I-LLM

I-LLM：首次实现了LLM全整形量化，精度逼近浮点，超过Smooth-Omini-AffineQuant

I-LLM: Efficient Integer-Only Inference for Fully-Quantized Low-Bit Large Language Models

Xing Hu[1*] **Yuan Cheng**[1,2*†] **Dawei Yang**[1◇]

Zhihang Yuan[1] **Jiangyong Yu**[1] **Chen Xu**[1] **Sifan Zhou**[1,3†]

[1]Houmo AI
[2]Nanjing University
[3]Southeast University

Abstract

Post-training quantization (PTQ) serves as a potent technique to accelerate the inference of large language models (LLMs). Nonetheless, existing works still necessitate a considerable number of floating-point (FP) operations during inference, including additional quantization and de-quantization, as well as non-linear operators such as RMSNorm and Softmax. This limitation hinders the deployment of LLMs on the edge and cloud devices. In this paper, we identify the primary obstacle to integer-only quantization for LLMs lies in the large fluctuation of activations across channels and tokens in both linear and non-linear operations. To address this issue, we propose I-LLM, a novel integer-only fully-quantized PTQ framework tailored for LLMs. Specifically, (1) we develop Fully-Smooth Block-Reconstruction (FSBR) to aggressively smooth inter-channel variations of all activations and weights. (2) to alleviate degradation caused by inter-token variations, we introduce a novel approach called Dynamic Integer-only MatMul (DI-MatMul). This method enables dynamic quantization in full-integer matrix multiplication by dynamically quantizing the input and outputs with integer-only operations. (3) we design DI-ClippedSoftmax, DI-Exp, and DI-Normalization, which utilize bit shift to execute non-linear operators efficiently while maintaining accuracy. The experiment shows that our I-LLM achieves comparable accuracy to the FP baseline and outperforms non-integer quantization methods. For example, I-LLM can operate at W4A4 with negligible loss of accuracy. To our knowledge, we are the first to bridge the gap between integer-only quantization and LLMs

we are the first to bridge the gap between integer-only quantization and LLMs. We've published our code on anonymous.4open.science, aiming to contribute to the advancement of this field.

1 Introduction

Large Language Models (LLMs) have paved the way for general artificial intelligence with their remarkable performance across a wide range of tasks. However, the rising number of parameters and computing power requirements of LLMs pose significant challenges when it comes to deployment.

Post-training quantization (PTQ) is a powerful technique employed to accelerate the inference process of LLMs. Previous PTQ methods for LLMs have primarily relied on simulated quantization (aka.

◇Corresponding author
*Equal contribution
†This work was conducted during his internship at Houmo

宣传一下我们的工作I-LLM，（据我们所知）这是第一个在LLM上实现了integer-only量化的方法，精度逼近浮点，超过Smooth/Omini/Affine Quant等SOTA方法。

- https://arxiv.org/abs/2405.17849
- 单位：后摩智能、南京大学、东南大学

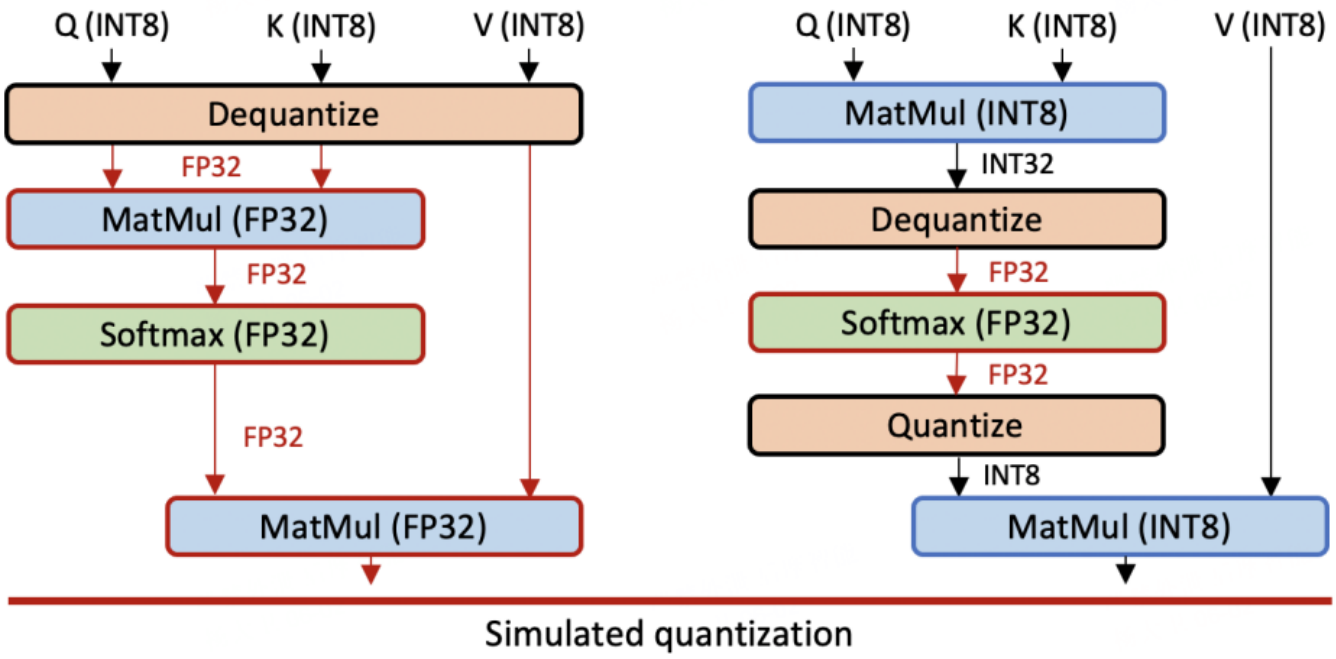
Abstract

PTQ能够有效加速LLMs的推理。然而，现有针对LLM的PTQ方案在推理过程中仍然需要相当多的浮点操作，例如额外的量化和反量化操作，以及复杂的非线性算子（如RMSNorm和Softmax）。这些约束阻碍了LLMs在边缘和云端设备上的部署。我们认为LLMs integer-only量化的主要问题在于linear和non-linear计算时激活值中在跨channel和跨token维度上有巨大波动。

为了解决这个问题，我们提出了I-LLM，这是一种针对LLM量身定制的integer-only PTQ框架。具体来说，（1）我们开发了Fully-Smooth Block-Reconstruction (FSBR)来平滑所有激活和权重channel间的变化。（2）为了减轻token间变化的影响，我们提出了Dynamic Integer-only MatMul (DI-MatMul)方法。该方法仅通过整数运算实现了全整形GEMM的动态量化输入和输出。（3）我们设计了DI-ClippingSoftmax、DI-Exp和DI-Normalization，它们利用高效的bit shift来计算非线性算子，同时保持了精度。

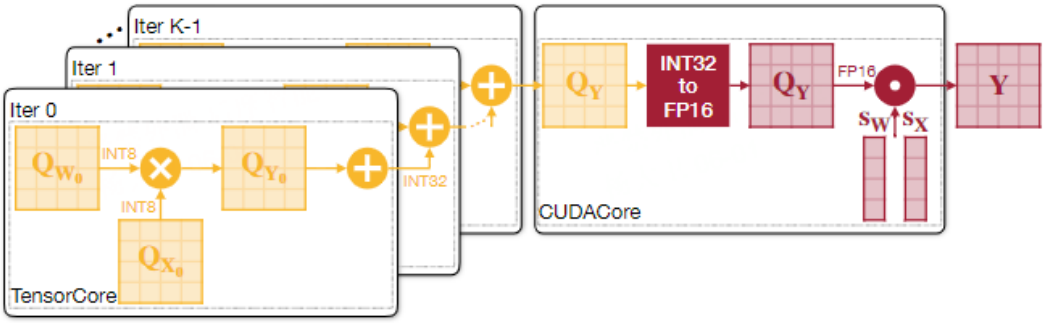
实验结果表明，I-LLM实现了与FP baseline相当的准确度，并且优于非整数量化方法。例如，I-LLM 可以在W4A4上运行，精度损失几乎可以忽略不计。据我们所知，I-LLM是第一个在LLM上实现了integer-only量化的工作。

Introduction

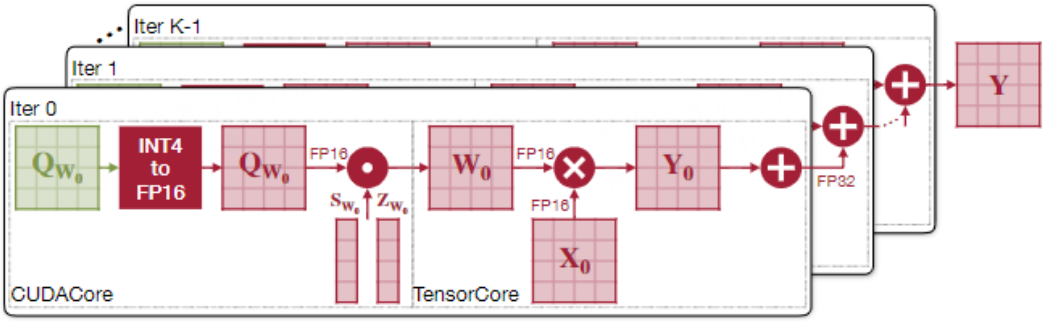


来源：I-Bert

LLMs的计算量和带宽需求很大，往往需要通过量化进行部署加速。然而，现有的LLMs量化方法在推理过程中仍然需要大量的浮点运算，例如Matmul引入了额外的量化和反量化操作、以及Softmax等非线性算子。如下图所示，(a) W8A8量化方案，推理时需要对输入的X进行量化，并且对输出的Y值也需要进行反量化到FP16；(b) W4A16量化方案，推理时需要将W反量化到FP16，随后执行FP16的GEMM运算。**这阻碍了LLMs在边缘和云设备上的部署，这是因为浮点运算代价较高，甚至有些设备没有浮点算力。**



(a) TensorRT-LLM (INT8 Weights and INT8 Activations)



(b) TensorRT-LLM (INT4 Weights and FP16 Activations)

来源：QServe

现有的Integer-only量化方案（如I-Vit、I-Bert）在LLMs上表现很差，因为这些方案是为CNN、Transformer等相对较小的模型设计的，无法处理LLMs激活值存在大量的离群点，包括线性算子（如FC）和非线性算子（如Softmax、SwiGlu）。

为了解决上述问题，我们发表了I-LLM这篇文章，（据我们所知）这是首个在LLMs上实现了Inter-only量化的工作，主要贡献包括：

1. 提出了Fully-Smooth Block-Reconstruction (FSBR)，用于平滑LLMs中所有激活和权重的通道间变化。
2. 提出了Dynamic Integer-only MatMul (DI-MatMul)，在INT GEMM的基础上利用整数操作实现激活的动态量化。
3. 设计了DI-ClippedSoftmax、DI-Exp和DI-Norm等整形推理的非线性算子，采用了高效的位移操作同时保持了精度。

实验表明，I-LLM在W6A6的配置下取得了与浮点模型相当的精度，在W4A4的甚至远优于Weight-only的量化方法。

Table 1: Quantitative weight-activation quantization PPL(↓) results of I-LLM. We report WikiText2 and C4 perplexity of LLaMA Family in this table.

#Bits	Method	LLaMA-7B		LLaMA-13B		LLaMA-30B		LLaMA2-7B		LLaMA2-13B		LLaMA3-8b	
		WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4
FP16	-	5.68	7.08	5.09	6.61	4.10	5.98	5.47	6.97	4.88	6.46	6.14	8.88
W6A6	SmoothQuant	6.03	7.47	5.42	6.97	4.55	6.34	6.2	7.76	5.18	6.76	7.08	10.16
	OmniQuant	5.96	7.43	5.28	6.84	4.38	6.22	5.87	7.48	5.14	6.74	6.97	10.08
	I-LLM	5.84	7.32	5.23	6.79	4.32	6.25	5.68	7.27	5.10	6.74	6.61	9.77
W4A4	SmoothQuant	22.25	32.32	40.05	47.18	192.40	122.38	83.12	77.27	35.88	43.19	418.88	312.86
	OmniQuant	11.26	14.51	10.87	13.78	10.33	12.49	14.26	18.02	12.30	14.55	437.88	315.69
	AffineQuant	10.28	13.64	10.32	13.44	9.35	11.58	12.69	15.76	11.45	13.97	-	-
	I-LLM	9.10	12.33	7.99	10.96	7.24	9.85	10.44	12.92	9.76	12.57	21.19	30.9

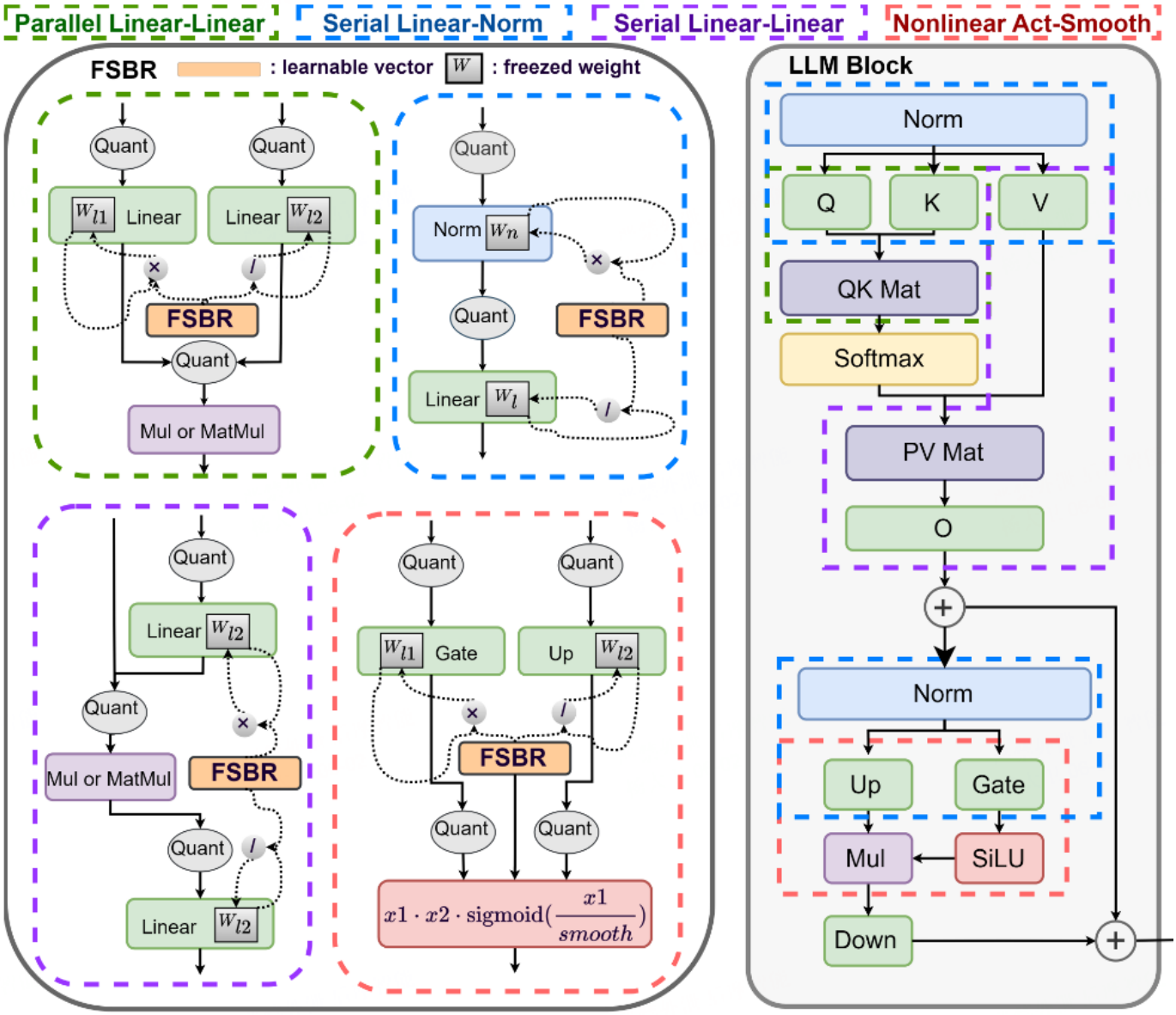
I-LLM在LLaMA上的效果

Method

Fully-Smooth Block-Reconstruction (FSBR)

问题识别： 在大型语言模型中，线性和非线性操作的激活值在不同通道和token之间会有较大的波动。这种波动导致量化后的模型性能下降。

为了缓解LLMs激活中的离群点，我们提出了FSBR。虽然相比Omniquant和Smoothquant的方法有一些相似之处，但他们主要集中于Serial norm-linear和Parallel linear-linear两种运算中的激活。我们认为有效缓解LLMs中所有的激活-激活和激活-权重对的差异能够显著提高准确性，在此基础上，我们的smooth方法增加了Serial Linear-Linear和Nonlinear Act-Smooth，如下图所示。



I-LLM的FSBR示意图

LLMs中非线性层的激活在通道和token维度上也存在较大差异，如下图所示。因此，我们考虑所有非线性层的激活，并在通道级别学习所有可能的等效平滑变换的平滑系数。一种直观的方法是为所有激活和权值训练一个平滑系数，以帮助恢复模型的量化精度。**然而，这对于Linear层比较容易实现，但是对Non-linear算子不能直接进行等价变换。**

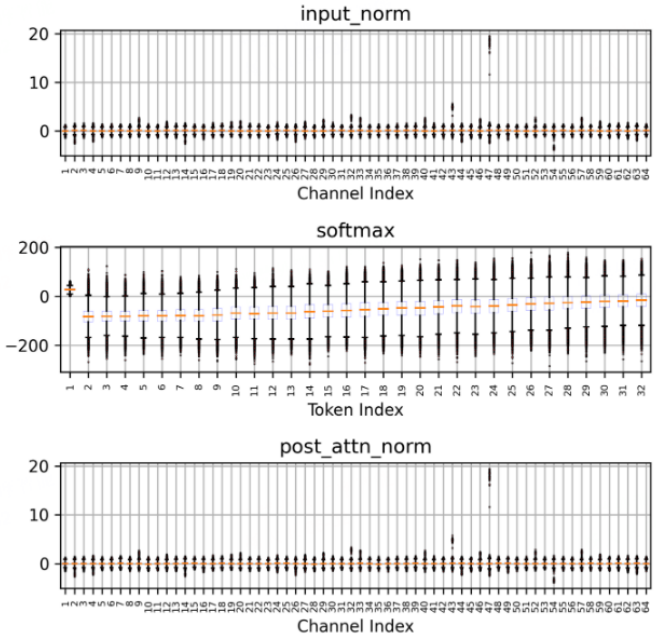


Figure 1: Differences in activations of the non-linear operator in LLaMA2-7b across the channel/token dimensions.

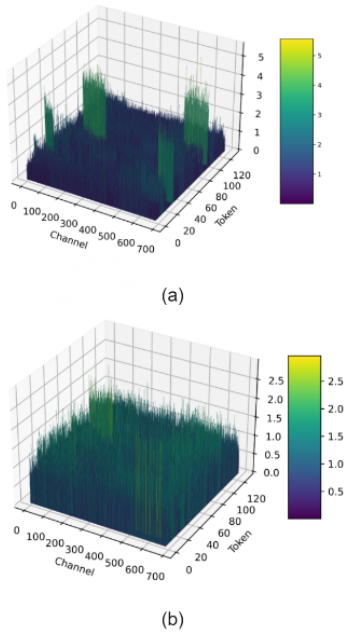


Figure 2: The output activation distribution of the gated unit in the SwiGLU before FSBR (a) and after FSBR (b).

LLM异常值很多

这里以SwiGLU为例：

$$\begin{aligned}\mathrm{SwiGLU}(\boldsymbol{x},\boldsymbol{W},\boldsymbol{V},\boldsymbol{b},\boldsymbol{c}) &= \mathrm{SiLU}(x\boldsymbol{W}+\boldsymbol{b})\otimes(x\boldsymbol{V}+\boldsymbol{c}) \\ &= x1\otimes x2\otimes\sigma(x1)\end{aligned} \tag{1}$$

为了联合优化$x1$和$x2$的分布，我们引入平滑因子s：

$$\begin{aligned}\mathrm{SwiGLU}(x,W,V,b,c) &= (x1\otimes s)\otimes(x2\oslash s)\otimes\sigma(x1\oslash s) \\ &= x1'\otimes x2'\otimes\sigma'\left(x1'\right)\end{aligned} \tag{2}$$

观察图2可以发现，SwiGLU函数的通道和token维度上的的分布不平衡经过FSBR得到了有效缓解。FSBR 的主要目的是平滑不同通道和token之间激活值的波动，从而提高量化模型的准确性。相比Smoothquan和Omniquant，FSBR提供了更多优化权重和激活分布的可能性，为优化权值和激活分布提供了更多可能性。通过跨通道的相互优化，网络对量化的鲁棒性有所提高，如论文中的表4所示。

Table 4: Impact of different PTQ methods and integer-only operators on LLaMA-7B PPL(↓) across WikiText2 and C4 datasets.

LLaMA-7B	W4A4		W6A6	
Method	WikiText2	C4	WikiText2	C4
SmoothQuant	256.58	218.47	6.09	7.6
OmniquantQuant	122.18	183.2	5.99	7.57
FSBR	**9.44**	**12.72**	**5.83**	**7.02**
+DI-CLippedSoftamx	9.44	12.72	5.83	7.02
+DI-Swiglu	9.12	12.38	5.83	7.04
+DI-Norm	9.52	12.63	5.85	7.35

I-LLM的消融实验

以下是FSBR的一些关键点：

1. **平滑激活值**： FSBR 通过为所有激活值和权重学习平滑系数来解决这个问题。这些平滑系数有助于在量化过程中保持模型的精度。
2. **等效平滑变换**： FSBR 考虑了所有可能的等效平滑变换，并在通道级别上学习平滑系数。这包括了Serial norm-linear、Parallel linear-linear、Serial Linear-Linear、Nonlinear Act-Smooth共四种范式。
3. **非线性操作的平滑**： 对于非线性操作，如SwiGLU（一种门控激活函数），FSBR 通过分解操作来应用平滑。

Dynamic Interger-only MatMul (DI-MatMul)

在LLMs中，即使应用了channel间平滑技术，token维度上仍然存在相当大的分布范围，而静态量化方法（即量化参数在运行时固定不变）在遇到超出校准集范围的输入时，往往准确性会严重下降。**因此，现有的量化方案为了保证精度，普遍采用per-token的量化方案，这种在线量化引入了大量的浮点操作** 。而全整形GEMM量化的最大难点在于量化参数（scale和zero point）必须是浮点数。

我们提出了DI-MatMul，旨在解决LLMs中GEMM量化推理中（如W4A4）的动态量化问题。为了避免浮点运算，我们dyadic number近似浮点乘法。以下是DI-MatMul实现的关键步骤：

1. **量化表示**： DI-MatMul使用dyadic number来表示量化步长s，即$s = m^I/2^{k^I}$，其中m^I和k^I都是8位整数。零点zp也以类似的方式表示。
2. **矩阵乘法公式**： DI-MatMul将定点矩阵乘法表达为$\boldsymbol{Y^I}, m_y^I, k_y^I, zp^I = \mathcal{M}\left(m_1^I, k_1^I, zp_1^I, \boldsymbol{X_1^I}, m_2^I, k_2^I, zp_2^I, \boldsymbol{X_2^I}\right)$，其中$Y^I$是量化输出，$X^I$是量化输入。

中间结果： 定点矩阵乘法的中间结果（partial sum）为$\boldsymbol{P^I} = \left(\boldsymbol{X_1^I} - zp_1^I\right)\left(\boldsymbol{X_2^I} - zp_2^I\right)$，对应的浮点结果$\boldsymbol{Y} = \boldsymbol{P^I}\frac{m_1^I m_2^I}{2^{k_1^I+k_2^I}}$，可以参考对称量化输出$Y^I \cdot s_o = s_w s_x \cdot W^I \times X^I$

1. **量化尺度计算**： 为了获得输出的量化scale，DI-MatMul$\frac{m_y^I}{2^{k_y^I}} \approx s_y = \frac{(p_{\max}^I - p_{\min}^I)\cdot m_1^I \cdot m_2^I}{n^I \cdot 2^{k_1^I+k_2^I}}$，其中 p_{max}^I、p_{min}^I是P^I中的最大和最小值。
2. **优化和迭代**： 为了找到最优的m_y^I和k_y^I，可能需要多次迭代，例如：$\arg\min_{m_y^I, k_y^I}\left\| \frac{(p_{\max}^I - p_{\min}^I)\cdot m_1^I \cdot m_2^I}{n\cdot 2^{k_1^I+k_2^I}} - \frac{m_y^I}{2^{k_y^I}} \right\|_1$，但是可以设置$m_y^I = 256$只搜索$k_y^I$的值。
3. **位移操作**： DI-MatMul利用位移操作来实现指数函数和除法操作，这些操作是动态量化所必需的。例如，使用位移操作来近似指数函数，而不是使用传统的浮点运算。

$$
\begin{aligned}
k_y^I &= \left\lfloor \log_2\left(\frac{256 \cdot n^I \cdot 2^{k_1^I+k_2^I}}{p_{\max}^I - p_{\min}^I}\right)\right\rfloor = \left\lfloor \log_2\left(\frac{n^I \cdot 2^{k_1^I+k_2^I+8}}{p_{\max}^I - p_{\min}^I}\right)\right\rfloor \\
m_y^I &= \left\lfloor \frac{\left(p_{\max}^I - p_{\min}^I\right)\cdot m_{x1}^I \cdot m_{x2}^I}{n^I} \gg \left(k_1^I + k_2^I - k_y^I\right)\right\rfloor \qquad (3) \\
z_y^I &= \left\lfloor \frac{-p_{\min}^I \cdot n^I}{p_{\max}^I - p_{\min}^I}\right\rceil, \quad Y^I = \left\lfloor \frac{\left(\boldsymbol{P}^I - p_{\min}^I\right)\cdot n^I}{p_{\max}^I - p_{\min}^I}\right\rceil
\end{aligned}
$$

整个过程中，DI-MatMul只使用整数运算，仅引入了一些额外的整数标量计算，使其比之前的方法更高效。**此外，DI-MatMul 能够主动识别并适应输入数据的多样性，从而减少量化误差，提高整体模型性能。** DI-MatMul 是 I-LLM 框架中实现整数量化推理的重要组成部分，它通过动态量化输入和输出，使得模型能够在保

持精度的同时，充分利用整数运算单元的效率，加速模型的推理过程。这对于在边缘设备上部署大型语言模型具有重要意义。

Dynamic Non-Linear Integer-only Operations

由于DI-MatMul支持了动态量化，导致激活值的scale在运行时是变化的，因此我们也提出了DI-ClippedSoftmax、DI-Exp、DI-Normalization等非线性算子的全整形动态量化方案。详情可以参考论文。

Experiments

实验结果表明，I-LLM在LLaMA和OPT系列的7B、13B、30B模型上都取得了惊人的效果，甚至超过了weight-only的量化方案，远超其他的Integer-only量化方案。例如其W6A6的精度逼近浮点，其W4A4也是超过了SmoothQuant、OnmiQuant、AffineQuant等SOTA方法。

Table 1: Quantitative weight-activation quantization PPL(↓) results of I-LLM. We report WikiText2 and C4 perplexity of LLaMA Family in this table.

#Bits	Method	LLaMA-7B		LLaMA-13B		LLaMA-30B		LLaMA2-7B		LLaMA2-13B		LLaMA3-8b	
		WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4	WikiText2	C4
FP16	-	5.68	7.08	5.09	6.61	4.10	5.98	5.47	6.97	4.88	6.46	6.14	8.88
W6A6	SmoothQuant	6.03	7.47	5.42	6.97	4.55	6.34	6.2	7.76	5.18	6.76	7.08	10.16
	OmniQuant	5.96	7.43	5.28	6.84	4.38	6.22	5.87	7.48	5.14	6.74	6.97	10.08
	I-LLM	5.84	7.32	5.23	6.79	4.32	6.25	5.68	7.27	5.10	6.74	6.61	9.77
W4A4	SmoothQuant	22.25	32.32	40.05	47.18	192.40	122.38	83.12	77.27	35.88	43.19	418.88	312.86
	OmniQuant	11.26	14.51	10.87	13.78	10.33	12.49	14.26	18.02	12.30	14.55	437.88	315.69
	AffineQuant	10.28	13.64	10.32	13.44	9.35	11.58	12.69	15.76	11.45	13.97	-	-
	I-LLM	9.10	12.33	7.99	10.96	7.24	9.85	10.44	12.92	9.76	12.57	21.19	30.9

I-LLM在LLaMA上的性能

Table 2: Quantitative weight-activation quantization PPL(↓) results on OPT Family of I-LLM.

#Bits	Method	OPT-6.7B		OPT-13B		OPT-30B	
		WikiText2	C4	WikiText2	C4	WikiText2	C4
FP16	-	10.86	11.74	10.13	11.20	9.56	10.69
W6A6	SmoothQuant	11.34	12.14	10.56	11.40	9.67	10.81
	RPTQ	11.19	12.08	11.00	11.68	10.22	11.73
	OmniQuant	10.96	11.81	10.21	11.27	9.62	10.76
	I-LLM	10.94	11.82	10.17	11.90	9.72	10.83
W4A4	SmoothQuant	1.8e4	1.5e4	7.4e3	5.6e3	1.2e4	8.3e3
	RPTQ	12.00	12.85	12.74	14.71	11.15	13.48
	OmniQuant	12.24	13.56	11.65	13.46	10.60	11.90
	I-LLM	12.20	12.21	11.45	13.41	10.53	11.66

I-LLM在OPT上的性能

I-LLM在zero-shot的任务上同样表现出色。

Table 3: The performance of various methods for 4-bit and 6-bit quantization on the LLaMA family models across six zero-shot datasets.

LLaMA / Acc(↑)	#Bits	Method	PIQA(↑)	ARC-e(↑)	ARC-c(↑)	BoolQ(↑)	HellaSwag(↑)	Winogrande(↑)	Avg.(↑)
LLaMA-7B	FP16	-	77.47	52.48	41.46	73.08	73.00	67.07	64.09
	W6A6	SmoothQuant	76.75	51.64	39.88	71.75	71.67	65.03	62.81
	W6A6	OmniQuant	77.09	51.89	40.87	72.53	71.61	65.03	63.17
	W6A6	**I-LLM**	76.99	52.66	40.78	72.94	71.31	65.67	**63.39**
	W4A4	SmoothQuant	49.80	30.40	25.80	49.10	27.40	48.00	38.41
	W4A4	LLM-QAT	51.50	27.90	23.90	61.30	31.10	51.90	41.27
	W4A4	LLM-QAT+SQ	55.90	35.50	26.40	62.40	47.80	50.60	46.43
	W4A4	OS+	62.73	39.98	30.29	60.21	44.39	52.96	48.43
	W4A4	OmniQuant	66.15	45.20	31.14	63.51	56.44	53.43	52.65
	W4A4	AffineQuant	69.37	42.55	31.91	63.73	57.65	55.33	53.42
	W4A4	**I-LLM**	67.25	45.58	32.59	63.88	58.89	57.06	**54.21**
LLaMA-13B	FP16	-	79.10	59.89	44.45	68.01	76.21	70.31	66.33
	W6A6	SmoothQuant	77.91	56.60	42.40	64.95	75.36	69.36	64.43
	W6A6	OmniQuant	78.40	57.28	42.91	67.00	75.82	68.27	64.95
	W6A6	**I-LLM**	77.48	56.94	44.03	64.92	75.24	69.14	**64.63**
	W4A4	SmoothQuant	61.04	39.18	30.80	61.80	52.29	51.06	49.36
	W4A4	OS+	63.00	40.32	30.38	60.34	53.61	51.54	49.86
	W4A4	OmniQuant	69.69	47.39	33.10	62.84	58.96	55.80	54.37
	W4A4	AffineQuant	66.32	43.90	29.61	64.10	56.88	54.70	52.58
	W4A4	**I-LLM**	67.95	48.15	34.47	62.29	63.13	59.98	**56.00**
LLaMA-30B	FP16	-	80.08	58.92	45.47	68.44	79.21	72.53	67.44
	W6A6	SmoothQuant	77.14	57.61	42.91	65.56	78.07	69.92	65.20
	W6A6	OmniQuant	78.40	57.28	42.91	67.00	75.82	68.27	64.95
	W6A6	**I-LLM**	79.43	58.88	45.14	73.36	78.51	72.61	**67.99**
	W4A4	SmoothQuant	58.65	35.53	27.73	60.42	35.56	48.06	44.83
	W4A4	OS+	67.63	46.17	34.40	60.70	54.32	52.64	52.62
	W4A4	OmniQuant	71.21	49.45	34.47	65.33	64.65	59.19	56.63
	W4A4	AffineQuant	66.32	43.90	29.61	64.10	56.88	54.70	52.58
	W4A4	**I-LLM**	71.38	51.81	37.12	65.69	67.79	61.40	**59.20**

I-LLM zero-shot性能

这项工作首次在LLMs上实现了全整形量化，为LLMs的高效部署和推理开辟了新的途径，尤其是边缘设备上。

顺便打个广告

- 大模型及多模态算法研究（开放场景的2D/3D感知、模型轻量化设计等）
- 模型加速优化研究（PTQ、QAT、混合精度量化、模型压缩等）
- 软硬件协同设计（AI模型加速、算子硬件化、指令集开发等）

开发方向（Mentor提供工程指导）

- AI工具链开发（模型解析、图优化等）
- AI算子设计和开发（如投影变换、超越函数、LayerNorm、Grid-sample等）
- 模型部署优化（性能优化、Benchmark验证等）

部分研究成果（近1年）

- A 22nm 64kb Lightning-like Hybrid Computing-in-Memory Macro with Compressor-based Adder-tree and Analog-storage Quantizer for Transformer and CNNs, ISSCC 2024
- MIM4DD: Mutual Information Maximization for Dataset Distillation, NeuIPS 2023.
- RPTQ: Reorder-based Post-training Quantization for Large Language Models. arXiv preprint 2023.
- Post-training Quantization on Diffusion Models. CVPR 2023
- PD-Quant: Post-Training Quantization based on Prediction Difference Metric. CVPR 2023.
- Latency-aware Spatial-wise Dynamic Networks, NeurIPS 2022.
- Flatfish: a Reinforcement Learning Approach for Application-Aware Address Mapping. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2022.
- PTQ4ViT: Post-Training Quantization Framework for Vision Transformers. European Conference on Computer Vision (ECCV), 2022.
- 3DPEE: 3D Point Positional Encoding for Multi-Camera 3D Object Detection Transformers. ICCV 2023.

👏 后摩智能于2020年在南京成立，是国内首家基于“存算一体”技术的智能驾驶芯片高新技术企业，在北京、上海、深圳等地方建有研发中心。后摩智能致力于突破智能计算芯片性能及功耗瓶颈，加速人工智能普惠落地。其提供的大算力、低功耗的高能效比芯片及解决方案，可应用于智能驾驶、泛机器人等边缘端，以及云端推理场景。2023年5月，发布了首款基于SRAM的存算一体大算力AI芯片产品，算力高达256Tops。

后摩智能 HOUMO.AI