why quantization

Scaling Law

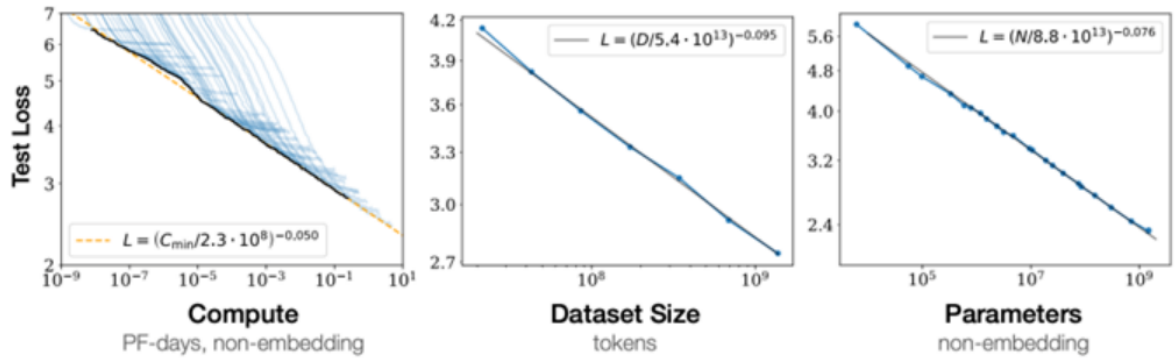
来自openai2020年的论文Scaling Laws for Neural Language Models

Having a sense of the capabilities of a model **before training** *can improve decisions around alignment, safety, and deployment.*
— GPT4 Technical Report

Scaling Law的本质是：模型最终效能可以藉由模型参数量、Dataset大小、训练计算量来预测。**因此我们可以藉由小模型、少量Data的训练，来快速验证我们的想法。**

定义

模型的性能依赖于模型的规模，包括参数量、数据集大小、计算量，模型的效果会随着三者的指数增加而平稳提高。

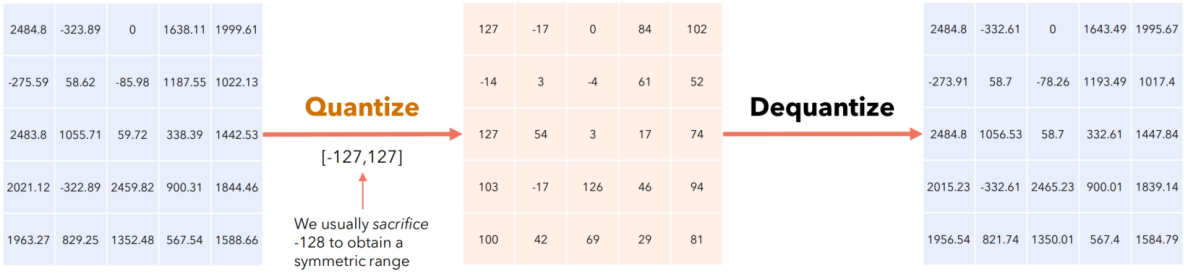


如图所示，模型的loss随着模型规模的指数增加而线性降低。这意味着模型的能力可以根据这三个变量估计。

对于Decoder-only的模型，计算量C(Flops)、模型参数量N（non-embedding parameter count）、数据大小D(token数)三者满足: C≈6ND (小模型不准)。模型的计算量C一定后，模型的性能即精度就基本确定。它的决策变量只有N和D，跟模型的具体结构诸如层数、 深度、 attention头个数（宽度）基本无关。相关性非常小，性能（即test loss）在2%的区间内。

因此，虽然lora可以一定程度上减少参数量，但是因为scaling law的存在，无法一味减少参数量，因此出现了Qlora。

what is quantization



量化是指将连续的无限值映射到一组较小的离散有限值的过程。在LLM的上下文中，它指的是将模型的权重从高精度数据类型转换为低精度数据类型的过程。

对模型的量化基于一个共识，那就是复杂的、高精度表示的模型在训练时是必要的，因为我们需要在优化时捕捉微小的梯度变化，然而在推理时并没有必要。也就是说，网络中存在很多不重要的参数，或者并不需要太细的精度来表示它们。另外，实验证明神经网络对噪声鲁棒，而量化其实也可看作是噪声。

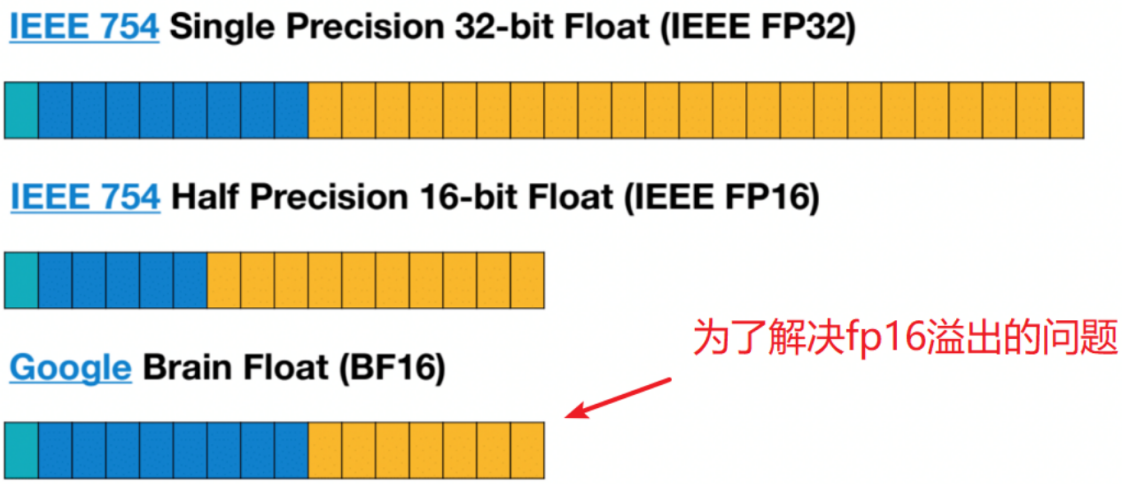
量化最核心的挑战是如何在减少表示精度的同时不让模型的准确度掉下来，即在压缩率与准确率损失间作trade-off。

浮点数的表示

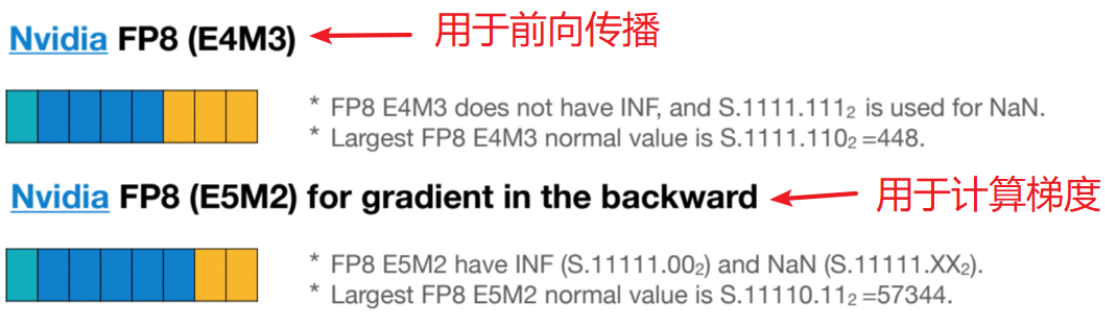
· **FP32（单精度）**使用 32 位来表示数字：1位表示符号，8位表示指数，其余 23 位表示有效数。虽然 FP32 提供高精度，但其缺点是计算量和内存占用量较高。

· **FP16（半精度）**使用 16 位来存储数字：1位用于符号，5位用于指数，10位用于有效数。尽管这使其内存效率更高并加速计算，但范围和精度的降低可能会导致数值不稳定，从而可能影响模型的准确性。

· **BF16**也是一种 16 位格式，但其中一位用于符号，*8位*用于指数，*7位*用于有效数。与 FP16 相比，BF16 扩大了可表示范围（和FP32一样），从而降低了下溢和上溢风险。尽管由于有效位数较少而导致精度降低，但 BF16 通常不会显着影响模型性能，并且对于深度学习任务来说是一个有用的折衷方案。



更进一步地，还有FP8甚至是FP4。实际上，对于大模型最常见的就是8bits量化(FP8/INT8)和4bits量化(FP4/NF4/INT4)。



因为训练过程中比较容易溢出。所以为了防止训练不稳定，只在计算MLP的过程中才会用到fp8，而在其他层依然使用bf16（layer norm甚至会用到fp32）。

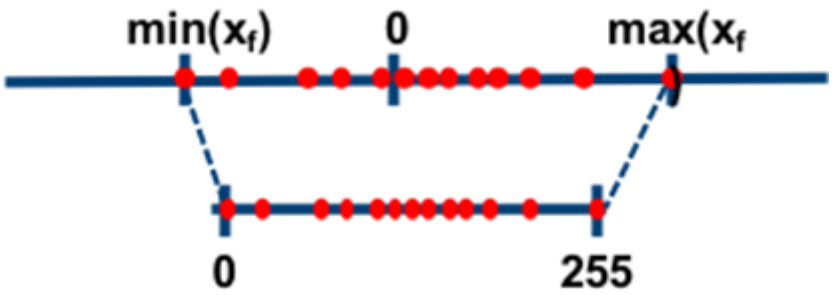
How to Quant

参考：闲话模型压缩之量化（Quantization）篇 stochastic rounding-CSDN博客

对于典型的浮点到整型的量化，其本质上是将实数域中的某一段映射到整数。

非对称（Asymmetric）量化

该方法中对原值域和量化后值域都不要求关于0对称。



线性量化的一般形式：$q = round(s * x + z)$

其中的x和q分别为量化前和后的数，s称为scaling factor，z称为zero point。
这个zero point即是原值域中的0在量化后的值。在weight或activation中会有不少0（比如padding，或者经过ReLU），因此我们在量化时需要让实数0在量化后可以被精确地表示。

为使量化后在指定位数整数表示范围内（如n位），这里scaling factor可以取：

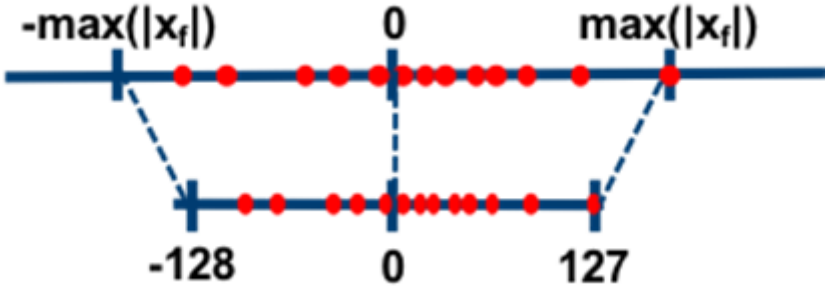
$\frac{2^n-1}{\max_x - \min_x}$

其中min_x 和 max_x 分别为量化对象（如weight或activation）的dynamic range下界与上界。那么问题来了，如果这个dynamic range很广，或者有些很离奇的outlier，那这样做就会便得比特位被浪费在那些并不密集的区域，导致信息的丢失。因此，还有种做法就是对dynamic range先做clip，把那些含信息少的区域切掉。即：

$q = \text{round}(s \cdot \text{clip}(x, \alpha, \beta) + z)$

其中α和β分别为clip的上下界，作为量化参数clipping value。这个取值也是门艺术，大了浪费比特位，小了把就把太多有用信息“切”掉了。

对称（Symmetric）量化



对称量化中的zero point设为0，是非对称量化的特例。由于计算机内部使用补码表示，其数值范围并不是关于0对称的。比如，8位有符号整型的表示范围是[-128, 127]，为实现真正对称，需要截取为[-127, 127]。这种方式保证了数值范围的对称性，避免了偏差。

对称量化的一个优点是计算相对简单。比如，对于量化后的乘法，只需要将原值乘法结果再乘以一个缩放因子：

$q = \text{round}(s \cdot \text{clip}(x, \alpha, \beta) + z)$

当零点 (z = 0) 时：

$q = s_x x \times s_y y = s_x s_y xy$

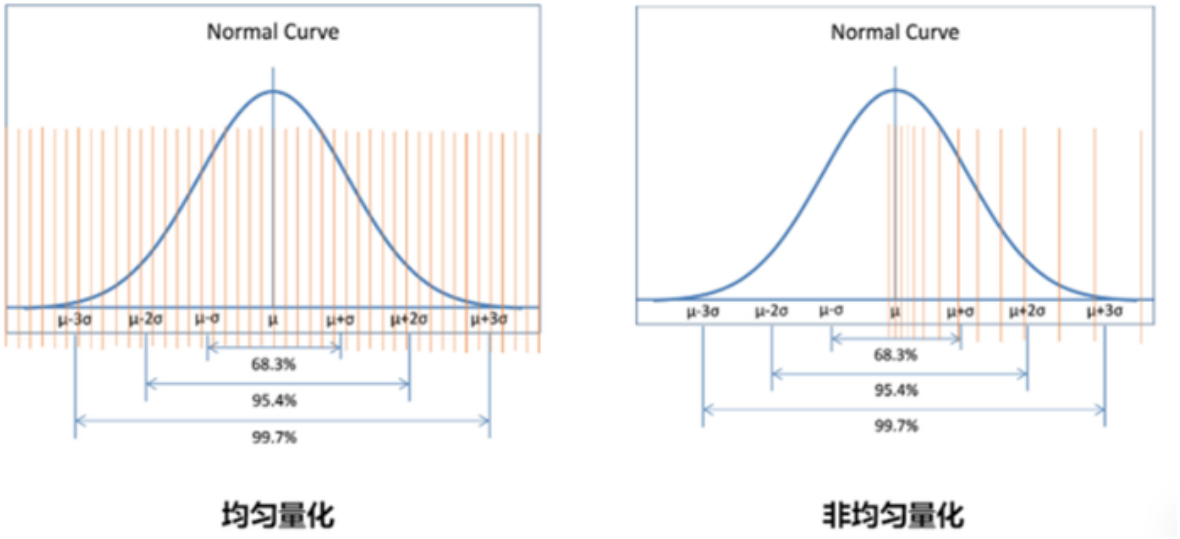
而对于非对称量化：

$$(s_x x + z_x) \times (s_y y + z_y) = s_x s_y xy + s_x x z_y + s_y y z_x + z_x z_y$$

虽然复杂度增加，但能更好地利用比特位表示范围。

总的来说，对称与非对称量化各有利弊，得看用在什么地方。如实际中weight的分布通常是关于0对称的高斯分布，那比较适用用对称量化。而ReLU后的activation是没有负数的，因此适用于非对称量化。另外，如果非对称量化中整个值域都落在正区间，那就可以用无符号整型表示，否则用有符号整型。

还有一种分类方式是**均匀（Uniform）量化与非均匀（Non-uniform）量化**：



均匀量化 **非均匀量化**

最简单就是让量化的等级间距离相等，这类量化称为均匀量化。但直觉上这样会有更多的信息丢失，因为一般来说dynamic range中肯定有些区域点密集，有些稀疏。

相应地，非均匀量化是指量化等级间不等长，如log quantization。再高级一点，那就是通过学习来得到更大自由度的映射（可通过查找表表示）。直觉上，非均匀量化似乎能达到更高的准确率，但它的缺点是不利于硬件加速。

按量化过程中是否需要进行训练可分为两类：

- **PTQ（Post-training quantization）**：顾名思义，就是在模型训练后做的量化。又可细分为两种：
  - **需要校准数据的PTQ**：利用少量未标注的数据集来统计量化参数，确保量化后的模型性能接近原始模型。
  - **完全不需要数据集的PTQ**：在没有任何数据的情况下，通过统计推断或预定义参数进行量化，适用于无法获取训练数据的情况。
- **QAT（Quantization Aware Training）**：在训练过程中引入量化操作，使模型在训练时就适应量化后的数值表示。
  - **从头训练**：从零开始训练模型，考虑量化影响。
  - **微调**：在全精度模型基础上进行少量训练，引入量化操作。

  低位数量化（如4位及以下）：由于信息丢失较多，通常需要训练介入，通过QAT来保持模型性能。

以**量化参数的共用范围**（或者说，粒度）来分，可分为：

- **Per-axis/per-channel**：对张量（tensor）的单个轴或通道（channel）使用独立的量化参数。例如，对于卷积层的权重，每个通道（channel）使用单独的量化参数。
- **Per-tensor/per-layer**：每个张量（tensor）或每一层（layer）有独立的量化参数。对于卷积层或全连接层来说，这意味着每层有单独的量化参数。

- **Global**：整个网络使用相同的量化参数。一般来说，对于8位量化，全局量化参数影响不明显，但到更低精度，就会对准确率有较大影响。