《Deep Gradient Compression: Reducing The Communication Bandwidth For Distributed Training》聚焦分布式训练中的通信带宽瓶颈问题，提出深度梯度压缩（DGC）技术，在不损失模型精度的前提下大幅降低梯度传输数据量，为大规模分布式训练（尤其是移动端联邦学习）提供高效解决方案。

**一、研究背景与问题**

**分布式训练的核心瓶颈在于**

（1）大规模分布式训练通过数据并行提升计算效率，但梯度交换的通信成本远超计算时间节省。

（2）联邦学习等移动端分布式场景中，设备存在带宽低、延迟高、连接不稳定、移动数据成本高的问题，传统梯度传输模式难以适配。

（3）此方法之前的梯度量化（如 1-bit SGD、TernGrad）、梯度稀疏化（如 Gradient Dropping）技术，或压缩比有限（如 TernGrad 仅 8×），或需修改模型结构（如 Gradient Dropping 需层归一化），且部分技术存在精度损失（如 Gradient Dropping 在机器翻译任务中 BLEU 分数降 0.3%）。

**二、深度梯度压缩技术（Deep Gradient Compression, DGC**）

DGC方法由以下六个部分组成：

1）梯度稀疏化（Gradient Sparsification）

2）局部梯度累积（Local Gradient Accumulation）

3）动量修正（Momentum Correction）

4）局部梯度裁剪（Local Gradient Clipping）

5）动量因子掩码（Momentum Factor Masking）

6）热身训练（Warm-up Training）

|  |
| --- |
| **Algorithm 1:** **Deep Gradient Compression for vanilla momentum SGD on node *k*** |
| **Input**: dataset *χ*  **Input:** minibatch size *b* per node  **Input:** momentum *m*  **Input:** the number of nodes *N*  **Input:** optimization function *SGD*  **Input:** initial parameters   1. **for**  **do** 3. **for****do** 4. Sample data *x* from *χ* 6. **end for** 7. **if** Gradient Clipping **then** 9. **end if**  12. **for do**     18. **end for** 19. *All-gather*: 21. **end for** |

**Algorithm 1**描述了DGC方法在节点*k*上的应用过程。

1至7行描述了节点局部梯度更新的过程。

8至10行是局部梯度裁剪操作。

11、12两行涉及到动量修正。

13至19行进行了梯度稀疏化操作，筛选出重要梯度，同时也保留次要梯度和其对应的动量项用于局部梯度累积。此外，这涉及到动量因子掩码操作，以应对延迟效应（Staleness Effect）。

20行是梯度聚合操作，得到全局梯度。

21行使用**vanilla momentum SGD**，通过全局梯度和历史模型参数更新模型参数。

1. **梯度稀疏化（Gradient Sparsification）**

梯度稀疏化的核心是“舍弃大部分微小的次要梯度，仅保留对模型参数更新由显著影响的重要梯度”。

注意，筛选梯度用到的阈值是取当前所有局部梯度绝对值的分位数。但实际实验中，这样的操作很费时间。因此，作者决定只取所有局部梯度的，再去取这些梯度的分位数。假如所取的阈值不太理想（即达到阈值的梯度多的超出预期），那就从已经筛选出来的梯度中取得更加准确的阈值。最后通过条件判断语句生成掩码用于筛选重要梯度和次要梯度。

1. **局部梯度累积（Local Gradient Accumulation）**

在分布式训练中，传统操作是每次迭代时，各节点都会传输自己的全部梯度。而局部梯度累积的定义是（1）节点每次仅传输绝对值超过阈值的重要梯度。（2）次要梯度则暂存本地，累积超过阈值之后方可传输。

局部梯度累积带来了稀疏更新间隔T的概念，它描述了节点在前轮迭代中因为未达阈值而不会传输梯度，只在第轮迭代时，梯度绝对值超过阈值时传输重要梯度参与模型参数更新。

此外，通过数学证明，我们可以发现：

在局部梯度累积下的模型参数的T步更新公式为：

而利用到增大批处理大小的训练方式，将batchSize从增大至。相应地，学习率也要放缩至，可得其模型参数的T步更新公式为：

(1)式和(2)式等价，说明局部梯度累积在模型训练精度上等效于增大批处理大小。与此同时，局部梯度累积还能再分布式训练场景下大幅减少通信带宽。

1. **动量修正（Momentum Correction）**

动量修正的目的是解决**梯度稀疏化**和**Momentum SGD**不兼容。

**Momentum SGD**的过程如下：

其中，是动量项。是动量折扣因子（Discounting Factor），通常取0.9，表示历史动量的折扣程度。设训练节点数为。表示第k节点上第t次迭代后的局部梯度。表示模型参数。表示学习率。

假如没有梯度稀疏化操作，**Momentum SGD**下的模型参数的T步更新公式为：

而由于梯度稀疏化的存在，模型参数的更新逻辑为：

结合稀疏更新间隔的定义可知，前轮几乎没有新梯度注入更新，而节点上一直进行局部梯度累积，于是有：

注：这边假设梯度的所有元素均达阈值，故经过掩码Mask筛选后仍与原先一致，简化推导。

所以得到：。

故模型参数更新公式变为:

又由于前T-1轮的模型参数几乎无实质上的更新，

所以有，

进而得到：

对比(4)式和(11)式，可发现：

由于梯度稀疏化，动量折扣因子的作用消失。这意味着最近历史动量和最早历史动量的影响权重相当。显然，这不符合**Momentum SGD**的设计初衷，反而会带偏模型训练。

简单来说，**Momentum SGD** 的模型参数更新依赖于完整梯度参与迭代。然而，DGC方法每次只传输重要梯度参与模型参数更新，导致上述的动量迭代偏离真实值，不利于模型正确训练。

动量修正原理是通过将局部梯度累积的对象从局部梯度变成动量项。如果我们将动量项视作梯度。那么**Momentum SGD**的模型参数更新公式与**Vanilla SGD**的模型参数更新公式无异。

早在先前我们证明了局部梯度累积对**Vanilla SGD**是有效的。同理可知，局部动量的累积也是对**Momentum SGD**有效的。因此，动量修正之后，我们的**Momentum SGD**操作如下：

通过这一修正，我们既能保证在**Momentum SGD**中，局部梯度累积在模型训练上的正确性，而且也成功将梯度稀疏化应用在**Momentum SGD**上。

1. **局部梯度裁剪（Local Gradient Clipping）**

局部梯度裁剪是为了适应分布式训练中节点的局部梯度累积，避免局部梯度爆炸而设计的梯度预处理操作。

与传统梯度裁剪不同，传统的梯度裁剪通常在所有节点梯度聚合后进行。而作者采用节点本地对累积的梯度裁剪。

它的操作流程是节点当前产生的梯度一定要先在本地完成裁剪，方可加入累积缓冲区。

裁剪的判断依据是以局部梯度的L2范数为标准，若，则。

1. **应对延迟效应（Staleness Effect）**

延迟效应（Staleness Effect）主要是指由于目前仅传输重要梯度，未达阈值的次要梯度有的会在节点本地一直累积，导致这些梯度的更新时机明显滞后于当前模型参数的迭代，进而出现干扰模型优化方向的现象。

论文是采取动量因子掩码和热身训练的方式应对。

动量因子掩码是如下操作：

这其实阻止了延迟梯度（delayed gradient）的对应动量累积，防止陈旧动量（stale momentum）干扰模型参数更新方向。

需要热身训练的原因在于训练初期，模型参数变化迅速，梯度分布更加多样且激进。若一开始梯度稀疏度过大，反而筛选不出有利于模型训练的梯度，不利于模型收敛。因此，我们要利用预热阶段较小学习率和较低的稀疏度，来缓解训练初期模型参数变化速度，减少被延迟传输的极端梯度的数量。