分类号 XXX 密级 XXX

UDC XXX

本 科 毕 业 论 文（设 计）

隐私保护的分布式机器学习系统原型设计与实现

学生姓名 曾 寔 学号 16090022049

指导教师 高峰

院、系、中心 信息科学与工程学院

专业年级 2016 级计算机科学与技术

中国海洋大学

隐私保护的分布式机器学习系统原型设计与实现

完成日期：_____

指导教师签字：_____

答辩小组成员签字：_____

_____

_____

_____

_____

_____

_____

隐私保护的分布式机器学习系统原型设计与实现

摘 要

基于神经网络的人工智能方法近年来取得巨大的成就，但是人工智能技术往往是基于大数据，即大数据驱动的人工智能。但是数据的匮乏在一定程度上限制了人工智能的发展，与此同时，数据源之间存在着难以打破的壁垒。一方面，人工智能所需数据来源广泛，遍及各领域，且各领域数据往往是以孤岛形式存在，整合各领域的数据也面临重重阻力。另一方面，随着大数据的发展，世界各国对用户数据隐私和安全管理也日趋严格。因此，要解决大数据的困境，就要保证数据隐私的同时，让机器学习系统高效和准确的利用各自的数据进行学习。我设计并实现了一个分布式机器学习系统的原型框架，使模型可利用多方数据信息协同学习，且保证了数据隐私安全以及无预测性能损失，并为研究者提供了可拓展的算法接口。在该框架基础上，实现了现有的分布式学习算法，基于实验对其优点和局限性进行分析和讨论。

关键词： 神经网络，分布式，隐私保护

Prototype design and implementation of distributed machine learning system for privacy protection

Abstract

Artificial intelligence method based on neural network has accomplished great achievements in recent years, and artificial intelligence technology is often based on big data, that is, big data-driven artificial intelligence. But the lack of data often limits the development of artificial intelligence. At the same time, there are barriers between data sources that are hard to break. On the one hand, AI needs a wide range of data sources, covering all fields, and the data in all fields are often in the form of isolated islands, so integrating the data in all fields also faces many obstacles. On the other hand, with the development of big data, the privacy and security management of user data are increasingly strict in the world. Therefore, in order to solve the dilemma of big data, it is necessary to ensure the privacy of data, at the same time, let the machine learning system efficiently and accurately use their own data for learning. I designed a basic model of distributed machine learning system, which can use multi-party data for model collaborative training, at the same time, ensure the data privacy security and no prediction accuracy loss, and provide researchers with scalable algorithm interface. And in the experiment, I use the current commonly used neural network and data set to test the effectiveness of the system.

Keywords: neural network, distributed ML, privacy-preserving

目 录

1. 引言

1.1 概述

1956 年，在由达特茅斯学院举办的一次会议上，计算机专家约翰·麦卡锡提出了“人工智能”一词。1997 年 5 月 11 日，IBM 的计算机系统“深蓝”战胜了国际象棋世界冠军卡斯帕罗夫。2006 年，Hinton 在神经网络的深度学习领域取得突破[1]，人类又一次看到机器赶超人类的希望, 也是标志性的技术进步。2016 2017 年，AlphaGo 战胜围棋冠军。至今为止，深度学习已经取得了令人瞩目的成就，但当前的人工智能发展仍然受到很多限制。AlphaGo 使用了超过 300,000 场棋局数据训练，才取得如此成绩，由此可见，训练数据的质量和数量是影响学习模型泛化效果的一大重要因素。

在这个信息爆炸的时代，人类社会中存有大量有用的数据，但是将如此庞大的分散数据集合起来所需要的资源开销极大。与此同时，世界各国和企业对隐私数据的保护和管理方面的意识日渐加强，例如，欧盟于 2018 年 5 月提出的 the General Data Protection Regulation (GDPR)[2] 旨在保护用户的个人隐私和数据安全。中国于 2017 年颁布的《中华人民共和国网络安全法》和《中华人民共和国民法通则》要求，互联网企业不得泄露或篡改其收集的个人信息，在与第三方进行数据交易时，必须确保拟议的合同遵守法律规定的数据保护义务。使得社会中的大量数据不能被合法的收集起来（如用户手机上的数据），并使数据源之间形成壁垒，且领域间的数据以孤岛形式存在。这些法规的建立，显然将有助于建立一个更具公民性的社会，但也将对人工智能目前常用的数据交易程序提出新的挑战。

现今，传统的人工智能学习方法往往是在单一机构下，使用已预处理好的大量数据训练模型，再将模型部署至应用。其中，数据的收集和预处理需要耗费大量的资源。诸多限制例如 GDPR 和《网络安全法》使得数据的收集和预处理面临诸多限制因素，如何在不违反相关法律法规、不泄露隐私的条件下，合理地利用处于数据孤岛状态的各方数据，训练出一个有效的模型，是现如今面临的问题和挑战。

1.2 面临的挑战

联邦学习中的许多关键特性是跨学科的，解决它们可能不仅需要机器学习，还需要分布式优化，密码学，安全性，差分隐私，公平性，压缩感知，系统，信息论，统计等等，许多最棘手的问题都在这些领域的交汇处。隐私和通信效率是联邦学习中的首要问题。如何发现与防范系统外部的攻击或系统内部的稳定，譬如数据信息窃取和恶劣的参与者等问题，往往具有研究的实际价值。此外，由于数据源的不同，各个数据管理方之间的数据实际上可能并不符合独立同分布（IID）的条件，而是 Non-IID 的数据集，从分布不同的各 Client 数据中，学习高效且通用的

模型也是需要解决的问题。

1.3 国内外研究现状

联邦学习的概念最早是在 2016 年由 McMchan 等人引入的[3]，提出了利用在有限的带宽内，大量分割且不可靠的设备中分布着的不均衡或 Non-IID（非独立同分布）的数据，协同学习共同的机器学习模型的方法。自从最初引入联邦学习一词以来，它就着重于移动和边缘设备应用程序[3][4]，而经过这些年的发展和研究，对联邦学习提出了新的定义：联邦学习是一种机器学习设置，其中多个实体（客户端）在中央服务器或服务提供商的协调下协作解决机器学习问题。每个客户的原始数据都存储在本地，并且不会交换或转移，从而代替了用于立即聚合的有针对性的更新用于实现学习目标。

针对新的问题和挑战，Selective SGD[5] 有效的利用有效的带宽，使客户端可以在保护隐私的同时交流数据集信息，从而协同获得更好的模型；Deep Gradient Compression[6] 极大的降低了分布式训练算法在训练过程中的网络带宽开销。

尽管保护隐私的数据分析已经进行了 50 多年的研究，但仅在过去的十年中，解决方案才得到大规模部署（例如[7][8]）。跨设备联邦学习和联邦数据分析现已应用于消费类数字产品。Google 在 Gboard 移动键盘[9][10][11][12] 和 Android Messages 中广泛使用了联合学习。尽管 Google 率先开发了跨设备 FL，但现在对此设置的兴趣更加广泛了：例如，Apple 在 iOS13[13] 中使用跨设备 FL，用于 QuickType 键盘和“Hey Siri”的人声分类器；而 Snips 已经探索了用于热点词检测的跨设备 FL[14]。跨部门的应用程序也已经提出或者有了描述，包括再保险的财务风险预测[15] 和医疗数据分割[16]。

对联邦学习技术的需求不断增长，导致出现了许多工具和框架。其中包括 TensorFlow Federated[38]，Federated AI Technology Enabler[34]，PySyft [342]，Leaf [35]，PaddleFL[36] 和 Clara Training Framework [33]。

1.4 本文内容

本文基于 Python 和深度学习框架 MXNet 设计了一个分布式学习系统原型框架，提供了隐私保护、分布式部署、多方网络交流模块等功能，并为用户提供该方向算法研究的快速实现解决方案。文章前半部分主要讨论了神经网络和分布式学习的理论实现基础，后半部分讨论了如何架构和设计一个分布式学习系统，最后在设计实现的分布式系统框架基础上，完成一个图像分类任务的分布式学习算法，并探究和讨论现有联邦学习算法的优点和局限性。

2. 神经网络

深度神经网络从大量高维数据中提取复杂特征，并且利用这些特征建立一个输入-输出模型。其结构往往含有多隐含神经网络层，从而实现一个将高维度输入映射至低纬度输出的一个方法。

2.1 神经元模型

神经元模型是神经网络中最基本的成分。在生物神经网络中，每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，改变这些神经元的电位；如果某神经元的电位超过一个“阈值”，那么它会被激活，向其他神经元发送化学物质。将上述情形抽象描述为图 2-1所示的简单模型，称为“M-P 神经元模型”。神经元接受来自 n 个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接进行传递，神经元将接收到的总输入与神经元的阈值进行比较，然后通过“激活函数”处理以产生神经元的输出。

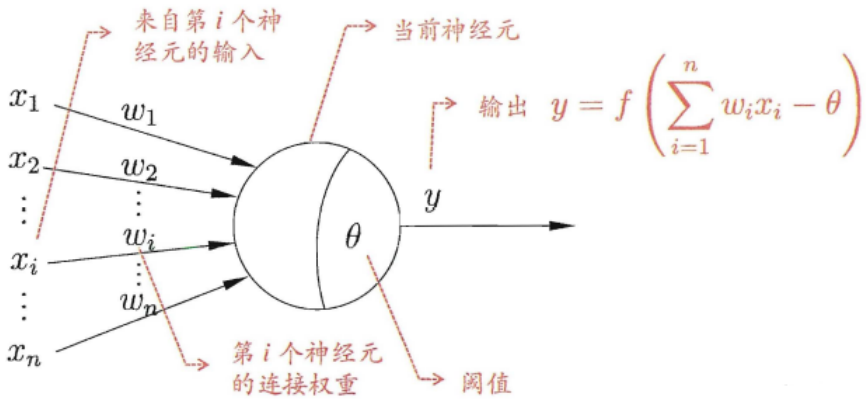


图 2-1: M-P 神经元模型

2.2 激活函数

激活函数的选择是构建神经网络过程中的重要环节。理想中的激活函数是阶跃函数，它将输入值映射为输出“0”或“1”，其中“1”对应神经元兴奋，“0”对应于神经元抑制。由于阶跃函数具有不连续、不太光滑等不好的性质，因而常选用其他函数作为激活函数。

常用的激活函数如图 2-2，Sigmoid 函数曾经被使用的很多，不过近年来，用它的人越来越少了。主要是因为它固有的一些缺点，即在深度神经网络中梯度反向传递时导致梯度爆炸和梯度消失，其中梯度爆炸发生的概率非常小，而梯度消失发生的概率比较大。Tanh 解决了 Sigmoid 函数的不是 zero-centered 输出问题，然而，梯度消失的问题和幂运算的问题仍然存在。而 ReLu 函数其实就是一个取

最大值函数。ReLU 虽然简单，但却是近几年的重要成果，它有以下几大优点：在正区间解决了梯度消失问题；计算速度非常快，只需要判断输入是否大于 0；收敛速度远快于 sigmoid 和 tanh。Leaky ReLu 为了解决 ReLu 存在的一些问题而提出并且理论上具有 ReLu 的所有优点，但是在实际操作当中，并没有完全证明 Leaky ReLu 总是好于 ReLu。

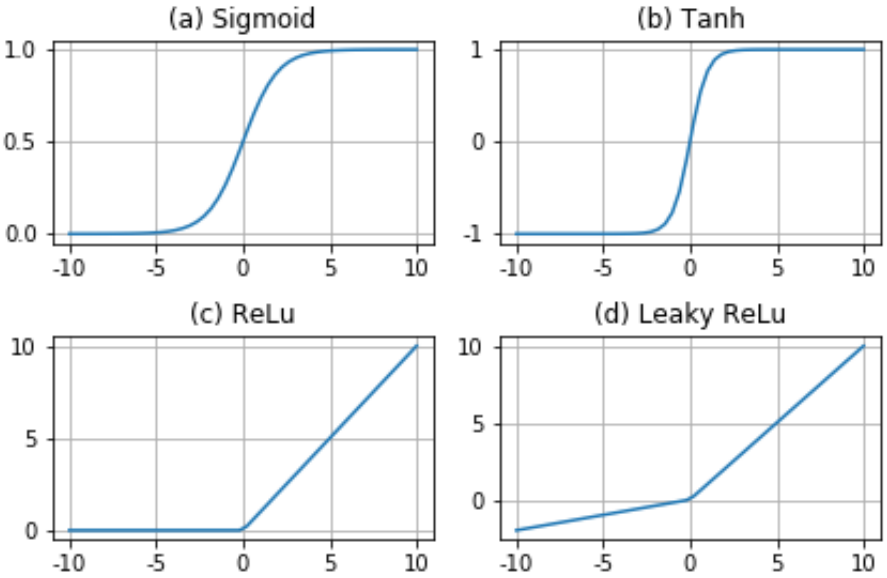


图 2-2: 常用激活函数

即使如此，ReLu 目前仍然是最常用的激活函数，在本文的实验模型中便是使用 ReLu 作为激活函数。

2.3 感知机

多层隐层感知机是最常见的学习网络架构 3-3。在一个典型的多层网络中，每个神经元接收前一层神经元的输出信号 x 和一个特殊神经元发出的偏置信号 b，然后计算其输入的加权平均 $wx+b$，称为总输入。神经元的输出是通过对输入值应用上文提到的非线性激活函数来计算的。神经网络第 k 层的输出 $a_k = f(W_k a_{k-1})$, 其中 f 为激活函数而 W_k 是决定每个输入信号贡献的权重矩阵。如果这个神经网络是一个分类模型，即将输入数据分类为有限个类（每个类由不同的输出神经元表示），那么最后一层神经网络的激活函数通常为 softmax 函数 $f(z_j) = e^{z_j} \cdot (\sum_k e^{z_k})^{-1}$, $\forall j$。在这种情况下，最后一层的任意神经元 j 输出为输入数据是属于当前类 j 的相对概率。

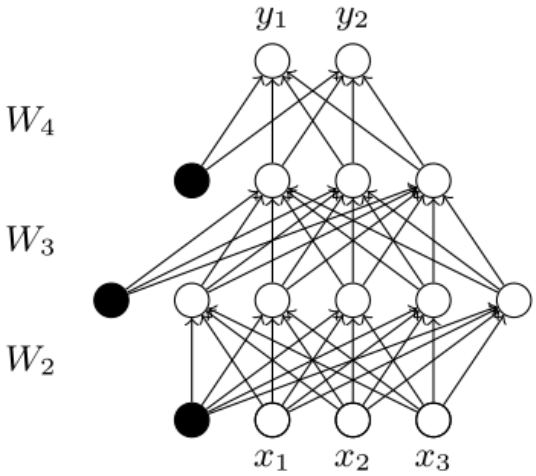


图 2-3: 多层感知机

2.4 训练方法

神经网络的学习是一个非线性函数的优化任务。在监督学习方面，目标函数是关于神经网络的输出和标签值的函数，描述了模型输出值与准确值之间的差异，模型的训练过程便是最小化目标函数的优化过程。

在神经网络的学习中，随机初始化神经元权值，作为梯度下降的起点。利用训练数据进行一次前向传播计算目标函数值，再反向传播计算每个神经元的梯度。以目标的负梯度方向调整神经网络权值，并作为下一次梯度下降的起点。重复上述过程，直到模型收敛或达到要求。关于神经网络的前向传播和反向传播计算细节可见[17]。

随机梯度下降算法（SGD）将数据集拆分为多个批次，随机抽取一个批次来计算目标函数和梯度再调整参数，所以又称为 MGBD（mini-batch gradient descent）。与标准的梯度下降法相比，随机梯度下降法在计算梯度时加入了随机因素，有机会在搜索过程中跳出局部最优。本文实验主要采用了随机梯度下降作为优化算法。

设 W 为神经网络的所有参数权值，特别的 W_j 为神经网络某一层的参数权值。E 为训练的目标函数，E 通常为 L^2 范数或者交叉熵[18]。反向传播过程会计算目标函数 E 关于每一层权值的偏导，然后通过将权值减去梯度的方法更新参数。对于单层参数 W_j 的更新方法如下：

$$W_j := W_j - \eta \frac{\partial E_i}{\partial W_j} \tag{2.1}$$

其中 η 是学习率，E_i 是在数据集第 i 个小批量的目标函数结果。

综上关于神经网络的学习过程，简单的可以描述为：

1) 随机初始化网络模型参数，作为模型迭代起点。

2) 利用训练数据集进行前向传播，计算目标函数函数。

3) 反向传播计算各层参数关于目标函数的梯度。

4) 根据式 2.1以负梯度方向调整各层网络参数，回到步骤 2)，直至模型收敛或达到要求。

3. 分布式学习

大规模分布式机器学习系统为解决怎样协调和利用大量的 GPU 集群，来快速且高效地完成深度学习模型的训练任务并达到较好的收敛结果。分布式机器学习在如何分配训练任务，调配计算资源，协调各个功能模块，以达到训练速度与精度的平衡方面，已经有了较为成熟的研究成果[19] [20] [21]。模型并行和数据并行是实现分布式机器学习的基本思想，论文[22] 提出的 DownpourSGD 结构的参数服务器 3-1很容易实现分布式深度学习，这种方式很适合大量数据和较小模型的训练。总之，现有的分布式机器学习方法主要是为了解决训练数据集太大或模型规模太大的问题，数据集仍然是集中于集群系统之中，传统的分布式学习便面临着数据收集和储存的成本问题，以及数据孤岛和隐私风险的限制。

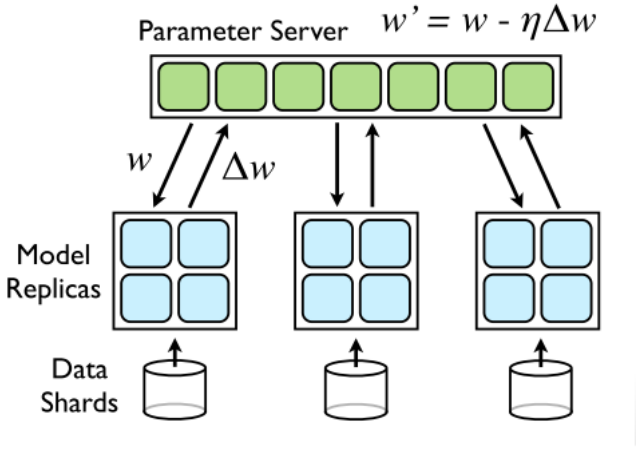


图 3-1: 参数服务器[22]

3.1 隐私保护的基本要求

分布式学习中隐私保护的基本要求即为：在模型训练的过程中，本地数据自始至终不能离开本地。因此，可以将学习过程中的模型分为全局模型和本地模型，且本地模型与全局模型保持同步。数据持有方可利用本地数据对本地模型进行训练，反向传播得到的梯度信息或已更新地本地模型信息通过网络传至全局模型管理方，用于全局模型的更新，见示意图 3-2。

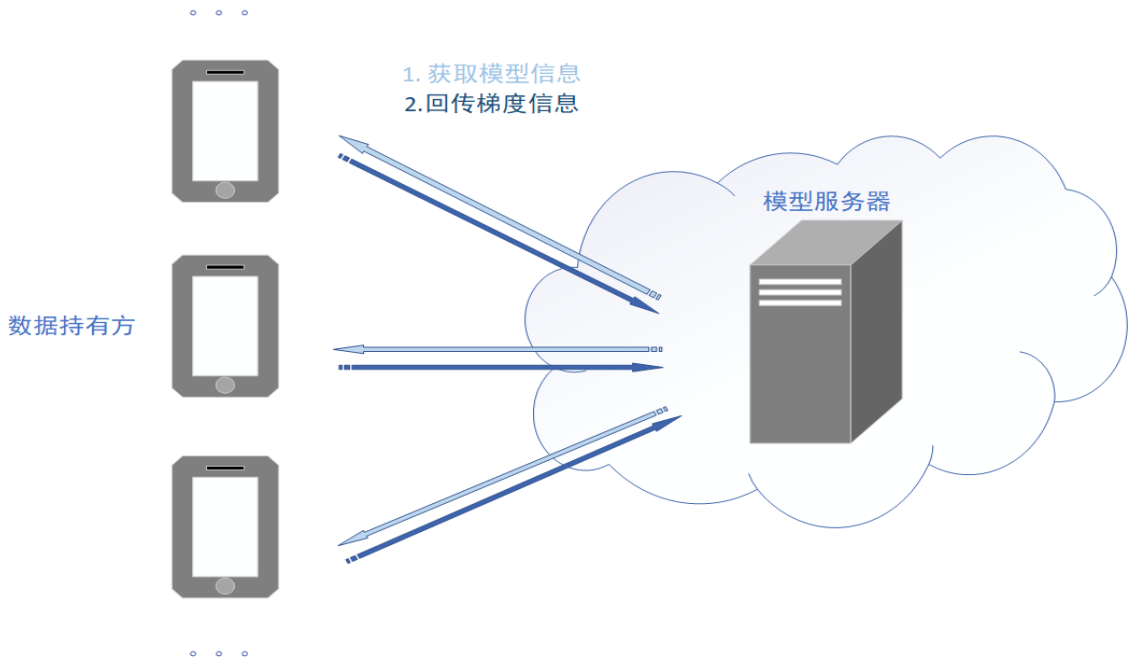


图 3-2: 信息交流示意图

3.2 学习系统架构简述

基于参数服务器的分布式深度学习系统思想和隐私保护的限制，参数服务器的通信拓扑结构是最为适合该学习算法的分布式系统架构。参数服务器的架构可以把参数服务器看做是一个媒介，在工作节点之间的媒介，负责数据交流，通信只发生在工作节点和参数服务器之间。

参数服务器负责管理全局模型，并维护与工作节点之间的通信和信息交流。工作节点即为数据的持有方，工作节点负责维护本地模型，并与参数服务器协同工作，合理利用本地数据更新全局模型。

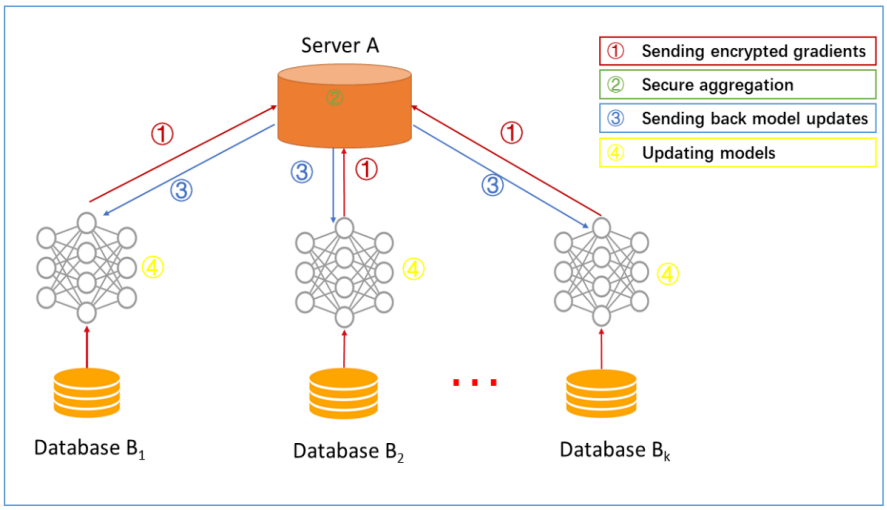
为了描述方便，仅描述参数服务器与单个 Client 之间的算法流程关系。综合隐私保护的要求，即数据不能离开本地，与 C/S 架构的分布式系统，可将分布式学习过程抽象为：

1) 参数服务器随机初始化全局模型 $W^{(global)}$，工作节点与参数服务器通信获得模型 $W^{(local)}$，特别的，$W^{(global)} = W^{(local)}$。

2) 工作节点利用本地数据执行模型的学习过程，并更新本地模型 $W^{(local)}$。

3) 工作节点将本地训练的模型或梯度上传至参数服务器。

4) 参数服务器接收信息，并根据信息更新全局模型 $W^{(global)}$。



D

图 3-3: 示意图，引自[23]

3.3 分布式最优化算法

随机梯度下降（SGD）多用于神经网络模型的训练，通过优化目标函数取得了更好的效果。而经典联邦学习方法通常是最小化以下目标函数：

$$\min_{w} F(w) = \sum_{k=1}^{m} p_k F_k(w) \tag{3.1}$$

其中，m 表示设备数量，F_k 是各个客户端的局部目标函数，p_k 为 Client 模型对应的权重，其值为 $\frac{n_k}{n}$。局部目标函数的优化处理过程为：

$$F_k(w) = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w) \tag{3.2}$$

其中，n_k 为第 k 个客户端局部样本数据数量，可以令 $p_k = n_k/n$，n 为整个联邦学习网络的数据集中符合经验最小化目标的样本总数。传统方法通过以下方式实现全局目标最优化：每一轮选择概率与 n_k 成正比的设备子集执行这些本地更新方法通过在每个设备上本地运行可变数量的迭代的优化器（例如 SGD）来实现灵活高效的通信。

优化算法的设计仍然是该方向研究的热点问题之一，合理的优化算法可以在加速模型收敛的同时，达到较好的泛化效果。在这个方面，已经有很多优秀的工作成果 [][][]，而上文的经典联邦学习便称为 Federated Averaging（后称 FedAvg）[3]，它在通信带宽有限以及数据分布不均衡的学习场景下，在多种模型和场景下能高效地训练模型并收敛至目标准确率。而后本文将在设计的系统框架下，开发并实现该算法，并对该算法的优点和局限性进行讨论，从而体现框架的可用性。具体实验内容将在第五章讨论。

3.3.1 Federated SGD

即使存有隐私风险，即数据不可离开本地的限制条件，SGD 算法仍可以自然的部署于分布式学习，本文称为 Federated SGD（后称 FedSGD）。其分为 Server 和 Client 两部分，算法伪代码见 1。在本文实现的框架中，其中 Client 端返回信息为模型或梯度取决于框架使用者的要求，是本文框架为开发者提供的功能接口。

Algorithm 1 Federated SGD
w 为模型权重参数; η 为模型的学习率; $l(w;b)$ 为目标函数。

Server Excutes:
  初始化模型权重参数 w_0
  repeat
    等待 Client 连接 Server
    if Client 请求模型 **then**
      将模型参数下发至 Clien
    else if Client 回传模型 **then**
      接收 Client 回传的模型 w_t
      Server 更新模型 $w_{t+1} := w_t$
    else if Client 回传梯度 **then**
      接收 Client 回传的梯度 $\nabla l(w;b)$
      Server 更新模型 $w_{t+1} := w_t - \eta\nabla(l(w;b))$
    else
      非法连接
    end if
  until 模型达到收敛条件
Client Excutes:
  向 Server 请求模型参数 w_t
  for batch $b \in B$ **do**
    更新模型 $w_t := w_t - \eta\nabla l(w;b)$
  end for
  返回模型 w_t 或梯度 $\sum \nabla l(w;b)$ 至 Server

FedSGD 属于同步算法，Server 端面向多 Client 端进行模型学习时，Server 需要等待当前 Client 完成任务并返回信息，且 Server 端将模型更新后才能进行下一轮的学习，使得 FedSGD 不适用于大规模学习，因为 Server 和大部分 Client 的时间用于等待，学习效率极低，其表现将在第五章讨论与分析。

3.3.2 Federated Averaging

FedAvg 算法通过实现客户端的并行化计算，在保证模型收敛的同时，降低了模型的更新轮次，从而降低了模型的收敛时间。FedAvg 算法由 3 个主要参数控制：C，每一轮迭代计算中，参与的 Client 数占所有 Client 的比率；E，Client 遍历本地数据集的次数，即 epoch；B，Client 训练时本地小批量数据的大小。特别的，当

$C = 0$ 时，即每一轮模型迭代只有一个 Client 参与，该算法退化为 FedSGD。三个参数对 Client 间的并行计算，单 Client 计算量大小做出的要求，通过提高并行和单个 Client 的计算量，使得模型收敛更快，将在第五章讨论。

对于一个拥有数据集大小为 n_k 的 Client，每一轮本地模型的更新次数由 $u_k = E\frac{n_k}{B}$。算法伪代码可见 2。算法可描述为，每一个 Client 采用本地数据集计算并对模型进行一次迭代，Server 将各 Client 取得的模型权重参数进行加权平均生成新模型，其中各 Client 的加权权重定义为其数据集大小占总数据集的比率（拥有更多数据的 Client 的模型更可信）。

Algorithm 2 Federated Averaging
Client 总数为 K 且以 k 标志；B 是本地数据批量大小；E 是本地训练 epoch；η 是学习率。

Server Excutes:
  初始化模型权重参数 w_0
  for 每一轮 t **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (随机选取 Clients 子集, 其数量为 m)
    for 每一个 Client $k \in S_t$ **do**
      $w_{t+1}^k \leftarrow$ClientUpdate(k,w_t) {各 Client 间并行计算}
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for
ClientUpdate(k, w): // 在 Client k 上运行
  $\mathcal{B} \leftarrow$ 将本地数据集以 batch 大小 B 分割
  for 每一个本地 epoch i 从 1 到 E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla l(w; b)$
    end for
  end for
  返回 w 至 Server

4. 系统架构

良好的系统架构使得系统具有高性能、高可用和可扩展性。该系统原型设计旨在为该方向研究者提供便捷的系统部署，即 Server 和 Client 的部署，并提供良好的网络通信策略，使研究者可以专注于算法的快速实现和想法的验证。

在系统架构层面，本文讨论的分布式学习系统需要广泛的点对点通信，即 Server 与 Client 之间的模型或梯度信息交流，并结合论文[22] 中参数服务器的思想，采取 C/S 架构是非常合理且自然的。基于计算机网络分层次的架构思想，本文将该分布式学习系统分为 Client 端和 Server 端两个部分，并将每个部分以功能划分为三层结构：网络层、逻辑层、计算层，见图 4-1。其中计算层负责模型的维

护，即模型定义，梯度计算，权值更新等功能，主要基于深度学习框架 MXNet[24] 实现。

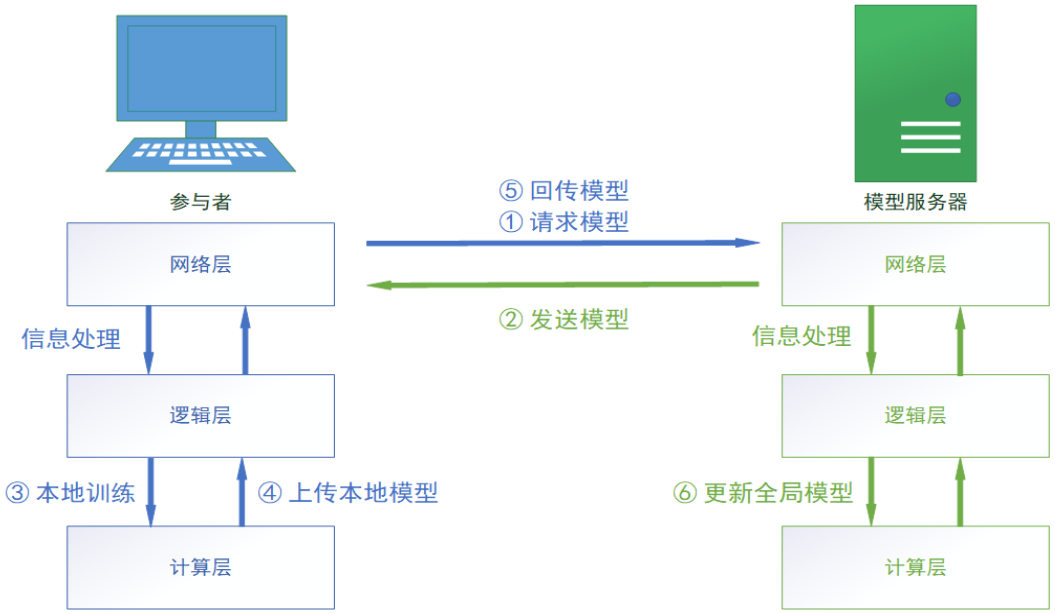


图 4-1: 架构示意图：箭头为信息流动情况，完整的一次模型更新过程为一个 round

4.1 网络层

网络层的具体任务便是处理 Server 与 Client 的通信任务。在联邦学习中的不同的应用场景，如联合移动设备上的少量数据集学习或联合各数据中心学习，网络层的通信协议也有所不同。利用移动设备上的数据时，往往要考虑移动设备当前状态：是否连接 wifi 或电量情况，涉及到通信交互的问题便是需要由网络层定义。

在本系统中，Server 端与 Client 端需要可靠的网络通信用于传输信息，因此网络通信协议采用 TCP 协议，网络通信层负责维护 Server 端与 Client 端的网络通信状态，即 TCP 连接。Server 端网络层主要负责处理来自 Client 端的请求信息，如模型参数请求或系统参数请求等，并将对应的响应任务下传至逻辑层，同时也需要处理逻辑层需要上传的信息发送任务。Client 端网络层主要任务与 Server 端基本相同，区别主要体现在需要发送或接收的数据不同。

网络层为研究者提供抽象的网络接口，如控制信息的接收与发送、文件的接收与发送、类实例的控制与发送。因此，用户只需要关注 Client 与 Server 间的数据交互关系和对应的逻辑处理，并将之送至逻辑层处理。

4.2 逻辑层

网络通信优化、模型更新算法和优化算法方向仍是联邦学习重点研究方向之一。如何高效地利用数据孤岛状态下的各方数据协同学习并合理处理 Non-IID 数据带来的问题。Deep Gradient Compression[6] 方法在保证模型精度的同时，大幅度降低了服务器与工作节点之间的通信资源消耗，论文中采用了多种方法，最核心的想法便是只传输重要的梯度信息来降低通信，可以简要概括为稀疏化更新。将梯度信息累积起来，知道当梯度值大于预设的阈值时，再将之发送至 Server，同时采用多种方法解决了这些方法带来的精度损失问题。Selective SGD[5] 算法在使 Client 之间通过可靠的 Server 分享部分信息，使得 Client 端的训练结果更可靠，并使用户可在模型准确率与数据隐私强度之间权衡。这两种方法的核心思想均是在模型信息上下传递通信前后，对模型信息进行处理，与本文框架的分层架构相对应的即为逻辑层。

通过对现有研究内容的分析与抽象，本文将逻辑层抽象为对数据信息的处理层，上文提到的两种方法的核心实现便对应本文框架的逻辑层。逻辑层作为网络层和计算层的中间层，负责处理上层往下传递的处理信号，以及处理计算层往上传递的计算结果，是整个系统架构的核心部分。逻辑层的可拓展性直接决定了框架的可拓展性。本文的框架中，为研究者提供了梯度或模型信息的流动接口，因此，研究者只需要关注如何对信息进行处理，即核心算法地实现与功能拓展。

4.3 计算层和架构总结

MXNet 是 amazon 的一个轻量化分布式可移植深度学习计算平台，在本文中用于计算层实现的核心框架。MXNet 提供强大的模型定义和自动求导机制，为计算层的实现提供了很多便利。计算层在 MXNet 框架下实现了模型定义、数据加载、模型训练、梯度计算等功能，负责模型的基本计算与维护。由于在本框架逻辑层中涉及对模型或梯度的处理，存在变形梯度或压缩梯度的情况，因此计算层为研究者提供了关于梯度和模型处理的一系列数据结构。

综合三层架构，本文的分布式学习系统可由三部分定义：通信算法、模型处理算法、模型结构，分别对应三层结构的用户自定义内容 4-2。框架为使用者提供了实现一个隐私保护的分布式系统需具备的基本功能模块。

5. 实验与分析

本章在上述框架上开发一个基础的隐私保护的分布式学习系统，在其上拓展实现 FedAvg 算法并完成一个图像分类任务，讨论现有算法的优势与局限性。在体现框架可用性的同时，并分析未来的研究难点与方向。

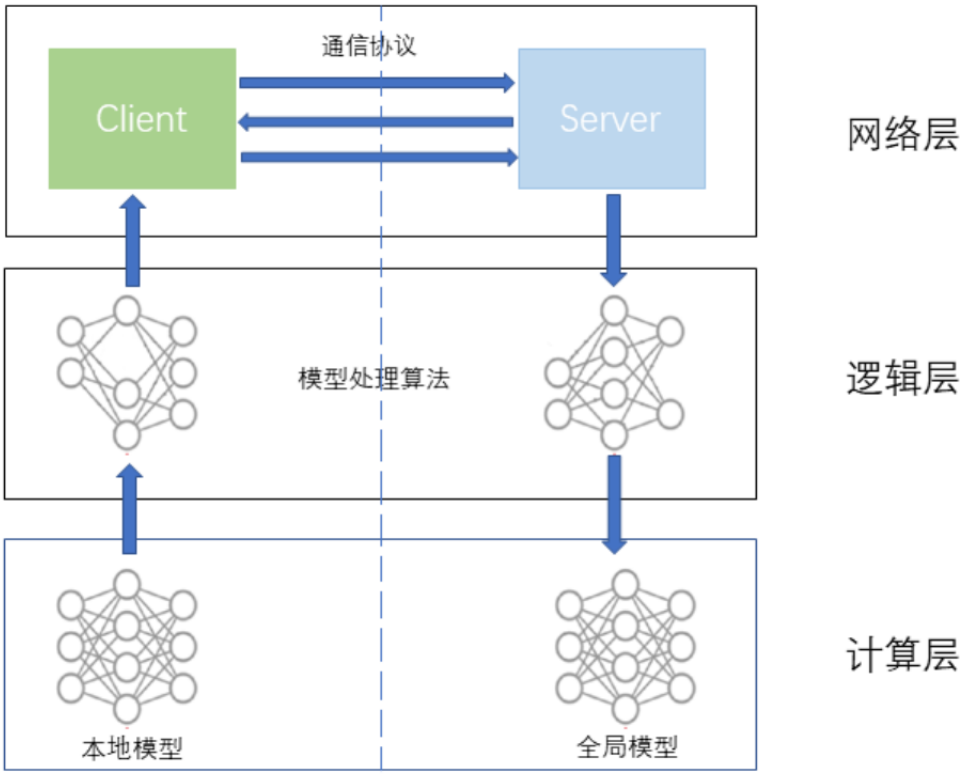


图 4-2: 框架拓展示意图

实验中，本文选择了常用的数据集 MNIST，MNIST 数据集包括 60,000 张图像的训练集和 10,000 张图像的验证集，每一张图片为 0-9 中的一个手写数字并带有标签。模型选取了简单的感知机模型（带有两个隐藏层，分别有 128 和 64 个神经元，并选取 ReLu 作为激活函数, 下文称 2NN）和卷积神经网络 LeNet-5[25]，在 Mnist 数据集上训练模型完成分类任务。

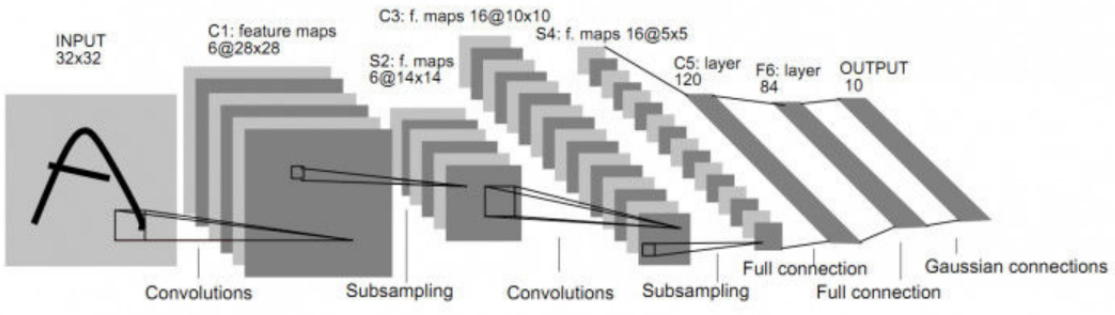


图 5-1: LeNet-5

在实验中，本文将训练数据集随机分割并分配至 100 个 Client，每个 Client 有 600 张训练图。而对于不平衡的数据集分割，本文将数据集先根据标签将数据排序，然后将数据分割为 300 张图片每份的数据子集，再随机将 2 份数据子集分配给每一个 Client。不平衡数据集中，由于 Mnist 数据集并不是平均每个标签 6000

张图，因此每个 Client 至多拥有三种且其中一种数字图极少的数据或至少拥有两种数字图数据。图 5-2和图 5-3是 Mnist 在两种数据集分割后，从 100 个 Client 中随机选出的 5 份数据的标签分布情况。

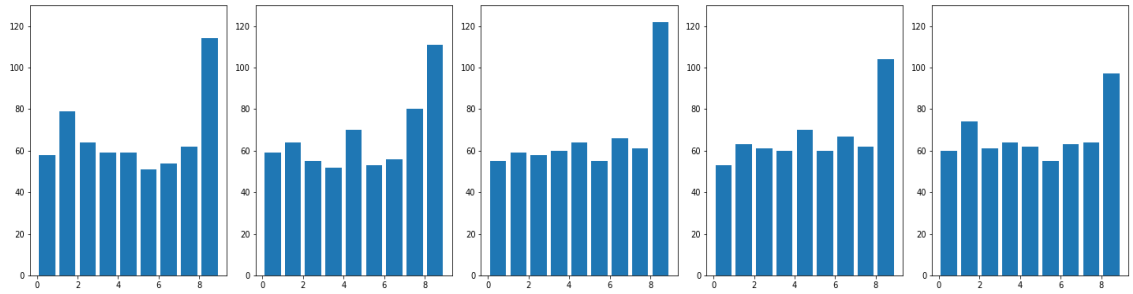


图 5-2: 随机数据集分布情况

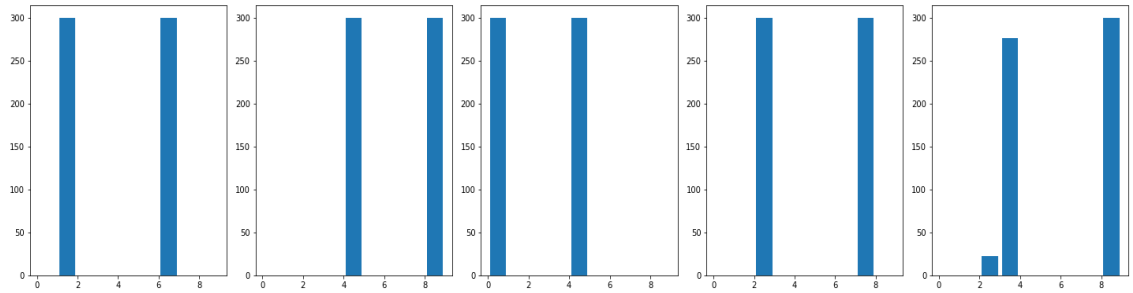


图 5-3: Non-IID 数据集分布情况

5.1 Federated SGD

本节实验着重讨论 FedSGD 的表现。在实验中，着重比较在集中的数据集与分散的数据集上训练模型时，模型的泛化性能与计算量损耗之间的差别与关系。分布式训练的资源开销包括通信资源和计算资源，而集中数据集训练下仅有计算资源损耗，为了比较的方便，本节仅对计算资源的消耗进行讨论。

本文定义对于单个数据在神经网络上做一次前向传播的计算量为 F，一次反向传播的计算量为 B。因此，一个模型的训练计算量可定义为 $nF + mB$, 其中 n 为前向传播次数，m 为反向传播次数。例如，对于 2NN 以 mini-batch 大小为 b，在数据集大小为 $\mathcal{B}$，遍历训练 e 轮数据集，所消耗的计算量为：

$$e(\mathcal{B}F_{mlp} + \frac{\mathcal{B}}{b}B_{mlp}) \tag{5.1}$$

对于 FedSGD，假设模型经过 n 轮通信完成训练，对于单轮训练的计算量可表示为 $e_{client}(\mathcal{B}_{client}F + \frac{\mathcal{B}_{client}}{b}B)$，其中 $\mathcal{B}_{client}$ 为本地数据集大小，则总计算量为 $ne_{client}(\mathcal{B}_{client}F + \frac{\mathcal{B}_{client}}{b}B)$。

实验中，分别取 2NN 和 LeNet-5 模型在验证集上的准率在 97% 和 99% 时的计算量作为基准，即模型在训练过程中，在验证集上准确率第一次超过预设准确率时的计算量。FedSGD 在单轮迭代中，本文设置 Client 的训练轮数 $E = 5$, 且有 $|\mathcal{B}_{Client}| = \frac{|\mathcal{B}|}{100}$，并由 5.1，可得在实验中，分布式训练与集中式训练的计算量关系式：

$$\frac{ne_{client}}{100}(\mathcal{B}F + \frac{\mathcal{B}}{b}B) = ne_{client}(\mathcal{B}_{client}F + \frac{\mathcal{B}_{client}}{b}B) \tag{5.2}$$

因此，分别取集中式训练时的遍历数据集轮数 e 和分布式训练时的 $\frac{ne_{client}}{100}$ 作为计算量对比数据，通过实验得表 5-1。

表 5-1: 数据集中式与分布式计算量对比

2NN	集中式训练				分布式训练，E=5			
学习率 η	B=200	B=100	B=50	B=10	B=200	B=100	B=50	B=10
0.01	84	41	23	6	83.3	43	23.55	7.55
0.02	43	21	11	3	43	23.55	13.2	5.75
0.04	21	11	6	3	23.55	14.4	10.2	5.35
LeNet-5								
0.01	132	66	29	11	127.75	91.5	36.5	13.55
0.02	54	27	18	4	71.5	35.5	26.65	11.75
0.04	34	13	11	6	45.5	36.2	12.45	11.8

实验中，不同参数的初始学习模型均为相同的随机初始化的模型，即迭代起点相同。

通过对比表 5-1中的实验数据可得，FedSGD 在各个参数下的计算量相比于数据集中式训练，在达到相同准确率时，FedSGD 均有不同程度的计算量损失，因为在被分割且数据不平衡的数据集上训练，需要通常更多次的迭代才能收敛到最优。此外，分布式计算需要额外的通信资源开销，且 FedSGD 在实际训练中，Server 与部分 Client 大部分时间都花在等待当前 Client 模型训练。完成模型训练的时间代价过大（低效率），是该算法的主要问题。

5.2 Federated Averaging

在关于 FedAvg 的实验中，着重关注 FedAvg 方法对 FedSGD 算法的提升，即模型更新轮次。实验中，仍然取模型 2NN 和 LeNet-5 第一次在验证集上达到准确率 97% 和 99% 作为基准线，比较 FedAvg 在迭代轮次上，性能的提升。实验数据见**??**，其中 $C = 0.0$ 即为 FedSGD，数据为 Server 端模型收敛至目标准确率所经过的迭代轮次 T。

参数 C 决定了系统 Client 端并行训练的强度，C 越大，越多的 Client 同时参与一轮的计算，相应地，系统的并行化更强。通过实验数据 5-2的纵向比较，对于不同的参数 C，FedAvg 算法在模型训练时的迭代轮次上的表现无较大差距，甚至在训练 LeNet-5 时，C 的增大使得算法性能有所下降。

表 5-2: FedAvg 与 FedSGD 实验结果数据

	2NN		**LeNet-5**	
C	B=10	B=100	B=10	B=100
0.0	107	288	236	724
0.1	28(3.8×)	99(2.9×)	82(2.9×)	80(9.1×)
0.2	34(3.1×)	99(2.9×)	51(4.6×)	90(9.0×)
0.5	38(2.8×)	101(2.9×)	49(4.8×)	114(6.4×)
1.0	35(3.1×)	102(2.8×)	38(6.2×)	134(5.4×)

E=5, 学习率 η=0.04，实验初始化模型与 5.1 节中的实验相同。

参数 B 和 E 决定了 Client 在单次迭代任务的计算量，理论上，计算量与参数 E 成正相关，而与 B 成负相关。根据实验数据 5-2的横向对比，在 **2NN** 和 **LeNet-5** 模型下，单机计算量的提升，即 B 的减小，相应对模型收敛速度有所提升。

表 5-3: FedAvg 迭代轮次与 E 的关系

	2NN		**LeNet-5**	
E	B=10,lr=0.01	B=100,lr=0.04	B=10,lr=0.01	B=100,lr=0.04
5	50	99	23	28
10	32	59	21	22
20	30	40	21	20

实验初始化模型与 5.1 节中的实验相同。

根据实验数据 5-3，在不同参数下，E 的提升对 FedAvg 算法效率均有不同程度的提升。在 $E = 20$ 的实验中，在 Client 端训练时，由于 Client 数据量过小，使得本地模型在训练集上过拟合，因此使得性能提升不是较为明显。

综合分析，FedAvg 相比于 FedSGD 在模型训练上表现更好，均有不同程度的速度的提升，提升效果取决于系统参数和模型结构决定的解空间的协同作用。值得一提的是，从 Client 的角度分析，在相同参数 B 和 E 下，执行 FedAvg 算法时，Client 的计算量负载更大：CK 为单轮迭代参与者数量，T 为收敛迭代轮次，FedSGD 算法下单轮迭代参与者数量为 1, 而 FedAvg 参与着数量远大于 FedSGD，因此 TCK 即为 FedAvg 算法在 Client 端的计算量负载。实验中 $K = 100$，则 FedAvg 在 Client 端的计算量消耗远大于 FedSGD。

FedAvg 相比于 FedSGD 实质上是计算量与训练时间的权衡，FedAvg 通过将模型训练过程并行化，并付出更多的计算资源获得了更短的训练时间。在实际应用场景中，人们往往希望模型训练时间更短，而不关心实际花费的计算量，因此 FedAvg 算法在实际的分布式训练场景中仍有用武之地。

5.3 不平衡数据集

在实际应用场景中，由于不同的 Client 往往处于不同的环境当中，因此各个数据源有差别，便导致了数据的分布情况有所差异。根据论文[26], 对于客户端 i 和客户端 j 的数据分布 $\mathcal{P}_i$ 和 $\mathcal{P}_j$ 有，$\mathcal{P}_i \neq \mathcal{P}_j$。在数据分布不平衡的数据集下训练的模型，都具有不同程度的性能损失。且数据的分布差异也存在不同的情况，为了便于描述，将 $\mathcal{P}_i(x, y)$ 重写为 $\mathcal{P}_i(x|y)\mathcal{P}_i(y)$ 和 $\mathcal{P}_i(y|x)\mathcal{P}_i(x)$。

1) 特征差异（数量偏移）：$\mathcal{P}_i(y|x)$ 相同而边缘分布 $\mathcal{P}_i(x)$ 不同。

2) 标签差异（数量偏移）：$\mathcal{P}_i(x|y)$ 相同而边缘分布 $\mathcal{P}_i(y)$ 不同。

3) 特征分布差异（概念偏移）：$\mathcal{P}_i(y)$ 相同而条件分布 $\mathcal{P}_i(x|y)$ 不同。

4) 标签分布差异（概念偏移）：$\mathcal{P}_i(x)$ 相同而条件分布 $\mathcal{P}_i(y|x)$ 不同。

5) 不平衡：不同标签下的数据量分布不同。

本节关注 FedAvg 和 FedSGD 能否在不平衡的数据集下仍然能使模型较好的收敛，以及它们训练模型达到目标准确率时的训练效率，并将其表现与在正常分布的数据集下训练的表现进行比较分析与讨论。为了进一步探讨上一节关于计算量与迭代轮次的联系，特别定义 $u = En/(KB)$，表示算法在 Client 端的计算负载情况。参数 u 代表一轮模型迭代（round）中，Client 端进行本地模型参数迭代的次数。

表 5-4: FedSGD 在 Non-IID 数据集的表现

LeNet-5: Accuracy>=98%; lr=0.02				
E	B	u	random	Non-IID
1	600	1	2941	2872(1.02×)
5	600	5	750	1474(0.51×)
1	50	12	319	1211(0.26×)
20	600	20	268	748(0.36×)
1	10	60	109	701(0.16×)
5	50	60	101	667(0.15×)
20	50	240	95	508(0.19×)
5	10	300	42	523(0.08×)

表 5-4为 FedSGD 算法在不同参数下在 Mnist 数据集上训练 LeNet-5 的表现情况。表中可得随着 Client 单轮模型迭代次数 u 的增加，总的迭代轮次降低。再次印证了合理增加 Client 计算量，可以提高整个系统在模型训练时的表现。在不平衡的数据集上训练，虽然上述关于计算量与系统效率的结论仍然成立，但算法

的效率大大降低。由此可见，数据集不平衡分布是联邦学习面临的一个极大的挑战。

表 5-5: FedAvg 在 Non-IID 数据集的表现

LeNet-5: Accuracy>=98%; lr=0.02; C=0.1				
E	B	u	FedSGD	FedAvg
1	600	1	2872	1466(1.96×)
5	600	5	1474	641(2.23×)
1	50	12	1211	507(2.39×)
20	600	20	738	.
1	10	60	701	.
5	50	60	667	.
20	50	240	508	.
5	10	300	523	.

表 5-5中是关于 FedAvg 算法在 Non-IID 数据集下的实验，将 C 固定为 0.1，即每轮有 10 个 Client 的本地模型参与全局模型的更新，选取 LeNet-5 在验证集准确率 98% 为目标准确率，表中数值为迭代轮次。从表格的数据分析可以得出，FedAvg 在 Non-IID 数据集上的表现仍然优秀，不同参数组合下，相比于 FedSGD 仍然有较大的提升。此外，上文关于计算量与迭代轮次的关系在 FedAvg 算法上仍然成立。

5.4 局限性讨论

FedAvg 作为联邦学习领域的经典算法，能够很好的处理不平衡数据集上的联邦学习的问题，相比于 FedSGD 在计算效率上也有很大的提升。但 FedAvg 仍存在一些局限性，根据式 3.1分析可得，FedAvg 存在模型偏向问题，其 Client 权重 $p_k = \frac{n_k}{n}$ 由本地数据集大小 n_k 决定，使得在数据集分布为 Non-IID 的情况下，全局模型往往会偏向于持有本地数据更大或者参与联邦学习迭代次数更多的 Client，即模型仅适用于部分 Client 的数据，而在某些处于弱势的 Client 的数据上表现不好。为了解决这个问题，Peng[27] 提出了 q-FedAvg 来解决模型准确度在 Client 端分布方差过大的问题，通过动态调整各个局部模型的合并权值，使得全局模型性能更平均。

在本文的实验中，FedAvg 在 Non-IID 的表现虽然有效但不够高效。在 Client 端训练时，数据分布不平衡使得模型在本地学习时容易过拟合，当多个过拟合后的模型在 Server 端合并更新时，Server 端模型往往会表现在验证集上准确率下降，而宏观来看，会表现为全局模型在验证集上准确率的抖动【图】。

6. 总结与展望

本文设计并实现了一个隐私保护的分布式机器学习框架原型，该框架采用三层架构设计，并为研究者提供了分布式系统的算法实现的快速解决方案。之后，本文基于该框架实现了 Mnist 数据集的图像分类任务学习的分布式学习算法，并拓展研究了 FedAvg 和 FedSGD 的实验，讨论了其各项表现。

将隐私保护的分布式系统部署在移动设备上是该方向的应用场景之一，在数据孤岛问题下，如果设计分布式学习算法使得多个数据持有方，在数据不平衡的情况下，保证数据隐私安全的同时，能协同有效的训练模型，仍然是一个重要的研究点。此外，对于系统鲁棒性的提升，即如何处理不稳定的参与者、对外部攻击的防御、恶意参与者的检测与防范仍是该方向的重要课题。

参考文献

[1] HINTON G E, OSINDERO S, TEH Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[2] VOIGT P, VON DEM BUSSCHE A. The eu general data protection regulation (gdpr)[J]. A Practical Guide, 1st Ed., Cham: Springer International Publishing, 2017.

[3] MCMAHAN H B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[J]. arXiv preprint arXiv:1602.05629, 2016.

[4] MCMAHAN H B, RAMAGE D. Federated learning: Collaborative machine learning without centralized training data[Z]. [S.l.: s.n.], 2017.

[5] SHOKRI R, SHMATIKOV V. Privacy-preserving deep learning[C]// Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. [S.l.: s.n.], 2015: 1310-1321.

[6] LIN Y, HAN S, MAO H, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training[J]. arXiv preprint arXiv:1712.01887, 2017.

[7] ERLINGSSON Ú, PIHUR V, KOROLOVA A. Rappor: Randomized aggregatable privacy-preserving ordinal response[C]//Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. [S.l.: s.n.], 2014: 1054-1067.

[8] TEAM D P. Learning with privacy at scale[Z]. [S.l.: s.n.], 2017.

[9] CHEN M, MATHEWS R, OUYANG T, et al. Federated learning of out-of-vocabulary words[J]. arXiv preprint arXiv:1903.10635, 2019.

[10] HARD A, RAO K, MATHEWS R, et al. Federated learning for mobile keyboard prediction[J]. arXiv preprint arXiv:1811.03604, 2018.

[11] RAMASWAMY S, MATHEWS R, RAO K, et al. Federated learning for emoji prediction in a mobile keyboard[J]. arXiv preprint arXiv:1906.04329, 2019.

[12] YANG T, ANDREW G, EICHNER H, et al. Applied federated learning: Improving google keyboard query suggestions[J]. arXiv preprint arXiv:1812.02903, 2018.

[13] APPLE. Private federated learning (neurips 2019 expo talk abstract)[Z]. [S.l.: s.n.], 2019.

[14] LEROY D, COUCKE A, LAVRIL T, et al. Federated learning for keyword spotting[C]//ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). [S.l.]: IEEE, 2019: 6341-6345.

[15] WEBANK. Webank and swiss re signed cooperation mou, 2019[J]. URL: https://finance.yahoo.com/news/webank-swiss-signed-cooperation-mou-112300218.html, 2019.

[16] COURTIOL P, MAUSSION C, MOARII M, et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome[J]. Nature medicine, 2019, 25(10): 1519-1525.

[17] 周志华. 机器学习[M]. China: 清华大学出版社, 2016.

[18] MURPHY K P. Machine learning: a probabilistic perspective[M]. [S.l.]: MIT press, 2012.

[19] HO Q, CIPAR J, CUI H, et al. More effective distributed ml via a stale synchronous parallel parameter server[C]//Advances in neural information processing systems. [S.l.: s.n.], 2013: 1223-1231.

[20] XING E P, HO Q, DAI W, et al. Petuum: A new platform for distributed machine learning on big data[J]. IEEE Transactions on Big Data, 2015, 1(2): 49-67.

[21] WEI J, DAI W, QIAO A, et al. Managed communication and consistency for fast data-parallel iterative analytics[C]//Proceedings of the Sixth ACM Symposium on Cloud Computing. [S.l.: s.n.], 2015: 381-394.

[22] DEAN J, CORRADO G, MONGA R, et al. Large scale distributed deep networks[C]//Advances in neural information processing systems. [S.l.: s.n.], 2012: 1223-1231.

[23] YANG Q, LIU Y, CHEN T, et al. Federated machine learning: Concept and applications[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2019, 10(2): 1-19.

[24] CHEN T, LI M, LI Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512.01274, 2015.

[25] LECUN Y, et al. Lenet-5, convolutional neural networks[J]. URL: http://yann. lecun. com/exdb/lenet, 2015, 20: 5.

[26] HSIEH K, PHANISHAYEE A, MUTLU O, et al. The non-iid data quagmire of decentralized machine learning[J]. arXiv preprint arXiv:1910.00189, 2019.

[27] PENG X, HUANG Z, ZHU Y, et al. Federated adversarial domain adaptation [J]. arXiv preprint arXiv:1911.02054, 2019.

致谢

附录