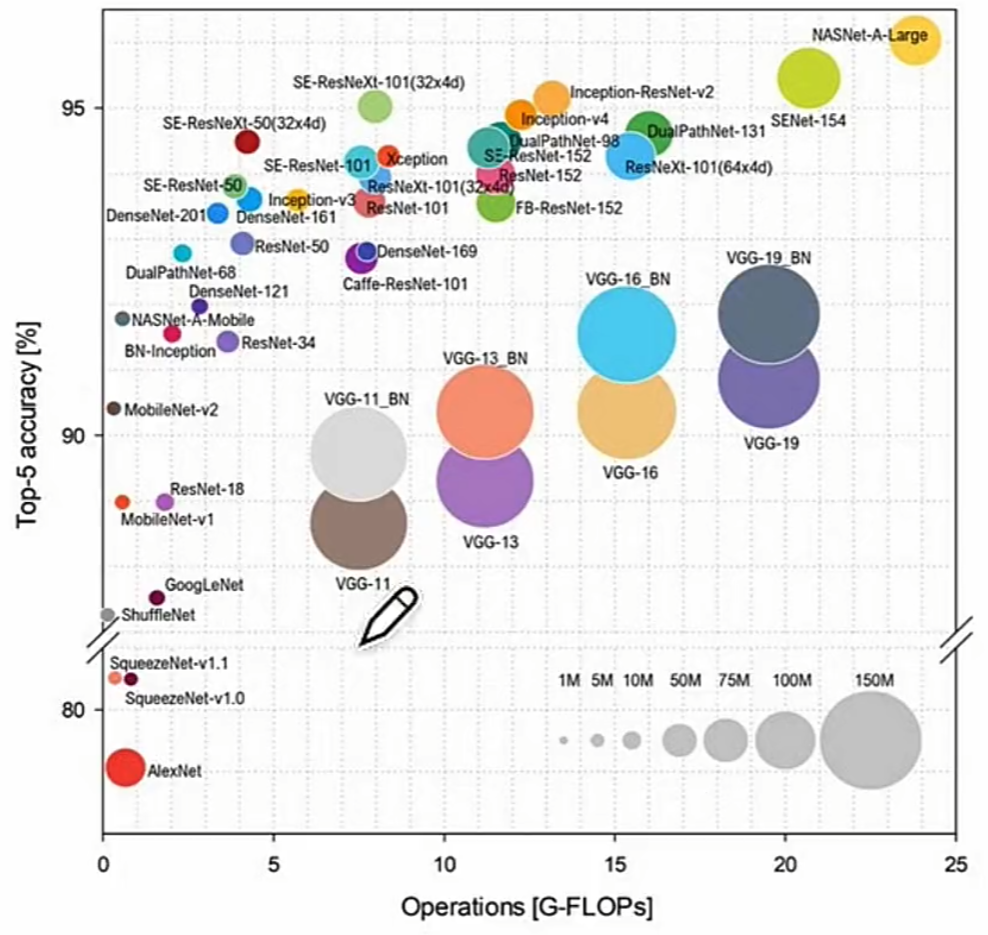


2021年8月份 由李菲菲和100多位学者联名发布了一份200多页的研究报告。综述了大模型预训练模型面临的机遇与挑战:

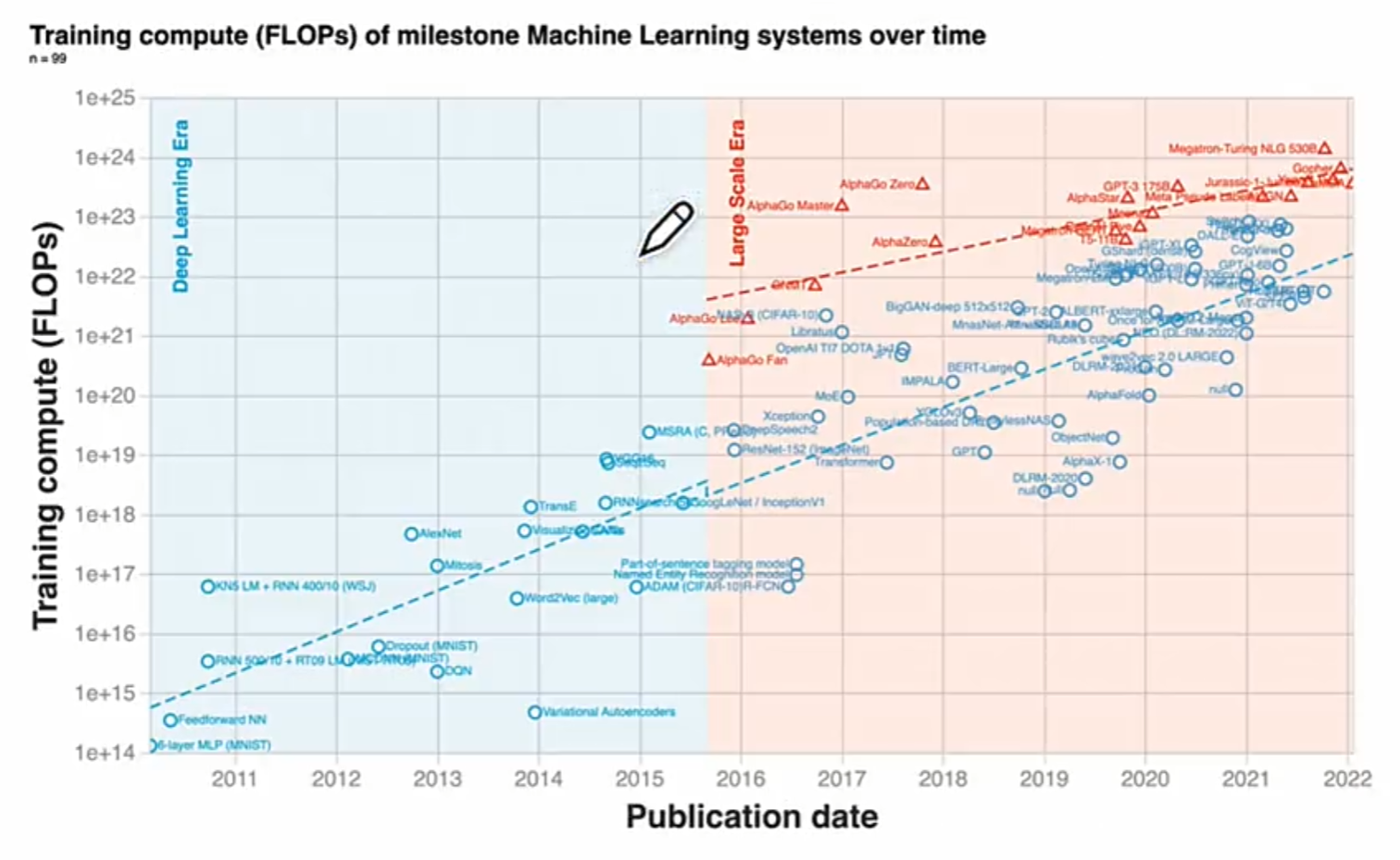
AI专家将大模型命名为foundation model

特性：涌现、同质化



2017年 transformer结构的首次提出使得深度学习模型参数量突破了一个亿

到了Bert网络模型的提出，参数量首次超过三亿规模，gpt-3的模型规模超过百亿。盘古超过了千亿，谷歌的switch transformer的问世一举突破了万亿规模

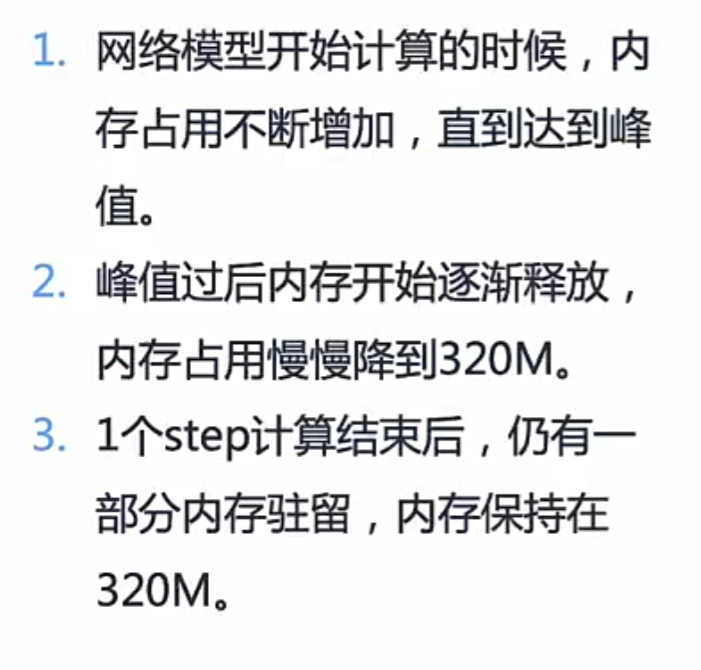


现在的AI模型处于手工作坊式，然后大模型提供了另外一种解决方案：预训练大模型+下游任务微调的解决方式，大模型预训练可以从大量标记和没有标记的数据中获取知识，通过将知识存储在大量参数当中，并对特定任务进行微调，很好的扩展了模型的泛化能力

大模型自监督学习方法，可以减少数据标注的成本，使得小样本的学习也能达到比以前更好的能力，并且参数规模越大，优势越明显，避免了开发人员进行大规模的训练。使用小样本就可以训练自己的模型，极大降低开发的成本。

模型精度的提升主要依赖于网络模型结构上的变更，但是随着神经网络结构设计的技术逐渐成熟并收敛，想要通过优化神经网络的结构来打破精度的局限是非常难的，但近几年随着数据规模和模型规模不断增大，模型的精度得到了进一步的提升，实验表示数据规模和模型规模的增大，确实能突破精度的局限。

以resnet50一轮迭代为例



静态内存主要是由模型参数优化器状态信息组成，在一个神经网络计算过程中，卷积或者全连接层中，权重将会长期保存。另外一些优化器会存储优化器的动量等信息。

动态内存：神经网络在计算的过程当中，每一层的向前输出of需要保存下来在反向传播的时候所使用。再而反向传播的过程当中需要记录下权重的梯度wg给优化器使用。另外梯度的输出og在反向传播作为上一层的输入，最后还有算子计算的临时变量等。这些数据积累下来会引起巨大开销。

静态内存与动态内存相互独立相互制约，所以要同时优化。

优化方式

# How to Train Really Large Models on Many GPUs?

DP最朴素的方法是复制相同的模型权重参数到多个workers上，给每个worker一部分数据来同时处理。如果模型size大于单个GPU节点内存时，这种方法是不能work的。GeePS (Cui et al. 2016) 提出了将暂时不用的模型参数卸载回CPU上。这种数据交换传输通常在后端进行，不会干扰训练。在每个mini batch结束后，workers需要同步梯度或参数，以保证学习效率，有两种主流的同步方法。

模型并行（Model parallelism，MP）用于解决模型权重不能放在单个节点的情况，计算和模型参数被分片到多台机器进行处理。和DP不同的是，DP中每个worker都有一个模型的完整副本，而MP只在一个worker上分配部分模型参数，因此对内存和计算的需求要小很多。

深度神经网络通常包含一个堆叠层，如果逐层拆分将连续的小层分配到工作层分区，操作起来并不难，但通过大量具有顺序依赖性的Workers来运行每个数据batch会花费大量的等待时间，计算资源的利用率也严重不足。

PP结合了MP和DP，来减少无效的时间。主要的思路是：将mini batch分割成多个micro batches，每个相同stage的worker同时处理一个micro batches。这种传输的调度方式和梯度被聚合的方式在不同的方法中又有区别。workers的数量，也被称为“pipeline depth”。

在 GPipe (Huang et al. 2019) 方法中，多个微批次处理结束时会同时聚合梯度和应用。同步梯度下降保证了学习的一致性和效率，与worker数量无关。如图3所示，“Bubble”仍然存在，但比图2少了很多。

GPipe论文表明，如果微批次的数量超过分区数量4倍（m>4d），则“Bubble”开销几乎可以忽略不计。

GPipe在吞吐量上可以取得和设备数量相近的线性加速，尽管它并不能总是保证模型参数被均匀地分布在不同的worke节点上。

PipeDream是一套融合了流水线(Pipeline)，模型并行(model-parallism)以及 数据并行（data parallelism）三个机制的高效模型训练方案。在图像模型上测试可以达到1.45至6.76的加速比

由于 PipeDream 没有跨所有工作线程的批处理结束全局梯度同步，因此 1F1B 的本机实现很容易导致使用不同版本的模型权重的微批处理的向前和向后传递，从而降低学习效率。PipeDream提出了一些设计来解决这个问题：

权重存储：每个工作人员跟踪多个模型版本，并确保在给定一个数据批次的正向和后向传递中使用相同的权重版本。

垂直同步（可选）：模型权重的版本与激活和渐变一起在阶段工作人员之间流动。然后计算采用从前一个工作线程传播的相应存储版本。此过程使工作线程之间的版本保持一致。请注意，它是异步的，与GPipe不同。

PipeDream核心在于解决两个问题：(1) 对于一个给定的模型与分布式系统，如何划分任务（即哪个节点负责哪些layer，某些layer是数据并行还是模型并行）（2）对于流水线模型，如何避免流水线本身带来的训练的问题。

在训练运行开始时，PipeDream 首先分析模型中每一层的计算内存成本和时间，然后优化将层划分为阶段的解决方案，这是一个动态编程问题

后来提出了 PipeDream 的两种变体，以通过隐藏模型版本减少内存占用（[Narayanan 等人，2021](https://arxiv.org/abs/2006.09503) 年）。

PipeDream-flush 会定期添加全局同步的管道刷新，就像 GPipe 一样。通过这种方式，它通过牺牲一点吞吐量来大大减少内存占用（即仅维护模型权重的单个版本）。

PipeDream-2BW只维护两个版本的模型权重，其中“2BW”是“双缓冲权重”的缩写。它每批次生成一个新的模型版本，应该大于管道深度，新更新的模型版本不能立即完全替换旧版本，因为一些剩余的向后传递仍然依赖于旧版本。总共只需要保存两个版本，因此内存成本大大降低。

模型并行和管道并行都会垂直拆分模型，而张量并行（Tensor Parallelism，TP）是将张量运算的计算水平划分到多个设备上。

以Transformer为例。Transformer架构主要由多层MLP和自注意力块组成。Megatron-LM（Shoeybi et al.2020）采用了一种简单的方法来并行计算层内MLP和自注意力。

Narayanan et al. (2021)提出将管道、张量和数据并行与新的管道调度策略相结合，提出了一种名为PTD-P的新方法。该方法不仅在设备上能够定位一组连续的层（“模型块”），还可以为每个wokers分配多个较小的连续层子集块（例如，设备1具有第1、2、9、10层；设备2具有第3、4、11、12层；每个具有两个模型块）

每个batch中，微批次的数量应精确除以wokers数量（m）。如果每个worker有v个模型块，那么与GPipe调度相比，管道的“bubble”时间可以减少 v 倍。

MoE

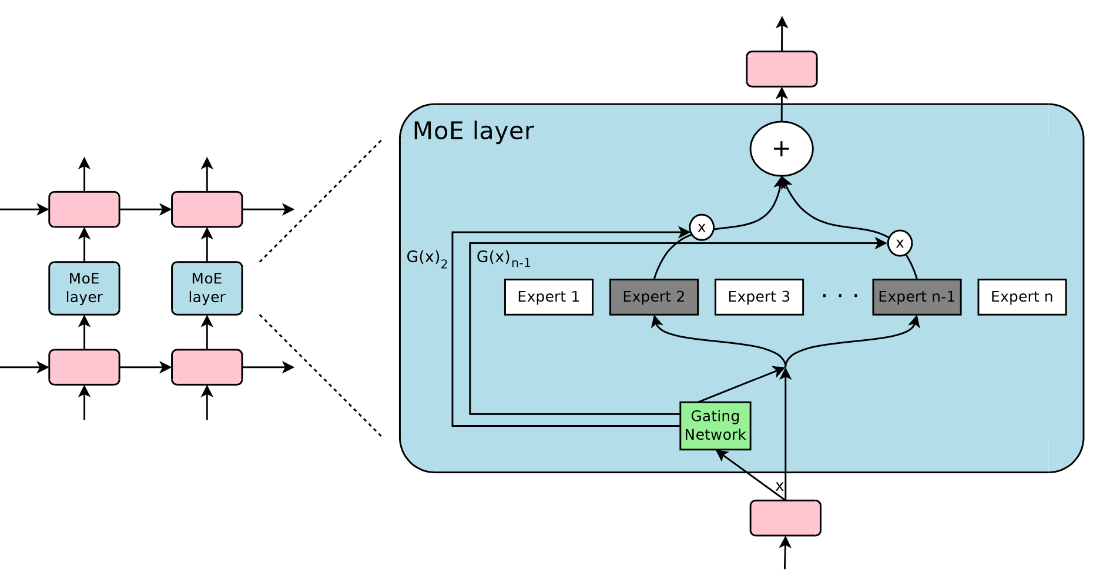
为了突破模型大小的限制，谷歌后来提出一种混合专家（MoE）方法，其核心理念是：集成学习，它假设集成多个弱学习器就会拥有一个强学习器。

在深度神经网络中，混合专家（MoE）通过连接多个专家的门机制（gating mechanism）实现集成（Shazeer等人，2017）。门机制激活不同网络的专家以产生不同的输出。作者在论文将其命名为“稀疏门控专家混合层（sparsely gated MoE）”。

仅一个MoE层包含：

1）前馈网络专家n；

2）可训练的门控网络G，通过学习n个专家的概率分布，将流量路由到几个特定的专家。

根据门控输出，并非每个专家都必须进行评估。当专家的数量太大时，可以考虑使用两层MoE。

Actiavation Recomputation

激活重新计算，也称“激活检查点”或“梯度检查点”（Chen et al，2016），其核心思路是牺牲计算时间来换取内存空间。它减少了训练层深层神经网络到的内存开销，每个batch只消耗额外的前向传递计算。

具体来说，该方法将层网络平均划分为d个分区，仅保存分区边界的激活，并在workers之间进行通信。计算梯度仍然需要在分区内层进行中间激活，以便在向后过程中重新计算梯度。激活重新计算的方法可以得出与模型大小有关次线性内存开销.

Mixed Precision Training

Narang & Micikevicius et al. (2018)介绍了一种使用半精度浮点（FP16）数训练模型而不损失模型精度的方法

其中涉及三种关键技术：

全精度权重复制：保持累积梯度的模型权重的全精度（FP32）复制。对于向前和向后的传递的信息做四舍五入至半精度处理，因为每次梯度更新（即梯度X学习率）太小，可能无法完全包含在FP16范围内。

缩放损失：放大损失以更好地处理小幅度的梯度（见图16），放大梯度以使其向可表示范围的右侧部分（包含较大的值）移动，从而保留可能丢失的值。

算术精度：对于常见的网络算法（如矢量点积、矢量元素求和归约），将部分结果累加到FP32中，然后输出保存为FP16。逐点操作可以在FP16或FP32中执行。

CPU Offloading

如果GPU内存已满，可以将暂时未使用的数据卸载到CPU，并在以后需要时将其读回（Rhu等人，2016）。不过，这种方法近年来并不太流行，因为它会延长模型训练的时间

Compression

模型权重在向前和向后传递的过程中会消耗大量内存。考虑到这两种传递方式会花费大量时间，Jain（Jain et al，2018）提出了一种数据编码策略，即在第一次传递后压缩中间结果，然后将其解码用于反向传播。

Jain和团队研发的Gist系统包含两种编码方案：一是特定于层的无损编码，包括 ReLU-Pool和 ReLU-Conv模式；二是有攻击性的有损编码，主要使用延迟精度缩减（DPR）。

Memory Efficient Optimizer

优化器也会消耗内存。以主流的Adam优化器为例，其内部需要维护动量和方差，这两者与梯度和模型参数比例基本相同。这意味着，我们需要节省4倍模型权重的内存。

为了减少内存消耗，学术界已经提出了几款主流优化器。与Adam相比，Adafactor（Shazeer et al.2018）优化器没有存储全部动量和变化，只跟踪移动平均数的每行和每列总和，然后根据这些总和估计二阶矩。

SM3（Anil et al.2019）优化器采用了一种不同的自适应优化方法。

ZeRO（Rajbhandari et al.2019）零冗余优化器节省了大型模型训练在两方面的内存消耗：

大多数内存由模型状态消耗，包括优化器状态（例如Adam动量和方差）、梯度和参数。混合精度训练也需要大量内存，因为除了FP16版本之外，优化器还需要保存FP32参数和其他优化器状态的副本。

剩余部分被激活、临时缓冲区以及不可用的碎片内存消耗。

ZeRO结合了ZeRO-DP和ZeRO-R两种方法。ZeRO-DP是一种增强的数据并行，避免了模型状态的简单冗余。它以动态的方式跨多个并行数据划分优化器状态、梯度和参数，以最小化通信量。ZeRO-R使用分区激活二次计算、恒定缓冲区大小和动态内存碎片，以优化剩余状态的内存消耗。