软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(3):1357−1373 [doi: 10.13328/j.cnki.jos.007466] [CSTR: 32375.14.jos.007466]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于块级多输出和知识自蒸馏的高效联邦学习框架*

刘建春[1,2], 梁文艺[1,2], 徐宏力[1,2], 马千飘[3], 黄刘生[1,2]

[1](中国科学技术大学 计算机科学与技术学院, 安徽 合肥 230026)

[2](中国科学技术大学苏州高等研究院, 江苏 苏州 215004)

[3](南京理工大学 计算机科学与工程学院, 江苏 南京 210094)

通信作者: 徐宏力, E-mail: xuhongil@ustc.edu.cn

摘　要: 联邦学习 (federated learning, FL) 是一种分布式模型训练框架, 允许多个客户端在边缘计算 (edge computing, EC) 环境中协同训练全局模型, 同时保护客户端的本地数据隐私. 然而, 在边缘网络中进行联邦学习训练时, 常常面临资源受限和数据异构 (或称非独立同分布数据) 的问题, 这会导致模型训练性能显著下降. 为了应对这些挑战, 提出了一种高效的联邦学习框架——FedAlt, 以提升边缘网络中模型训练的性能 (如测试精度) 和减少资源开销. FedAlt 在经典联邦学习算法 FedAvg 的基础上引入了块级多输出和知识自蒸馏技术, 使客户端在本地训练时能够更有效地吸收模型表征层信息, 从而缓解非独立同分布数据对模型训练的负面影响. 具体而言, 将模型划分为多个连续的模型块, 服务器在每个全局训练轮次开始时仅向客户端发送前部分的全局模型块, 从而减少通信开销. 然后, 客户端将全局模型和本地模型进行组合, 并利用知识自蒸馏技术吸收模型表征层的信息, 以应对数据异构带来的挑战. 此外, 考虑到通信开销随传输的模型块数量增加而增加, 分别在服务器和客户端设计了自适应算法, 即服务器分发模型块算法和客户端块级多输出正则化算法, 根据客户端的数据分布、计算能力和通信能力来动态地调整服务器分发的模型块数量. 大量实验结果表明, 与现有方法相比, FedAlt 在有限的通信带宽条件下, 可以提升约 2.64% 的平均测试精度.

关键词: 边缘计算; 联邦学习; 数据异构; 资源有限

中图法分类号: TP18

中文引用格式: 刘建春, 梁文艺, 徐宏力, 马千飘, 黄刘生. 基于块级多输出和知识自蒸馏的高效联邦学习框架. 软件学报, 2026, 37(3): 1357–1373. http://www.jos.org.cn/1000-9825/7466.htm

英文引用格式: Liu JC, Liang WY, Xu HL, Ma QP, Huang LS. Efficient Federated Learning Framework with Block-wise Multi-output and Knowledge Self-distillation. Ruan Jian Xue Bao/Journal of Software, 2026, 37(3): 1357–1373 (in Chinese). http://www.jos.org.cn/1000-9825/7466.htm

Efficient Federated Learning Framework with Block-wise Multi-output and Knowledge Self-distillation

LIU Jian-Chun[1,2], LIANG Wen-Yi[1,2], XU Hong-Li[1,2], MA Qian-Piao[3], HUANG Liu-Sheng[1,2]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China)

[2](Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou 215004, China)

[3](School of Computer Science and Engineering, Nanjing University of Science & Technology, Nanjing 210094, China)

Abstract: Federated learning (FL) is a distributed model training framework that allows multiple clients to collaboratively train a global model in an edge computing (EC) environment while preserving the privacy of clients' local data. However, federated learning in edge networks often faces challenges such as resource constraints and data heterogeneity, also known as non-independent and identically

* 基金项目: 国家自然科学基金 (62132019); 江苏省自然科学基金 (BK20230275); 安徽省自然科学基金 (2408085QF185)
收稿时间: 2024-09-23; 修改时间: 2025-03-17; 采用时间: 2025-05-08; jos 在线出版时间: 2025-12-03
CNKI 网络首发时间: 2025-12-04

中国知网 https://www.cnki.net

distributed (non-IID) data, which significantly degrade model training performance. To address these challenges, this study proposes an efficient federated learning framework—FedAlt, aiming to enhance model training performance (e.g., test accuracy) in edge networks while reducing resource consumption. FedAlt builds upon the classic federated learning algorithm FedAvg by incorporating block-wise multi-output and self-knowledge distillation techniques. These enhancements enable clients to more effectively absorb information from the model's representational layers during local training, mitigating the negative impact of non-IID data on model training. Specifically, the model is divided into multiple consecutive blocks, and at the start of each global training round, the server sends only the initial blocks of the global model to the clients, reducing communication overhead. Clients then combine the global model with their local models and use self-knowledge distillation techniques to absorb information from the model's representational layers, addressing data heterogeneity challenges. Moreover, considering that communication overhead increases with the number of transmitted model blocks, adaptive algorithms are designed for both the server and client sides: the model block distribution algorithm and the block-wise multi-output regularization algorithm. These algorithms dynamically adjust the number of distributed model blocks based on the client's data distribution, computational capacity, and communication capabilities. Extensive experimental results show that, compared to existing methods, FedAlt improves average test accuracy by approximately 2.64% under limited communication bandwidth conditions.

Key words: edge computing; federated learning; data heterogeneity; resource limitation

1 引 言

传统云计算模式虽然具有高度的灵活性和可扩展性, 但在处理大规模数据、实时分析以及满足低延迟需求方面存在一定的局限性[1]. 为了解决这些问题, 边缘计算 (edge computing, EC) 作为一种新兴的计算模式, 正逐渐引起广泛关注[2]. 边缘计算是指在靠近物或数据源的系统边缘侧, 集成存储、计算、网络等能力的开放平台[3]. 这种分布式的计算模式可以为用户就近提供大数据计算 (如模型训练) 服务, 从而降低网络延迟和数据传输成本. 然而, 随着边缘计算的普及, 在涉及用户隐私和敏感数据的场景中, 数据隐私和安全性等问题日益凸显. 在此背景下, 联邦学习 (federated learning, FL)[4]作为一种适用于边缘场景的分布式机器学习方法逐渐受到重视. 在联邦学习中, 多个客户端分别使用自身的本地数据进行模型训练, 然后将更新后的本地模型上传到服务器进行全局聚合. 服务器聚合得到全局模型之后将其返回给客户端并开始下一轮的训练. 这种服务器与客户端之间的传输过程一直持续到全局模型收敛或者资源耗尽为止. 由于联邦学习传输模型而不是数据, 因此显著减少了模型训练过程中的通信延迟. 此外, 服务器不会访问客户端的本地数据从而保护了用户隐私.

然而, 在边缘计算环境下实现高效的联邦学习仍然面临诸多挑战[5]. 其中, 数据异构性也称为非独立同分布 (non-independent and identically distributed, non-IID) 数据, 是当前联邦学习中最为显著的挑战[6]. 数据异构指的是参与设备之间的数据分布和特征存在较大差异, 导致在联邦学习过程中出现数据不均衡和特征不匹配等问题. 在这种情况下, 某个参与设备上训练的模型可能无法直接泛化到其他参与设备. 例如, 在医疗领域中, 不同医院的患者数据可能包括不同的疾病种类和严重程度, 以及不同的医疗记录格式, 这种异构性给跨医院的模型训练和共享带来了巨大挑战. 此外, 联邦学习还面临着网络中通信资源有限的问题[7]. 参与设备之间有限的通信带宽限制了模型参数和更新信息的传输效率. 例如, 在一个银行的联邦学习场景中, 各个分行通过联邦学习来改进欺诈检测模型. 然而, 由于分行之间的网络连接速度有限, 传输大量参数和更新信息可能会导致通信拥堵和延迟, 从而影响模型的训练效率和精度.

正则化技术是应对联邦学习中客户端之间数据异构性问题的一种有效方法. 基于 FedAvg[4]、FedProx[8]通过在客户端本地损失函数中增加一个表示全局模型与本地模型距离的正则化项, 使得客户端的本地训练更加稳定. FedMLB[9]将模型划分为多个块, 利用多条本地计算分支构建多个模型块, 并通过知识蒸馏来汲取本地计算分支的知识, 以获取更多的全局信息. 尽管这些正则化方法有效地缓解了数据异构性问题, 但它们增加了模型层次的计算时间, 减缓了模型的收敛速度[10]. 为了减少联邦学习训练过程中通信资源开销, Jiang 等人[11]提出了 FedMP, 通过自适应模型剪枝, 使客户端根据自身状态调整本地模型的大小, 从而减少服务器与客户端之间的通信开销. Kim 等人[12]提出了 DepthFL, 为客户端定制不同大小的模型并通过深度缩放使得服务器和客户端之间只传输部分模型从而减少了通信开销. 然而, 这些方法都使得模型受训练的参数减少, 从而降低全局模型训练的性能.

中国知网 https://www.cnki.net

为了应对联邦学习中数据异构和资源受限问题, 本文提出了一个高效的联邦学习框架——FedAlt. 该框架在FedAvg 基础上引入块级多输出和知识自蒸馏技术, 使得客户端在本地训练时吸收更多的模型表征层信息以缓解non-IID 数据问题. 具体来说, 首先将模型分为多个连续的模型块. 为了减少通信开销, 服务器在每个全局轮次开始时仅向客户端发送前面一部分全局模型块. 然后客户端将全局模型和本地模型进行组合, 并使用知识自蒸馏技术来吸收模型表征层的信息以缓解数据异构带来的负面影响. 由于在知识自蒸馏过程中, 额外的模型表征层输出数量与客户端接收的全局模型块数量有关, 此技术被称为块级多输出. 此外, 由于通信开销随传输的模型块数量增加而增加, 而更多的模型块可以使客户端在本地训练时吸收更多的表征层信息, 从而提高全局模型训练性能. 因此, 本文在服务器和客户端分别设计了相应的算法, 即服务器分发模型块算法 (block-wise model distribution, BMD) 和客户端块级多输出正则化算法 (block-wise multi-output regularization, BMR), 它们通过客户端不同的数据分布、计算和通信能力来确定其传输的模型块数量. 综上所述, 本文的主要贡献如下.

● 提出了一个高效的联邦学习框架——FedAlt. 它通过仅分发部分全局模型来减少通信开销, 并通过引入知识自蒸馏和块级多输出技术来吸收更多模型表征层的信息, 从而缓解了 non-IID 数据问题, 提高模型训练性能 (如测试精度).

● 在服务器和客户端分别设计了高效的算法, 即服务器分发模型块算法 BMD 和客户端块级多输出正则化算法 BMR. 它们根据客户端之间不同的本地数据分布、计算和通信能力来确定服务器分发给客户端最合适的模型块数量.

● 通过大量实验评估和验证了 FedAlt 的训练性能. 实验结果表明, 在通信资源有限的边缘场景下, FedAlt 相对于基线方法在给定通信带宽预算的情况下可以平均提升约 2.64% 的测试精度.

本文第 2 节介绍本文的相关基础知识以及国内外相关工作. 第 3 节系统地介绍 FedAvg 以及 FedAlt 框架的训练流程, 同时通过例子解释如何基于知识自蒸馏和块级多输出进行正则化, 并给出问题的形式化定义. 针对该问题, 在第 4 节中提供两个探索性算法. 第 5 节展示实验结果. 第 6 节对本文进行总结.

2 基础知识和相关工作

2.1 知识蒸馏

在传统的深度学习模型训练中, 通常依赖于大量的数据和复杂的模型[13]. 然而, 随着深度学习技术的不断发展, 出现了许多参数庞大、性能卓越的大型模型, 如 GPT-3[14]等. 这些复杂模型虽然具备出色的性能, 但由于其庞大的参数量和高计算复杂度, 不适用于资源受限的环境, 如移动设备或嵌入式系统. 此外, 这些模型在部署时可能面临推理速度慢、存储需求大等问题. 因此, 如何将大型模型的知识有效地转移到小型模型中, 以实现轻量级部署和高效推理, 成为当前研究的热点之一. 在这一背景下, 知识蒸馏技术[15]应运而生. 知识蒸馏是一种模型压缩技术, 旨在通过将一个复杂的模型 (通常被称为教师模型) 的知识转移给一个简化的模型 (通常被称为学生模型) 来提高学生模型的性能和泛化能力, 其基本思想是利用已经训练好的复杂模型的知识来指导学生模型的训练. 具体而言, 在训练过程中, 将教师模型生成的软目标 (即软标签) 作为额外的监督信号, 同时最小化学生模型的损失函数和学生模型与教师模型输出之间的差异, 从而将教师模型的知识转移给学生模型. 这种方法使得学生模型在保持性能的同时, 显著减少了参数量和计算复杂度, 适用于资源受限的环境. 知识蒸馏技术的优势在于, 它能够在性能损失较小的情况下, 大幅减少模型的规模和计算复杂度, 使得轻量级模型能够在资源有限的设备上运行, 同时保持较高的推理速度和准确性. 知识蒸馏过程类似于人类教育中的知识传承: 教师将自己的知识和经验传授给学生, 期望学生能够在较短时间内掌握核心知识和技能. 在实现知识蒸馏时, 通常包含以下几个步骤.

(1) 训练教师模型. 首先, 使用大量数据对一个庞大的神经网络模型进行训练, 使其具备强大的知识表示能力和深度学习能力.

(2) 知识提取. 在教师模型训练完成后, 通过特定方法提取其中的知识, 这些知识可以包括模型参数、特征表示、类别概率等.

中国知网 https://www.cnki.net

(3) 构建学生模型. 构建一个较小规模的神经网络模型作为学生模型, 该模型通常具有较少的参数和更简单的结构.

(4) 蒸馏训练. 将教师模型的知识注入学生模型中, 通过优化学生模型的参数, 使其能够模仿教师模型的输出. 这个过程可以通过最小化教师模型和学生模型之间的损失函数来实现.

知识蒸馏技术在计算机视觉、自然语言处理、语音识别、医疗影像分析、智能物联网等领域都有着广泛的应用前景, 其通过将复杂模型的知识转移给简化模型实现模型的压缩和性能提升, 为实现在资源受限环境下部署深度学习模型提供了重要的解决方案.

2.2 相关工作

由于边缘设备处于不同的真实环境中, 它们产生的数据分布往往难以与整个边缘网络的数据分布一致, 即这些设备之间的数据分布通常是非独立同分布的 (non-IID), 也被称为数据异构性[6]. 这种非独立同分布的数据会导致联邦学习训练出的全局模型精度下降以及模型收敛速度减慢等. 为了应对这一挑战, 国内外研究人员提出了多种解决方案, 目前主要集中在两个方向: 提升全局模型的泛化性能和发展个性化联邦学习.

在提升全局模型泛化性能方面, Yang 等人[16]提出了一种基于图神经网络的方法, 旨在解决联邦学习中不同参与方数据之间的异构性问题. 他们通过图神经网络实现了跨模态特征提取和表示学习, 从而统一了数据表示, 增强了模型的泛化能力. Sattler 等人[6]设计了一种新的模型更新策略和参数聚合算法, 以有效应对 non-IID 数据带来的挑战, 提升模型性能和收敛速度. Caldarola 等人[17]提出了一种基于窗口聚合模型的方法, 通过窗口化的模型平均技术来提高模型在异构环境下的泛化能力. 该方法能够更有效地利用不同设备上的模型参数, 并通过动态调整窗口大小来适应设备之间的异构性, 从而改善模型的泛化性能. 此外, Zhu 等人[18]提出了一种针对数据异构联邦学习的无数据知识蒸馏方法. 该方法首先在一组与设备上的数据无关的模型上进行训练, 之后利用这个模型来进行设备上的本地模型训练, 而无须访问设备上的实际数据, 从而提高了模型的泛化性能和学习效率.

在个性化联邦学习方面, 每个设备都维护着自己的本地模型, 模型参数根据本地数据进行更新. 个性化模型可以根据不同设备的数据特点进行定制, 从而更好地适应数据的异构性. 对于个性化联邦学习而言, Achituve 等人[19]利用高斯过程为个性化的本地模型建模, 以适应不同设备上的数据分布. 每个边缘设备都维护一个高斯过程模型, 根据本地数据更新其均值和协方差, 然后通过联邦学习来整合所有边缘设备上的高斯过程模型, 得到一个全局的模型. Collins 等人[20]提出了一种新的联邦学习框架和算法, 用于学习跨客户端的共享数据表示和每个客户端的唯一本地头. 所提算法利用跨客户端的分布式计算能力, 对表示的每次更新执行许多关于低维局部参数的局部更新. Zhang 等人[21]利用异构模型设置的潜力, 提出了一种新的训练框架, 为不同的客户使用个性化模型. 具体来说, 他们将原始个性化 FL 中的聚合过程制定为个性化的组知识转移训练算法, 使每个客户端能够在服务器端维护个性化的软预测, 以指导其他客户端的本地训练. Marfoq 等人[22]利用深度神经网络从非表格数据 (如图像和文本) 中提取高质量矢量表示 (嵌入) 的能力, 提出了一种基于局部记忆的个性化机制. 尽管这些方法在一定程度上缓解了 non-IID 数据带来的影响, 但由于在训练过程中客户端和服务器之间传输的是整个模型, 这些方法在资源受限的边缘网络中 (如通信资源受限时),性能仍然会显著下降.

为了减少通信开销, 模型压缩技术常常被应用于联邦学习中. 模型压缩通常包括参数稀疏化、梯度量化、模型压缩算法和联合优化等技术. 参数稀疏化技术通过将模型中的参数稀疏化, 即将一些模型中的参数设置为零或接近零, 从而减少需要传输的参数数量, 这可以通过稀疏矩阵或截断梯度等技术来实现. 对于联邦学习中的稀疏化机制, Stich 等人[23]通过跟踪内存中积累的误差从而在联邦学习中引入误差补偿机制来提高模型压缩的训练性能. 在联邦学习中, 通常需要传输梯度信息以更新全局模型. 梯度量化技术将高精度的梯度信息量化为低精度的形式, 从而减少传输的数据量, 这种方法可以通过减少梯度的位数或使用更少的比特来实现. Basu 等人[24]提出使用量化器 (随机或确定性 1 位符号) 结合稀疏化和误差补偿来实现进一步的模型压缩并减少通信资源. 模型压缩算法通过压缩模型的表示形式来减少模型的大小. 典型的方法包括权重剪枝[25]、模型量化[26]和低秩分解[27]等. 通过这些技术, 可以大幅减少模型参数的数量, 从而降低传输开销. 联合优化方法[28]将模型压缩和联邦学习的优化目标结合起来进行联合优化. 通过在模型训练过程中考虑通信开销, 可以设计出更加高效的模型压缩策略, 从而降低通信

中国知网 https://www.cnki.net

成本. 这些模型压缩技术可以帮助联邦学习系统在边缘场景中更高效地进行模型更新, 从而加速模型收敛并降低通信开销. 然而, 这些技术往往会损害模型精度并产生更多的计算开销以达到目标测试精度. 将知识蒸馏技术应用于联邦学习使得服务器和客户端之间传输的不再是传输模型而是 Logit 输出, 从而在有效减少带宽开销的同时减轻系统异构性带来的模型训练性能下降的影响[29]. 此外, 知识蒸馏通过提供公共的未标记数据集作为代理数据集, 同时利用所有客户端本地模型的综合知识来丰富全局模型, 从而减轻了非独立同分布设置下的负面影响. 然而, 在知识蒸馏过程中客户端需要访问公共的未标记的代理数据集, 这在现实场景中通常很难实现[30]. Zhang 等人[31]提出了无数据知识蒸馏, 通过小型数据生成器生成数据以避免使用公共未标记代理数据集. 然而, 数据生成器往往需要使用客户端的标签信息, 这可能会泄露用户的隐私. 在我们的方法中, 客户端本地使用的自身的模型块产生额外的输出因而没有泄露隐私. 服务器仅向客户端传送部分模型从而减少了通信开销. 此外, 我们提出块级多输出正则化技术并引入知识自蒸馏技术吸收了更多模型表征层信息, 从而有效缓解了数据异构问题.

3 网络建模与问题定义

3.1 联邦学习训练流程

在联邦学习中, 多个客户端可以在中心服务器的协同下训练一个高效的全局模型. 具有 N 个客户端的联邦学习架构的目标是最小化平均损失函数, 如下所示:

$$\min_{x} f(x)=\frac{1}{N}\sum_{n=1}^{N} f_n(x) \tag{1}$$

其中, x 是模型参数, $f_n(x)$ 是客户端 $n\in\{1,\ldots,N\}$ 的本地损失函数. 客户端 n 的本地损失函数被定义为:

$$f_n(x)\triangleq \mathbb{E}_{\varphi_n\sim D_n}[F_n(x;\varphi_n)] \tag{2}$$

其中, φ_n 是客户端 n 的本地数据集 D_n 的一个样本, $F_n(x;\varphi_n)$ 是客户端 n 中对应的该样本的本地损失函数值.

在联邦学习中, 为使得训练出的全局模型可以达到给定目标精度, 客户端与服务器之间需要进行一定轮次的全局训练. 为了减少计算开销和通信开销, 联邦学习通常只选择一部分客户端参与模型训练并且在两个相邻的全局轮次之间使客户端进行多次本地模型更新 (即每个全局轮次包含多次本地迭代). 我们使用 e 来表示两个相邻的全局轮次之间的本地迭代数量. 此外, 我们使用 K ($K\leqslant N$) 表示参与联邦学习中模型训练的客户端数量. 如图 1 所示, 在每一个全局轮次中, 主流的联邦学习框架 (如 FedAvg) 主要被分为以下 3 个阶段[8].

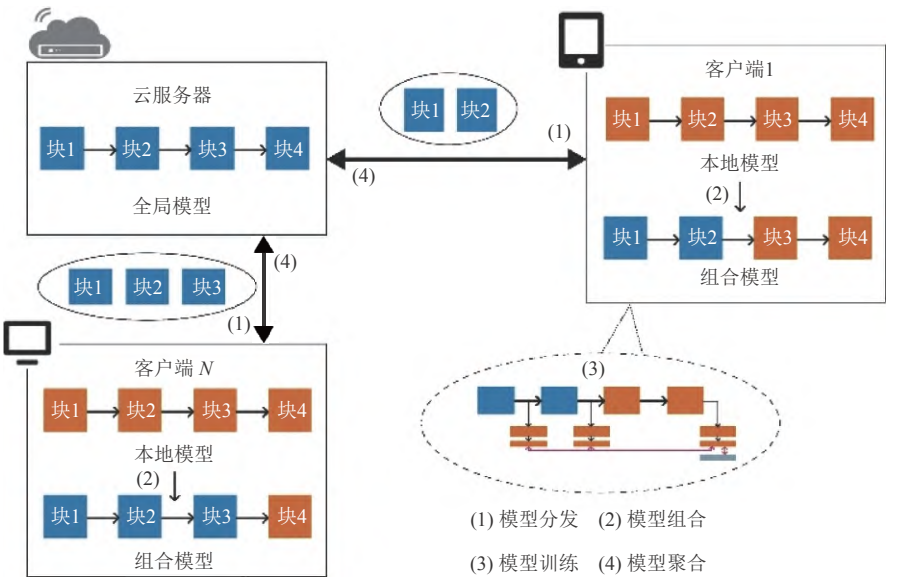


图 1 所提联邦学习框架 FedAlt 的训练流程

中国知网 https://www.cnki.net

(1) 模型分发. 在某一全局轮次 $t \in \{0,1,\ldots,T\}$ 的开始阶段, 服务器会将当前的全局模型发送给 K 个客户端, 其中 T 是全局训练的总轮次数量.

(2) 本地更新. 每个客户端 $k \in \{1,\ldots,K\}$ 会在它的本地数据集 D_k 上使用收到的全局模型执行 e 次本地迭代, 然后客户端 k 会将更新后的本地模型上传给服务器.

(3) 模型聚合. 在服务器收集到所有的 K 个客户端上传的本地模型之后, 服务器按照特定的模型聚合规则来进行全局模型的更新 (比如模型参数平均)[32]. 之后, 联邦学习继续下一个全局轮次的模型训练.

3.2 所提框架训练流程

当客户端之间的本地数据集分布是非独立同分布时, 传统联邦学习的模型训练性能会严重下降[8]. 因此, 我们提出了一个结合知识自蒸馏和块级多输出的框架 (即 FedAlt) 来缓解这一问题. FedAlt 通过使用多个模型块的多条输出与模型输出进行知识自蒸馏, 使客户端在进行本地更新时吸收来自更多前面模型块的表征层信息, 从而缓解了数据的非独立同分布问题. 此外, 我们根据客户端之间不断变化的通信和计算能力使服务器分发不同数量的全局模型块, 从而充分利用网络中的资源.

具体来说, 我们将模型按照深度神经网络中的层次结构划分为 M 个模型块, 如图 1 所示的例子, 全局模型和本地模型都是按照 ResNet-18 网络模型[33], 并根据其自身的残差结构将其按照神经网络的顺序划分为 4 个模型块 (即, $M=4$), 其中每一个模型块对应一个残差块. 由于当前的深度学习模型往往具有层次结构[34], 所以本节这种划分模型块的方法可以很好地应用到大部分深度学习模型中. 需要注意的是, 我们所提的 FedAlt 框架同样适用于 GCN 这种浅层模型 (如, $M=2$). FedAlt 和 FedAvg 的主要不同体现在模型分发和本地更新阶段. 此外, 本节在模型分发和本地更新之间新增了一个模型组合阶段, 用来描述客户端在收到全局模型块后如何覆盖自己的本地模型. 结合图 1, FedAlt 的工作流程具体被分为以下 4 个阶段.

(1) 模型分发. 在某一全局轮次 $t \in \{0,1,\ldots,T\}$ 的开始阶段, 服务器会根据客户端的本地数据分布、计算能力和通信能力发送不同数量的连续的全局模型块. 注意, 这些全局模型块是全局模型的前面一部分模型块. 相比于 FedAvg, FedAlt 只发送了部分模型, 因此显著减少了通信开销. 我们分别使用 x^t 和 m_k^t 来表示第 t 个全局轮次的全局模型和服务器发送给客户端 k 的全局模型块的数量. 为了防止客户端会遗忘全局模型的信息从而使客户端本地出现模型漂移[8], 我们采用了一种新的模型传送的方法, 即服务器每隔 τ 轮就发送一次整个的全局模型. 也就是说, 对于客户端 k 而言, 当 t 是 $\tau+1$ 的倍数时服务器会发送整个的全局模型给客户端 k. 否则服务器会将前面的 m_k^t 个全局模型块分别发送给客户端 k. 为区别服务器是发送 m_k^t 个模型块还是发送整个的全局模型, 我们将 m_k^t 的最大值设置为 $M-1$, 即 $m_k^t \in \{0,\ldots,M-1\}$. 当 $M=0$ 时, 表示服务器发送整个的全局模型.

(2) 模型组合. 如果客户端收到前面 m_k^t 块全局模型, 它会将上一轮保存的本地模型的后面 $M-m_k^t$ 块本地模型块与接收到的全局模型块进行组合, 形成一个组合模型, 之后的本地更新阶段会直接在这个组合模型上进行更新. 如果客户端收到的是整个全局模型, 那么客户端会直接将这个全局模型覆盖掉上一轮保存的本地模型, 即客户端直接将接收到的全局模型用作组合模型, 之后基于这个组合模型进行更新. 我们使用 x_k^t 来表示客户端 k 在全局轮次 t 的本地模型. 此外, 我们分别使用 $x^{t,i}$ 和 $x_k^{t,i}$ 表示第 t 个轮次的全局模型的第 i 个模型块和第 t 个轮次的客户端 k 的本地模型的第 i 个模型块. 例如, 在某个全局轮次 t, 假设客户端 k 接受了来自服务器的两块全局模型块 (即 $x^{t,1}$ 和 $x^{t,2}$), 则客户端 k 会将这两块全局模型块与上一个全局轮次 $t-1$ 保存下来的本地模型的后两个本地模型块 (即 $x_k^{t,3}$ 和 $x_k^{t,4}$) 组合形成组合模型. 之后, 这个组合模型会按照图 2 所示结构并采用知识自蒸馏技术进行本地更新, 具体会在第 3.3 节进行详细介绍. 在图 2 中, 黑色的箭头指示着正向传播的路径, 瓶颈层的作用是使得各个模型块的输出维度一致; q 表示经过正向传播得到的输出, q_m 表示经过所有的模型块得到的输出; y 和 ζ 分别表示数据的真实标签和损失项; 双向箭头表示损失函数中不同的损失项.

(3) 模型训练. 在客户端 k 形成组合模型之后, 客户端 k 将这个组合模型视为新的本地模型 x_k^t. 之后, 客户端会按照图 2 的计算路径执行 e 次本地迭代. 然后, 客户端 k 将训练更新后的本地模型 x_k^t 上传给服务器, 同时, 保存更新后的本地模型 x_k^t, 用来形成下一全局轮次的组合模型. 注意, 客户端进行本地更新时, 本地模型中的全局模型块

中国知网 https://www.cnki.net

的参数也会进行更新, 即客户端对本地模型的所有参数都会进行更新. 此外, 当客户端上传本地模型时, 并不上传瓶颈层的参数.

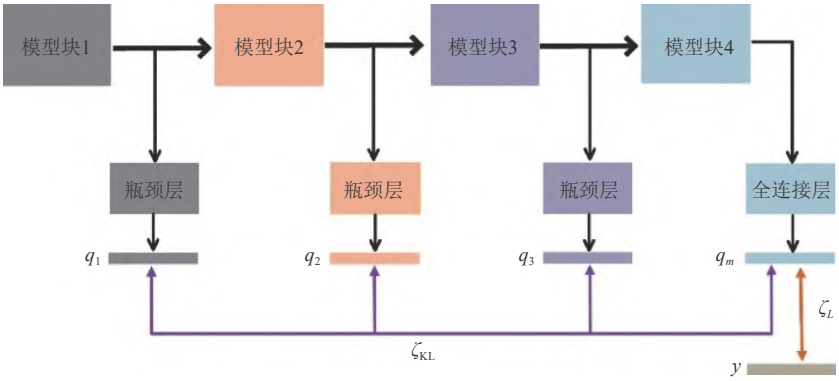


图 2 基于知识自蒸馏和块级多输出的模型训练

(4) 模型聚合. 在服务器收集到所有的 K 个客户端上传的本地模型之后, 服务器按照下面的模型聚合规则来进行全局模型的更新:

$$x^{t+1}=\frac{1}{K}\sum_{k=1}^{K}x_k^{t+1} \tag{3}$$

注意, 客户端并不会上传瓶颈层的参数, 所以瓶颈层不会参与模型聚合. 最后, FedAlt 的模型训练在第 $t+1$ 个全局轮次继续进行模型广播并一直持续到全局模型收敛或者网络资源耗尽为止.

3.3 块级多输出正则化

客户端的本地更新过程会按照图 2 所示的结构进行更新. FedAvg 的本地模型训练只会得到 q_m 这一条本地路径的输出. 而在 FedAlt 中, 客户端会根据服务器决策出的全局模型块的数量来决定产生多少条输出, 客户端每接收一个全局模型块就会多一条输出, 即对于客户端 k 而言共产生 m_k+1 条输出. 产生输出的路径如图 2 所示, 我们在每个额外的输出之前加上一个瓶颈层来统一输出的维度. 有了这多条额外输出, 客户端使用知识自蒸馏技术起到了正则化的效果, 因此客户端可以吸取更多来自前面模型块的表征层信息, 从而使得本地模型向全局模型方向靠拢. 值得注意的是, 当服务器发送给客户端的是整个的全局模型时, 客户端的本地更新过程是按照接收到 $M-1$ 块全局模型块进行的, 即客户端直接按照图 2 所示结构进行本地更新.

为了更好地说明基于知识自蒸馏和块级多输出的正则化技术, 我们结合图 2 给出了一个具体的例子. 假设我们将一个模型分成了 4 个连续的模型块, 即 $M=4$. 在图 2 中, 我们将模型块 4 和全连接层分开画出, 而在后面的描述中, 服务器分发最后一块模型块时是指服务器将会把最后一块模型块以及全连接层一起发送. 此外, 瓶颈层并不参与服务器与客户端之间的模型传输. 例如, 当服务器发送给某个客户端 k 模型块 4 时, 它会将模型块 4 与其后面的全连接层同时发送给客户端 k. 而当服务器发送给某个客户端 k 模型块 3 时, 它并不会将模型块 3 后面的瓶颈层发送给客户端 k. 在某一个全局轮次 t 中, 服务器将前 2 块全局模型块 $x^{t,1}$ 和 $x^{t,2}$ 发送给某个客户端 k, 即 $m_k^t=2$. 结合图 2, 模型块 1 与模型块 2 分别对应 $x^{t,1}$ 和 $x^{t,2}$, 而模型块 3 和模型块 4 分别对应客户端 k 在上一个全局轮次 $t-1$ 的本地模型的后两块, 即 $x_k^{t-1,3}$ 和 $x_k^{t-1,4}$. 之后客户端 k 基于这个组合起来的模型进行本地更新. 如图 2 所示, 客户端 k 的输入经过本地正向传播会得到 3 条输出, 即 $\{q_1,q_2,q_m\}$. q_m 是经过了所有的模型块得到的本地输出, 而 q_i 是经过 i 个模型块和一个最后的瓶颈层得到的输出. 例如, q_2 是由输入经过模型块 1、模型块 2 和一个瓶颈层得到的输出. 得到 3 条输出之后, 客户端 k 会将这 3 条输出放入到损失函数中. 其中, 本地输出和真实标签之间会采用标准的交叉熵损失, 而块级多输出正则化中额外增加的输出与本地输出之间将使用 KL 散度来进行额外知识的提取. 损失函数将在第 4 节中具体说明. 注意, 为了清楚地表示所有情况, 图 2 展示的是服务器分发给客户端 k 整个全局模型的场景. 而在图 1 所示的例子中, 由于此时客户端 1 只接收了两个全局模型块, 因此并不存在 q_3 这条输

中国知网 https://www.cnki.net

出, 同时也不存在产生 q_3 这条输出的计算路径.

本文的主要目的是在提高全局模型的泛化性能的同时减少通信开销, 框架所采用的块级多输出正则化可以实现这一目的. 利用多条额外的输出可以使客户端本地模型吸收更多来自前面模型表征层的信息, 从而让本地训练更加稳定并且向全局模型靠拢, 减轻了客户端之间本地非独立同分布数据带来的负面影响. 此外, 服务器采取分发部分模型块的方式减少了通信开销. 服务器在决策发送客户端全局模型块的过程中充分考虑到客户端的本地数据分布情况、计算能力以及通信能力. 直观上来看, 当客户端的本地数据分布偏离全局数据分布时, 客户端需要更多的全局模型块来吸收更多的表征层信息. 而当客户端接收更多的全局模型块时, 则会消耗更多的通信资源. 因此, 如何在数据异构场景下决策出客户端在每一个全局轮次需要接收的全局模型块的数量, 从而保证全局模型性能是一个关键的问题.

3.4 问题定义

在本节中, 我们给出在联邦学习框架下为每个客户端发送最适合模型块问题的形式化定义. 假设在网络中一共有 N 个客户端, 其中有 K 个客户端参与训练. 我们主要考虑网络中的计算资源和通信资源, 使用 C 和 B 来表示计算资源和通信资源的总预算, 即训练过程中消耗的计算资源和通信资源不能超过 C 和 B. 此外, 我们使用 c 和 b 来分别表示使用一个批次的数据对模型的梯度更新所需的计算资源和传递完整模型所需要的通信资源. 为了方便进行表示, 我们假设模型被均分为 M 个块且每个模型块的参数数量都是相同的, 则每传递一个模型块产生 b/M 的通信开销. 由于全连接层参数数量相对于整个模型而言很少, 因此全连接层带来的计算开销和通信开销可以忽略不计. 此外, 我们假设输入通过瓶颈层需要的计算开销为 c_b, 以及客户端 k 本地有 n_k 个批次的数据, 并且在某一个全局轮次 t 中接收到 m_k^t 个模型块, 则客户端 k 在第 t 个全局轮次产生的计算开销为:

$$c_k^t = (m_k^t \cdot c_b + c) \cdot n_k \cdot e \tag{4}$$

其中, e 表示连续两个全局轮次之间的本地迭代数量. 公式 (4) 中 $m_k^t \cdot c_b$ 表示客户端 k 每接收一个模型块则会多产生 c_b 的计算开销. 服务器每隔 τ 轮会传送整个的全局模型, 此时客户端 k 的计算开销为:

$$c_k^t = [(M-1) \cdot c_b + c] \cdot n_k \cdot e \tag{5}$$

结合公式 (4) 和公式 (5), 可以得到客户端 k 的每个全局轮次的平均计算开销为:

$$c_k^t = \left(\frac{M - 1 + \tau \cdot m_k^t}{\tau + 1} \cdot c_b + c \right) \cdot n_k \cdot e \tag{6}$$

其中, $m_k^t \in \{0, \ldots, M-1\}$. 注意, 当服务器传送整个模型时, $m_k^t = 0$. 对于通信开销, 假设在某一轮全局通信轮次 t 中, 客户端 k 下载和上传 m_k^t 个模型块, 则客户端 k 在第 t 个全局轮次产生的通信开销为:

$$b_k^t = \left(1 + \frac{m_k^t}{M}\right) \cdot b \tag{7}$$

其中, 1 表示客户端上传整个全局模型. 同理, 服务器每隔 τ 轮传送完整的全局模型, 此时客户端 k 的通信开销为 $2 \cdot b$. 综上所述, 可以得到客户端 k 的每个全局轮次的平均通信开销为:

$$b_k^t = \left(\frac{\tau}{\tau + 1} \cdot \left(1 + \frac{m_k^t}{M}\right) + \frac{2}{\tau + 1} \right) \cdot b \tag{8}$$

本文的优化目标是最小化全局损失函数, 同时为每一个客户端找到最优的模型块数量, 从而定制最适合的本地模型. 因此, 该问题可以被定义成如下形式:

$$\min_{T \in \{1,2,3,\ldots\}} f\left(x^T\right) \tag{9}$$

$$\text{s.t.} \begin{cases} \sum_{t=1}^{T} \sum_{k=1}^{K} c_k^t \leqslant C \\ \sum_{t=1}^{T} \sum_{k=1}^{K} b_k^t \leqslant B \\ m_k^t \in \{0, \ldots, M-1\}, \ \forall k, t \end{cases} \tag{10}$$

其中, 当 $m_k^t = 0$ 时表示服务器传送给客户端 k 整个的全局模型. 第 1 组不等式保证了网络中的计算资源约束, 第 2

中国知网 https://www.cnki.net

组不等式保证了网络中的通信资源约束, 第 3 组等式表示优化的变量 m 应该是一个整数.

由于上述问题中优化变量 m 是一个整数, 因此这是一个典型的整数规划问题. 而由于整数规划问题是 NP 难的问题[35], 所以直接解决该问题是一项很困难的任务. 为了解决这一问题, 本文使用能够反映客户端实时变化状态的反馈变量, 并基于贪心策略来使服务器为每个客户端决策出最优的模型块数量. 算法的详细内容将在第 4 节展开说明.

4 算法设计

本节中, 我们在服务器和客户端分别提出了一个算法来解决问题 (9). 首先基于客户端的本地模型更新结构为客户端定制了一个本地损失函数, 之后设计了一个反馈变量来表示客户端的状态信息, 最后计算出决策变量用于确定客户端需要接收的模型块的数量.

4.1 准备工作

(1) 本地损失函数. 由于输入数据经过本文设计的块级多输出正则化结构之后会有多条输出, 且本节在多条输出之间应用了知识自蒸馏技术来汲取额外的模型表征层信息, 所以本节将本地损失函数分为了两部分, 一部分是本地损失项而另一部分是知识蒸馏项. 其中, 本节使用交叉熵损失函数来表示本地损失项, 该函数是深度学习领域中一种最常见的损失函数[36]. 具体的, 我们使用 $CrossEntropy(a,b)$ 来表示两个相同维度变量 a 和 b 的交叉熵损失, 则本地损失函数中的损失项可以表示为:

$$L_{\text{ce}} = CrossEntropy(q_m, y) \tag{11}$$

其中, q_m 表示数据经过完整的本地路径得到的输出, y 表示数据的真实标签. 此外, 我们使用 Kullback-Leibler (KL) 散度[37]来表示本地损失函数中的知识蒸馏项. $KL(a,b)$ 表示两个相同维度向量 a 和 b 之间的 KL 散度, 则本地函数的知识蒸馏项可以表示如下:

$$L_{\text{KL}} = \frac{1}{m-1}\sum_{j=1}^{m-1} KL(\hat{q}_j, \hat{q}_m) \tag{12}$$

其中, $\hat{q}$ 表示知识蒸馏过程中使用了超参数温度 δ 之后得到的输出, 而温度 δ 在公式 (12) 中的作用是影响 Softmax 函数输出的平滑性. 将本地损失项和知识蒸馏项相结合, 我们给出了客户端 k 如下的本地损失函数:

$$L_k = L_{\text{ce}} + \alpha \cdot L_{\text{KL}} \tag{13}$$

其中, α ($\alpha \geqslant 0$) 表示客户端 k 的本地损失函数中知识蒸馏项的权重. 客户端 k 根据公式 (13) 中的本地损失函数进行本地模型更新.

(2) 反馈变量. 解决问题 (9) 的关键是在每一轮全局训练开始时, 为每个客户端确定要接收的最适合的全局模型块的数量, 同时提高全局模型的性能. 根据问题 (9) 的约束条件, 客户端 k 需要接收的全局模型块的数量 m_k 应与客户端 k 的通信能力、计算能力以及本地数据分布情况有关. 对于数据分布, 我们采用本地模型和聚合后的全局模型之间的差异来衡量客户端本地数据分布和全局数据分布的差异. 根据之前的研究表明, 本地数据分布和全局数据分布之间的差异与本地模型和全局模型之间的差异成正比[38]. 我们使用 θ_k 来表示客户端 k 的本地模型和聚合后的全局模型之间的差异, 则 θ_k 可以表示为 $\|x^t - x_k^t\|^2$. 为了便于客户端之间相互衡量, 需要进行归一化, 因此 θ_k 最终被表示如下:

$$\theta_k = \frac{\|x^t - x_k^t\|^2}{\sum_{k'=1}^{K} \|x^t - x_{k'}^t\|^2} \tag{14}$$

直观上来看, 当客户端 k 的本地数据分布与全局数据分布差异较大时, 客户端需要接收更多的全局模型块通过知识自蒸馏技术吸收更多的模型表征层信息, 因此 m_k 应与 θ_k 成正比. 对于通信能力和计算能力, 我们分别使用通信时间和计算时间来量化它们. 我们使用 $H_{k,b}$ 和 $H_{k,c}$ 来分别表示客户端 k 在某一个全局轮次的通信时间和计算时间. 为了便于比较, 我们使用了如下归一化的形式来表示客户端 k 的通信时间和计算时间:

$$H'_{k,b} = \frac{H_{k,b}}{\sum_{k'=1}^{K} H_{k',b}},\ H'_{k,c} = \frac{H_{k,c}}{\sum_{k'=1}^{K} H_{k',c}} \tag{15}$$

中国知网 https://www.cnki.net

当客户端 k 收到更多的全局模型块时, 客户端需要消耗更多的通信时间来进行通信, 同时在本地更新时需要更多的计算时间进行计算. 因而客户端 k 接收的全局模型的数量 m_k 应与客户端 k 的计算时间和通信时间成反比. 即客户端的通信能力和计算能力越强, 客户端的通信时间和计算时间越短, 需要接收的全局模型块就越多.

基于上述分析, 我们设计了一个反馈变量 $Z_{k,m}$. 它表示当客户端 k 收到 m 个本地模型块之后回馈给服务器相关信息, 并综合这些信息算出来的能反映客户端状态的一个变量. 然后, 服务器根据这个算出来的反馈变量进行决策. 我们将 $Z_{k,m}$ 定义如下:

$$Z_{k,m}^{t}=\frac{\theta_k\cdot\Delta L_{\mathrm{ce}}}{\mathrm{e}^{H'_{k,c}+\beta\cdot H'_{k,b}}} \tag{16}$$

其中, 使用指数函数是为了增加通信时间和计算时间对反馈变量的影响程度, β ($\beta\leqslant 0$) 表示客户端的计算能力对反馈变量的影响, L_{ce} 是客户端本地损失函数中的本地损失项, $\Delta\zeta_L$ 象征着客户端 k 的本地模型的精度提升. 使用超参数 β 的原因是当瓶颈层参数数量相对模型的总参数数量很小时, 各个客户端之间的计算时间与接收的模型块数量相关性较小, 因而算法可以将 β 设置成较小的值. ΔL_{ce} 的值为上一个全局轮次的 L_{ce} 减去本轮全局轮次 L_{ce} 的值. 服务器会根据每个客户端的反馈信息来计算出反馈变量 $Z_{k,m}$ 的值, 并根据反馈变量做出下一个全局轮次的决策.

4.2 算法详细描述

为了决策出每一个客户端应当接收的模型块的数量, 我们设计了如下算法. 服务器需要存储一个变量 R_k 来存储客户端 k 之前全局轮次的反馈信息, 我们将 R_k 定义如下:

$$R_k=\{R_{k,0},R_{k,1},\ldots,R_{k,M-1}\} \tag{17}$$

其中, $R_{k,m}$ 对应着给客户端 k 发送 m 个全局模型块后收到的反馈信息, 而 $R_{k,0}$ 对应的则是服务器给客户端 k 发送整个全局模型块后收到的反馈信息. 我们需要记住先前轮次的反馈信息从而避免某一轮次带来的错误的反馈信息的干扰, 因此 $R_{k,m}$ 应该按照如下规则进行更新:

$$R_{k,m}=\eta Z_{k,m}+(1-\eta)R_{k,m} \tag{18}$$

其中, η ($0\leqslant\eta\leqslant 1$) 代表实时反馈信息的权重. 由于全局模型块的数量在算法开始时是被随机选择的, 并且客户端的状态在训练过程中会发生变化, 因而我们不能总是选择对应 $R_{k,m}$ 最大的 m 作为应发送给客户端 k 的全局模型块的数量. 因此, 我们为反馈信息添加了一个惩罚项, 最终我们使用决策变量 $D_{k,m}$ 来决策发送给客户端 k 的全局模型块的数量, $D_{k,m}$ 定义如下:

$$D_{k,m}=R_{k,m}+\frac{\sqrt{\ln(t+1)}}{F_{k,m}+1} \tag{19}$$

其中, $F_{k,m}$ 表示客户端 k 接收 m 个全局模型块的全局轮次的频次. 我们选择 $D_{k,m}$ 对应的最大的 m 作为最终的客户端 k 接收的全局模型块的数量, 即 $m=\underset{m}{\operatorname{argmax}}D_{k,m}$. 在公式 (19) 中, 第 2 项的分母项表示对于客户端 k 而言经常被服务器选择的 m 的优先级减弱, 分子项则表示随着训练的进行, 公式 (19) 的第 2 项对决策变量 D 的影响将会变弱. 具体的算法如算法 1 和算法 2 所示. 注意, 当全局轮次 t 为 $\tau+1$ 的倍数时, 服务器直接使 $m_k^t=0$ 并传送整个的全局模型, 而客户端则是直接将接收的全局模型用作组合模型.

算法 1. 服务器分发模型块算法 (BMD).

输入: 参与训练的客户端数量 K, 超参数 β、η, 间隔 τ;
输出: 全局模型 x^T.

1. 初始化 x^0, x_k^0, $R_k=0$, $F_{k,m}=0$, $m_k^1=M/2$;
2. **for** 对于每个全局轮次 $t=\{1,2,\ldots,T\}$ **do**
3. **for** 对于每个客户端 $k\in\{1,2,\ldots,K\}$, 服务器 **do**
4. 根据公式 (15)、(16)、(17) 和 (18) 更新 $R_{k,m}$;

中国知网 https://www.cnki.net

5. **for** 对于每个 $R_{k,m} \in R_m$ **do**
6. 根据公式 (19) 计算 $D_{k,m}$;
7. **end**
8. 为客户端 k 做出本轮决策: $m_k^t = \underset{m}{\operatorname{argmax}} V_{k,m}$;
9. 更新客户端 k 接收 m 个全局模型块的频次: $F_{k,m} = F_{k,m} + 1$;
10. 客户端 k 执行算法 2;
11. **end**
12. 服务器收到客户端本地模型后通过公式 (3) 聚合模型以获得 x^t;
13. **End**

算法 2. 客户端块级多输出正则化算法 (BMR).

输入: 全局模型块数量 m_k^t, 本地批次大小 B, 本地迭代次数 e, 温度 δ, 学习率 γ;
输出: 客户端 k 的本地模型 x_k^t, 通信时间 $H_{k,b}$, 计算时间 $H_{k,c}$.

1. 将 m_k^t 块全局模型块和 $M - m_k^t$ 本地模型块进行结合形成组合模型;
2. **for** 对于每个本地迭代 $E = \{1,2,\ldots,e\}$ **do**
3. 按照图 2 所示得到多条输出;
4. 根据公式 (13) 进行本地训练迭代;
5. **end**
6. 记录通信时间 $H_{k,b}$ 和计算时间 $H_{k,c}$;
7. 将 $H_{k,b}$、$H_{k,c}$ 和 x_k^t 返回给服务器;

在算法 1 中, 我们首先对参数进行初始化 (第 1 行). 接着, 对于每个客户端 k 通过相关公式求解得到 $D_{k,m}$ (第 3–7 行). 我们为客户端 k 做出本轮决策得到 m_k^t, 并更新客户端上接收 m 个全局模型块的频次 (第 8、9 行). 随后, 每个客户端分别执行算法 2 (第 10 行). 最后, 服务器会对接收到的模型进行聚合获得全局模型 (第 12 行). 在算法 2 中, 客户端首先对本地模型和全局模型进行组合 (第 1 行), 随后对组合模型进行多次训练迭代 (第 2–5 行). 在此过程中, 我们记录下通信时间和计算时间, 并返回给服务器进行接下来的运算 (第 6、7 行).

此外, 我们也对算法的时间复杂度进行了分析. 具体而言, 在每一轮训练 $t \in \{1,\ldots,T\}$ 中, 每个客户端 $k \in \{1,\ldots,K\}$ 首先会计算收到 m (最大为 M) 个全局模型块的反馈信息, 接着客户端进行 e 次本地迭代执行块级多输出算法, 需要消耗 O(M+e) 的时间复杂度, 综上所述, 整个算法的时间复杂度为 O(TK(M+e)).

5 实验分析

在本节中, 我们分别做了 5 组实验来验证所提框架 FedAlt 的有效性和高效性. 首先介绍实验配置, 然后展示实验的结果.

5.1 实验设置

● 实验环境. 我们构建了一个联邦学习的仿真环境, 由一个 AMAX 深度学习工作站 (Intel(R) Xeon(R) Gold 5218R CPU, 8 NVIDIA GeForce RTX 3090 GPUs, 256 GB RAM) 组成, 并且在 PyTorch 框架下[39]完成了所有的实验, Python 库的 MPI 被用来建立客户端和参数服务器之间的通信. 参考文献 [40] 中的实验设置, 我们生成了 100 个客户端并且随机激活了其中的 10 个客户端, 通过这种设置我们可以模拟 FedAlt 和所有基线方法的联邦训练过程.

● 数据集和模型. 我们在 CIFAR-10 和 CIFAR-100 数据集[41]上分别评估了 FedAlt 和所有基线方法的性能.

中国知网 https://www.cnki.net

CIFAR-10 和 CIFAR-100 是在当前联邦学习研究方向下最常使用的评估算法性能的数据集. CIFAR-10 和 CIFAR-100 都是包含了 60 000 张彩色 RGB 图片的数据集, 图片大小为 32×32, 其中, 50 000 张图片用作训练集, 10 000 张图片用作测试集. 在实验期间, 我们把所有的数据平均划分到每个客户端, 因此每个客户端有 5 000 张训练图片. 此外, 我们在 CIFAR-10 和 CIFAR-100 数据集上使用 ResNet-18 模型进行训练. ResNet-18 是一个常见的深度学习模型, 常常用于图片识别任务, 模型的大小约为 42 MB. 由于 ResNet 模型所特有的残差块的存在, 我们可以很容易地将模型进行分块来实现 FedAlt 的分块机制. 在实验中, 我们将 ResNet-18 模型按照残差块的层次分成了 4 个模型块. 注意, 由于现在的深度学习模型通常都具有层次结构, 所以这种模型分块的方式很容易扩展到其他的深度学习模型之中.

● 数据划分. 客户端本地不同的数据分布 (即独立同分布和非独立同分布) 对模型的训练性能有很大的影响. 当客户端本地数据分布都是独立同分布时, 训练出的模型性能会很好; 当客户端本地的数据分布是非独立同分布时, 训练出的全局模型性能往往会很差[8]. 而现实情况中客户端的数据往往是非独立同分布的, 因此我们主要探究非独立同分布数据对训练出的全局模型性能的影响. 我们使用 ε 来表示客户端本地数据分布的非独立同分布程度, 其中 ε 的范围为 $\{0,\ldots,9\}$. 当 $\varepsilon=0$ 时, 表示客户端之间本地的数据分布都是独立同分布的, 这是一种理想的情况. 当 $\varepsilon\in\{1,\ldots,9\}$ 时, 对于 CIFAR-10 数据集, 它表示客户端本地数据中有 ε×10% 的数据属于同一个类, 而剩余的数据被均匀分布到剩下 9 个类中; 对于 CIFAR-100 数据集, 它表示客户端本地数据中缺少 ε×10% 类的图片, 而这些数据被均匀分布到剩下 $100-\varepsilon$×10% 个类中. 在第 5.2 节中, 我们主要使用 $\varepsilon=7$ 进行实验.

● 基线方法和度量指标. 我们将 3 种基线方法与 FedAlt 进行比较, 这 3 种基线方法分别是 FedAvg[4]、FedProx[8]、MOON[42]和 BOSE[43]. FedAvg 是第 1 个提出联邦学习的方法, 它使用了最简单模型平均进行模型聚合. FedProx 在 FedAvg 的基础上在客户端本地损失函数增加了一个正则化项, 这个正则化项描述了客户端本地模型和全局模型的距离, 从而使得客户端本地训练更加的稳定. MOON 在 FedAvg 的基础上增加了模型层次的对抗学习, 利用模型表征之间的相似性来纠正客户端的本地训练. BOSE 通过添加额外的分类器将一个模型细分为多个模型块. BOSE 在训练过程中会评估每个模型块的收敛状态, 并基于此为异构节点分配不同的模型块进行训练. 在本文中, 我们使用以下 3 个衡量指标来评估我们提出的方法的性能. (1) 测试精度: 在每个全局轮次, 我们都统计训练出的全局模型在测试数据集上的精度. (2) 时间开销: 当训练出的全局模型达到给定精度时, 我们会记录训练的完成时间 (包含计算时间和通信时间). 在每个全局轮次, 系统的完成时间等于参与训练的客户端的最长的完成时间. (3) 通信开销: 在模型训练期间, 我们记录服务器用于接收模型和分发模型所带来的总的通信开销.

5.2 实验结果

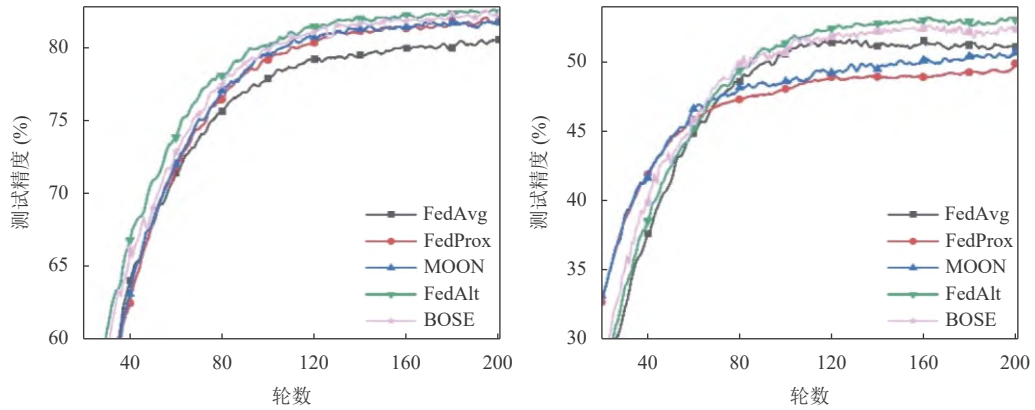
我们做了 5 组实验来证明我们提出的方法的有效性, 仿真实验的结果如下所示.

● 收敛性能. 我们首先展示不同方法训练出的全局模型在 CIFAR-10 和 CIFAR-100 数据集上的测试精度. 注意, 这组实验是在非独立同分布水平 $\varepsilon=7$ 下进行的. 从图 3 和图 4 中可以看到, 在 $\varepsilon=7$ 的条件下无论是在 CIFAR-10 还是 CIFAR-100 数据集上, 使用 FedAlt 方法全局训练 200 轮后得到的全局模型具有最高的测试精度和最低的损失值. 例如, 在 CIFAR-10 数据集上进行测试, FedAlt 训练出的全局模型的测试精度为 82.48%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的测试精度分别为 80.56%、81.74%、81.73% 和 82.24%; 在 CIFAR-100 数据集上进行测试, FedAlt 训练出的全局模型的测试精度为 53.07%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的测试精度分别为 51.07%、49.89%、50.69% 和 52.38%, 这平均提高了 2.07% 的精度. 从这些实验结果可以看出, 即使在不考虑资源损耗的条件下, 我们提出的方法 FedAlt 仍然具有最好的收敛性能.

● 时间损耗. 我们在具有时间约束的条件下进行实验来测试 FedAlt 和其他 3 种基线方法的性能. 由图 5 可知, 无论在 CIFAR-10 还是 CIFAR-100 上, 当给定一定时间训练之后, FedAlt 相比于其他基线方法都拥有最好的性能. 例如, 对于 CIFAR-10 数据集, 当给定训练时间不超过 1 400 s 时, FedAlt 训练出的全局模型的最高测试精度为 81.87%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的最高测试精度分别为 80.48%、81.40%、77.20% 和 81.56%; 对于 CIFAR-100 数据集, 当给定训练时间不超过 1 400 s 时, FedAlt 训练出的全局模型的最高

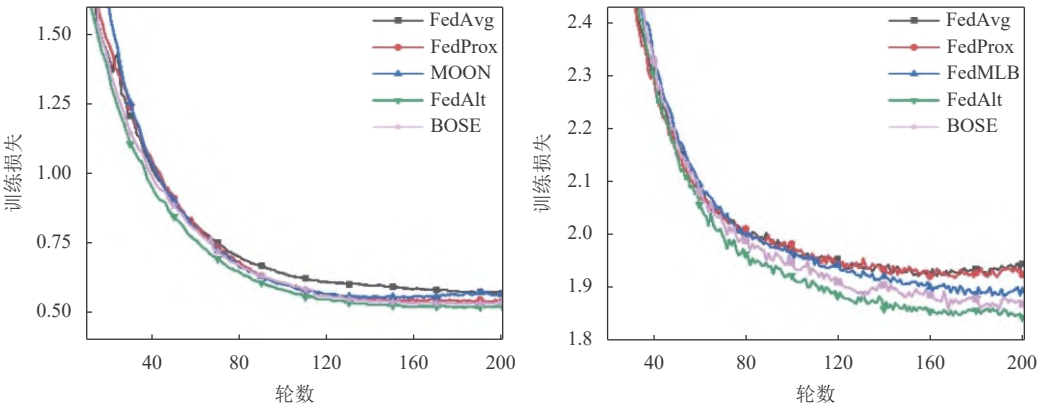
中国知网 https://www.cnki.net

测试精度为 52.63%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的最高测试精度分别为 50.96%、48.94%、48.18% 和 51.42%. 因此, 我们提出的 FedAlt 相比于其他 4 种基线方法, 由于块级多输出正则化和知识自蒸馏技术的引入加速了全局模型的收敛速度.



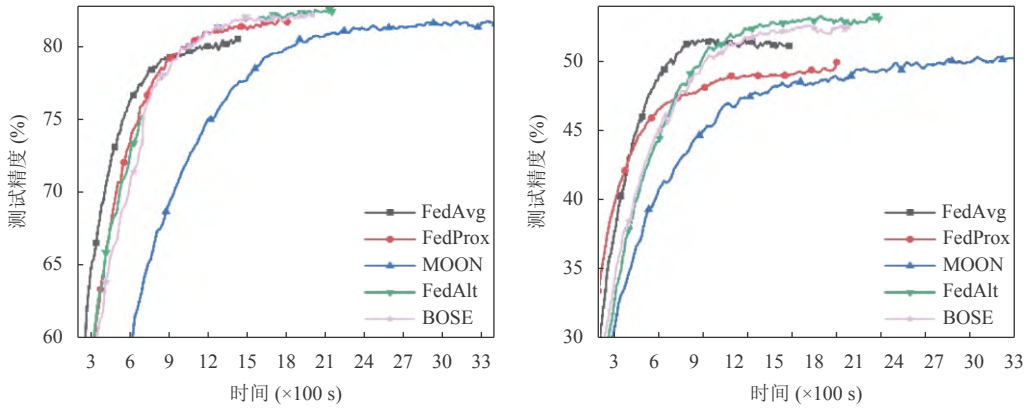
(a) 不同方法在 CIFAR-10 数据集上的测试精度 (b) 不同方法在 CIFAR-100 数据集上的测试精度

图 3 在 $\varepsilon = 7$ 下, 不同方法在 CIFAR-10 和 CIFAR-100 数据集上的测试精度



(a) 不同方法在 CIFAR-10 数据集上的损失 (b) 不同方法在 CIFAR-100 数据集上的损失

图 4 在 $\varepsilon = 7$ 下, 不同方法在 CIFAR-10 和 CIFAR-100 数据集上的训练损失

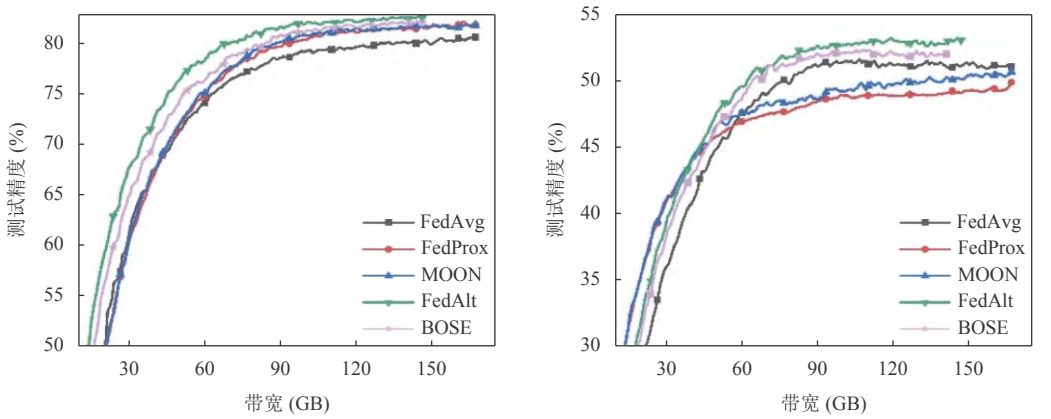


(a) 在 CIFAR-10 数据集上具有时间约束的测试精度 (b) 在 CIFAR-100 数据集上具有时间约束的测试精度

图 5 不同方法在 CIFAR-10 和 CIFAR-100 上随完成时间约束变化的测试精度

中国知网 https://www.cnki.net

● 带宽损耗. 我们在具有带宽约束的条件下进行实验来测试 FedAlt 和其他 3 种基线方法的性能. 由图 6 可知, 无论在 CIFAR-10 还是 CIFAR-100 上, 当给定一定带宽损耗进行联邦学习训练之后, FedAlt 相比于其他基线方法都拥有最好的性能. 例如, 对于 CIFAR-10 数据集, 当给定带宽预算不超过 90 GB 时, FedAlt 训练出的全局模型的最高测试精度为 81.52%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的最高测试精度分别为 78.52%、79.60%、80.09% 和 80.76%; 对于 CIFAR-100 数据集, 当给定带宽预算不超过 90 GB 时, FedAlt 训练出的全局模型的最高测试精度为 52.67%, 而 FedAvg、FedProx、MOON 和 BOSE 训练出的全局模型的最高测试精度分别为 51.26%、48.45%、48.55% 和 51.89%, 我们的方法相比于基线方法平均提高了约 2.64% 的精度. 这是因为大部分基线方法在服务器和客户端之间传输的始终是完整的模型, 而 FedAlt 在全局轮次开始时服务器只向客户端发送部分模型块, 所以 FedAlt 相对于这些基线方法大大减少了带宽的损耗.



(a) 在 CIFAR-10 数据集上具有带宽约束的测试精度　　(b) 在 CIFAR-100 数据集上具有带宽约束的测试精度

图 6　不同方法在 CIFAR-10 和 CIFAR-100 上具有带宽约束的测试精度

● 不同 non-IID 程度的影响. 在不同的 non-IID 程度下, 我们统计了不同方法在 CIFAR-100 数据集上经过固定轮次 (例如 200 轮) 的全局训练得到的模型的测试精度. 根据表 1, 在 IID 设置下, FedAlt 训练出的全局模型的测试精度与其他 4 种方法几乎相同. 然而, 通过使用块级多输出正则化和知识自蒸馏技术, FedAlt 在 non-IID 设置下可以实现最高的测试精度. FedAlt 相对于 4 个基线方法的测试精度提升会随着 non-IID 程度的增加而增加. 例如, 在极端 non-IID 设置 (即 $\varepsilon = 9$) 下, CIFAR100 上 FedAlt 的测试精度为 40.4%, 而 FedAvg、FedProx、MOON 和 BOSE 的测试精度分别为 37.6%、37.9%、38.6% 和 39.25%. 因此, FedAlt 有效缓解了由 non-IID 问题导致的模型训练性能下降问题.

表 1　不同 non-IID 程度 (ε) 下不同方法训练得到的全局模型的测试精度

ε	FedAvg	FedProx	MOON	BOSE	FedAlt
0	56.32	**56.46**	56.26	56.29	56.31
5	52.40	52.62	54.54	54.66	**55.37**
7	51.07	49.89	50.65	51.74	**53.07**
9	37.61	37.96	38.62	39.25	**40.41**

● 温度超参数的影响. 客户端本地损失函数中知识蒸馏项中的温度超参数 δ 对训练性能也会产生影响. 在实验中, 我们通过使用不同的温度来测试 FedAlt 的性能. 表 2 展示了不同的温度值在两种数据集 CIFAR-10 和 CIFAR-100 上对 FedAlt 训练出的全局模型的性能影响. 可以看到, 在 CIFAR-10 数据集上进行训练时, $\delta = 2.5$ 时训练出的全局模型拥有最好的性能; 在 CIFAR-100 数据集上进行训练时, $\delta = 1.0$ 时训练出的全局模型拥有最好的性能. 注意, 我们在其余实验过程中统一使用 $\delta = 1.0$ 与其余基线方法进行对比.

中国知网 https://www.cnki.net

表 2　不同温度超参数 δ 在 CIFAR-10 和 CIFAR-100 数据集上对 FedAlt 的测试精度的影响 (%)

数据集	$\delta=0.5$	$\delta=1.0$	$\delta=1.5$	$\delta=2.0$	$\delta=2.5$	$\delta=3.0$
CIFAR-10	58.33	82.45	81.65	81.88	**82.61**	82.30
CIFAR-100	50.03	**54.19**	52.88	51.30	51.75	50.67

● 消融实验. 最后, 我们测试了 FedAlt 中两个关键部分, 即块级多输出 (BMR) 和知识自蒸馏 (KD), 对于所提框架的性能影响. 具体而言, 我们分别测试了 FedAlt、FedAlt w/o BMR 和 FedAlt w/o KD 在给定带宽预算 (100 GB) 和 non-IID 程度 (ε=7) 的性能 (即损失函数和测试精度). 如图 7 所示, FedAlt 的性能明显优于其他两种方案. 例如, 当训练轮数为 100 时, FedAlt 的测试精度为 80.04%, 而 FedAlt w/o BMR 和 FedAlt w/o KD 的测试精度分别为 70.89% 和 71.64%. 换而言之, 相比于其他两种方案, FedAlt 可以提高平均 8.78% 的精度. 在所提框架 FedAlt 中, 知识自蒸馏可以改善方案在 non-IID 数据分布下的性能, 块级多输出可以显著减少带宽消耗, 因此在给定带宽的前提下, FedAlt 和 FedAlt w/o KD 的性能显著优于 FedAlt w/o BMR. 综上所述, 块级多输出和知识自蒸馏对于 FedAlt 的性能改进有显著的必要性.

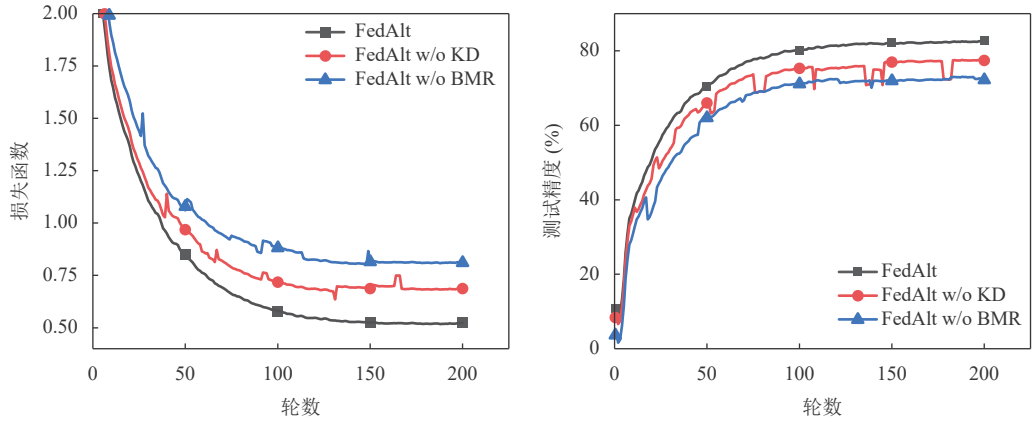


图 7　BMR 和 KD 对于 FedAlt 的性能影响测试

总之, FedAlt 相比于其他基线方法拥有最好的性能. 首先, 当客户端之间的本地数据分布是非独立同分布时, FedAlt 在经历一定轮数的全局训练之后获得的全局模型相比于其他基线方法训练出的全局模型具有最高的测试精度. 其次, 即使是在资源 (完成时间和网络带宽) 约束条件下, FedAlt 训练出的全局模型也具有最好的泛化性能.

6 总　结

本文提出了 FedAlt 框架, 在 FedAvg 基础上通过结合知识自蒸馏和块级多输出的正则化来缓解数据异构带来的模型训练性能下降问题. 此外, 本文在服务器和客户端分别设计了相应的算法 (BMD 和 BMR), 根据客户端的不同的本地数据分布、计算和通信能力来确定服务器分发给客户端的模型块数量. 最后, 本文通过大量实验证明, 与现有方法相比, FedAlt 可以有效提高模型测试精度, 同时显著减少网络带宽损耗.

References

[1] Jain P, Gyanchandani M, Khare N. Big data privacy: A technological perspective and review. Journal of Big Data, 2016, 3: 25. [doi: 10.1186/s40537-016-0059-y]

[2] Shi WS, Cao J, Zhang Q, Li YHZ, Xu LY. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637–646. [doi: 10.1109/JIOT.2016.2579198]

[3] Shi WS, Sun H, Cao J, Zhang Q, Liu W. Edge computing——An emerging computing model for the Internet of everything era. Journal of Computer Research and Development, 2017, 54(5): 907–924 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2017.20160941]

中国知网 https://www.cnki.net

[4] McMahan B, Moore E, Ramage D, Hampson S, Arcas BAY. Communication-efficient learning of deep networks from decentralized data. In: Proc. of the 20th Int'l Conf. on Artificial Intelligence and Statistics. 2017. 1273–1282.

[5] Zhang C, Xie Y, Bai H, Yu B, Li WH, Gao Y. A survey on federated learning. Knowledge-based Systems, 2021, 216: 106775. [doi: 10.1016/j.knosys.2021.106775]

[6] Sattler F, Wiedemann S, Müller KR, Samek W. Robust and communication-efficient federated learning from non-IID data. IEEE Trans. on Neural Networks and Learning Systems, 2020, 31(9): 3400–3413. [doi: 10.1109/TNNLS.2019.2944481]

[7] Hamer J, Mohri M, Suresh AT. FedBoost: A communication-efficient algorithm for federated learning. In: Proc. of the 37th Int'l Conf. on Machine Learning. 2020. 3973–3983.

[8] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated optimization in heterogeneous networks. In: Proc. of the 3rd MLSys Conf. 2020. 429–450.

[9] Kim J, Kim G, Han B. Multi-level branched regularization for federated learning. In: Proc. of the 39th Int'l Conf. on Machine Learning. 2022. 11058–11073.

[10] Chen HK, Frikha A, Krompass D, Gu JD, Tresp V. FRAug: Tackling federated learning with non-IID features via representation augmentation. In: Proc. of the 2023 IEEE/CVF Int'l Conf. on Computer Vision. Paris: IEEE, 2023. 4826–4836. [doi: 10.1109/ICCV51070.2023.00447]

[11] Jiang ZD, Xu Y, Xu HL, Wang ZY, Qiao CM, Zhao YM. FedMP: Federated learning through adaptive model pruning in heterogeneous edge computing. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022. 767–779. [doi: 10.1109/ICDE53745.2022.00062]

[12] Kim M, Yu S, Kim S, Moon SM. DepthFL: Depthwise federated learning for heterogeneous clients. In: Proc. of the 11th Int'l Conf. on Learning Representations. 2022.

[13] Suvorov R, Logacheva E, Mashikhin A, Remizova A, Ashukha A, Silvestrov A, Kong N, Goka H, Park K, Lempitsky V. Resolution-robust large mask inpainting with fourier convolutions. In: Proc. of the 2022 IEEE/CVF Winter Conf. on Applications of Computer Vision. Waikoloa: IEEE, 2022. 3172–3182. [doi: 10.1109/WACV51458.2022.00323]

[14] Brown TB, Mann B, Ryder N, *et al*. Language models are few-shot learners. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 1877–1901.

[15] Zhang LF, Song JB, Gao AN, Chen JW, Bao CL, Ma KS. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In: Proc. of the 2019 IEEE/CVF Int'l Conf. on Computer Vision. Seoul: IEEE, 2019. 3712–3721. [doi: 10.1109/ICCV.2019.00381]

[16] Yang Q, Liu Y, Chen TJ, Tong YX. Federated machine learning: Concept and applications. ACM Trans. on Intelligent Systems and Technology, 2019, 10(2): 12. [doi: 10.1145/3298981]

[17] Caldarola D, Dputo B, Ciccone M. Window-based model averaging improves generalization in heterogeneous federated learning. In: Proc. of the 2023 IEEE/CVF Int'l Conf. on Computer Vision Workshops. Paris: IEEE, 2023. 2255–2263. [doi: 10.1109/ICCVW60793.2023.00240]

[18] Zhu ZD, Hong JY, Zhou JY. Data-free knowledge distillation for heterogeneous federated learning. In: Proc. of the 38th Int'l Conf. on Machine Learning. 2021. 12878–12889.

[19] Achituve I, Shamsian A, Navon A, Chechik G, Fetaya E. Personalized federated learning with Gaussian processes. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 8392–8406.

[20] Collins L, Hassani H, Mokhtari A, Shakkottai S. Exploiting shared representations for personalized federated learning. In: Proc. of the 38th Int'l Conf. on Machine Learning. 2021. 2089–2099.

[21] Zhang J, Guo S, Ma XS, Wang HZ, Xu WC, Wu FJ. Parameterized knowledge transfer for personalized federated learning. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 10092–10104.

[22] Marfoq O, Neglia G, Kameni L, Vidal R. Personalized federated learning through local memorization. In: Proc. of the 39th Int'l Conf. on Machine Learning. 2022. 15070–15092.

[23] Stich SU, Cordonnier JB, Jaggi M. Sparsified SGD with memory. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal: Curran Associates Inc., 2018. 4452–4463.

[24] Basu D, Data D, Karakus C, Diggavi S. Qsparse-local-SGD: Distributed SGD with quantization, sparsification, and local computations. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 14695–14706.

[25] Jiang Y, Wang SQ, Valls V, Ko BJ, Lee WH, Leung KK, Tassiulas L. Model pruning enables efficient federated learning on edge devices. IEEE Trans. on Neural Networks and Learning Systems, 2023, 34(12): 10374–10386. [doi: 10.1109/TNNLS.2022.3166101]

[26] Tonellotto N, Gotta A, Nardini FM, Gadler D, Silvestri F. Neural network quantization in federated learning at the edge. Information Sciences, 2021, 575: 417–436. [doi: 10.1016/j.ins.2021.06.039]

[27] Idelbayev Y, Carreira-Perpiñán MÁ. Low-rank compression of neural nets: Learning the rank of each layer. In: Proc. of the 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Seattle: IEEE, 2020. 8046–8056. [doi: 10.1109/CVPR42600.2020.

中国知网 https://www.cnki.net

00807]
[28] Fan X, Wang Y, Huo Y, Tian Z. Joint optimization of communications and federated learning over the air. IEEE Trans. on Wireless Communications, 2022, 21(6): 4434–4449. [doi: 10.1109/TWC.2021.3130111]
[29] Ozkara K, Singh N, Data D, Diggavi S. QUPED: Quantized personalization via distillation with applications to federated learning. In: Proc. of the 35th Int'l Conf. on Neural Information Processing Systems. Curran Associates Inc., 2021. 3622–3634.
[30] Chen MZ, Gündüz D, Huang KB, Saad W, Bennis M, Feljan AV, Poor HV. Distributed learning in wireless networks: Recent progress and future challenges. IEEE Journal on Selected Areas in Communications, 2021, 39(12): 3579–3605. [doi: 10.1109/JSAC.2021.3118346]
[31] Zhang L, Shen L, Ding L, Tao DC, Duan LY. Fine-tuning global model via data-free knowledge distillation for non-IID federated learning. In: Proc. of the 2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. New Orleans: IEEE, 2022. 10164–10173. [doi: 10.1109/CVPR52688.2022.00993]
[32] Pillutla K, Kakade SM, Harchaoui Z. Robust aggregation for federated learning. IEEE Trans. on Signal Processing, 2022, 70: 1142–1154. [doi: 10.1109/TSP.2022.3153135]
[33] He KM, Zhang XY, Ren SQ, Sun J. Deep residual learning for image recognition. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 770–778. [doi: 10.1109/CVPR.2016.90]
[34] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436–444. [doi: 10.1038/nature14539]
[35] Liu JC, Xu HL, Zhao GM, Qian C, Fan XP, Huang LS. Incremental server deployment for scalable NFV-enabled networks. In: Proc. of the 2020 IEEE Conf. on Computer Communications (INFOCOM 2020). Toronto: IEEE, 2020. 2361–2370. [doi: 10.1109/INFOCOM41043.2020.9155364]
[36] de Boer PT, Kroese DP, Mannor S, Rubinstein RY. A tutorial on the cross-entropy method. Annals of Operations Research, 2005, 134(1): 19–67. [doi: 10.1007/s10479-005-5724-z]
[37] van Erven T, Harremoes P. Rényi divergence and Kullback-Leibler divergence. IEEE Trans. on Information Theory, 2014, 60(7): 3797–3820. [doi: 10.1109/TIT.2014.2320500]
[38] Acar DAE, Zhao Y, Zhu RZ, Navarro RM, Mattina M, Whatmough P, Saligrama V. Debiasing model updates for improving personalized federated training. In: Proc. of the 38th Int'l Conf. on Machine Learning. 2021. 21–31.
[39] Paszke A, Gross S, Massa F, *et al*. PyTorch: An imperative style, high-performance deep learning library. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 8026–8037.
[40] Liu JC, Xu HL, Wang L, Xu Y, Qian C, Huang JY, Huang H. Adaptive asynchronous federated learning in resource-constrained edge computing. IEEE Trans. on Mobile Computing, 2023, 22(2): 674–690. [doi: 10.1109/TMC.2021.3096846]
[41] Krizhevsky A. Learning multiple layers of features from tiny images [MS. Thesis]. Department of Computer Science, University of Toronto, 2009.
[42] Li QB, He BS, Song D. Model-contrastive federated learning. In: Proc. of the 2021 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Nashville: IEEE, 2021. 10708–10717. [doi: 10.1109/CVPR46437.2021.01057]
[43] Wang L, Xu Y, Xu HL, Jiang ZD, Chen M, Zhang WY, Qian C. BOSE: Block-wise federated learning in heterogeneous edge computing. IEEE/ACM Trans. on Networking, 2024, 32(2): 1362–1377. [doi: 10.1109/TNET.2023.3316421]

附中文参考文献

[3] 施巍松, 孙辉, 曹杰, 张权, 刘伟. 边缘计算: 万物互联时代新型计算模型. 计算机研究与发展, 2017, 54(5): 907–924. [doi: 10.7544/issn1000-1239.2017.20160941]

作者简介

刘建春, 博士, 副研究员, CCF 专业会员, 主要研究领域为物联网, 计算机网络, 边缘智能.
梁文艺, 硕士生, 主要研究领域为边缘智能, 流推理加速.
徐宏力, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为物联网, 计算机网络, 边缘智能.
马千飘, 博士, 副教授, CCF 专业会员, 主要研究领域为软件工程, 软件安全性, 形式化方法.
黄刘生, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为网络, 信息安全, 云计算, 大数据.

中国知网 https://www.cnki.net