大模型微调相关知识

1. 背景概念定义：

- FLOPS 注意全部大写 是floating point of per second的缩写，意指每秒浮点运算次数。用来衡量硬件的性能。
- FLOPs 是floating point of operations的缩写，是浮点运算次数，可以用来衡量算法/模型复杂度。

2. 大模型理论最小计算量

- FLOPs = 6 * 模型的参数量 * 训练数据的 token 数。
    - 这里的 6 就是每个 token 在模型正向传播和反向传播的时候所需的乘法、加法计算次数
- 参考链接：
    - https://medium.com/@dzmitrybahdanau/the-flops-calculus-of-language-model-training-3b19c1f025e4
    - https://github.com/MrYxJ/calculate-flops.pytorch/blob/main/README_CN.md

3. 训练内存

    a. 模型参数（float32）：模型参数（B） * 4

    b. 反向梯度（float32）：等于参数量*每个梯度参数所需内存，模型参数（B） * 4

    c. 优化器参数：如果用最经典的 Adam 优化器，它需要用 32 位浮点来计算，否则单纯使用 16 位浮点来计算的误差太大，模型容易不收敛。因此，每个参数需要存 4 字节的 32 位版本（正向传播时用 16 位版本，优化时用 32 位版本，这叫做 mixed-precision），还需要存 4 字节的 momentum 和 4 字节的 variance，一共 12 字节。如果是用类似 SGD 的优化器，可以不存 variance，只需要 8 字节。模型参数（B） * 12（GB）

    d. 正向传播状态（计算最小训练内存，极限状态下每层都从头开始计算，可以假定忽略）

        i. 正向传播的中间状态（activation）是反向传播时计算梯度必需的，而且跟 batch size 成正比。Batch size 越大，每次读取模型参数内存能做的计算就

越多，这样对 GPU 内存带宽的压力就越小。但是，正向传播的中间状态数量是跟 batch size 成正比的，GPU 内存容量又会成为瓶颈。

ii. 重计算方法：正向传播中间状态占的内存太多了，可以用算力换内存，就是不存储那么多梯度和每一层的正向传播的中间状态，而是在计算到某一层的时候再临时从头开始重算正向传播的中间状态，这样这层的正向传播中间状态就不用保存了，Reducing Activation Recomputation in Large Transformer Models

4. 大模型训练耗时估计：

   a. 模型参数量和训练总tokens数决定了训练transformer模型需要的计算量。给定硬件GPU类型的情况下，可以估计所需要的训练时间。给定计算量，训练时间（也就是GPU算完这么多flops的计算时间）不仅跟GPU类型有关，还与GPU利用率有关。计算端到端训练的GPU利用率时，不仅要考虑前向传递和后向传递的计算时间，还要考虑CPU加载数据、优化器更新、多卡通信和记录日志的时间。一般来讲，**GPU利用率一般在**0.3~0.55**之间**。

   b. 对于每个token，每个模型参数，进行2次浮点数计算。使用激活重计算技术来减少中间激活显存需要进行一次额外的前向传递，因此前向传递 + 后向传递 + 激活重计算的系数=1+2+1=4。使用**激活重计算**的一次训练迭代中，对于每个token，每个模型参数，需要进行2*4=8次浮点数运算。在给定训练tokens数、硬件环境配置的情况下，训练transformer模型的计算时间为：time = (8 * tokens * model_size) / (GPU 数量 * GPU 峰值 flops * GPU 利用率)

   c. 1TFLOPS就是每秒计算1万亿次浮点运算，等于10^12

5. 常见显卡算力峰值（FP16 精度）

   a. A100 80GB PCIe：312 TFLOPS

   b. A10 24GB PCIe：125 TFLOPS

   c. A800 80GB PCIe: 312 TFLOPS

6. 训练模型参数量与训练数据量的统计

   a. 2020，OpenAI：Scaling Laws for Neural Language Models

   b. 2022，DeepMind，Training Compute-Optimal Large Language Models

      i. 简略版：每个参数需要大约 20 个文本token

7. epoch 的设置。

a. epoch，指的是模型训练过程中完成的一次全体训练样本的全部训练迭代。在传统的深度学习模型训练中，epoch的设置可以类似于传统机器学习模型训练的迭代次数，一般认为越多的epoch会让模型拟合得越好，在LLM时代，很多模型的epoch只有1次或者几次。例如，2022年谷歌的PaLM模型，其训练的epoch数量只有1。而MetaAI训练的LLaMA模型，在不同数据集上训练的epoch设置都是1-2

b. 重复Token对模型性能的影响：**To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis**

c. 模型参数规模的增长与模型需要的tokens数量基本是呈线性的

d. 多轮epoch的训练会降低模型性能

e. 更大规模的数据集会缓解重复epochs对模型性能下降的影响

f. 提高数据集的质量也无法挽救重复训练带来的过拟合

g. 参数数量和FLOPs在重复训练上的影响

h. 小计算量模型的过拟合趋势与大计算量的差不多

i. Dropout是一个被大语言模型忽视的正则技术，虽然慢，但是可以降低多epochs的影响，在训练过程中逐渐使用dropout是有效的策略，dropout对不同规模模型的影响不同

j. 总结：根据前面的实验我们知道，如果在tokens数量一定的数据集上做多epochs的模型训练，会影响模型的性能，降低模型的效果。这在预训练和下游任务都会产生影响。但是，随着模型的发展，高质量数据集的tokens数将很快用完。而采用正则技术虽然会影响模型训练效率，但是会降低这种影响。

8. token 和存储之间的关系

Model	Baichuan-7B	LLaMA	Falcon	mpt-7B	ChatGLM	moss-moon-003
Compress Rate	0.737	1.312	1.049	1.206	0.631	0.659
Vocab Size	64,000	32,000	65,024	50,254	130,344	106,029

对 Baichuan 1 的推理优化迁移到 Baichuan 2

由于很多用户在 Baichuan 1 (Baichuan-7B, Baichuan-13B)上做了很多优化的工作，例如编译优化、量化等，为了将这些工作零成本地应用于 Baichuan 2，用户可以对 Baichuan 2 模型做一个离线转换，转换后就可以当做 Baichuan 1 模型来使用。具体来说，用户只需要利用以下脚本离线对 Baichuan 2 模型的最后一层 lm_head 做归一化，并替换掉 `lm_head.weight` 即可。替换完后，就可以像对 Baichuan 1 模型一样对转换后的模型做编译优化等工作了。

- 1token对应 1.4-1.7汉字，一个汉字对应 2 个字节（utf8），那么 1B 的 token 对应约 3G 存储汉字