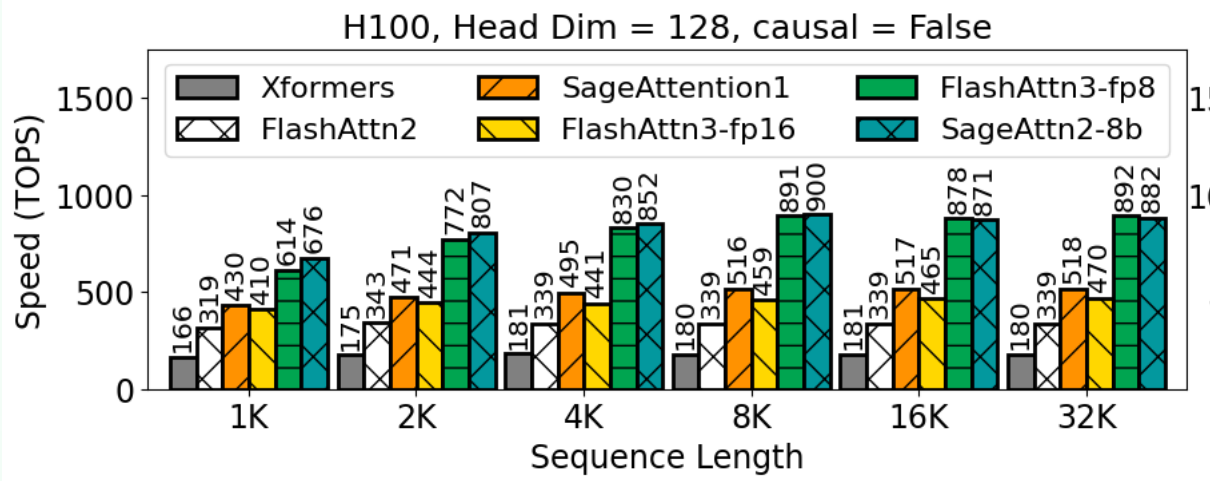
Results

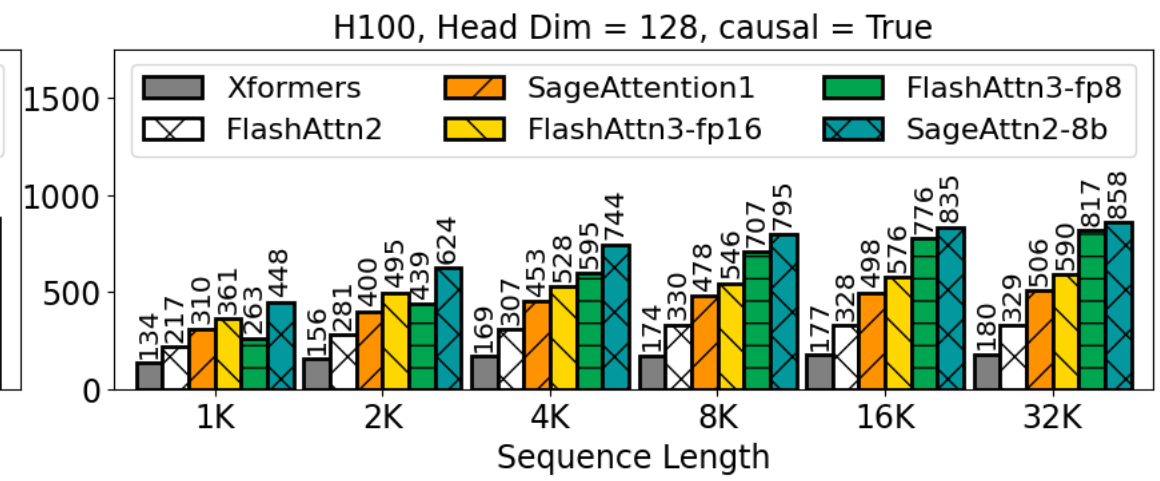
- SageAttention3, 1K seqlen, 964 Tflops @ RTX5090
- FlashAttention3, 32K seqlen, 892 Tflops @ H100

5090 beats H100!!! 10x cheaper!!!









KV量化

- 注意到只有KV需要量化，Q不需要量化
- Again，旋转。把困难的问题都留给Q
- $S = QK^T = \left(Q\mathrm{diag}(s)\right)(\mathrm{diag}(s)^{-1}K^T)$

- 目前可以做到4-bit KV
- Llama-3-8B, seqlen=8192
- KV缓存：1GB
- 4比特量化后：256MB
- 16GB显存即可做到batchsize=64，配合权重量化理论上足够打满GPU算力

谢谢

清华大学计算机系 陈键飞

《大模型计算》课程团队