PTQ

Post-training quantization(PTQ)

先训练模型再做量化，量化时拿一些训练数据集做校准，得到量化过程中的常数（比如缩放数值的选择），量化之后不再微调参数。后面介绍的都是PTQ的方法。

ZeroCount

激活值的绝对值范围要远大于参数，因此激活值需要做更细粒度的量化。

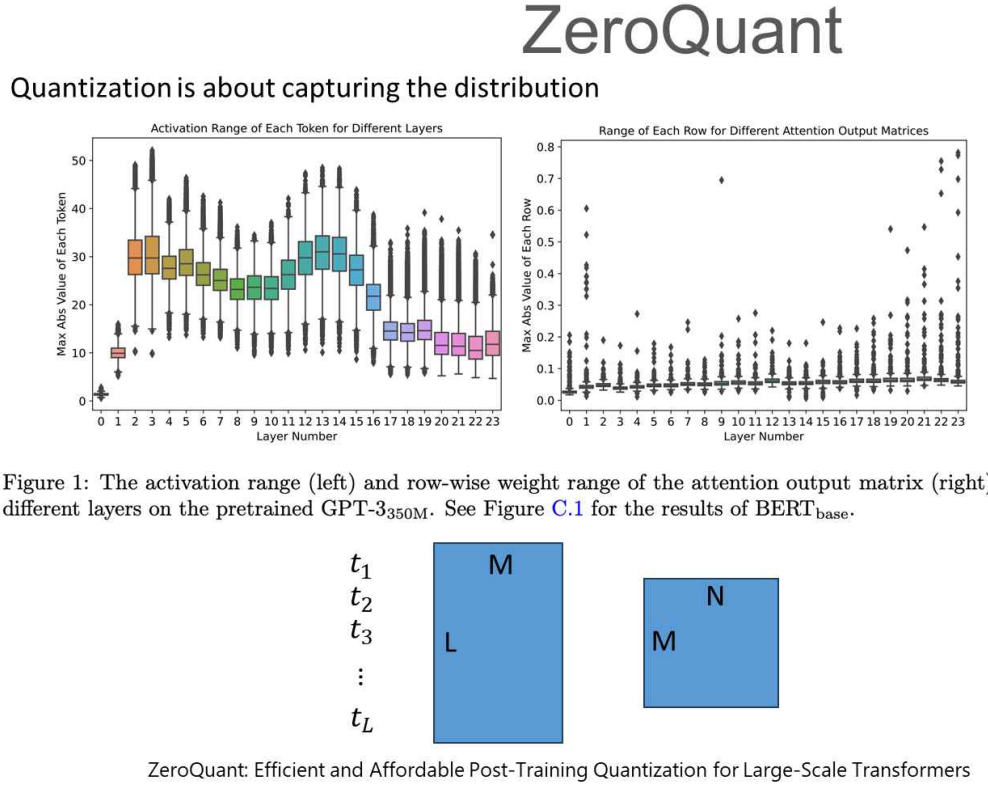
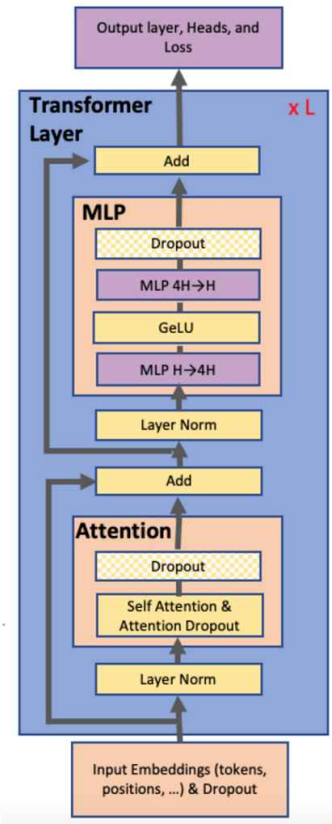


Figure 1: The activation range (left) and row-wise weight range of the attention output matrix (right) of different layers on the pretrained GPT-3_{350M}. See Figure C.1 for the results of $BERT_{base}$.

ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers



对于激活值X，采用token-wise粒度；对于权重Q，采用group-wise粒度。

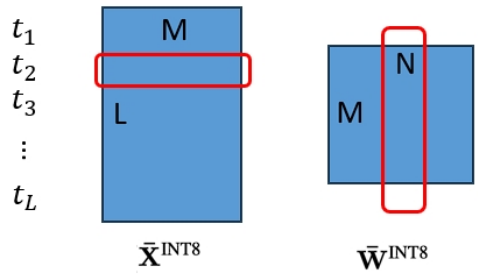
为什么激活值X采用横向量而权重Q采用列向量？

- 可以化简为低精度矩阵乘法再乘以量化常数矩阵的形式，GPU做低精度矩阵乘法更快。

量化后的精度损失怎么弥补？

- 采用层级蒸馏的方式，对每一层分别计算量化后和量化前的激活值的MSE损失。蒸馏是通过降低教师模型（在这里是未量化模型）和学生模型（在这里是量化后模型）之间的距离，使得学生模型更像教师模型。

1. Token wise quantization for activation
+ (channel wise) Group wise quantization for weights



$$\begin{matrix} q_{t_1} * t_1 \\ q_{t_2} * t_2 \\ q_{t_3} * t_3 \\ \vdots \quad\quad \vdots \\ q_{t_L} * t_L \end{matrix} \quad \Delta_{\mathbf{X}}^{\text{FP16}} \quad \bar{\mathbf{X}}^{\text{INT8}} \qquad \times \qquad \begin{matrix} q_{w_1} & q_{w_2} & & q_{w_N} & \Delta_{\mathbf{W}}^{\text{FP16}} \\ * & * & & * & \\ w_1 & w_2 & \cdots & w_N & \bar{\mathbf{W}}^{\text{INT8}} \end{matrix}$$

$$\mathbf{Y} = \text{diag}(\Delta_{\mathbf{X}}^{\text{FP16}}) \cdot (\bar{\mathbf{X}}^{\text{INT8}} \cdot \bar{\mathbf{W}}^{\text{INT8}}) \cdot \text{diag}(\Delta_{\mathbf{W}}^{\text{FP16}})$$

2. Layer wise distillation

$$\mathcal{L}_{LKD,k} = MSE\left(L_k \cdot L_{k-1} \cdot L_{k-2} \cdot \ldots \cdot L_1(\boldsymbol{X}) - \widehat{L}_k \cdot L_{k-1} \cdot L_{k-2} \cdot \ldots \cdot L_1(\boldsymbol{X})\right)$$

LLM.int8()

观察到激活值中存在一些离群值（大于某个超参数），绝对值显著大于其他值，严重影响量化结果。LLM.int8()采用混合精度分阶段量化方法，将包含离群值的特征提取出来保留16bit，其余权重和激活使用8bit量化。

So, 特殊情况特殊处理

$$\mathbf{C}_{f16} \approx \sum_{h \in O} \mathbf{X}_{f16}^h \mathbf{W}_{f16}^h + \mathbf{S}_{f16} \cdot \sum_{h \notin O} \mathbf{X}_{i8}^h \mathbf{W}_{i8}^h$$

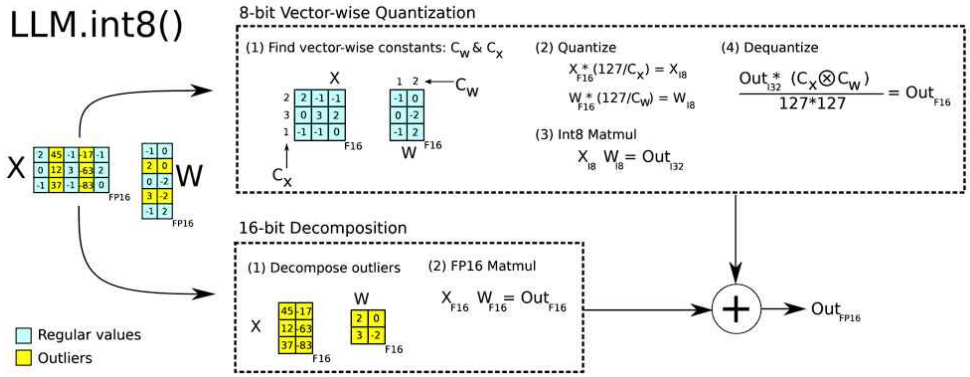


Figure 2: Schematic of LLM.int8(). Given 16-bit floating-point inputs $\mathbf{X}_{f16}$ and weights $\mathbf{W}_{f16}$, the features and weights are decomposed into sub-matrices of large magnitude features and other values. The outlier feature matrices are multiplied in 16-bit. All other values are multiplied in 8-bit. We perform 8-bit vector-wise multiplication by scaling by row and column-wise absolute maximum of $\mathbf{C}_x$ and $\mathbf{C}_w$ and then quantizing the outputs to Int8. The Int32 matrix multiplication outputs $\mathbf{Out}_{i32}$ are dequantization by the outer product of the normalization constants $\mathbf{C}_x \otimes \mathbf{C}_w$. Finally, both outlier and regular outputs are accumulated in 16-bit floating point outputs.

1. 从输入的隐含状态中，按列提取异常值 (离群特征，即大于某个阈值的值)。
2. 对离群特征进行 FP16 矩阵运算，对非离群特征进行量化，做 INT8 矩阵运算；
3. 反量化非离群值的矩阵乘结果，并与离群值矩阵乘结果相加，获得最终的 FP16 结果。

从下图可知，当参数在6B至6.7B之间时，离群值数量大幅度增加，受影响的层的数量也大幅增加。而离群值的数量随着perplexity的递减而指数级增加（后面介绍离群值）。大量异常值特征及其不对称分布破坏了 Int8 量化精度，由于量化前数据分布范围太大，导致量化后的大量正常值被量化为0，从而消除了信息。

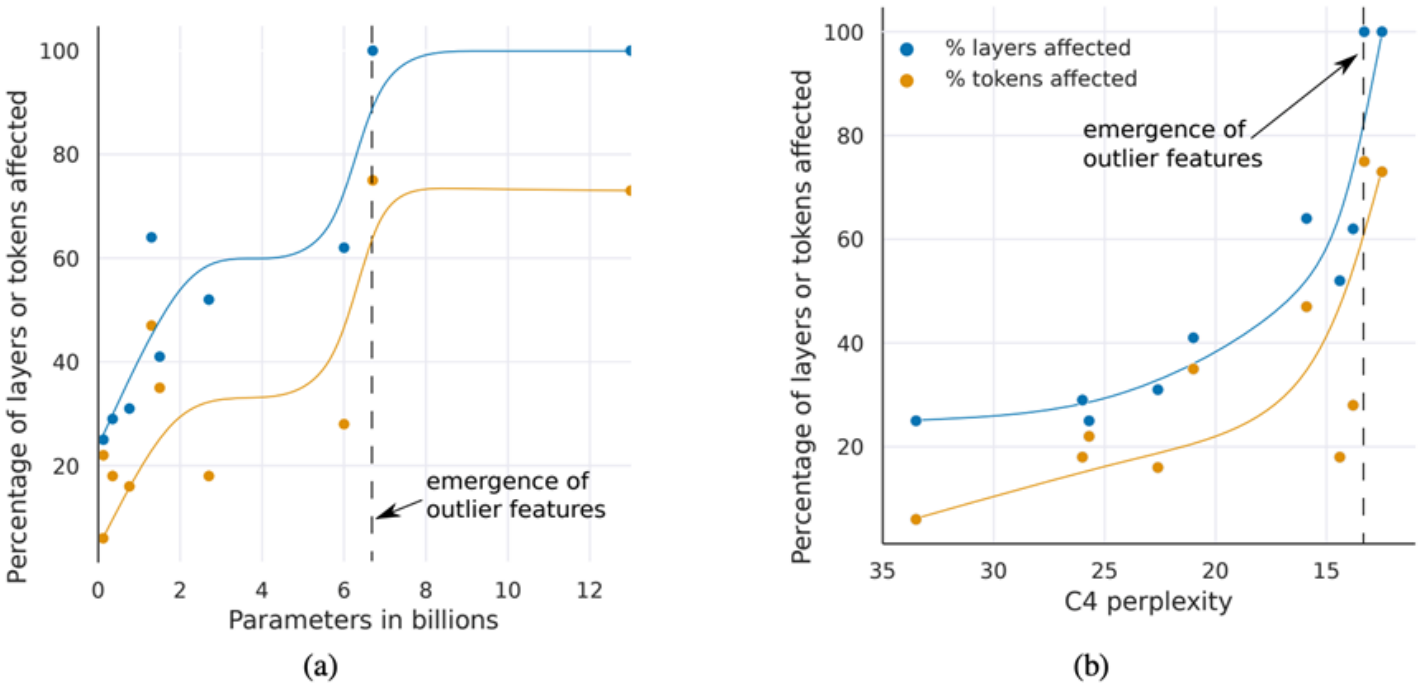


Figure 3: Percentage of layers and all sequence dimensions affected by large magnitude outlier features across the transformer by (a) model size or (b) C4 perplexity. Lines are B-spline interpolations of 4 and 9 linear segments for (a) and (b). Once the phase shift occurs, outliers are present in all layers and in about 75% of all sequence dimensions. While (a) suggest a sudden phase shift in parameter size, (b) suggests a gradual exponential phase shift as perplexity decreases. The stark shift in (a) co-occurs with the sudden degradation of performance in quantization methods.

论文总结了当模型参数超过6.7B时离群值的特征：

- 如果以幅度大于等于 6 的值为离群点，则在所有的模型中，至少 25% 的 transformer 层有离群点，至少 6% 的序列维度中有离群点。
- 离群值表现出了非常明显的非对称性（1-sided，也就是大部分为正或者大部分为负），包含离群点的特征维度的最大幅值往往是其他不包含离群点的特征维度的幅值的 3-20 倍，不包含离群值的特征维度的幅值范围通常是 [-3.5, 3.5] 之间。
- 模型越大，C4 验证困惑度（PPL）越低，出现的离群点越多，包含离群点的 transformer 层也越多，并且开始出现在所有序列维度中，大于等于 6.7B 的模型中随处可见。
- 尽管离群点仅占 0.1% 左右，但其对较大的 softmax 概率至关重要，如果移除离群值，softmax 概率的 top1 分数平均降低 20% 左右。
- 由于离群值在序列维度上具有不对称分布，因此会影响对称的 absmax 量化，而更偏向于非对称的 zero-point 量化。

			Outliers		Frequency			Top-1 softmax p	
Model	PPL↓	Params	Count	1-sided	Layers	SDims	Quartiles	w/ Outlier	No Outlier
GPT2	33.5	117M	1	1	25%	6%	(-8, -7, -6)	45%	19%
GPT2	26.0	345M	2	1	29%	18%	(6, 7, 8)	45%	19%
FSEQ	25.7	125M	2	2	25%	22%	(-40, -23, -11)	32%	24%
GPT2	22.6	762M	2	0	31%	16%	(-9, -6, 9)	41%	18%
GPT2	21.0	1.5B	2	1	41%	35%	(-11, -9, -7)	41%	25%
FSEQ	15.9	1.3B	4	3	64%	47%	(-33, -21, -11)	39%	15%
FSEQ	14.4	2.7B	5	5	52%	18%	(-25, -16, -9)	45%	13%
GPT-J	13.8	6.0B	6	6	62%	28%	(-21, -17, -14)	55%	10%
FSEQ	13.3	6.7B	6	6	100%	75%	(-44, -40, -35)	35%	13%
FSEQ	12.5	13B	7	6	100%	73%	(-63, -58, -45)	37%	16%

困惑度perplexity

参考文献：https://www.statlect.com/asymptotic-theory/empirical-distribution

信息量：

设随机变量X的概率密度函数为p(X)，对于 $X = x$ 这件事的信息量为：

$$I(x) = -log_2 p(x)$$

log底数为2表示用二进制数据对x进行编码时，所需要的bit数。当 $p(x) = 1$ 时，信息量为0，即不需要编码，也知道x一定会发生。当 $p(x) = 0.5$ 时，信息量为1，此时需要1个bit对其进行编码。

信息熵

随机变量X的平均信息量，定义为：

$$H(x) = -\sum_{x \in \mathcal{X}} p(x) log_2 p(x)$$

其中 $\mathcal{X}$ 时X的样本空间。

交叉熵和困惑度

设两个定义在相同样本空间上的概率分布 p 和 q，假设p为真实概率分布，q是对p的近似。使用q来衡量随机变量X的信息熵为：

$$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) log_2 q(x)$$

即p和q之间的交叉熵，定义概率分布的perplexity为：

$$PPL(q) = 2^{H(p,q)}$$

现实中，当真实概率分布未知时，采用经验分布替代真实概率分布。在样本集 $\{x_i\}_{i=1}^n$ 上，经验分布定义为：

$$\hat{F}_n(x) = \begin{cases} 0, & if \quad x < x_1 \\ \dfrac{i}{n}, & if \quad x_i \le x < x_{i+1} \\ 1, & if \quad x \ge x_n \end{cases}$$

其中 $\hat{F}_n(x)$ 为累积概率分布函数，根据 **Glivenko–Cantelli 定理**，当 $n \to \infty$ 时，经验分布函数收敛到真实分布函数。基于经验分布函数有 $P(X = x) = \dfrac{1}{n}$

$$PPL(q) = 2^{-\frac{1}{n}\sum_{x\in\mathcal{X}} log_2\, q(x)}$$

$$PPL(q) = 2^{-\frac{1}{n} log_2 \prod_{x\in\mathcal{X}} q(x)}$$

$$PPL(q) = \left(\prod_{x\in\mathcal{X}} q(x)\right)^{-\frac{1}{n}}$$

语言模型的困惑度

语言模型可以看作是给定一个token序列，根据已知的token来预测下一个token出现的概率。整个句子出现的概率为：

$$p(X_1, X_2, ..., X_m) = p(X_1)p(X_2|X_1)...p(X_m|X_1, ...X_{m-1})$$

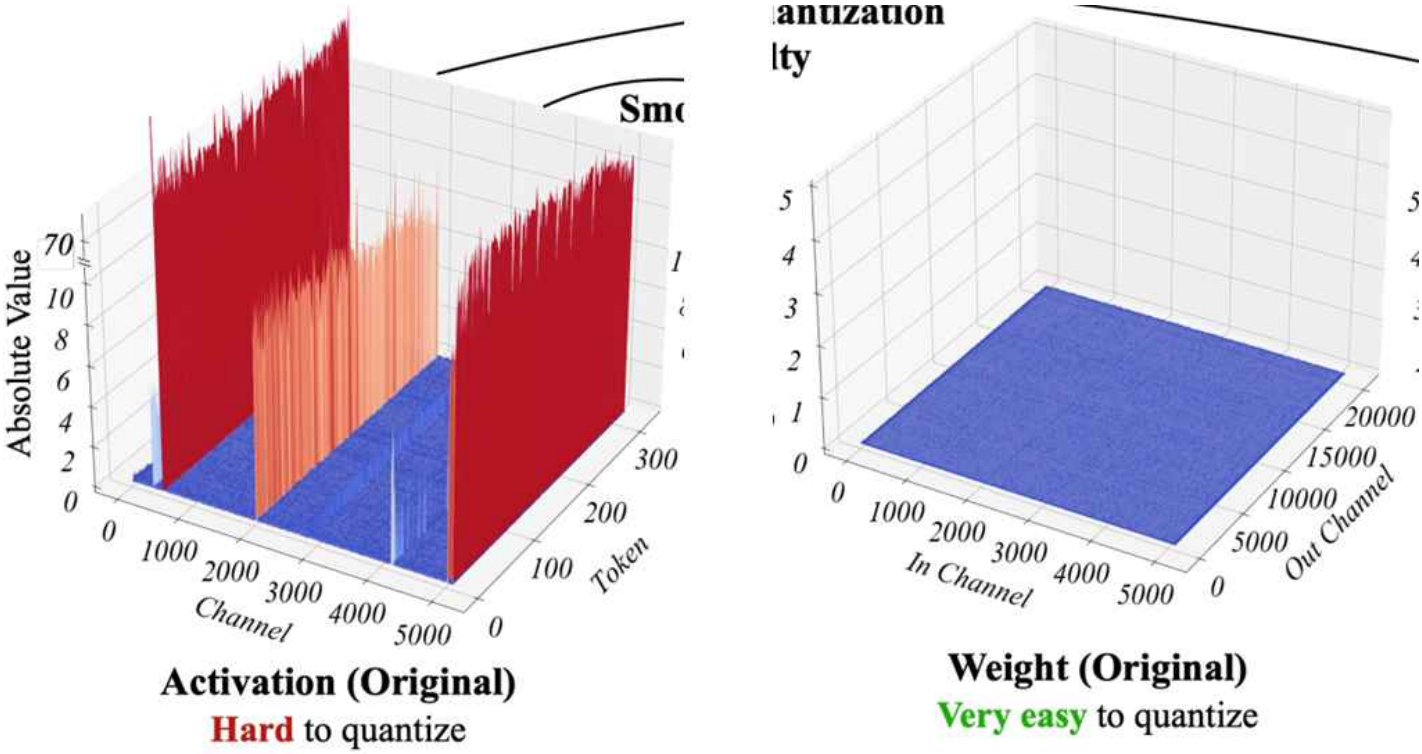
对于长度为m的序列，其困惑度为：

$$PPL = \left(\prod_{i=1}^{m} q(X_i|X_1, ..., X_{i-1})\right)^{-\frac{1}{n}}$$

PPL越小说明q越大，即我们期望的序列出现的概率就越高。一般模型越大、训练数据越多越丰富，复杂度越低。

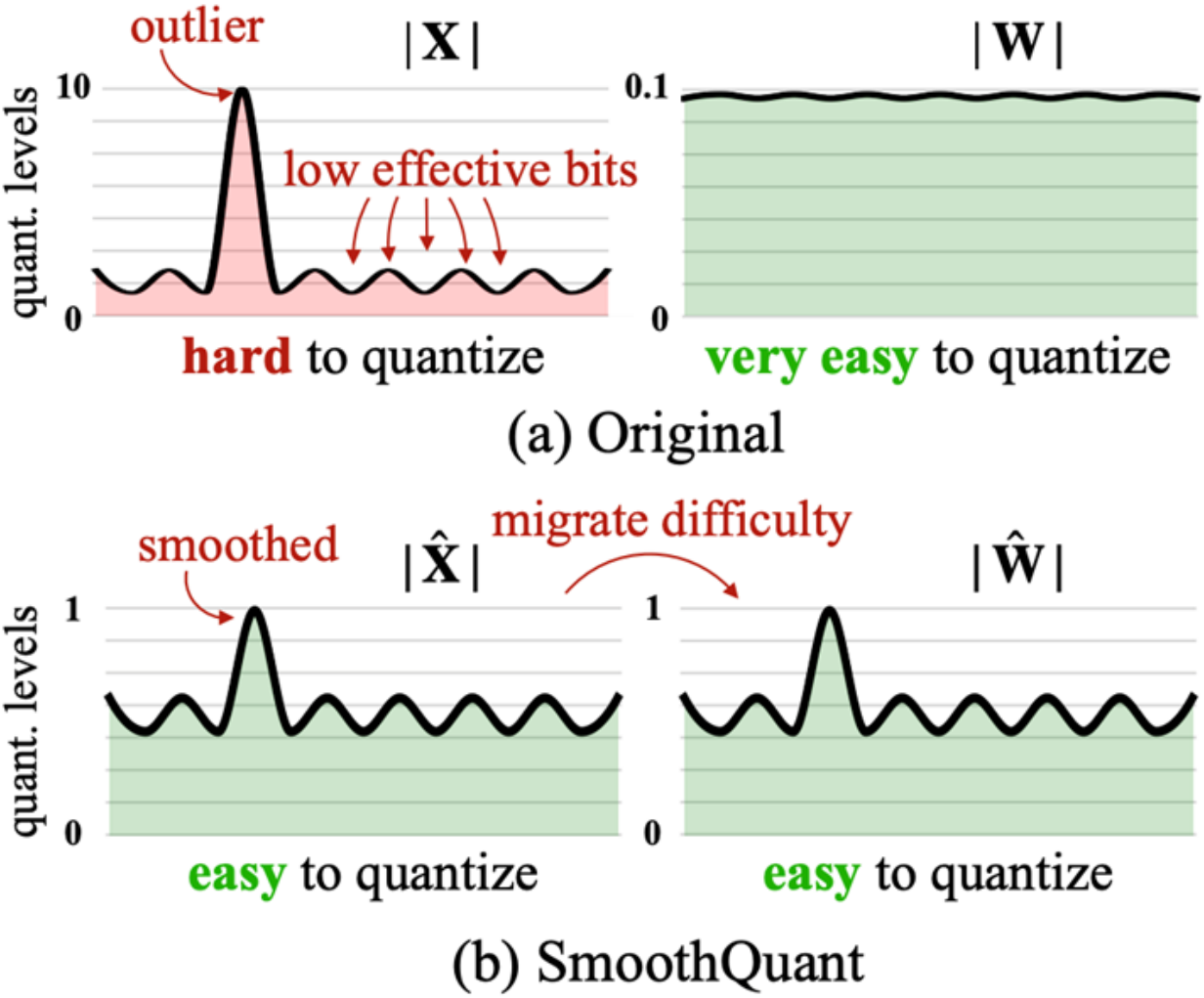
Smooth Quant

观察到激活值要比权重矩阵更难量化，且离群值总是出现在固定的channel上。图中channel表示一个token的hidden size，token表示sequence length。



由于要将权重和激活值矩阵相乘，可以对激活值除以一个缩放常数而对权重乘以一个缩放常数，从而使得权重和激活值都容易量化。

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



缩放值选择：首先计算激活值每一个channel的最大值 $|X_j|$ 和权重矩阵每个channel1的最大值 $|W_j|$ ，根据观察，将超参数 α 设为0.5时准确率最高。

更简单的方式是，在预训练数据集上提前统计缩放值，省去每次都要计算S的时间。

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

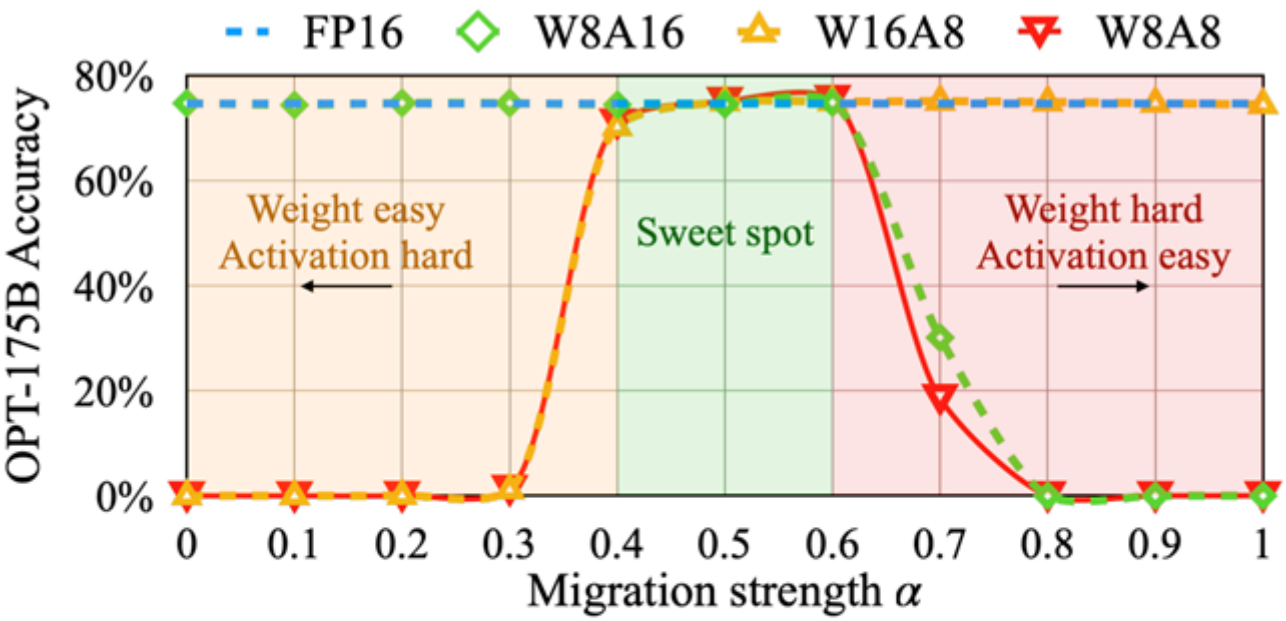
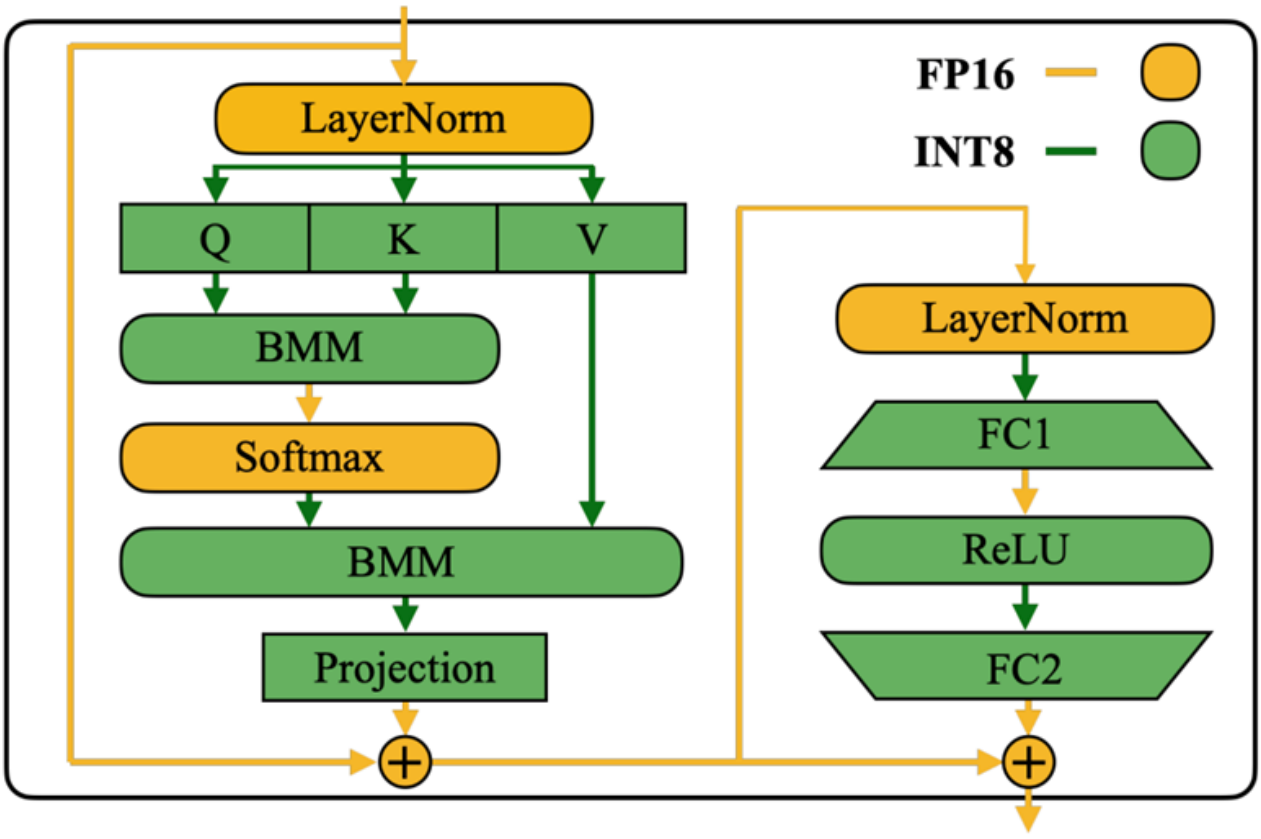


Figure 10: A suitable migration strength α (sweet spot) makes both activations and weights easy to quantize. If the α is too large, weights will be hard to quantize; if too small, activations will be hard to quantize.

对于计算量大的线性层和BMM都采用int8量化以加速计算，而Norm个激活层用FP16



Weight only（GPTQ）

只对权重矩阵进行量化，做矩阵乘时再反量化权重矩阵。采用Approximated Optimal Brain Quantization将量化问题转化成优化问题，最小化：

$$\text{argmin}_{\widehat{\mathbf{W}}} ||\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}||_2^2.$$

GPTQ需要在预训练数据集上训练最优化方程。

注释：GPTQ的反量化实际上增大计算量，但是GPU计算是冗余的，真正慢的是将参数从显存搬到寄存器的过程。

Weight only （AWQ）

相当于不对activation做量化的Smooth Quant。AWQ也与训练数据集上提前统计缩放值，但对预训练数据集的敏感性比GPTQ更弱，即预训练数据集不需要与下游任务接近。