# 论文1: AdaRound：Up or Down? Adaptive Rounding for Post-Training Quantization(ICML2020)

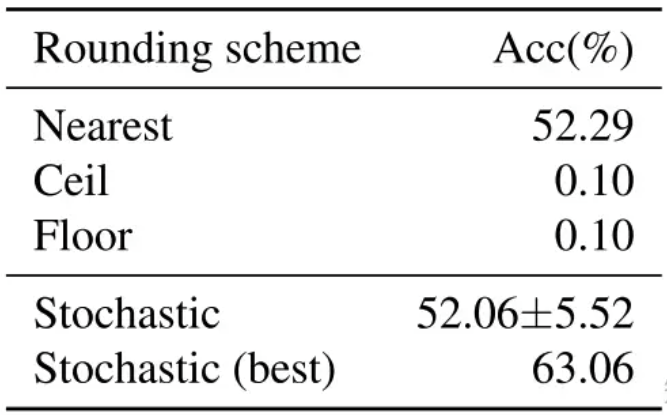
## 作用：

round函数的改进，和量化方法应该正交，但不知道为什么没有人联合量化方法使用，或许可以借鉴。

## 研究背景：

4bit的量化是很难的，其中一个原因是量化网格过大导致量化误差很大，之前的许多量化方法都可以归类为找出最好的scale，忽视了round函数带来的误差，本文证明了四舍五入的round方式并非量化误差最小的round函数，从而提出了一种新的round方法来减少量化误差。

## 研究问题：

* 实验发现传统的舍入方法四舍五入在PTQ中不是最优的，需要更精细的方法来减少量化导致的性能下降。
* 

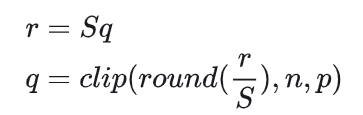
## 研究层次：

* 软件级
* PTQ，Weight-only量化。关注量化方法中的round函数。

## 研究方向：

PTQ量化+round函数优化 **强相关**

## 研究差异：

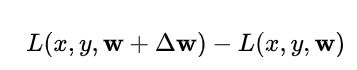
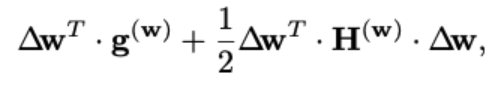
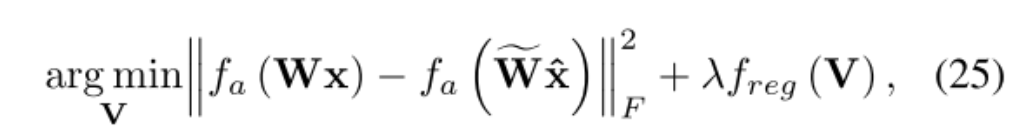
* 同：还是PTQ的方法。
* 异：
  + 过往的模型量化的研究核心点都是如何寻一个更优scale值使得量化后的模型精度更高。
  + 高通提出了一种名为AdaRound的后量化算法，该算法的关注点不再是如何为每个kernel/channel/tensor寻找更好的scale，而是**对量化过程中的round做改进。**
  + 

## 挑战：

* 量化到4bit的挑战巨大，传统的量化方法是想办法找一个更好的scale，但是在低bit的量化下，round函数带来的量化误差可能更大，传统的四舍五入在PTQ中不是最优的，需要更精细的方法减少round导致的量化误差。

## 文章提出的解决方案:

**AdaRound：**

* 首先文章对量化过程中的Round带来的误差做了理论和定量的分析，分析Round对于最终 loss的影响。于是问题转变为一个优化问题。即对模型进行量化后，损失函数的变化可以表示为：
* 
* 如果公式的结果很小，那么就可以认为 round 带来的差异不足以使模型的效果发生大的变化。公式泰勒展开后变为：
* 
* 假设模型训练的很好，则loss函数的1阶导数非常接近0，可以忽略。于是可以只关注后一项。
* 可以看出对于 来说，如果不是对角阵，则四舍五入不一定是最优的方法，因为有非平方项的存在。
* 一系列的数学推导后，作者得出了round函数导致量化损失的loss函数：
* 

可以直接使用梯度下降算法进行求解每层的round方向。

**总结**： 这篇文章作为 Adaround 的开篇，主要讲了为什么 round 会对模型的效果产生较大的影响，并从数学本质出发，建模了产生这一影响的原因，也由此推出了优化 round 的目标函数。

重读adaRound主要原因是这个方法看上去和最优化scale的量化方法是正交的，而且该方法在4bit量化中效果很好，三年前的文章没有在LLM中进行实验，因此感觉是可以重新被重视的文章。

# 论文2: LUT-GEMM: Quantized Matrix Multiplication based on LUTs for Efficient Inference in Large-Scale Gener(2022年6月挂arXiv)

## 作用：

LUT-GEMM 提出了一个高效的LLM推理框架，实现了低于4位的量化。主要创新点在GEMM算法的改动上，因为2，3位的GEMM乘法甚至不需要计算只需要查表就可以得到结果。一种空间换时间的想法，比较有意思。

## 研究背景：

现代自监督学习和Transformer架构的进步使自然语言处理取得了显著进展。但是，强大的NLP模型需要增大模型规模，这导致了计算和内存需求的显著增加。需要一种方法来降低LLM的推理延迟并保证精度。

## 研究问题:

LLM在单batch的GEMM上并行度不足。本文聚焦于大模型上GEMM的优化，来降低单batch的LLM推理延迟，并减轻了LLM对GEMM并行度的要求。

## 研究层次:

软件级，涉及到GPU优化，GEMM优化，PTQ， **强相关**

Weight-only量化

## 研究方向:

LLM+PTQ量化+GEMM优化

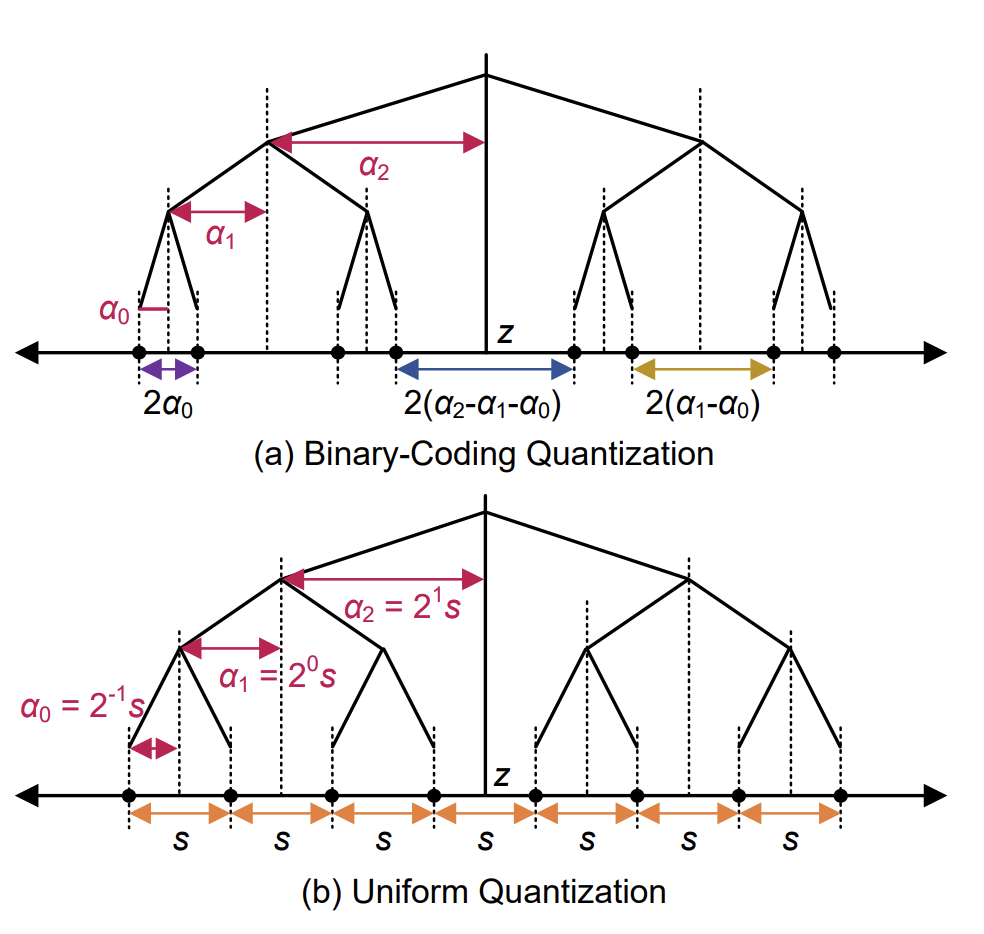
## 研究差异：

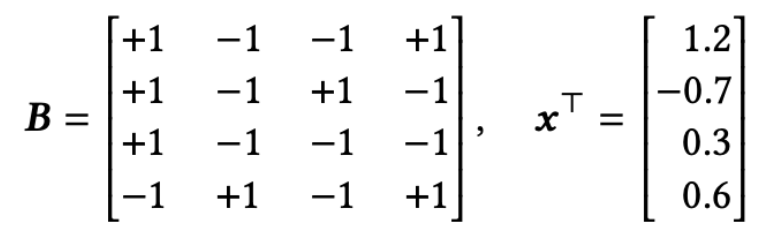
* 同：以往的研究也关注于LLM的GPU优化。但没有太多在GEMM内部做工作的文章。
* 异：LUT-GEMM特别关注于weight量化和LUT（查找表）方法，去除了量化和非量化方法之间的反量化过程。

## 挑战:

1.**非均匀量化方法的复杂性**：大多数非均匀量化方法涉及低并行性的操作，并且可能缺乏硬件支持。

2.**如何减少反量化过程的开销**：对于均匀和非均匀量化方法，反量化过程有很大的开销。LUT-GEMM通过实施高效的基于LUT的操作，无需反量化，减少了所需GPU的数量，并提高了推理性能。

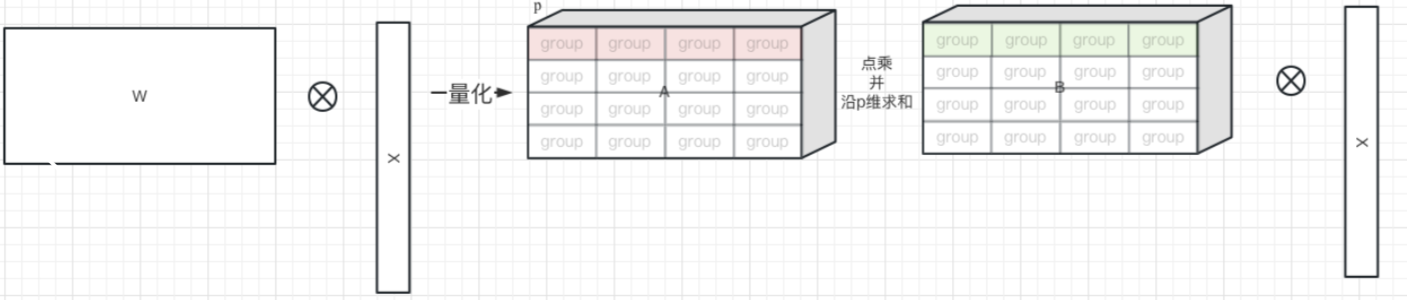


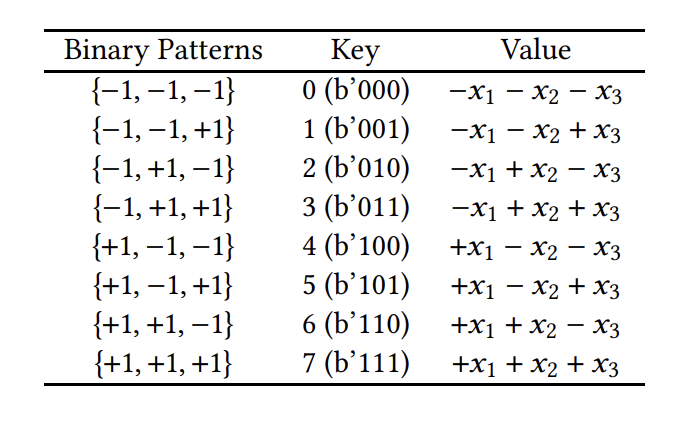
3.**GPU性能优化：**由于GPU位级访问内存很慢，为了避免位级内存访问并有效地执行，可以预先计算全精度激活和二进制模式weight的所有可能组合。即LUT-GEMM的由来。

## 文章提出的解决方案:

1.提出了一种可以偏置的BCQ格式，可以同时适应均匀或非均匀的量化方法。

2.linear层的计算为Y=WX。经过BCQ量化，可以将W分解成两个 张量A、B。A是scale，B是binary。A和B的点积，沿k维求和的结果就约等于W。因此省去了反量化的计算开销。



3.为了避免位级内存访问并有效地执行，可以预先计算全精度激活和二进制模式weight的所有可能组合。

# 论文3: SmoothQuant+: Accurate and Efficient 4-bit Post-Training WeightQuantization for LLM (23年12月挂arXiv)

## 作用：

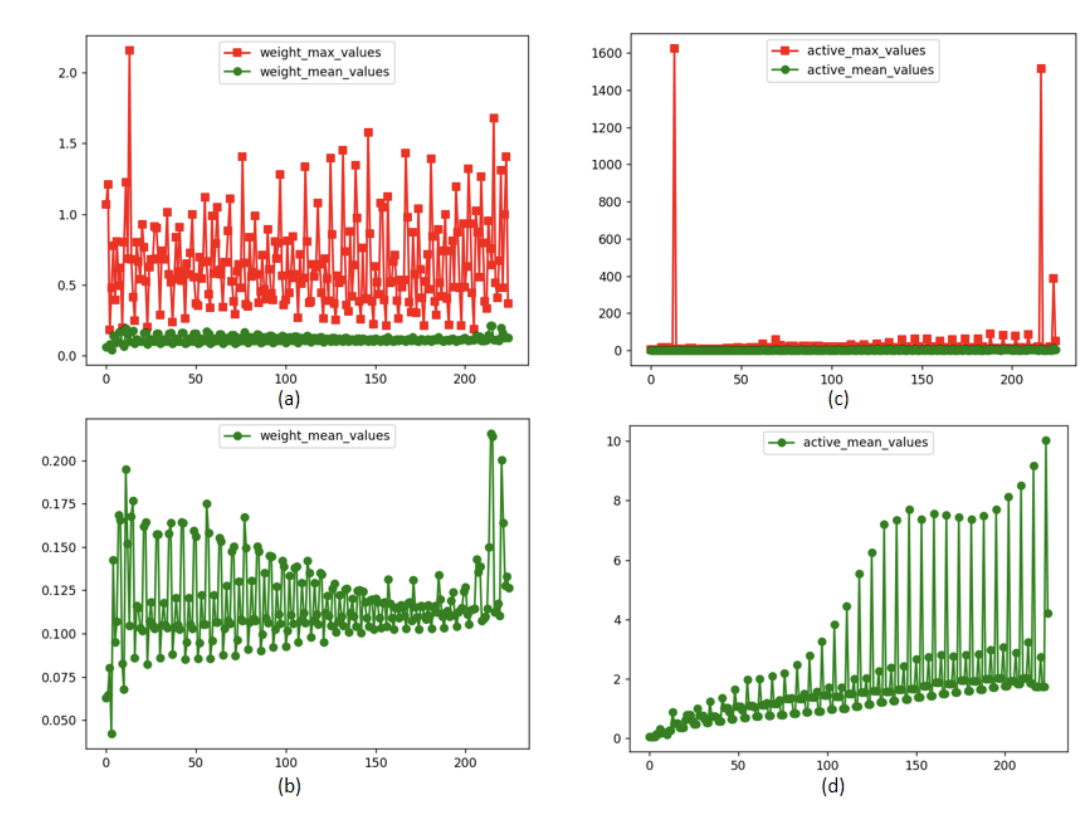
SmoothQuant的最新扩展工作，在算法的细节上进行了优化。达到了sota。

## 1 研究背景：

当 LLMs 的模型参数量超过 6.7B 的时候，激活中会成片的出现大幅值的离群点(outliers)，它们会导致量化误差增大，精度下降。这些激活上的离群点会出现在几乎所有的 token 上但是规律出现于某些固定的 channel 中。

## 2 研究问题:

权重的幅度分布均匀，按道理应该容易量化，可是无论是 GPTQ 还是 AWQ，OWQ，在进行 4-bit 权重量化时，都有较大的精度损失。本文对此问题进行了深入研究。



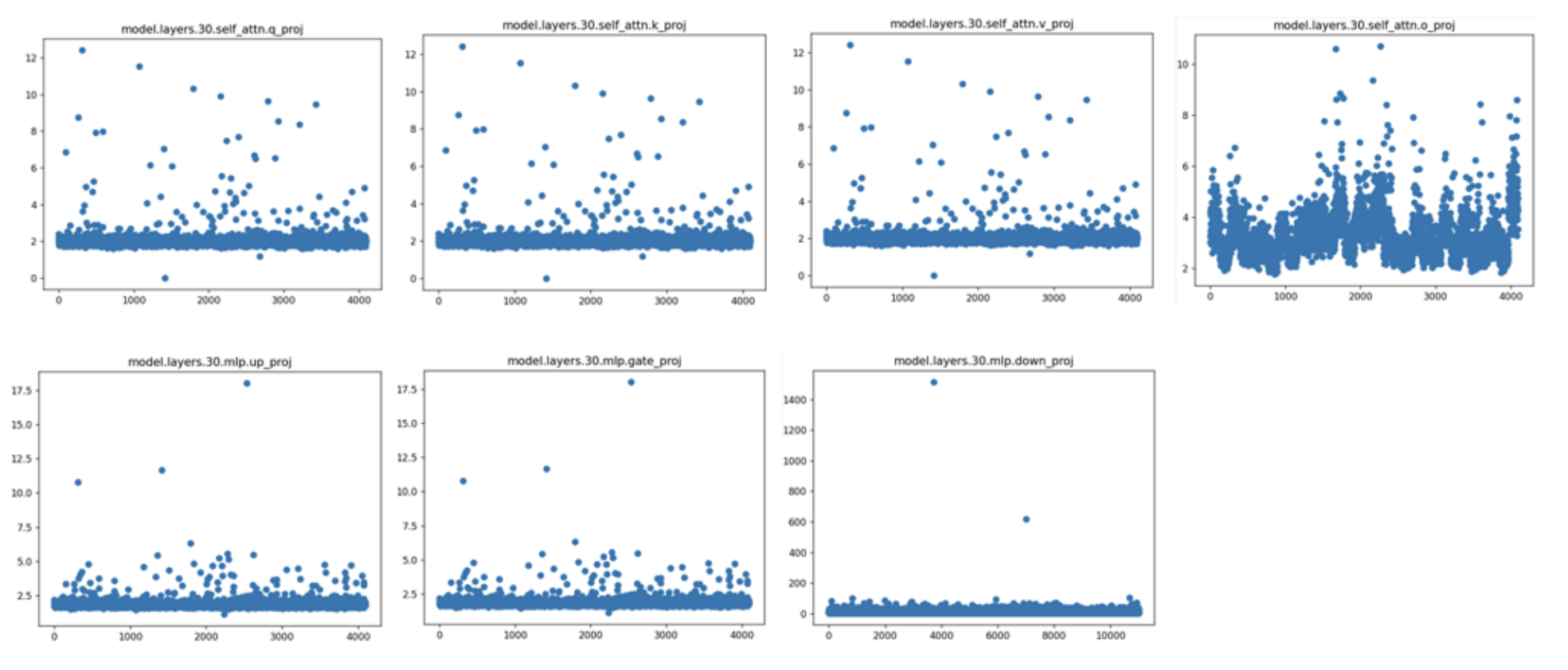
## 3 研究层次:

* 软件级,PTQ，Token-channel层级
* Weight-only量化

## 4 研究方向:

LLM+PTQ量化+异常值规律 **强相关**

## 5 研究差异：

* 同：和AWQ，smoothquant等相同，重点都在量化中处理异常值的方法。
* 异：深入探寻了channel中异常值的规律，观察到激活中的离群点总是出现在一小部分固定的 channels 中。如果一个 channel 有一个异常点，它持续的出现在所有 token 中。这些离群点的幅度往往是其他激活幅度的100倍。
* 

## 6 挑战:

1. 权重的幅度分布均匀，按道理应该容易量化，可是无论是 GPTQ 还是 AWQ，OWQ，在进行 4-bit 权重量化时，都有较大的精度损失。
2. 模型部署后的性能也十分重要，量化后如何实现模型的高效推理是很大的挑战。

## 7 文章提出的解决方案:

1.**激活权与重平滑：**SmoothQuant+ 采用了和 SmoothQuant 一样的对模型的激活和权重进行平滑的方式，但是出发点不同。

SmoothQuant+ 在对 LLMs 的 4-bit 量化中，先 smooth 模型，再量化权重。在这个过程中，对激活不做量化，但是仍然要 smooth 激活，主要是因为 LLMs 中激活的离群点造成了较大量化误差，平滑激活能减小量化误差。

SmoothQuant 在 smooth 模型后，需要根据不同的量化级别，对激活和权重都进行 8-bit 量化，其中权重只进行了per-tensor量化。

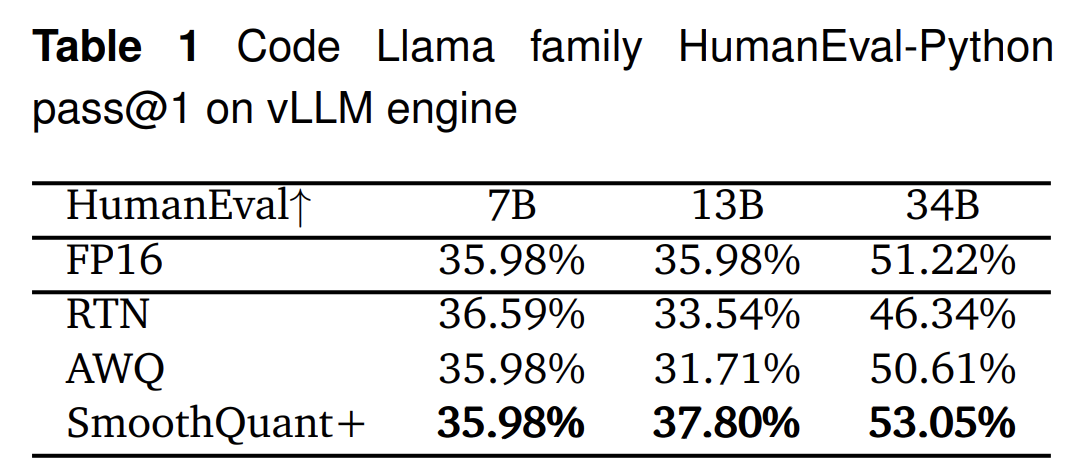
SmoothQuant+ 是对 SmoothQuant 的进一步扩展，只对权重进行的 4-bit group-wise 量化。AWQ 和 SmoothQuant+ 一样进行4-bit权重量化，也对激活和权重进行了平滑，但平滑的目的不同造成了平滑的具体实现不同。

AWQ 平滑激活的目的是保护权重中的 salient weights（重要的权重），在选择 importance factor 时，选用了平均值而非最大值。

2.**量化参数搜参：**AWQ 在搜索超参时是按层进行的，设定的目标函数是指定层的量化误差最小。AWQ 在计算这个量化误差时，使用了未量化时每一层的输入作为激励，没有考虑前面层的量化对后面层的影响从而造成误差累积问题。

作者认为这样按层搜索超参而不考虑误差累积是不合适的，这会造成最后模型量化精度的降低。对于这个问题smoothquant＋考虑了前一层量化误差的影响。

而且按线性层搜索，随着模型规模的增大，线性层增多，搜索的时间会显著增加。SmoothQuant+ 是对整个模型进行搜索，搜索过程迅速。

结果达到4bit量化sota：

# 论文4: OWQ: Lessons learned from activation outliers for weight quantization in large language models(23年6月挂arXiv)

## 作用：

权重重要性是有区别的，要去识别保护重要的权重。

## 研究背景：

LLM 的weigt量化至关重要，因为它可以减少内存占用，降低推理延迟。但LLM存在异常值（老生常谈)

## 研究问题：

主要比较对象是OPTQ，但文章其实和AWQ非常相似，在重点权重（OWQ称之为脆弱权重）识别上不太一样。

## 研究层次：

* 软件级
* PTQ，Weight-only量化。

## 研究方向：

LLM+PTQ量化+重点权重识别 **强相关**

## 研究差异：

* 同：和AWQ等文章一样，都是识别并保护重要权重。
* 异：重要权重的识别方法以及保护重要权重的手段不同。

## 挑战：

* LLM的异常值非常难量化，如何识别这些异常值并保护在LLM量化中非常重要。
* 从weight角度识别异常值的做法有局限性，应该从激活值的角度考虑。

## 文章提出的解决方案:

* **OWQ对weight“脆弱权重”的处理方法：**

1.**分析激活异常值的影响**：OWQ首先分析了激活异常值对量化引起的误差的放大作用。

2.**识别弱列**：OWQ将这些由激活异常值影响而变得对量化敏感的权重称为“弱列”。如果在量化过程中所有权重都被赋予相同的位宽，则与这些激活异常值相关的弱列可能导致显著的输出扰动，从而导致重大的量化误差

3.OWQ设计了一个算法，在OPTQ的基础上增加了额外的处理流程，对这些弱列应用FP16存储。

4.原始权重的“弱列”被填充为0。

* OWQ不使用A\*W的中间结果的最大值来确定“脆弱权重”，而是使用黑塞矩阵确定权重量化的敏感性从而确定“脆弱权重”。

# 论文5: Integer or Floating Point? New Outlooks for Low-Bit Quantization on Large Language Models (2023年7月挂arXiv)

## 作用：

对当前量化中INT和FP在LLM中的影响和量化误差做了实验与总结。并提出了MoFQ量化方法，对INT和FP作用的分析值得一看，但提出的混合量化方法好像并不实用。

## 研究背景：

高效部署大语言模型（LLM，Large Language Model）需要低比特量化来减少模型大小、降低推理成本。在以往的工作中，研究人员广泛使用低比特整型数（例如INT8/INT4）进行模型量化，但随着低比特浮点（如FP8/FP4）得到了越来越多计算硬件和系统支持（如FP8在NV H100上或者支持，FP4也收到了广泛关注），一个问题自然而然出现了：**INT和FP哪个在LLM量化中更有优势？**

## 研究问题:

GPTQ这类工作的量化数制都是INT，现有工作使用INT数制进行LLM量化的原因有两个：

1）低比特INT量化的效果被证明在小模型上较好；

2）之前的硬件不支持其他低比特数制计算。GPTQ这类工作的量化过程往往需要特别定制的优化。以GPTQ为例，量化过程中会对W进行分块量化，并使用基于hessian矩阵的方法调整未量化的W，从而弥补量化误差。这个过程需要依赖calibration，而且计算开销也很大（A100上，用GPTQ的LLaMA-65B量化时长在1小时以上）。

当前，另一种低比特数制FP8的计算单元已经出现在最新的硬件上。例如，NVIDIA的H100 GPU为FP8和INT8操作提供相同的峰值性能。此外，其他CPU和GPU供应商，如Intel、AMD和Qualcomm，正在积极将FP8计算硬件整合到他们的硬件中。

受到这个硬件趋势的启发，作者提出了一个有趣的问题：**考虑到具有相同位宽的INT和FP格式可以表示相同数量的离散值（例如，INT8和FP8都可以表示2^8 = 256个值），但值分布不同，它们对模型的高效推理与量化误差有什么不同的影响？**

## 研究层次:

软硬结合，关注FP和INT的量化误差与硬件开销。同时做了LLM的量化算法。

## 研究方向:

数据类型的量化误差，硬件开销。 **强相关**

## 研究差异：

* 同：关注LLM量化。
* 异：先进行了数据类型的比较，通过这个观察，提出了MoFQ算法来进行量化。

## 挑战:

1.当前的LLMs的量化方法大都选用INT量化，但并没有研究证实在LLM中int格式是最优的。

2.

## 文章提出的解决方案:

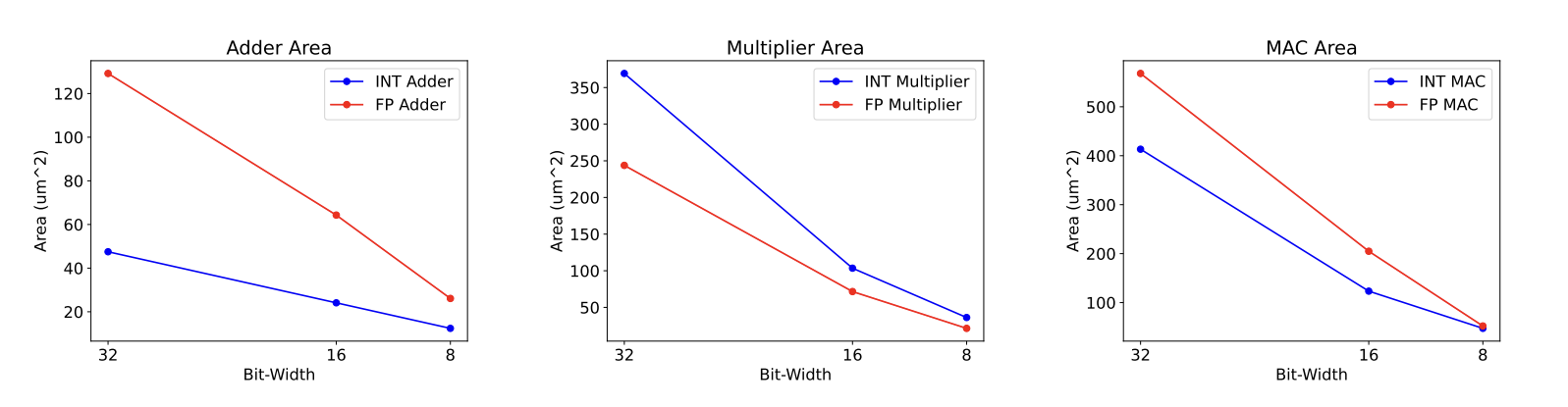
**文章对数据类型的观察：**

**1）FP8与INT8的硬件开销接近，因此硬件厂商有机会在接近的面积提供算力一致的FP8/INT8计算硬件。**

作者首先比较不同位宽的INT和FP运算符（包括加法器、乘法器和乘累加（MAC）单元）的硬件成本。

作者利用Synopsys Design Compiler测量了TSMC的7nm工艺库中每种运算符的面积开销。对于FP8运算符，作者展示了E5M2格式的结果，E4M3的结果与之类似。可见，FP乘法器所需面积小于INT乘法器，而FP加法器所需面积大于INT加法器。

对于MAC单元，它作为DNN中矩阵乘法的基本构建块，FP运算通常比INT运算需要更多的面积。然而，随着位宽的减小，这种差异逐渐缩小。当位宽来到8位，FP和INT MAC单元的面积要求几乎相同。这一观察表明，INT8和FP8具有类似的硬件开销。



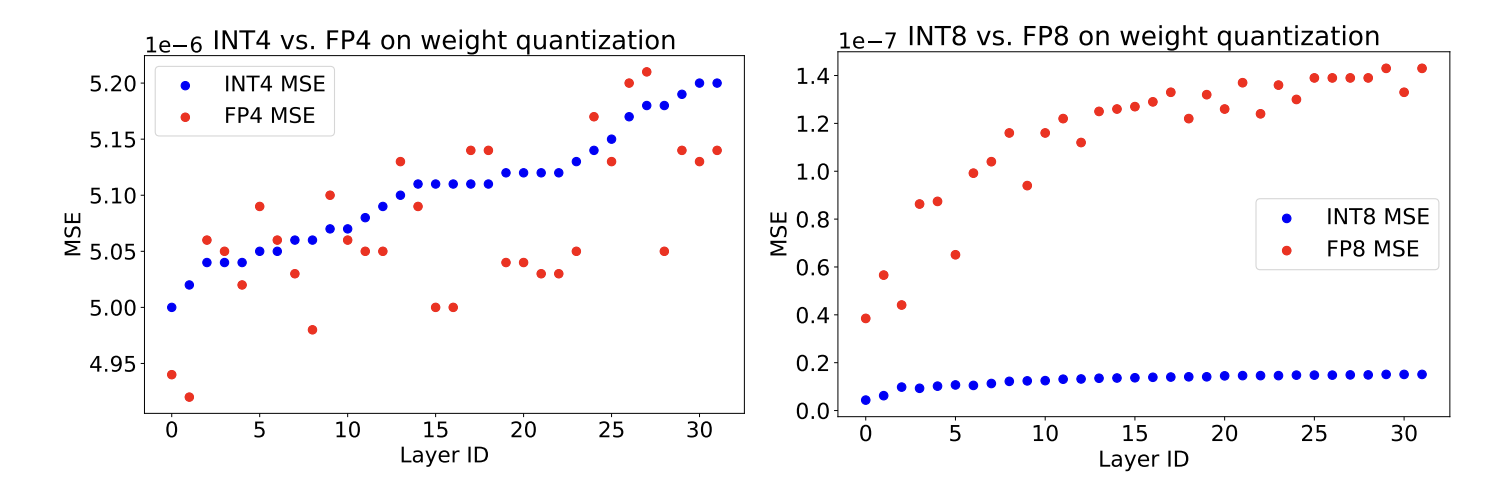
**2）LLM中不同tensor，倾向的量化类型不同。**

作者以LLaMA-65B为例，比较了不同线性层在不同类型下的量化误差，其中FP类型使用的是FP8-E4M3和FP4-E2M1。

LLM中的线性层可以表示为，其中是权重tensor，在推理过程中不会变化，作者称之为静态tensor，和分别是输入和输出的activation tensor，这种tensor中的元素会根据模型输入的变化而变化，作者称之为动态tensor。

作者分别对静态和动态tensor进行了8bit和4bit的量化误差分析，作者采用了量化前后tensor的MSE （mean squared error）作为衡量tensor量化效果的指标，该指标越小说明量化效果越好。

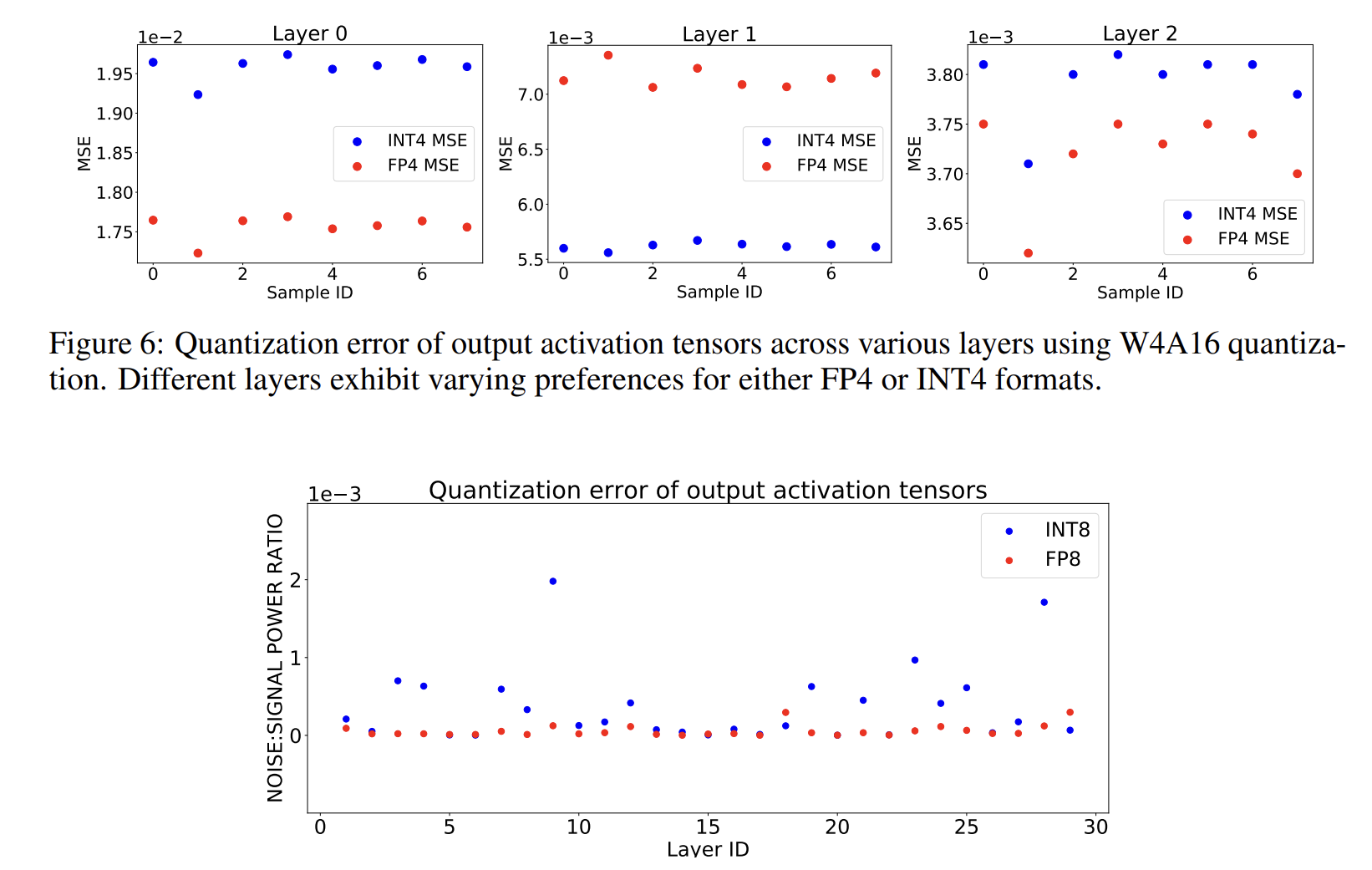
**weight量化观察：**



从图中可以看出，当量化位宽是4比特，有些层使用INT4的量化误差更小，有些则是FP4更小。但对于8比特量化来说，在所有层上都是INT8的量化误差更低。

**activation量化观察：**

每层输入tensor的量化误差，最终会体现在输出tensor的误差上。文章发现，不管量化任务是W4A16还是 W8A8，输出tensor的量化误差都遵循相同规律：有些层使用INT的量化误差更小，有些则是FP更小。

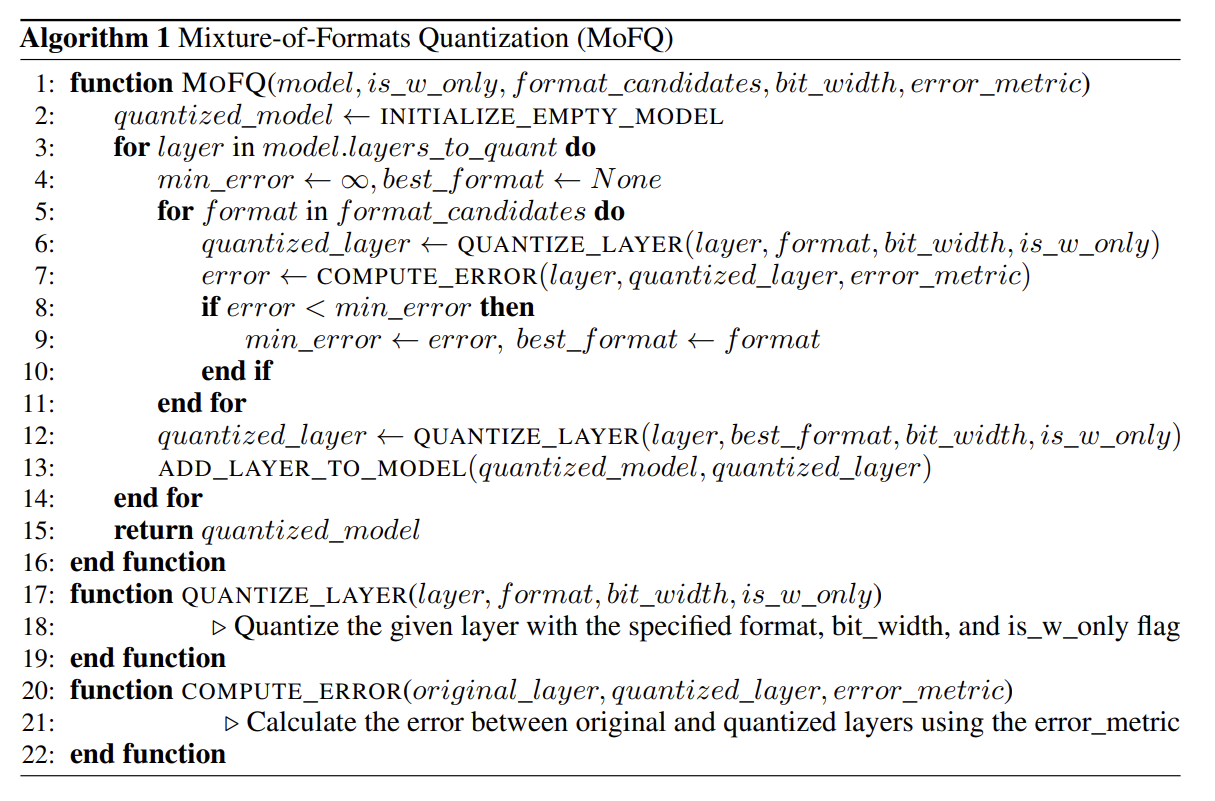


由此，得出结论：

由于LLM中不同层的W/A数据分布不尽相同，并不是**所有层用INT来量化，都能达到最低的量化误差**。此外，由于FP格式近零密集、远零稀疏的分布特点，天然和W/A tensor的分布更接近，因此作者发现LLM中的相当一部分层**使用FP量化后的误差比INT更低**。由此，作者认为：

应该为模型的每一层选择更适合的量化数制，才能最大化降低模型整体的量化精度损失。

**量化方法：混合数制量化MoFQ（Mixture of Formats Quantization）:**



MoFQ算法的输入：

1）待量化模型“model”；

2）W-only/WA量化选择“is\_w\_only”；

3）数制池“format\_candidates”，是每一层待量化参数能选择的数制，本工作用的是INT和FP数制；

4）量化位宽“bit\_width”，整个模型的全局设置；

5）误差评估方法“error\_metric”。算法运行时，会根据误差评估方法逐层选择合适的数制。这样一来，每层W或者WA使用的数制一致，但不同层的数制会有所不同。在量化后的模型部署时，这种简单、粗粒度的量化不会对模型推理速度造成影响。

**总结：**文章对int和FP的分析值得一看，提出的结论比较有启发，但提出的量化方法使用了混合INT和FP的精度，感觉硬件效率并不高，方法也不新颖，不是很值得看。