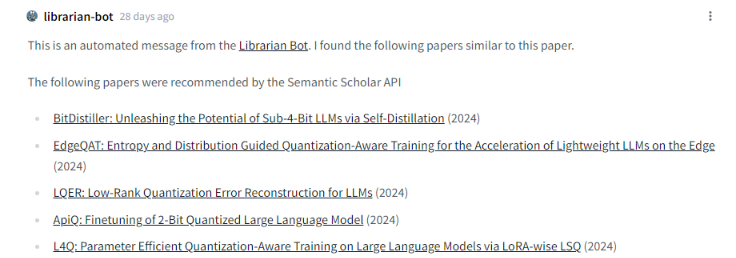
# LLM模型量化

[深度学习的未来将是稀疏的](https://zhuanlan.zhihu.com/p/376638174)



# 调研情况-20240320

在arXiv上使用关键字“compression survey large language model”，得到18篇文献。

经过筛选，近期LLM模型加速和压缩相关的文章如下：

* **A Survey on Knowledge Distillation of Large Language Models**

* **Faster and Lighter LLMs: A Survey on Current Challenges and Way Forward**

* **The Efficiency Spectrum of Large Language Models: An Algorithmic Survey**

* **Model Compression and Efficient Inference for Large Language Models: A Survey**

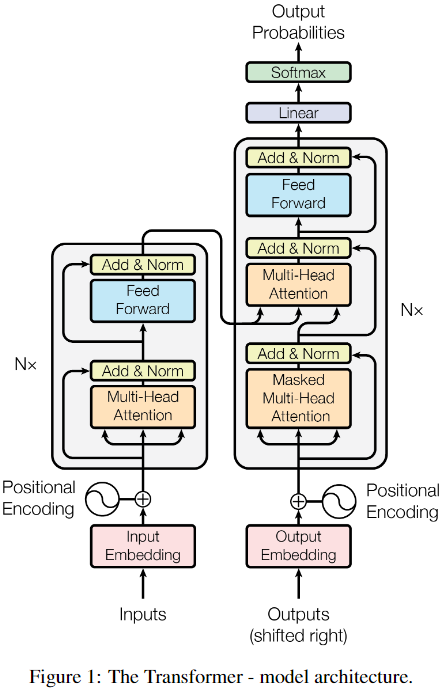
* **A Comprehensive Survey of Compression Algorithms for Language Models**

* **A Survey on Model Compression for Large Language Models**

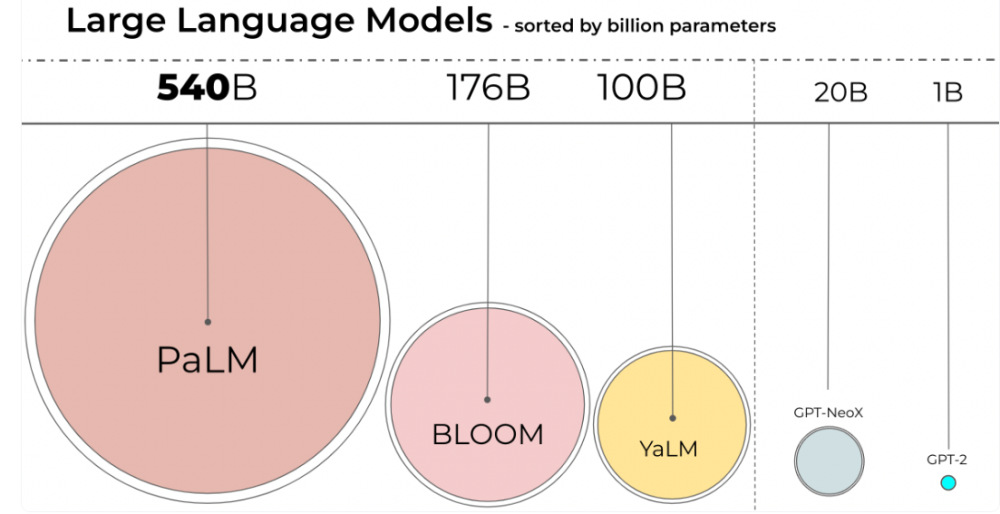
* **A Survey on Transformer Compression**

# LLM模型规模快速增长

Attention is all you need论文中提出的Transformer的结构如下：



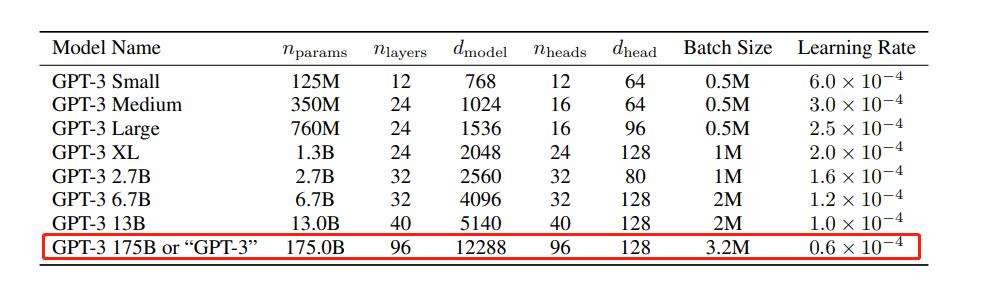
LLM模型的规模快速增大[8]



模型的参数量主要包括两个部分：

1. 词汇embedding

1. 模型权重

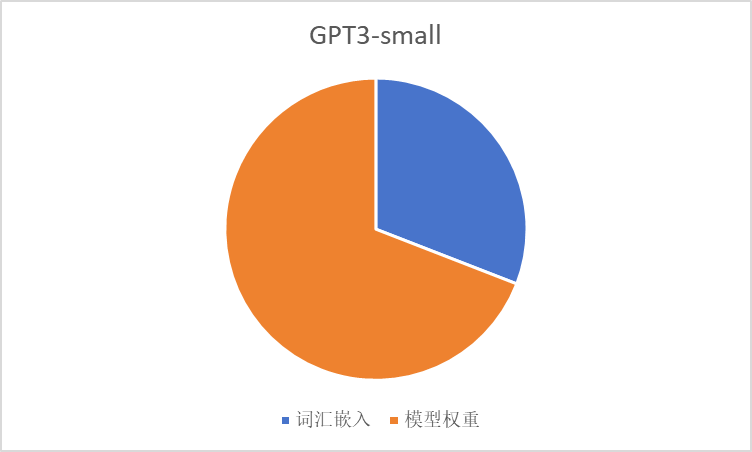


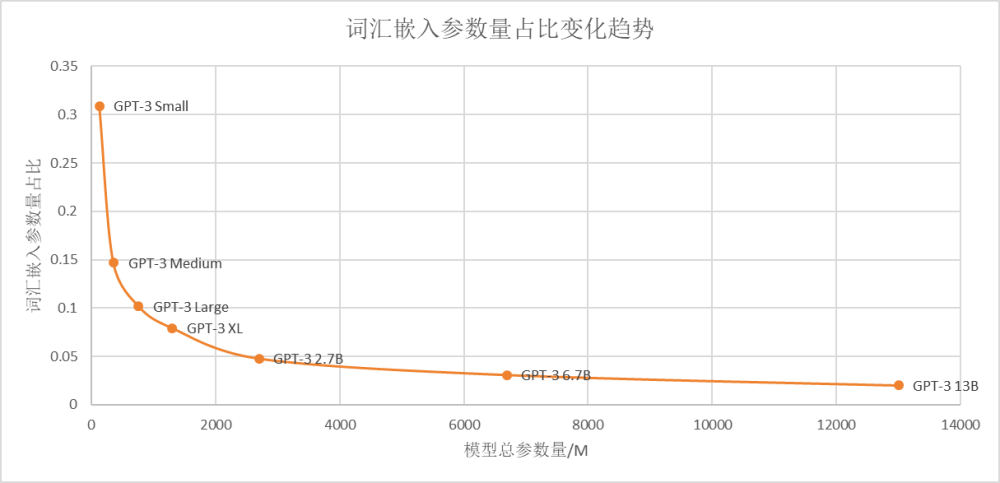
其中，模型的权重占大头，而且随着模型的增大，模型权重的占比持续增大。

例如：

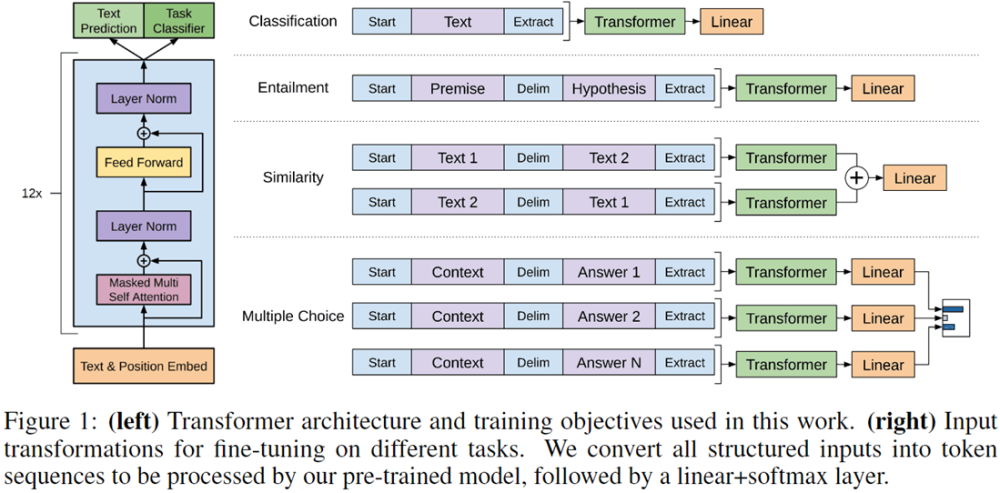
GPT3-small 125M 词汇嵌入向量维度为768，词汇表大小是50257

GPT3 175B 词汇嵌入向量维度为12288，词汇表大小是50257





以GPT系列的decoder-only架构为例，GPT-1的模型架构



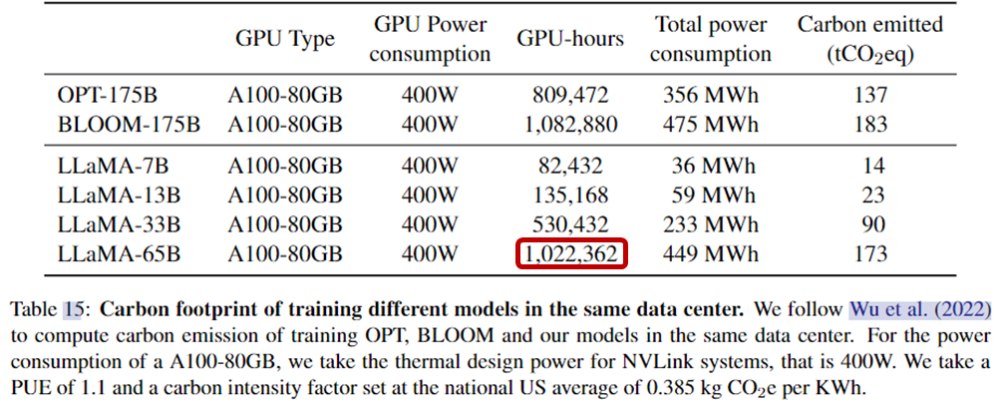
GPT模型的规模快速增大：

1. 响应延时上升

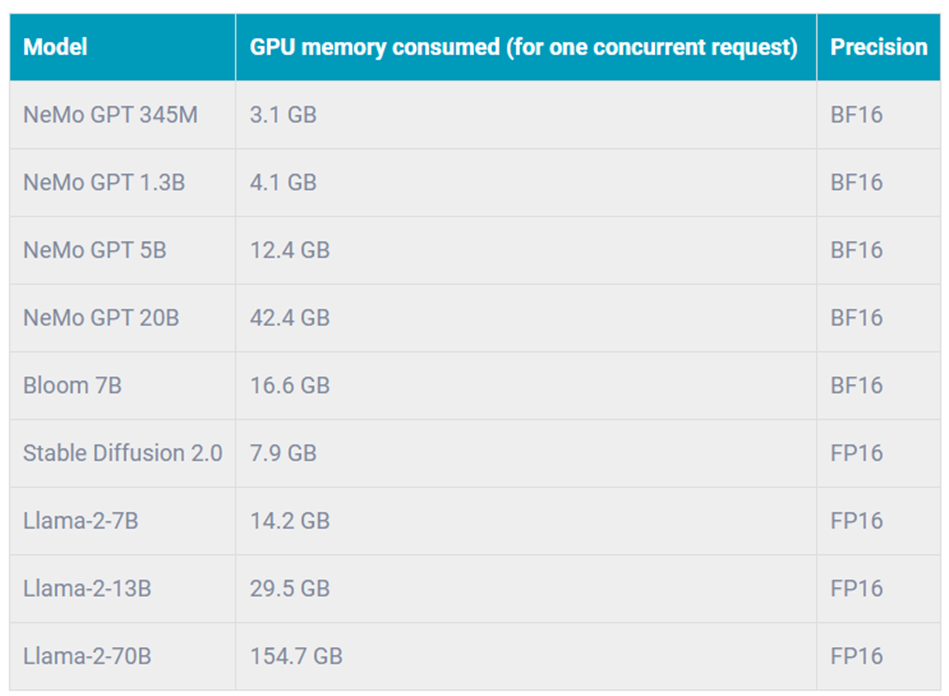
1. 训练/推理计算量上升

1. 内存需求上升

Llama-65B 预训练1.4T语料需要1M A100卡时



单精度推理，1B的参数量~2.1GB内存



针对LLM模型规模增大带来的问题，LLM模型的压缩方法如火如荼的发展：

1. 量化

1. 剪枝

1. 知识蒸馏



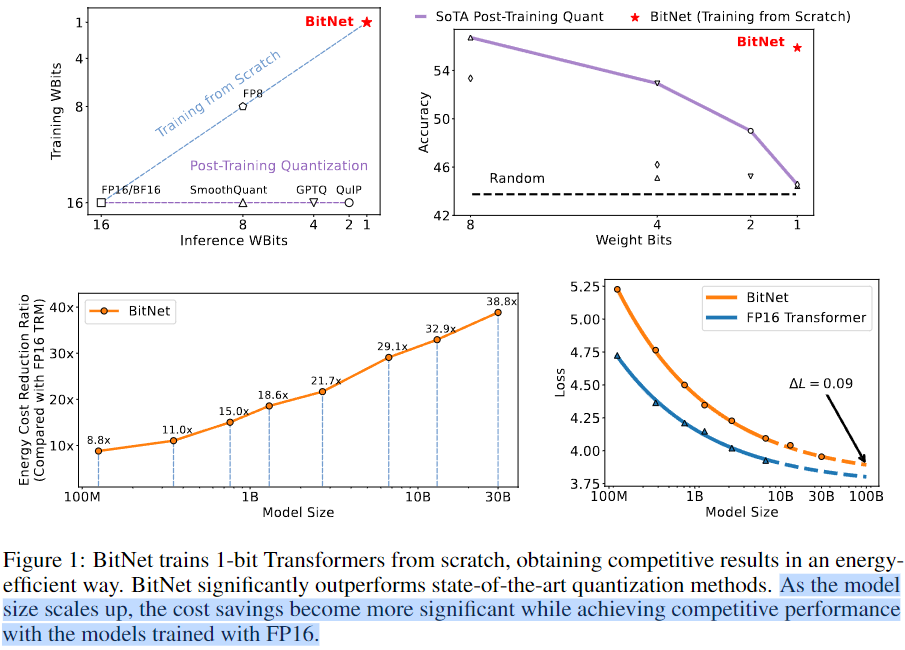
# PTQ

## GPTQ

## AWQ

# QAT

# BitNet: Scaling 1-bit Transformers for Large Language Models[1]



模型的量化工作一直存在。

在CNN网络的领域中就曾经出现过2-bit量化的工作[2,3]

但是在Transformer模型架构上的2-bit量化工作不多。

量化的工作分为两类：

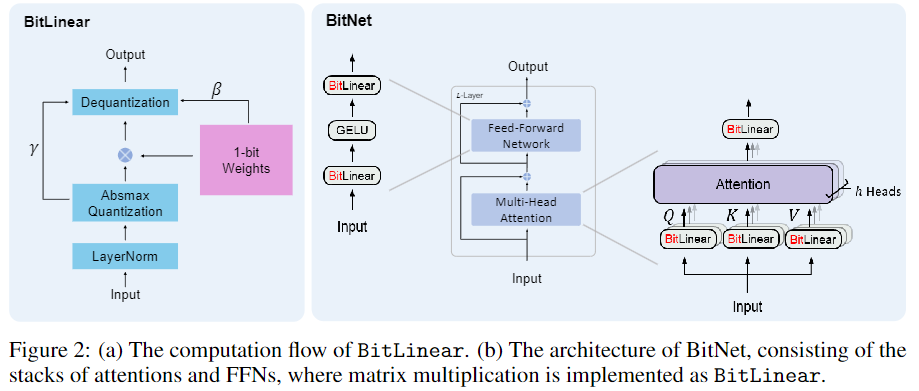
* post-training quantization

* quantization-aware training

前者不影响训练过程，在推理过程中牺牲效果换速度。

后者对整个pipeline做量化，在直觉上，会比前者有更高的上限。

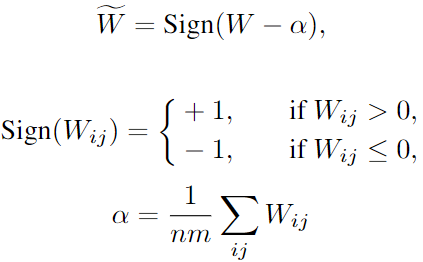
模型架构：



关于模型的数学描述推导，文章写得一坨，小错误一堆。

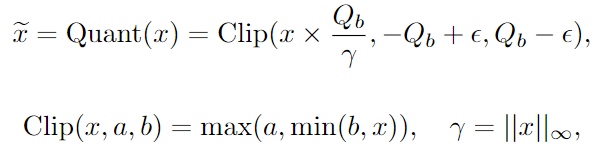
## BitLinear的结构

1. 将初始化的weight二值化（Kaiming Initialization或者Xavier Initialization）



1. 对输入做LayerNorm

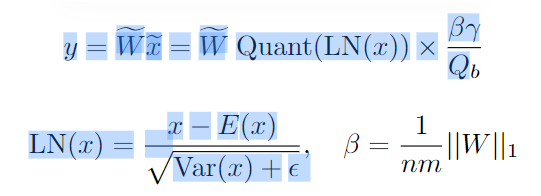
1. 将LayerNorm的结果量化为8bit的数（文章暂时探究了8bit）



1. 二值化的weight和8bit的输入做乘法：



1. 将乘积结果缩放，并且转换回原精度，整个流程为：



注意：**经过上述的缩放，使得BitLinear层的输出和普通的Linear层的输出结果的方差和均值在同一量级上**。

例如，对于同样的输入和权重初始化，BitLinear的输出和普通Linear的输出数据特性如下：

BitLinear, the mean and var of output:

-0.0038397007156163454

0.252113938331604

Linear, the mean and var of output\_no\_quan:

-0.010896746069192886

0.34233447909355164

## 模型训练

BitLinear替换Transformer中的Multi-head Attention和Feed Forward模块中的Linear层。

为了适应以上的改变，BitNet模型的训练有三个主要改进：

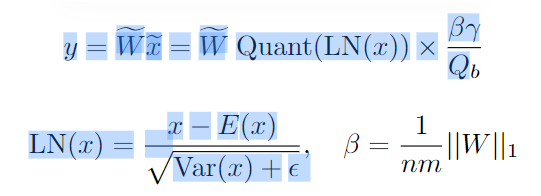
1. Straight-through estimator ，在反向传播的过程中跳过BitLinear中的不可微分函数。[4]

1. Mixed precision training，混合精度的训练

1. Large learning rate，为了适应量化的参数，将学习率扩大

## 计算效率

将Transformer中MHA和FFN中的Linear层换成如下所示的BitLinear层。

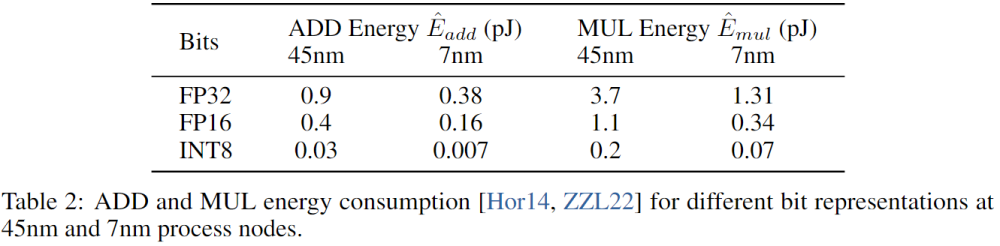


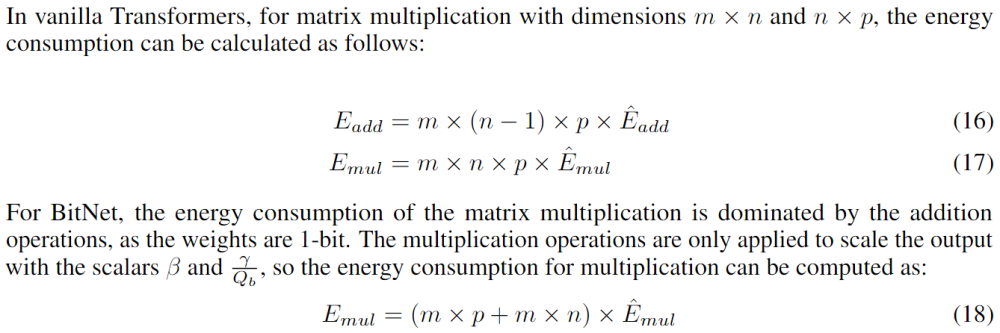
从形式上来看，计算过程变得复杂了。

从实现上来看，计算量和内存求变小了。

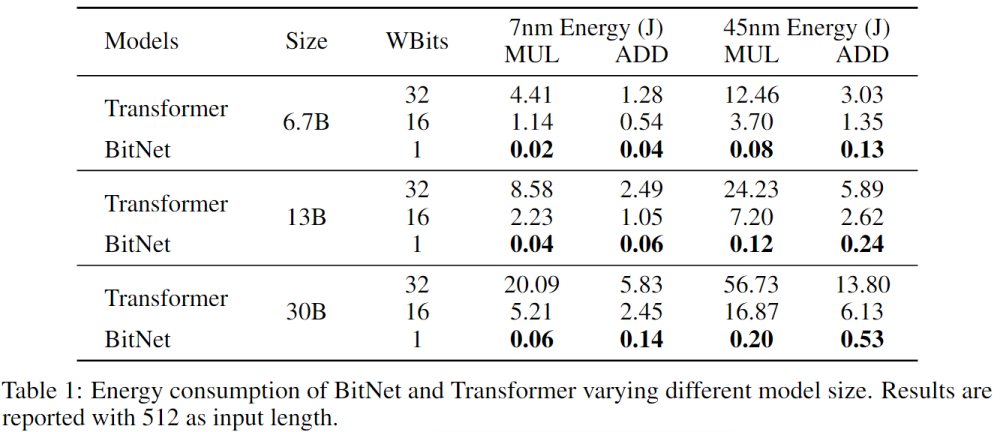
We mainly focus on the calculation for the matrix multiplication, since it contributes the most to the cost of large language models.

根据以往的研究结论，计算机实现不同的算术计算过程的能量消耗：

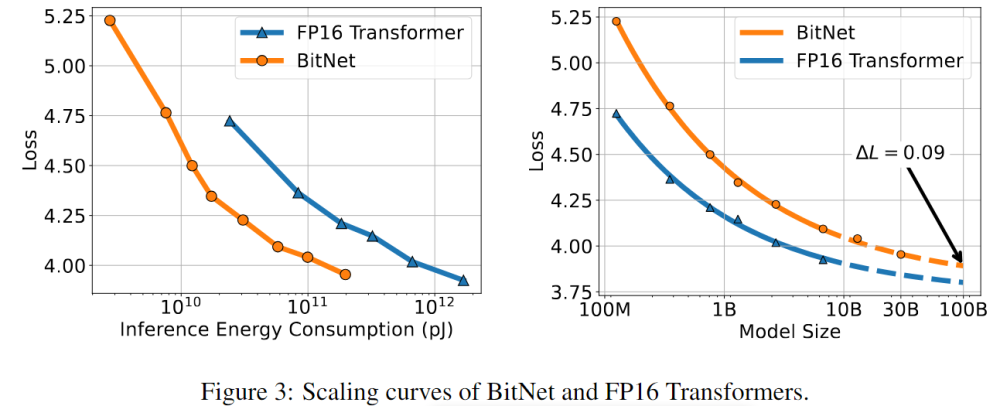




能耗实验结果



## 与FP16 Transformer模型的比较



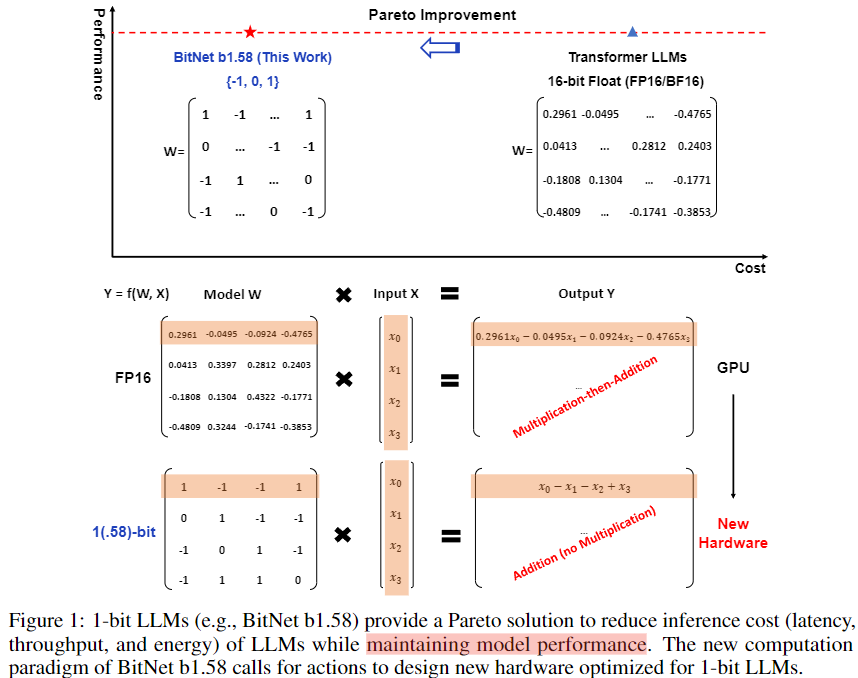


1. BitNet同样展现出类似的scaling law特性

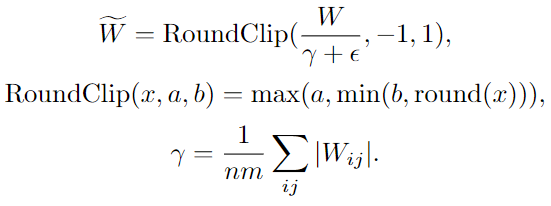
1. 从计算能耗的角度来看，BitNet优势明显

# BitNet b1.58[5]

方法总览：



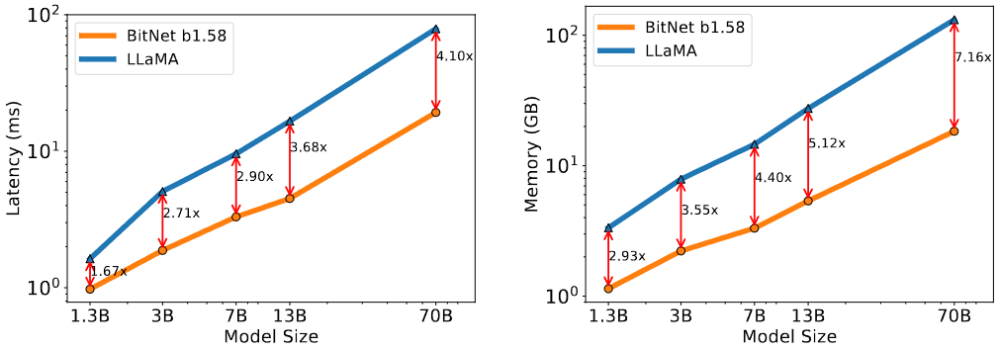
在BitNet的基础上，改变weight的量化方法：



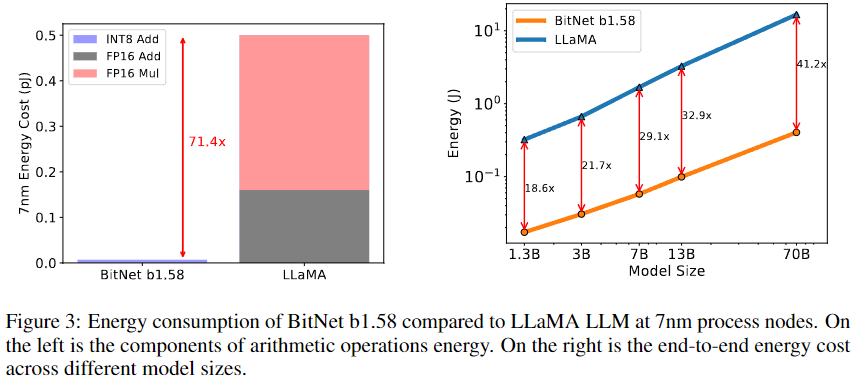
## 实验结果

结合BitNet b1.58和Llama2的其他组件，以全精度FP16的Llama为对比对象

Latency和Memory的比较

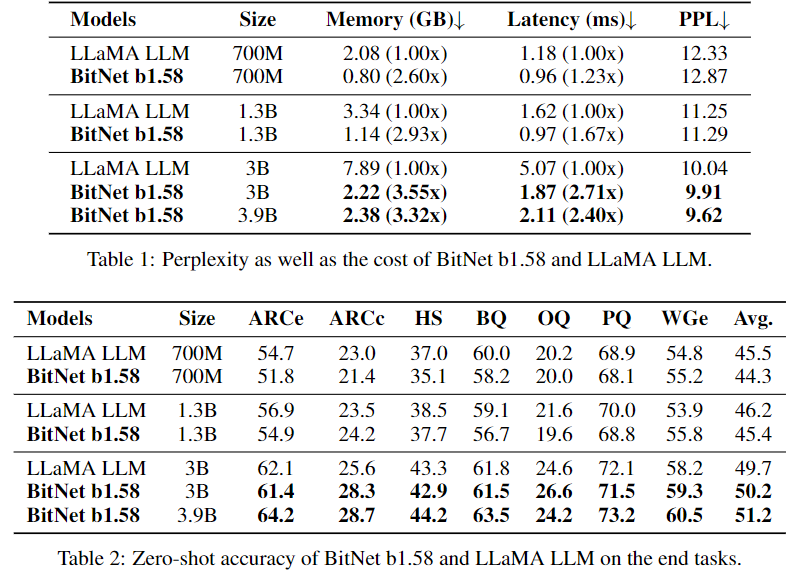


能量损耗的比较

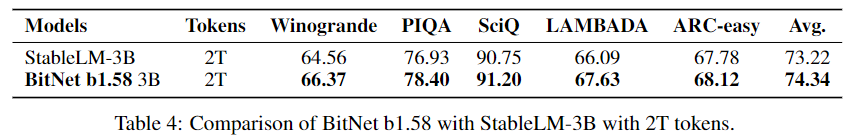


性能比较

和Llama（FP16）进行比较，在100billion（0.1T）的数据集上进行预训练



和StableLM-3B进行比较，在2 trillion（2T）的数据集上进行预训练



## 后续工作

1-bit MOE模型

长距离LLM的原生支持

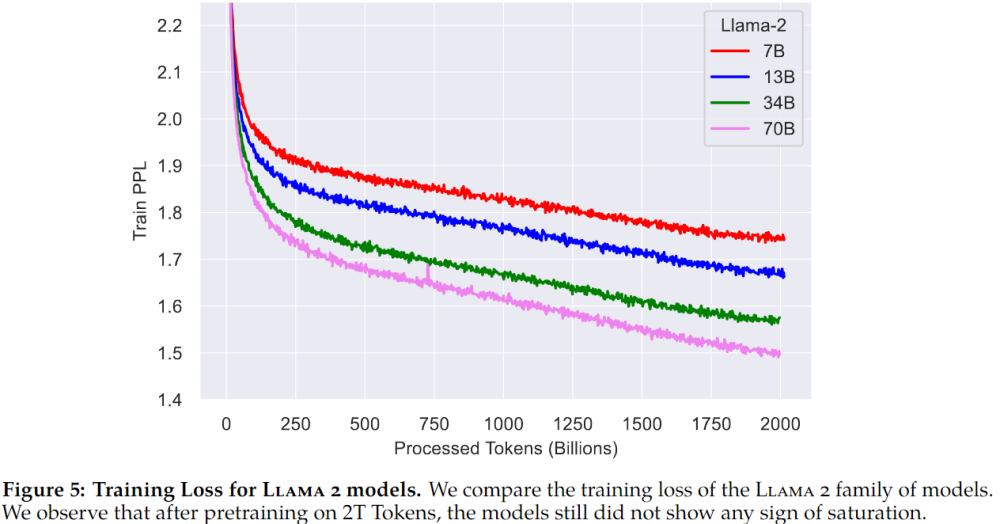
边缘和移动计算设备上的LLM

支持1-bit LLM的新硬件架构

Llama 2在 2 trillion token（2T）上训练

BitNet b1.58及其对比模型在 100 billion token（0.1T）上训练

根据Llama 2的论文，此时的模型还没有被完全训练。



问题：

为什么只报告了最高3B的性能，没有报告更大模型的性能？

可能是没有做训练，只计算了复杂度

怎么看待BitNet的实验结果：

从学术的角度来看，有两种可能性：

1. 稀疏性的Bias符合某种底层规律，所以BitNet对模型的能力是正向的提升。

1. 稀疏性的Bias是负向约束，其效果的优势来自于以下可能：

* 1. 和Llama模型的比较实验中，训练规模过小，Llama没有完全被训练。线索是Llama 2的Loss scope图

* 1. 和StableLM 3B的比较实验中，。。。。

从这个角度来看，BitNet b1.58给模型加上了强inductive bias，但是性能没有明显下降是合理的。

但是，这不能保证随着训练规模的进一步加大，这个inductive bias是否会变成性能上升的瓶颈。

这个瓶颈的强度和位置，决定了我们为了效率的提升能做到哪一步。

应用上：

在模型普遍难以完全训练的前提下，

对于相对简单的任务，使用BitNet b1.58已经有足够的收益。

# OneBit

## 背景

介于BitNet和BitNet b1.58中间的工作；

在BitNet W1A16的基础上改进，引入两个全精度的向量，已保持较高的性能。

因此，相比于BitNet:

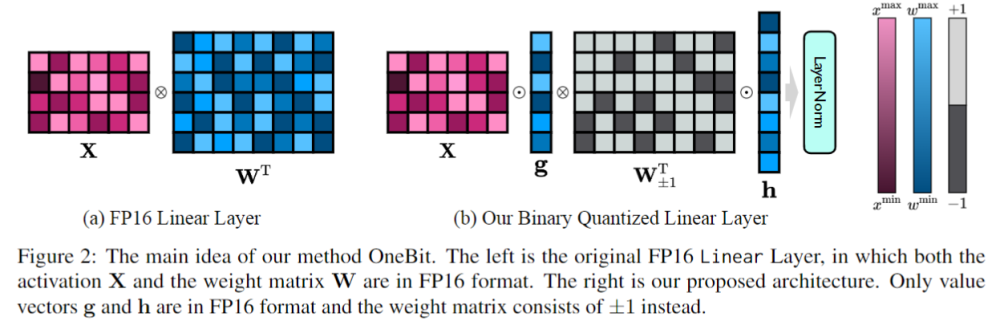
1. 内存需求略有上升，几乎不变

1. 推理时间（延时），大幅上升，逼近全精度模型

1. 性能提升，训练稳定性提升

1. **可以使用已训练好的全精度模型进行初始化**

## OneBit的结构



按照图中的计算顺序，无需在内存中保存一个W尺寸的FP16精度的矩阵，进而实现了内存需求降低。

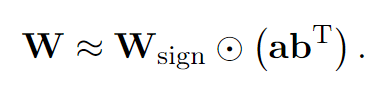
## OneBit的初始化

如上图所示，OneBit的核心是将FP16的权重矩阵W分解为一个1-bit矩阵和两个FP16的向量g/h。

1. 具体而言，假设存在一个已经训练好/经过训练的FP16模型，将权重矩阵分解为符合矩阵和值矩阵



1. 对值矩阵采用秩1近似（SVD[7]或NMF实现）



1. 经过变换，得到



和上图的对应关系是：

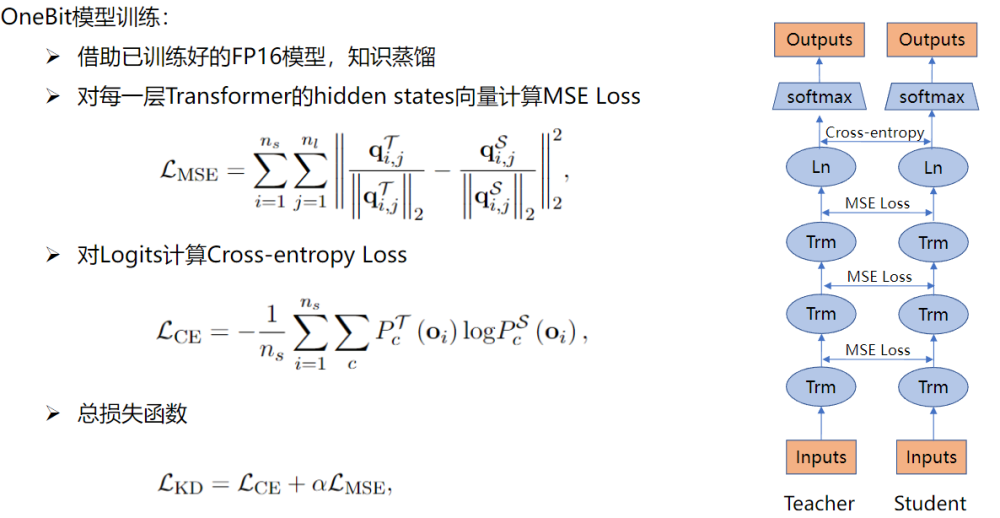
为什么不省略步骤1，直接对权重矩阵进行秩1近似呢？

如果直接对矩阵进行秩1近似，抛弃，将导致权重的近似误差增大，性能下降，得不偿失。

数学证明见 Appendix A

## OneBit的进一步训练：知识蒸馏

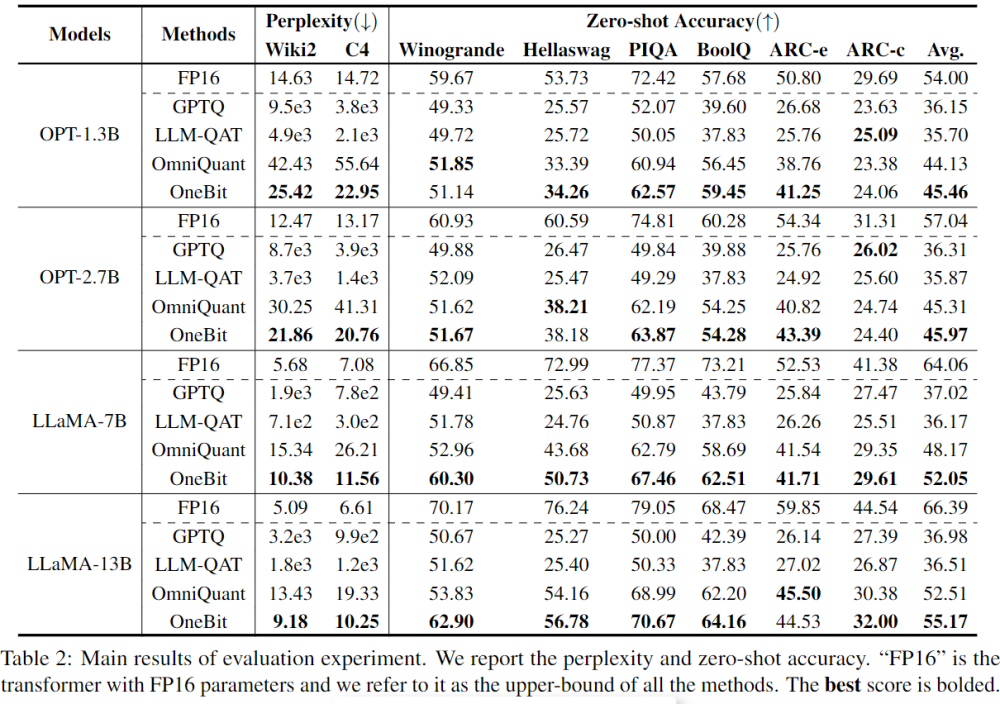
在初始化之后，借助已训练好的全精度模型，做知识蒸馏。

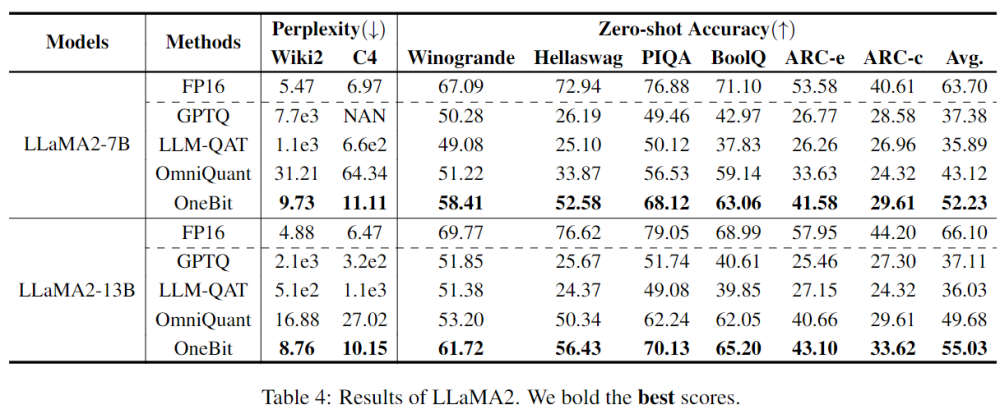




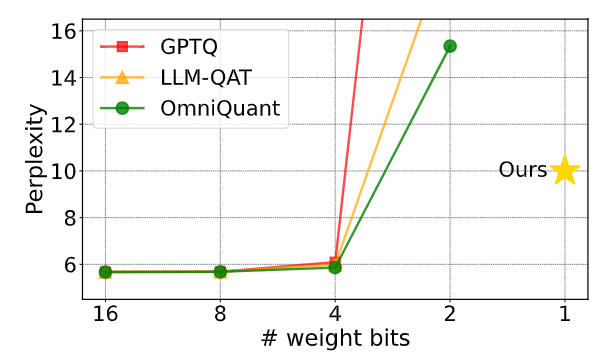
## OneBit实验结果

困惑度和标准测试集上的性能

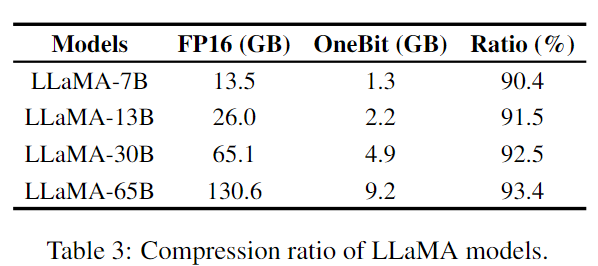




和其他量化工作的直观性能比较：



和FP16模型的内存需求比较，降低90%以上



## OneBit实验结论

* 作为一种QTA方法，在2bit以下时，仍然可以保持较高性能（83%）

* 模型可压缩90%，仍然呈现Scaling Law

* 随着模型参数规模的增大，压缩率提升，性能接近FP16模型

# 参考文献

[1] BitNet: Scaling 1-bit Transformers for Large Language Models

[2] Binary nerual network

[3] Ternary weight networks

[4] Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation

[5] The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits

[6] OneBit: Towards Extremely Low-bit Large Language Models

[7] [SVD和低秩矩阵近似（Low-rank Matrix Approximation）的数据压缩](https://zhuanlan.zhihu.com/p/447385674)

[8] <https://huggingface.co/blog/hf-bitsandbytes-integration>