

# OMNIQUANT: OMNIDIRECTIONALLY CALIBRATED QUANTIZATION FOR LARGE LANGUAGE MODELS(ICLR 2024)

## 摘要

大型语言模型 (LLM) 彻底改变了自然语言处理任务。然而，它们的实际部署受到其巨大的内存和计算要求的阻碍。尽管最近的训练后量化 (PTQ) 方法在减少内存占用和提高 LLM 的计算效率方面很有效，但它们需要手动制作量化参数，导致性能低下，尤其是在极低位量化中。为了解决这个问题，我们为 LLM 引入了一种全向校准量化 (OmniQuant) 技术，该技术通过高效优化各种量化参数，在保持 PTQ 计算效率的同时，在各种量化设置中实现了良好的性能。OmniQuant 包含两个创新组件，包括可学习权重裁剪 (LWC) 和可学习等效变换 (LET)。LWC 通过优化剪辑阈值来调制权重的极值。同时，LET 通过将量化的挑战从激活转移到权重来解决激活异常值。OmniQuant 在可微分框架内使用逐块误差最小化运行，可以有效地优化仅权重和权重激活量化的量化过程。例如，使用 OmniQuant 可以在 116 小时内使用单个 A100-40G GPU 使用 128 个样品处理大小为 7-70B 的 LLaMA-2 型号系列。广泛的实验验证了 OmniQuant 在各种量化配置中的卓越性能，例如 W4A4 (4 位权重、4 位激活)、W6A6、W4A16、W3A16 和 W2A16。此外，OmniQuant 在指令调整模型中展示了有效性，并在真实设备上的推理速度和内存减少方面取得了显著改进。

关键词: OmniQuant; Learnable Weight Clipping; Learnable Equivalent Transformation

## 1 引言

该论文针对现有模型普遍体积庞大，为了进一步减少模型内存占用和加速推理，以及改进现有PTQ手动设置参数的缺点，作者提出OmniQuant，既能在多种量化设置上达到很好的性能，同时也能保持如PTQ般好的计算效率。OmniQuant有两个核心的创新点，分别是Learnable Weight Clipping (LWC)和Learnable Equivalent Transformation (LET)，这两个部件使得模型的参数变得对量化友好。结果表明，OmniQuant在语言生成和零样本任务方面优于当前方法，在各种量化配置中具有卓越的性能，例如 W4A4 (4 位权重、4 位激活)、W6A6、W4A16、W3A16 和 W2A16。大模型压缩是现在比较热点的研究方向，模型的规模变得越来越大，如果能将模型压缩到合适的大小且能保持一定的性能，将会极大推动模型在边缘侧的部署，因此具有良好的前景。这篇文章是一篇目前能在众多量化压缩方法脱颖而出的荣获ICLR2024 spotlight的论文非常吸引我，这也是我选择复现这篇论文的理由之一。

## 2 相关工作

### 2.1 量化方法

量化降低了神经网络的位精度，从而产生了更小的模型和更快的推理速度。目前的方法主要分为量化感知训练（QAT）和训练后量化（PTQ）。虽然 QAT 通过在训练期间模拟量化来保持性能，但其训练成本使其不适合 LLM。AdaRound [7] 和 BRECQ [5] 等 PTQ 技术使用梯度优化来确定最佳舍入，但对于较大的模型，调整所有权重非常耗时。因此，大多数 LLM 量化方法优先考虑无训练 PTQ，这限制了低位情况下的性能。我们的目标是将梯度更新集成到 LLM 量化中，反映 QAT 的方法，同时保持 PTQ 的效率。

### 2.2 针对大语言模型的量化

考虑到量化的对象，退出 LLM 量化可以分为两个字段：weightonly 量化和 weight-activation 量化。

#### 2.2.1 仅量化权重

仅权重量化侧重于将权重转换为低位值。例如，GPTQ [3] 使用逐块重建进行 3/4 位量化。SpQR [2]、OWQ [4] 和 AWQ [6] 强调与更高幅度激活相关的权重的重要性。因此，SpQR 和 OWQ 采用混合精度量化来保护重要权重，而 AWQ 选择通道级缩放以避免混合精度的硬件效率低下。Qlora [1] 和 INT2.1 通过参数高效的微调来恢复量化模型的功能。相比之下，我们的方法直接增强了量化过程，使 OmniQuant 成为 Qlora 和 INT2.1 的补充。

#### 2.2.2 量化权重和激活

权重激活量化会同时压缩权重和激活。SmoothQuant、LLM.int8（）和 Outlier Suppression 通过管理激活异常值来实现 W8A8 量化。LLM.int8（）使用混合精度分解，而其他两个使用通道缩放。此外，Outlier Suppression+增加了通道移位以驱动 W6A6 量化。与以前的启发式设计不同，我们使用梯度优化并将等效变换扩展到注意力机制，进一步促进了 K/V 缓存量化。最近，RPTQ 和 LLM-QAT 已经实现了 W4A4 量化。但是，RPTQ 采用对部署不友好的组激活量化，而 LLM-QAT 采用耗时的 QAT。与 RPTQ 和 LLM-QAT 不同，我们通过易于部署的每标记量化实现 W4A4 量化，并保持 PTQ 效率。

## 3 本文方法

### 3.1 本文方法概述

在一个 transformer block 里 OmniQuant 的细节如图 1 所示：

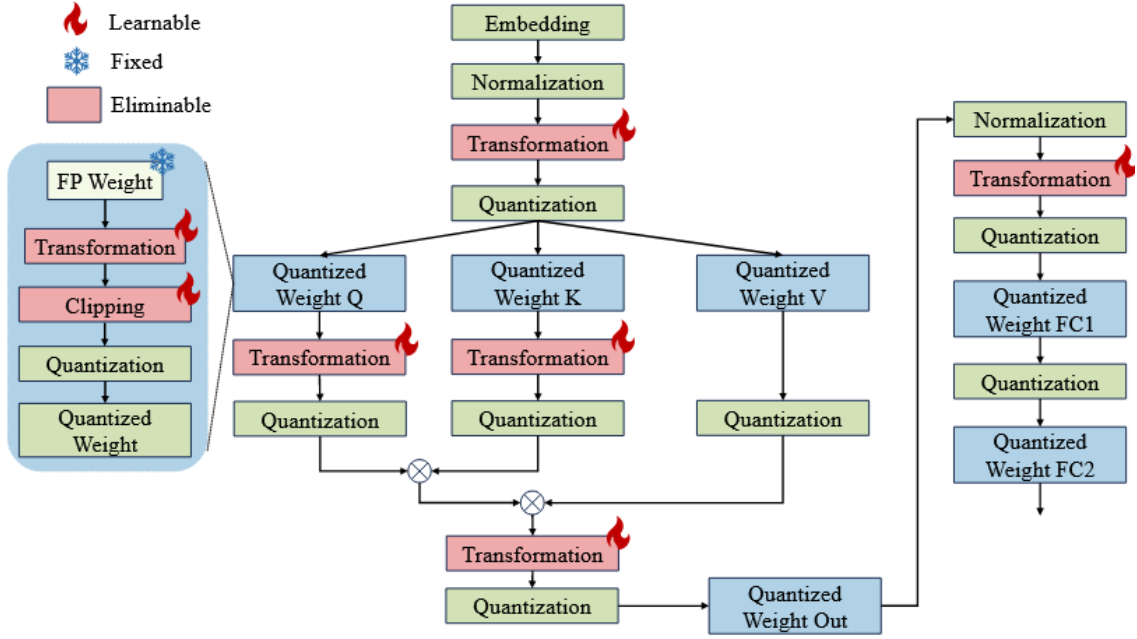


图 1. 方法示意图

OmniQuant 由两个关键组件组成，其中包含不同类型的可学习量化参数，包括可学习权重裁剪 (LWC) 和可学习等效变换 (LET)。具体来说，LWC 通过优化限幅阈值来调节权重的极值。与此同时，LET 通过在 Transformer 编码器中学习数学上等效的变换来解决激活异常值。OmniQuant 不是联合优化 LLM 中的所有参数，而是在逐块量化误差最小化框架下按顺序量化一层参数，然后再进入下一层。这样，OmniQuant 可以使用简单的随机梯度下降 (SGD) 算法进行有效优化。由于可微分优化，LWC 和 LET 可以无缝集成到量化中。我们发现 LWC 可以减轻量化权重的难度，而 LET 进一步将量化的挑战从激活转移到权重，从而促进 OmniQuant 成为适用于仅权重和权重激活量化的通用量化框架。值得注意的是，OmniQuant 没有为量化模型引入额外的计算或参数，因为 LWC 中的限幅阈值和 LET 中的等效因子可以融合到量化权重中。

### 3.2 最小化分块量化误差

先前具有梯度优化的 PTQ 方法，例如 AdaRound、BRECQ 等，无法应用于具有数十亿参数的模型，因为它们由于解空间巨大而难以优化。因此提出了一种具有逐块量化误差最小化的新优化管道，而不是转变整个模型，其中可以以可微分的方式优化附加量化参数。

### 3.3 可学习的权重剪枝

OmniQuant 采用可学习权重裁剪 (LWC) 模块来降低量化权重的难度。与之前具有可学习裁剪阈值的方法类似，LWC 也通过优化裁剪阈值来确定权重的最佳动态范围。

### 3.4 可学习的等价变换

除了通过优化裁剪阈值来实现量化友好权重的 LWC 之外，我们还通过可学习的等效变换 (LET) 进一步降低了权重激活量化的难度。考虑到激活图中的异常值是系统性的并且是

特定通道所独有的，之前的方法，例如 SmoothQuant，通过数学上等效的变换将量化的难度从激活转移到权重。然而，他们手工设计了等效参数，导致结果不理想。由于包含了逐块量化误差最小化，LET 可以以可微分的方式确定最佳等效参数。受 SmoothQuant (Xiao et al., 2023) 和 Outlier Suppression+ (Wei et al., 2023) 的启发，我们采用逐通道缩放和逐通道移位来操纵激活分布，为异常值问题提供了有效的解决方案。

## 4 复现细节

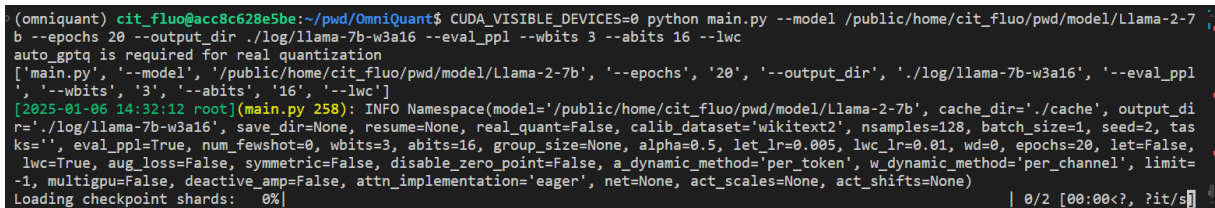
### 4.1 与已有开源代码对比

本次复现参考了论文已有的源代码。源代码实现了简单且高效的针对大模型的量化，包括了仅权重量化(W4A16/W3A16/W2A16)和权重激活量化 (W6A6, W4A4)，支持的模型包括 LLaMA-1, LLaMA-2-Chat, OPT, Falcon, Mixtral-7Bx8等。还可以利用MLC-LLM框架在GPU和手机上运行 LLaMa-2-Chat (7B/13B) (W3A16g128量化)。本次复现工作调整了一下参数以及，加入了SmoothQuant里面的平滑操作，但是并没有任何性能上的提升，所以最后仅仅接近论文的结果，未能完全复现。

### 4.2 使用说明

一个量化的例子是使用一下命令来实现W3A16的量化：

```
CUDA_VISIBLE_DEVICES=0 python main.py --model
/public/home/cit_fluo/pwd/model/Llama-2-7b-hf
--epochs 20 --output_dir ./log/llama-7b-w3a16
--eval_ppl --wbits 3 --abits 16 --lwc
```



```
(omniquant) cit_fluo@acc8c628e5be:~/pwd/OmniQuant$ CUDA_VISIBLE_DEVICES=0 python main.py --model /public/home/cit_fluo/pwd/model/Llama-2-7
b --epochs 20 --output_dir ./log/llama-7b-w3a16 --eval_ppl --wbits 3 --abits 16 --lwc
auto_gptq is required for real quantization
['main.py', '--model', '/public/home/cit_fluo/pwd/model/Llama-2-7b', '--epochs', '20', '--output_dir', './log/llama-7b-w3a16', '--eval_ppl', '--wbits', '3', '--abits', '16', '--lwc']
[2025-01-06 14:32:12 root](main.py 258): INFO Namespace(model='/public/home/cit_fluo/pwd/model/Llama-2-7b', cache_dir='./cache', output_dir='./log/llama-7b-w3a16', save_dir=None, resume=None, real_quant=False, calib_dataset='wikitext2', nsamples=128, batch_size=1, seed=2, tasks='eval_ppl', eval_ppl=True, num_fewshot=0, wbits=3, abits=16, group_size=None, alpha=0.5, let_lr=0.005, lwc_lr=0.01, wd=0, epochs=20, let=False, lwc=True, aug_loss=False, symmetric=False, disable_zero_point=False, a_dynamic_method='per_token', w_dynamic_method='per_channel', limit=-1, multigpu=False, deactivate_amp=False, attn_implementation='eager', net=None, act_scales=None, act_shifts=None)
Loading checkpoint shards: 0%|
```

图 2. W3A16量化

## 5 实验结果分析

在W3A16量化中，复现出来的OmniQuant在WikiText2 perplexity上的表现为6.64，在 C4 perplexity上的表现为8.68；在W3A16g128量化中，复现出来的OmniQuant在WikiText2 perplexity上的表现为6.20，在 C4 perplexity上的表现为7.87。以上都是仅量化权重。而在W4A4量化中，复现出来的OmniQuant在WikiText2 perplexity上的表现为16.22，在 C4 perplexity上的表现为21.29。这是量化权重和激活值。实验结果对比如下图：

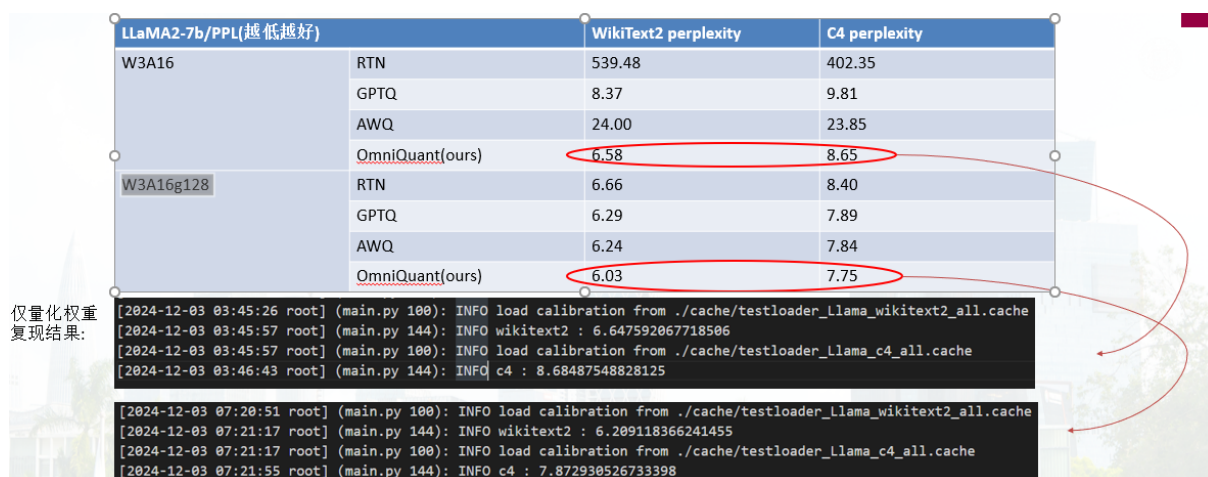


图 3. 仅量化权重

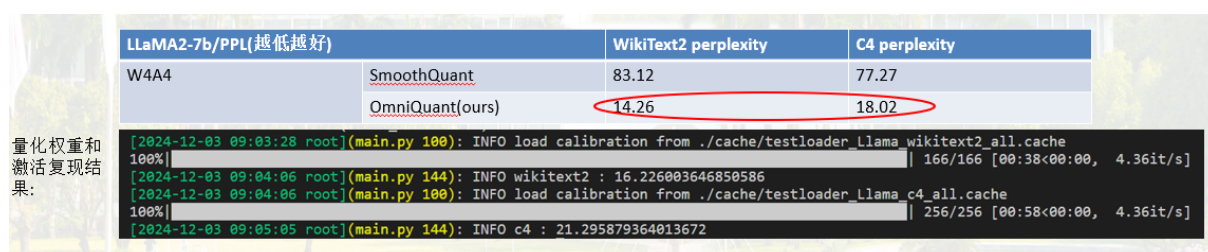


图 4. 量化权重和激活

复现出来的困惑度比论文的高（困惑度越低越好）可能的原因有用的模型是.hf版和参数调得不好。

## 6 总结与展望

本次复现自认为做的工作还不够多，复现出来的结果与论文所呈现的还是差了一点，可能实验做得比较少。未来我会在大模型量化的应用方面深耕，重点在于后训练量化这一领域，把它应用到mamba等主流模型里。

## 参考文献

- [1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.
- [2] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.
- [3] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

- [4] Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364, 2024.
- [5] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.
- [6] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.
- [7] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.