计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200300097

一种基于4Bit编码的深度学习梯度压缩算法

蒋文斌　符　智　彭　晶　祝　简
华中科技大学计算机科学与技术学院大数据技术与系统国家工程研究中心　武汉 430074

摘　要　对梯度数据进行压缩，是一种减少多机间通信开销的有效方法，如MXNet系统中的2Bit方法等。但这类方法存在一个突出的问题，即过高的压缩比会导致精度及收敛速度下降，尤其是对规模较大的深度神经网络模型。针对上述问题，提出了一种新的4Bit梯度压缩策略。该方法采用4个比特位表示一个具体的梯度值(通常为32位的浮点数)。相对于2Bit，该方法能够对梯度值进行更细粒度的近似，从而提高训练结果的准确率和收敛性。进一步地，根据网络模型每一层梯度特性的不同，选择不同的近似阈值，使得压缩后的数值更合理，从而进一步加快模型的收敛速度并提高最终准确率；具体地，兼顾操作的方便性和分布的合理性，根据每层梯度特性的不同，设置3组不同的阈值，以满足不同层梯度差异化特性的需求。实验结果表明，使用多组阈值的4Bit梯度压缩策略虽然在加速方面略逊于2Bit方法，但其准确率更高，实用性更强，能够在保持模型更高精度的前提下减少分布式深度学习系统的通信开销，这对于在资源受限环境下实现性能更好的深度学习模型非常有意义。

关键词：深度学习；梯度压缩策略；分布式训练
中图法分类号　TP183

4Bit-based Gradient Compression Method for Distributed Deep Learning System

JIANG Wen-bin，FU Zhi，PENG Jing and ZHU Jian
National Engineering Research Center for Big Data Technology and System，School of Computer Science and Technology，Huazhong University of Science and Technology，Wuhan 430074，China

Abstract　In order to reduce the communication overhead of distributed deep learning system，compression of gradient data before transmission is an effective method，such as 2Bit method in MXNet. However，there is a problem in this kind of method，that is，too high compression ratio will lead to decline in accuracy and convergence speed，especially for larger network models. To address this problem，a new gradient compression strategy called 4Bit is proposed. Four bits are used to represent a specific gradient value. Compared with 2Bit，this method can approximate the gradient more finely，thus improving the accuracy of training results and convergence speed. Furthermore，different approximation thresholds are selected according to the gradient characteristics of each layer of the network model，which makes the compressed values more reasonable，and finally improves the convergence speed and final accuracy of the model. The experimental results show that，although 4Bit is slightly lower than the 2Bit method in terms of acceleration，its accuracy is higher and practicability is better by using more bits and multiple thresholds. It is very meaningful to reduce the communication overhead of the distributed deep learning system while maintaining high accuracy by 4Bit.

Keywords　Deep learning，Gradient compression strategy，Distributed training

1　引言

随着大数据时代的到来，深度学习(Deep Learning，DL)[1-2]作为一类重要的数据分析、挖掘方法，在许多领域得到了广泛应用，如语音识别[3-4]、图像识别和检索[5-6]、自然语言理解[7-8]等。深度学习已成为学术界与工业界的一个重要研究热点[9]。

当前，许多深度学习任务都需要强大计算能力的支撑，GPU已经成为大多数深度学习系统的标准配置[10]。为获得更大计算能力的支持，分布式的深度学习系统成为了当前的主流。然而，多计算单元之间的通信开销问题，是分布式深度学习系统一个重要的、制约系统性能的瓶颈。

深度学习的训练过程会产生包括梯度数据在内的大量数据，这些数据需要传送给其他节点，以便完成训练任务。这类数据，尤其是梯度数据的通信开销很大，严重影响了分布式训练的效果，尤其是在带宽资源受限的环境下。因此，减少通信

到稿日期：2020-03-16　返修日期：2020-05-18　本文已加入开放科学计划(OSID)，请扫描上方二维码获取补充信息。
基金项目：国家自然科学基金(61672250)
This work was supported by the National Natural Science Foundation of China (61672250).
通信作者：蒋文斌(wenbinjiang@hust.edu.cn)

开销是提高深度学习系统效率的重要途径。

实际上,已经有不少研究者提出了各类方法来减少通信开销,包括 MXNet 中的 2Bit 方法[11]等。但这类方法存在一个问题,即过高的压缩比会导致精度及收敛速度下降,尤其是对较大的网络模型。针对上述问题,本文提出了一种新的 4Bit 梯度压缩策略,其采用 4 个比特位表示一个具体的梯度值。相对于 2Bit,该方法能够将梯度值进行更细粒度的离散,从而提高训练结果的准确率和收敛速度。

本文的研究目标是:针对深度学习分布式训练过程中的通信开销问题,使用梯度压缩的方法来减少多机之间的通信数据。

深度学习分布式训练的一般逻辑是:多个 worker 计算本地梯度,然后把所有 worker 的计算结果汇总到一个节点上更新权重(这里主要考虑数据并行[12]的模式,模型并行由于过于复杂的逻辑和过大的边际开销,在实际的深度学习系统中很少采用)。在多个 worker 传输本地计算的梯度到更新权重节点(即 server)的过程中,就存在大量的通信开销。梯度压缩方法则是把多个 worker 本地计算的梯度由一个占用较大内存的数据转化为占用较小内存的数据,然后再传输到 server,这样就可以降低每个数据占用的通信带宽,从而减小通信开销。梯度压缩虽然能减少通信量,但是数据在压缩过程中被相当程度地离散化后与原来的精确值有较大的误差,这就意味着精度的损失。如果过度压缩,那么最终训练结果的准确率可能会很低,模型就可能失去意义。因此,在进行梯度压缩时要注意保证训练结果的准确率。

本文的研究工作及主要贡献如下:

(1)设计并实现 4Bit 梯度压缩方法。将每个 32 位浮点表示的梯度压缩为 4 位的数据,其中 1 位表示正负,剩下 3 位表示数值,这样就能有 8 个不同阈值,可以尽可能地保证训练结果的准确率并压缩通信量。

(2)基于层级进行梯度特性分析。通过观察不同层的梯度数据的分布范围和特点,按层设置尽可能合理的梯度压缩阈值,从而使设置的阈值与原梯度的契合度更高,进而提高训练结果的准确率。

(3)实验验证与评估。将 4Bit 梯度压缩方法与 MXNet 框架下原有的 2Bit 梯度压缩方法及 baseline(即无梯度压缩的方法)进行对比,以验证所提方法的优势。

2 相关工作

分布式深度学习系统的通信效率一直是深度学习系统研究的核心问题,备受关注。一种改进的方法是通过提高网络硬件性能来提高通信效率。文献[13]从硬件角度提出了一种改进方法 iRDMA。该方法基于 RDMA 技术,采用参数服务器架构,优化了底层通信网络环境,提高了通信效率。但这类方法需要较大的成本投入。

在硬件条件一定的情况下,改进传输策略也是打破通信瓶颈的一种常用方法。

文献[14]提出了一种无等待误差逆传播(Wait-Free BackproPagation,WFBP)方法,即在后向传播时,无需等到所有层参数计算完成,可提前执行通信操作。不过,这种改进方法总体效果有限。该工作还提出了一种充分因子(Sufficient Factors)方法,通过将高维矩阵进行分解来降低数据的传输量,从而提高通信效率。但矩阵分解对矩阵有严格的要求,且需要较大的额外计算开销,从而限制了该方法的效率和应用范围。

一种更有效的方法是通过压缩需要传送的数据来减少通信开销,最终提高通信效率。TernGrad[15]是杜克大学提出的一种分布式深度学习系统通信数据压缩方法,用 2 个比特位表示一个梯度值,从而将梯度数据所占用的传输带宽空间压缩到了原来的 1/16。同样地,MXNet 系统中也集成了一个 2 比特(2Bit)的压缩方法[11],同样将所有的 32 位梯度数据压缩为 2 位数据再传输。该方法将每个浮点型的梯度值随机量化为 3 种值,即{11,10,00},分别表示设定的阈值、阈值的负数和 0 值。上述方法的共同特点是压缩效果明显,但准确率受到一定的影响。

如何在准确率和通信效率之间实现更好的平衡,是一个比较有挑战性的问题。本文提出的 4Bit 梯度压缩方法希望在这两者之间找到一个更好的平衡点。

3 4Bit 梯度压缩

3.1 4Bit 梯度压缩方法的设计

梯度的取值范围较大,而 2Bit 梯度压缩方法把所有梯度只分为 3 个值(阈值的正/负值和 0),这种过度压缩对准确率产生了较大的影响。因此,本文考虑设计一个 4Bit 梯度压缩方法,把梯度划分为 15 个值,这样可以在一定程度上保证训练结果的精度,也能有效地减少通信开销。

下面介绍 4Bit 梯度压缩方法的具体设计。

首先,设置 7 个阈值,分别用 X1－X7 表示,且将这 7 个数值按从小到大的顺序排列(以方便后续的数值分类压缩)。因为梯度可正可负,所以还需要再设置 7 个阈值的相反数 Y1－Y7 分别对应 X1－X7 的值,即 Y1＝－X1,其他数类似。每一个阈值需要对应一个 4 位的压缩值。这里设置 X1－X7 对应的 4 位压缩值分别是 0x9－0xF;同样,设置 Y1－Y7 对应的 4 位压缩值分别是 0x1－0x7 这 7 个值。另外,还有一个 4 位压缩值 0x0 表示阈值 0。阈值及其对应的压缩值分布如图 1 所示。

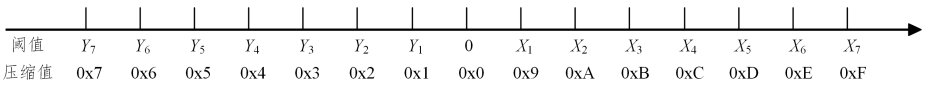

图 1 阈值分布图

Fig. 1 Threshold distribution map

4Bit 梯度压缩方法的具体步骤是：遍历每一个梯度，先判断其正负值，再将梯度分别和对应的正负阈值从小到大地进行比较，以将其压缩成靠近 0 的阈值，压缩后的 4 位数据是阈值对应的压缩值。例如，一个梯度值为 W，且 W 是正数，若 $W>X_1$ 且 $W<X_2$，则 W 压缩为 X_1。梯度为其他值或负数的情况类似。

worker 端把梯度压缩后 push 给 server，server 需要对压缩数据进行解压缩操作。梯度解压缩的方法与梯度压缩的方法相反。解压后的梯度保持原压缩梯度的顺序。

3.2 4Bit 梯度压缩方法的实现

本文提出的 4Bit 梯度压缩方法针对不同网络的不同层使用不同的梯度压缩方法，以加强梯度压缩的合理性。这里的不同梯度压缩方法主要指使用的梯度阈值不同，即需要有多组阈值。上述方案主要在 MXNet 的 Python 层中实现。考虑到一层网络的大部分梯度应该能较好地分布在设定好的阈值区间内，这里通过求平均值来初步决定这一层网络应该选择哪一组阈值。

在 Python 层，求出所有梯度之后、更新权重之前，需要通过各层的名字（“conv”“pool”或“fc”）来判断是否需要进行 4Bit 梯度压缩，如果进行 4Bit 梯度压缩，又应该选择哪一组阈值。

在 C++层，首先要获取多机传输时梯度应存放的地址。因为 kvstore 类的建立，传输的梯度的首地址固定，每一层的梯度首地址只需要用层的名字进行索引就可以得到。

在确定编码方案后，就要进行具体的梯度压缩操作。本文采用多线程的方式来实现。每个线程负责把 8 个原始梯度压缩成 1 个 32 位数据。压缩前先进行初始化，如算法 1 所示。主要步骤：(1)确定该线程压缩梯度的存放地址(第 1 行，out 是压缩后梯度存放的首地址，out_block_id 是偏移地址)；(2)初始化地址内的数据为 0(第 2 行)；(3)因为每个线程处理 8 个梯度，要将线程号乘 8 以得到处理梯度的起始编号，再获取结束梯度编号(第 3—4 行)；(4)遍历要处理的梯度，进行压缩(第 6—9 行)。

算法 1 参数初始化

输入：每一层的梯度矩阵 gradlist，压缩后梯度存放的首地址 out，线程编号 out_block_id

输出：压缩后的数值矩阵

```
1. compr_block=out+out_block_id;
2. *compr_block=0;
3. start =out_block_id≪3;
4. end=(start+8<=original_size) ? start+8:original_size;
5. block_ptr=reinterpret_cast < char * > (compr_block);
6. for i:= start to end
7.    curr_byte=block_ptr+((i-start)≫1);
8.    byte_position=i & 1;
9.    CompressGradient(i,gradlist,curr_byte,byte_position).
```

因为压缩值是 4 位的，所以用 char * 变量进行按位或来写数据。CompressGradient()函数是根据前面 4Bit 压缩算法的思想设计的，将梯度和阈值进行比较来确定压缩值，并将压缩值写在对应位置，如算法 2 所示。主要步骤：(1)将梯度与上一次压缩后的差值相加(第 1 行)；(2)判断梯度的正负(第 2 行)；(3)将梯度和阈值进行比较，将对应的压缩值写在对应的地址，并获取这次压缩的差值(第 3—24 行)。

算法 2 梯度压缩

输入：梯度序号 i，梯度矩阵 gradlist，写压缩值地址 curr_byte，位置 byte_position

输出：压缩后的数值矩阵

```
1. residual[i] += grad[i];
2. if (residual[i]>0) then
3.    if (residual[i]>threshold[6]) then
4.       residual[i]=residual[i]-hreshold[6];
5.       *curr_byte=posbits[position][6];
   ...
21.  else if (residual[i] > threshold[0]) then
22.     residual[i]=residual[i]-threshold[0];
23.     *curr_byte=posbits[position][0];
24. //负数情况类似
```

接下来，worker 把压缩后的梯度等数据 push 到 server 端。server 端接收到数据后进行解压缩操作。为了提高效率，解压缩的过程也使用了多线程，每个线程负责解压 1 个原始梯度，因为从 worker 传输过来的 32 位压缩梯度是由 8 个原始梯度压缩得到的，所以一个 32 位的压缩数据需要 8 个线程进行解压缩。

4 基于层级的梯度特性分析

在 4Bit 梯度压缩算法中，需要针对不同模型层设置不同的阈值组。为了让阈值组设置得更加合理，有必要分析不同网络的不同层的梯度特点。这里主要通过 3 个典型的例子分析，来探讨梯度分布特点及阈值组的设置方法。这 3 个例子分别是：Cifar10 数据集上的 Inception[16] 网络、Cifar10 数据集上的 ResNet-110 网络和 Cifar100[17] 数据集上的 DFN[18] 网络。

4.1 Inception 网络层级梯度特性的分析

Cifar10 数据集上的 Inception 网络的梯度折线图如图 2 所示。从图中可以看出，每一层的梯度大于 0.4 的占比都比较小，除了全连接层和第一层卷积层在部分周期内大于 0.4 的梯度稍多一点，其他层基本都没有大于 0.4 的梯度。

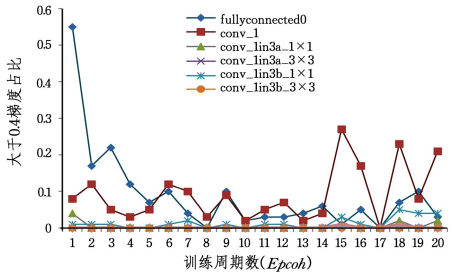

图 2 Cifar10 上 Inception 网络梯度大于 0.4 的比例图

Fig. 2 Gradients greater than 0.4 scale of Inception on Cifar10

进一步地，图 3 给出了 Cifar10 数据集上 Inception 网络若干层卷积层训练一个周期得到的梯度平均值。从图中可以看出，前面几个卷积层的梯度平均值是比较大的，越靠后的梯度平均值越小，从 conv_in5a 层开始就只有 0.05 了。这种前后平均值相差较大的情况，在设置阈值时要多加注意。

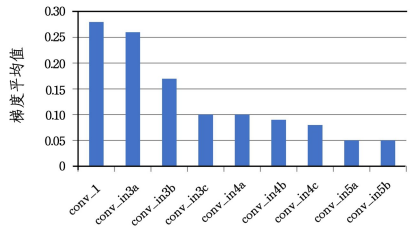

图 3 Cifar10 上 Inception 网络若干层的梯度平均值

Fig. 3 Gradient average values of Inception layers on Cifar10

4.2 ResNet-110 网络层级梯度特性的分析

Cifar10 数据集上 ResNet-110 网络各层的梯度分布特点如图 4 所示。从图中可以看出，每一层的梯度大于 0.4 的占比都比较小，除了全连接层和 conv0 卷积层外，其他层基本都没有大于 0.4 的梯度。

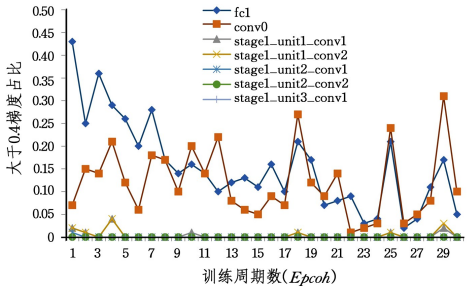

图 4 Cifar10 上 ResNet-110 梯度大于 0.4 的比例图

Fig. 4 Gradients greater than 0.4 scale of ResNet-110 on Cifar10

图 5 给出了若干层卷积层训练一个周期得到的梯度平均值。因为 ResNet-110 网络有 110 层，为分析方便，将其分成几个大模块。每个大模块中靠前的卷积层的平均值较大，如图中 stage1_unit1_conv1，stage2_unit1_conv1 和 stage3_unit1_conv1 层；与 Inception 网络一样，靠后的层特性的平均值都很小。

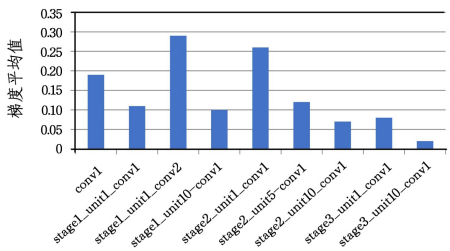

图 5 Cifar10 上 ResNet-110 网络若干层的梯度平均值

Fig. 5 Gradient average values of ResNet-110 layers on Cifar10

4.3 DFN 网络层级梯度特性的分析

图 6 给出了 Cifar100 数据集上 DFN 网络各层梯度分布的特点。从图中可以看出每一层的梯度大于 0.8 的占比都比较小。该数据集网络较前两者不同的是，g0_conv 的梯度值大于 0.8 的比例比全连接层的更大，其他层的梯度值也较大，不过所占比例不多。

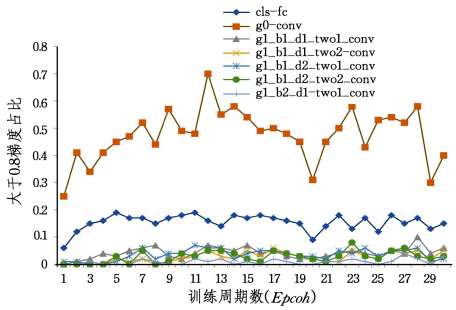

图 6 Cifar100 上 DFN 梯度大于 0.8 的比例图

Fig. 6 Gradients greater than 0.8 scale of DFN on Cifar100

可以看出，整体靠前的几个卷积层梯度的平均值非常大，尤其是 g0_conv 层；靠后的卷积层梯度的平均值相对小得多。这也突显了设置多组阈值的重要性：对于这类两极分化严重的网络，如果只用一组阈值，将难以保证梯度压缩后训练结果的准确率。

图 7 给出了 Cifar100 数据集上 DFN 网络若干层卷积层训练一个周期得到的梯度平均值。

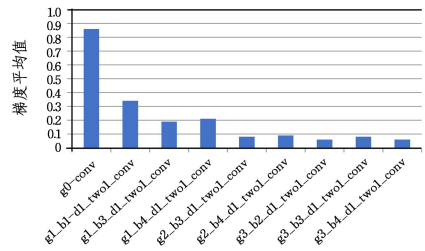

图 7 Cifar100 上 DFN 网络若干层的梯度平均值

Fig. 7 Gradient average values of DFN layers on Cifar100

4.4 多组阈值设置策略

从图 3、图 5 和图 7 可以看出，不同的网络其不同层的梯度平均值相差是比较大的，尤其是图 7 中浅层卷积层和深层卷积层两者的梯度平均值相差巨大。如果只使用一组阈值来进行 4Bit 梯度压缩，训练结果的准确率将难以得到保证。因此，设置多组阈值十分重要。对于每层网络，根据该层所有梯度的均值，选择均值附近的一组数作为阈值组，可较好地覆盖该层的所有梯度。与 4Bit 相比，2Bit 由于只有一个阈值，导致梯度值小于该阈值的梯度全部被量化为 0，损失了较多的梯度信息。

通过观察上述 3 组例子，可以看出梯度分布的一些特点：一般的网络层梯度值应该是偏小的，尤其是层数比较靠后的卷积层；大部分的梯度数值都小于 0.05，同时有一部分的梯度值在 0.1～0.4 之间，而更小的一部分梯度值大于 0.4，但这一部分数量少的梯度重要性更高，因此在阈值的划分上要有所倾向。本文考虑针对这类网络层，将阈值设置为{0.04，0.07，0.1，0.2，0.3，0.4，0.6}。当梯度的平均值在 0.1～0.5 之间时，选择使用该组阈值。

对于全连接层和第一层卷积层，一般情况下，其梯度多数是比较大的。虽然也有很大一部分的梯度小于 0.5，但是其梯度的平均值甚至能达到 0.8。为了满足这种梯度类型的网络层的需要，有必要设置一组值偏大的阈值组，我们取阈值组为{0.1，0.3，0.5，0.6，0.7，0.8，0.9}。当梯度的平均值大于 0.5 时，使用该组阈值。

另外，从前面的折线图可以看出，网络模型后面的卷积层整体的梯度可能都很小，甚至最大值小于 0.1。考虑到这类情况，再设置一组值较小的阈值组{0.01,0.03,0.05,0.06,0.07,0.08,0.09}。当梯度的平均值小于 0.1 时，使用该组阈值。

以上 3 组阈值分别覆盖了梯度分布的 3 种情况，能更好地适应不同层之间的差异性需求，从而提高训练结果的准确率。表 1 列出了阈值的具体设置情况。

表 1 阈值分布表

Table 1 Threshold distributions

	设置阈值	使用条件
第一组阈值	0,0.04,0.07,0.1,0.2,0.3,0.4,0.6	梯度均值为 0.1～0.5
第二组阈值	0,0.1,0.3,0.5,0.6,0.7,0.8,0.9	梯度均值大于 0.5
第三组阈值	0,0.01,0.03,0.05,0.06,0.07,0.08,0.09	梯度均值小于 0.1

5 应用测试与对比分析

5.1 测试用例及方法说明

实验测试主要在 5 种数据集及网络模型上进行，包括 Cifar10 数据集＋Inception 网络、Cifar100 数据集＋Inception 网络、Cifar100 数据集＋DFN 网络、Cifar10 数据集＋ResNet-110 网络和 ImageNet[19] 数据集＋ResNet-50 网络。

测试的环境是一个拥有 4 个服务器节点的集群，服务器之间使用万兆网互连。集群中服务器的 CPU 均为 Intel Xeon E5-2680 @ 2.40 GHz(2 颗/服务器)，GPU 版本为 NVIDIA Tesla P100(2 个/服务器)，内存为 256 GB。

实验分别测试 4Bit 梯度压缩策略、2Bit 梯度压缩策略和 baseline 无梯度压缩策略，并对结果进行分析。另外，分别在 4 个节点和 2 个节点上进行了测试，并比较几种方法的效果。

5.2 准确率的测试与对比

首先测试只用一组阈值和使用多组阈值的 4Bit 梯度压缩策略。为了验证 4Bit 梯度压缩策略中使用多组阈值能够有效地提高训练结果的准确率，我们测试了 Cifar10＋Inception 网络，分别用 2Bit 和 4Bit 只使用第一组阈值及 4Bit 使用 3 组阈值这 3 种方案进行训练的情况，具体结果如图 8 所示。

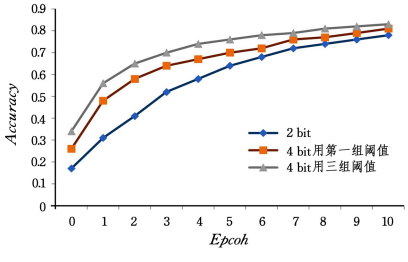

图 8 Cifar10 上 Inception 梯度压缩策略的准确率对比

Fig. 8 Accuracy comparison of gradient compression strategies (Inception on Cifar10)

从图 8 中可以看出，在 4Bit 梯度压缩策略中如果只使用一组阈值，训练结果的准确率就已经高于 2Bit 梯度压缩策略。当在 4Bit 梯度压缩策略中根据梯度的分布特性来选择更合适的阈值时，得到的训练结果的准确率更高。这说明使用多组阈值可以有效地提高准确率。后面的测试中，4Bit 梯度压缩策略均使用多种阈值进行压缩操作。

接下来分别给出使用多组阈值的 4Bit 策略、2Bit 策略和 baseline 无梯度压缩策略这 3 种方法对前面给出的 5 种网络的准确率测试结果。

首先是 Cifar10 数据集＋Inception 网络，训练周期为 30 个，学习率为 0.1，每个周期结束都调节学习率，学习率调节因子为 0.94。如图 9 所示，当使用 4 个节点训练时，在训练周期足够多后，4Bit，2Bit 和 baseline 策略最终的准确率都很接近。但是在前 20 个周期内，baseline 的准确率一开始就比较高，4Bit 策略和 baseline 比较接近，每个周期结束时准确率相差只有 5%左右；而 2Bit 梯度压缩策略的准确率在前 20 个周期内则与 baseline 相差较大。在使用 2 个节点训练时(见图 10)，可以得到相似的结论。

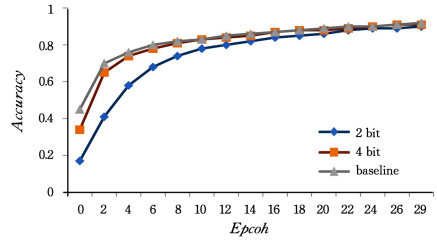

图 9 Cifar10 上 Inception 梯度压缩策略的准确率对比(4 节点)

Fig. 9 Accuracy comparison of gradient compression strategies (Inception on Cifar10) in 4-node

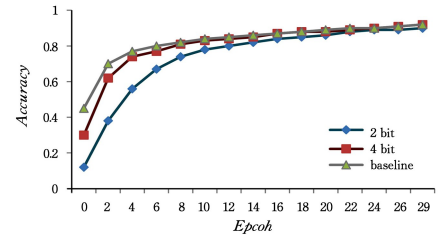

图 10 Cifar10 上 Inception 梯度压缩策略的准确率对比(2 节点)

Fig. 10 Accuracy comparison of gradient compression strategies (Inception on Cifar10) in 2-node

下面对其余几种数据集网络只给出 4 节点的情况(2 节点情况类似)。

图 11 给出了 Cifar100 数据集＋Inception 网络上的实验结果。从图中可以看出，与 Cifar10 数据集＋Inception 网络相比，在前 20 个周期内，3 种策略的准确率更接近一些，但 2Bit 策略的准确率明显低于 4Bit 策略和 baseline 的准确率。

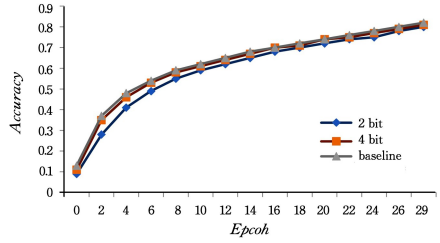

图 11 Cifar100 上 Inception 压缩策略的准确率对比(4 节点)

Fig. 11 Accuracy comparison of gradient compression strategies (Inception on Cifar100) in 4-node

图 12 给出的是 Cifar100 数据集＋DFN 网络的实验结果，同样可以看出 2Bit 策略的准确率比 4Bit 策略和 baseline 的低。图 13 给出的是 Cifar10 数据集＋ResNet-110 网络的实验结果，情况类似。

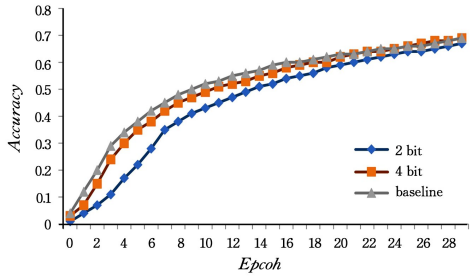

图 12 Cifar100 上 DFN 梯度压缩策略的准确率对比(4 节点)

Fig. 12 Accuracy comparison of gradient compression strategies (DFN on Cifar100) in 4-node

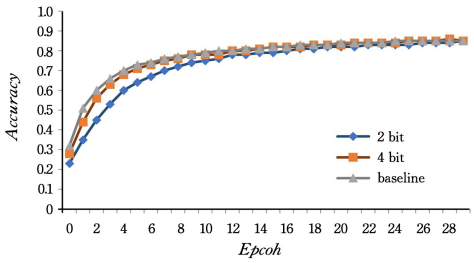

图 13 Cifar10 上 ResNet-110 压缩策略的准确率对比(4 节点)

Fig. 13 Accuracy comparison of gradient compression strategies (ResNet-110 on Cifar10) in 4-node

图 14 给出了 ImageNet 数据集＋ResNet-50 网络上的实验结果。从图中可以看出，在 ImageNet 数据集＋ResNet-50 网络中，4Bit，2Bit 和 baseline 3 种策略的准确率都很接近。

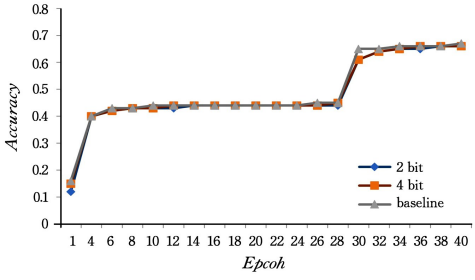

图 14 ImageNet 上 ResNet-50 压缩策略的准确率对比(4 节点)

Fig. 14 Accuracy comparison of gradient compression strategies (ResNet-50 onImageNet) in 4-node

从本节几个测试案例的结果可以看出，在整个训练过程中，4Bit 策略的准确率与 baseline 比较接近，明显好于 2Bit 策略。这是使用了多组阈值的 4Bit 策略较 2Bit 策略的优势之处，也体现了文中梯度特性分析的正确性。

5.3 消耗时间的测试与对比

表 2 列出了 3 种策略在 Cifar10＋Inception 消耗时间的测试结果。可以看出，4Bit 策略比 2Bit 策略消耗的时间略长，但差别不大。表 3 列出了 Cifar100＋Inception 网络的测试结果，情况与表 2 类似。

表 2 3 种策略在 Cifar10＋Inception 的耗时对比

Table 2 Time-consuming of three strategies in Cifar10＋Inception

压缩策略 \ 节点数	4 个节点		2 个节点	
	平均耗时/s	加速比	平均耗时/s	加速比
baseline	114		118.3	
2Bit	83	1.373	90.4	1.309
4Bit	89	1.281	97.1	1.218

表 3 3 种策略在 Cifar100＋Inception 的耗时对比

Table 3 Time-consuming of three strategies in Cifar100＋Inception

压缩策略 \ 节点数	4 个节点		2 个节点	
	平均耗时/s	加速比	平均耗时/s	加速比
baseline	109.3		113.7	
2Bit	86.4	1.265	93.1	1.221
4Bit	88.6	1.233	95.3	1.193

从实验测试结果可以看出，使用多组阈值的 4Bit 策略的准确率比只使用一组阈值策略的准确率要高，说明文中基于层级的梯度特性分析是正确的，采取的策略合理。4Bit 策略相对于 baseline 的加速比虽然略逊于 2Bit 策略，但 2Bit 在某些网络上的精度不理想，如在 Cifar100 上训练的 DFN 最后收敛的精度比 baseline 约低一个百分点，且 2Bit 在所有网络上的收敛速度慢，而 4Bit 在这些网络上都可以达到接近 baseline 的精度且收敛速度接近于 baseline。4Bit 策略在训练时的准确率明显高于 2Bit 策略，特别是在训练的前中期(这说明 4Bit 策略具有更好的收敛性)，因此 4Bit 方法是一个不错的选择。4Bit 在精度和加速比上做了平衡，确保在精度不受影响的情况下有较大的加速比。

进一步地，我们通过控制网络有效带宽实验也验证了：如果网络带宽变窄，baseline 方法性能会明显恶化，而 4Bit 方法的效果明显更好；如果进一步降低网络带宽，2Bit 方法会好于 4Bit 方法的效果。当然，这个时候模型的精度会下降。我们需要根据实际情况对两者进行取舍。

结束语 本文针对深度学习分布式训练中通信开销的瓶颈问题，提出了一种新的 4Bit 梯度压缩方法，将需要进行传输的 32 位梯度压缩成 4 位数据，有效减少了通信开销，缩短了训练过程消耗的时间。同时，基于层级梯度特性分析，提出了一种根据网络模型的特点分层设置不同阈值组的方法，提高了 4Bit 梯度压缩方法的准确率。相比于 MXNet 框架中的 2Bit 梯度压缩方法，本文提出的 4Bit 梯度压缩方法在准确率和消耗时间的平衡上效果更好，能在更短的时间内达到较高的准确率。

参 考 文 献

[1] LECUN Y, BENGIO Y, HINTON G. Deep Learning [J]. Nature, 2015, 521(7553): 436-444.

[2] YIN B, WANG W, WANG L. Review of Deep Learning [J]. Journal of Beijing University of Technology, 2015, 41(1): 48-59.

[3] HINTON G, DENG L, YU D, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups[J]. IEEE Signal Processing Magazine,

2012,29(6):82-97.

[4] GRAVES A, MOHAMED A, HINTON G. Speech Recognition with Deep Recurrent Neural Networks[C]// International Conference on Acoustics, Speech and Signal Processing. USA: IEEE, 2013: 6645-6649.

[5] DAI Y L, HE L, HUANG Z C. Unsupervised image hashing algorithm based on sparse-autoencoder[J]. Computer Engineering, 2019, 45(5): 222-225, 236.

[6] FARABET C, COUPRIE C, NAJMAN L, et al. Learning Hierarchical Features for Scene Labeling[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(8): 1915-1929.

[7] SUTSKEVER I, VINYALS O, LE Q. Sequence to Sequence Learning with Neural Networks[C]// Advances in Neural Information Processing Systems 27. USA: MIT press, 2014: 3104-3112.

[8] COLLOBERT R, WESTON J, BOTTOU L, et al. Natural Language Processing (Almost) from Scratch[J]. Journal of Machine Learning Research, 2011, 12(8): 2493-2537.

[9] YU K, JIA L, CHEN Y, et al. Deep Learning: Yesterday, Today, and Tomorrow[J]. Journal of Computer Research and Development, 2013, 50(9): 1799-1804.

[10] CHE S, BOYER M, MENG J, et al. A Performance Study of General-purpose Applications on Graphics Processors Using CUDA[J]. Journal of Parallel and Distributed Computing, 2008, 68(10): 1370-1380.

[11] HUILGOL R. 2bit Gradient Compression [EB/OL]. https://github.com/apache/incubator-mxnet/pull/8662.

[12] DEAN J, CORRADO G, MONGA R, et al. Large Scale Distributed Deep Networks[C]// Advances in Neural Information Processing Systems 25. USA: Curran Associates Inc, 2012: 1223-1231.

[13] REN Y, WU X, LI Z, et al. iRDMA: Efficient Use of RDMA in Distributed Deep Learning Systems[C]// Proceedings of the 2017 IEEE 19th International Conference on High Performance Computing and Communications. USA: IEEE, 2017: 231-238.

[14] ZHANG H, ZHENG Z, XU S, et al. Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters[C]// Proceedings of the 2017 USENIX Annual Technical Conference. USA: USENIX Association, 2017: 181-193.

[15] WEN W, XU C, YAN F, et al. TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning[C]// Advances in Neural Information Processing Systems 30. USA: Curran Associates Inc, 2017: 1508-1518.

[16] IOFFE S, SZEGEDY C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. ArXiv: 1502.03167, 2015.

[17] KRIZHEVSKY A, HINTON G. Learning Multiple Layers of Features from Tiny Images[R]. Toronto: University of Toronto, 2009.

[18] ZHAO L, WANG J, LI X, et al. On the Connection of Deep Fusion to Ensembling[J]. ArXiv: 1611.07718, 2016.

[19] RUSSAKOVSKY O, DENG J, SU H, et al. ImageNet Large Scale Visual Recognition Challenge[J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

JIANG Wen-bin, born in 1975, Ph. D, professor, is a member of China Computer Federation. His main research interests include distributed computing and machine learning.