doi：10.13413/j.cnki.jdxblxb.2024270

面向数据并行深度学习的准确率感知稀疏梯度融合算法

李洪亮，张 蒙，王子琛，李 想
（吉林大学 计算机科学与技术学院，长春 130012）

摘要：针对数据并行的深度学习作业中梯度同步导致的性能瓶颈问题，提出一种动态的稀疏梯度融合算法. 该算法将梯度压缩、流水线技术与张量融合技术进行协同建模，建立稀疏梯度融合行为对准确率影响的理论模型，并基于此寻找加快梯度同步的同时提高验证准确率的梯度融合方案，以解决稀疏梯度融合导致验证准确率不稳定的问题. 实验结果表明，该稀疏梯度融合算法比分层稀疏化方法缩短了 1.63 倍的通信时间，比已有的稀疏梯度融合算法缩短了 2.68 倍的收敛时间.

关键词：并行深度学习；梯度稀疏化；张量融合；通信流水线技术
中图分类号：TP391 **文献标志码**：A **文章编号**：1671-5489(2025)05-1356-10

Accuracy-Aware Sparse Gradient Fusion Algorithm for Data-Parallel Deep Learning

LI Hongliang, ZHANG Meng, WANG Zichen, LI Xiang
(*College of Computer Science and Technology, Jilin University, Changchun* 130012, *China*)

Abstract: Aiming at the problem of the performance bottleneck caused by gradient synchronization in data-parallel deep learning tasks, we proposed a dynamic sparse gradient fusion algorithm. The algorithm synergistically modelled gradient compression, pipeine techniques, and tensor fusion technology to establish a theoretical model of the impact of sparse gradient fusion behavior on accuracy. Based on this, the gradient fusion scheme was found to accelerate gradient synchronization while improving validation accuracy, so as to solve the problem of unstable validation accuracy caused by sparse gradient fusion. Experimental results show that the sparse gradient fusion algorithm reduces communication time by 1.63 times compared to layer-wise sparsification method, and reduces convergence time by 2.68 times compared to existing sparse gradient fusion algorithms.

Keywords: parallel deep learning; gradient sparsification; tensor fusion; communication pipeline technology

随着模型规模的不断增加，分布式机器学习已成为深度学习训练的主流范式. 越来越多的并行训练被提出，如数据并行[1]、模型并行[2]、流水线并行[3]及其组合[4]. 其中，数据并行广泛应用于多个设备上，加速大数据集深度学习作业的训练. 为获得高收敛性能，通常选择使用同步优化器，如随机

收稿日期：2024-07-02.

第一作者简介：李洪亮(1983—)，男，汉族，博士，副教授，从事分布式系统与虚拟化的研究，E-mail：lihongliang@jlu.edu.cn.
通信作者简介：李 想(1983—)，女，汉族，硕士，工程师，从事计算机网络和分布式系统的研究，E-mail：lxiang@jlu.edu.cn.

基金项目：吉林省自然科学基金面上项目(批准号：20230101062JC).

梯度下降优化器，但这需要机器之间频繁进行全局梯度同步，从而导致通信开销成为训练性能的瓶颈. 此外，硬件加速器计算能力的增长速度远超通信性能的增长速度，导致作业训练过程中单位时间内通信和计算的比例增加，进一步加剧了上述问题.

目前，缓解分布式深度学习通信瓶颈的主流方法是梯度压缩和重叠通信与计算技术(流水线技术). 梯度压缩技术[5-7]传输部分梯度，在减少数据传输量的同时保持良好的模型准确率. 重叠通信与计算技术[8]利用深度神经网络分层的特点，在反向传播过程中，流水线化梯度的反向计算任务和通信任务以隐藏部分通信开销[9]. 通常将分层流水线技术和梯度压缩技术组合使用[10]以实现快速的梯度同步更新，同时保证验证准确率. 然而，分层流水线技术的层级通信通常会带来不可忽略的启动开销[11]. 为此，可使用张量融合技术将相邻层融合以进一步减少通信启动的开销. 但这会产生一个新问题，即融合操作会改变梯度数据集，进而影响梯度稀疏化对相关张量的效果，导致验证准确率不稳定.

本文针对上述稀疏梯度融合导致的准确率不稳定问题，提出一个稀疏梯度融合的准确率相关模型.该模型定义了分层稀疏误差量化每层对验证准确率的影响并比较融合前后的稀疏误差；在此基础上，建立融合前后稀疏误差的比值关系，进而推导出能提高验证准确率的融合条件. 结合通信减少的融合条件，本文实现了一种面向数据并行的准确率感知稀疏梯度融合(sparse-error-aware merged gradient sparsification stochastic gradient descent，SEAMGS-SGD)算法，旨在寻找能加快梯度同步并提高验证准确率的融合方案，并在 GPU 集群上测试. 实验结果表明，本文的 SEAMGS-SGD 算法能达到与原始 SGD(stochastic gradient descent)相似的验证准确率，与仅考虑通信开销的融合算法相比，提高了验证准确率并缩短了收敛时间.

1　研究背景

1.1　分布式深度学习的通信瓶颈

在数据并行训练场景中，每个节点上的设备(GPU)都使用分配的数据子集独立训练备份模型[1]. 设备在一次迭代计算完成后，进行梯度更新，其分为同步梯度更新和异步梯度更新. 同步梯度更新是在每轮迭代计算结束后，全部设备进行全局梯度同步，随即开启下一轮迭代，这种方式模型收敛效果好，但同步开销大[12]. 异步更新则是在梯度同步过程中，设备之间不相互等待，这种方式同步开销小，但可能导致模型收敛速度变慢甚至难以收敛的问题[13]. 在同步梯度更新过程中，数据通信量随节点数量递增，这并不利于并行深度学习作业的可扩展性，使通信开销成为训练的瓶颈，甚至超过计算开销[14]. 通常通过以下两种方式解决.

1) 减少数据传输量：使用梯度压缩技术，通常分为梯度稀疏化[5]和梯度量化. 梯度稀疏化是在梯度的所有维度中选择一部分维度进行传输，而将其他维度置零，梯度量化则是以低精度数据传输梯度[6-7].

2) 优化通信行为：利用深度神经网络的分层结构，重叠深度学习训练的反向计算任务和通信任务，尽可能地将通信任务的开销隐藏在计算任务开销中，即无等待的反向传播 WFBP(wait-free backropagation)[8]，如图 1(A)所示. 由于每层启动通信产生额外的启动开销，原始的 WFBP 效率低下，MG-WFBP(merging gradients WFBP)[9]提出利用张量融合技术适当融合相邻的层之后传输，如图 1(B)所示，进一步减少了通信启动带来的额外开销.

结合优化通信行为和减少数据传输量这两种方式，通过分层梯度压缩控制每层数据传输量以便更好地将通信任务开销隐藏在计算任务开销后，如图 1(C)所示. 文献[15]简单地将分层梯度稀疏和 WFBP 结合，发现不可忽略的通信启动开销导致训练效率低下. 在此基础上，文献[16]通过梯度融合进一步减少通信启动开销，如图 1(D)所示. 这种结合结果证明能很大程度地减少通信开销. 但对于稠密梯度，张量融合不会导致验证准确率的损失，但对于稀疏梯度情况则不同，针对稀疏梯度的融合可能会导致训练精度不稳定.

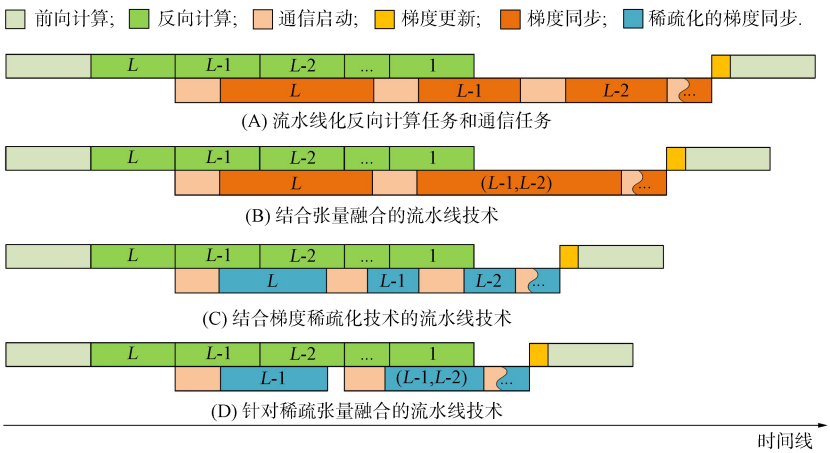

图1 反向计算任务和通信任务的流水线执行流程

Fig. 1 Pipeline execution process of reverse computation tasks and communication tasks

1.2 相关工作

1.2.1 梯度压缩

梯度压缩作为一种有损的减少通信的方法主要分为两种技术：一种是梯度量化，将梯度值压缩成低精度值后传输；另一种是梯度稀疏化，在每次迭代结束时选择一小部分梯度传输而将不需要传输的梯度清零，常用的选择算法有随机选择的 Rand-K 算法[17]和选择前 K 个最大值的 Top-K 算法[18-19]. 一些研究关注如何在不显著影响训练过程收敛性的情况下实现更高的压缩比. 例如，具有误差补偿的 Top-K 稀疏化可将 99%的局部梯度归零，而不会严重损失精度[7]. 梯度量化至少使用1位进行数据传输，所以压缩率存在上限，而梯度稀疏能更大程度地减少通信开销. 本文主要关注梯度稀疏化. 一些研究使用动态压缩技术提高验证准确率[20]，MIPD[20]基于模型可解释性量化每层重要性，分层设置压缩率，AC-SGD(adaptively-compressed stochastic gradient descent)[17]建立给定通信开销下最小化深度学习模型收敛误差问题模型，动态调整整个训练过程的压缩率.

1.2.2 重叠计算和通信的流水线技术

减少通信开销的另一种有效方法是利用深度神经网络分层结构在反向传播阶段将当前层的通信任务(梯度同步)和上一层计算任务(梯度计算)流水线化，以隐藏通信开销，从而提高系统的吞吐量. 这种方法也称为无等待反向传播(WFBP)，已作为现代深度学习框架(如 TensorFlow，Pytorch-DDP 和 Horovod)中的默认机制实现[21]. 由于许多深度神经网络模型有大量层，在分布式训练中每层仅传输少量的数据，由于传输每层数据都需要启动时间(传输延迟)，同步少量数据无法充分利用当前网络拓扑中的网络带宽，导致 WFBP 效率低下. MG-WFBP[9]基于将一些短通信任务合并为单个任务可减少总体通信时间这一事实，提出张量融合技术，通过融合相邻层减少通信启动次数，并制定了一个优化问题最小化管道通信和计算的训练时间. DeAR[21]考虑到 WFBP 仅在反向传播阶段利用梯度计算的时间进行通信，未考虑到前向传播阶段，而前向传播阶段占每次迭代中总计算时间的 1/3，在此基础上，DeAR 将全规约(allreduce)原语解耦成两个连续的操作，这两个操作分别与反向传播和前向传播重叠，无需额外的通信.

结合重叠计算和通信的流水线技术与梯度压缩技术，可以进一步减少通信开销，LAGS-SGD (layer-wise adaptive gradient sparsification SGD)[15]提出通过计算通信开销与计算开销的比率尽可能地将通信开销隐藏在计算后，并证明了分层稀疏策略的收敛性. 由于不可忽略的通信启动延迟导致其有限的性能提升，OMGS-SGD (communication-efficient distributed deep learning with merged gradient sparsification SGD)[16]在此基础上进一步引入张量融合减少通信启动开销，并提出稀疏张量融合最优化问题以最小化迭代时间. 目前工作主要关注如何通过稀疏张量融合减少通信开销而忽略了稀疏张量融合对验证准确率的影响.

1.3　稀疏张量融合策略对训练性能的影响

由于梯度压缩会导致验证准确率的损失，因此梯度压缩行为在不同压缩率和梯度数据集情况下对验证准确率的影响不同. 张量融合操作会改变梯度数据集，从而改变梯度稀疏化对相关张量的影响. 理论和经验均已证明，分层压缩产生的稀疏误差与全局产生的稀疏误差不同，并且分层压缩这种更细粒度的压缩方式更有利于验证准确率的提高[22]. 在本文建立的模型中，假设两层融合后的数据传输量等于原有分层稀疏后得到的两层的数据传输量之和，即融合前后的数据传输量不变，只是融合两层后减少了一次通信启动开销. 尽管仍挑选相同数量的梯度，但融合后的梯度样本集合(样本和数量)发生了变化，这对整体训练会产生不同的影响. 针对该问题，本文首先量化了梯度稀疏化对验证准确率的影响，在此基础上进一步量化了融合前后验证准确率的变化，推导出融合后验证准确率增加的条件. 在决策张量融合过程中同时考虑减少通信开销和提高验证准确率.

2　问题模型及算法设计

2.1　稀疏梯度融合问题描述

通过张量融合减少通信开销是本文的主要目标. 在通信效率方面，由于通信启动延迟，分别传输两条大小为 d_1 和 d_2 的消息比传输一条大小为 d_1+d_2 的消息通信开销更大. 由于深度神经网络的分层结构，第 l 层的通信需在计算完该层梯度后才能执行，融合某些层会导致其他层的通信延迟，因此本文通过时间线模型进行数学推导，得到通信开销减少的条件. 在验证准确率方面，由于稀疏梯度融合会导致验证准确率不稳定，为提高稀疏梯度融合后的验证准确率，本文首先定义稀疏梯度融合的准确率模型，量化融合前后稀疏误差变化，并推导出提高验证准确率的融合条件. 通过遍历所有层，判断当前层是否与下一层融合时判断通信开销减少条件和验证准确率提高条件，同时满足则融合，否则不融合. 寻找在满足验证准确率提高的条件下，最小化迭代时间的稀疏梯度融合方案. 本文用 α 表示单个梯度同步操作的启动时延，β 表示每个字节的传输时间，$d^{(l)}$ 表示第 l 层梯度大小，$k^{(l)}$ 表示第 l 层稀疏后梯度大小，$\rho^{(l)}$ 表示第 l 层的稀疏率，$\boldsymbol{g}^{(l)}$ 表示第 l 层的梯度矩阵，L 表示模型的可学习层数(或张量个数)，$t_b^{(l)}$ 表示每次迭代中第 l 层梯度计算时间，$\tau_b^{(l)}$ 表示每次迭代中第 l 层梯度计算开始时间，$t_c^{(l)}$ 表示每次迭代中第 l 层梯度同步时间，$\tau_c^{(l)}$ 表示每次迭代中第 l 层梯度同步开始时间.

2.2　稀疏梯度融合的准确率模型

对于任意一个向量 $\boldsymbol{x}\in\mathbb{R}^d$，$0<k<d$，本文使用 Rand K 稀疏算子(在梯度向量的所有 d 个元素中随机挑选 k 个值)进行稀疏化. 由文献[15]可知，使用分层稀疏化方法更新参数，其中使用稀疏梯度更新的参数 $\boldsymbol{v}_{t+1}$ 与使用未稀疏更新的参数 $\boldsymbol{x}_{t+1}$ 间的差值可表示为

$$\| \boldsymbol{v}_{t+1}-\boldsymbol{x}_{t+1} \|^2=\sum_{l=1}^{L}E\left[\left\|\left(\frac{1}{p}\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}-\frac{1}{p}\sum_{p=1}^{P}\text{Rand }K(\boldsymbol{g}_t^{p,(l)},k^{(l)})\right)\right\|^2\right]=$$

$$\sum_{l=1}^{L}\left(1-\frac{k^{(l)}}{d^{(l)}}\right)\left\|\frac{1}{p}\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2. \tag{1}$$

定义 1　第 l 层梯度稀疏化对参数更新结果产生的误差，用第 l 层梯度二范数表示：

$$\delta^{(l)}=\left(1-\frac{k^{(l)}}{d^{(l)}}\right)\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2, \tag{2}$$

其中 $\delta^{(l)}$ 为第 l 层的稀疏误差，$\dfrac{k^{(l)}}{d^{(l)}}$ 为第 l 层的稀疏率 $\rho^{(l)}$.

定义 2　当第 l 层通信开始的时间点不传输这一层的梯度，而是融合第 l 层和第($l-1$)层的数据一起传输，则称($l-1$)层为融合层.

当第 l 层与第($l-1$)层融合，则第 l 层与其相邻层第($l-1$)层有如下性质.

性质 1　第 l 层不进行数据传输：

$$t_c^{(l)}=0. \tag{3}$$

性质 2　第($l-1$)层梯度元素的数量增加 $d^{(l)}$ 个：

$$d^{(l-1)}=d^{(l-1)}+d^{(l)}. \tag{4}$$

性质3 第$(l-1)$层梯度计算在第l层梯度计算结束时马上开始:

$$\tau_b^{(l-1)}=\tau_b^{(l)}+t_b^{(l)}. \tag{5}$$

性质4 假设融合前后的数据传输量不变,第$(l-1)$层稀疏后梯度元素的个数增加$k^{(l)}$个:

$$k^{(l-1)}=k^{(l-1)}+k^{(l)}. \tag{6}$$

则融合层$(l-1)$层的稀疏率为

$$\rho^{(l-1)}=\frac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}. \tag{7}$$

性质5 张量融合操作是拼接两个相邻层的张量,因此第$(l-1)$层张量二范数的平方值可表示为

$$\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\Big\|^2=\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\Big\|^2+\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\Big\|^2. \tag{8}$$

通过上述定义,可得两个相邻层单独稀疏产生的稀疏误差δ和两个相邻层合并产生的稀疏误差$\tilde{\delta}$分别为

$$\delta=\Big(1-\frac{k^{(l)}}{d^{(l)}}\Big)\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\Big\|^2+\Big(1-\frac{k^{(l-1)}}{d^{(l-1)}}\Big)\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\Big\|^2, \tag{9}$$

$$\tilde{\delta}=\Big(1-\frac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}\Big)\Big(\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\Big\|^2+\Big\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\Big\|^2\Big). \tag{10}$$

判断融合第l层和第$(l-1)$层稀疏误差是否减少,如果融合后的稀疏误差值减少了,说明融合后对验证准确率的影响变小,则融合第l层和第$(l-1)$层;反之,则不融合.

2.3 准确率感知的稀疏梯度融合策略

为寻找平衡通信效率和验证准确率的稀疏梯度融合方案,本文综合考虑两方面决定是否融合.一方面,为加速梯度同步,合理融合相邻层减少通信开销.本文建立了时间线模型表示张量计算和通信的时间关系,如果两层融合能减少通信开销则融合,反之则不融合.另一方面,在此基础上,根据稀疏梯度融合的准确率模型,量化针对稀疏梯度的张量融合引入的稀疏误差,比较融合前后产生的稀疏误差,融合后稀疏误差减少则融合,反之则不融合.

2.3.1 减少通信开销

对于任意$l(1<l\leqslant L)$层,第$(l-1)$层通信结束的时间表示为$\mu_c^{(l-1)}$,则可得

$$\begin{aligned}\mu_c^{(l-1)}&=\tau_c^{(l-1)}+t_c^{(l-1)}=\max\{\tau_b^{(l-1)}+t_b^{(l-1)},\tau_c^{(l)}+t_c^{(l)}\}+t_c^{(l-1)}=\\&\max\{\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)},\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)}\}+\alpha+\beta\rho^{(l-1)}d^{(l-1)}.\end{aligned} \tag{11}$$

检验融合第l层与第$(l-1)$层后$\mu_c^{(l-1)}$是否减少,如果减少则融合,反之则不融合.假设第l层与第$(l-1)$层融合,第$(l-1)$层通信结束的时间表示为$\tilde{\mu}_c^{(l-1)}$,则可得

$$\tilde{\mu}_c^{(l-1)}=\max\{\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)},\tau_c^{(l)}\}+t_c^{(l-1)}+\alpha+\beta(\rho^{(l-1)}d^{(l-1)}+\rho^{(l)}d^{(l)}). \tag{12}$$

令t_{reduce}表示融合后减少的通信时间,则

$$\begin{aligned}t_{\text{reduce}}&=\mu_c^{(l-1)}-\tilde{\mu}_c^{(l-1)}=\max\{\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)},\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)}\}-\\&\max\{\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)},\tau_c^{(l)}\}-\beta\rho^{(l)}d^{(l)}.\end{aligned} \tag{13}$$

为去除两个最大值运算符,分类讨论4种情形.易得$\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)}>\tau_c^{(l)}$,下面详细讨论另外3种情形.

情形1) 第$(l-1)$层的反向计算结束时间大于第l层通信结束时间,且大于第l层通信开始时间:

$$\begin{aligned}&\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}>\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)},\\&\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}>\tau_c^{(l)},\quad t_{\text{reduce}}=-\beta\rho^{(l)}d^{(l)}<0.\end{aligned} \tag{14}$$

t_{reduce}的值小于0,因此在这种情形下融合第l层与第$(l-1)$层不能减少通信开销.

情形2) 第$(l-1)$层的反向计算结束时间小于第l层通信开始时间:

$$\begin{aligned}&\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}<\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)},\quad \tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}<\tau_c^{(l)},\\&t_{\text{reduce}}=\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)}-\tau_c^{(l)}-\beta\rho^{(l)}d^{(l)}<\alpha.\end{aligned} \tag{15}$$

t_{reduce}的值大于 0，因此在这种情形下融合第 l 层与第$(l-1)$层可减少通信开销.

情形 3) 第$(l-1)$层的反向计算结束时间大于第 l 层通信开始时间，且小于第 l 层通信结束时间：

$$\begin{aligned}&\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}<\tau_c^{(l)}+\alpha+\beta\rho^{(l)}d^{(l)},\quad \tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}>\tau_c^{(l)},\\&t_{\text{reduce}}=\tau_c^{(l)}+\alpha-(\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}),\end{aligned}\tag{16}$$

在这种情形下，当且仅当

$$\tau_c^{(l)}+\alpha>\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}\tag{17}$$

时，融合第 l 层与第$(l-1)$层可减少通信开销.

综合以上 3 种情形可知，当满足式(17)时，记为通信开销融合条件 Q1，融合第 l 层与第$(l-1)$层可以减少通信开销.

2.3.2　提高验证准确率

设 γ 表示第 l 层与第$(l-1)$层融合后和融合前引入稀疏误差的比值，则根据式(9)和式(10)可得

$$\gamma=\frac{\tilde{\delta}}{\delta}=\frac{\left(1-\dfrac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}\right)\left(\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2+\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2\right)}{\left(1-\dfrac{k^{(l)}}{d^{(l)}}\right)\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2+\left(1-\dfrac{k^{(l-1)}}{d^{(l-1)}}\right)\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2}\leqslant 1.\tag{18}$$

经推导可知，式(18)等价于

$$\begin{cases}\dfrac{\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2}{d^{(l)}}\geqslant\dfrac{\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2}{d^{(l-1)}},\quad \rho^{(l)}<\rho^{(l-1)},\\[2ex]\dfrac{\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2}{d^{(l)}}\geqslant\dfrac{\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2}{d^{(l-1)}},\quad \rho^{(l)}>\rho^{(l-1)}.\end{cases}\tag{19}$$

证明：对式(18)中不等式进行推导，可得

$$\begin{aligned}&\frac{\left(1-\dfrac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}\right)\left(\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2+\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2\right)}{\left(1-\dfrac{k^{(l)}}{d^{(l)}}\right)\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2+\left(1-\dfrac{k^{(l-1)}}{d^{(l-1)}}\right)\left\|\sum\limits_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|^2}\leqslant 1\Leftrightarrow\\&\left(\frac{k^{(l-1)}}{d^{(l-1)}}-\frac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}\right)\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|\leqslant\left(\frac{k^{(l)}+k^{(l-1)}}{d^{(l)}+d^{(l-1)}}-\frac{k^{(l)}}{d^{(l)}}\right)\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2\Leftrightarrow\\&\frac{(\rho^{(l-1)}-\rho^{(l)})\,d^{(l)}}{d^{(l)}+d^{(l-1)}}\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l-1)}\right\|\leqslant\frac{(\rho^{(l-1)}-\rho^{(l)})\,d^{(l-1)}}{d^{(l)}+d^{(l-1)}}\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2,\end{aligned}\tag{20}$$

基于 $\rho^{(l-1)}-\rho^{(l)}$的正负性，分类讨论可得式(19). 证毕.

当满足式(19)时，$\gamma\leqslant 1$ 成立，第 l 层与第$(l-1)$层融合后比融合前引入了更少的稀疏误差，即融合后会有更高的验证准确率，因此执行融合操作. 为简化条件，本文进一步定义第 l 层的权重为

$$w^{(l)}=\frac{1}{d^{(l)}}\left\|\sum_{p=1}^{P}\boldsymbol{g}_t^{p,(l)}\right\|^2.\tag{21}$$

因此，由式(19)和式(21)可得下列等价条件，记为准确率融合判断条件 Q2：

$$(\rho^{(l-1)}-\rho^{(l)})(w^{(l-1)}-w^{(l)})<0.\tag{22}$$

式(2)表明，若第$(l-1)$层的权值小于等于第 l 层，则第$(l-1)$层保留得更多(稀疏率更高). 两层单独稀疏不能保证权重大的第 l 层多保留，权重小的层少保留. 而将两层融合可以适当地平衡. 只有满足条件 Q2，融合第 l 层和第$(l-1)$层才能减少稀疏误差，进而提高验证准确率.

基于上述推导可知，在满足通信开销融合条件 Q1 的基础上保证融合减少通信开销的同时判断验证准确率融合条件 Q2，寻找张量融合方案在保证减少通信开销的同时提高验证准确率，表示为

$$\begin{cases}\text{Q1}=(\tau_c^{(l)}+\alpha>\tau_b^{(l)}+t_b^{(l)}+t_b^{(l-1)}),\\\text{Q2}=((\rho^{(l-1)}-\rho^{(l)})(w^{(l-1)}-w^{(l)})<0),\end{cases}\tag{23}$$

$$Q1 \wedge Q2 \Rightarrow l = l_f, \tag{24}$$

$$l = \begin{cases} l_f, & 1 < l \leqslant L, Q1 \wedge Q2 \text{ 为真}, \\ l_n, & \text{其他}. \end{cases} \tag{25}$$

式(23)表示当且仅当某层同时满足条件Q1和Q2时，将其与下一层融合，使其成为融合层能带来训练性能的提升，本文将没有融合操作的层称为原始层.

2.4 算法设计

为寻找能加速梯度同步并保证准确率的稀疏张量融合方案，本文使用下列算法查找对给定深度神经网络(DNN)模型中所有符合条件的融合层. 为减少算法的计算开销，设置固定间隔重复计算融合方案.

算法1 准确率感知稀疏梯度融合算法.

输入：通信延迟时间 α，每个字节传输时间 β，每层稀疏率 $\rho^{(l)}$，模型层数 L，每层梯度元素个数 $d^{(l)}$，每层反向计算时间 t_b，每层梯度值 $g^{(l)}$；

输出：融合层的集合 f；

步骤1) 初始化融合层集合，所有层均为原始层 $f[1,2,\cdots,L]=[1_n,2_n,\cdots,L_n]$；

步骤2) for $t-1$ mod interval$=0$ do

步骤3) 分别计算每层通信开始时间，计算开始时间，权重；

步骤4) for $l=L$ to 2 do

步骤5) 计算第 l 层是否满足条件Q1和Q2；

步骤6) if 第 l 层满足条件Q1和Q2 then

步骤7) 第 l 层执行融合操作；

步骤8) 重新计算第 l 层通信开始时间；

步骤9) 将第 l 层加入到融合层集合 f；

步骤10) end if

步骤11) return.

算法1使用贪婪算法得到融合层集合，保证每次融合都同时满足条件Q1和Q2，并保证在融合减少通信开销的同时提高验证准确率.

3 实 验

3.1 实现原型

本文基于公开代码的用于低带宽网络上深度学习的通信高效分布式训练框架OMGS-SGD在PyTorch实现了SEAMGS-SGD的原型(https://github.com/Bluejasmine00/SEAMGS.git). 主要实现分层稀疏化并在张量融合过程中动态调整稀疏率，在此基础上，实现核心功能函数实时计算减少通信开销并提高验证准确率的稀疏梯度融合方案.

3.2 实验环境

本文选取4种神经网络VGG16,ResNet20,ResNet56和ResNet110训练数据集Cifar-10，训练批次大小分别是128,64,128和128. 在4节点集群上进行实验，节点之间通过以太网相互通信，数据传输速度为10 Gbps. 每个节点配备一个NVIDIA A30的GPU. 所有GPU机器运行64位的Ubuntu 20.04，使用CUDA toolkit 11.4版本和PyTorch 1.10.0. 通信库使用OpenMPI-4.1.5和NCCL-2.20.5. 使用高度优化的分布式训练库Horovod-0.28.1.

将准确率感知的稀疏梯度融合算法与下列算法进行比较：1) 原始的无等待反向传输算法(P-SGD)，不进行张量融合和梯度稀疏化；2) Rand K 稀疏化算法(Rand K-SGD)，使用随机选择稀疏化方法，所有层反向计算结束后进行全局稀疏化操作，不进行流水线化和张量融合；3) 分层梯度稀疏化算法，层级通信，使用流水线化技术但不进行张量融合，使用随机选择梯度稀疏化；4) 固定融合层大小的融合算法(merged gradient sparsification stochastic gradient descent，MGS-SGD)，固定每次传输的数据大小并进行随机选择梯度稀疏化，使用流水线化技术. 本文SEAMGS-SGD算法同时考虑验

证准确率条件和通信开销条件进行梯度融合，使用随机选择梯度稀疏化.

每层张量计算时间通过真实捕获收集没有梯度稀疏化的 DNN 训练作业 50 次迭代的每层张量反向计算时间后取平均；每层张量梯度信息通过每 100 次迭代真实捕获每层张量反向计算得到的梯度值后取平均；每层通信时间通过模型预测，收集每层张量大小，利用通信成本模型和网络带宽预测通信时间.

基于模型可解释性设置每层稀疏率，文献[20]的实验结果表明，每层参数重要性与每层参数的二范数大小正相关，二范数大的参数相关梯度需优先被更新. 所以根据每层参数的二范数设置每层的权重，进一步设置每层的稀疏率，为尽可能减少通信开销，本文将稀疏率设置在[0.001,002]内.

3.3　实验结果分析

3.3.1　通信效率

表 1 为不同算法使用 4 节点的集群训练 4 种神经网络中的迭代执行时间. 由表 1 可见，准确率感知的稀疏梯度融合算法能尽可能流水线化计算任务和通信任务，从而达到最短的迭代执行时间. 原始无等待的反向传输算法(P-SGD)执行时间最长，因为其未进行梯度稀疏化且使用层级通信引入了不可忽略的通信启动开销. 使得分层梯度稀疏化算法(LAGS-SGD)和固定融合层大小的融合算法(MGS-SGD)弱化了流水线化技术带来的优势，甚至可能比 Rand K 稀疏化算法(Rand K-SGD)更慢. 本文 SEAMGS-SGD 算法能基于对张量计算任务和通信任务的时间线模型和通信环境(节点数量、带宽大小等)动态调整稀疏张量融合方案，尽可能多地重叠计算任务和通信任务.

表 1　不同算法 1 000 次迭代的平均迭代执行时间比较

Table 1　Comparison of average iteration execution time for 1 000 iterations of different algorithms　s

模型	算法				
	P-SGD	Rand K-SGD	LAGS-SGD	MGS-SGD	SEAMGS-SGD
VGG16	0.525	0.301	0.402	0.375	0.276
ResNet20	0.084	0.070	0.075	0.069	0.060
ResNet56	0.199	0.158	0.179	0.173	0.131
ResNet110	0.344	0.264	0.299	0.268	0.210

在 VGG16 模型上，SEAMGS-SGD 算法相比于其他 4 种算法其缩短迭代时间最高达 1.9 倍；在 ResNet20 模型上，SEAMGS-SGD 算法相比于其他 4 种算法缩短迭代时间 1.15～1.4 倍；在 ResNet56 模型上，SEAMGS-SGD 算法相比于其他 4 种算法缩短迭代时间 1.20～1.51 倍；在 ResNet110 模型上，SEAMGS-SGD 算法相比于其他 4 种算法缩短迭代时间 1.25～1.63 倍. 由于 VGG16 的参数量最大(14 728 266 个)，反向计算时间相对通信时间很小，通信时间只能有一小部分与反向计算时间重叠，所以相对于 Rand K-SGD，LAGS-SGD 和 MGS-SGD 算法提升并不明显，但相对于 P-SGD 算法的性能提升很明显，说明动态分层稀疏化减少了大量的通信开销，占主导作用. 而在层数更多的卷积神经网络上，反向计算时间和通信时间能更好地重叠.

3.3.2　收敛性能

图 2 为 4 种算法在不同网络上验证准确率的对比结果. 由图 2 可见，SEAMGS-SGD 算法的验证准确率与不进行梯度稀疏化的 P-SGD 最接近，这是因为融合时考虑了梯度稀疏化对验证准确率的影响，使梯度稀疏化操作更精确，优于 LAGS-SGD 和 MGS-SGD 算法. SEAMGS-SGD 算法是基于 LAGS-SGD 算法建立的模型，旨在引入的稀疏噪声值大于等于 LAGS-SGD 算法，所以验证准确率较高，进一步证明了稀疏梯度融合准确率模型的有效性. MGS-SGD 算法虽然在一定程度上减少了通信开销，但由于其未考虑与验证准确率相关的梯度、稀疏率等信息动态调整张量融合方案，导致其在训练中的准确率性能不稳定，在层数较少的 VGG16，ResNet20，ResNet56 模型上高于或相当于 LAGS-SGD 算法的验证准确率，在层数较多的 ResNet110 模型上低于 LAGS-SGD 算法的验证准确率，说明更深的模型需要更细粒度的张量融合方案. 本文算法在 VGG16 和 ResNet110 模型上性能提升较明显，是因为这两个网络的参数量很大，即对梯度稀疏化更敏感，稀疏率和稀疏样本的变化都会更易导致训练性能的变化. 模型越大，SEAMGS-SGD 算法在验证准确率方面的性能提升越明显.

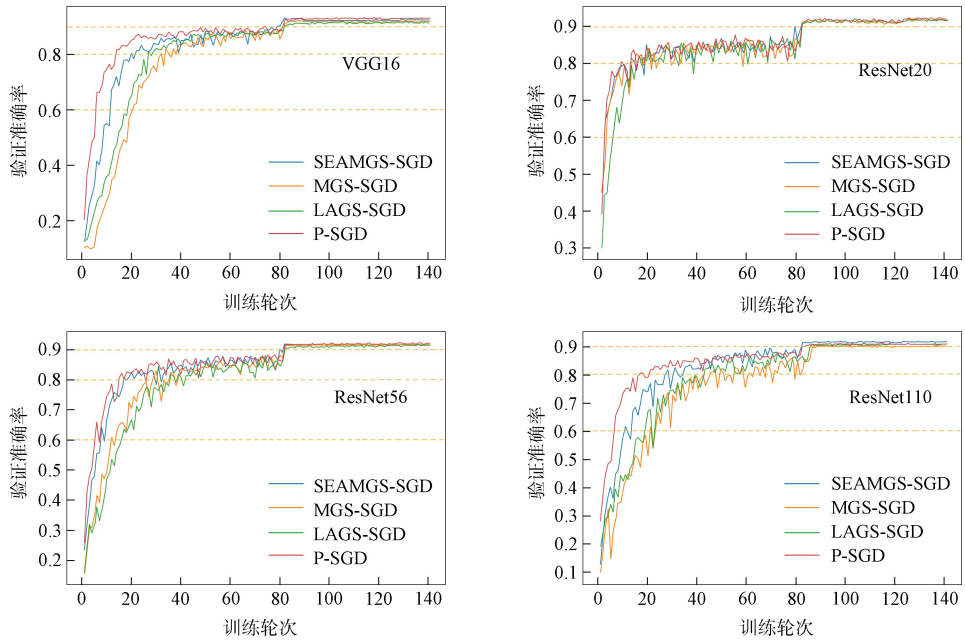

图2 不同算法在不同网络上验证准确率的对比结果

Fig. 2 Comparison results of test accuracy of different algorithms on different networks

因为SEAMGS-SGD算法平均迭代时间最短，准确率上升速度与P-SGD算法相当，所以能在最短时间内达到收敛状态. 在训练前期，本文算法验证准确率升高较快，达到60%验证准确率的实际时间相较其他3种算法缩短了1.42～2.68倍，达到80%验证准确率的实际时间相较其他3种算法缩短了2.35～14倍，并能在最短时间达到收敛状态，收敛后的准确率与P-SGD算法相当.

综上所述，针对分布式深度学习中数据并行训练的通信瓶颈问题，本文提出了一种准确率感知的稀疏梯度融合算法. 传统针对稀疏梯度的张量融合技术仅考虑提升通信效率，而未考虑稀疏梯度融合过程中对验证准确率的影响变化，导致验证准确率不稳定. 为在减少通信开销的同时保持高准确率验证，本文建立了稀疏梯度融合的准确率模型，量化了融合前后梯度稀疏化对验证准确率的影响. 基于此模型，提出了一种动态稀疏梯度融合算法，寻找既能加快梯度同步又能提高验证准确率的融合方案. 实验结果表明：与其他稀疏化方法相比本文方法达到最短通信时间同时提高了验证准确率，缩短了最高达2.68倍的收敛时间；该算法在4个不同模型上的验证准确率优化效果并不稳定，在ResNet20模型上提升效果较弱，而在参数越多、层数越深的模型(VGG16,ResNet56和ResNet10)上，本文算法表现出更好的优化性能.

参 考 文 献

[1] LI S W, LU K, LAI Z Q, et al. A Multidimensional Communication Scheduling Method for Hybrid Parallel DNN Training [J]. IEEE Transactions on Parallel and Distributed Systems, 2024, 35(8): 1415-1428.

[2] KIM T, KIM H, YU G I, et al. Bpipe: Memory-Balanced Pipeline Parallelism for Training Large Language Models [C]//Proceedings of the 40th International Conference on Machine Learning. New York: ACM, 2023: 16639-16653.

[3] RAFAILOV R, SHARMA A, MITCHELL E, et al. Direct Preference Optimization: Your Language Model Is Secretly a Reward Model [J]. Advances in Neural Information Processing Systems, 2024, 36: 15-27.

[4] ZHAO S X, LI F X, CHEN X X, et al. VPipe: A Virtualized Acceleration System for Achieving Efficient and Scalable Pipeline Parallel DNN Training [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(3): 489-506.

[5] WANG Z Q, DUAN Q Y, XU Y D, et al. An Efficient Bandwidth-Adaptive Gradient Compression Algorithm for

Distributed Training of Deep Neural Networks [J]. Journal of Systems Architecture, 2024, 150: 103-116.

[6] SEIDE F, FU H, DROPPO J, et al. 1-Bit Stochastic Gradient Descent and Its Application to Data-Parallel Distributed Training of Speech DNNs [C]//Fifteenth Annual Conference of the International Speech Communication Association. New York: ACM, 2014: 1058-1062.

[7] LI S G, HOEFLER T. Near-Optimal Sparse Allreduce for Distributed Deep Learning [C]//Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. New York: ACM, 2022: 135-149.

[8] ZHANG H, ZHENG Z Y, XU S Z, et al. Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters [C]//2017 USENIX Annual Technical Conference. [S.l.]: USENIX, 2017: 181-193.

[9] SHI S H, CHU X W, LI B. MG-WFBP: Merging Gradients Wisely for Efficient Communication in Distributed Deep Learning [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(8): 1903-1917.

[10] WANG Z, LIN H B, ZHU Y B, et al. Hi-Speed DNN Training with Espresso: Unleashing the Full Potential of Gradient Compression with Near-Optimal Usage Strategies [C]//Proceedings of the Eighteenth European Conference on Computer Systems. New York: ACM, 2023: 867-882.

[11] ZHANG L, ZHANG L T, SHI S H, et al. Evaluation and Optimization of Gradient Compression for Distributed Deep Learning [C]//2023 IEEE 43rd International Conference on Distributed Computing Systems. Piscataway, NJ: IEEE, 2023: 361-371.

[12] UM T, OH B, KANG M, et al. Metis: Fast Automatic Distributed Training on Heterogeneous GPUs [C]//2024 USENIX Annual Technical Conference. [S.l.]: USENIX, 2024: 563-578.

[13] BOTTOU L. Large-Scale Machine Learning with Stochastic Gradient Descent [C]//19th International Conference on Computational Statistics. [S.l.]: Physica-Ver Lag HD, 2010: 177-186.

[14] WANG Z, LIN H B, ZHU Y B, et al. Hi-speed DNN Training with Espresso: Unleashing the Full Potential of Gradient Compression with Near-Optimal Usage Strategies [C]//Proceedings of the Eighteenth European Conference on Computer Systems. New York: ACM, 2023: 867-882.

[15] SHI S H, TANG Z H, WANG Q, et al. Layer-Wise Adaptive Gradient Sparsification for Distributed Deep Learning with Convergence Guarantees [C]//Proceedings of the 24th European Conference on Artificial Intelligence. [S.l.]: IOS Press, 2020: 1-8.

[16] SHI S H, WANG Q, CHU X W, et al. Communication-Efficient Distributed Deep Learning with Merged Gradient Sparsification on GPUs [C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. Piscataway, NJ: IEEE, 2020: 406-415.

[17] YAN G, LI T, HUANG S L, et al. AC-SGD: Adaptively Compressed SGD for Communication-Efficient Distributed Learning [J]. IEEE Journal on Selected Areas in Communications, 2022, 40(9): 2678-2693.

[18] CHEN C, LI M, YANG C. bbTopk: Bandwidth-Aware Sparse Allreduce with Blocked Sparsification for Efficient Distributed Training [C]//2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS). Piscataway, NJ: IEEE, 2023: 73-83.

[19] SHI S H, WANG Q, ZHAO K Y, et al. A Distributed Synchronous SGD Algorithm with Global Top-k Sparsification for Low Bandwidth Networks [C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Piscataway, NJ: IEEE, 2019: 2238-2247.

[20] ZHANG Z R, WANG C L. MIPD: An Adaptive Gradient Sparsification Framework for Distributed DNNs Training [J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(11): 3053-3066.

[21] ZHANG L, SHI S H, CHU X W, et al. DeAR: Accelerating Distributed Deep Learning with Fine-Grained All-Reduce Pipelining [C]//2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS). Piscataway, NJ: IEEE, 2023: 142-153.

[22] DUTTA A, BERGOU E H, ABDELMONIEM A M, et al. On the Discrepancy between the Theoretical Analysis and Practical Implementations of Compressed Communication for Distributed Deep Learning [C]//The Thirty-Fourth AAAI Conference on Artificial Intelligence. Palo Alto: AAAI, 2020: 3817-3824.

（责任编辑：韩　啸）