量化（2）：SmoothQuant

相关阅读：

量化（1）：概念介绍

本博客学习在 llm 推理中广泛使用的 SmoothQuant，并在 llama 以及 MoE 架构模型上实践

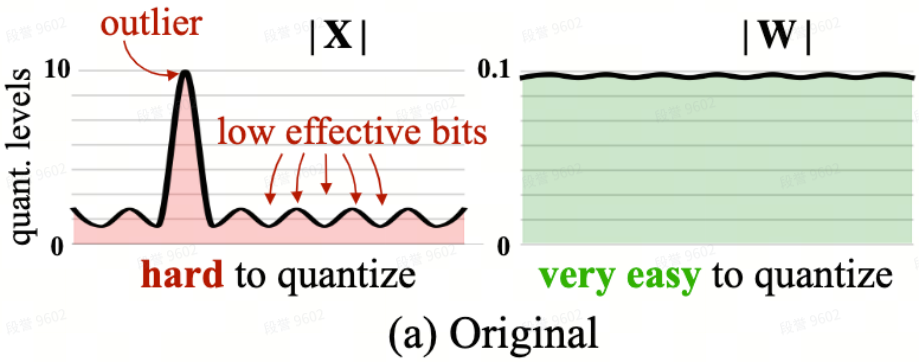
论文解读

SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models

1. 问题分析

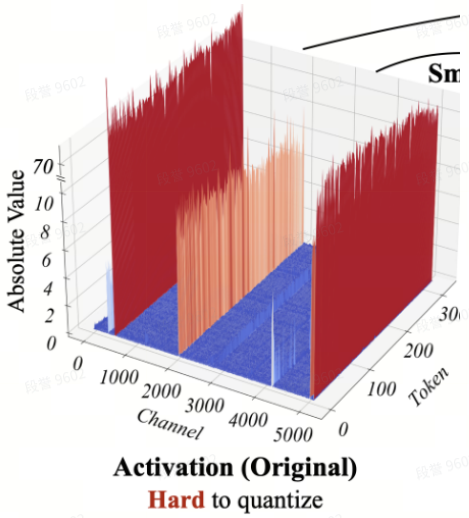
正如 int8 量化（1） 中所讲述， outliers 会大幅降低量化精度，使得绝大部份数据的有效比特位很低。

作者通过分析 llm 推理时， 激活值和权重的分布发现，激活值往往会有一些 outliers，很难量化，而权重的分布比较平滑，非常容易量化。



(a) Original

那么这些 outlier 到底分布在激活值的哪些地方呢，作者通过分析发现，这些 outlier 基本上都分布在 activation 的channel 纬度：

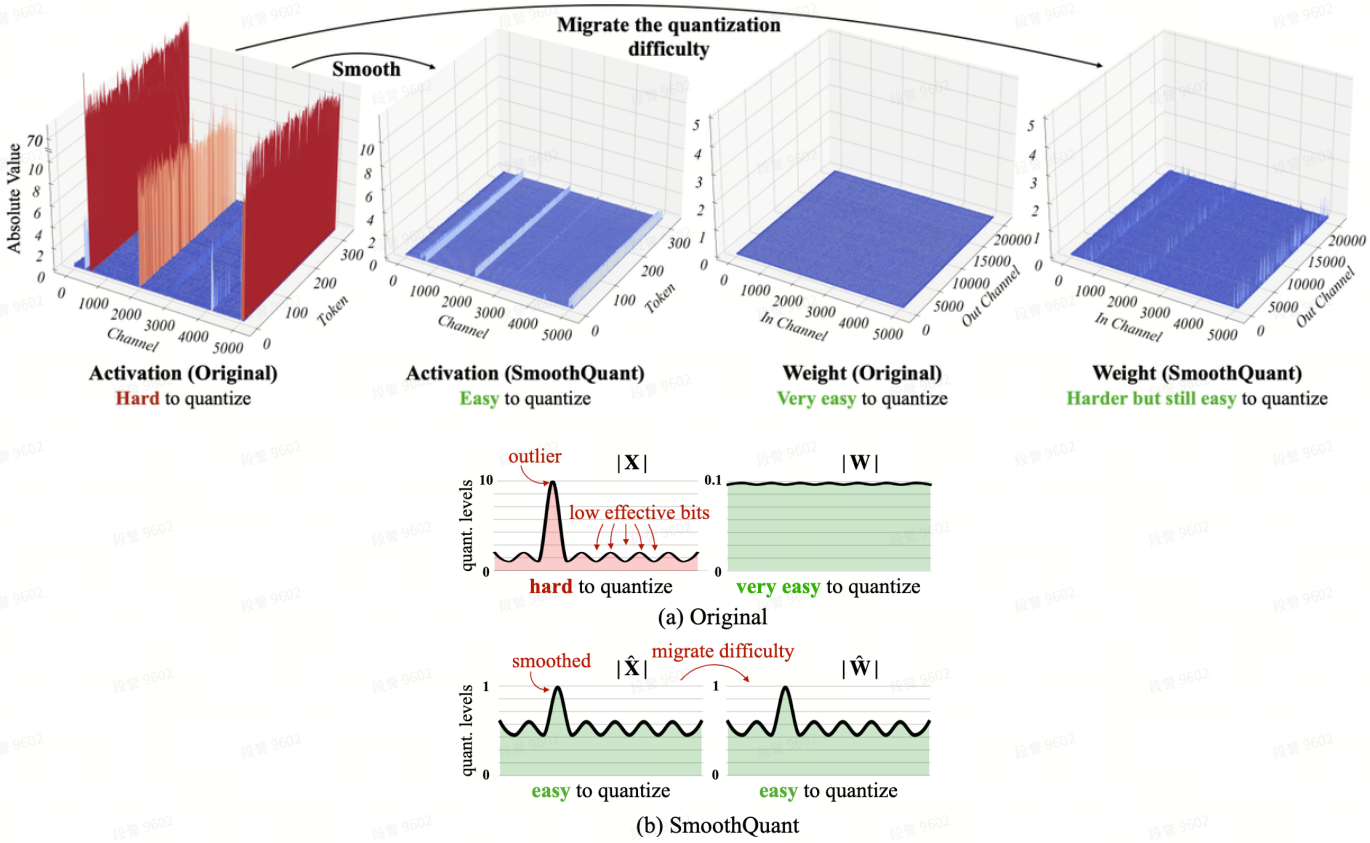


Activation (Original)
Hard to quantize

2. 解决方案

activations 难量化，weights容易量化，那能不能把难度转移一点 `migrate the quantization difficulty`，

如下图:



转移之后，activations 和 weights 都容易量化了，而且矩阵是线性运算，这种转移在数学上是**完全等价**的。

3. 具体措施

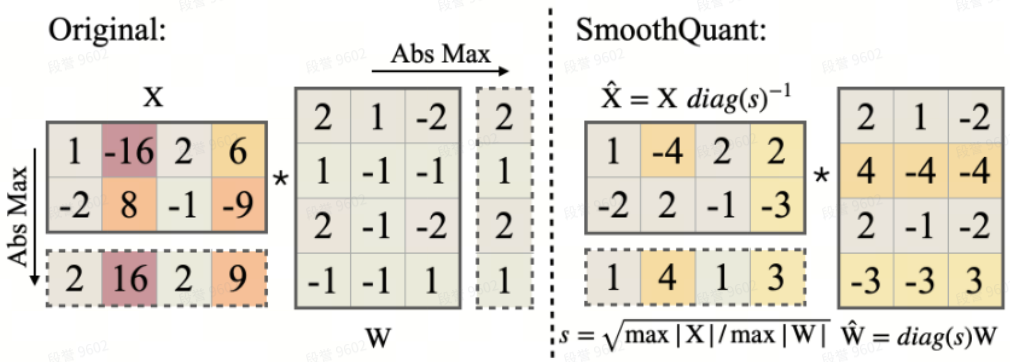


Figure 5: Main idea of SmoothQuant when α is 0.5. The smoothing factor s is obtained on calibration samples and the entire transformation is performed offline. At runtime, the activations are smooth without scaling.

这里解读下这个图：

首先在 channel 维度上收集 X 和 W 的绝对值的最大值，得到 `[2,16,2,9]` 和 `[2,1,2,1]`，然后根据公式，计算转移的 scale：

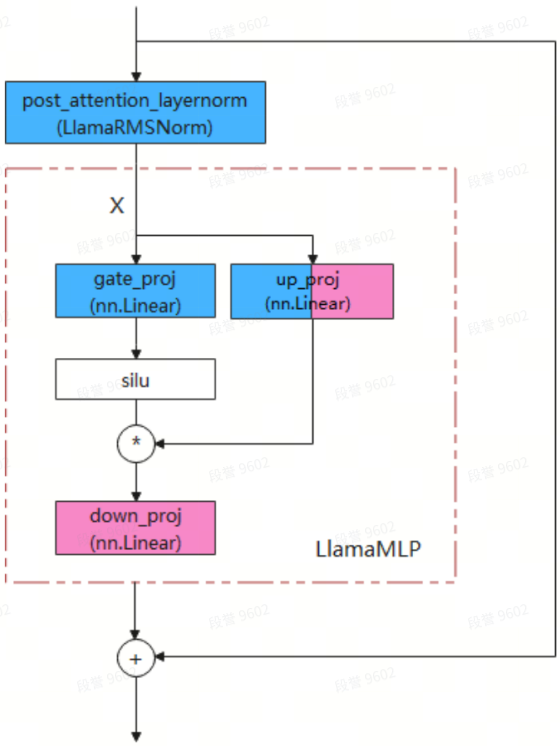
$$s = \sqrt{max|X|/max|W|}$$

得到 `s = [1,4,1,3]`

最后，X 每一行点除这个 s， W 每一列点除这个s，搞定！

4. 推理时的算子融合

The same color block indicates using the same smoothing factor



> where gate_proj and up_proj have the same input X which is the output of LlamaRMSNorm named post_attention_layernorm. Dividing X by the smoothing factor can be integrated into post_attention_layernorm’ s parameters. To keep the mathematical equivalence, the weights of
>
> gate_proj and up_proj are multiplied by the smoothing factor, respectively. For down_proj, only up_proj is a linear operation in front of it, so the operation of dividing its input by the smooth-
>
> ing factor is fused into the weights of up_proj, and the weights of down_proj are multiplied by the smoothing factor to maintain the mathematical equivalence of the model.

Llama 模型的量化

对于 llama 架构的模型，主要有以下 8 个 **gemm：**

Attenton 中 `q_proj, k_proj, v_porj, o_proj`

Mlp 中 `gate_proj, up_proj, down_proj`

> 注：`q_proj, k_proj, v_porj, gate_proj, up_proj` 这五个 gemm 前面有 LayerNorm 算子，可以把对 X 点除 s 这个操作，转移到 layernorm 的weight 中，**ln.weight** 点除 s

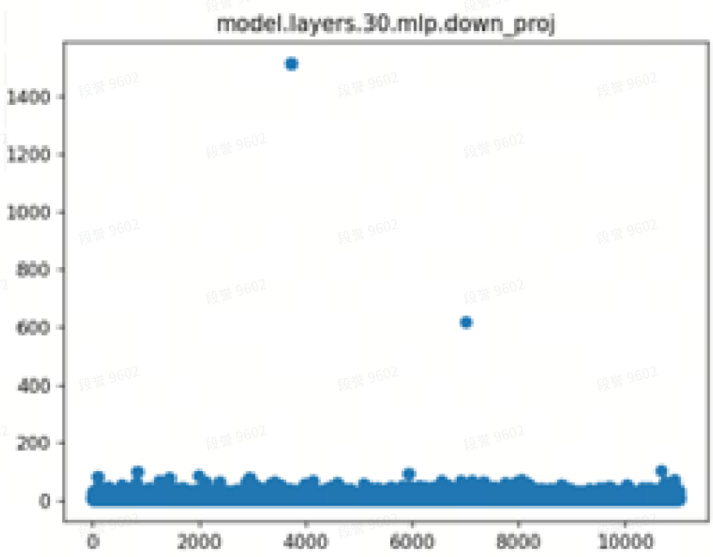
量化方案：**per-tensor** （最大化加速），模型： **Skywork-13B-chat**

数据集： **lambada**， 测试用1000条输入

选择量化的gemm / 数据集上的正确率	fp16	SmoothQuant W8A8	Naive W8A8
`q_proj, k_proj, v_porj` `gate_proj, up_proj`	0.876	0.87	0.863
`q_proj, k_proj, v_porj, o_proj` `gate_proj, up_proj`	0.876	0.866	0.858
`q_proj, k_proj, v_porj, o_proj` `gate_proj, up_proj, down_proj`	0.876	0.832	0.646

可以发现，mlp 中最后一个 gemm (`down_proj`)是影响 Naive W8A8 精度的主要原因，使用 SmoothQuant 后，精度有明显提升。

最后一个 mlp 对精度的影响为何如此大：本质原因还是 outliers 值太大，下图是某篇论文中作者的研究。



在个别 channel 上，outliers 是其他均值的 100 倍+。

其他一些论文对这个也有一些解释：

> This may be because the high sparsity of features after the non-linear layer leads to unstable gradients when applying learnable equivalent transformations.

重要是中间的非线形层 silu 的影响

8x7B-MoE 模型量化

该 MoE 的架构属于 mixtral-moe，DecoderLayer 有32 层，每一层又架构如下：

```
attention 参数量:  4096*4096 + 4096*1024*2 + 4096*4096
q_proj: 4096*4096
k_v_proj: 4096*1024*2,
O_proj: 4096*4096


mlp: 4096*14336*3
gate_porj :  4096*14336
up_porj :  4096*14336
down_proj:  4096*14336

每一层两个专家:
 (4096*4096 + 4096*1024*2 + 4096*4096 + 4096*14336*3 * 2) *32
```

```
MixtralForCausalLM(
  (model): MixtralModel(
    (embed_tokens): Embedding(65536, 4096, padding_idx=0)
    (layers): ModuleList(
      (0-31): 32 x MixtralDecoderLayer(
        (self_attn): MixtralSdpaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): MixtralRotaryEmbedding()
        )
        (block_sparse_moe): MixtralSparseMoeBlock(
          (gate): Linear(in_features=4096, out_features=8, bias=False)
          (experts): ModuleList(
            (0-7): 8 x MixtralBLockSparseTop2MLP(
              (w1): Linear(in_features=4096, out_features=14336, bias=False)
              (w2): Linear(in_features=14336, out_features=4096, bias=False)
```

```
                (w3): Linear(in_features=4096, out_features=14336, bias=False)
                (act_fn): SiLU()
              )
            )
          )
          (input_layernorm): MixtralRMSNorm()
          (post_attention_layernorm): MixtralRMSNorm()
        )
      )
      (norm): MixtralRMSNorm()
    )
    (lm_head): Linear(in_features=4096, out_features=65536, bias=False)
  )
```

数据集： **lambada**， 测试用1000条输入

Weight only 量化

moe 的 group gemm 目前不支持 int8 * int8，量化方案使用 **weight only**，

1. 只对 experts 中的 gemm 做量化，weight only **int8**

量化的gemm / 数据集上的正确率	fp16	Naive W8A16
`gate_proj, up_proj, down_proj`	0.817	0.816

2. 只对 experts 中的 gemm 做量化，weight only **int4**

量化的gemm / 数据集上的正确率	fp16	Naive W4A16	Smooth + W4A16
`gate_proj, up_proj, down_proj`	0.817	0.791	0.802

3. 在 2 的基础上，对 qkv 的gemm，做 W8A8 量化（per_tensor），这样可以提高生成速度，并进一步节省显存

量化的gemm / 数据集上的正确率	fp16	Naive W4A16	Smooth + W4A16	Smooth + experts_W4A16 + qkv_W8A8
`qkv_proj: w8A8`	0.817	0.791	0.802	0.794

`gate_proj, up_proj, down_proj:experts`				

虽然，weight 是平滑的、容易量化的，但是误差也受激活值的影响。

> 从公式可以看出，量化损失不仅和权重有关，和激活也有关，他们是相乘的关系。我们认为权重量化的误差被激活的离群点放大了，通过平滑激活离群点，同时对应调整权重，能够大大降低量化误差。

The quantization loss is as follows:

$$\mathrm{E} = \left\| \mathbf{X}\mathbf{W}^{\mathrm{FP16}} - \mathbf{X}\hat{\mathbf{W}}^{\mathrm{FP16}} \right\|_2^2 \qquad (4)$$

因此，与 直接 weight only 量化相比，**量化前，先 Smooth LLMs**，可以减少误差。实验的结果和论文： SmoothQuant+: Accurate and Efficient 4-bit Post-Training WeightQuantization for LLM 相呼应。

SmoothQuant 其他系列论文解读

1. SmoothQuant+: Accurate and Efficient 4-bit Post-Training WeightQuantization for LLM

正如 MoE 量化中提到的，SmoothQuant+ 的思想就是，在进行 W4 量化时，先 smooth 一下模型。与直接 W4 量化相比，这种方式可以减小精度损失。

> 量化损失不仅和权重有关，和激活也有关，他们是相乘的关系。我们认为权重量化的误差被激活的离群点放大了，通过平滑激活离群点，同时对应调整权重，能够大大降低量化误差。