

本科生毕业设计

|  |
| --- |
| 不同通信模式下分布式深度学习模型训练方法效率评估 |

|  |  |
| --- | --- |
| 院 系 | 计算机科学与技术 |
| 专业班级 | CS1608 |
| 姓 名 | 蔡灯 |
| 学 号 | U201614739 |
| 指导教师 | 李瑞轩 |

2020年06月10日

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘 要

深度学习在语音识别，图像分类，自然语言处理等领域取得显著的成果，但是训练深度学习模型需要大量的计算力，这需要耗费大量的时间。而在深度学习中引入分布式训练极大缩短训练时间成为可能。分布式训练随着节点数的增加，通信开销显得极为重要，如何选择合适的通信模式很大程度上决定了其训练效率和准确率。

分布式深度学习框架分为两部分：优化算法，通信模式。优化算法使用了基于动量的随机梯度下降，显著提高了训练速度。通信模式选择了同步模式，异步模式，限制异步频率的部分同步模式,补偿梯度延迟的异步模式。此外，构建了节点计算能力不均衡，通信状况不均衡的分布式环境,使用parameter-server通信架构。针对数据集mnist和cifar10使用卷积神经网络训练。实验结果表明，在相同迭代次数下，在节点计算能力均衡下，通信状况良好的情况下，同步模式下准确率最高，训练稳定，训练时间比其他两种通信模式差距不大；在节点计算能力不均衡，通信状况不稳定的情况下，异步模式准确率较其他两种模式差距微乎其微，训练时间比同步模式差距较大。同时数据量越大，训练时间和准确率的差距体现的越明显。

**关键词**：分布式深度学习；随机梯度下降；同步；异步；限制异步频率的部分同步;参数服务器

Abstract

Deep learning has achieved remarkable results in speech recognition, image classification, natural language processing and other fields, but training the deep learning model requires a lot of computational power, which takes a lot of time. However, it is possible to introduce distributed training in deep learning to greatly shorten the training time. With the increase of the number of nodes, the communication cost of distributed training becomes very important. How to choose an appropriate communication mode largely determines its training efficiency and accuracy.

The distributed deep learning framework is divided into two parts: optimization algorithm and communication mode. The algorithm uses random gradient descent based on momentum to improve the training speed. The communication mode selects synchronous mode, asynchronous mode, partial synchronous mode which limits the asynchronous frequency, Asynchronous mode that compensates for gradient delay. In addition, the distributed environment with unbalanced computing power and unbalanced communication is constructed. Convolutional neural network was used to train mnist and cifar10 data sets. The experimental results show that under the same number of iterations,the condition of good communication condition and the equilibrium of computing power of nodes, the accuracy is the highest, the training is stable, and the training time is not much different from the other two communication modes. In the case of unbalanced computing capacity of nodes and unstable communication, the accuracy of asynchronous mode is very smaller compared with the other two modes, and the training time is larger than that of synchronous mode. At the same time, the bigger the amount of data is, the more obvious the difference is between training time and accuracy.

**Keywords:** Distributed deep learning; Stochastic gradient descent; Synchronou-s; Asynchronous; Partial synchronization that limits the asynchronous frequency; Asynchronous mode that compensates for gradient delay; parameter-server

目 录

[摘 要 I](#_Toc40008605)

[Abstract II](#_Toc40008606)

[1 绪 论 1](#_Toc40008607)

[1.1 课题背景 1](#_Toc40008608)

[1.2 国内外研究现状 2](#_Toc40008609)

[1.3 研究目的和主要内容 3](#_Toc40008610)

[1.4 论文结构 4](#_Toc40008611)

[1.5 课题来源 4](#_Toc40008612)

[2 CNN及其优化算法SGD 5](#_Toc40008613)

[2.1 卷积神经网络 5](#_Toc40008614)

[2.2 同步SGD 7](#_Toc40008615)

[2.3 异步SGD 7](#_Toc40008616)

[2.4 限制异步频率的部分同步SGD 8](#_Toc40008617)

[2.5 补偿梯度延迟的异步SGD 9](#_Toc40008618)

[2.6 基于动量的SGD 10](#_Toc40008619)

[2.7 本章小结 11](#_Toc40008620)

[3 节点间通信架构 13](#_Toc40008621)

[3.1 Parameter-server 13](#_Toc40008622)

[3.2 Ring All-Reduce 14](#_Toc40008623)

[3.3 本章小结 17](#_Toc40008624)

[4 分布式编程实现 18](#_Toc40008625)

[4.1 计算节点实现 18](#_Toc40008626)

[4.2 参数服务节点实现 19](#_Toc40008627)

[4.3 构建CNN模型 20](#_Toc40008628)

[4.4 设计中考虑的制约因素 20](#_Toc40008629)

[4.5 成本估算 21](#_Toc40008630)

[4.6 本章小结 21](#_Toc40008631)

[5 实验结果与分析 22](#_Toc40008632)

[5.1 分布式环境 22](#_Toc40008633)

[5.2 实验结果与分析 22](#_Toc40008634)

[5.3 本章小结 24](#_Toc40008635)

[6 总结与展望 25](#_Toc40008636)

[致 谢 26](#_Toc40008637)

[参考文献 27](#_Toc40008638)

# 绪 论

本章主要介绍了分布式深度学习的研究背景，面临的挑战和趋势，国内外的发展现状，同时还简要阐明了该课题的来源，研究目的和主要内容，以及介绍了论文结构。

## 课题背景

### 研究背景

随着互联网的蓬勃，数据正在急剧增长，世界上的互联网公司和用户每天都在产生数以TB计的数据，这些海量数据中心又含有大量的有价值的信息 [1]。重新焕发光彩的深度学习是一种能够高效准确的提炼这些价值的优良工具，近年来深度学习在视觉[2]、语言[3] 和智能推理[4] 领域的复杂任务上大放异彩。深度学习和无监督学习已经证实，模型的性能的提高可以通过大规模数据来训练复杂的模型。但是，实际中深度网络模型往往有数百万甚至数十亿个参数需要训练，这使得问题变得棘手。第一个在 ImageNet  分类任务上取得当前最佳结果的深度学习算法在单个 GPU 上训练足足耗费了一周时间。显然在现在数据飞速增长的时代，这样的训练效率极为不够的，因为现在的模型训练所使用的真实数据和ImageNet 数据集数据量相比都相形见绌。现在存在以分布式深度学习训练的极大提高训练效率，同时还能够达到在单GPU上相当的准确率。理论上，分布式训练的速度和训练节点数量成线性关系，同时还具有在高网络延迟下具有收敛性以及对网络断开的容错性。

### 分布式深度学习训练策略

分布式深度学习策略总体上分为两种：模型并行，数据并行。

在模型并行中，单个的模型拆分在多个节点上。模型训练的效果取决于模型并行结构的优劣。具有更多数据的模型可以动态的分配更多的计算力，因此，在大兴模型使用模型并行策略会显著提高性能，从而缩短训练时间。一般来说，只有当模型架构太大以至于单台机器无法有足够的内存可以容纳并且某些模块可以并行化时，才能应用模型并行化。模型并行化可用在某些模型中，比如目标检测网络[5]**。**，这种模型的绘制边界框和类别预测部分是相互独立的。一般而言，大多数网络只可以分配到个位数GPU上，这制了可实现的可扩展数量。

在数据并行中，数据被均等划分给不同的节点。每个节点拥有模型的一个副本。每个节点训练器所划分的数据集的一个子集并且更新局部模型参数。同时这些局部参数是全局共享的，从而通过累积算法更新一个全局模型参数，这个全局模型参数又会分发到各个节点，作为其新的局部参数。相比较模型并行，数据并行方式能够支持更大的训练规模，提供更好的扩展性，因此数据并行是深度学习最常采用的分布式训练策略。

### 面临的挑战和趋势

传统的云计算环境下，参与训练的计算节点均在强大的数据中心，而随着5G等标志着物联网时代到来的新兴技术，计算将更多地偏向端设备。那么主要的挑战是在保证准确率的上基础上降低通信开销且具有容错性，因为端设备之间的通信状况将是极为复杂的。在更低功耗的端设备上进行训练已经以联盟学习（federated learning）[6] 的形式获得一些发展势头，也出现了一些用于安全和去中心化训练的现代深度学习框架构建模块。

## 国内外研究现状

分布式深度学习框架可分为两部分：分布式训练优化算法，节点间通信。该小节主要介绍这两部分国内外的研究现状。

### 优化算法

分布式深度学习常用的一种算法是随机梯度下降（SGD）,该算法大致可分为同步SGD,异步SGD，限制异步频率的部分同步SGD。当然针对SGD的规则也同样可以应用到其SGD变体中，如EASGD[7]，Momentum SGD[9], Nesterov accelerated gradient(NAG)[9]，同样也可以应用到其他算法如Adam [10]、RMSProp [11] 等等 [12]。

同步SGD[13]是所有节点使用相同的模型参数来训练本地数据集。等到所有节点mini-batch训练完成，收集它们的梯度进行更新，然后再分发给各个节点。在同步模式下，所有节点的的模型参数始终是一致的。同步SGD理论上和单机SGD是等价的。由于需要等待所有节点计算梯度，那么同步的效率会受到效率最低的节点的制约。

异步SGD[14]是节点完成一个mini-batch训练后直接更新模型参数，不需要等待其他节点。每一次更新，不同节点将会获取最新的模型参数，但是节点获取的时间是有延迟的，所以得到的模型参数也可能不一致。异步SGD的训练速度较同步快很多，节点数越多这种差距越明显，但由于梯度延迟会导致模型参数震荡，出现次优解。

### 节点间通信

节点间通信框架比较成熟的有两种：parameter-server[15]和ring all-reducer[16]。这已经是一个成熟的课题了。TensorFlow[17]和 PyTorch[18] 等现代深度学习框架已经实现了send，receive,scatter、gather、 All Reduce等过程，因为其允许在最优的时间内完成互连节点之间的梯度传输。

分布式深度学习神经网络模型训练中，计算开销与通信开销的配资决定了其模型并行性的有效性。如果通信开销占比较小，计算节点计算能力均衡，整个计算集群的结束时间是大致相同的，这样才能取得理想的训练效果。如果通信开销占比较大，节点之间的通信将会成为模型训练的瓶颈。梯度压缩和混合精度训练都是大有潜力的技术，可以提高网络的整体吞吐量。近期的一项研究[19] 已经发现使用周期性的学习率能将实现网络收敛所需的 epoch 数量降低 10 倍，使其成为了分布式训练方面一个很有潜力的研究方向。

## 研究目的和主要内容

在本次课题的研究中，基于不同通信模式的分布式深度学习模型训练方法效率的评估研究主要有以下方面的工作：（1）掌握分布式训练算法与通信原理；（2）熟悉框架的使用，能够编程实现分布式训练方法；（3）实现出不同的通信场景；（4）对实验结果提出自己的分析，比较不同通信模式下的效率。

## 论文结构

本文的主要内容如下：

第一章首先介绍了课题的产生的研究背景，其次是国内外对分布式深度学习的研究现状以及当前所面临的的挑战和发展趋势，最后阐明了课题的研究目的和主要内容。

第二章详细介绍了卷积神经网络（CNN），分布式深度学习框架的CNN所用到的优化算法部分：同步SGD和异步SGD以及限制异步频率的部分同步SGD，同时也介绍了基于动量的SGD变体。

第三章详细介绍了节点间通信的架构parameter-server。

第四章详细展示了基于分布式训练框架Pytorch的实验过程和设计中考虑的限制因素和成本估算。

第五章对实验结果提出分析，比较不同通信模式下的效率

第六章总结了所做的工作，并计划了下一步的工作以及展望了分布式深度学习的前景。

## 课题来源

本课题是在大四下华中科技大学做毕业设计选择的课题。该课题也是华中科技大学李瑞轩老师作指导。

# CNN及其优化算法SGD

上一章绪论中简要介绍了分布式深度学习训练框架，本章主要详细介绍CNN和其所使用的的优化算法部分：同步SGD，异步SGD,限制异步频率的部分同步SGD和SGD的变体：补偿梯度延迟SGD(DC-ASGD)[20] [21][22],基于动量的SGD。

## 卷积神经网络

卷积神经网络(CNN)[23]最近在图像和语音识别领域大放异彩。CNN主要包含有3层，提取特征的卷积层，降低参数的池化层，用于分类的全连接层。

介绍CNN之前，先了解神经网络的构造，如图2‑1，这是一个简单的三层神经网络，所谓神经网络就是按照一定规则连接起来的多个神经元所构成的网络。

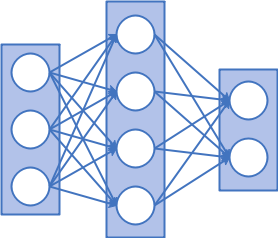


图2‑1 神经网络的基本架构

### 卷积层

在卷积层中，利用局部感知和权值共享大大减少神经网络中的参数。局部感知是受人类视觉系统所启发的，对一张图片，相邻部分联系比较紧密，距离较远则联系较弱，每个神经元没有必要对图片中所有像素点全部感知，只需要感知局部特征，然后将局部特征聚合成全局特征。所以使用局部感知器，在CNN中也就是卷积核，也可以理解为特征滤波器，在二维空间维度上其只接受图片一小块区域作为输入，一个像素点即为一个神经元，但是在深度上和输入量保持一致。另一种减少参数的方法权值共享则是卷积层的每个输出神经元和部分输入神经元局部连接，但是它们局部连接的参数是相同的，可以将这一个的卷积核参数看作是某一种特征的提取方式，和位置无关。

在这里说明一下输出数据体的尺寸，它由深度(depth)，滑动步长(stride)，零填充(zero\_paddig)来控制。深度和卷积核的数目相同，其实也代表了特征提取数目；滑动步长则表示卷积核在输入数据体上滑动的步长，为1，则代表移动一个像素点的位置，一般取1或2，只会使得输出数据体尺寸变小；零填充则是在边缘处填充0，恢复由于卷积核滑动所造成的变小的数据体尺寸。

通过卷积核计算后输出的数据体还需要通过激活函数把特征保留并映射出来（保留特征，消除一些数据中的噪声），提供网络的非线性建模能力，在卷积层一般使用Relu激活函数。

### 池化层

在连续的卷积层中间会加入池化层。池化层使用降采样的方式逐步降低数据体的空间尺寸，从而减少网络参数，减少计算量，同时也能够对过拟合有一定的抑制作用。池化层采样尺寸一般是2\*2，一般采用的池化方法有两种：MaxPool，AveragePool。MaxPool选取滑动窗口的最大值，AveragePool选取滑动窗口所有值的平均值。

### 全连接层

全连接层则和神经网络一样的。如果说卷积层，池化层和激活函数把原始数据映射到特征空间中，全连接层则起到将局部特征空间映射到样本空间，起到分类的作用。

CNN架构形式一般是INPUT -> [[CONV -> RELU] \* N -> POOL?] \* M -> [FC -> RELU] \* K -> FC，其中POOL?表示可选，用图表示如图2‑2，输入时3通道RGB原始图像，经过一层卷积后，获得深度为16，尺寸大小为32\*32的特征图谱，经过pool层后，得到深度为16，尺寸大小为16\*16的特征图谱，在经过激活函数Relu和flatten，一维化成了长度为16\*16\*16的向量，经过全连接输出一个长度为10的向量，经过softmax对其进行分类。

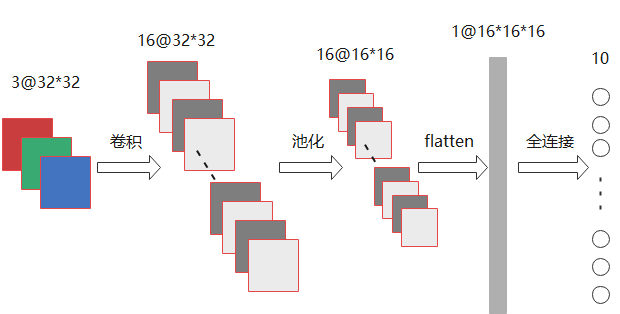


图2‑2 CNN结构示意图

## 同步SGD

随机梯度下降[24]是分布式深度学习中常用的一种优化算法。其更新公式为：

(2‑1)

式中为更新前模型参数，为学习率，为损失函数在样本点对模型参数的梯度。

同步SGD在每轮的mini-batch训练中，会等待所有的计算节点完成本地局部梯度计算，然后将每个工作节点上计算的本地局部随机梯度收集，平均得到全局梯度，并按照公式(2‑1)更新模型参数。之后，工作节点接收更新之后的模型，并进入下一轮迭代。理论上来说，同步SGD的收敛性等价于单机SGD的收敛性，但由于同步SGD要等待所有的计算节点完成梯度计算，因此同步SGD的训练效率会受到计算能力最差的节点的制约。

## 异步SGD

异步SGD是节点完成一个mini-batch训练后直接更新模型参数，不需要等待其他节点，因此，异步SGD的训练效率要比同步SGD快。然而每一次更新，不同节点将会获取最新的模型参数，但是节点获取的时间是有延迟的，使用的梯度时间节点是落后于模型参数的时间节点的，所以得到的模型参数也可能不一致。因此更新模型参数的梯度是延迟的，会降低算法的稳定性的模型的准确度。 异步SGD模型参数更新过程如图2‑3。

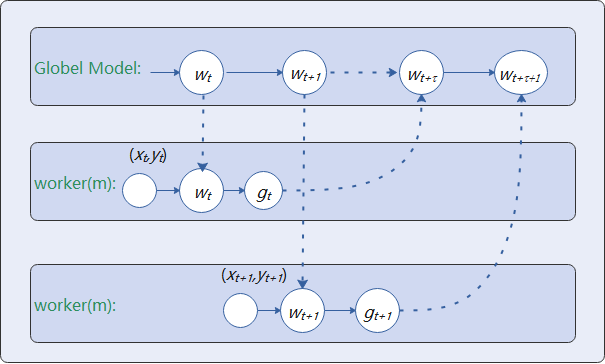


图2‑3异步SGD模型参数更新示意图

在异步SGD训练mini-batch过程中，某个计算节点work(m)在第t次迭代从全局模型获得最新模型参数，从本地随机选取mini-batch,计算出对应的随机梯度,并将随机梯度发送给全局模型。由于计算梯度需要时间，当该计算节点将随机梯度传回全局模型时，全局模型参数已被其他计算节点更新到,则异步SGD更新公式为：

(2‑2)

式中是t+τ轮更新后的模型参数，是t+τ轮当前模型参数，为

学习率，为损失函数在样本点对模型参数的梯度。

和公式(2‑1)相比，公式(2‑2)中使用的更新模型参数的随机梯度比同步SGD应该使用的产生了τ轮的延迟。故我们称异步SGD的随机梯度为延迟梯度。延迟梯度所造成的后果是：由于每次模型参数更新时使用的并不是正确的随机梯度，这会使得模型准确率降低，损失函数容易出现震荡，甚至会出现次优解，这种现象会随着计算节点数的增加愈发严重。

## 限制异步频率的部分同步SGD

限制异步频率的部分同步SGD(Limited- Asynchronous-Frequency SGD简称LAF-SGD)的模型参数更新过程和异步SGD类似，如图2‑3所示，其更新公式和异步SGD更新公式(2‑2)相同，只不过此时的τ值有取值限制。如果将τ理解成异步频率，τ=0，LAF-SGD退化成了同步SGD；τ=∞或者τ=计算节点数（在异步SGD模型参数更新中，随机梯度延迟的平均数是计算节点数[25]），LAF-SGD则进化成了异步SGD。也就是说，LAF-SGD并没有解决甚至改善梯度延迟的问题，只是限制了梯度延迟的迭代次数，所以使用该算法优化的模型的准确率和训练时间是介于同步SGD和异步SGD之间的。

## 补偿梯度延迟的异步SGD

上文提到异步SGD由于存在梯度延迟会导致模型准确率下降，而补偿梯度延迟的异步SGD（DC-ASGD）则可以改善梯度延迟，从而达到保持训练速度的同时还能提高模型准确率的效果。正确随机梯度和延迟梯度的关系如公式(2‑3)，将在处泰勒展开：

(2‑3)

式中为损失函数在模型参数处的梯度，为损失函数在模型参数的梯度，也是损失函数的Hessian矩阵,是在模型参数的梯度的Hessian矩阵。从公式(2‑3)中可以看出，异步SGD的延迟梯度是真实梯度的零阶近似，其与各项产生了延迟。于是一个很自然的想法，将所有的高阶项都计算出来，就可以修正延时梯度为真实梯度。然而，高阶项是无穷多个的，所以我们在牺牲一定精度的情况下选择使用一阶近似，则真实梯度的近似公式变为：

(2‑4)

然而还存在一个问题，现代深度神经网络中具有上百万参数甚至更多，计算和存储Hessian矩阵几乎是不可能的。寻找Hessian矩阵的近似则成为了DC-ASGD能否有效的关键。由费舍尔信息矩阵的定义可知，Hessian矩阵的一个渐进无偏估计如公式(2‑5)所示：

(2‑5)

同时根据前人的探究，在深度神经网络让Hessian矩阵的对角元矩阵来近似Hessian矩阵,在保持精度的同时还能降低运算量和存储量，因此我们可以使用diag()来近似损失函数的Hessian矩阵。同时，引入一个(0,1]之间的参数λ对偏差和方差进行调节。由此，DC-ASGD的更新公式如公式(2‑6)所示：

(2‑6)

式中是t+τ轮更新后的模型参数，是t+τ轮当前模型参数，是t轮模型参数，为学习率，为损失函数在mini-batch对模型参数的梯度，是t轮时对损失函数的Hessian矩阵近似的方差调节参数，(0,1]。

从公式(2‑6)中可以看到，对延迟梯度的补偿几乎没有增加计算量和存储复杂度。

## 基于动量的SGD

如果把梯度下降的过程比作小球从山坡滚到山谷，SGD则可看成是小球每走一步都是按照当前位置最陡的方向走的，那么基于动量的SGD（Momentum SGD）则可看成小球带有初速度，和当前位置的梯度所产生的速度合成，从而加速向山谷滚去，其下降过程可由图2‑4所示：

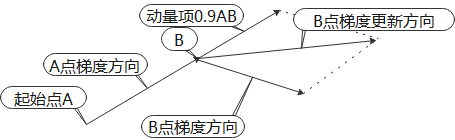


图2‑4 动量SGD梯度更新示意图

动量SGD的更新公式如(2‑7) (2‑8)下：

(2‑7)

(2‑8)

式中表示t-1轮时的累积梯度即动量项，其中，为学习率，表示t轮的动量项，表示累积梯度衰减系数，是损失函数在模型参数处的t轮梯度,表示t轮时的模型参数，表示t+1轮时的模型参数。

事实上，公式(2‑7) (2‑8)也等价于以下形式(2‑9)(2‑10)：

(2‑9)

(2‑10)

式中是t轮时累积梯度，是t-1轮累积梯度，是累积梯度衰减系数，是损失函数在模型参数处的t轮梯度,表示t轮时的模型参数，表示t+1轮时的模型参数。

动量SGD的更新公式写成公式(2‑9)(2‑10)这种形式，和同步SGD的更新公式(2‑1)，异步SGD的更新公式(2‑2)，DC-ASGD的更新公式进行对比，我们发现，他们的形式是一致的，只是对于使用的梯度的处理不同。同时写成公式(2‑9)的形式和图2‑4对比更加形象。由此，我们很容易的将动量应用到以上4种算法中区去。SGD的更新过程中步幅只取决于当前位置的梯度，而在动量SGD中，每一步的步幅不仅取决于当前位置梯度，还取决于过去的梯度，其中公式(2‑9)中取值越大，说明对于过去梯度的依赖就越大，由于,随着迭代次数的增加，对于之前的梯度的依赖会越来越小，同时累积的梯度并不会越来越大，其存在一个极限值。假设每次获得的梯度都是相同的，对于公式(2‑7)，使用比值法易知该公式在∞处存在极限，等式两边取极限得到

(2‑11)

式中表示能达到的最大下降速率，表示学习率，表示梯度衰减系数，表示梯度。当时，动量SGD的最大下降速度是SGD的10倍。一般而言，取值0.5,0.9,0.99，对应的梯度下降速率是2倍，10倍，100倍，同时的调参优先级没有学习率那么高，取固定值即可。因此，动量SGD能够加速训练过程。

动量SGD的优点：(1)下降初期，使用上一次的累计梯度，梯度方向一致，能起到加速下降的作用；(2)下降中后期，最局部最小值处来回震荡时，梯度下降值更大，能够跳出陷阱；(3)在梯度下降方向改变时，减少更新，从而使得梯度加速向极值处下降，可以抑制震荡，加快收敛。

## 本章小结

本章详细介绍了CNN的层级结构及各层级作用,以及在分布式环境下所使用的模型参数更新的优化算法SGD。首先阐述了分布式环境下标准的同步SGD；异步SGD存在梯度延迟问题，会降低模型准确率，同时训练过程容易产生震荡；LAF-SGD则是同步SGD和异步SGD的折；然后又提出了异步SGD的改进版本DC-ASGD，改善了异步SGD的梯度延迟，保持模型准确率的同时不降低训练效率，最后描述了在以上这些版本使用动量来加速模型参数的收敛。

# 节点间通信架构

前面论述了分布式深度学习中的CNN和其各种版本的优化算法SGD，这一章将阐述分布式深度学习框架的另一重要部分节点间通信架构：parameter-server和ring all-reduce。

## Parameter-server

Parameter-server架构（PS架构）是分布式深度学习使用最频繁的通信架构。在PS架构中，分布式集群将节点分为两类：计算节点，参数服务节点。计算节点训练本地所分配到的数据集子集，并计算出相应的本地局部模型参数梯度，并发送给参数服务节点；参数服务节点根据从各个节点收集到的局部模型参数梯度，更新全局模型参数，同时将全局模型参数广播发送给各个节点。

其工作过程如图3‑1所示。

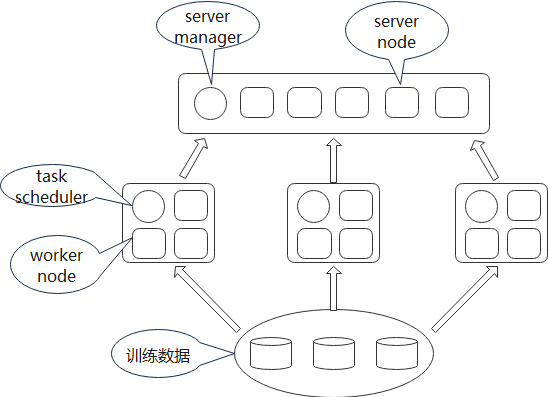


图3‑1 PS架构示意图

参数服务节点(server node)可以和其他server node通信，每个server node负责自己分配的模型参数，整个参数服务组（server group）共同维持全局的模型参数更新。

参数服务管理节点（server manager node）负责管理server node，比如各个server node 的模型参数的一致性以及server node的模型参数的分配情况。

计算节点(worker node) 之间没有通信，只和其对应的server node通信。每一个计算节点组(worker group)都有一个task scheduler,负责向worker node分配任务，并监控worker nodes的运行情况。当worker node 退出和重新加入，task scheduler 负责重新分配任务给worker node。这使得worker group可以灵活变更, 可以支持多任务的并行运算。

PS架构的优点：(1)支持异步通信可以减少通信；(2)宽松的一致性使得可以在模型收敛速度和模型准确度上做平衡；(3)使用分布式hash表可以让server node和worker node动态加入，不需要重启系统;(4) 机器错误恢复时间短;(5) 全局共享的参数使用向量和矩阵表示，这又可以很方便的使用高性能多线程优化。

## Ring All-Reduce

PS架构中，当worker node数目较多时，PS节点的网络带宽将成为架构的瓶颈。而Ring All-Ruduce由于充分利用每个worker node的网络带宽从而使得架构扩展性更强。在Ring All-Ruduce中，没有将节点进行区分，每一个节点既行使了计算节点的功能，同时也承担一部分参数服务节点的功能，不在需要中心节点来聚合模型参数的梯度。该架构让所有节点组成一个逻辑上的通信环路。每一个节点从上行接收数据，同时也向下行节点发送数据，因而，能够可以充分利用每一个节点的上下行网络带宽。其架构如图3‑2所示。

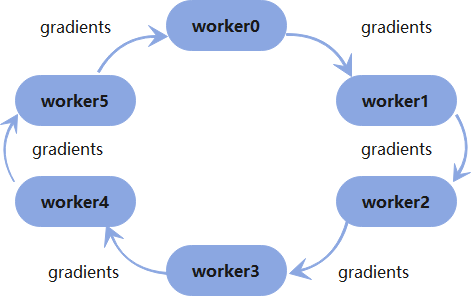


图3‑2 Ring All-Reduce架构示意图

假定节点数为num\_worker,使用Ring All-Reduce架构的基本流程如下：

1. 把每个worker node 的梯度张量分割成长度基本相等的num\_worker个分片；
2. ScatterReduce阶段：通过 num\_worker - 1 轮通信和相加，在每个 wo-rk node上都计算出一个梯度张量分片的和；
3. AllGather 阶段：通过 num\_worker - 1 轮通信和覆盖，将上个阶段计算出的每个梯度张量分片的和广播到其他work node；
4. 在每一个work node上合并梯度张量，得到梯度和在取其平均值。

下面以3个work node为例，详细展示整个流程。

分割梯度张量为3个分片，如图3‑3所示。

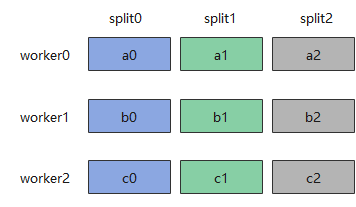


图3‑3梯度张量分割

ScatterReduce阶段如图3‑4所示，经过2轮后，每个worker node都获得了一个梯度张量分片的和。

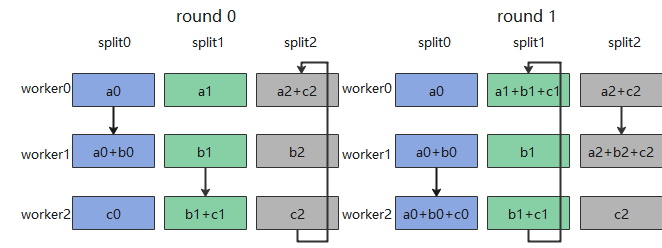


图3‑4 ScatterReduce 过程

AllGather阶段如图3‑5所示，经过2轮后，每一个worker node的每一个梯度张量分片上都有各个worker node相应分片的梯度和。

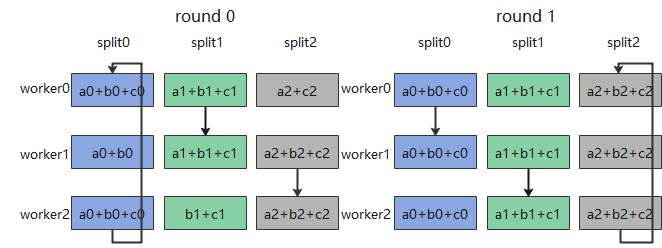


图3‑5 AllGather 过程

将每个worker node的梯度张量分片求其平均，再合并分片成完整梯度张量，如图3‑6所示。

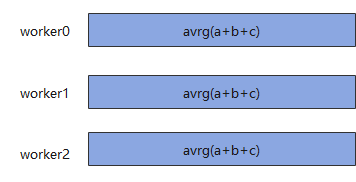


图3‑6 梯度张量分片合并

从上述过程可以看出，通信数据量的上限不会随着节点数增多而增加，一次Ring All-Reduce的通信数据量是,因此相比PS架构中，Ring All-Reduce中带宽是优化的，因为分布式集群中的每一个worker node的上下行带宽都被充分利用。在深度学习过程中，神经网络的训练，由于使用BP算法，后面层的梯度比前面层的梯度先计算，使用Ring All-Reduce可以利用这个特性，在计算前面层梯度的同时传递后面层的梯度，从而提高训练效率。

## 本章小结

本章详细描述了分布式深度学习框架中的节点间的通信架构：parameter-server和Ring All-Reduce。parameter-server将设备分成计算节点和参数服务节点，具有高扩展性和容错率，但是通信带宽会成为系统扩展性的瓶颈。Ring All-Reduce充分利用节点的上下行带宽，其通行数据量不会随着节点扩展而增加，是通信带宽优化的。

# 分布式编程实现

在本章我们将在pytorch上实现基于PS通信架构的分布深度学习模型训练。在PS架构中，我们需要分别实现计算节点和参数服务节点的功能，还构建了基于mnist数据集简单的CNN和基于cifar10数据集的较为复杂的CNN。同时还阐述了实验的现实制约因素和成本估算。

## 计算节点实现

计算节点的主要功能是计算梯度，其实现过程如：

(1)首先每个计算节点按照节点序号（rank）在本地加载数据集的数据子集，不需要节点间数据传输，以免占用大量时间。同时还要对数据集预处理，在mnist数据集上按照期望值为0.1307，标准差为0.3081正则化；在cifar10数据集上按照期望值为(0.485,0,456,0.406)，标准差为(0.229,0.224,0.225)正则化；

(2)接着注册一个网络任务队列，便于参数服务节点任务分配，注册一个分布式进程空间，该空间内以rank作为通信地址来通信，计算节点需要知道参数服务节点的IP地址和端口，为了方便运行和测试，将参数服务节点的设备IP地址设置为静态IP，其进程端口号也固定设置；

(3)然后本地启动一个训练进程，训练进程中加载CNN模型，选择交叉熵损失函数，从参数服务节点获取相同的随机模型参数，之后开始epoch(一个epoch对应整个数据子集全部训练一遍)循环训练，在该循环中，每训练一个mini-batch，都向参数服务节点发送计算好的梯度，同时等待接收参数服务节点发送的更新后的模型参数，这两个通信过程是阻塞的；

(4)模型训练完成后，在训练集和测试集上预测，计算准确率，并发送给参数服务节点聚合，同时绘制出损失曲线，观察其收敛性。计算节点的实现流程如图4‑1所示。

在作为计算节点的所有设备上均运行该py文件时，命令行参数除了rank不同，其他参数设置相同。

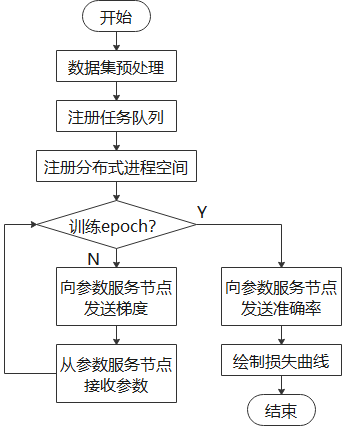


图4‑1计算节点工作流程图

## 参数服务节点实现

参数服务节点猪要负责模型参数更新，实现过程如：

(1)注册一个和计算节点相同设置的网络任务队列，同时注册一个分布式进程空间，初始化设置IP地址设为该设备的IP地址，指定一个空闲端口号；

(2)本地启动一个参数服务进程，加载CNN模型，随机化模型参数，向各个计算节点分发模型参数，保持所有节点的初始模型参数一致之后开始epoch循环训练，在这之前维护一个记录有各计算节点时间戳的序列，时间戳初始化均为0，用于同步。在该循环中，等待计算节点发送梯度，在DC-ASGD中，需要对梯度按照公式(2‑6)修正梯度，使用修正梯度更新模型参数，计算序列中时间戳的最大差距，如果超过容忍步数，则将计算节点序号加入等待队列，否则发送给相应的计算节点最新的模型参数，同时检查等待队列，满足容忍步数，发送模型参数。

(3)模型训练完成后，记录训练时间，收集计算节点的准确率并记录。其工作流程如图4‑2所示。

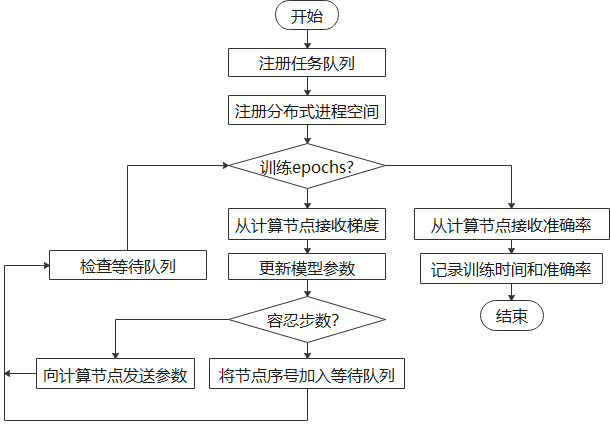


图4‑2 参数服务节点工作流程

## 构建CNN模型

CNN模型构造形式conv2d(in\_channels,out\_channels,kernerl\_size,stride,padding)->Relu->maxpool->flatten->linear(in\_size,out\_size),其中conv2d表示卷积层，in\_channels表示输入深度，out\_channels表示输出深度，kernerl\_size表示卷积核大小，stride表示卷积核滑动步长，padding表示零填充长度，Relu表示relu激活函数，maxpool表示最大值池化函数，flatten表示张量一维向量化，linear表示全连接层，in\_size，表示输入向量大小，out\_size表示输出向量大小。

mnist数据集上的CNN模型为：conv2d(1,16,5,1,2)->relu->maxpool->conv2d(16,32,5,1,2)->relu->maxpool->flatten->linear(32\*7\*7,10);cifar10数据集上的CNN模型为：conv2d(3,32,3,1,1)->relu->maxpool->conv2d(32,64,3,1,1)->relu->maxpool->conv2d(64,128,3,1,1)->relu->flatten->linear(128\*8\*8,10)。

## 设计中考虑的制约因素

本文主要目的主要是讨论不同通信模式下分布式深度学习模型训练方法的效率评估。其中分布式环境的选取至关重要，分布式环境一般是在云上的计算集群，由于经济因素，只能选择使用vmvare在本人8g内存，AMD显卡的Windows 10系统上搭建了一个简单的分布式环境。由于只有8G内存，最多只能虚拟化4个计算节点和一个参数服务节点，节点数过少，每个节点的内存很小，导致无法使用复杂的CNN模型，实验结果差异也较小。尤其是在cifar10数据集上，其准确率只能训练到80%左右，AMD不支持cuda，训练只能在cpu上进行，训练速度较慢。

## 成本估算

选用putnam模型进行开发成本估算,由于开发过程比较简单。模型中L=785，td=0.25年,Ck=8000,由公式K = L3/(Ck3\*td4)=0.24人年。

## 本章小结

本章详细描述了在计算节点和参数服务节点的工作流程以及相应数据集的CNN模型的构建，以及对实验制约因素和成本估算的说明。

# 实验结果与分析

本章主要展示了在不同分布式环境、不同通信条件、不同通信模式下的实验结果以及对其结果的分析。先详细介绍了在本地计算机上虚拟化构建的分布式环境，其次介绍了4组对比实验结果。

## 分布式环境

受计算机硬件的限制，在Windows10上使用VMVARE构建了两种分布式环境，所有节点的操作系统都是ubuntu18.04：4台1G内存计算机节点，1台2G内存参数服务节点，节点计算能力均衡；2台1G内存计算节点，2台2G内存计算节点，同时共用1台2G内存计算节点作为参数服务器，节点计算能力不均衡。由于所有节点都是虚拟机下，网络通信在一个局域网下，网络延迟基本一致，故为了模拟真实网络通信情况，在计算节点中通信部分等待一个随机的延迟时间，随机延迟时间为[20,100]。

## 实验结果与分析

(1)在分布式环境上，没有模拟网络延迟，在mnist数据集上训练10epochs,在cifar10数据集上训练20epochs,使用同步SGD(SSGD)，异步SGD(ASGD)，限制异步频率的部分同步SGD(LAF-SGD-n(n表示容忍步数)，DC-ASGD优化算法，训练CNN模型，训练结果见表5‑1，训练结果均取训练10次的平均值。

表5‑1 分布式环境1无随机延迟训练结果

|  | Mnist | | | Cifar10 | | |
| --- | --- | --- | --- | --- | --- | --- |
| Train\_acc | Test-acc | time | Train\_acc | Test-acc | time |
| SSGD | 0.9829 | 0.9826 | 1053 | 0.8078 | 0.8012 | 3558 |
| LAF-SGD-1 | 0.9821 | 0.9820 | 1044 | 0.7998 | 0.7964 | 3488 |
| LAF-SGD-2 | 0.9815 | 0.9816 | 1031 | 0.7967 | 0.7922 | 3425 |
| LAF-SGD-4 | 0.9809 | 0.9805 | 1018 | 0.7912 | 0.7892 | 3374 |
| ASGD | 0.9807 | 0.9802 | 1013 | 0.7902 | 0.7886 | 3355 |
| DC-ASGD | 0.9824 | 0.9822 | 1019 | 0.8012 | 0.7998 | 3371 |

(2)在分布式环境上，没有网络延迟，训练结果见

表5‑2(无特别说明，后面实验条件均和(1)相同)。

表5‑2 分布式环境2无随机延迟训练结果

|  | Mnist | | | Cifar10 | | |
| --- | --- | --- | --- | --- | --- | --- |
| Train\_acc | Test-acc | time | Train\_acc | Test-acc | time |
| SSGD | 0.9827 | 0.9822 | 1058 | 0.8065 | 0.8012 | 3548 |
| LAF-SGD-1 | 0.9822 | 0.9818 | 1028 | 0.7989 | 0.7928 | 3452 |
| LAF-SGD-2 | 0.9813 | 0.9810 | 1008 | 0.7963 | 0.7902 | 3384 |
| LAF-SGD-4 | 0.9807 | 0.9803 | 992 | 0.7902 | 0.7868 | 3294 |
| ASGD | 0.9804 | 0.9801 | 989 | 0.7892 | 0.7854 | 3282 |
| DC-ASGD | 0.9823 | 0.9821 | 998 | 0.8012 | 0.7998 | 3301 |

(3) 在分布式环境上，加载随机网络延迟，训练结果见表5‑3

表5‑3分布式环境1有随机延迟训练结果

|  | Mnist | | | Cifar10 | | |
| --- | --- | --- | --- | --- | --- | --- |
| Train\_acc | Test-acc | time | Train\_acc | Test-acc | time |
| SSGD | 0.9828 | 0.9827 | 1685 | 0.8076 | 0.8015 | 4634 |
| LAF-SGD-1 | 0.9820 | 0.9821 | 1564 | 0.8011 | 0.7955 | 4442 |
| LAF-SGD-2 | 0.9814 | 0.9813 | 1498 | 0.7961 | 0.7914 | 4224 |
| LAF-SGD-4 | 0.9804 | 0.9809 | 1345 | 0.7919 | 0.7892 | 3898 |
| ASGD | 0.9803 | 0.9802 | 1323 | 0.7912 | 0.7882 | 3865 |
| DC-ASGD | 0.9822 | 0.9820 | 1019 | 0.8024 | 0.8002 | 3884 |

(4)在分布式环境上，加载随机网络延迟，训练结果见表5‑4。

表5‑4分布式环境2有随机延迟训练结果

|  | Mnist | | | Cifar10 | | |
| --- | --- | --- | --- | --- | --- | --- |
| Train\_acc | Test-acc | Time | Train\_acc | Test-acc | time |
| SSGD | 0.9824 | 0.9822 | 1724 | 0.8054 | 0.8014 | 4378 |
| LAF-SGD-1 | 0.9818 | 0.9814 | 1578 | 0.7978 | 0.7943 | 4098 |
| LAF-SGD-2 | 0.9808 | 0.9811 | 1425 | 0.7943 | 0.7912 | 3882 |
| LAF-SGD-4 | 0.9801 | 0.9798 | 1314 | 0.7908 | 0.7874 | 3594 |
| ASGD | 0.9798 | 0.9796 | 1295 | 0.7896 | 0.7858 | 3564 |
| DC-ASGD | 0.9818 | 0.9820 | 998 | 0.8005 | 0.7984 | 3301 |

从表5‑1、

表5‑2、(3) 在分布式环境上，加载随机网络延迟，训练结果见表5‑3

表5‑3、表5‑4中可以看出，无论计算节点计算能力是否均衡、通信条件是否均衡下，SSGD拥有最高的准确率，训练时间最长；ASGD准确率最低，训练时间最短；其改进版DC-ASGD准确率接近SSGD，训练时间却和ASGD保持一致；LAF-SGD的准确率和训练时间介于SSGD和ASGD之间，而且准确率变化不大，但是训练时间差异较大。

在表5‑1中，在mnist数据集上，SSGD比ASGD训练时间多4% ；在cifar10数据集上，SSGD比ASGD训练时间多6.2%；在

表5‑2中，在mnist数据集上，SSGD比ASGD训练时间多7%，在cifar10数据集上，SSGD比ASGD训练时间多8.1%；在表5‑3中，在mnist数据集上，SSGD比ASGD训练时间多27%，在cifar10数据集上，SSGD比ASGD训练时间多19.8%；在表5‑4中，在mnist数据集上，SSGD比ASGD训练时间多33%，在cifar10数据集上，SSGD比ASGD训练时间多22.8%。SSGD比ASGD在训练效率的差距与节点计算能力是否均衡，通信条件是否均衡，模型复杂度有关。节点计算能力均衡，通信条件均衡，模型简单实适合使用同步通信；节点计算能力不均衡，通信条件不均衡，模型复杂适合使用异步通信，尤其是其优化版补偿梯度延时的异步通信；同时也可以在准确率和训练时间做权衡时考虑使用限制异步频率的部分同步。

## 本章小结

本章对在不同分布式环境下不同通信模式下的实验结果作出了分析。模型越复杂，计算节点越不均衡，通信条件越不均衡，同步SGD和异步SGD之间的准确率和训练时间差异越明显。总而言之，在不同场景，对准确率和训练效率之间做权衡，选择不同的通信模式。

# 总结与展望

本文探究了不同通信模式分布式深度学习模型训练方法效率评估。实验对比了在不同数据集上不同通信模式的模型训练时间和准确率，同时特别探究了不同的分布式环境所适应的通信模式，得出如下结论：

1. 同步模式下，模型训练时间最长，准确率最高；异步模式，模型训练时间最短，准确率最低；限制异步频率的部分同步模式下，模型训练时间和准确率均介于同步和异步之间，容忍步数越大，越接近异步模式；补偿梯度延迟的异步模式下，相较于原始异步，其模型训练时间几乎不变，准确率提升较大；
2. 在节点计算能力均衡，节点网络通信条件均衡情况下，适合用同步模式；在节点计算能力不均衡，节点网络通信条件不均衡，适合用补偿梯度延迟的异步模式。

传统的云计算环境下，参与训练的计算节点均在强大的数据中心，而随着5G等标志着物联网时代到来的新兴技术，计算将更多地偏向端设备。在这种更高的扩展性，更复杂的网络通信条件的分布式环境，学习便意味着选择合理的通信方式来训练模型，我们还需要做一下工作：

1) 以端设备构建的分布式环境下不同通信模式下的训练方法评估;

2) 探究具有更强扩展性和更高容错性的通信框架。

在更低功耗的端设备上进行训练已经以联盟学习(federated learning)的形式获得一些发展势头,有望解决通信瓶颈的问题。

致 谢

毕业论文完成之时，首先要感谢我的指导老师李瑞轩老师，他在分布式深度学习领域扎根多年，了解前沿知识，根据我的个人实际情况，选择了这个十分契合我的课题。从这个课题中，我了解到当前计算机届正在发生的历史性的变革。考虑到课题的难度，李瑞轩老师还安排了实验室的一位在这方面有丰富工程经验的师兄李成杰给我做实践指导。对于这个课题，我一开始连题目中的通信模式，分布式，深度学习，都不曾接触过。我万分感谢李成杰师兄帮我用通俗易懂的方式帮我初步了解了这几个概念，否则我连搜索使用的关键词都不准确。网上关于分布式深度学习的编程实现及其有限且简单，整个分布式深度学习的编程工作，是李成杰师兄帮我搭建了基础框架。编程中碰到的一些细节问题，有的甚至网上很难搜索到，师兄却能一语惊醒梦中人，令我茅塞顿开。

同时，我也要感谢我们班的同学，因为刚接触深度学习，很多基础知识都不知道，连搜索的关键词都不知道，和同学们的讨论和单方面的请教，让我快速了解了一些基本知识，也加深了对深度学习的理解。在安装框架时，碰到的一些问题，网上解决版本各异，往往花费大量时间还找不到正确方法，是踩过坑的同学前人栽树，我后人乘凉，他们也推荐了一些好用的写毕业论文所需要的工具。

在此，我对指导老师李瑞轩老师，李成杰师兄，班级里的同学致以真诚的谢意！

参考文献

1. Constine J. How big is Facebook's data? 2.5 billion pieces of content and 500+ terabytes ingested every day[J]. http://techcrunch. com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day, 2012.
2. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM,2017,60(6).
3. Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
4. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
5. Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.
6. Konečný J, McMahan H B, Yu F X, et al. Federated learning: Strategies for improving communication efficiency[J]. arXiv preprint arXiv:1610.05492, 2016.
7. Zhang S, Choromanska A E, LeCun Y. Deep learning with elastic averaging SGD[C]//Advances in neural information processing systems. 2015: 685-693.
8. Yu H, Jin R, Yang S. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization[J]. arXiv preprint arXiv:1905.03817, 2019.
9. Dozat T. Incorporating nesterov momentum into adam[J]. 2016.
10. Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.
11. Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural networks for machine learning, 2012, 4(2): 26-31.
12. Ruder S. An overview of gradient descent optimization algorithms[J]. arXiv preprint arXiv:1609.04747, 2016.
13. Goyal P, Dollár P, Girshick R, et al. Accurate, large minibatch sgd: Training imagenet in 1 hour[J]. arXiv preprint arXiv:1706.02677, 2017.
14. Dean J, Corrado G, Monga R, et al. Large scale distributed deep networks[C]//Advances in neural information processing systems. 2012: 1223-1231.
15. Li M, Zhou L, Yang Z, et al. Parameter server for distributed machine learning[C]//Big Learning NIPS Workshop. 2013, 6: 2.
16. Thakur R, Rabenseifner R, Gropp W. Optimization of collective communication operations in MPICH[J]. The International Journal of High Performance Computing Applications, 2005, 19(1): 49-66.
17. Abadi M, Barham P, Chen J, et al. Tensorflow: A system for large-scale machine learning[C]//12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016: 265-283.
18. Ketkar N. Introduction to pytorch[M]//Deep learning with python. Apress, Berkeley, CA, 2017: 195-208.
19. Smith L N. Cyclical learning rates for training neural networks[C]//2017 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2017: 464-472.
20. Zheng S, Meng Q, Wang T, et al. Asynchronous stochastic gradient descent with delay compensation[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 4120-4129.
21. Sun S, Chen W, Bian J, et al. Ensemble-compression: A new method for parallel training of deep neural networks[C]//Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2017: 187-202.
22. Meng Q, Chen W, Wang Y, et al. Convergence analysis of distributed stochastic gradient descent with shuffling[J]. Neurocomputing, 2019, 337: 46-57.
23. Chua L O, Roska T. The CNN paradigm[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 1993, 40(3): 147-156.
24. Bottou L. Large-scale machine learning with stochastic gradient descent[M]//Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010: 177-186.
25. Chen J, Pan X, Monga R, et al. Revisiting distributed synchronous SGD[J]. arXiv preprint arXiv:1604.00981, 2016.