软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007266] [CSTR: 32375.14.jos.007266]

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

云边联邦学习系统下抗投毒攻击的防御方法*

赵亚茹[1,2], 张建标[1,2], 曹益皓[1,2], 黄浩翔[1,2]

[1](北京工业大学 计算机学院, 北京 100124)

[2](可信计算北京市重点实验室 (北京工业大学), 北京 100124)

通信作者: 张建标, E-mail: zjb@bjut.edu.cn

摘　要: 随着海量数据的涌现和智能应用需求的日益增长, 保障数据安全成为提高数据质量、实现数据价值的重要举措. 其中, 云边端架构是高效处理和优化数据的新兴技术, 联邦学习 (FL) 作为一个高效的去中心化的机器学习范式, 能够为数据提供隐私保护, 近年来引起了学术界及工业界的广泛关注. 然而, 联邦学习展示出了固有的脆弱性使其易于遭受投毒攻击. 现有绝大多数抵抗投毒攻击的防御方法依赖于连续更新空间, 但在实际场景中面向灵活的攻击方式和攻击场景可能是欠鲁棒的. 鉴于此, 提出一种面向云边联邦学习系统 (CEFL) 抵抗投毒攻击的防御方法 FedDiscrete. 其关键思想是在客户端利用网络模型边的分数计算本地排名, 实现离散更新空间的创建. 进一步地, 为了兼顾参与 FL 任务的客户端之间的公平性, 引入贡献度指标, 这样, FedDiscrete 能够通过分配更新后的全局排名对可能的攻击者实施惩罚. 广泛的实验结果表明所提方法在抵抗投毒攻击方面表现出显著的优势和鲁棒性, 且适用于独立同分布 (IID) 和非独立同分布 (non-IID) 场景, 能够为 CEFL 系统提供保护.

关键词: 联邦学习; 投毒攻击; 防御策略; 离散更新空间; 云边端架构

中图法分类号: TP309

中文引用格式: 赵亚茹, 张建标, 曹益皓, 黄浩翔. 云边联邦学习系统下抗投毒攻击的防御方法. 软件学报. http://www.jos.org.cn/1000-9825/7266.htm

英文引用格式: Zhao YR, Zhang JB, Cao YH, Huang HX. Defense Method Against Poisoning Attacks in Cloud-edge Federated Learning Systems. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7266.htm

Defense Method Against Poisoning Attacks in Cloud-edge Federated Learning Systems

ZHAO Ya-Ru[1,2], ZHANG Jian-Biao[1,2], CAO Yi-Hao[1,2], HUANG Hao-Xiang[1,2]

[1](College of Computer Science, Beijing University of Technology, Beijing 100124, China)

[2](Beijing Key Laboratory of Trusted Computing (Beijing University of Technology), Beijing 100124, China)

Abstract: With the proliferation of massive data and the ever-growing demand for intelligent applications, ensuring data security has become a critical measure for enhancing data quality and realizing data value. The cloud-edge-client architecture has emerged as a promising technology for efficient data processing and optimization. Federated learning (FL), an efficient decentralized machine learning paradigm that can provide privacy protection for data, has garnered extensive attention from academia and industry in recent years. However, FL has demonstrated inherent vulnerabilities that render it highly susceptible to poisoning attacks. Most existing methods for defending against poisoning attacks rely on continuously updated space, but in practical scenarios, those methods may be less robust when facing flexible attack strategies and varied attack scenarios. Therefore, this study proposes FedDiscrete, a defense method for resisting poisoning attacks in cloud-edge FL (CEFL) systems. The key idea is to compute local rankings on the client side using the scores of network model edges to create discrete update space. To ensure fairness among clients participating in the FL task, this study also introduces a contribution metric. In this way, FedDiscrete can penalize potential attackers by allocating updated global rankings. Extensive experiments demonstrate that the proposed method exhibits significant advantages and robustness against poisoning attacks, and is

* 基金项目: 北京市自然科学基金 (M21039)
收稿时间: 2023-10-09; 修改时间: 2024-05-25, 2024-06-30; 采用时间: 2024-08-02; jos 在线出版时间: 2024-12-11

applicable to both independent and identically distributed (IID) and non-IID scenarios, providing protection for CEFL systems.

Key words: federated learning (FL); poisoning attack; defense strategy; discrete update space; cloud-edge-client architecture

随着机器学习 (ML) 的快速发展, 终端设备产生的数据呈现爆发式增长的趋势, 这就对数据的优化和高效的处理提出了新的要求. 在这个背景下, 云边端架构[1,2]受到了学术界的广泛关注和探索, 这是因为其结合了边缘计算能够将计算和存储资源分布在靠近数据源头处执行计算任务的位置优势和云计算能够提供充足的计算和存储资源的优势, 实现数据计算、存储和通信效率之间的权衡. 鉴于终端和边缘计算设备收集的数据包含各种隐私信息, 为了避免隐私泄露, 联邦学习 (FL)[3,4]作为一种分布式的 ML 范式, 逐渐成为高效建模和隐私敏感的云边缘应用场景的一种有吸引力的技术. 图 1 展示了云边缘环境下的联邦学习框架 (CEFL). CEFL 跨云服务器、边缘服务器和终端设备实现训练一个高质量的联邦模型的目标. 不同于集中式学习框架, 在 FL 辅助的云边缘计算中, 其能够将训练模型卸载到多个边缘服务器以执行边缘聚合并获得边缘模型, 进一步地, 上传边缘模型到云服务器执行全局聚合, 以训练一个共享的全局模型, 该过程只传输模型参数, 从而降低用户隐私泄露风险.

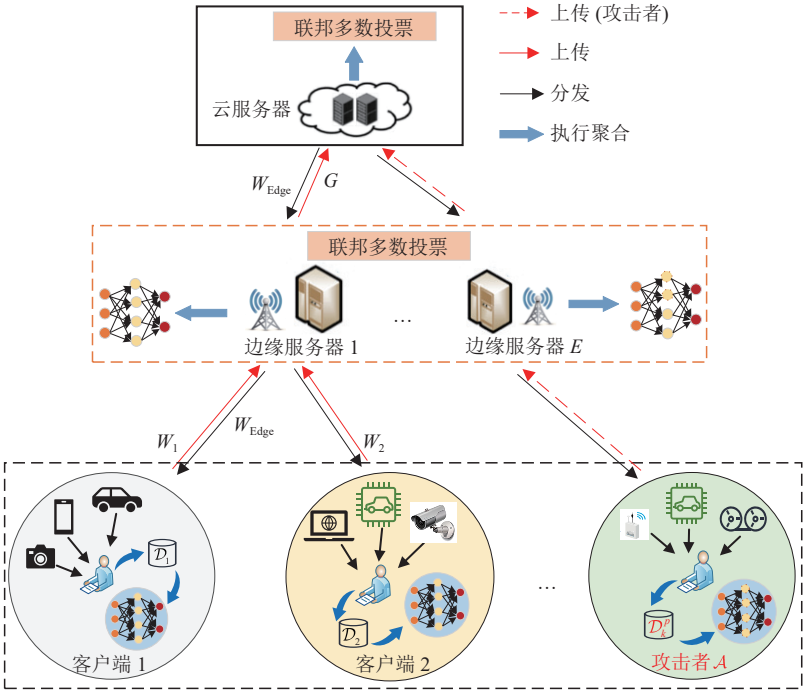


图 1 CEFL 框架

然而, CEFL 的分布式架构及固有的脆弱性[5]使其易于遭受潜在的安全威胁. 一方面是因为边缘环境下设备的多样化和计算节点的异构性导致安全防护能力较弱的节点易于面临恶意攻击; 另一方面, 良性客户端很容易受恶意攻击者的干扰, 比如, 攻击者利用上传的模型参数恢复原始数据的统计特征, 导致隐私泄露, 或者攻击者在本地数据集中注入恶意样本 (操纵标签向量或输入矩阵), 干扰联邦模型收敛或使其发生偏离. 这些脆弱性对许多安全领域带来严重威胁, 比如, 工业物联网[6]、自动驾驶[7]和视频识别[8]等. 例如, 针对交通标志的投毒将会干扰自动驾驶汽车正确识别标志, 进而增加事故发生的风险.

在本文中, 我们集中于数据投毒攻击, 攻击者发起投毒攻击的核心思想是利用 CEFL 系统的脆弱性诱使全局模型间接地学习攻击者选择的特定模式. 在实际场景中, 数据投毒攻击往往应用广泛且更具隐蔽性, 这给对抗投毒攻击的防御机制的探讨带来了挑战. 目前, 一些面向 FL 系统的防御方法主要可以分为以下几类: (1) 检测异常数据[9−11], 旨在识别训练数据中的恶意或异常样本, 比如, 搜索离群点、检测不符合正常数据分布的样本; (2) 鲁棒性

训练[12−14], 旨在改进模型训练过程、采用对抗训练策略或设计聚合策略来提高模型鲁棒性并降低攻击影响; (3) 投毒检测和识别[15−17], 利用客户端上传的模型参数 (权重或梯度) 之间的差异或行为分析方法来检测并移除可能的攻击者; (4) 着手于隐私角度[18−20], 采用差分隐私和同态加密技术等来保护 FL 中的隐私信息, 防止数据和模型遭受投毒攻击. 以上这些方法都是依赖于连续更新空间. 然而在真实场景中, 连续空间能够提供给攻击者更多选择模型更新的机会, 致使攻击者能够更容易地搜索最优攻击参数, 进而实施目标性的投毒攻击. 相比之下, 离散化空间在一定程度上限制了攻击者选择模型更新的范围. 结合离散更新空间, 可以从本质上减少攻击者的可选择范围, 从而针对性地提高防御策略的性能.

为了便于理解, 图 2 给出了客户端更新空间的一个简单示例. 可以看到, 在连续空间中 (图 2(a)), 客户端更新可以在安全空间内连续变化, 这也是现有研究广泛采用的方法. 在这种情况下, 攻击者能够试图在更新空间中搜索恶意更新或干扰良性更新, 从而成功发起投毒攻击; 而在稀疏空间中 (图 2(b)), 每次更新只能从有限的若干个离散值中选择, 并以跳跃的方式 (弧状箭头表示) 进行搜索, 其中, 模型更新与离散值之间存在一一对应关系. 特别地, 直接采用离散更新空间取代连续更新空间将会对 CEFL 系统带来一定的影响, 主要包括: (1) 关键信息丢失, 连续空间可以通过微小变化捕捉数据中的细微模式和结构, 而离散空间无法精确表示这些差异; (2) 收敛速度慢, 连续空间允许模型在每次更新中进行微小调整并快速逼近最优解, 而离散空间可能需要更多轮迭代达到相当的水平; (3) 计算时间和通信代价增加, 连续空间允许模型参数在更细粒度的范围内调整以寻找全局最优解, 而离散空间限制了参数的可调范围, 可能导致陷入局部最优, 难以跳出. 因此, 合理利用离散空间开发鲁棒的防御策略存在一定难度.

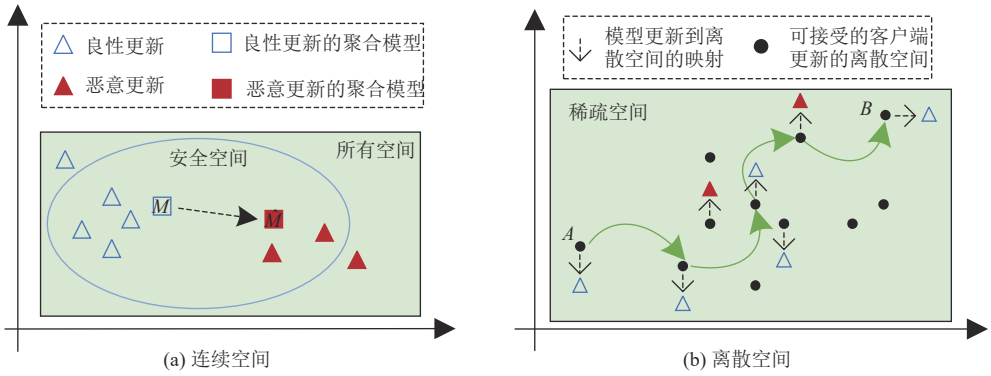


图 2 客户端更新空间示例

相反, 如果在创建离散空间的同时能够保留客户端的本地特征模式, 则关键信息丢失的概率将大大降低. 此外, 考虑到参与 FL 协作训练的客户端的贡献不同, 恶意客户端在发起投毒攻击之后, 通过上传恶意更新欺骗服务器, 同时从其他良性客户端的模型更新中搭便车受益, 进而对全局模型的性能造成影响. 基于此, 本文依赖于离散更新空间并兼顾客户端之间的公平性来探讨抵抗投毒攻击的防御方法, 其主要有以下 3 个挑战.

(1) 如何为不同客户端创建离散更新空间? 如上文所述, 在许多实际场景中, 由于客户端更新空间的连续性, 导致其难以创建离散空间. 一种有效应对连续更新空间的方法是利用机器学习技术, 期望充分利用客户端训练的本地模型, 结合权重参数作为基准, 依据神经网络特定层的连接边, 以构建离散更新空间. 所以, 如何创建可行且有效的离散空间, 对防御性能至关重要.

(2) 如何保证参与 FL 任务训练的不同客户端之间的协作公平性? 由于客户端的多样化使得具有较低贡献的参与者能够从贡献较高的参与者中受益, 最终导致全局模型的性能下降且加剧了客户端之间的不公平性. 因而亟需一种策略使其不仅能够量化参与者真实的贡献度大小, 同时也能够保留客户端的本地特征模式.

(3) 面对多样化的攻击能力和不同的应用场景 (独立同分布 (IID) 和非独立同分布 (non-IID)), 如何保持防御算法的高效性和鲁棒性? 一方面, 机器学习模型参数量大和维度高的特点造成的计算代价和通信开销不可忽视; 另一方面, 目前大多防御方法是面向 IID 场景, 而在 non-IID 场景下的性能较弱, 甚至不适用于 non-IID 场景. 因此,

如何确保在不同的攻击场景下, 尽可能地降低 FL 任务执行过程所需的通信开销也是一大挑战.

为了应对上述挑战, 本文提出了一种面向 CEFL 系统的防御方法 FedDiscrete (federated discrete). 其动机是攻击者发起投毒攻击的基本实践: 修改样本源标签为目标标签, 注入投毒样本到本地数据集中, 干扰客户端本地训练并试图从中获益, 从而导致攻击者和良性参与者构建的离散更新空间存在明显差异. 因此, FedDiscrete 的关键思想是首先计算每个参与者的贡献度, 在保证客户端之间公平性的同时也尽可能保留本地关键特征. 在此基础上, 本文基于每个客户端的本地模型连接边的分数排名来创建离散更新空间, 在边缘服务器和云服务器端采用多数投票聚合机制, 并通过迭代更新全局排名的方式来限制攻击者选择恶意更新的范围, 进而缓解投毒攻击的影响并增强防御鲁棒性. 同时, 为了降低训练过程的计算代价和通信开销, 进一步引入稀疏化策略, 以减少神经网络的参数数量并降低模型复杂度.

本文第 1 节介绍现有对抗联邦学习投毒攻击的防御方法和研究现状. 第 2 节介绍威胁模型和防御能力. 第 3 节介绍本文提出的 FedDiscrete 防御的设计细节并给出讨论. 第 4 节进一步设计广泛的实验并进行评估和讨论. 最后, 第 5 节总结全文.

1 联邦学习投毒攻击与防御相关工作

防御机制是规避 FL 系统遭受投毒攻击的重要手段. 现有防御方法的探讨主要包括以下 4 类: (1) 直接检测客户端属性或模型更新. 例如, 受进化聚类的启发, Shi 等人[21]在服务器端通过比较两轮之间的检测结果来识别可能的恶意客户端, 以缓解攻击影响. Zhang 等人[17]提出了 FLDetector 框架, 通过评估客户端本地模型在多次迭代中的更新方向与服务器预测的模型更新方向之间的一致性来检测恶意客户端. 而且, Zhao 等人[22]首先分析了多标签翻转攻击方法带来的影响, 进一步提出一种根据投毒样本和干净样本所训练的梯度特征之间的差异识别恶意攻击者的防御方法. Ben Saad 等人[9]提出了另一种框架, 其首先使用深度强化算法并基于声誉值动态地选择网络切片作为可信参与者, 进一步利用无监督 ML 识别投毒模型更新, 以自动检测 FL 过程中的恶意参与者, 从而保障零接触 B5G 网络中 FL 安全. 类似地, Al-Maslamani 等人[23]引入声誉概念来度量每个客户端的可信赖性, 进一步采用基于深度强化学习的深度确定性策略梯度算法 (DDPG) 来提升 FL 模型准确度和稳定性, 结果表明其性能优于基于深度 Q 网络的声誉方法 (DQN)[24]. Liu 等人[25]利用多权重主观逻辑模型 (SLM) 直接从交互历史中计算候选客户端的声誉值, 间接从其他服务器推荐的声誉意见中计算. 此外, Cao 等人[10]开发了一种拜占庭鲁棒的防御方法 FLTrust, 通过比较客户端本地更新与服务器端训练的模型更新方向之间的差异, 给本地更新分配相应的信任分数, 进而限制攻击者利用本地模型更新制造的恶意影响. 后来, Han 等人[11]探讨了 IID 设置下模型是否投毒与其在干净数据集上训练的损失是否符合正态分布之间的关系, 基于此, 提出了一种防御方法 ND_defense, 其主要是通过计算每个客户端模型的损失来检测并移除异常的客户端模型. 然而, 上述两种方法均假设服务器拥有一个干净数据集. (2) 鲁棒性训练. 例如, Zhao 等人[12]设计了一种客户端交叉验证的防御方案以检测异常更新, 也提出一种动态分配客户端的机制, 将检测任务分配给最合适的客户端, 以解决不适用 non-IID 设置的问题. 然而, 该检测方法成本较高, 不适合部署在边缘节点上. 之后, Kumar 等人[13]提出了一种对抗模型投毒攻击鲁棒的防御机制 FedClean, 该方法包含一个信誉机制以追踪每个代理的可信度并选择可信赖的代理执行模型训练, 以及一个更新质量控制机制来检测恶意更新. 基于样本重用的思想, Wang 等人[26]提出了有限聚合策略以缓解投毒样本造成的影响并提高防御的鲁棒性. Zhang 等人[14]提出了聚合方法 SAFELearning, 通过划分客户端为多个子组执行聚合并比较全局模型之间的差异, 移除超出阈值的子模型, 实现防御后门攻击并保护 FL 场景隐私. 特别地, Mozaffari 等人[27]提出了联邦秩学习方法 (FRL), 旨在通过稀疏化客户端的更新空间来协同训练一个全局模型并提高 FL 对投毒攻击的鲁棒性, 但这个方法集中于探讨无目标投毒攻击且忽略了参与者可能搭便车的情况. (3) 对抗性训练. 比如, Zhang 等人[28]分析了基于生成对抗网络 (GAN) 的投毒攻击, 攻击者通过训练 GAN 生成近似训练数据集的样本, 进而生成投毒更新并上传缩放后的投毒更新到服务器, 从而破坏全局模型. Zhang 等人[15]提出了一种防御方法 RobustFL, 该方法通过在服务器端建立一个基于 logits 的预测模型来决策给定的预测模型属于哪个参与者, 并以对抗训练的

方式避免该预测行为, 以缓解投毒攻击的影响. 然而, 该方案假设服务器端部署了一个预测模型. 进一步地, Zhao 等人[29]利用 GAN 在训练过程中生成审计数据, 通过审计其模型准确性来消除对手. 更近地, 为了检测金融欺诈检测场景下基于 GAN 的投毒攻击并实施隐私保护, Qiao 等人[16]提出了一种隐私感知和增量防御 (PID) 方法, 通过累积参与者模型参数的偏移量来表示模型参数的移动趋势, 以区分对手和正常参与者, 最后引入差分隐私保护技术来保护 PID 中参与者训练数据的隐私信息, 但是该方法没有考虑参与者的贡献大小. (4) 着手于隐私角度. 为了对抗移动边缘计算下的投毒攻击, Zhao 等人[18]开发了一种隐私保护方案, 利用特征学习模型从用户的位置数据中推断出投毒位置. 此外, Liu 等人[30]集中于 IID 数据的隐私增强 FL (PEFL), 通过采用同态加密技术并基于梯度数据来检测并惩罚投毒者, 但是这个方法对安全性做了较强的假设. Cao 等人[19]提出了一个集成 FL 框架 FLCert, 通过将客户分成不同组来学习全局模型, 然后执行多数投票以对输入进行分类, 并为该过程提供可证明的安全保证, 以防御恶意客户端的投毒攻击. 然而, 上述方法未能对 non-IID 场景展开探讨. 进一步地, Ma 等人[20]设计了一种基于同态加密技术的隐私保护防御策略 ShieldFL, 旨在通过测量两个加密梯度之间的距离并使用余弦相似度的拜占庭容错聚合方法实现 IID 和 non-IID 数据的鲁棒性, 以抵抗加密模型投毒.

现有对抗投毒攻击的防御方法: (1) 大多依赖于连续更新空间, 面向离散更新空间开展防御机制的探讨仍然较少; (2) 忽略了参与 FL 协作训练的客户端之间的公平性, 差异化的贡献度将诱发更强大的投毒攻击能力, 导致难以检测出攻击者; (3) 基于隐私保护或针对特定的攻击场景, 可能不适用于 non-IID 场景; (4) 一些工作依赖于预先假设, 比如, 假设防御者或服务器端拥有验证数据集或辅助模型, 这在 FL 中显然是不实际的.

2 威胁模型和防御能力

在本文中, 我们集中于 FL 辅助云边缘计算场景下的攻击设置, 而且, 我们考虑标签翻转攻击 (LFA) 方法[31]. 其中, 小部分终端设备视为攻击者并试图将精心制作的恶意数据注入本地数据集, 执行本地训练, 使得在良性参与者与攻击者之间形成差异化的模型更新和离散空间, 进而影响边缘聚合和全局聚合的性能, 同时, 这也会造成一种现象: 具有较小贡献的攻击者搭便车良性参与者, 最终影响全局模型的性能. 更具体地, 我们从 3 个方面阐述威胁模型: 攻击者的目标, 攻击者的知识和攻击者的能力, 相应地, 我们也介绍了防御者的能力. 为了方便, 我们给出了一些符号及描述, 如表 1 所示.

表 1 符号及描述

符号	描述	符号	描述
N	客户端总数	b	攻击者占比
K	每轮参与者的数量	M	攻击者数量
$\mathcal{A}$	攻击者	c_i	第 i 个客户端
D (D^p)	(投毒) 训练数据集	λ	贡献度
e	网络连接边	E	边缘节点数量
G	全局模型	R	全局迭代轮数
W	局部模型	w	权重参数
W_{Edge}	边缘模型	s_i	第 i 个边的分数
m	每个边缘节点聚合参与者的数量	S_i (S_g/S_G)	客户端 i 的本地排名 (边缘服务器端/云服务器端的全局排名)
Acc	模型准确度	$Cpre$/$Crec$	类精度/类召回率
ASR	攻击成功率	$Cpre_{tc}$/$Crec_{tc}$	目标类精度和目标类召回率

攻击者的目标: 在我们的攻击设置中, 攻击者最终目标是篡改样本源标签 (sc) 为期望的目标标签 (tc), 制造本地投毒数据集及恶意模型更新, 干扰离散更新空间的创建, 进而影响模型的预测性能. 通常, 攻击者试图部分控制 CEFL 系统, 确保在主任务上达到较高的模型精度, 而在特定的、攻击者期望的子任务上表现出较弱的性能.

攻击者的知识: 指攻击者对目标 ML 模型的了解程度. 在白盒攻击中, 攻击者隐藏在参与者中参与 FL 任务, 且知晓模型结构、学习算法、边缘模型和全局模型参数. 相反, 在黑盒攻击中, 攻击者不清楚上述信息, 但是可以迭

代地创建恶意样本, 达到破坏模型训练性能的目的, 甚至推理出与原始训练数据集相近分布的替代数据集.

攻击者的能力: 指攻击者如何控制训练数据集及控制的程度. 在 FL 任务中, 攻击者加入 CEFL 系统执行协作训练, 基于本地数据集执行 LFA 方法, 且通常做出较低的贡献. 特别地, 攻击者的能力受其比例大小的约束. 一般地, 攻击者占比不超过 50%, 主要原因是超出 50% 的攻击者将导致 FL 任务的学习速度显著降低且易于检测到[32]. 相反, 攻击者不能直接干扰边缘服务器和云服务器的聚合过程, 且无法访问和控制其他参与者的训练过程.

防御者的能力: 顾名思义, 其取决于攻击者的能力大小及防御者获取目标模型和训练数据的知识. 显然, 防御者的能力面对不同的攻击能力, 表现也不相同. 例如, Fung 等人[33]提出了 FoolsGold 防御, 通过计算贡献相似度来调整客户端的学习率, 进而实现对 FL 投毒攻击的检测. 典型地, TRIM[34]通过在投毒数据集上训练回归模型并以估计残差的方式实现防御. 现有防御机制失败的原因之一是连续更新空间的采用给攻击者提供了选择大量模型更新的机会. 基于这个思想, FedDiscrete 依赖于离散更新空间探讨防御性能. 就防御者的角度而言, 充分利用离散更新空间的特点并兼顾 CEFL 系统的脆弱性是必要的, 以构建更强大的防御能力并实现隐私保护.

3 FedDiscrete 设计

本节展示了防御方案 FedDiscrete 的细节. 第 3.1 节介绍了参与者贡献度的评估方法和算法流程. 第 3.2 节提供了离散更新空间的创建过程. 第 3.3 节给出了攻击者识别方法以及整个防御算法流程. 第 3.4 节讨论了 FedDiscrete 防御的可行性和有效性. 注意, 本文开发 FedDiscrete 方法的依据是离散更新空间比连续更新空间能够提供给攻击者更少选择模型更新的机会.

3.1 评估参与者贡献度

为了确保参与 FL 任务训练的 K 个客户端之间的公平性, 我们引入贡献度指标, 以计算每个参与者的贡献大小. 具体地, 基于文献 [35] 中已经证明的“最后一层模型参数对于检测攻击行为有着直接的影响”的思想, 在本文中, 我们首先利用客户端本地模型 $\tilde{W}_i^{r+1}$ ($\hat{W}_j^{r+1}$) ($i, j \in [1, K]$) 来捕捉更健壮的权重参数特征, 其主要是通过获取最后一层权重特征实现, 记为 $\tilde{w} = \{\tilde{w}_1^r, \tilde{w}_2^r, \ldots, \tilde{w}_K^r\}$, 其中, $\tilde{W}_i^{r+1}$ 和 $\hat{W}_j^{r+1}$ 分别表示第 i 个良性参与者和第 j 个恶意参与者在第 $r+1$ 轮训练的本地模型, 求解贡献度的详细步骤如下.

(1) 首先, 获取初始全局模型 G^0 的最后一层权重参数 $\tilde{w}^0$ 并更新 $\tilde{w}$, 即 $\tilde{w} = \tilde{w} - \tilde{w}^0 = \{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_K\}$. 基于此, 我们能够以最后一层权重参数作为基准实现参与者贡献度的计算. 值得注意的是, 直接利用 ML 模型所有层的权重特征将包含较多的冗余信息, 而且, 其余层与攻击模式的相关性有着较弱的联系, 导致无法有效检测攻击者.

(2) 进一步地, 我们采用典型的主成分分析方法 (PCA)[36]来提取 $\tilde{w}$ 的 2 个主特征, 记为 u, 目的是生成更能反映客户端属性的特征并降低高维度参数特征对检测攻击的影响.

(3) 然后, 我们采用余弦相似度 (cos(·)) 求解每两个参与者的权重特征之间的相似性, 记为 $\widetilde{cs}$, 表示为公式 (1), 这主要是基于攻击者和良性参与者的权重之间存在差异性的事实. 同样地, 基于 $\widetilde{cs}$, 利用 PCA 方法获取主特征, 该降维处理减少了攻击者隐藏主要特征的可能; 进一步利用 Median 函数求解质心 ct, 由于攻击者的数量总是不超过良性客户端的数量, 因此, 质心将总是落在多数良性参与者之间;

$$\cos(w_i, w_j) = \cos(\theta) = \frac{w_i, w_j}{\|w_i\| \cdot \|w_j\|} \tag{1}$$

(4) 接下来, 结合 u 和 ct 求解余弦相似度 cs, 以更直观地观察质心与每个参与者的权重之间的差异, 并将其视为贡献度 λ. 而且, 良性参与者将与质心保持相似的方向, 即接近 1, 相反, 攻击者将接近 0.

随着迭代的持续, 利用与质心的相似值累计求和每个参与者的贡献度, 并更新保存在客户端, 以用于离散更新空间的创建. 我们知道, 距离质心越近, 贡献度累积越大. 特别地, 在计算贡献度的过程中, 我们采用第一四分位数 (q_1) 作为阈值来决策带有不同贡献的参与者对 FL 任务训练过程造成的影响, 这主要是因为较大的阈值将隔离更多的良性参与者, 从而减少了有效的数据源并降低模型整体性能; 较小的阈值可能导致贡献较低的参与者未被检测, 从而使模型继续遭受其影响[37]. 最后, 我们对贡献度 λ 进行处理, 对于贡献度为负值的参与者, 我们将其设置

为 0. 与以往方法不同, 我们求解参与者贡献度的过程不仅保留了权重特征的高保真性, 也权衡了参与者之间的公平性, 避免出现一些攻击者搭便车的情况. 算法 1 展示了求解参与者贡献度的流程.

算法 1. 求解参与者贡献度.

输入: $\bar{W}_i^r$, $\hat{W}_j^r$, w^0;

输出: λ_i.

1. 基于 w^0 求解最后一层权重参数数 $\tilde{w}^0$;
2. **for** $r \in [0, R-1]$ **do**
3. 接收 K 个参与者上传的本地模型 $\bar{W}_i^r$ ($\hat{W}_j^r$), $\forall i, j \in (1,\ldots,K)$ & $i \neq j$;
4. 获取 $\bar{W}_i^r$ ($\hat{W}_j^r$) 的最后一层权重参数: $\tilde{w} = \{\tilde{w}_1^r, \tilde{w}_2^r, \ldots, \tilde{w}_K^r\}$;
5. 更新 $\tilde{w}$: $\tilde{w} = \tilde{w}^0 - \tilde{w} = \{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_K\}$;
6. 获得 $\tilde{w}$ 的主特征: $u = ((u_1^1, u_1^2), \ldots, (u_i^1, u_i^2), \ldots, (u_K^1, u_K^2)) \leftarrow \mathrm{PCA}((\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_K), components = 2)$;
7. 对 $\tilde{w}$ 求解余弦相似度: $\widetilde{cs} = (\widetilde{cs}_{1,1}, \widetilde{cs}_{1,2}, \ldots, \widetilde{cs}_{i,j}, \ldots, \widetilde{cs}_{K,K}) \leftarrow \cos(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_K)$;
8. 提取 $\widetilde{cs}$ 的主特征: $((s_1^1, s_1^2), \ldots, (s_i^1, s_i^2), \ldots, (s_K^1, s_K^2)) \leftarrow \mathrm{PCA}(\widetilde{cs}_{1,1}, \widetilde{cs}_{1,2}, \ldots, \widetilde{cs}_{i,j}, \ldots, \widetilde{cs}_{K,K})$;
9. 求解质心 ct: $ct \leftarrow \mathrm{Median}((s_1^1, s_1^2), \ldots, (s_i^1, s_i^2), \ldots, (s_K^1, s_K^2))$;
10. $cs = (cs_1, cs_2, \ldots, cs_K) \leftarrow \cos(u, ct)$;
11. 累计并存储贡献度: $\lambda_i^{r+1} = \lambda_i^r + cs_i^{r+1}$;
12. 采用第一四分位数作为最佳阈值: $\begin{cases} q_1 \leftarrow Q_1(\lambda, 0.25) \\ \lambda \leftarrow \lambda_i - q_1 \end{cases}$;
13. 归一化处理: $\lambda = \lambda / \max_i(\lambda)$;
14. **for** $i \in [1, K]$ **do**
15. **if** $\lambda_i < 0$ **then**
16. $\lambda_i = 0$
17. **end if**
18. **end for**
19. **end for**
20. **return** λ_i

3.2 创建离散更新空间

在 CEFL 系统中, 云服务器首先为 G^0 初始化随机权重 w^0 和相应的分数 s^0, 然后分发 G^0 给所有 E 个边缘服务器和 K 个参与者 (共享 w^0 和 s^0). 其中, 本文采用 Signed Kaiming Constant 算法[38]生成 w^0 以及 Kaiming Uniform 算法[39]生成 s^0. 我们知道 FedDiscrete 的训练目标是在不改变权重大小的情况下搜索信誉度最高的边. 特别地, 在构建离散更新空间的过程中, 我们仅考虑网络模型的卷积层和全连接层, 通过逐层排序模型连接边的分数 s^0 来求解初始全局排名 S_G^0. 所谓的排名是指向量从低到高排序后该向量元素对应的索引值, 本文使用 *ARGSORT* 函数计算. 例如, 假设该网络模型存在 6 个边, 即 $w^0 = [w_0, w_1, w_2, w_3, w_4, w_5]$, 对应 $s^0 = [0.5, 0.2, 0.3, 0.4, 0.7, 1.2]$, 则有 $S_G^1 = [1, 2, 3, 0, 4, 5]$, 这一过程形式化表示为 $S_G^1 \leftarrow ARGSORT(s^0)$.

在客户端, 我们采用 edge-popup (EP) 算法[38]稀疏化网络模型, 以搜索子网络. EP 算法的核心思想是通过迭代地移除网络中不重要的边 (连接权重), 达到减少网络参数数量、降低网络的规模和提高模型效率的目的. 具体地, 客户端首先接收来自云服务器端分发的 G^0 (w^0 和 s^0), 即 $W_{i\in[1,K]}^0 \leftarrow W_{\mathrm{Edge}}^{(i\in[1,E],0)} \leftarrow G^0$. 以第 r 轮为例, 边缘服务器共享全局排名 S_g^r 到客户端, 基于 S_g^r, 排序每个客户端本地模型中每一层连接边的分数. 假设 $S_g^r = [2, 3, 0, 5, 1, 4]$, 客户端 c_i 的本地模型连接边的分数为 $s^r = [0.5, 0.2, 0.3, 0.4, 0.7, 1.2]$, 这就意味着边 e_2 应该有最低的分数 $s_2 = 0.2$, 边 e_4

有最高的分数 $s_4 = 1.2$, 因此, 对于客户端 c_i, 更新后边的分数为 $s^r = [0.4, 0.7, 0.2, 0.3, 1.2, 0.5]$, 该过程表达为 $s^r[S_g^r] \leftarrow SORT(s^r)$. 值得强调的是, 此时的 s^r 不是子网络模型更新后的边的分数向量, 而是在此基础上, 利用第 3.1 节所求解的参与者的贡献度 (作为加权系数) 执行加权操作后获得的, 即 $s^r \leftarrow \lambda_i \times s^r$. 其次, 在 EP 算法中, 每个客户端 c_i 基于本地数据集 D_i 执行本地训练, 在前向传播过程中, 选择分数较高的前 k 个边, 其中, k 表示初始网络中的边保留在最终在子网络中的边的总数的比例并设置 $k = 50\%$, 在反向传播过程中实现分数的更新, 迭代循环直至达到 $epoch$ 上界. 此时, 获得本地子网络模型 $\bar{W}_i$ 并作为该轮的全局模型, 即 $\bar{W}_i \leftarrow EP(D_i, W_i, s, k, epoch)$. 注意, 随着 EP 算法不断迭代, 边的分数也在不断更新, 本地排名是基于最终的更新分数实现. 同时, 对于每个客户端, 依赖于本地子网络模型能够获得边的分数 s^r 和本地排名 S_i^r. 例如, 客户端 c_1 的排名是 $S_1^r = [4, 0, 2, 3, 5, 1]$, 则我们选择分数最高的前 50% 个边, 即 c_1 使用边 {3, 5, 1} 构成子网络模型. 注意, 在整个训练过程中, 只更新边的分数, 而不更新模型参数. 紧接着, 依赖于 K 个客户端的本地排名, 即 $S_{i\in[1,K]}^r$, 创建离散更新空间. 因此, 该过程可以表示为 $S_i^r \leftarrow ARGSORT(\bar{W}_i, s^r, S_i^r)$ (良性参与者) 或者 $S_i^r \leftarrow ARGSORT(\hat{W}_i, s^r, S_i^r)$ (恶意攻击者).

接下来, 客户端上传本地排名构建的离散更新空间到边缘服务器端. 对于每个边缘服务器, 基于接收到的客户端本地排名执行多数投票聚合机制, 这里采用投票函数 $VOTE$ 实现. 具体地, 对于一个客户端 c_i, 本地排名的索引值 j 表示对应边 $S_i^r[j]$ 的信誉度值. 比如, 客户端 c_1 的本地排名为 $S_1^r = [4, 0, 2, 3, 5, 1]$, 则我们视为 c_1 的边 e_4 在索引 $j = 0$ 处具有最低的信誉度 (分数), 相反, 边 e_1 在 $j = 5$ 处具有最高的信誉度. 因此, $VOTE$ 函数将分配信誉度 0 给 e_4, 1 给 e_0, 2 给 e_2 及 5 给 e_1 等. 换句话说, 对应边的分数越低, 其信誉度值也越低. 简言之, 基于每个边缘服务器聚合的 m 个客户端的本地排名 $S_{i\in[1,m]}^r$, $VOTE$ 函数将计算所有边的信誉度并通过赋值 $S_{i\in[1,m]}^r$ 向量中对应边的索引实现. 类似地 c_2, c_3 等, 直至所有客户端中对应边的信誉度统计完成. 然后, 累计计算并求和所有 m 个客户端本地排名中对应边的信誉度, 最终返回第 1 个边缘服务器求解的所有边 $e_{i\in[0,5]}$ 的总信誉度构成的向量, 记为 H_1, 即 $H_1 = SUM(SUM(ARGSORT(S_i^r)))$. 最后, 边缘服务器通过排序最终信誉度值来计算全局排名 S_g^r, 即 $S_g^r = ARGSORT(H_1)$. 该过程形象化表达为 $S_g^r = VOTE(S_{i\in[1,m]}^r)$. 类似地, 直至所有 E 个边缘服务器统计完成.

最后, 边缘服务器上传 E 个全局排名 $S_{g,i\in[1,E]}^r$ 构建的离散空间到云服务器. 在云端, 基于全局排名同样执行多数投票聚合机制以获得最终的全局排名 S_G^{r+1}, 该过程可以表示为 $S_G^{r+1} = VOTE(S_{g,i\in[1,E]}^r)$. 迭代地, 在第 $r+1$ 轮, 客户端执行本地模型训练, 并基于求解的 S_G^{r+1} 来更新本地排名, 直至达到最大迭代轮或满足终止条件.

在本文中, 创建离散空间及服务器端执行聚合的整个过程均是逐层实现的, 且良性参与者和恶意参与者均是基于各自本地模型独立完成离散空间的构建, 因此, 在边缘聚合操作之前需要先融合所有客户端求解的本地排名构成的离散空间. 不同于以往利用迭代更新的权重参数, 我们是基于网络模型连接边的分数创建离散更新空间, 以这种方式, 我们可以减少攻击者可选择的模型更新空间.

3.3 攻击者识别

FedDiscrete 防御可以以一种简单的方式提高 CEFL 系统的安全性: 通过设置阈值 q_1 来约束参与者的贡献大小, 具体来说, 我们通过评估每个参与者在所有训练轮次中的贡献度, 动态调整其本地模型更新的权重, 以降低潜在攻击者对模型性能带来的影响, 同时保留了对全局模型有益的差异性更新. 而且, 我们采用与攻击行为紧密相关的最后一层权重参数实现贡献度的计算. 这种方式不仅能有效地减轻投毒攻击的影响, 还能够促进联邦模型在 non-IID 数据设置下的稳健训练; 在此基础上, 在所有参与者上传本地排名构建的离散更新空间之前, 我们使用每个参与者的实际贡献大小来约束其模型连接边的分数, 以缓解其在下一轮模型训练过程中对全局模型的恶意影响, 通过不断地迭代训练来削弱攻击者的贡献, 进而提高模型的鲁棒性以及客户端之间的公平性. 值得强调的是, FedDiscrete 防御的主要目标是集中于缓解攻击者造成的投毒影响, 而不是检测出攻击者.

更具体地说, 对于每个攻击者, 其通过对含有预先设置的源标签的样本执行 LFA 恶意操作, 以输出攻击者期望的目标标签, 实现目标投毒攻击的目的. 在我们的模型测试中, 所有参与者均是按照相同的数据划分方法, 包括 IID 和 non-IID 设置, 而攻击者主要是在本地训练过程中实施恶意攻击行为, 这就使得攻击者与良性参与者之间在模型更新的特征、参数调整的方向和幅度上产生显著差异. 因此, 在不清楚参与者是否为攻击者的情况下, 利用贡

献度指标并构建离散更新空间来降低攻击影响是可行的. 此外, 由于我们规避了客户端直接上传整个局部模型执行 FL 任务, 且通信代价主要来自客户端上传本地排名到边缘服务器端以及边缘服务器上传全局排名到云服务器端, 而排名向量远小于全局模型, 显然, 这很大程度上降低了计算代价和通信开销, 理论上来讲, 这将显著提高 FedDiscrete 的高效性. 最后, 我们给出了 FedDiscrete 防御策略, 如算法 2 所示.

算法 2. FedDiscrete 防御策略.

输入: D, b, 稀疏度 k, 参与者的贡献度 λ, 局部模型及权重参数 W/w, 本地训练代数 $epoch$, R;
输出: S_G^{r+1}.

//系统初始化
1. 创建 CEFL 环境: 定义云服务器, E 个边缘服务器, 确定客户端属性, 给所有客户端划分训练数据集及分布情况、初始化全局模型 G^0;
2. 初始化 G^0 的随机权重 w^0 和分数 s^0;
3. 排序 s^0 并求解初始全局排名: $S_G^1 \leftarrow ARGSORT(s^0)$;
4. **for** $r \in [0, R-1]$ **do**
//客户端: 创建离散更新空间
5. 从 N 个客户端中随机选择 K 个参与者并确定攻击者数量: $M \leftarrow \max(N \cdot b,\ 1)$;
6. 分发 G^0 给所有参与者和边缘服务器: $W_{i\in[1,K]}^0 \leftarrow W_{\text{Edge}}^{(i\in[1,E],0)} \leftarrow G^0$;
7. **for** $c_{i\in[1,K]}$ in parallel **do**
8. 基于 S_g^r 和 λ_i (算法 1) 求解客户端本地模型连接边的分数: $s^r[S_g^r] \leftarrow SORT(\lambda_i \times s^r)$;
9. 采用 EP 算法搜索本地子网络: $\bar{W}_i \leftarrow EP(D_i, W_i, s, k, epoch)$;
10. **if** $c_i \in \mathcal{A}$ **then** //恶意攻击者
11. 利用 LFA 生成投毒数据 D_i^p;
12. 基于 D_i^p 和 $\bar{W}_i$ 执行本地训练, 生成投毒模型 $\hat{W}_i^r$ 并获得 s^r 和 S_i^r;
13. **else** //良性参与者
14. 基于 D_i 和 $\bar{W}_i$ 执行本地训练, 生成本地模型 $\bar{W}_i^r$ 并获得 s^r 和 S_i^r;
15. 对于每个客户端, 创建离散更新空间: $S_i^r \leftarrow ARGSORT(\bar{W}_i \text{ or } \hat{W}_i, s^r, S_i^r)$;
16. **end for**
//边缘服务器: 多数投票聚合机制
17. **for** $i \in [1, E]$ **do**
18. $S_{g,i}^r = VOTE(S_{i\in[1,m]}^r)$;
19. **end for**
//云服务器: 多数投票聚合机制
20. $S_G^{r+1} = VOTE(S_{g,i\in[1,E]}^r)$;
21. **end for**
22. **return** S_G^{r+1}

3.4 讨 论

在我们的工作中, 提出了一种防御算法 FedDiscrete, 以缓解对 CEFL 系统的投毒攻击. 注意, 模型更新空间是否连续对防御策略的性能有着直接的影响, 通常, 连续更新空间使得客户端拥有更多选择和学习本地更新的可能, 这就给攻击者发起投毒攻击创造了机会, 使得设计有效的防御方法更加困难. 因此, 我们调查了 FedDiscrete 算法在不同的训练数据分布下面向差异化的攻击能力表现出的性能. 而且, 比较现有的工作, 结果表明 FedDiscrete 在

IID 和 non-IID 场景下均具有明显的优势, 具体的实验细节展示在第 4 节.

此外, 我们也从 FL 高效性的角度来讨论 FedDiscrete 的性能. 不同于以往基于连续更新空间的防御方法, 上传客户端本地训练的模型更新到服务器端, 其极大地增加了计算代价和通信开销, FedDiscrete 规避了直接上传本地模型更新, 而是集中于上传每个客户端的本地排名向量到服务器端, 这在协同训练的全局任务中显然降低了计算成本和通信开销. 而且, 与集中式 ML 框架不同, 我们的 FL 任务的学习效率主要取决于客户端的计算成本、客户端和边缘服务器以及边缘端和云服务器之间的通信成本、边缘服务器和云服务器执行聚合算法的计算成本. 在客户端, 本地训练是离线进行的. 根据 3.2 节的描述, 一方面, 我们知道 FedDiscrete 的额外成本主要来自于客户端计算参与者贡献度的过程, 其时间复杂度为 $\mathrm{O}(\tilde{d}_w \cdot T_{\mathrm{Local}})$, 其中, $\tilde{d}_w$ 表示最后一层权重参数的维度大小, T_{Local} 表示本地训练阶段的时间, 这主要是因为贡献度是基于模型权重参数通过主特征提取、线性计算和缩放操作迭代实现的; 另一方面, FedDiscrete 防御创建离散更新空间的计算成本受本地排名计算的支配, 其本质是对网络中的卷积层和全连接层的连接边的分数所构成的固定维度大小的向量进行排序操作, 则对于一个客户端而言, 其时间复杂度为 $\mathrm{O}(d_{\mathrm{Local}} \times \log(d_{\mathrm{Local}}))$, 其中, d_{Local} 表示该客户端构建的离散更新空间的维度大小. 由此可知, 模型越复杂, 参数数量增加, 离散化操作的计算量越大. 在检测过程中, 通信开销主要来自于所有客户端上传本地排名构建的离散空间到边缘端的过程以及边缘端上传 E 个边缘模型的全局排名构建的离散空间到云端的过程. 我们知道, 客户端是通过随机选择的方式参与 FL 任务, 这就使得每个客户端参与的轮数不同, 对于前者, 每个客户端平均参与到边缘模型的次数为 $\frac{K}{N} \cdot \frac{E \cdot m}{K} = \frac{E \cdot m}{N}$, 则对应通信开销为 $\mathrm{O}\left(\frac{Em}{N} \cdot \frac{m}{K} \cdot R \cdot P_{\mathrm{Local}}\right)$, 其中, P_{Local} 表示客户端上传的离散更新空间的参数数量; 类似地, 对于后者, 一个边缘模型的通信开销为 $\mathrm{O}(P_{\mathrm{Edge}} \cdot R)$, P_{Edge} 表示边缘模型上传的离散更新空间的参数数量. 因此, 检测过程的总通信开销为 $\mathrm{O}\left(\left(\frac{Em^2}{NK} \cdot P_{\mathrm{Local}} + P_{\mathrm{Edge}}\right)R\right)$. 相较于边缘端接收客户端上传的离散更新空间的过程所产生的通信开销而言, 后者所产生的通信开销是微不足道的, 因此, 在本文中, 我们主要关注前者产生的通信开销. 显然, 根据第 3.2 节的介绍, 这就很大程度上降低了上传和下载模型参数的通信成本, 而且, 客户端通信的模型参数通常是 32 bit 或 64 bit (比如, FedAvg 算法). 在服务器端, 其操作是在线执行的. 在 CEFL 系统下, 边缘服务器接收客户端上传的离散更新空间并执行多数投票聚合, 以获得边缘模型; 同样地, 云端接收边缘模型上传的离散更新空间并执行全局聚合, 以获得全局模型. 因此, 服务器端的计算代价主要源于边缘端和云端的多数投票聚合操作, 即依赖于客户端的本地排名, 通过线性求和的方式计算所有边的信誉值, 然后排序并返回对应索引以获得全局排名, 该过程的时间复杂度通常是线性的, 即为 $\mathrm{O}(K \times d_{\mathrm{Local}} + E \times d_{\mathrm{Edge}})$, 其中, K 是客户端数量, d_{Edge} 是边缘端构建的离散更新空间的维度大小. 一般地, 在 CEFL 系统下, 聚合阶段的计算能力比客户端强大得多, 因此, 我们认为聚合阶段的线性成本也是有限的, 可以忽略不计. 综上, 就提高 FL 高效性和减少计算成本而言, 我们的方法具有一定的可行性.

进一步地, 目前有一些工作与本文的思想类似, 它们基于连续更新空间或者离散更新空间设计防御方法, 以移除针对 FL 投毒攻击的恶意更新. 例如, Li 等人[40]提出了一种两阶段的防御方法 LoMar, 其主要是通过对客户端的模型更新打分以区分恶意和干净更新; Lyu 等人[41]探讨了一种利用声誉强制参与者收敛到不同的模型, 以实现客户端之间的公平性; Mozaffari 等人[27]提出了 FRL 方法, 通过创建离散更新空间减少可选择的模型更新. 我们认可上述的思想, 因为在 CEFL 系统中, 良性参与者和恶意攻击者所训练的模型更新之间存在差异, 鉴于此, 通过移除可能的恶意更新实施防御是可行的; 另一方面, 离散更新空间的使用会导致部分关键信息丢失, 比较连续更新空间, 其能够在选择模型更新空间上缓解攻击者的投毒能力, 基于这个思想实施防御同样也是可行的.

然而, LoMar[40]基于连续更新空间实施防御, 但未能考虑到模型更新的复杂性 (ML 模型大和维度高的特点), 这主要是由于 ML 模型的维度也是影响防御性能的一个重要因素. CFFL[41]假设服务器端拥有一个与原始训练数据具有相似分布的验证数据集, 显然, 该方法类似于现有集中式 ML 防御方法的思想, 不适用于 FL 场景. FRL[27]方法集中于无目标投毒攻击且忽略了协作训练的参与者之间可能存在搭便车的现象. 相反, FedDiscrete 防御能够在不明确训练数据分布的情况下, 利用离散更新空间并引入参与者的贡献实施防御, 同时也保留了 CEFL 的隐私特性. 此外, 文献 [31] 通过识别良性更新和投毒更新并表示为不同的簇实现防御, 显然, 在未能选择最优决策边界或

面对具有隐身度量的攻击时, 这种方法很容易失败. 最后, 我们还评估了 FedDiscrete 的防御在面对动态的攻击能力时的效果. 总之, 比较现有的防御机制, 所提方法 FedDiscrete 能够从离散化模型更新的新颖角度来探讨缓解投毒攻击的防御策略, 同时引入贡献度指标来确保参与 FL 训练任务的客户端之间的公平性, 通过不断削弱攻击者所做出的贡献, 从而创建一个适用于云边缘计算环境的可行且有效的防御机制.

4 实验评估

为了评估 FedDiscrete 方法的性能, 我们采用 Python 创建虚拟的云边端环境, 并在 CEFL 框架下执行了广泛的实验. 其中, 对于边缘端, 设置 E ($E = 5$) 个边缘节点, 且每个边缘节点平均接收 m ($m = 2$) 个参与者上传的本地模型更新. 所有实验均在 Python 环境下使用 PyTorch 实现, 具体配置为: 英特尔酷睿 i7-12700H CPU 处理器, 16 GB 内存, 英伟达 GeForce RTX3070Ti GPU.

4.1 实验设置

数据集: 本文采用了 FL 领域中 3 种经典的数据集, 包括手写数字识别数据集 MNIST[42], 图像数据集 CIFAR-10[43]和时尚服装分类数据集 Fashion-MNIST[44], 其广泛使用在现有投毒攻击和防御方法中. MNIST 和 Fashion-MNIST 分别是由从 0 到 9 的手写数字和服装形成的 28×28 的灰度图像, 且均是由 60 000 张图像组成的训练数据集和 10 000 张图像组成的测试数据集; CIFAR-10 是由 60000 张 32×32 的彩色图像构成的 10 个类, 包括 50000 张图像用于训练和 10000 张图像用于测试, 每个类别有 6000 个图像. 特别地, 本文也采用一种医学图像数据集 PathMNIST[45], 它是由不同病人的病理切片图像形成的 28×28 的彩色图像, 且包含 9 个类别的病理学图像, 每个类别对应不同的病变类型.

模型设置: 本文实验主要是为了评估防御机制的综合性能, 而不是搜索具有最好的 FL 性能的学习模型. 基于此, 我们定义了 CEFL 框架下的一些默认设置和学习模型. 对于 MNIST 数据集, 我们采用典型的 LeNet 模型[46], 训练 SGD 的批次大小为 64, 且在 IID 和 non-IID 场景下分别执行全局迭代轮数 $R = 50$ 和 $R = 100$. 对于 CIFAR-10 数据集, 我们采用卷积神经网络模型 Conv8, 其是由 8 个卷积层和 3 个全连接层构成的学习模型, 训练 SGD 的批次大小为 32, 全局迭代轮数 $R = 50$. 对于 Fashion-MNIST 数据集, 我们采用一个包含两个卷积层和一个全连接层构成的学习模型, 其具有与 MNIST 相同的设置. 不管 IID 还是 non-IID 场景, 我们在每个数据集上均设置学习率为 0.01, 以实现较小的训练误差并快速收敛, 且客户端本地训练迭代 $epoch = 15$. 对于 PathMNIST 数据集, 我们同样采用一个包含两个卷积层和一个全连接层构成的学习模型, 并设置训练 SGD 的批次大小为 32, 本地训练迭代 $epoch = 5$.

攻击设置: 在本文中, 我们设置 $N = 100$, $K = 10$. 而且, 我们考虑了不同的攻击能力, 主要表现为每轮参与者中攻击者的比例 (b), 范围为 10%–50%, 即 $b = 10\%, 20\%, 30\%, 40\%$ 和 50%, 则对应攻击者的数量为 $M = 1, 2, 3, 4$ 和 5. 注意, b 值是由参与每轮训练的客户端的数量确定, 即 $M = b \times K$, 不同于以往的一些工作是基于所有客户端的数量选择攻击者, 使得实际参与任务训练的客户端中可能没有或者存在多数攻击者, 导致无法准确探讨攻击者的投毒影响以及防御性能. 为了确保训练的公平性和攻击的不确定性, 我们从 N 个客户端中随机选择参与者. 其次, 我们考虑广泛使用的标签翻转投毒的攻击形式, 即每次翻转源标签类为目标标签类, 以对子任务实施投毒, 记为 $sc \to tc$. 具体地, 对于 MNIST 和 Fashion-MNIST 数据集, 我们设置 $sc \to tc$ 为 $4 \to 6$, 对于 CIFAR-10 数据集, 设置 $sc \to tc$ 为 $5 : \text{cars} \to 3 : \text{frogs}$, 对于 PathMNIST 数据集, 我们设置 $sc \to tc$ 为 $1 \to 0$. 除了攻击者数量和攻击形式外, 我们还将不同的攻击场景作为评估防御性能的重要因素, 包括 IID 和 non-IID 设置. 不管是 MNIST, Fashion-MNIST, 还是 CIFAR-10, 我们均采用迪利克雷分布[47]给 N 个参与者划分训练数据集, 对于 IID 攻击场景, 我们设置超参数 $\alpha = 1000000$, 这样, 良性参与者和恶意攻击者很可能共享同一类样本, 这就给投毒攻击带来了挑战; 对于 non-IID 攻击场景, 我们设置 $\alpha = 1$, 以生成 non-IID 数据.

防御算法: 本文中, 我们比较了所提出的 FedDiscrete 防御算法与现有经典的防御方法, 主要包括标准的 FedAvg, MKrum, Median 和 FoolsGold. 具体介绍如下.

FedAvg[3]: 一种常用的 FL 算法, 通过加权平均来聚合模型参数. 其基本思想是将本地模型参数上传到服务器, 计算所有模型参数的平均值, 然后分发平均值给所有本地客户端, 迭代训练直至满足收敛条件.

MKrum[48]: 在 Krum 算法的基础上, 基于欧氏距离的拜占庭容错 ML 算法, 通过计算并求和每两个梯度之间的欧氏距离, 搜索出具有最小分数的梯度, 通过对选择的若干个梯度求平均来更新模型参数, 直至模型收敛.

Median[49]: 使用聚合器对所有客户端的参数更新值排序, 并在每次迭代中选择中值作为全局模型的贡献.

FoolsGold[33]: 通过对恶意更新的学习率进行惩罚来解决攻击者问题, 然后, 衡量每个更新和全局模型之间的角度差异实现恶意更新的检测, 且其能够防御 sybil 攻击.

此外, 我们还对比了应用在不同攻击场景下的较新颖的防御方法, 包括 Krum[48], FLTrust[10], Trimmed-mean[50], Auror[51], PEFL[30], RobustFL[15], CONTRA[52], 以及面向离散更新空间的防御方法[23−25], 详细的描述在第 1 节.

评估指标: 为了准确地评估实验结果, 我们引入了多个评价指标, 包括模型准确度 (*Acc*), 类精度 (*Cpre*), 类召回率 (*Crec*) 和 *F*1-*score* (*F*1). 具体地, *Acc* 用于衡量在所有预测样本中预测正确的样本所占的比例; *Cpre* 用于评估正确预测的正例样本占所有预测的正例样本的比例; *Crec* 指正确预测的正例样本占所有正例样本的比例, *F*1 定义为 *Crec* 和 *Cpre* 的调和平均值, 表达为公式 (2)–公式 (4) 所示. 我们知道越高的 *Crec*, *Cpre* 和 *F*1, 说明防御效果越好. 进一步地, 我们也采用混淆矩阵和攻击成功率 (*ASR*) 评估防御方法的有效性和鲁棒性. 具体而言, 混淆矩阵依赖于真实标签和预测标签, 用于直观地观察实施 FedDiscrete 防御后的分类效果; *ASR* 用于衡量带有源标签的目标样本被错误分类为攻击者渴望的目标标签的比例. 最后, 我们对计算代价和通信开销进行评估以讨论 FedDiscrete 的可行性和高效性. 特别地, 在本文中, 我们主要集中于客户端-边缘服务器端的上行通信成本.

$$Cpre = TP/(TP+FP) \tag{2}$$

$$Crec = TP/(TP+FN) \tag{3}$$

$$F1 = \frac{Crec \times Cpre}{2\times(Crec+Cpre)} \tag{4}$$

其中, *TP* (true positive) 表示分类器将正例正确分类的样本数, *FN* (false negative) 表示分类器将正例错误分类为反例的样本数, *FP* (false positive) 表示错误地分类为正例的反例样本的数量.

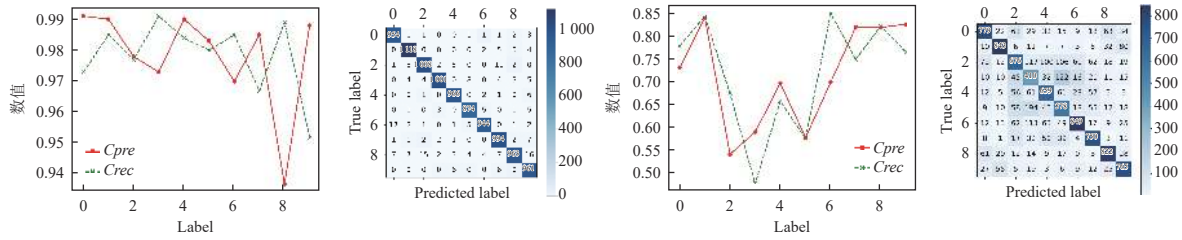
值得注意的是, 对于每个测试, 我们均执行了 5 次实验并计算评价指标的平均值, 以避免结果的随机性.

4.2 实验结果与分析

在本节中, 我们从多个角度展示了我们的实验结果并进行详细的分析和讨论.

4.2.1 无攻击者投毒场景的性能评估

首先, 我们评估了没有恶意攻击者发起投毒攻击 ($b=0$) 情况下的模型性能, 包括 IID 和 non-IID 两种场景, 并基于 *Acc*, *Cpre*, *Crec* 指标和混淆矩阵对结果进行分析, 实验结果如图 3, 图 4 所示.



(a) MNIST (b) CIFAR-10

图 3 IID 场景下的指标评估

在这种情况下, 对于 IID 场景, 我们的模型测试在 MNIST 和 CIFAR-10 数据集下分别获得了 *Acc* 为 97.83% 和 71.31%; 对于 non-IID 场景, 模型测试阶段分别获得了 *Acc* 为 97.26% 和 69.07%. 无论是 IID 还是 non-IID 场景, 我们的模型在 MNIST 数据集上实现了较高的 *Cpre* 和 *Crec*, 且从混淆矩阵中可以看出, 类标签几乎可以准确分

类. 对于 CIFAR-10 数据集, 我们注意到, 模型测试整体上表现出较好的性能, 混淆矩阵也展示出图像大体上能够准确分类. 显然, MNIST 上获得了更优的效果, 主要原因是 CIFAR-10 包含大量的彩色图像, 拥有更复杂的本地特征模式, 使得训练获得更优的联邦模型更加困难. 结果表明即使面向更具复杂特征模式的 CIFAR-10 数据集, 我们的模型仍然能够获得可行的准确度, 这也进一步反映出在 CEFL 框架下执行 FL 任务具有一定的可行性.

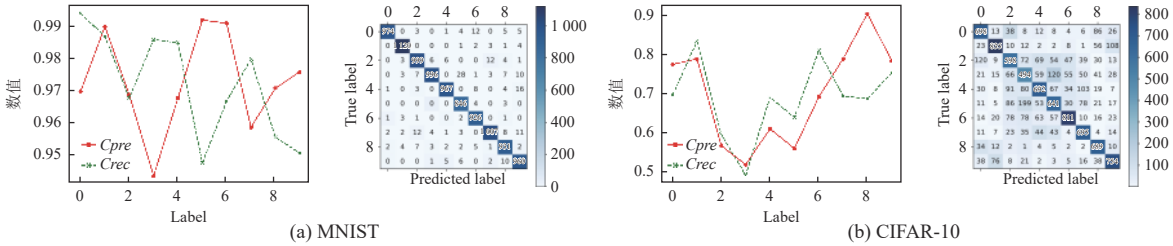


图 4 non-IID 场景下的指标评估

4.2.2 离散更新空间 vs. 连续更新空间

接下来, 我们对比了 IID 场景下基于连续更新空间在不同的 b 下的防御效果, 并采用 *Acc* 和 *ASR* 指标进行评估, 实验结果如表 2 所示.

表 2 不同更新空间下的 *Acc* 和 *ASR* 对比 (%)

数据集	更新空间	b = 10%		b = 20%		b = 30%		b = 40%	
		Acc	*ASR*	*Acc*	*ASR*	*Acc*	*ASR*	*Acc*	*ASR*
MNIST	连续	**99.17**	**0.20**	**99.16**	**0.20**	**99.10**	**0.27**	**99.10**	**0.31**
	离散	96.61	0.56	94.17	0.22	96.72	2.47	93.97	1.12
CIFAR-10	连续	**75.99**	12.40	**75.90**	14.90	**75.73**	**14.50**	**75.05**	**14.90**
	离散	63.06	**6.90**	60.56	**13.00**	61.25	20.60	60.85	23.10
Fashion-MNIST	连续	**90.33**	**4.20**	**90.19**	**4.00**	**90.59**	**4.40**	**90.46**	**4.60**
	离散	86.05	8.90	85.87	8.30	81.83	9.50	71.18	12.40

从表 2 可以观察到, 对于 MNIST 数据集, 随着 b 的增加, 在离散更新空间下获得了 *Acc*, 均高于 93.00%. 比较连续更新空间, 离散更新空间获得了略低的 *Acc*, 下降幅度为 2.38%–5.13%, 同时也实现了略高的 *ASR*, 增加幅度为 0.02%–2.20%; 对于 CIFAR-10 数据集, 在离散更新空间下实现了 *Acc*, 均高于 60.00%. 比较连续更新空间, 离散更新空间实现了略低的 *Acc*, 下降幅度为 12.93%–15.34%, 特别地, 在 b = 10% 和 b = 20% 时, *ASR* 分别下降了 5.50% 和 1.90%. 出现上述现象主要是因为连续空间给攻击者提供了更多选择模型更新并在每次更新中快速定位符合自己需求的最优解的机会, 而离散空间大大缓解了这一点, 甚至需要更多轮迭代来准确捕捉到最优解; 对于 Fashion-MNIST 数据集, 在离散更新空间下实现了 *Acc*, 均高于 80.00%, 除了 b = 40% 时获得了 *Acc* 71.18%. 同样地, 比较连续更新空间, 离散更新空间实现了略低的 *Acc* 和 *ASR*, 下降幅度分别为 4.28%–19.28% 以及 4.30%–7.80%. 综上, 就防御投毒攻击而言, 结果揭示了 FedDiscrete 引入离散更新空间实施防御的思想具有一定的可行性, 但相较于连续更新空间, 其在一定程度上弱化了模型性能, 这也进一步例证了采用离散更新空间的特点, 同时也反映出 CIFAR-10 数据集具有更强的敏感性. 通过对 *Acc* 和 *ASR* 的考虑, 我们推测当恶意攻击者数量不断增加时, FedDiscrete 仍然能够表现出较好的防御性能.

4.2.3 不同防御算法下的评估

然后, 在 IID 和 non-IID 场景下, 基于 *Acc* 和 *ASR* 指标对比了 FedDiscrete 与其他 4 种经典的防御算法在不同 b 值下的效果. 特别地, 为了公平性, 我们首先对 FedDiscrete 方法执行离散更新空间的消融实验, 即基于连续更新空间并兼顾攻击者搭便车来执行 *Acc* 比较. 结合第 4.2.2 节的结论“CIFAR-10 更具敏感性”, 因此, 这里只报告 CIFAR-10 下的防御性能, 结果展示在图 5 中. 其中, 图 5(a) 和 (b) 为 IID 场景设置, 图 5(c) 和 (d) 为 non-IID 场景设置.

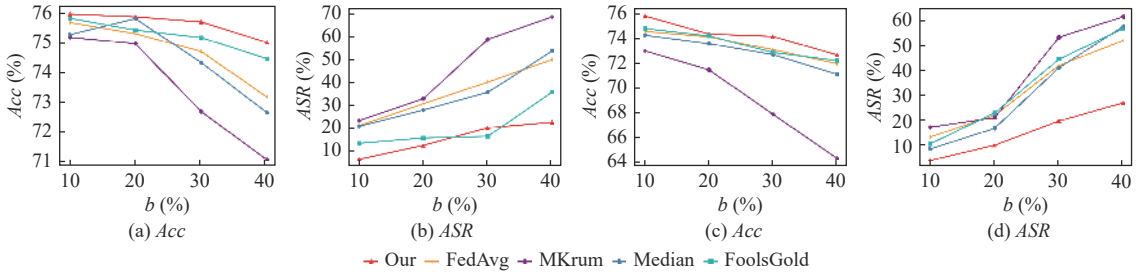


图 5　不同防御算法在 CIFAR-10 数据集下的 *Acc* 和 *ASR* 比较

我们注意到, 在所有比较的防御方法中, 随着 b 的增加, FedDiscrete 防御模型在 non-IID 场景下均获得了最高的 *Acc* 和最低的 *ASR* , 表现出最好的防御性能. 例如, 在 $b=40\%$ 时, FedDiscrete 实现了 *ASR* 27.60%, 明显低于 FedAvg, MKrum, Median 和 FoolsGold 防御所获得的 *ASR* , 分别为 52.40%, 61.80%, 58.00% 和 57.10%, 类似地 *Acc* . 而且, 在 IID 场景下, FedDiscrete 仍然获得了较好的防御效果, 除了在 $b=30\%$ 时, FedDiscrete 有第 2 好的 *ASR* 20.60%, 略高于 FoolsGold 防御实现的 *ASR* 16.90%, 仅增加了 3.70%, 在这种情况下, 即使从 $b=10\%$ 到 $b=40\%$, 我们的防御方案实现了 *ASR* , 仅浮动了 16.20%, 以及 *Acc* 浮动了 0.94%.

而且, 从图 5(a) 和 (d) 中也可以观察到, 无论是在 IID 还是 non-IID 场景下, MKrum 均获得了最差的 *Acc* 和 *ASR* , 因此, 我们可以视为 MKrum 在防御数据投毒攻击方面不是有效的. FedAvg 和 Median 在 IID 场景下实现了相当的结果, 但 *ASR* 明显弱于我们的防御方案. 在这 4 种对比的防御方法中, FoolsGold 在 IID 场景下表现最好, 且在 $b=30\%$ 时实现了较低的 *ASR* , 略优于 FedDiscrete 方法, 然而, FoolsGold 在 non-IID 场景下表现出较差的防御效果, 这表明 FoolsGold 在检测并应对投毒攻击方面可能不是可行和有效的. 综上可知, 结果表明 FedDiscrete 在抵抗投毒攻击方面具有一定的有效性, 同时也反映出兼顾攻击者搭便车以及采用离散更新空间实施防御的策略是可行和有效的, 这也进一步强调了第 4.2.2 节中的结论. 此外, 这个结果也支持了我们在第 4.2.2 节中的推测.

4.2.4　不同攻击者数量下的评估

在不同的攻击者占比 b (10%, 20%, 30% 和 40%) 下, 我们首先展示了 IID 场景下 FedDiscrete 在 MNIST 和 CIFAR-10 数据集上的防御性能, 并采用 *Acc* , *Cpre* , *Crec* , *F*1 和 *ASR* 指标进行评估. 值得注意的是, 本节引入了 $b=0$ 的情况作为对比基准, 并使用 $Cpre_{tc}$, $Crec_{tc}$ 和 $F1_{tc}$ 指标集中于评估目标标签的测试性能.

对于 MNIST, 如图 6(a) 所示, 随着 b 的增加, 目标标签的类精度 ($Cpre_{tc}$) 和类召回率 ($Crec_{tc}$) 明显下降, *Acc* 和 $F1_{tc}$ 在 $b=20\%$ 和 30% 时出现反向增长, 主要原因在于每一轮随机选择的攻击者并非相同, 局部模型训练受到不同程度的攻击影响, 使得每一轮的离散更新空间及参与者的贡献度存在差异. 尽管 b 不断增加, 但 FedDiscrete 仍然实现了 90% 以上的 *Acc* , $Crec_{tc}$ 和 $F1_{tc}$, 特别地, 除了在 $b=40\%$ 时实现了 $Cpre_{tc}$ 88.00%, 其表现出同样的性能. 同时可以观察到, 不同 b 下实现的 *Acc* 均弱于没有发起投毒攻击 ($b=0$) 的情况下所获得的 *Acc* 97.83%.

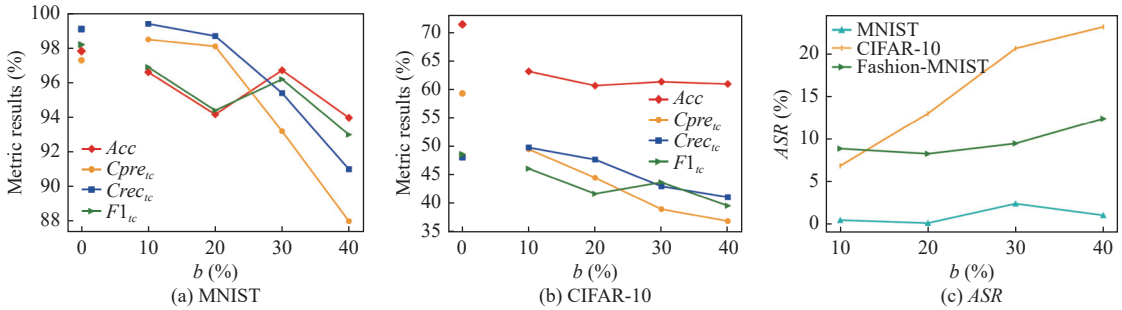


图 6　不同 b 下的指标对比

对于 CIFAR-10, 如图 6(b) 所示, b 的增加对 Acc 没有显著的影响, 其始终稳定在 60.56%–63.06% 之间, 但显然低于 $b=0$ 时获得的 Acc 71.31%. 同时, 随着 b 从 10% 增加到 40%, $Cpre_{tc}$, $Crec_{tc}$ 和 $F1_{tc}$ 均呈现下降趋势, 甚至在 $b=40\%$ 时实现了 $Cpre_{tc}$ 36.80%, $Crec_{tc}$ 41.00% 和 $F1_{tc}$ 39.50%. 实验结果表明恶意攻击者数量的增加对 FL 子任务的性能有着显著的影响, 甚至难以准确识别出目标标签, 这也为进一步开发更鲁棒的防御机制提供思路.

ASR 评估: 然后, 我们采用 *ASR* 指标分析 FedDiscrete 防御在 3 种数据集下的性能. 具体而言, 从图 6(c) 中可以发现, 随着 b 的增加, 比较 CIFAR-10, FedDiscrete 在 MNIST 数据集下求解的 *ASR* 变化幅度较小, 仅为 2.25%, 并稳定在 1.09% 左右, 在 Fashion-MNIST 数据集下同样实现了较小的变化幅度 4.10%. 相反, 在 CIFAR-10 下实现了 16.20% 的变化幅度, 即从 6.90% 增加到 23.10%, 结合第 4.2.3 节, 我们知道 FedDiscrete 仍然实现了较低的 *ASR*. 结果表明我们的 FedDiscrete 方案应用在 IID 场景下具有明显的优势.

混淆矩阵评估: 为了更直观地展示 FedDiscrete 在不同 b 下的防御性能, 我们采用混淆矩阵进行评估. 从图 7 和图 8 中可以看到, FedDiscrete 在 MNIST 数据集上总是保持着更好的分类效果; 而在 CIFAR-10 数据集下获得了相对大的分类误差, 但大体上仍能够识别, 这主要是因为 CIFAR-10 虽然表现出较强的敏感性, 但 FedDiscrete 更擅长学习本地更新的关键特征模式, 特别是在分类任务中. 而且, 随着 b 的增加, 两种数据集下的模型分类效果呈现下降趋势.

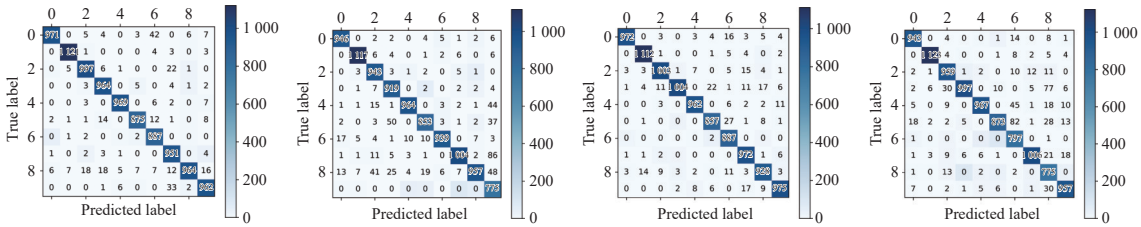


图 7 在 MNIST 数据集上不同 b 下混淆矩阵评估

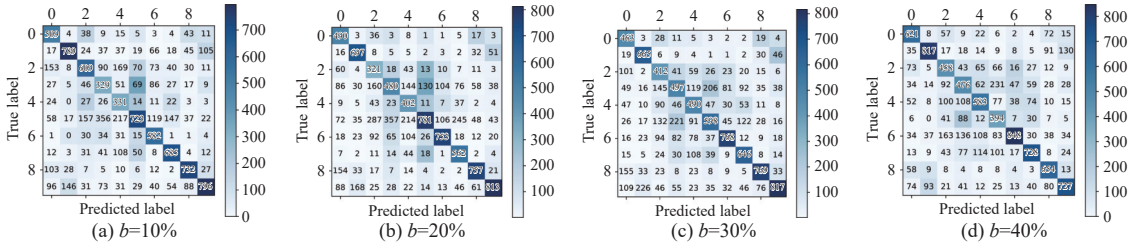


图 8 在 CIFAR-10 数据集上不同 b 下混淆矩阵评估

综上所述, 实验结果表明, 即使面临不同的攻击者数量造成的动态的攻击能力, 我们的防御方案 FedDiscrete 仍能够展示出较强的鲁棒性, 这也反映出兼顾模型更新的本地特征模式和借助参与者的贡献大小权衡公平性的策略是可行的, 有利于提高防御效果. 此外, 这个结果也进一步证实了我们在第 4.2.2 节的推测.

4.2.5 不同攻击场景下的评估

在这个部分中, 我们调查了 FedDiscrete 在不同攻击场景下的性能, 并进一步与现有的防御方法进行比较.

1) *Acc* 评估: 表 3 给出了 FedDiscrete 和现有其他防御方法的比较, 包括 Krum, FLTrust, Trimmed-mean, Auror 和 PEFL. 通过执行相同的迭代轮数 ($R=100$), 探讨 MNIST 数据集下不同的防御方法在 $b=50\%$ 时的 *Acc* 效果, 其中, 我们的方法在 IID 场景下仅执行迭代轮数 $R=50$. 可以注意到, 在 IID 和 non-IID 设置下, 我们的防御方法均获得较好的 *Acc*, 分别为 88.34% 和 78.99%. 具体来说, 对于 Trimmed-mean, 在 non-IID 下实现了 *Acc* 47.0%, 比较我们的方法, 其下降了 *Acc* 31.99%, 这主要是因为该防御易于遭受攻击者数量的影响; 对于 Krum, 其在两种设置下均表现欠佳; 对于 FLTrust 和 Auror, 这两个方法分别表现出极端的结果, 例如, FLTrust 在 IID 设置下表现出最

差的性能, *Acc* 为 75.1%, 虽然 Auror 在 IID 场景下获得了最优的 *Acc* 89.2%, 略高于我们的方法 0.86%, 但其在 non-IID 下实现了最差的性能 *Acc* 为 33.5%; 对于 PEFL, 虽然在 IID 下与 FedDiscrete 方法表现出相当的性能, 仅低于我们的方法 0.14%, 但其在 non-IID 下表现出较弱的性能, 主要原因在于 FLTrust 对每个客户端本地模型更新分配信任分数, 易于导致攻击者有意规避特定轮的检测, 且 Krum, Auror 和 PEFL 都难以有效识别出 non-IID 下的投毒梯度与易发散的良性梯度. 此外, 在 non-IID 场景下, 比较其他防御算法, FedDiscrete 提高了 *Acc* 17.49%–45.49%, 例如, FedDiscrete 获得 *Acc* 78.99%, 而 Auror 方案仅为 *Acc* 33.5%, 类似地 IID 场景.

表 3 *b*=50% 时不同防御算法在不同攻击场景下的 *Acc* 比较 (%)

场景	Our	Trimmed-mean[50]	Krum[48]	FLTrust[10]	Auror[51]	PEFL[30]
IID	88.34±0.46	82.3	77.5	75.1	89.2	88.2
non-IID	78.99±2.67	47.0	61.5	37.8	33.5	46.4

此外, 我们在 IID 场景下比较了两种新颖的防御方法. 第 1 个是评估 RobustFL 防御在 MNIST 数据集上的性能. 具体地, RobustFL 在 b = 10% 和 20% 时分别实现了 *Acc* 94.17% 和 91.86%, 弱于我们的方法所获得的 *Acc* 96.61% 和 94.17%; 另一个是评估利用离散更新空间的防御方法 FRL 在 CIFAR-10 数据集上的性能, 实验结果如表 4 所示 (粗体表示更优的结果). 从表 4 中可以看到, 在无攻击者场景下, FedDiscrete 防御实现了 *Acc* 71.31%, 弱于 FRL 实现了 *Acc* 77.60%. 然而, 随着攻击者数量的增加, 当 b = 10% 时, FRL 实现了 *Acc* 41.70%, 其表现出大幅度的下降, 而 FedDiscrete 实现了 *Acc* 63.06%, 具有较小的下降幅度; 类似地, 当 b = 20% 时, FRL 仍然实现了较低的 *Acc* 39.70%, 而 FedDiscrete 获得了 *Acc* 60.56%. 进一步地, 我们也比较了 b = 10% 场景下 3 种同样采用离散更新空间的防御方法的性能, 结果如表 5 所示. 从表 5 中可以看到, 在 non-IID 场景下, 文献 [23−25] 均获得了较低的 *Acc* , 而所提出的 FedDiscrete 防御方法实现了较高的 *Acc* 96.15%.

表 4 在 IID 场景和 CIFAR-10 数据集上不同 b 值下防御算法的性能评估 (%)

b	Our	FRL[27]
0	71.31±0.33	**77.60**
10	**63.06**±0.53	41.70
20	**60.56**±0.82	39.70

表 5 b=10% 时在 non-IID 场景和 MNIST 数据集上不同防御算法下的 *Acc* 评估 (%)

算法	Our	DDPG[23]	DQN[24]	SLM[25]
Acc	96.15±1.05	65.89	52.25	57.75

综上可知, 结果表明即使面向存在半数攻击者的 IID 和 non-IID 场景, 我们的防御方法 FedDiscrete 仍表现出了较强的可行性和有效性, 而且, 我们也可以猜测当 b 逐渐增加至接近攻击者的上限时, 防御能力可能是有限的, 同时反映出兼顾攻击者搭便车在一定程度上能够极大地提高防御性能.

2) FedDiscrete vs. CONTRA: 然后, 基于 *Acc* 指标, 我们评估了不同 b 下 FedDiscrete 和 CONTRA 防御方案在 MNIST 数据集下的性能, 实验结果如表 6 所示.

表 6 在 MNIST 数据集上不同 b 下的 *Acc* 比较与 CONTRA 方案 (%)

场景	方案	b = 10%	b = 20%	b = 30%	b = 50%
IID	Our	**96.61**±0.16	**94.17**±0.51	**96.72**±0.34	**88.34**±0.46
	CONTRA [52]	85.44	85.22	83.20	82.34
non-IID	Our	**96.15**±1.05	**92.76**±0.96	**92.60**±1.89	**78.99**±2.67
	CONTRA [52]	73.92	73.22	72.82	70.42

从表 6 中观察到, FedDiscrete 的防御性能受到 b 值的影响, 且随着 b 的增加呈现下降趋势. 比较 CONTRA 方案, 无论是在 IID 还是 non-IID 场景下, 即使面对不同的攻击能力, 我们的方法仍然获得了更好的 *Acc* . 而且, 在 IID 下, 当 b = 30% 时, CONTRA 实现了 *Acc* 83.20%, 比较我们的方法, 其下降了 13.52%; 类似地, 在 non-IID 下, 我们的方法在 b = 10% 时获得了更优的 *Acc* 96.15%, 比较 CONTRA 防御提高了 22.23%. 我们认为 CONTRA 方案失败的原因可能是其忽略了对局部模型参数的特征模式的分析和选择, 另一个原因可能是其引入的数据分布超参数

的比例 $\alpha \in (0.05, 1000)$, 未能清晰划分两种攻击场景的界限. 结果例证了即使在 non-IID 攻击场景下, FedDiscrete 防御仍然是有效的和鲁棒的, 并优于先进的解决方案.

3) 不同指标下 FedDiscrete 性能评估: 为了公平, 我们也给出了 non-IID 场景下 FedDiscrete 在 MNIST 和 CIFAR-10 数据集上的指标评估. 注意, 这里仍然是采用 Acc, $Cpre$, $Crec$, $F1$ 和 ASR 指标, 其中, 图 9(a) 的结果是在 MNIST 数据集上实现, 图 9(b) 的结果是在 CIFAR-10 上实现, 最后, 图 9(c) 提供了不同 b 下 ASR 的变化.

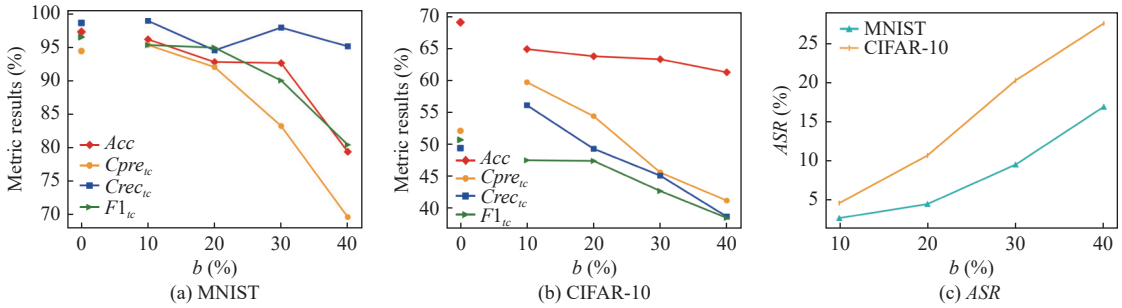


图 9 在 non-IID 场景不同 b 下的指标评估

首先, 如图 9(a) 所示, 随着 b 的增加, Acc, $Cpre_{tc}$ 和 $F1_{tc}$ 以不同的幅度下降, 而 $Crec_{tc}$ 在 $b = 20\%$ 和 $b = 30\%$ 时出现了反向增长, 但从 $b = 10\%$ 到 $b = 40\%$ 整体上仅下降 3.80%. 其次, 在图 9(b) 中, 我们发现 b 的增加使得所有指标均以不同的幅度不断下降. 具体地, Acc 浮动相对平缓, 从 $b = 10\%$ 时实现的 64.88% 到 $b = 40\%$ 的 61.26%, 仅下降 3.62%. $Cpre_{tc}$ 和 $Crec_{tc}$ 表现相似, 均呈现大幅度下降趋势, $F1_{tc}$ 有相对较小的下降幅度. 相同的是, $Cpre_{tc}$, $Crec_{tc}$ 和 $F1_{tc}$ 在 $b = 40\%$ 时均获得了 40.00% 左右的结果, 这证实了第 4.2.4 节 1) 中的猜测. 综上, 相较于第 4.2.4 节描述的 IID 场景而言, 我们发现 FedDiscrete 防御在 non-IID 下的表现更多样化和复杂化, 这也与 non-IID 场景具有数据多样性和攻击形式多样性的特点相关. 特别地, 比较无攻击者投毒的场景, $Cpre_{tc}$ 和 $Crec_{tc}$ 在 $b = 10\%$ 时都实现了更好的性能, 这主要是因为仅存在少数攻击者时, 检测并防御攻击者是可能的, 这也体现了 FedDiscrete 防御的有效性. 接下来, 图 9(c) 展示了 b 的增加导致 ASR 也不断增长, 尤其是 CIFAR-10 数据集, 其从 $b = 10\%$ 到 $b = 40\%$ 实现了从 4.60% 到 27.60% 的明显增长, 而 FedDiscrete 在 MNIST 数据集下实现了 ASR 从 2.69% 到 16.93% 的增长.

4) PathMNIST 数据集下的性能评估: 此外, 基于 Acc, $Cpre$, $Crec$ 和 $F1$ 这 4 个指标, 我们也评估了 IID 场景下 FedDiscrete 方法在 PathMNIST 数据集上的性能, 实验结果如表 7 和图 10 所示.

表 7 不同 b 下的 FedDiscrete 性能评估 (%)

b	Acc
0	73.05
10	71.69
20	69.12
30	70.00
40	63.73

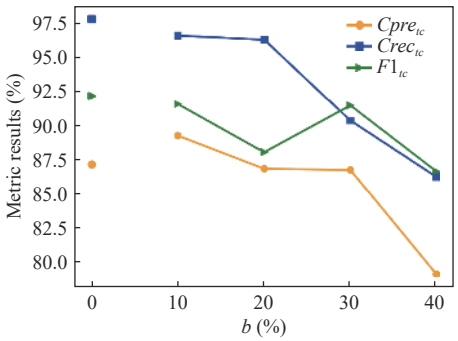


图 10 不同指标下的 FedDiscrete 性能评估: IID

从表 7 中可以观察到, 在没有攻击者发起投毒攻击的情况下, FedDiscrete 实现了最高的 Acc 73.05%, 且防御性能随着攻击者数量的增加而呈现下降趋势. 具体地, 当 $b = 10\%$ 时, FedDiscrete 算法在 IID 场景下实现了 Acc

71.69%, 甚至在 $b = 40\%$ 时也获得了 Acc 63.73%, 比较 CIFAR-10 数据集, FedDiscrete 获得了更优的防御效果. 结果表明 FedDiscrete 利用离散更新空间在 PathMNIST 数据集上实施防御仍然是有效的. 同时, 我们也使用 $Cpre_{tc}$, $Crec_{tc}$ 和 $F1_{tc}$ 指标来进一步验证 FedDiscrete 在子任务 (目标标签类“0”) 上的测试性能. 从图 10 中可以看到, 在不同的 b 值下, FedDiscrete 在子任务上均实现了较高的 $Cpre_{tc}$, $Crec_{tc}$ 和 $F1_{tc}$, 且均高于 85.00%, 除了当 $b = 40\%$ 时获得了 $Cpre_{tc}$ 79.20%. 特别地, $F1_{tc}$ 在 $b = 20\%$ 和 $b = 30\%$ 时出现了反向增长, 我们分析可能的原因是, 当攻击者的数量增加到一定程度时, 投毒行为将更易于被检测. 综上, 结果例证了我们的防御方法 FedDiscrete 在抵抗目标投毒攻击方面具有较强的鲁棒性, 也进一步表明 FedDiscrete 方法适用于分布式的云边缘计算场景.

总体来看, 我们的 FedDiscrete 防御在 MNIST, CIFAR-10 和 PathMNIST 数据集上总是表现出较好的性能. 我们认为少数的恶意攻击者对目标标签的干扰是微弱的, 这也突出 FedDiscrete 优先捕捉更健壮的参数特征实施防御的必要性. 此外, 上述实验进一步例证了 FedDiscrete 引入参与者贡献度到离散更新空间的策略是有效的, 且适用于 IID 和 non-IID 场景并表现出显著的优势, 这充分表明该策略能够作为一个有效的基准防御投毒攻击, 并为 CEFL 系统下执行 FL 任务的过程移除更多的恶意攻击者和恶意更新, 以增强系统安全性.

4.2.6 计算代价和通信开销

最后, 我们讨论并评估了在 CEFL 系统下 FedDiscrete 防御算法的计算代价和通信开销. 我们方案的局限性主要体现在客户端求解参与者贡献度及创建离散更新空间的计算成本, 上传客户端本地排名到边缘服务器及云服务器的通信开销, 边缘端和云端执行多数投票聚合的计算成本这 3 个方面, 详细介绍展示在第 3.3 节. 注意到, 对于每个客户端而言, 其计算成本和通信开销与全局迭代轮数是线性的.

此外, 我们在 MNIST, CIFAR-10 和 PathMNIST 数据集下运行了实验以评估我们的防御方案在不同攻击场景下的计算代价和通信开销, 结果展示在表 8. 注意, 实验结果展示了一个通信轮的情况. 就计算代价而言, 从表 8 中可以看到 3 种数据集下均获得了较低的结果, 具体地, 比较 MNIST, 在 IID 场景下, CIFAR-10 和 PathMNIST 均需要更长的时间, 这主要是由于越复杂的输入数据的特性和网络结构将迫使客户端本地训练及服务器端的聚合操作消耗更多的时间, 比如, 需要花费更长的时间计算本地边排名, 类似地, 在 non-IID 场景下的 CIFAR-10 数据集. 而且, 可以注意到, 当采用特定数据集训练本地模型并上传离散更新空间后, FedDiscrete 防御的计算代价几乎不受攻击场景的干扰. 就通信开销而言, 我们发现 MNIST, PathMNIST 和 CIFAR-10 数据集之间存在一定的差异, 这主要依赖于 CIFAR-10 具有更复杂的图像模式以及采用包含更多参数的 Conv8 网络结构的事实, 从而导致模型训练和特征学习相对困难. 进一步地, 可以发现通信开销与攻击场景之间没有直接的关联, 这是因为通信开销主要取决于模型的结构和大小, 越大的模型需要上传更大维度的离散更新空间到服务器端. 具体而言, 在 MNIST, PathMNIST 和 CIFAR-10 数据集下, 我们的防御方案能够实现平均每个客户端的通信开销仅为 3.175M, 3.157M 和 10.304M, 就我们所关心的而言, 即防御抵抗 CEFL 系统下的投毒攻击, 这是完全可行的. 结果表明我们的防御方法能够在不牺牲联邦模型准确性的前提下, 花费较小的计算成本和通信开销.

表 8 FedDiscrete 的计算代价和通信开销

指标	场景	MNIST	CIFAR-10	PathMNIST
计算代价 (s)	IID	2.699	8.551	7.341
	non-IID	3.112	8.584	—
通信开销 (M)	IID non-IID	3.175	10.304	3.157

5 总 结

在本文中, 我们探索了 CEFL 系统所面临的投毒攻击问题, 提出了一种新颖的防御机制 FedDiscrete. 其关键思想是构建离散更新空间, 实现对攻击者选择更多模型更新的限制. 基于此, 我们在客户端依赖于网络模型连接边的分数来计算每个客户端的本地排名, 以搜索子网络模型、实现离散更新空间的创建并降低通信成本. 此外, 我们在

客户端引入了贡献度指标来评估每个参与者的真实贡献大小, 尽可能避免攻击者搭便车的情况. 通过这样做, 不仅削弱了攻击者的贡献大小以确保公平性, 而且降低了其制造恶意可用更新空间的可能, 从而缓解攻击者对全局性能的影响. 广泛的实验结果表明, FedDiscrete 是一种有效且鲁棒的防御方案, 面向动态的攻击能力、不同的攻击形式和攻击场景都能展现出较强的优势. 我们相信, 本文的工作有助于加深对 CEFL 实际场景下的投毒攻击与防御策略的理解, 在未来工作中, 我们将 (1) 利用数据分布的特性设计针对特定数据集优化的离散更新空间方法, 以提高模型训练效率; (2) 从数据集的特征复杂度和防御方法的粒度等方面出发, 探索一个能够在离散更新空间下对更复杂的本地特征模式的数据集具有鲁棒性的防御方法, 为进一步开发更高效、鲁棒的防御检测技术并应用于更广泛的领域提供新思路.

References:

[1] Tong X, Zhang Z, Jin CQ, Zhou AY. Blockchain for end-edge-cloud architecture: A survey. Chinese Journal of Computers, 2021, 44(12): 2345–2366 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.02345]

[2] Shi L, Shu JG, Zhang WZ, Liu Y. HFL-DP: Hierarchical federated learning with differential privacy. In: Proc. of the 2021 IEEE Global Communications Conf. (GLOBECOM). Madrid: IEEE, 2021. 1–7. [doi: 10.1109/GLOBECOM46510.2021.9685644]

[3] McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. 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.

[4] Tyagi S, Rajput IS, Pandey R. Federated learning: Applications, security hazards and defense measures. In: Proc. of the 2023 Int'l Conf. on Device Intelligence, Computing and Communication Technologies (DICCT). Dehradun: IEEE, 2023. 477–482. [doi: 10.1109/DICCT56244.2023.10110075]

[5] Li XJ, Wu GW, Yao L, Zhang WZ, Zhang B. Progress and future challenges of security attacks and defense mechanisms in machine learning. Ruan Jian Xue Bao/Journal of Software, 2021, 32(2): 406–423 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6147.htm [doi: 10.13328/j.cnki.jos.006147]

[6] Li SH, Ngai E, Voigt T. Byzantine-robust aggregation in federated learning empowered industrial IoT. IEEE Trans. on Industrial Informatics, 2023, 19(2): 1165–1175. [doi: 10.1109/TII.2021.3128164]

[7] Ren K, Wang Q, Wang C, Qin Z, Lin XD. The security of autonomous driving: Threats, defenses, and future directions. Proc. of the IEEE, 2020, 108(2): 357–372. [doi: 10.1109/JPROC.2019.2948775]

[8] Xie SY, Yan Y, Hong Y. Stealthy 3D poisoning attack on video recognition models. IEEE Trans. on Dependable and Secure Computing, 2023, 20(2): 1730–1743. [doi: 10.1109/TDSC.2022.3163397]

[9] Ben Saad S, Brik B, Ksentini A. Toward securing federated learning against poisoning attacks in zero touch B5G networks. IEEE Trans. on Network and Service Management, 2023, 20(2): 1612–1624. [doi: 10.1109/TNSM.2023.3278838]

[10] Cao XY, Fang MH, Liu J, Gong NZ. FLTrust: Byzantine-robust federated learning via trust bootstrapping. arXiv:2012.13995, 2022.

[11] Han F, Zhang Y, Zhao M. Defending poisoning attacks in federated learning via loss value normal distribution. In: Proc. of the 26th Int'l Conf. on Computer Supported Cooperative Work in Design (CSCWD). Rio de Janeiro: IEEE, 2023. 1644–1649. [doi: 10.1109/CSCWD57460.2023.10152846]

[12] Zhao LC, Hu SS, Wang Q, Jiang JL, Shen C, Luo XY, Hu PF. Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. IEEE Trans. on Dependable and Secure Computing, 2021, 18(5): 2029–2041. [doi: 10.1109/TDSC.2020.2986205]

[13] Kumar A, Khimani V, Chatzopoulos D, Hui P. FedClean: A defense mechanism against parameter poisoning attacks in federated learning. In: Proc. of the 2022 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). Singapore: IEEE, 2022. 4333–4337. [doi: 10.1109/ICASSP43922.2022.9747497]

[14] Zhang ZS, Li JR, Yu SC, Makaya C. SAFELearning: Secure aggregation in federated learning with backdoor detectability. IEEE Trans. on Information Forensics and Security, 2023, 18: 3289–3304. [doi: 10.1109/TIFS.2023.3280032]

[15] Zhang JL, Ge CP, Hu F, Chen B. RobustFL: Robust federated learning against poisoning attacks in industrial IoT systems. IEEE Trans. on Industrial Informatics, 2022, 18(9): 6388–6397. [doi: 10.1109/TII.2021.3132954]

[16] Qiao FF, Li Z, Kong YB. A privacy-aware and incremental defense method against GAN-based poisoning attack. IEEE Trans. on Computational Social Systems, 2024, 11(2): 1708–1721. [doi: 10.1109/TCSS.2023.3263241]

[17] Zhang ZX, Cao XY, Jia JY, Gong NZ. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In: Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Washington: ACM, 2022. 2545–2555. [doi: 10.1145/3534678.3539231]

[18] Zhao P, Huang HJ, Zhao XH, Huang DY. P[3]: Privacy-preserving scheme against poisoning attacks in mobile-edge computing. IEEE Trans. on Computational Social Systems, 2020, 7(3): 818–826. [doi: 10.1109/TCSS.2019.2960824]

[19] Cao XY, Zhang ZX, Jia JY, Gong NZ. FLCert: Provably secure federated learning against poisoning attacks. IEEE Trans. on Information Forensics and Security, 2022, 17: 3691–3705. [doi: 10.1109/TIFS.2022.3212174]

[20] Ma ZR, Ma JF, Miao YB, Li YJ, Deng RH. ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning. IEEE Trans. on Information Forensics and Security, 2022, 17: 1639–1654. [doi: 10.1109/TIFS.2022.3169918]

[21] Shi ZS, Ding XY, Li FG, Chen YN, Li CR. Mitigation of poisoning attack in federated learning by using historical distance detection. In: Proc. of the 5th Cyber Security in Networking Conf. (CSNet). Abu Dhabi: IEEE, 2021. 10–17. [doi: 10.1109/CSNet52717.2021.9614278]

[22] Zhao YR, Zhang JB, Cao YH. Manipulating vulnerability: Poisoning attacks and countermeasures in federated cloud-edge-client learning for image classification. Knowledge-based Systems, 2023, 259: 110072. [doi: 10.1016/j.knosys.2022.110072]

[23] Al-Maslamani NM, Ciftler BS, Abdallah M, Mahmoud MMEA. Toward secure federated learning for IoT using DRL-enabled reputation mechanism. IEEE Internet of Things Journal, 2022, 9(21): 21971–21983. [doi: 10.1109/JIOT.2022.3184812]

[24] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533. [doi: 10.1038/nature14236]

[25] Liu YN, Li KQ, Jin YW, Zhang Y, Qu WY. A novel reputation computation model based on subjective logic for mobile ad hoc networks. Future Generation Computer Systems, 2011, 27(5): 547–554. [doi: 10.1016/j.future.2010.03.006]

[26] Wang WX, Levine A, Feizi S. Improved certified defenses against data poisoning with (deterministic) finite aggregation. arXiv: 2202.02628, 2022.

[27] Mozaffari H, Shejwalkar V, Houmansadr A. Every vote counts: Ranking-based training of federated learning to resist poisoning attacks. In: Proc. of the 32nd USENIX Conf. on Security Symp. Anaheim: USENIX Association, 2023. 1721–1738.

[28] Zhang JL, Chen JJ, Wu D, Chen B, Yu S. Poisoning attack in federated learning using generative adversarial nets. In: Proc. of the 18th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications and the 13th IEEE Int'l Conf. on Big Data Science and Engineering (TrustCom/BigDataSE). Rotorua: IEEE, 2019. 374–380. [doi: 10.1109/TrustCom/BigDataSE.2019.00057]

[29] Zhao Y, Chen JJ, Zhang JL, Wu D, Blumenstein M, Yu S. Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks. Concurrency and Computation: Practice and Experience, 2022, 34(7): e5906. [doi: 10.1002/cpe.5906]

[30] Liu XY, Li HW, Xu GW, Chen ZQ, Huang XM, Lu RX. Privacy-enhanced federated learning against poisoning adversaries. IEEE Trans. on Information Forensics and Security, 2021, 16: 4574–4588. [doi: 10.1109/TIFS.2021.3108434]

[31] Tolpegin V, Truex S, Gursoy ME, Liu L. Data poisoning attacks against federated learning systems. In: Proc. of the 25th European Symp. on Research in Computer Security. Guildford: Springer, 2020. 480–501. [doi: 10.1007/978-3-030-58951-6_24]

[32] Fraboni Y, Vidal R, Lorenzi M. Free-rider attacks on model aggregation in federated learning. In: Proc. of the 24th Int'l Conf. on Artificial Intelligence and Statistics. San Diego: PMLR, 2021. 1846–1854.

[33] Fung C, Yoon CJM, Beschastnikh I. Mitigating sybils in federated learning poisoning. arXiv:1808.04866, 2020.

[34] Jagielski M, Oprea A, Biggio B, Liu C, Nita-Rotaru C, Li B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In: Proc. of the 2018 IEEE Symp. on Security and Privacy (SP). San Francisco: IEEE, 2018. 19–35. [doi: 10.1109/SP.2018.00057]

[35] Jebreel NM, Domingo-Ferrer J. FL-Defender: Combating targeted attacks in federated learning. Knowledge-based Systems, 2023, 260: 110178. [doi: 10.1016/j.knosys.2022.110178]

[36] Sharipuddin, Purnama B, Kurniabudi, Winanto EA, Stiawan D, Hanapi D, Idris MYB, Budiarto R. Features extraction on IoT intrusion detection system using principal components analysis (PCA). In: Proc. of the 7th Int'l Conf. on Electrical Engineering, Computer Sciences and Informatics. Yogyakarta: IEEE, 2020. 114–118. [doi: 10.23919/EECSI50503.2020.9251292]

[37] Zhao YR, Cao YH, Zhang JB, Huang HX, Liu YH. FlexibleFL: Mitigating poisoning attacks with contributions in cloud-edge federated learning systems. Information Sciences, 2024, 664: 120350. [doi: 10.1016/j.ins.2024.120350]

[38] Ramanujan V, Wortsman M, Kembhavi A, Farhadi A, Rastegari M. What's hidden in a randomly weighted neural network? In: Proc. of the 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). Seattle: IEEE, 2020. 11890–11899. [doi: 10.1109/CVPR42600.2020.01191]

[39] He KM, Zhang XY, Ren SQ, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proc. of the 2015 IEEE Int'l Conf. on Computer Vision. Santiago: IEEE, 2015. 1026–1034. [doi: 10.1109/ICCV.2015.123]

[40] Li XY, Qu Z, Zhao SQ, Tang B, Lu Z, Liu Y. LoMar: A local defense against poisoning attack on federated learning. IEEE Trans. on Dependable and Secure Computing, 2023, 20(1): 437–450. [doi: 10.1109/TDSC.2021.3135422]

[41] Lyu L, Xu XY, Wang Q, Yu H. Collaborative fairness in federated learning. In: Yang Q, Fan LX, Yu H, eds. Federated Learning: Privacy and Incentive. Cham: Springer, 2020. 189–204. [doi: 10.1007/978-3-030-63076-8_14]

[42] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

[43] Krizhevsky A. Learning multiple layers of features from tiny images [MS. Thesis]. Toronto: University of Toronto, 2009.

[44] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747, 2017.

[45] Yang JC, Shi R, Wei DL, Liu ZQ, Zhao L, Ke BL, Pfister H, Ni BB. MedMNIST v2—A large-scale lightweight benchmark for 2D and 3D biomedical image classification. Scientific Data, 2023, 10(1): 41. [doi: 10.1038/s41597-022-01721-8]

[46] Wortsman M, Ramanujan V, Liu R, Kembhavi A, Rastegari M, Yosinski J, Farhadi A. Supermasks in superposition. arXiv:2006.14769, 2020.

[47] Minka TP. Estimating a Dirichlet distribution. 2000. https://tminka.github.io/papers/dirichlet/minka-dirichlet.pdf

[48] Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 118–128.

[49] Yin D, Chen YD, Ramchandran K, Bartlett P. Byzantine-robust distributed learning: Towards optimal statistical rates. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 5650–5659.

[50] Fang MH, Cao XY, Jia JY, Gong NZ. Local model poisoning attacks to Byzantine-robust federated learning. In: Proc. of the 29th USENIX Conf. on Security Symp. USENIX Association, 2020. 1623–1640.

[51] Shen SQ, Tople S, Saxena P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In: Proc. of the 32nd Annual Conf. on Computer Security Applications. Los Angeles: ACM, 2016. 508–519. [doi: 10.1145/2991079.2991125]

[52] Awan S, Luo B, Li FJ. CONTRA: Defending against poisoning attacks in federated learning. In: Proc. of the 26th European Symp. on Research in Computer Security. Darmstadt: Springer, 2021. 455–475. [doi: 10.1007/978-3-030-88418-5_22]

附中文参考文献:

[1] 佟兴, 张召, 金澈清, 周傲英. 面向端边云协同架构的区块链技术综述. 计算机学报, 2021, 44(12): 2345–2366. [doi: 10.11897/SP.J.1016.2021.02345]

[5] 李欣姣, 吴国伟, 姚琳, 张伟哲, 张宾. 机器学习安全攻击与防御机制研究进展和未来挑战. 软件学报, 2021, 32(2): 406–423. http://www.jos.org.cn/1000-9825/6147.htm [doi: 10.13328/j.cnki.jos.006147]

赵亚茹(1996－), 女, 博士生, 主要研究领域为联邦学习, 信息安全, 边缘计算.

曹益皓(1997－), 男, 博士生, 主要研究领域为可信计算, 联邦学习, 信息安全.

张建标(1969－), 男, 博士, 教授, 博士生导师, 主要研究领域为可信计算, 网络安全, 区块链技术.

黄浩翔(1992－), 男, 博士生, 主要研究领域为可信计算, 云计算, 访问控制.