软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007701] [CSTR: 32375.14.jos.007701]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于特征压缩与特征融合的高效分裂联邦学习框架*

廖云铭[1,2], 马欣莹[1,2], 许 杨[1,2], 徐宏力[1,2], 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)

通信作者: 许杨, E-mail: xuyangcs@ustc.edu.cn

摘 要: 为了提高人工智能应用的性能, 大规模模型因其优秀的广义能力而受到越来越多的关注. 然而, 大规模模型的训练和传输会给资源有限的边缘节点带来巨大的计算和通信负担, 并且完整模型的交换可能会侵犯模型的隐私. 为了减轻节点的负担并保护模型的隐私, 集成数据并行和模型并行的分裂联邦学习被提出. 然而, 除了资源有限的挑战, 分裂联邦学习在边缘计算系统中还面临着另外两个关键的挑战, 即统计异构性和系统异构性. 为了解决这些挑战, 结合分裂联邦学习的特性, 提出一种新的基于特征压缩与特征融合的高效分裂联邦学习框架 SplitCP. 具体地说, 特征压缩的目的是降低节点的通信资源消耗, 并通过为异构节点分配合适的压缩比以提高训练效率. 而特征融合的目的是将节点的特征融合成一个混合特征序列, 近似等同于独立同分布数据的特征, 克服统计异构性挑战来提高模型的精度. 此外, SplitCP 通过分析自适应特征压缩与特征融合的耦合关系, 探索如何联合优化以提升分裂联邦学习的模型训练性能. 在使用 80 台 NVIDIA Jetson 边缘设备搭建的物理平台上进行广泛的实验, 实验结果表明, 与基线相比, SplitCP 可以将最终模型精度提高 6.46%–26.15%, 同时实现训练加速约 1.31–3.51 倍.

关键词: 边缘计算; 分裂联邦学习; 数据异构; 系统异构

中图法分类号: TP18

中文引用格式: 廖云铭, 马欣莹, 许杨, 徐宏力, 黄刘生. 基于特征压缩与特征融合的高效分裂联邦学习框架. 软件学报. http://www.jos.org.cn/1000-9825/7701.htm

英文引用格式: Liao YM, Ma XY, Xu Y, Xu HL, Huang LS. Efficient Split Federated Learning Framework with Feature Compression and Feature Fusion. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7701.htm

Efficient Split Federated Learning Framework with Feature Compression and Feature Fusion

LIAO Yun-Ming[1,2], MA Xin-Ying[1,2], XU Yang[1,2], XU Hong-Li[1,2], 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)

Abstract: To improve the performance of AI applications, large-scale models have received increasing attention due to their excellent generalization ability. However, the training and transmission of large-scale models impose significant computational and communication burdens on resource-constrained edge nodes, and the exchange of complete models may violate model privacy. To reduce the burden on nodes and protect model privacy, split federated learning (SFL), which integrates data parallelism and model parallelism, is proposed. However, in addition to the challenge of resource limitations, SFL also faces two other critical challenges in edge computing (EC) systems, i.e., statistical heterogeneity and system heterogeneity. To address these challenges, this study proposes a novel, efficient SFL framework based on feature compression and feature fusion, termed SplitCP, by incorporating the characteristics of SFL. Specifically, feature compression aims to reduce the communication resource consumption of nodes and improve training efficiency by assigning appropriate compression ratios to heterogeneous nodes. Feature fusion aims to merge the features of nodes into a mixed feature sequence, which is approximately equivalent to the features of independent and identically distributed (IID) data, thus overcoming the challenge of

* 基金项目: 国家自然科学基金 (624B2136, 62472401, 62132019, 62502487); 中央高校基本科研业务费专项资金 (WK2150250044)
收稿时间: 2025-11-17; 修改时间: 2026-03-04; 采用时间: 2026-05-14; jos 在线出版时间: 2026-08-26

statistical heterogeneity and improving model accuracy. Moreover, by analyzing the coupling relationship between adaptive feature compression and feature fusion, SplitCP explores how to jointly optimize them to improve the model training performance of SFL. Extensive experiments are conducted on a physical platform built with 80 NVIDIA Jetson edge devices, and the experimental results show that, compared with the baselines, SplitCP can improve the final model accuracy by 6.46% to 26.15% while achieving training speedups of approximately 1.31× to 3.51×.

Key words: edge computing; split federated learning; data heterogeneity; system heterogeneity

作为边缘智能中一种新兴和流行的技术, 联邦学习 (federated learning, FL) 提出节点间通过数据并行方式来协作训练全局模型[1]. 在参数服务器 (parameter server, PS)[2]的协调下, 参与的边缘节点定期在其本地数据集上训练深度学习模型, 然后在不公开原始数据的情况下将模型推送到参数服务器上进行全局聚合[3]. 谷歌曾利用联邦学习在保护数据隐私的前提下开发了可以改善用户体验的 Gboard 应用程序[4]. 为了提高人工智能应用程序或服务的性能, 增加深度学习模型的参数量通常是实用且有效的[5]. 然而, 由于 CPU、GPU、内存等硬件限制, 训练大规模模型对边缘节点挑战巨大[6]. 此外, 在边缘节点和参数服务器间进行大规模模型的传输也会导致显著的通信延迟, 且完整模型的交换可能会侵犯模型的隐私[7].

为了减轻资源受限边缘节点的计算/通信负担, 并更好地保护模型隐私, 集成数据并行和模型并行的分裂联邦学习 (split federated learning, SFL) 被提出[8]. 分裂联邦学习在分裂层将整个模型拆分为两个子模型, 即底部模型和顶部模型. 底部模型 (靠近输入层) 主要在边缘节点进行本地训练, 而顶部模型 (靠近输出层) 在参数服务器上进行训练[9]. 因此, 分裂联邦学习可以显著降低边缘节点的计算负载, 使训练大规模模型可行且有效[10]. 与典型的联邦学习不同, 在分裂联邦学习中, 只有底部模型和特征 (即前向传播时分裂层的输出)/梯度 (即反向传播时分裂层的输入) 在节点和参数服务器之间进行交换. 由于特征以及底部模型的尺寸均比整个模型的尺寸要小得多, 因此通信负载将被显著释放. 例如, 一个 16 层的 VGG16[11]的大小约为 321 MB, 而在第 13 层将 VGG16 分裂为两部分模型, 并将批处理大小设置为 64 时, 它的底部模型和对应特征的大小分别约为 56 MB 和 3 MB. 同时, 考虑到节点只能访问部分模型, 而原始数据仍在本地处理, 用户和模型的隐私都将得到有效的保护. 此外, 现有的差分隐私[12]和同态加密[13]等隐私保护技术也可以进一步保护分裂联邦学习训练过程中特征/梯度的隐私.

虽然分裂联邦学习展现了上述优势, 但其在边缘计算的实际应用中仍然面临着两个关键的挑战. (1) 统计异构: 首先, 边缘节点总是根据其位置和用户偏好来收集本地数据[14]. 此外, 为了防止隐私泄露, 边缘节点的原始数据不与其他节点共享, 也不会发送到参数服务器, 导致边缘节点间的数据分布通常是非独立同分布 (non-independent and identically distributed, non-IID) 的, 即统计异构性[15]. 在实际的模型训练过程中, 统计异构性通常会降低模型收敛速度甚至破坏模型精度. (2) 系统异构: 在边缘计算系统中, 边缘节点通常配置不同的芯片以及其他硬件, 其提供的计算和通信能力差异明显[16]. 边缘节点之间的计算能力 (例如 CPU 频率) 和通信能力 (例如带宽、吞吐量) 差异可达 10 倍甚至 100 倍[17]. 系统异构性对同步模型的训练过程产生了显著的负面影响, 迫使能力强的边缘节点等待能力弱的边缘节点, 从而导致模型训练效率严重下降[18]. 为了扩大分裂联邦学习解决异构性的能力, 我们首先探讨了分裂联邦学习的独特特性, 并提出了一种新的基于特征压缩与特征融合的高效分裂联邦学习框架 SplitCP. 该框架的设计是基于两个基本的观察结果. (1) 边缘节点需要频繁地与参数服务器交换特征或梯度, 消耗大量通信资源, 特征压缩可以降低节点的通信资源消耗. 同时为不同的边缘节点分配适当的特征压缩比将有助于适应能力多样性[19]. 例如, 能力弱的边缘节点可以分配较大的特征压缩比, 使边缘节点之间执行正反向传播的时间消耗近似, 从而解决系统异构性. (2) 在分裂联邦学习中, 顶部模型可以视为分类器[20], 非独立同分布数据的特征总是误导顶部模型的收敛方向, 导致模型精度的下降. 如果我们合并来自不同边缘节点的特征, 形成一个混合特征序列, 使其近似等同于来自独立同分布 (independent and identically distributed, IID) 数据小批量数据的特征, 那么顶部模型可以沿着合理的最优方向更新.

基于上述见解, SplitCP 探索通过结合特征压缩和特征融合技术来建立一个高效的分裂联邦学习系统. 该系统设计的难点在于特征压缩与特征融合之间的相互作用. 一方面, 为了充分利用本地数据, 我们希望在每次迭代中从不同的节点收集足够的特征, 使得特征序列可以形成近似等同于独立同分布数据的特征. 为此, 每一轮均需要动态

选择合适的训练节点集并配置批处理大小来克服统计异构性的影响, 但批处理大小的配置可能会放大异构节点间的本地计算时间差异, 进而降低训练效率. 另一方面, SplitCP 需要为不同节点分配适当的特征压缩比来解决系统异构性挑战, 但节点间的特征压缩比差异过大会导致特征融合的效果下降, 影响最终模型训练精度. 只有通过联合优化特征压缩与特征融合, 为每轮参与训练的节点集联合决策合适的特征压缩比与批处理大小, SplitCP 才可以很好地同时解决异构性挑战, 并实现高效的分裂联邦学习系统, 根据我们的调研, 这一点在现有文献中尚未进行研究. 简而言之, 本文的主要贡献总结如下.

● 深入挖掘分裂联邦学习的特性, 并提出一种新的基于特征压缩与特征融合的高效分裂联邦学习框架 SplitCP, 以克服数据异构和系统异构挑战.

● 分析特征压缩和特征融合对训练性能的联合影响, 并得到它们之间的耦合关系. 然后, SplitCP 在异构性限制下动态地选择每轮参与训练的边缘节点, 并配置合适的特征压缩比与批处理大小, 从而提高模型的准确性和训练效率.

● 通过一个拥有 80 台 NVIDIA Jetson 边缘设备的物理平台对 SplitCP 的性能进行广泛评估. 实验结果表明, 与基线相比, SplitCP 可以将最终模型精度提高约 6.46%–26.15%, 同时实现模型训练加速约 1.31–3.51 倍.

本文第 1 节介绍分裂联邦学习与联邦学习的国内外相关工作. 第 2 节系统地介绍相关基础知识以及本文的研究动机. 接着, 本文在第 3 节中详细阐述 SplitCP 中的系统概述、控制模块以及训练模块设计. 然后, 本文在第 4 节展示实验设置与实验结果. 最后, 第 5 节对本文进行总结.

1 相关工作

1.1 分裂联邦学习

现有的分裂联邦学习 (SFL) 研究最初旨在训练大规模深度学习模型时, 将计算任务从资源受限的边缘节点卸载到参数服务器. 然而, 这些方法普遍无法同时解决系统异构性与统计异构性挑战. 例如, Thapa 等人[21]证明了分裂联邦学习的可行性, 并首次提出了名为 SplitFed 的方法, 该方法在每次本地模型更新后都会聚合底部模型. 虽然该机制能够实现联合训练, 但频繁的聚合模型造成了很高的通信资源开销. 为缓解通信负担, Han 等人[22]提出了 LocSplitFed, 通过节点模型与参数服务器模型更新的并行化处理, 并结合基于局部损失的训练策略, 无需聚合底部模型, 实现快速且通信高效的分裂联邦学习. 随后, Oh 等人[23]提出了 LocFedMix-SL, 该方法通过增大本地更新频次, 减少底部模型聚合的次数, 在保留 SplitFed 和 LocSplitFed 的优势下实现高效分裂联邦学习. 然而, LocFedMix-SL 仍未能充分释放异构边缘节点的算力潜能, 其训练效率仍受制于系统异构性. Liao 等人[10]则提出了自适应分裂联邦学习 AdaSFL, 通过为不同边缘节点分配差异化的批处理大小, 适配节点间的异构算力与通信能力, 以提升训练效率. 但该方法依然无法有效应对统计异构性, 在实际应用场景中面临最终训练模型精度的下降. 尽管 MergeSFL[24]通过联合特征融合与批处理大小调控优化来同时应对系统异构与统计异构挑战, 但其忽略了节点需要频繁地与参数服务器交换特征和梯度产生的通信资源消耗大与通信延迟高的问题. SplitCP 在 MergeSFL 的基础上引入特征压缩技术来降低分裂联邦学习的通信资源开销, 并通过联合自适应特征压缩与特征融合优化进一步克服异构挑战.

1.2 联邦学习

在分裂联邦学习出现之前, 系统异构性与统计异构性已在诸多典型的联邦学习 (FL) 研究中得到广泛探讨与部分解决[25]. 在系统异构性方面, 一些研究[26]致力于优化不同节点的本地更新频率与批处理大小. 例如, Xu 等人[26]提出了 FedLamp 框架, 为计算与通信能力较强的边缘节点分配更高的本地更新频率; Ma 等人[27]则提出了为异构节点动态调整批处理大小与学习率的策略. 在统计异构性方面, 部分工作[28]通过主动选择高价值数据样本来提升全局模型的收敛性. 例如, Li 等人[28]提出在训练中优先使用重要性权重高的样本; Shin 等人[25]提出了 FedBalancer, 通过引入截止时间控制策略以优化模型的时间-精度表现. 此外, 其他研究[29]通过节点选择策略来同时缓解系统与统计异构性. 以 Li 等人[16]提出的 PyramidFL 为例, 该方法通过度量已选节点与未选节点间的分布差异, 实现细粒度的节点选择, 从而更充分地利用异构节点的计算资源与本地数据.

然而, 上述联邦学习研究往往难以直接应用在节点资源受限的模型训练场景中. 原因在于, 当边缘节点无法承载完整大规模模型训练时, 相关优化策略将失效. 此外, 这些方法的优化思路也无法直接迁移至分裂联邦学习或SplitCP 框架中, 因为在分裂联邦学习中, 边缘节点仅保留和训练底部模型, 而顶部模型驻留于参数服务器, 双方需持续交换特征与梯度信息才能完成一次模型更新迭代, 这对系统的带宽需求与同步机制提出了新的挑战.

2 背景与动机

2.1 联邦学习

边缘计算推动了计算资源从集中式云端向分布式边缘节点的迁移, 与此同时, 各行各业对模型本地化部署及数据主权保护的需求持续升级[18]. 在此背景下, 联邦学习 (FL) 作为创新的分布式机器学习框架被提出并快速发展, 其核心特点是数据不动模型动[30]. 具体来说, 各参与方 (如边缘设备或机构) 无需共享本地数据, 仅通过传输模型参数或梯度信息, 即可在参数服务器的协调下完成全局共享模型的训练任务. 通过规避原始数据的集中存储或传输流程, 联邦学习既有效保护了用户数据隐私, 同时也破解了边缘计算环境中普遍存在的数据孤岛难题[31]. 联邦学习的训练流程从参数服务器初始化全局模型开始, 随后分发给各个参与训练的节点; 每个节点基于本地数据进行模型参数更新, 随后上传至服务器; 服务器通过联邦平均算法 (FedAvg[32]) 加权聚合参数, 生成新一轮全局模型并再次下发. 上述过程循环迭代直至全局模型收敛, 或者资源 (例如时间、计算资源、通信资源) 耗尽[33].

经典的联邦学习系统包含 N 个边缘节点和一个参数服务器 (PS), 每个边缘节点 $i \in \{1,2,\ldots,N\}$ 各自持有本地数据集 $\mathbb{D}_i$, 并负责维护本地模型 $\mathbf{w}_i$, 参数服务器则负责维护全局模型 $\mathbf{w}$. 联邦学习的优化目标是训练一个全局模型 $\mathbf{w}^*$, 使得经验损失函数 $F(\mathbf{w})$ 最小化. 数学表述如下:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} F(\mathbf{w}) = \sum_{i=1}^{N} \frac{d_i}{D} F_i(\mathbf{w}) \tag{1}$$

其中, $F_i(\mathbf{w})$ 表示边缘节点 i 的损失函数, d_i 表示数据集 $\mathbb{D}_i$ 中包含的样本个数, $D=\sum_{i=1}^{N} d_i$ 表示总的样本个数. 由于多数机器学习任务具有高度非线性的结构特性, 联邦学习的优化问题通常采用基于梯度信息的迭代下降算法进行求解. 以小批量随机梯度下降 (stochastic gradient descent, SGD) 算法为例, 边缘节点在单个数据批量 (mini-batch) 上执行梯度下降操作被视为一次本地更新, 一轮训练周期内往往进行多次本地更新. 联邦学习的主要步骤如下.

(1) 初始化: 当训练流程启动时, 参数服务器首先构建全局模型 $\mathbf{w}_0$, 并对通信迭代周期、数据批处理大小及学习率等核心超参数进行配置, 接着向参与训练的边缘节点同步传输 $\mathbf{w}_0$.

(2) 本地更新: 边缘节点接收到全局模型后, 基于本地数据进行模型训练. 我们定义第 h 个通信轮次内, 第 k 次本地更新时边缘节点 i 的本地模型为 $\mathbf{w}_i^{h,k}$. 因此, 边缘节点 i 的本地模型更新过程可表述为:

$$\mathbf{w}_i^{h,k+1} = \mathbf{w}_i^{h,k} - \eta \nabla F_i(\mathbf{w}_i^{h,k}) \tag{2}$$

其中, η 和 $\nabla F_i(\mathbf{w}_i^{h,k})$ 分别代表节点 i 的学习率和梯度. 在完成预设的本地更新次数后, 边缘节点 i 将更新参数 $\mathbf{w}_i^h$ 上传到服务器端.

(3) 模型聚合与分发: 在收到所有的更新参数后, 参数服务器执行聚合并更新全局模型, 过程如下:

$$\mathbf{w}^{h+1} = \sum_{i=1}^{N} \frac{d_i}{D} \mathbf{w}_i^h \tag{3}$$

更新后的全局模型 $\mathbf{w}^{h+1}$ 将继续同步给所有节点, 并重复上述步骤 (2)、(3) 开启新一轮训练, 直至模型收敛.

2.2 分裂联邦学习

考虑一个带有参数服务器和 N 个边缘节点的边缘计算系统, 分裂联邦学习 (SFL) 通过一个由参数服务器协调的、松散的节点联合会来执行深度学习任务. 分裂联邦学习的基本思想是在分裂层上将模型 $\mathbf{w}$ 分成两个子模型, 即 $\mathbf{w} = [\mathbf{w}_b, \mathbf{w}_p]$. $\mathbf{w}_b$ 表示底部模型, $\mathbf{w}_p$ 表示顶部模型. 为了便于描述, 我们以 CNN 模型为例. 底部模型通常由输入

层和卷积层组成, 而顶部模型通常由全连接层和输出层组成[34]. 在分裂联邦学习中, 参数服务器维护顶部模型 $\mathbf{w}_p$, 而每个节点 i 使用其本地数据 $\mathbb{D}_i$ 训练一个底部模型 $\mathbf{w}_{b,i}$. 此外, 边缘节点通过与参数服务器上的顶部模型进行持续交换特征/梯度来完成训练过程. 分裂联邦学习的目标是找到最优的模型 $\mathbf{w}^* = [\mathbf{w}_b^*, \mathbf{w}_p^*]$, 使损失函数 $F(\mathbf{w})$ 最小化, 如下:

$$\min_{\mathbf{w}=[\mathbf{w}_b,\mathbf{w}_p]} F(\mathbf{w}) \triangleq F_p(\mathbf{w}_p) + \frac{1}{N}\sum_{i=1}^{N} F_{b,i}(\mathbf{w}_{b,i}) \tag{4}$$

其中, $F_{b,i}(\mathbf{w}_{b,i})$ 和 $F_p(\mathbf{w}_p)$ 分别表示底部模型 $\mathbf{w}_{b,i}$ 和顶部模型 $\mathbf{w}_p$ 的损失函数. 由于大多数深度学习任务的内在复杂性, 获得该等式的封闭解通常具有挑战性. 然而, 上述等式可以通过小批量随机梯度下降算法来求解.

分裂联邦学习的基本训练过程包括 3 个主要阶段, 即特定节点底部模型的前向/反向传播、顶部模型的前向/反向传播以及底部模型在参数服务器上的全局聚合[35]. 首先, 每个节点使用一批数据样本进行前向传播, 并将分裂层输出的特征发送到参数服务器. 随后, 参数服务器基于所有节点的特征执行前向/反向传播来更新顶部模型. 然后, 参数服务器将反向传播的梯度发送回节点, 使其可通过反向传播来更新底部模型. 这种完整的前向/反向传播过程被认为是一个本地迭代. 经过多次本地迭代, 参数服务器聚合所有节点的底部模型, 并将聚合的底部模型发送给节点进行后续训练. 以上整个训练过程可被视为一个通信轮次. 设 $\mathbf{w}_{b,i}^{h,k}$ 表示在第 h 轮中第 k 次迭代时节点 i 的底部模型. 底部模型一次本地迭代更新如下:

$$\mathbf{w}_{b,i}^{h,k+1} = \mathbf{w}_{b,i}^{h,k} - \eta\tilde{\nabla}F_{b,i}(\mathbf{w}_{b,i}^{h,k}) \tag{5}$$

其中, $\tilde{\nabla}F_{b,i}(\mathbf{w}_{b,i}^{h,k}) = \dfrac{1}{|D_i|}\sum_{x\in D_i}\nabla\ell(x;\mathbf{w}_{b,i}^{h,k})$ 是小批量数据 D_i (批处理大小 $d_i = |D_i|$) 的梯度, 而 $\nabla\ell(x;\mathbf{w}_{b,i}^{h,k})$ 表示在给定底部模型 $\mathbf{w}_{b,i}^{h,k}$ 和输入样本 x 的情况下的随机梯度. 设 $\mathbf{w}_p^{h,k}$ 表示在第 h 轮中第 k 次迭代的顶部模型. 顶部模型一次本地迭代更新如下:

$$\mathbf{w}_p^{h,k+1} = \mathbf{w}_p^{h,k} - \eta\sum_{i=1}^{N}\frac{d_i}{\sum_{i=1}^{N} d_i}\tilde{\nabla}F_{p,i}(\mathbf{w}_p^{h,k}) \tag{6}$$

其中, $\tilde{\nabla}F_{p,i}(\mathbf{w}_p^{h,k}) = \dfrac{1}{|D_i|}\sum_{x\in D_i}\nabla\ell(\mathbf{w}_{b,i}^{h,k}(x);\mathbf{w}_p^{h,k})$ 是顶部损失函数的梯度, 而 $\nabla\ell(\mathbf{w}_{b,i}^{h,k}(x);\mathbf{w}_p^{h,k})$ 表示在给定输入数据样本 x 的情况下, 针对顶部模型 $\mathbf{w}_p^{h,k}$ 和底部模型 $\mathbf{w}_{b,i}^{h,k}$ 的输出 (即特征) 的随机梯度. 在本地更新后, 参数服务器接收并聚合来自所有节点的底部模型:

$$\mathbf{w}_b^{h+1} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{w}_{b,i}^{h+1} \tag{7}$$

2.3 特征融合的重要性

传统集中式训练收集所有数据到云侧, 其数据满足独立同分布 (IID) 的假设, 但是, 由于边缘节点地理上分散, 且数据不共享, 其本地数据分布差异显著, 呈现明显的非独立同分布 (non-IID) 问题 (即统计异构性), 通常会恶化模型训练性能. 在典型的分裂联邦学习框架 (例如 LocFedMix-SL) 中, 参数服务器直接采用每个边缘节点的特征来完成顶部模型的前向/反向传播, 并按顺序将相应的梯度发送回边缘节点. 然而, 来自不同边缘节点的非独立同分布数据的特征可能会阻碍模型沿着最优方向更新, 从而降低模型准确性[36]. 为了应对统计异构挑战, 我们在分裂联邦学习中引入了特征融合策略, 图 1 展示了特征融合的分裂联邦学习与传统框架的不同. 特征融合是将来自不同边缘节点的特征 (小批量) 合并成一个混合特征序列 (大批量), 这个序列近似等同于从独立同分布批量数据中派生出的特征, 并被用于执行顶部模型的前向/反向传播. 这种跨节点的特征融合与在单节点层面简单增大批处理大小有着本质区别. 在数据 non-IID 的场景下, 单节点增大批处理大小通常只能减小经验分布的采样方差, 无法改变本地数据分布偏置的根本问题. 相反, SplitCP 的特征融合手段是跨节点的, 其优势在于能够让原本分散在各个节点的数据实现互通. 通过在参数服务器端将不同标签分布的特征张量在批处理维度上进行拼接, 生成的混合特征

序列在整体分布上更趋近于全局真实的 IID. 这不仅直接校正了顶部模型梯度的下降方向, 而且由于拼接操作纯粹在参数服务器的内存中完成, 整个过程并未引入任何额外的节点间横向通信开销. 随后, 参数服务器将混合的大批量梯度拆分成对应于每个边缘节点的多个小批量梯度, 然后将这些梯度分发到边缘节点进行底部模型的反向传播与参数更新. 特征融合使顶部模型和底部模型都将沿着优化空间中相对正确的方向更新, 这有助于提高模型准确性.

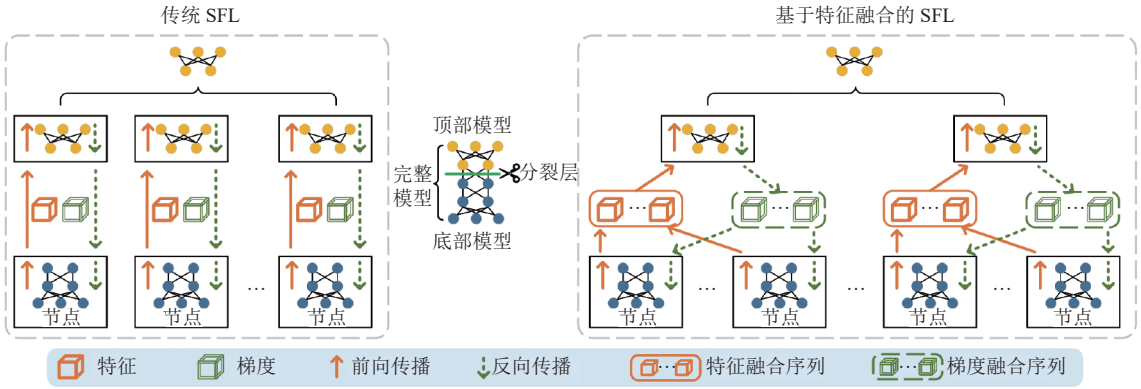


图 1 典型的分裂联邦学习与带有特征融合的分裂联邦学习示意图

为了验证不同策略的效果, 我们定义并比较了以下变体: 典型的分裂联邦学习框架 (SFL-T)、引入自适应特征压缩的分裂联邦学习 (SFL-FC)、引入批处理大小调控的分裂联邦学习 (SFL-BR), 以及带有特征融合的分裂联邦学习 (SFL-FP). 为了验证特征融合的有效性, 我们进行了一组预实验, 使用 SFL-T 和 SFL-FP 在 10 个边缘节点上训练 AlexNet 模型. 我们将来自 CIFAR-10 数据集的非独立同分布数据样本分发给参与训练的边缘节点, 且这些节点的数据合在一起是独立同分布的. 我们记录了使用 SFL-T 和 SFL-FP 训练模型的训练过程和最终模型精度. 从图 2 和图 3 可以看出, 与 SFL-T 相比, SFL-FP 提高了模型精度约 18.2%.

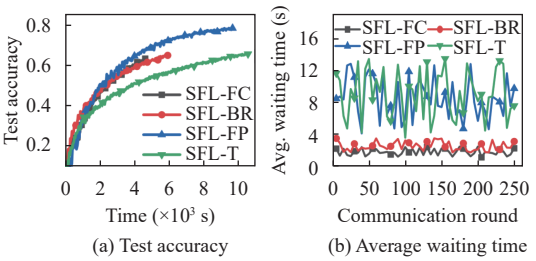


图 2 在非独立同分布 (non-IID) 数据下 4 种方法的模型训练过程与平均等待时间

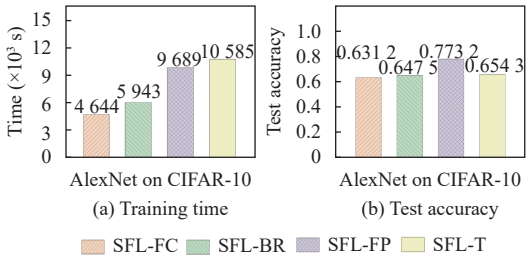


图 3 在非独立同分布 (non-IID) 数据下 4 种方法的训练性能

此外, 为了更好地呈现特征融合对梯度的影响, 我们分别使用 SFL-FP 和 SFL-T 执行一次模型更新迭代, 并通过执行主成分分析 (principal component analysis, PCA) 将顶部模型反向传播的梯度在二维向量空间中可视化. 我们使模型更新迭代从相同的顶部和底部模型开始, 并且各个边缘节点的小批量数据是非独立同分布的, 而这些小批量数据的并集则遵循独立同分布. 图 4(a) 显示了 SFL-T 和 SFL-FP 的顶部模型反向传播的梯度在二维向量空间中被可视化的结果, 图 4(b) 展示了对应于 3 个随机选择的边缘节点的底部模型的梯度, 其中虚线箭头和实线箭头分别表示 SFL-FP 和 SFL-T 中的梯度向量. 此外, 我们还对整个模型 (即顶部和底部模型的组合) 就上述独立同分布小批量数据并集进行了一次迭代的独立模型训练, 由独立随机梯度下降 (SGD) 算法派生的梯度通常指示了正确的优化方向.

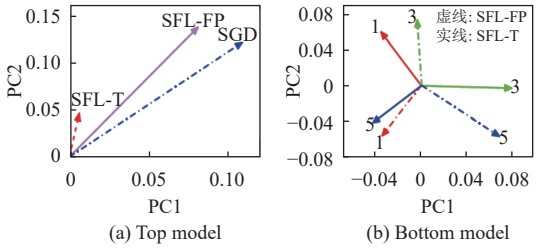


图 4　顶部模型与底部模型梯度可视化

通过图 4(a), 我们观察到 SFL-FP 派生的梯度比 SFL-T 的梯度更接近于 SGD 算法派生的梯度, 因为 SFL-FP 采用的特征融合可以视为对梯度的一种正则化操作, 其确保了顶部模型沿着比 SFL-T 更为正确的方向更新. 此外, 从图 4(b) 中也可以看到 SFL-FP 中顶部模型返回的梯度方向也与 SFL-T 的梯度方向截然不同, 并从图 2(a) 中我们可以得知 SFL-FP 中顶部模型返回的梯度更有助于有效地更新底部模型. 简而言之, SFL-FP 使训练后的模型能够获得比 SFL-T 更快的收敛速度和更高的模型精度, 这证明了特征融合在解决统计异构性方面的优势.

2.4　特征压缩与批处理大小调控的重要性

由于系统异构性, 不同边缘节点的底部模型每次本地迭代的计算时间以及特征的传输时间可能存在显著差异. 并且边缘节点需要参数服务器频繁地交换特征和梯度, 这会消耗大量通信资源, 并导致较长的通信延迟. 为此, 有必要在边缘节点进行特征压缩[37], 节省通信资源的同时加速模型训练. 此外, 考虑到不同边缘节点在计算与通信能力方面的差异与异构性, 可进一步为不同边缘节点配置多样的特征压缩比来降低同步屏障下异构边缘节点的等待时间 (自适应特征压缩), 提高训练效率. 具体而言, 计算与通信能力较弱的节点可采用更高的特征压缩比, 以降低通信负担; 而能力较强的节点则可采用较低的压缩率, 以保留更多特征信息, 从而在通信效率与模型精度之间取得平衡. 然而, 边缘节点间差异过大的特征压缩比将会影响模型训练以及特征融合的效果, 进而导致最终模型精度下降. 另一方面, 也可以通过批处理大小调节机制[38], 根据边缘节点的计算与通信能力自适应地分配不同的批处理大小. 具有较高算力和带宽的节点被配置较大的批处理大小, 以在每次迭代中处理更多数据; 而能力较低的节点则分配较小的批处理大小, 从而有效减少等待时间并缓解系统异构性带来的性能瓶颈. 但批处理大小的调控无法减少模型训练过程中通信资源的消耗.

为了说明 SFL-FC 与 SFL-BR 在异构边缘节点中进行模型训练的优势, 我们在图 2 中呈现了 SFL-FC 和 SFL-BR 的模型训练过程以及每一个通信轮次的平均等待时间, 在图 3 中呈现了 SFL-FC 和 SFL-BR 进行模型训练时的最终模型精度与完成时间. 实验结果显示, 自适应特征压缩以及批处理大小调控都可以有效降低异构边缘节点进行模型训练的平均等待时间, 提高训练效率, 减少模型训练完成时间. 具体而言, SFL-FC 和 SFL-BR 分别仅用 4 644 s 和 5 943 s 就实现了模型收敛, 而 SFL-T 则需要 10 585 s 才能实现模型收敛. 此外, 与 SFL-T 相比, SFL-FC 和 SFL-BR 将每轮的平均等待时间减少了约 69% 和 67%, 并且在达到相似精度时可将模型训练时间缩短约 46% 和 41%. 这些结果显示了自适应特征压缩与批处理大小调控在应对系统异构性方面具有显著优势. 同时, 虽然 SFL-FC 相比 SFL-BR 可以实现更快的模型收敛, 但是 SFL-FC 相比 SFL-BR 在精度上有所下降, 这是因为自适应特征压缩可能会导致在传输过程中, 部分关键的特征信息丢失. 特别是当系统为了强行对齐异构节点的执行时间, 而向计算与通信能力较弱的边缘节点分配极高的特征压缩比时, 会引入显著的压缩误差或数据稀疏化损失. 这种特征失真不仅削弱了前向传播时特征表达的完整性, 也会对反向传播时梯度的精确度造成负面影响, 进而导致模型最终收敛精度的轻微下降. 这也进一步说明了单一的特征压缩策略难以在通信效率与最终模型精度之间实现平衡.

2.5　讨　论

在处理数据 non-IID 带来的统计异构性挑战时, 一个常见的推断是尝试通过在单节点层面增大批处理大小来近似全局分布. 然而, 这种局部调整通常只能减小节点本地经验分布的采样方差, 并不必然改变跨节点混合分布的偏差来源. 因此, 仅靠节点自身的参数调节难以从根本上解决统计偏置带来的梯度偏移问题.

基于上述分析, SplitCP 框架不再将节点发送的特征分布接近 IID 作为必须满足的天然前置条件, 而是将其转

化为节点选择与特征融合过程中的主动优化目标. 具体而言, 这里接近 IID 的对象是指本轮参与特征融合的边缘节点集所产生的混合特征标签分布, 度量指标为该分布与全局真实独立同分布之间的 Kullback-Leibler (*KL*) 散度. 为了实现这一目标, SplitCP 的控制模块通过贪心增量搜索算法, 主动挑选那些本地数据分布与当前混合集合互补的边缘节点加入训练. 这种基于特征互补性的跨节点融合策略, 能够在 PS 端通过批处理维度的拼接, 主动推动混合特征序列在数学分布上稳步趋近于全局 IID 分布, 从而有效校正顶部模型梯度的下降方向.

此外, 对于参与融合的等效批处理大小 (equivalent batch size), 即各节点批处理大小之和, 系统表现出明显的规模敏感性与收益饱和特征. 在特征融合初期 (即等效 batch 较小时), 随着互补节点的加入, 混合分布中的类别缺失得到迅速弥补, *KL* 散度呈现急剧下降趋势, 此时特征融合对模型精度提升的收益最为显著. 然而, 当等效 batch 达到一定规模, 且混合特征已较为均衡地覆盖全局主要类别时, *KL* 散度的下降趋势会显著放缓, 模型精度的提升出现边际收益递减并逐渐趋于饱和. 此时, 若继续盲目扩大等效 batch, 对缓解统计异构性的实质贡献微乎其微, 反而会徒增系统的带宽与内存压力. 这也正是 SplitCP 中引入边际收益阈值以动态截断节点选择的核心理论依据, 从而确保系统在模型收益与物理开销之间取得合理的平衡.

3 系统概述与设计

3.1 系统概述

如图 5 所示, SplitCP 含两个关键模块, 即控制模块和训练模块. 在每一轮开始时, 参数服务器 (PS) 的控制模块首先从所有边缘节点收集状态信息, 包含标签分布、计算和通信能力等状态信息 (①). 随后, 控制模块评估边缘节点的状态, 并在综合考虑统计异构和系统异构的情况下动态选择部分边缘节点参与该轮模型训练. 一旦控制模块做出决策, 还会为被选中的节点分配合适的特征压缩比和批处理大小配置 (②), 这些边缘节点同时会接收最新的底部模型 (③) 进行该轮模型训练. 在训练模块中, 被选中的边缘节点使用其本地数据训练底部模型, 并在每次迭代中发送压缩的特征 (④) 到参数服务器. 在每次迭代中, 参数服务器通过融合来自该轮参与训练的边缘节点的特征, 力求获得一个大尺寸的混合特征序列, 其分布接近独立同分布以克服统计异构性. 之后, 参数服务器利用该混合特征序列完成前向和反向传播, 以更新顶部模型. 在顶部模型的反向传播之后, 参数服务器必须将混合的大尺寸梯度分割为对应于每个边缘节点的小梯度. 随后, 参数服务器将相应的梯度 (⑤) 回传给相应的边缘节点. 经过一定数量的本地迭代后, 参数服务器从所有该轮参与训练的边缘节点聚合底部模型 (⑥), 以获得用于下一轮训练的更新后的底部模型. 值得注意的是, 由于被选中的边缘节点被配置了不同的批处理大小, 因此在执行底部模型聚合时, 将根据节点的批处理大小分配自适应的底部模型聚合权重, 进一步保证底部模型的性能.

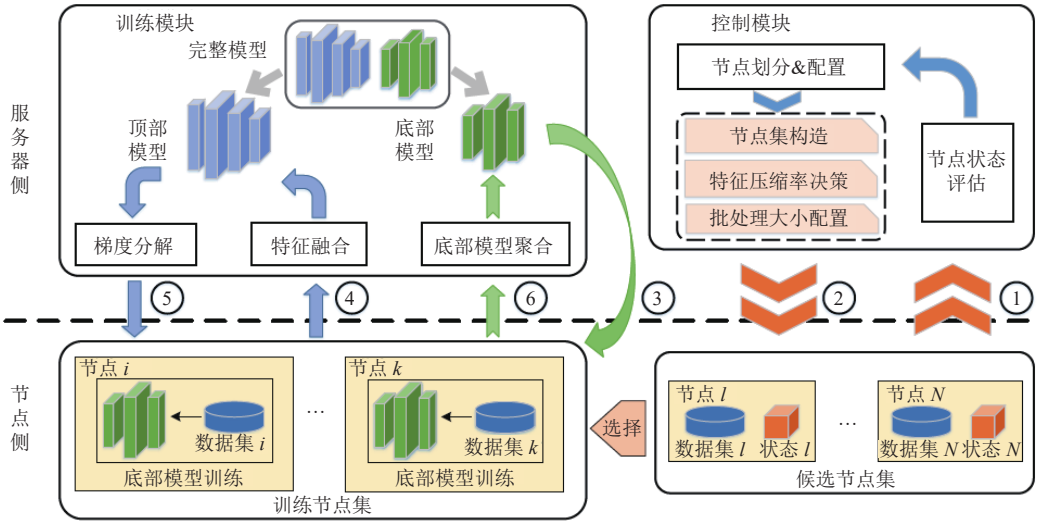


图 5 SplitCP 的系统流程

3.2 控制模块

在每个通信轮次中, 参数服务器的控制模块首先需要评估所有异构边缘节点的状态信息, 并根据它们的计算和通信能力、数据分布等状态信息决策满足要求的参与模型训练的节点集, 并为每一个参与训练的节点配置合适的特征压缩比和批处理大小.

边缘节点状态估计: 为了做出有效的决策, 收集关于参数服务器 (例如, 入口带宽) 以及所有边缘节点 (例如, 标签分布、计算和通信能力) 当前工作状态的信息至关重要[39]. 具体来说, 在边缘计算系统中, 参数服务器的可用入口带宽 B^h 在第 h 个通信轮次中通常是有限的, 并且参数服务器总是消耗大部分带宽来与边缘节点交换特征/梯度. 为防止参数服务器成为瓶颈, 应确保其占用的带宽不超过带宽预算 B^h. 因此, 在第 h 轮开始时, 估计可用入口带宽 B^h 对于确定 SplitCP 中被选中的边缘节点数量与相应特征压缩比以及批处理大小至关重要. 我们基于参数服务器在前几轮的行为来分析入口带宽的统计分布, 并利用这些统计结果来估计第 h 轮中参数服务器的可用入口带宽 B^h.

为了使模型沿着相对最优的方向更新来解决非独立同分布数据问题, SplitCP 融合来自不同边缘节点的特征以形成一个混合特征序列, 该序列近似等同于从独立同分布数据派生的特征. 在此, 标签分布, 即一个用于参数化 M 个类别上的类别标签分类分布的向量 $\mathbf{V}$ ($v_j \geqslant 0, j \in [1, M]$ 且 $\sum_{j=1}^{M} v_j = 1$), 是辅助实现特征融合所必需的. 由于在典型的分裂联邦学习框架[40]中, 边缘节点会传递带有相应标签的特征以在参数服务器上继续顶部模型的前向传播, 因此参数服务器可以直接收集边缘节点特征的标签, 并获得边缘节点 i 关于小批量的标签分布 $\mathbf{V}_i$. 另外, 如果边缘节点不愿意或不允许共享标签, 每个边缘节点 i 可以根据其全部本地数据推导出标签分布 $\mathbf{V}_i$, 并在训练前报告给参数服务器, 这可以保护特定样本的标签信息不被暴露. 考虑到隐私泄露是分裂联邦学习中的一个重要挑战, 一些流行的隐私保护技术, 例如差分隐私、同态加密等相关技术[41], 都可以应用于保护原始数据、特征/梯度和模型的隐私, 并且这些技术与 SplitCP 的主要关注点是正交的.

对边缘节点随时间变化的计算和通信能力的估计, 对于 SplitCP 为参与的边缘节点制定合适特征压缩比与批处理大小策略也至关重要. 作为代理指标, 我们采用边缘节点 i 在第 h 轮处理单个数据样本的计算时间 μ_i^h 和相应的传输时间 β_i^h 来指示边缘节点的计算和通信能力, 并且这些时间可以在模型训练期间由边缘节点直接记录. 在第 h 轮模型训练开始前, 参数服务器可以从所有边缘节点收集最新的计算时间 $\hat{\mu}_i^h$ 和传输时间 $\hat{\beta}_i^h$. 此外, 我们引入了带有边缘节点历史状态的移动平均来提高估计的鲁棒性[42]. 具体而言, 参数服务器通过计算带有参数 $\alpha \in [0, 1]$ (例如, 在我们的实验设置 $\alpha = 0.8$) 的移动平均值, 来估计边缘节点 i 在第 h 轮的计算时间 μ_i^h 和相应的传输时间 β_i^h, 如下所示:

$$\mu_i^h = \alpha \cdot \mu_i^{h-1} + (1-\alpha) \cdot \hat{\mu}_i^h \tag{8}$$

$$\beta_i^h = \alpha \cdot \beta_i^{h-1} + (1-\alpha) \cdot \hat{\beta}_i^h \tag{9}$$

此外, 值得注意的是, 改进状态估计技术并非我们的重点, 其他现有的先进估计技术[43]也可以应用于 SplitCP.

边缘节点选择与配置: 基于估计的状态信息, 控制模块尝试选择并安排一部分边缘节点, 这些边缘节点被动态配置了合适特征压缩比与批处理大小进行特征融合的分裂联邦模型训练. 我们将本地更新频率记为 τ, 其为固定值, 并且在训练过程中对所有边缘节点均保持一致, 这与典型的分裂联邦学习框架 (如 LocFedMix-SL) 一致. 因此, 给定第 h 轮中边缘节点 i 的单位计算时间 μ_i^h 及其对应的传输时间 β_i^h, 我们可将该节点在特征压缩比为 $1-\gamma_i^h$ 和批处理大小为 d_i^h 时的持续时间 (包括计算与通信时间) 表示为:

$$t_i^h = \tau \cdot d_i^h \cdot (\mu_i^h + \gamma_i^h \cdot \beta_i^h) \tag{10}$$

因此, 当使用所有边缘节点进行训练时, 可以通过所有节点的持续时间获得第 h 轮的总体完成时间, 定义为: $t^h = \max\{t_i^h \mid \forall i \in [N]\}$, 即持续时间最长的边缘节点的执行时间. 于是, 第 h 轮中边缘节点 i 的等待时间可表示为 $t^h - t_i^h$. 相应地, 所有边缘节点的平均等待时间可表示为:

$$\mathcal{W}^h = \frac{1}{N} \sum_{i=1}^{N} (t^h - t_i^h) \tag{11}$$

为最小化平均等待时间, SplitCP 将调节所有边缘节点的特征压缩比与批处理大小, 以便使它们的执行时间保持一致. 这样可以确保平均等待时间足够小, 从而减轻同步屏障带来的负面影响, 并提高训练效率. 该调节规则可表示为:

$$\left\lfloor \frac{d_i^h \cdot (\mu_i^h + \gamma_i^h \cdot \beta_i^h)}{d_j^h \cdot (\mu_j^h + \gamma_j^h \cdot \beta_j^h)} \right\rfloor = 1, \quad \forall i \in [1,2,\ldots,N] \tag{12}$$

通过该规则, 可以统一联合决策所有边缘节点的特征压缩比与批处理大小. 为方便起见, 我们采用 top-k 稀疏化 (top-k sparsification) 作为压缩算子, 同时其他特征压缩算子 (例如随机-k 稀疏化、自编码器压缩器、随机化 top-k 稀疏化) 也可应用于 SplitCP 中.

由于第 h 轮可用入口带宽 B^h 的限制, 每一个通信轮都让所有边缘节点都参与训练实际上是不可行的. 因此, SplitCP 选择 R^h 个边缘节点并构建边缘节点集 S^h 用于特征融合和模型训练. PS 与 S^h 中边缘节点通信所占用的带宽受限如下:

$$\sum_{i \in S^h} d_i^h \cdot \gamma_i^h \cdot c \leqslant B^h \tag{13}$$

其中, c 是一个常数, 表示传输一个数据样本特征所占用的带宽.

为了解决统计异构挑战, 混合特征序列需要近似等同于从独立同分布数据派生出的特征. 我们首先将 IID 分布定义为 Φ_0. 如果所有边缘节点的数据都遵循独立同分布, 我们可以得到 $\Phi_0 = \frac{1}{N}\sum_{i=1}^{N} \mathbf{V}_i$, 其中 $\mathbf{V}_i$ 是边缘节点 i 的标签分布. 考虑到在第 h 轮中大小为 $R^h = \left|S^h\right|$ 的边缘节点集 S^h, 来自 S^h 中边缘节点的数据的标签分布表示为:

$$\Phi_h = \sum_{i \in S^h} \frac{d_i^h \cdot \mathbf{V}_i}{\sum_{i \in S^h} d_i^h} \tag{14}$$

用于特征融合的边缘节点集 S^h 的混合特征序列, 期望满足其标签分布 Φ_h 与独立同分布 Φ_0 近似一致的要求. 我们引入 KL 散度 $KL(\Phi_h \,\|\, \Phi_0)$ 来衡量 Φ_h 和 Φ_0 之间的差距, 如下所示:

$$KL(\Phi_h \,\|\, \Phi_0) = \sum_{j=1}^{M} \Phi_h(v_j) \log \frac{\Phi_h(v_j)}{\Phi_0(v_j)} \tag{15}$$

为了平衡所有边缘节点对模型训练的贡献, 我们定义了参与频率 K_i 来追踪边缘节点 i 参与训练的次数. 那么, 选择边缘节点 i 进行未来训练的优先级 p_i 表示如下:

$$p_i = \frac{\sum_{i=1}^{N} (K_i + 1)}{K_i + 1} \tag{16}$$

这表明参与频率较低的边缘节点将有较高的优先级被选中. 我们采用贪心增量搜索算法来构建边缘节点集合 S^h, 以在带宽与时间约束下逐步逼近最优解. 具体而言, 控制模块在每一轮中首先计算所有候选节点的计算与传输开销、标签分布差异度及其统计重要性, 然后根据单位资源消耗带来的 KL 散度下降量作为边际效益指标, 逐步挑选贡献最大的节点加入集合 S^h, 直到增益低于阈值或达到带宽限制. 实际运行中, 已选边缘节点集合 S^h 的混合标签分布与独立同分布之间仍可能存在差异. 为进一步降低这种非独立同分布偏差, 同时兼顾系统时延与通信效率, 我们在贪心搜索完成后引入了批处理大小与特征压缩比的联动启发式调整机制. 控制模块首先根据各节点的计算与传输能力, 利用时间均衡原则为每个节点分配初始批处理大小, 使各节点的本来训练时间尽量接近. 随后, 若整体带宽占用超过预算 B^h, 则按照节点的重要性优先级逐步提高统计贡献较小节点的压缩率 γ_i, 从而在保证关键节点信息保真的同时节省带宽资源. 最后, 对所有节点的批处理大小进行按比例微调, 使系统在满足带宽与时间约束的前提下, 实现更高的带宽利用率与更平衡的训练进度.

在上述节点选择与配置流程中, 各边缘节点的批处理大小、特征压缩率以及最终参与训练的节点规模, 并非需要人工预设的静态超参数, 而是系统基于当前资源状态动态求解的配置变量. 本节设计的贪心搜索与启发式联动机制, 在实际部署中能够实现参数的自适应调整. 具体而言, 参数服务器通过监控底层物理环境的变化, 当网络

可用带宽受限或部分节点算力下降时, 系统无需人工干预即可自动调高相应节点的特征压缩率或按比例缩小批处理大小; 同时, 单轮接入的节点数量也受限于实时可用带宽上限. 这种机制使 SplitCP 能够根据边缘计算环境的动态变化, 实时计算出合适的参数联动方案, 有效避免了人工调参的开销, 提升了框架在异构环境下的自适应能力. 此外, 该启发式联动调整机制还能有效应对系统波动与带宽估计误差. 在复杂的边缘网络中, 若因信道突变导致移动平均估计的带宽出现偏差, 该机制能通过对所有节点的压缩率和批处理大小进行等比例缩放, 确保在面临不可预见的网络波动时, 实际产生的通信量仍严格限制在物理带宽约束之内, 进一步增强了系统的容错性与跨设备稳定性.

此外, 在实际的大规模边缘计算部署中, 参数服务器自身也可能面临算力或内存瓶颈. 由于其需并发接收并缓存多节点的特征张量, 同时执行大批量的特征拼接与反向传播计算, 极易成为系统短板. 为提升框架在服务器硬件资源受限场景下的适应性, SplitCP 的边缘节点选择模块本质上充当了一种轻量化的调度机制. 当检测到参数服务器端负载过高时, 系统可通过动态调低贪心算法中的带宽预算或轮次时间阈值, 自适应地限制单轮并行参与训练的边缘节点数量 (即节点准入控制). 尽管这种截断策略因减小了等效批处理大小而在一定程度上影响了单轮收敛速度, 但考虑到参数服务器端的内存消耗峰值与并发节点数呈严格正相关, 主动限制节点规模能够有效控制计算并发量, 防止引发内存溢出或计算阻塞. 该机制在保障模型最终收敛精度的同时, 切实提升了 SplitCP 在实际受限硬件条件下的部署可行性.

3.3 训练模块

在控制模块生成边缘节点选择决策后, 它会将特征压缩比和批处理大小的配置分发给被选中的边缘节点, 这些配置将用于指导后续的模型训练过程. 此外, 参数服务器向被选中的边缘节点广播最新的底部模型并运行训练模块. 训练模块包括 4 个阶段, 即底部模型训练、特征融合、梯度分发和底部模型聚合.

底部模型训练: 对于第 h 轮中的某个边缘节点 i, 我们采用批处理大小为 d_i^h 的小批量随机梯度下降算法来更新底部模型. 为了保证模型收敛, 我们设置每个边缘节点 i 的本地学习率 η_i^h 与其批处理大小 d_i^h 成正比. 因此, 在第 h 轮中第 $k+1$ 次迭代更新底部模型的过程表示为:

$$\mathbf{w}_{b,i}^{h,k+1} = \mathbf{w}_{b,i}^{h,k} - \eta_i^h \cdot \tilde{\nabla} F_{b,i}(\mathbf{w}_{b,i}^{h,k}) \tag{17}$$

特征融合: 一旦边缘节点 i 在第 h 轮的第 k 次迭代执行前向传播, 它会将压缩后的特征 $\mathbf{g}_i^{h,k}$ 以及相应的标签发送到参数服务器. 在收到本轮训练边缘节点的特征后, 参数服务器将这些特征进行融合, 并获得一个混合特征序列, 该特征序列近似从一个独立同分布小批量数据派生出的特征. 我们将边缘节点集 S^h 中的从 i 到 j 的边缘节点的混合特征序列表示为 $\mathbf{G}^{h,k} = [\mathbf{g}_i^{h,k}, \ldots, \mathbf{g}_j^{h,k}]$, 其中 $|j-i| = R^h$. 因此, 参数服务器在第 h 轮的第 k 次迭代中使用混合特征序列 $\mathbf{G}^{h,k}$ 执行前向/反向传播以更新顶部模型, 如下所示:

$$\mathbf{w}_p^{h,k+1} = \mathbf{w}_p^{h,k} - \eta^h \cdot \tilde{\nabla} F_p(\mathbf{w}_p^{h,k}) \tag{18}$$

其中, $\tilde{\nabla} F_p(\mathbf{w}_p^{h,k}) = \tilde{\nabla} \ell(\mathbf{G}^{h,k}; \mathbf{w}_p^{h,k}) \Big/ \left(\sum_{i \in S^h} d_i^h\right)$, 且 η^h 是第 h 轮中顶部模型的学习率. 如果边缘节点只传递特征而不带标签, 参数服务器将返回顶部模型的输出 logits 给边缘节点, 以便在本地计算损失. 然后, 边缘节点将损失值发送回参数服务器, 以计算梯度并完成反向传播.

梯度分发: 在执行顶部模型的反向传播后, 参数服务器在第 h 轮的第 k 次迭代获得了混合的反向传播梯度 $\widehat{\mathbf{G}}^{h,k}$. 为了正确更新不同边缘节点的底部模型, 边缘节点有必要获得与其在特征融合阶段上传的特征相对应的梯度. 具体来说, 参数服务器首先将混合的大尺寸梯度 $\widehat{\mathbf{G}}^{h,k}$ 分割成多个小尺寸梯度 $[\widehat{\mathbf{g}}_i^{h,k}, \ldots, \widehat{\mathbf{g}}_j^{h,k}]$, 这些梯度对应于边缘节点集 S^h 中从边缘节点 i 到 j 的被选中边缘节点. 然后, 参数服务器根据上述公式将相应的梯度分派给相应的边缘节点, 以完成反向传播.

底部模型聚合: 在第 h 轮总共执行了 τ 次迭代后, 边缘节点集 S^h 中的被选中边缘节点将其底部模型推送到参数服务器进行集中聚合. 考虑到为模型训练配置了不同批处理大小的边缘节点, 其底部模型的更新和训练程度各不相同, 这需要自适应的权重聚合来保证聚合后底部模型的性能. 因此, 参数服务器采用与边缘节点的批处理大小

相关的自适应权重来聚合底部模型, 如下所示:

$$\mathbf{w}_b^h = \sum_{i \in S^h} \frac{d_i^h \cdot \mathbf{w}_{b,i}^h}{\sum_{i \in S^h} d_i^h} \tag{19}$$

聚合后的底部模型将由参数服务器分发给下一轮参与模型训练的边缘节点, 或者在模型收敛后与顶部模型结合形成一个完整的模型用于下游人工智能任务.

在上述训练流程中, 所提及的隐私优势主要侧重于模型隐私. 通过将完整模型拆分, 边缘节点与 PS 分别独立持有底部与顶部网络, 双方均无法获取彼此的完整模型结构与参数. 在数据隐私层面, 本文设定了诚实且好奇 (honest-but-curious) 的 PS 威胁模型. 在默认的高效训练流程中, PS 可见经过非线性变换及特征压缩后的稀疏特征与对应标签, 但不可见边缘节点的原始明文样本. 若针对数据隐私要求更为严格的场景, 系统可采用不上传标签的替代流程: PS 在完成前向传播后, 将输出层的 logits 回传给端侧节点; 端侧利用本地真实标签计算 Loss 并求导, 随后将针对 logits 的梯度回传至 PS 以完成反向传播. 此流程将标签信息保留在边缘侧, 其代价是每轮额外增加的通信量为两倍的节点批处理大小与总类别数之积. 由于分类类别数通常远小于特征维度, 该额外通信开销处于可接受范围内. 此外, SplitCP 中采用的特征压缩机制通过过滤掉部分前向激活值, 客观上增加了攻击者利用稀疏特征实施逆向重构攻击的难度, 为系统提供了基础的数据隐私保护.

4 实验分析

在本节中, 我们分别做了多组实验来验证所提框架 SplitCP 的有效性和高效性. 我们首先介绍一下实验配置, 之后对实验结果进行展示与分析.

4.1 实验设置

实验环境: 我们在一个边缘计算硬件原型系统上进行了广泛的实验, 以评估 SplitCP 的性能. 具体来说, 我们采用一台深度学习 GPU 工作站作为参数服务器, 该工作站配备了 Intel(R) Core(TM) i9-10900X CPU、4 块 NVIDIA GeForce RTX 2080Ti GPU 和 256 GB RAM. 此外, 我们指定了 80 台 NVIDIA Jetson 套件, 包括 30 台 Jetson TX2 设备、40 台 Jetson NX 设备和 10 台 Jetson AGX 设备, 作为边缘节点来构建一个异构系统. 值得注意的是, TX2 配备了 256 核的 Pascal GPU 和一个由 2 核 Denver2 与 4 核 ARM Cortex A57 组成的 CPU 集群. NX 则配备了 384 核的 NVIDIA Volta GPU 和一个 6 核的 NVIDIA Carmel ARMv8.2 CPU. Jetson Xavier NX 在 NVIDIA 软件栈上的性能比 Jetson TX2 提升了 10 倍以上. 最后, AGX 以其 512 核的 NVIDIA Volta GPU 和一个 8 核的 NVIDIA Carmel ARMv8.2 CPU 脱颖而出. 在实验中, 我们基于 Docker Swarm 和 PyTorch 深度学习库[44]搭建了软件平台. Docker Swarm 是一个分布式软件开发套件, 它有助于构建分布式系统, 并能够监控每个设备的运行状态. PyTorch 库则便于在设备上实现模型训练. 此外, 为了简化设备间的通信, 我们实现了消息传递接口 MPI (message passing interface), 其中包括一系列发送和接收函数.

数据集和模型: 我们在 4 个经典数据集和 4 个深度神经网络 (deep neural network, DNN) 模型上评估了 SplitCP 的性能.

(1) 人体活动识别应用: 在此应用中, 我们采用了人体活动识别 (HAR) 数据集, 该数据集从 30 位个体中收集, 包含 7 352 个训练样本和 2 947 个测试样本. 我们训练了一个普通的 CNN 模型, 该模型包含 3 个 5×5 卷积层和 2 个全连接层, 专为 HAR 数据集定制, 表示为 CNN-H.

(2) 语音识别应用: 我们采用谷歌语音数据集 (简称 Speech)[45]进行语音识别任务, 该任务允许计算机或设备识别和解释口语. 该数据集包含 85 511 个训练音频片段和 4 890 个测试音频片段. 在 Speech 数据集上训练的模型是一个 CNN 网络 (表示为 CNN-S), 包含 4 个一维卷积层和 1 个全连接层.

(3) 物品识别应用: 我们采用 CIFAR-10 数据集进行评估, 这是一个图像数据集, 由 10 个类别的 60 000 张 32×32 彩色图像组成 (50 000 用于训练, 10 000 用于测试). 我们为 CIFAR-10 使用了一个大小为 136 MB 的 8 层 AlexNet. AlexNet 由 3 个 3×3 卷积层、1 个 7×7 卷积层、1 个 11×11 卷积层、2 个全连接隐藏层和 1 个 Softmax

输出层组成.

(4) 图像分类应用: ImageNet 是一个用于图像识别的数据集, 包含来自 1 000 个类别的 1 281 167 张训练图像、50 000 张验证图像和 100 000 张测试图像. 为了适应资源受限的边缘节点, 我们创建了 ImageNet 的一个子集, 称为 IMAGE-100, 它包含 1 000 个类别中的 100 个, 并且每个样本都被调整为 64×64×3 的形状. 对于这个最复杂的任务, 我们采用了一个著名的、大小为 321 MB 的大型模型 VGG16, 它比 AlexNet 大得多, 用于对 IMAGE-100 中的图像进行分类. VGG16 由 13 个 3×3 卷积核的卷积层、2 个全连接层和 1 个 Softmax 输出层组成.

系统异构性设置: 为了使边缘节点具有异构的计算和通信能力, 我们采用了以下实验设置.

(1) 计算能力方面: 所有的 Jetson TX2、NX 和 AGX 都可以配置为在不同模式下工作, 这些模式指定了工作 CPU 的数量和 CPU/GPU 的频率, 从而使它们能以不同的计算能力工作. 具体来说, TX2 可以在 4 种模式中的一种工作, 而 NX 和 AGX 则各有 8 种工作模式. 例如, 处于最高性能模式的 AGX (即 AGX 的模式 0) 的训练速度比处于最低性能模式的 TX2 (即 TX2 的模式 1) 快 100 倍. 为了进一步反映设备上资源的动态变化, 我们每 20 个通信轮次为设备随机更改一次模式.

(2) 通信能力方面: 所有设备通过 WiFi 路由器连接到 PS. 我们将设备分成 4 组, 每组 20 台. 然后将这些组放置在距离 WiFi 路由器不同距离的位置, 即 2 m、8 m、14 m 和 20 m. 由于信道噪声的随机性和设备间的竞争, PS 和设备之间的带宽在训练过程中会动态变化. 设备的带宽通过 iperf3 进行测量, 其波动范围在 1–30 Mb/s 之间.

统计异构性设置: 在实验中, 每个边缘节点的训练样本都是通过一个向量 v 独立抽取的. 为了创建非独立同分布 (non-IID) 数据集, 我们从狄利克雷分布 (Dirichlet distribution)[46]中抽样, 即 $v \sim Dir(\delta q)$, 其中 q 代表先验类别分布, $\delta > 0$ 是一个控制边缘节点间同质性的集中度参数. 当 $\delta \to \infty$ 时, 所有边缘节点都具有与先验类别分布相同的分布 (即 IID); 当 $\delta \to 0$ 时, 每个边缘节点只持有来自一个类别的样本, 这表示高度的统计异构性. 我们为 δ 指定了 6 个值 (例如 ∞, 1, 0.5, 0.25, 0.2, 0.1) 以生成覆盖不同同质性谱的数据分布, 并定义 $p = 1/\delta$ (即 $p = 0, 1, 2, 4, 5, 10$) 来量化 non-IID 水平. 统计异构性的程度随着 p 的增加而增加, $p = 0$ 是 IID 的特例.

基线方法: 我们通过与 4 个基线模型的比较来衡量 SplitCP 的有效性.

(1) LocFedMix-SL[23]是一种典型且先进的 SFL 方法. 它提出降低底部模型的聚合频率以节省流量消耗, 但无法充分利用异构边缘节点的能力.

(2) AdaSFL[10]是一种最先进的 SFL 方法. 它为不同的边缘节点分配自适应且多样的批处理大小以解决系统异构性问题, 但仍无法处理统计异构性.

(3) PyramidFL[16]是一种具有细粒度边缘节点选择的最先进 FL 方法. 它关注已选择和未选择边缘节点之间的差异, 以充分利用不同边缘节点的计算资源和数据.

(4) GAS[47]是先进的异步分裂联邦学习方法, 引入了生成式激活辅助更新机制, 利用生成模块平衡来自不同节点的中间特征分布解决数据异构问题.

评估指标: 我们采用以下指标来评估 SplitCP 和基线模型的性能.

(1) 模型精度: 该指标反映了不同方法训练的模型在测试数据集上的准确性, 通过模型正确预测的数据占所有测试数据的比例来衡量. 具体来说, 我们评估每个轮次中全局模型 (在 SFL 中是底部和顶部模型的组合) 的模型精度, 并记录不同方法的最终模型精度.

(2) 训练时间: 该指标表示训练一个模型以达到目标准确率所需的总时间. 为了公平比较, 我们将目标准确率设置为所有方法都能达到的准确率. 我们记录每个轮次的完成时间并累加得到总训练时间. 此外, 我们还记录平均等待时间以反映不同方法的训练效率.

(3) 网络流量: 该指标计算为在达到目标准确率时, PS 与边缘节点之间传输模型或特征/梯度所产生的流量总和.

参数设置: 默认情况下, 每组实验将为 CNN-H 运行 150 个通信轮次, 为 CNN-S、AlexNet 和 VGG16 运行 250 个通信轮次. 对于 CNN-H, 学习率初始化为 0.1, 衰减率指定为 0.98. CNN-S、AlexNet 和 VGG16 的学习率和衰减率相同, 分别初始化为 0.1 和 0.993. 我们设置 CNN-H 的本地更新频率 $\tau = 10$, CNN-S 和 AlexNet 的 $\tau = 30$, VGG16 的 $\tau = 40$. 对于 SFL 方法, 我们分别在第 3、4、5 和 13 层对 CNN-H、CNN-S、AlexNet 和 VGG16 进行切分.

4.2 实验结果

总体性能: 首先, 我们在 IID 数据集上进行了一系列实验, 以评估 SplitCP 和基线模型的性能. 这些方法的训练过程如图 6 所示. 从结果来看, 所有方法最终都达到了相似的模型精度. 然而, SplitCP 实现了最快的收敛速度, 在所有 4 个数据集上都显著优于其他方法. 例如, 根据图 6(a), 在 HAR 数据集上, SplitCP 用 981 s 使 CNN-H 达到 87% 的准确率, 而 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 分别耗时 1 324 s、1 946 s、1 471 s 和 2 939 s. 根据图 6(b), 在 Speech 数据集上, SplitCP 在总完成时间方面也优于其他方法. 此外, 图 6(c) 显示, 在 CIFAR-10 数据集上, 与 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 方法相比, SplitCP 将 AlexNet 模型训练总时间分别减少了约 21%、48%、39% 和 62%. 此外, 对于在 IMAGE-100 数据集上训练 VGG16 模型, 如图 6(d) 所示, 与 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 相比, SplitCP 的训练速度分别提升了约 1.31 倍、2.05 倍、1.65 倍和 2.9 倍. 这些结果证明了 SplitCP 在解决系统异构挑战的优越性, 可以提高模型训练效率, 减少模型训练完成时间.

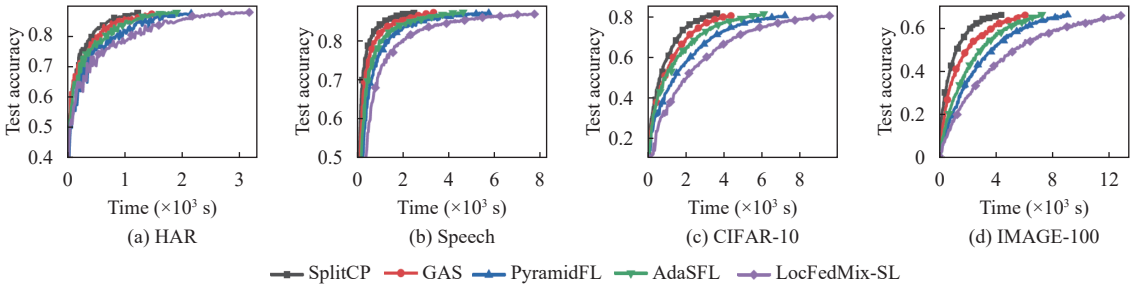


图 6 在 4 个独立同分布数据集上 5 种方法的测试精度

其次, 我们还在非独立同分布程度 $p = 10$ 的所有数据集上对这些方法进行了一系列实验, 结果如图 7 所示. 我们观察到, SplitCP 保持了与独立同分布数据相似的收敛速度, 并在这些方法中实现了最高的模型精度. 例如, 根据图 7(a), 在 HAR 数据集上, SplitCP 在 1 227 s 内使 CNN-H 达到了 86.91% 的准确率, 而 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 分别需要 1 471 s、2 207 s、1 887 s 和 3 154 s 才能达到 81.42%、79.44%、72.53% 和 72.38% 的准确率. 同样, 图 7(b) 表明, 在 Speech 数据集上, 与 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 相比, SplitCP 对 CNN-S 的最终模型精度分别提升了约 6.46%、7.02%、25.69% 和 26.15%. 此外, 根据图 7(c), 在 CIFAR-10 上训练 AlexNet 模型达到约 60% 的模型精度时, 与 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 相比, SplitCP 将总完成时间分别减少了约 43%、71%、76% 和 87%. 此外, 如图 7(d) 所示, 在 IMAGE-100 数据集上训练 VGG16 模型, 在相同的 4 000 s 训练时间内, 与 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL 相比, SplitCP 的模型精度分别提升了约 16.73%、31.72%、36.05% 和 43.68%. 这些结果表明, SplitCP 通过联合自适应特征压缩和批处理大小调节的特征融合技术, 能有效同时应对异构性挑战.

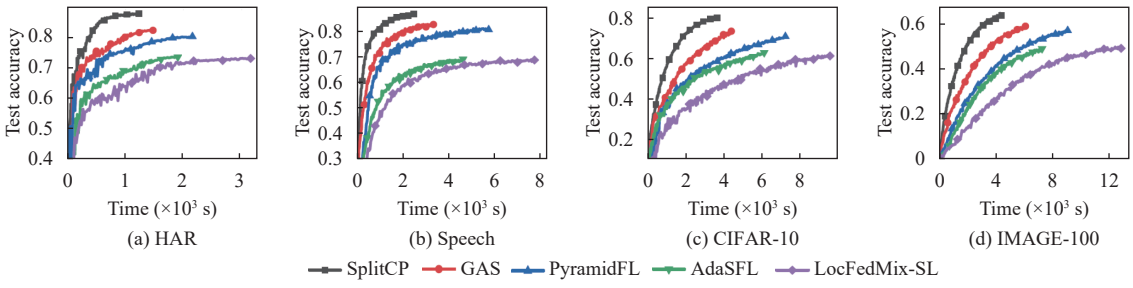


图 7 在 4 个非独立同分布数据集上 5 种方法的测试精度

此外, 为了说明 SplitCP 在节省通信资源方面的优势, 我们在图 8 中展示了这些方法在达到不同目标准确率时的网络流量消耗. 从结果来看, 对于所有 4 个数据集, 所有方法的网络流量消耗都随着目标准确率的提高而增加.

此外, 在所有方法中, SplitCP 始终消耗最少的网络流量. 此外, 与典型的 FL 方法 (即 PyramidFL) 相比, 模型分裂 (即 SplitCP、GAS 和 AdaSFL) 有助于节省更多的网络流量. 具有自适应本地更新频率的 AdaSFL 减少了网络流量消耗, 而具有自适应特征压缩的 SplitCP 进一步减少了网络流量消耗. 如图 8(b) 所示, 在 Speech 数据集上训练 CNN-S 达到 87% 的准确率时, SplitCP、GAS、AdaSFL 和 LocFedMix-SL 分别消耗 877 MB、1 992 MB、1 694 MB 和 2 398 MB 的流量, 而 PyramidFL 消耗 3 397 MB. 此外, 如图 8(d) 所示, 在 IMAGE-100 上训练 VGG16 达到 65% 的准确率时, 与基线模型 (即 GAS、PyramidFL、AdaSFL 和 LocFedMix-SL) 相比, SplitCP 节省了约 59%、73%、55% 和 68% 的网络流量消耗.

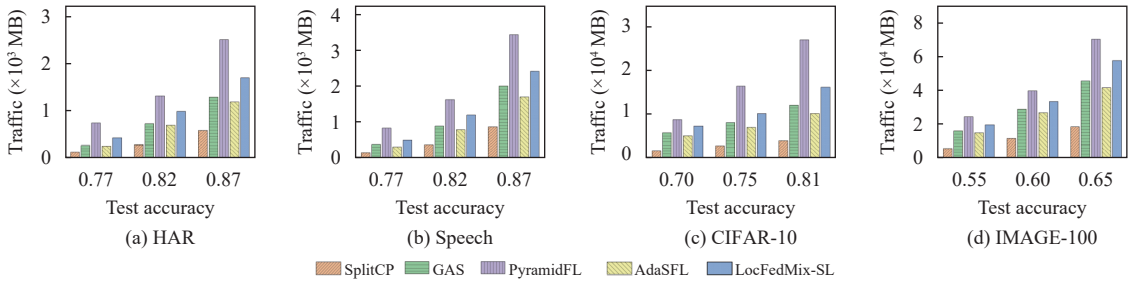


图 8　5 种方法在达到不同目标精度时的网络通信流量消耗

为了进一步证明 SplitCP 对系统异构性的鲁棒性, 我们在图 9 中展示了 5 种方法在 4 个数据集上的平均等待时间. 具有自适应且多样化批处理大小以适应异构边缘节点的 AdaSFL 实现了最短的等待时间, 但 SplitCP 的等待时间接近 AdaSFL 和 GAS, 并且远小于其他方法. 例如, 根据图 9(a), 在 HAR 数据集上, SplitCP 对 CNN-H 的平均等待时间与 AdaSFL 以及 GAS 均为 1.1 s, 而 PyramidFL 和 LocFedMix-SL 的平均等待时间分别为 3.4 s 和 5.9 s. 考虑到边缘节点的能力各不相同, LocFedMix-SL 在模型训练时使用固定且相同的批处理大小, 并且没有进行模型压缩, 没有考虑系统异构性, 因此导致了不可忽略的等待时间. PyramidFL 通过自适应边缘节点选择以充分利用不同边缘节点的计算资源和数据, 在一定程度上减少了平均等待时间. 具体来说, 根据图 9(d), 与 PyramidFL 和 LocFedMix-SL 相比, SplitCP 可以将 VGG16 在 IMAGE-100 上的平均训练等待时间分别减少约 67% 和 80%.

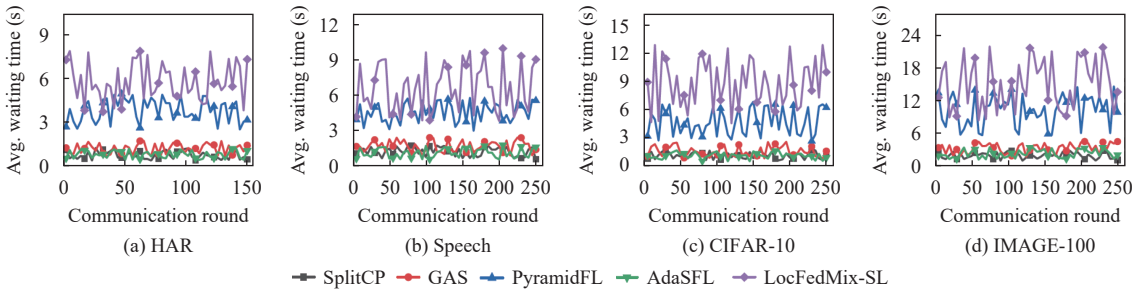


图 9　在 4 个数据集上 5 种方法的平均等待时间

non-IID 水平的影响: 为了证明 SplitCP 在处理非独立同分布数据方面的有效性, 我们在图 10 中展示了不同方法在不同非独立同分布程度下的模型精度, 其中横轴表示数据集的非独立同分布程度. 如图 10 所示, 5 种方法在所有数据集上训练的模型的模型精度都随着非独立同分布程度的增加而下降. 然而, SplitCP 在所有数据集上始终优于其他方法. 未考虑系统和统计异构性挑战的 LocFedMix-SL 和 AdaSFL, 在非独立同分布数据集上表现出最低的模型准确率. PyramidFL 专注于已选择和剩余边缘节点之间的差异, 以充分利用不同边缘节点的计算资源和数据, 可以在一定程度上减轻非独立同分布数据对模型训练的影响. GAS 允许在服务器端对模型进行集中更新以显著减轻由数据异构性引起的深度模型漂移, 并且通过维持每个标签的激活分布并生成激活值来保证各标签数据的平衡, 再结合 logits 调整技术校准损失函数, 从而有效缓解了由非独立同分布数据带来的更新偏差, 因此其模型精

度表现优于 PyramidFL, 且随数据异构性加深时的性能下降相对较缓. 具体来说, 根据图 10(a), 在 HAR 数据集上非独立同分布程度为 $p = 10$ 时, SplitCP、GAS 和 PyramidFL 分别达到了 86.91%、81.42% 和 79.44% 的准确率, 而 LocFedMix-SL 和 AdaSFL 仅达到 72.53% 和 72.38% 的准确率. 此外, 如图 10(b) 所示, 在谷歌语音数据集上从独立同分布过渡到非独立同分布程度 $p = 10$ 时, SplitCP、GAS 和 PyramidFL 的准确率仅分别下降了 1.72%、5.96% 和 7.23%, 而 AdaSFL 和 LocFedMix-SL 的准确率损失分别为 18.54% 和 18.74%. 值得注意的是, 根据图 10(c), 在 CIFAR-10 数据集上非独立同分布程度为 $p = 10$ 时, 与基线模型 (即 GAS、PyramidFL、AdaSFL、LocFedMix-SL) 相比, SplitCP 的模型精度分别提升了约 8.18%、12.92%、27.45%、28.49%. 这些结果进一步证明了 SplitCP 在克服统计异构方面的有效性, 同时也与之前总体性能比较的实验结果互相印证.

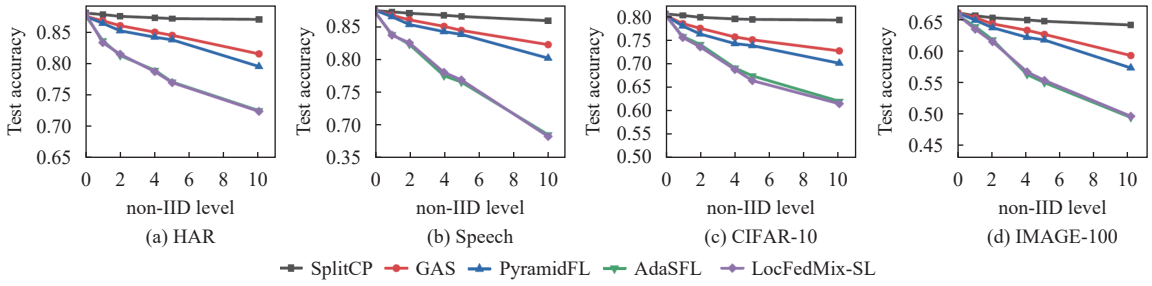
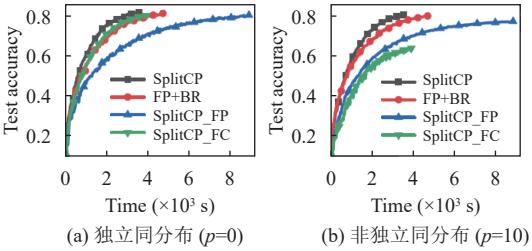


图 10 模型精度随数据非独立同分布程度变化的情况

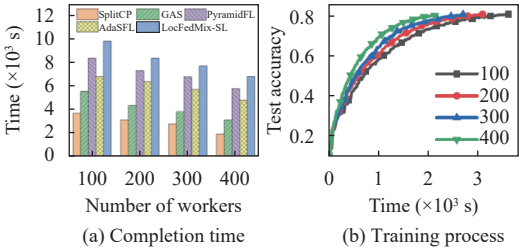
关键策略的影响: SplitCP 包含特征融合关键技术并开发了自适应特征压缩与批处理大小调控两项关键策略以提升分裂联邦学习的性能. 在此, 我们针对 AlexNet 在 CIFAR-10 上的独立同分布 ($p = 0$) 和非独立同分布 ($p = 10$) 进行了几组实验, 以评估特征融合、自适应特征压缩与批处理大小调控的有效性. 我们采用联合特征融合与批处理大小调控的方法 (表示为 FP+BR)、只进行特征融合的 SplitCP 方法 (表示为 SplitCP_FP)、只进行自适应特征压缩的 SplitCP 方法 (表示为 SplitCP_FC) 作为基线方法来进一步体现特征融合以及另外两个关键策略在 SplitCP 中的重要性. 具体来说, 在 SplitCP_FC 中, 参数服务器直接应用具有相同批处理大小的边缘节点的自适应压缩后的特征来分别执行前向/反向传播, 而不进行特征融合. 而在 SplitCP_FP 中, 所有边缘节点不进行特征压缩, 并都被分配了相同的批处理大小, 即 SplitCP 中批处理大小的平均值, 用于特征融合和模型训练. 根据图 11, 在独立同分布数据集上, FP+BR 的收敛速度慢于 SplitCP_FC, 但 SplitCP_FC 的模型收敛精度略低于 FP+BR, 这是因为节点间过大的压缩比差异会直接影响模型训练的精度. 没有针对异构节点进行优化的 SplitCP_FP 方法实现了最慢的模型收敛速度. 而在非独立同分布数据集上, SplitCP_FC 的精度损失更加严重, 而 SplitCP_FP 虽然收敛慢, 但可以实现高精度. 并且 FP+BR 可以同时克服统计异构和系统异构的影响, 收敛快的同时实现高精度. 而在特征融合、自适应特征压缩与批处理大小调控的共同作用下, 相比于 FP+BR、SplitCP_FP 与 SplitCP_FC, SplitCP 可以提升训练速度约 1.32 倍、2.48 倍与 1.11 倍, 并提升模型精度约 0.53%、4.07% 与 25.69%. 这些结果进一步体现了 SplitCP 方法在克服统计异构与系统异构方面的有效性.

系统规模的影响: 为了证明 SplitCP 的鲁棒性, 我们评估了 SplitCP 和基线模型在不同规模的参与边缘节点下的性能. 我们通过广泛的仿真实验, 在 4 种规模的参与模型训练节点总数下 (即 100、200、300、400) 在 CIFAR-10 数据集训练 AlexNet 模型. 这些实验在一台 AMAX 深度学习工作站上进行, 该工作站配备了 Intel(R) Xeon(R) Gold 5218R CPU、8 块 NVIDIA GeForce RTX 3090 GPU 和 256 GB 内存. 这些方法达到 80% 准确率的完成时间结果如图 12(a) 所示, 而 SplitCP 在不同规模下的训练过程如图 12(b) 所示. 随着参与边缘节点数量的增加, 所有方法都实现了更快的收敛. 例如, 与 100、200 和 300 个边缘节点的 SplitCP 相比, 拥有 400 个边缘节点的 SplitCP 分别实现了 1.69 倍、1.33 倍和 1.17 倍的加速. 原因是单个边缘节点上的样本数量有限, 而更多的边缘节点在每个轮次中为训练贡献更多的本地数据. 此外, 在不同规模的边缘节点下, 与基线模型 (即 GAS、PyramidFL、AdaSFL、LocFedMix-SL) 相比, SplitCP 达到目标准确率的速度快了 1.52–3.51 倍. 这些实验结果进一步说明了 SplitCP 方法的高效性与鲁棒性.



(a) 独立同分布 (p=0) (b) 非独立同分布 (p=10)

图 11 特征融合与批处理大小调节的效果



(a) Completion time (b) Training process

图 12 不同节点数量下的性能比较

5 总 结

在本文中, 我们设计并实现了一种新的分裂联邦学习框架 SplitCP, 该框架引入了特征融合、自适应特征压缩与批处理大小调控机制, 以应对系统异构性和统计异构性问题. 通过为异构边缘节点联合配置合适的特征压缩比与批处理大小, 并将各节点的特征融合为混合特征序列, SplitCP 能够提升 SFL 的模型精度和训练效率. 实验结果表明, SplitCP 在性能上显著优于基线方法, 实现了 1.31–3.51 倍的加速比, 并使最终模型精度提升了 6.46%–26.15%.

References

[1] Liu M, Xia YH, Zhao HT, Guo L, Shi Z, Zhu HB. Federated learning technologies for 6G industrial Internet of Things: From requirements, vision to challenges, opportunities. Journal of Electronics & Information Technology, 2024, 46(12): 4335–4353 (in Chinese with English abstract). [doi: 10.11999/JEIT240574]

[2] Provatas N, Konstantinou I, Koziris N. A survey on parameter server architecture: Approaches for optimizing distributed centralized learning. IEEE Access, 2025, 13: 30993–31015. [doi: 10.1109/ACCESS.2025.3535085]

[3] Li ZS, Xie RC, Sun L, Huang T. A survey of mobile edge computing. Telecommunications Science, 2018, 34(1): 2018011 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-0801.2018011]

[4] Yang T, Andrew G, Eichner H, Sun HC, Li W, Kong N, Ramage D, Beaufays F. Applied federated learning: Improving Google keyboard query suggestions. arXiv:1812.02903, 2018.

[5] Zhang WY, He ZZ, Liu LY, Jia ZH, Liu YX, Gruteser M, Raychaudhuri D, Zhang YY. Elf: Accelerate high-resolution mobile deep vision with content-aware parallel offloading. In: Proc. of the 27th Annual Int'l Conf. on Mobile Computing and Networking. New Orleans: ACM, 2021. 201–214. [doi: 10.1145/3447993.3448628]

[6] Pal S, Uniyal M, Park J, Vepakomma P, Raskar R, Bennis M, Jeon M, Choi J. Server-side local gradient averaging and learning rate acceleration for scalable split learning. arXiv:2112.05929, 2021.

[7] Zhao ZM, Liu F, Cai ZP, Xiao N. Edge computing: Platforms, applications and challenges. Journal of Computer Research and Development, 2018, 55(2): 327–337 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2018.20170228]

[8] Lin Z, Qu GQ, Wei W, Chen XH, Leung KK. AdaptSFL: Adaptive split federated learning in resource-constrained edge networks. IEEE Trans. on Networking, 2025, 33(6): 2993–3008. [doi: 10.1109/TON.2025.3577790]

[9] Lin Z, Wei W, Chen Z, Lam CT, Chen XH, Gao Y, Luo J. Hierarchical split federated learning: Convergence analysis and system optimization. IEEE Trans. on Mobile Computing, 2025, 24(10): 9352–9367. [doi: 10.1109/TMC.2025.3565509]

[10] Liao YM, Xu Y, Xu HL, Yao ZW, Wang L, Qiao CM. Accelerating federated learning with data and model parallelism in edge computing. IEEE/ACM Trans. on Networking, 2024, 32(1): 904–918. [doi: 10.1109/TNET.2023.3299851]

[11] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego: ICLR, 2015. [doi: 10.48550/arXiv.1409.1556]

[12] Wu MQ, Cheng GL, Ye DD, Kang JW, Yu R, Wu Y, Pan M. Federated split learning with data and label privacy preservation in vehicular networks. IEEE Trans. on Vehicular Technology, 2024, 73(1): 1223–1238. [doi: 10.1109/TVT.2023.3304176]

[13] Yang ZY, Chen YY, Huangfu HJ, Ran MS, Wang H, Li XX, Zhang Y. Dynamic corrected split federated learning with homomorphic encryption for U-shaped medical image networks. IEEE Journal of Biomedical and Health Informatics, 2023, 27(12): 5946–5957. [doi: 10.1109/JBHI.2023.3317632]

[14] Xie YX, Wang Z, Gao DW, Chen DY, Yao LY, Kuang WR, Li YL, Ding BL, Zhou JR. FederatedScope: A flexible federated learning platform for heterogeneity. Proc. of the VLDB Endowment, 2023, 16(5): 1059–1072. [doi: 10.14778/3579075.3579081]

[15] Liao YM, Xu Y, Xu HL, Wang L, Qian C. Adaptive configuration for heterogeneous participants in decentralized federated learning. In: Proc. of the 2023 IEEE Conf. on Computer Communications. New York City: IEEE, 2023. 1–10. [doi: 10.1109/INFOCOM53939.2023.10228945]

[16] Li CN, Zeng X, Zhang M, Cao ZC. PyramidFL: A fine-grained client selection framework for efficient federated learning. In: Proc. of the 28th Annual Int'l Conf. on Mobile Computing and Networking. Sydney: ACM, 2022. 158–171. [doi: 10.1145/3495243.3517017]

[17] Chen S, Xu Y, Xu HL, Jiang ZD, Qiao CM. Decentralized federated learning with intermediate results in mobile edge computing. IEEE Trans. on Mobile Computing, 2024, 23(1): 341–358. [doi: 10.1109/TMC.2022.3221212]

[18] George AS, George ASH, Baskar T. Edge computing and the future of cloud computing: A survey of industry perspectives and predictions. Partners Universal Int'l Research Journal, 2023, 2(2): 19–44. [doi: 10.5281/zenodo.8020101]

[19] Xiao WJ, Hao YX, Liang JB, Hu L, Alqahtani SA, Chen M. Adaptive compression offloading and resource allocation for edge vision computing. IEEE Trans. on Cognitive Communications and Networking, 2024, 10(6): 2357–2369. [doi: 10.1109/TCCN.2024.3400820]

[20] Zhang M, Qu LQ, Singh P, Kalpathy-Cramer J, Rubin DL. SplitAVG: A heterogeneity-aware federated deep learning method for medical imaging. IEEE Journal of Biomedical and Health Informatics, 2022, 26(9): 4635–4644. [doi: 10.1109/JBHI.2022.3185956]

[21] Thapa C, Arachchige PCM, Camtepe S, Sun LC. SplitFed: When federated learning meets split learning. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. Palo Alto: AAAI Press, 2022. 8485–8493. [doi: 10.1609/aaai.v36i8.20825]

[22] Han DJ, Bhatti HI, Lee J, Moon J. Accelerating federated learning with split learning on locally generated losses. In: Proc. of the 2021 Int'l Workshop on Federated Learning for User Privacy and Data Confidentiality. ICML, 2021.

[23] Oh S, Park J, Vepakomma P, Baek S, Raskar R, Bennis M, Kim SL. LocFedMix-SL: Localize, federate, and mix for improved scalability, convergence, and latency in split learning. In: Proc. of the 2022 ACM Web Conf. Lyon: ACM, 2022. 3347–3357. [doi: 10.1145/3485447.3512153]

[24] Liao YM, Xu Y, Xu HL, Wang L, Yao ZW, Qiao CM. MergeSFL: Split federated learning with feature merging and batch size regulation. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering (ICDE). Utrecht: IEEE, 2024. 2054–2067. [doi: 10.1109/ICDE60146.2024.00164]

[25] Shin J, Li YC, Liu YX, Lee SJ. FedBalancer: Data and pace control for efficient federated learning on heterogeneous clients. In: Proc. of the 20th Annual Int'l Conf. on Mobile Systems, Applications and Services. Portland: ACM, 2022. 436–449. [doi: 10.1145/3498361.3538917]

[26] Xu Y, Liao YM, Xu HL, Ma ZG, Wang L, Liu JC. Adaptive control of local updating and model compression for efficient federated learning. IEEE Trans. on Mobile Computing, 2023, 22(10): 5675–5689. [doi: 10.1109/TMC.2022.3186936]

[27] Ma ZG, Xu Y, Xu HL, Meng ZY, Huang LS, Xue YX. Adaptive batch size for federated learning in resource-constrained edge computing. IEEE Trans. on Mobile Computing, 2023, 22(1): 37–53. [doi: 10.1109/TMC.2021.3075291]

[28] Li AR, Zhang L, Tan JT, Qin YX, Wang JH, Li XY. Sample-level data selection for federated learning. In: Proc. of the 2021 IEEE Conf. on Computer Communications. Vancouver: IEEE, 2021. 1–10. [doi: 10.1109/INFOCOM42981.2021.9488723]

[29] Lai F, Zhu XF, Madhyastha HV, Chowdhury M. Oort: Efficient federated learning via guided participant selection. In: Proc. of the 15th USENIX Symp. on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2021. 19–35.

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

[31] Ficili I, Giacobbe M, Tricomi G, Puliafito A. From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI. Sensors, 2025, 25(6): 1763. [doi: 10.3390/s25061763]

[32] 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. Fort Lauderdale: PMLR, 2017. 1273–1282.

[33] 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]

[34] Hafi H, Brik B, Frangoudis PA, Ksentini A, Bagaa M. Split federated learning for 6G enabled-networks: Requirements, challenges, and future directions. IEEE Access, 2024, 12: 9890–9930. [doi: 10.1109/ACCESS.2024.3351600]

[35] Shen JL, Cheng N, Wang XC, Lyu F, Xu WC, Liu Z, Aldubaikhy K, Shen XM. RingSFL: An adaptive split federated learning towards taming client heterogeneity. IEEE Transactions on Mobile Computing, 2024, 23(5): 5462–5478. [doi: 10.1109/TMC.2023.3309633]

[36] Lu ZL, Pan H, Dai YY, Si XM, Zhang Y. Federated learning with non-IID data: A survey. IEEE Internet of Things Journal, 2024, 11(11): 19188–19209. [doi: 10.1109/JIOT.2024.3376548]

[37] Nasif AS, Othman ZA, Sani NS, Hasan MK, Abudaqqa Y. Huffman deep compression of edge node data for reducing IoT network traffic. IEEE Access, 2024, 12: 122988–122997. [doi: 10.1109/ACCESS.2024.3452669]

[38] Shi D, Li L, Wu MQ, Shu ML, Yu R, Pan M, Han Z. To talk or to work: Dynamic batch sizes assisted time efficient federated learning over future mobile edge devices. IEEE Trans. on Wireless Communications, 2022, 21(12): 11038–11050. [doi: 10.1109/TWC.2022.3189320]

[39] Zhu HW, Guo ZH, Ye MH. DINA: Toward determined in-network aggregation for distributed machine learning. IEEE Trans. on Networking, 2025, 33(4): 1454–1468. [doi: 10.1109/TON.2025.3535707]

[40] Liu XL, Deng YS, Mahmoodi T. Wireless distributed learning: A new hybrid split and federated learning approach. IEEE Trans. on Wireless Communications, 2023, 22(4): 2650–2665. [doi: 10.1109/TWC.2022.3213411]

[41] Aziz R, Banerjee S, Bouzefrane S, Le Vinh T. Exploring homomorphic encryption and differential privacy techniques towards secure federated learning paradigm. Future Internet, 2023, 15(9): 310. [doi: 10.3390/fi15090310]

[42] Leroy D, Coucke A, Lavril T, Gisselbrecht T, Dureau J. Federated learning for keyword spotting. In: Proc. of the 2019 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). Brighton: IEEE, 2019. 6341–6345. [doi: 10.1109/ICASSP.2019.8683546]

[43] Yue CQ, Jin RF, Suh K, Qin YY, Wang B, Wei W. LinkForecast: Cellular link bandwidth prediction in LTE networks. IEEE Trans. on Mobile Computing, 2018, 17(7): 1582–1594. [doi: 10.1109/TMC.2017.2756937]

[44] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin ZM, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai JJ, Chintala S. 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. 721.

[45] Warden P. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv:1804.03209, 2018.

[46] Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaeni Y. Bayesian nonparametric federated learning of neural networks. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 7252–7261.

[47] Yang JR, Liu Y. GAS: Generative activation-aided asynchronous split federated learning. In: Proc. of the 39th AAAI Conf. on Artificial Intelligence. Philadelphia: AAAI Press, 2025. 21956–21964. [doi: 10.1609/aaai.v39i20.35503]

附中文参考文献

[1] 刘淼, 夏雨虹, 赵海涛, 郭亮, 施政, 朱洪波. 面向 6G 工业物联网的联邦学习: 从需求、愿景到挑战、机遇. 电子与信息学报, 2024, 46(12): 4335–4353. [doi: 10.11999/JEIT240574]

[3] 李子姝, 谢人超, 孙礼, 黄韬. 移动边缘计算综述. 电信科学, 2018, 34(1): 2018011. [doi: 10.11959/j.issn.1000-0801.2018011]

[7] 赵梓铭, 刘芳, 蔡志平, 肖侬. 边缘计算: 平台、应用与挑战. 计算机研究与发展, 2018, 55(2): 327–337. [doi: 10.7544/issn1000-1239.2018.20170228]

作者简介

廖云铭, 博士, CCF 专业会员, 主要研究领域为物联网, 边缘智能, 联邦学习.

马欣莹, 硕士生, 主要研究领域为边缘智能, 端云协同推理.

许杨, 博士, 副教授, CCF 专业会员, 主要研究领域为物联网, 边缘智能.

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.

黄刘生, 博士, 教授, 博士生导师, 主要研究领域为网络, 信息安全, 云计算, 大数据.