软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007183] [CSTR: 32375.14.jos.007183]

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

联邦原型学习的特征图中毒攻击和双重防御机制*

王瑞锦[1], 王金波[1], 张凤荔[1], 李经伟[2], 李增鹏[3], 陈 厅[2]

[1](电子科技大学 信息与软件工程学院, 四川 成都 610054)

[2](电子科技大学 计算机科学与工程学院, 四川 成都 611731)

[3](山东大学 网络空间安全学院, 山东 青岛 266237)

通信作者: 王瑞锦, E-mail: ruijinwang@uestc.edu.cn

摘 要: 联邦学习是一种无需用户共享私有数据、以分布式迭代协作训练全局机器学习模型的框架. 目前流行的联邦学习方法 FedProto 采用抽象类原型 (称为特征图) 聚合, 优化模型收敛速度和泛化能力. 然而, 该方法未考虑所聚合的特征图的正确性, 而错误的特征图可能导致模型训练失效. 为此, 首先探索针对 FedProto 的特征图中毒攻击, 论证攻击者只需通过置乱训练数据的标签, 便可将模型的推测准确率至多降低 81.72%. 为了抵御上述攻击, 进一步提出双重防御机制, 分别通过全知识蒸馏和特征图甄别排除错误的特征图. 基于真实数据集的实验表明, 防御机制可将受攻击模型的推测准确率提升 1–5 倍, 且仅增加 2% 系统运行时间.

关键词: 联邦学习; 数据异构; 知识蒸馏; 特征图中毒攻击; 双重防御机制

中图法分类号: TP309

中文引用格式: 王瑞锦, 王金波, 张凤荔, 李经伟, 李增鹏, 陈厅. 联邦原型学习的特征图中毒攻击和双重防御机制. 软件学报. http://www.jos.org.cn/1000-9825/7183.htm

英文引用格式: Wang RJ, Wang JB, Zhang FL, Li JW, Li ZP, Chen TN. Feature Map Poisoning Attack and Dual Defense Mechanism for Federated Prototype Learning. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7183.htm

Feature Map Poisoning Attack and Dual Defense Mechanism for Federated Prototype Learning

WANG Rui-Jin[1], WANG Jin-Bo[1], ZHANG Feng-Li[1], LI Jing-Wei[2], LI Zeng-Peng[3], CHEN Ting[2]

[1](School of Information and Software Engineering, University of Electronic Science and technology of China, Chengdu 610054, China)

[2](School of Computer Science and Engineering, University of Electronic Science and technology of China, Chengdu 611731, China)

[3](School of Cyber Science and Technology, Shandong University, Qingdao 266237, China)

Abstract: Federated learning, a framework for training global machine learning models through distributed iterative collaboration without sharing private data, has gained prevalence. FedProto, a widely used federated learning approach, employs abstract class prototypes, termed feature maps, to enhance model convergence speed and generalization capacity. However, this approach overlooks the verification of the aggregated feature maps' accuracy, risking model training failures due to incorrect feature maps. This study investigates a feature map poisoning attack on FedProto, revealing that malicious actors can degrade inference accuracy by up to 81.72% through tampering with the training data labels. To counter such attacks, we propose a dual defense mechanism utilizing knowledge distillation and feature map validation. Experimental results on authentic datasets demonstrate that this defense strategy can enhance the compromised model inference accuracy by a factor of 1 to 5, with only a marginal 2% increase in operational time.

Key words: federated learning; data heterogeneous; knowledge distillation; feature map poisoning attack; dual defense mechanism

联邦学习 (federated learning)[1]因其在数据隐私保护方面的优势而受到广泛的欢迎和应用. 与传统的集中式机器学习 (machine learning) 不同, 联邦学习无需将数据从本地转移至其他不受信的第三方处理中心, 而是将机器学

* 基金项目: 国家重点研发计划 (2022YFB4501200, 2022YFB3304303); 国家自然科学基金 (62271128, 61972073, U2333207); 成都市重点研发支撑计划"揭榜挂帅"项目 (2022-JB00-00013-GX); 四川省科技计划重点研发项目 (2022ZDZX0004, 2023YFG0029, 2023YFG0150, 2022YFG0212, 2021YFS0391); 四川省科技计划"揭榜挂帅"项目 (2023YFG0374, 2023YFG0373); 山东省自然科学基金 (ZR2023MF045)
收稿时间: 2023-09-14; 修改时间: 2024-01-12; 采用时间: 2024-03-26; jos 在线出版时间: 2024-11-20

习模型转移至第三方处理中心进行分析处理, 处理中心也无法通过模型获取到原始数据.

尽管联邦学习在学习效率和隐私保护[2–7]上有着天然的优势, 但是它在模型优化过程中仍然面临着数据异构和模型异构两大挑战. 现已有不少研究尝试去减缓这两大挑战带来的威胁. 针对数据异构挑战, 在局部模型优化过程中为损失函数添加额外的正则项[8,9]是一种常用的解决方案, 它通过约束局部模型的参数更新, 有效地减小参数“漂移”, 使局部模型和全局模型的最优点逼近, 最终达到提高模型准确率的效果. 另一种策略是使用个性化的联邦学习 (personalized federated learning, PFL)[10–12], 即通过全局模型和本地个性化数据为每个终端生成个性化模型. 针对模型异构, 通常的解决方案是使用知识蒸馏 (knowledge distillation, KD)[13–16]. 基于知识蒸馏的联邦学习[17–19]利用公共数据集和本地私有数据集, 将具有不同模型结构的教师模型 (经过预训练的模型, 该模型产生训练数据的软标签以供学生模型学习) 的知识转移至一个学生模型 (该模型利用训练数据和教师模型提供的软标签以及真实标签进行训练), 达到学生模型具备不同教师知识的目的, 实现了联邦学习中知识聚合的要求, 是一种常用于解决模型异构的有效方法.

上述的研究对联邦学习中的数据异构和模型异构问题在不同程度上进行了改善, 但是这些改善通常只针对其中之一. 对此, Tan 等人[20]于 2022 年提出了联邦原型学习方法 (FedProto), 将传统联邦学习中梯度的聚合转化为原型 (原型的概念见第 1.2 节, 原型即为特征图) 的聚合, 消除了终端节点之间数据分布和模型结构差异导致的梯度不对齐, 使模型聚合不依赖于本地模型结构和数据分布, 优化了模型的收敛速度和泛化能力, 提高联邦学习对数据异构和模型异构的容忍度. 本质上, FedProto 将模型梯度聚合转化为特征图聚合, 支持终端节点的本地模型优化, 同时受到本地数据分类任务和特征图拟合的两方面影响. 由于本地数据分类任务占据优化目标的主体, 所以终端节点可以在保证本地模型正常收敛的同时, 通过特征图拟合学习并生成更加高效的个性化模型. 由于 FedProto 对于数据和模型异构的容忍性以及学习的高效性, 目前已经被多项研究在临床医疗、元宇宙和自动驾驶等领域进行实验验证.

本文研究发现 FedProto 并未考虑起到关键作用的特征图的正确性, 即终端节点可能受攻击者控制而传递错误的特征图. 为此, 本文探索了一种利用错误特征图的中毒攻击 (feature map poisoning attack, FMPA), 该攻击的目的是使得所有局部模型在整个数据集上的预测准确率下降, 甚至不可用. FMPA 利用特征图难以保证正确性的特点, 控制多个终端节点恶意的生成错误特征图并上传至中心服务器, 以破坏全局特征图的正确性, 最终使局部模型往错误的方向更新. 我们通过实验证明, 利用错误特征图进行中毒攻击的效果会随着中毒特征图的数量增大而变得显著. 具体表现在, 恶意终端节点在每轮通信中都上传错误的特征图, 当累计投入的中毒特征图数量逐渐增多时, 其他终端节点的模型准确率会逐渐降低. 另外, 当恶意终端节点数量变多时, 本地模型准确率也会下降得更快. 我们在 MNIST 数据集上进行了大量实验, 具体为, 当无恶意终端节点时, 20 个终端节点的平均准确为 98.36%; 当有 2 个恶意终端节点且每训练 5 轮投一次毒时, 平均准确率降为 98.06%; 当有 4 个恶意终端节点且每 3 轮投一次毒时, 准确率降为 88.63%; 当有 10 个恶意终端节点且每 1 轮投一次毒时, 准确率降为 16.64%, 相比于不受攻击时的准确率降低了 81.72%.

为了应对特征图中毒攻击给 FedProto 带来的危害, 我们提出了针对特征图中毒攻击的双重防御机制 (defend feature map poisoning attack, De-FMPA). De-FMPA 机制分为两个阶段: 第 1 阶段, 利用全知识蒸馏技术快速筛选出某些恶意终端节点, 这些节点使用中毒特征数据 (经过标签置乱后的数据) 训练本地模型. 全知识蒸馏技术可以通过将局部模型的知识转移到中心服务器的学生模型中, 通过对学生模型的性能和知识进行评估, 可以快速确定哪些终端节点存在恶意行为. 第 2 阶段, 利用特征图甄别技术持续地保证中心服务器接收的特征图的正确性. 在中心服务器上维护了一个用于判别特征图正确性的神经网络模型, 该模型可以对接收到的特征图进行分析和判别, 以识别是否存在中毒特征图. 如果检测到中毒特征图, 则将其丢弃不用. 我们通过实验验证了 De-FMPA 防御机制可以防御 FMPA 攻击. 基于 MNIST 数据集的实验表明, 当有一半的恶意终端节点每 3 轮和每 1 轮投一次毒的情况下, De-FMPA 可以提升模型平均准确率.

本文的主要贡献如下.

(1) FedProto 中, 用于传递知识的特征图难以保证正确性. 对此设计出特征图中毒攻击 FMPA.

(2) 提出了 De-FMPA 双重防御机制, 利用全知识蒸馏和特征图甄别分两阶段进行防御.

(3) 在 MNIST 和 CIFAR10 数据集上的实验表明, FMPA 攻击可以将模型准确率下降至多 81.72%, De-FMPA 可以防御攻击者发起的 FMPA 并提高模型准确率至 5 倍.

1 预备知识

为了方便阅读下文, 我们用表 1 展示了本文所使用的相关符号标记.

表 1 符号表

符号	描述
$C_{i,j}^{r}$	第 r 轮训练中终端节点 i 的数据类别为 j 的本地特征图
$C_{i,j'}^{r}$	第 r 轮训练中终端节点 i 的数据类别为 j 的中毒特征图
$x_{i,j}$	终端节点 i 的类别为 j 的特征数据
$D_{i,j}$	终端节点 i 的类别为 j 所有数据集合
$f_i(\theta_i)$	终端节点 i 的模型的表示层函数
C_{j}^{r}	第 r 轮训练中心服务器聚合来自不同终端节点的所有类别为 j 的全局特征图
$C_{j'}^{r}$	第 r 轮训练中心服务器聚合来自不同终端节点的所有类别为 j 的全局特征图, 其中包括了某些本地中毒特征图.
N_j	所有终端节点中数据类别为 j 的数据集合
V	参与联邦学习的全部终端节点集合
σ_i	终端节点 i 的正则项
MSE	均方误差函数
CE	交叉熵函数
KL	KL散度函数
$F_i(\theta_i)$	终端节点 i 的神经网络模型

1.1 联邦学习和 FedProto

传统机器学习方法[21]要求用户将本地训练数据上传至高算力的中心服务器, 但是这种方法将产生巨大的通信传输开销. 同时中心服务器能够直接获取用户的所有数据, 导致潜在的敏感信息泄漏风险. 为此, 谷歌公司提出了一种分布式机器学习方法——联邦学习[1]. 联邦学习允许分布在不同地理位置的用户 (称为联邦学习的终端节点) 基于其各自的私有数据, 联合训练神经网络模型, 同时确保每个终端节点无需向其他节点 (和中心服务器) 分享其私有数据. 具体的, FL 由一个中心服务器和若干终端节点共同参与, 包含多轮迭代训练过程. 详述任一轮迭代计算过程如下.

(1) 参与节点选择: 中心服务器从所有终端节点中选择部分终端节点参与当前轮次的迭代计算;

(2) 节点本地训练: 每个被选中的终端节点基于本地私有数据 (基于上一轮全局模型参数), 训练本地模型并将本地模型参数上传至中心服务器;

(3) 模型参数聚合: 中心服务器加权聚合来自参与当前轮次计算的所有终端节点的模型参数, 形成当前轮次的全局模型;

(4) 本地模型更新: 中心服务器向各终端节点下发当前轮次聚合的全局模型参数.

联邦学习解决了传统集中式机器学习的通信开销大、隐私数据保护差问题, 但面临跨终端节点的模型异构和数据异构两大不足. 模型异构是指, 由于参与联邦学习的不同终端节点之间的计算和存储资源不同, 导致它们所维护的神经网络模型的深度、每一层网络的维度各异. 中心服务器不能将不同结构的本地模型进行聚合, 往往难以获得精准的全局模型. 数据异构是指不同终端节点的本地数据是非独立同分布 (non-IID) 的, 意味着这些终端节点持有不同的数据标签, 导致所训练出的本地模型存在不同的适用范围. 如果中心服务器将这些本地模型直接进行聚合, 所得到的全局模型对不同终端节点上的数据的预测准确率极低.

为了解决上述不足, Tan 等人[20]提出基于特征的新型异构联邦学习框架 FedProto, 使用特征图替代传统联邦学习的基于模型梯度的聚合方式. 特征图是指原始数据经过多层神经网络的中间层 (例如第 2 层与倒数第 2 层之间的神经网络层) 产生的中间结果. 因此, 即使各个终端节点的模型层数、每一层神经网络的维度存在差异, 只要基于中间层产生的特征图的维度相同, 中心服务器仍然可以成功聚合特征图, 从而解决模型异构问题. 同时, FedProto 只利用与本地特征图的标签一致的全局特征图去计算模型优化的损失函数 (目标函数) 的正则项, 将与本地特征图标签不一致的全局特征图 (这些标签不一致的全局特征图即为异构数据) 舍弃, 避免异构数据对于本地模型的收敛, 所以 FedProto 中数据异构性并不会影响本地模型的正常收敛. 具体的 FedProto 方法正确性的理论分析可以参考文献 [20] 的第 4 节收敛性证明部分. 由于 FedProto 可容忍数据和模型异构性的优势, 得到了学术者和产业者的广泛关注和欢迎. 上述的模型优化的损失函数指的是, 利用模型对特征数据的预测标签和其真实标签计算出距离 (常见的损失函数有交叉熵函数、*KL* 散度、均方误差), 距离越大则预测标签和真实标签越不相似. 我们需要在每一轮训练中降低损失函数的值以优化模型参数.

具体地, 假设完整的神经网络模型 $F(\theta)$, 中间层为 $f(\theta)$, 这里 θ 是模型参数. 同时, 训练数据记为 (x,y), 其中 x 为原始数据, y 为对应的标签, 则模型的最终训练输出为 $F(\theta,x)$, 原始数据 x 的特征图为 $f(\theta,x)$. 详述第 r 轮 FedProto 迭代计算步骤如下.

(1) 终端节点接收第 $r-1$ 轮中心服务器下发的全局特征图.

(2) 终端节点基于本地训练数据为每个标签产生特征图. 具体的, 终端节点的本地训练数据包括特征数据 x 和标签 y, 每个标签都包含大量不同的特征数据, 特征数据经过本地模型 $F(\theta)$ 的中间层 $f(\theta)$ 将会产生对应的特征图 $f(\theta,x)$, 终端节点为每个标签的所有特征数据所对应的特征图计算平均值, 得到第 r 轮的本地特征图. 例如, 第 r 轮中终端节点 i 的标签为 j 的本地特征图为 $C_{i,j}^r$.

(3) 终端节点计算第 $r-1$ 轮全局特征图和本地特征图的均方误差, 以此作为模型优化的损失函数的一个正则项 σ_i. 例如, 终端节点 i 的模型优化的损失函数为 $CE(F(\theta,x),y)+\lambda\sigma_i$, CE 为交叉熵函数, λ 为权重 (λ 值越大, 正则项 σ_i 对于损失函数的影响就越大, 实验中设置为 1).

(4) 中心服务器接受到来自终端节点上传的本地特征图, 为每个标签都加权聚合生成第 r 轮全局特征图. 例如, 中心服务器计算标签为 j 的全局特征图 $C_j^r=\sum_{i\in V}\frac{|D_{i,j}|}{N_j}C_{i,j}^r$. 权值的计算方式为 $\frac{|D_{i,j}|}{N_j}$, 其中 $|D_{i,j}|$ 为终端节点 i 的标签为 j 的数据量, N_j 为全部终端节点的标签为 j 的数据量之和.

FedProto 反复迭代上述步骤至终端节点的本地模型达到收敛状态, 此时全局模型预测准确率达到最优. 除了解决模型异构和数据异构问题以外, FedProto 还具有如下优势: 1) 中心服务器难以通过特征图还原原始训练数据, 从而保护用户数据隐私; 2) 与本地模型参数相比, 特征图的数据规模往往较小, 有助于降低联邦学习的通信开销. 因此, FedProto 成为异构联邦学习的经典基础方法, 后续工作进一步解决其中的异构数据优化方法、全局模型退化、模型过拟合等问题. 例如 Ma 等人[22]针对数据异构的不同联邦学习方法进行调查研究, 得出改进数据异构的方法通常从模型、算法、框架角度出发. Chen 等人[23]基于 FedProto 的原型思想提出适用于面向临床医疗的深度学习算法, 可以有效解决全局模型退化以及数据异构的问题. Sun 等人[24]在终端节点的局部更新中引入代理修正项以保持一致性正则化, 同时结合随机梯度下降和邻近的额外梯度上升减轻局部模型的过拟合.

1.2 数据中毒攻击

数据中毒攻击[25–33]是指攻击者在数据准备阶段有意识地投入不正确或有偏移的数据, 以降低所训练模型的可用性. 联邦学习也面临数据中毒攻击的威胁. 假设参与联邦学习的某些终端节点是恶意的, 这些节点首先在本地构造中毒数据, 然后基于中毒数据训练本地模型. 中心服务器聚合所有终端节点的本地模型参数获得全局模型. 由于恶意终端节点的本地模型及参数均有缺陷, 因此全局模型也存在缺陷. 经过若干轮迭代, 最终训练出具有缺陷的推理模型.

本文考虑通过数据中毒攻击, 降低模型的推测准确率. Nuding 等人[26]设计了基于标签反转的数据中毒攻击方法. 假设恶意终端节点 k 的本地训练数据集为 $D^k=\{x_i,y_i\mid 0<i\leqslant M^k\}$, 其中 x_i 表示 D^k 中第 i 个特征数据, y_i 表示 x_i

对应的正确标签, M^k 表示 D^k 的大小; 一个训练批次的 batch_size=4. 为了确保训练数据不可用, 标签反转方法基于原始训练数据 $\{(x_i, x_{i+1}, x_{i+2}, x_{i+3}), (y_i, y_{i+1}, y_{i+2}, y_{i+3})\}$ 构造不正确的训练数据 $\{(x_i, x_{i+1}, x_{i+2}, x_{i+3}), (y_{i+3}, y_{i+2}, y_{i+1}, y_i)\}$, 错误的训练数据将使得模型准确率降低. 与现有工作不同, 本文提出的针对 FedProto 的 FMPA 攻击将采用随机方式置乱特征图与标签的对应关系.

1.3 知识蒸馏

知识蒸馏 (KD)[13–16]是一种机器学习技术, 旨在通过将一个复杂模型的知识转移到一个简化的模型中, 以减小模型规模同时提高模型准确率. 通常情况下, 复杂模型 (通常称为“教师模型”) 拥有更多的参数和更高的计算能力, 能够更好地捕捉数据的细节和复杂性. 然而, 复杂模型可能在计算资源、存储空间或实时应用等方面存在限制. 这时候, 可以使用知识蒸馏将复杂模型中的知识转移到一个简化模型 (通常称为“学生模型”) 中, 以在资源受限的情况下获得类似的性能. 在知识蒸馏中, 教师模型的知识通过两种方式传递给学生模型.

(1) 软目标: 教师模型的输出不仅包含最终的预测结果, 还包含一种概率分布. 这种概率分布即是教师模型在将特征数据对映射到不同标签的概率或者“软”偏好. 学生模型通过尝试拟合这个“软目标”来学习教师模型的知识.

(2) 知识蒸馏损失: 除了软目标外, 还可以使用额外的损失函数来衡量学生模型的输出与教师模型的输出之间的差异. 这个损失函数通常包括分类损失和模型之间的距离度量, 例如均方差损失.

通过这种传递知识的方式, 学生模型可以从教师模型中学习到更多的信息, 包括类别之间的关系、决策边界和特征表示等. 这有助于提高学生模型的泛化能力和准确率, 尽管学生模型可能比教师模型简化得多.

2 特征图中毒攻击

本节详细介绍特征图中毒攻击. 首先, 根据中心服务器和终端节点再 FedProto 中的行为, 我们构造了本文的威胁模型. 其次, 我们讨论了 FMPA 攻击的理论依据并提供了攻击流程. 最后, 我们通过实验验证了 FMPA 攻击的有效性, 在有 50% 的恶意终端节点时, 最多可以降低 81.72% 的模型准确率.

2.1 威胁模型

在本文中, 我们主要关注特征图的安全漏洞可能会被恶意攻击者利用并造成模型准确率降低这个问题, 因此, 我们做出以下假设.

假设 1. 所有参与联邦学习的终端节点都同意由中心服务器分配调度任务, 终端节点同意中心服务器根据局部特征图聚合成的全局特征图并发布给每个终端节点.

假设 2. 中心服务器是诚实的, 即它不会恶意的生成错误的全局特征图, 它的行为仅是聚合来自终端节点的本地特征图并下发.

和其他关于中毒攻击的研究[25–33]一样, 本文中攻击者的目标是恶意操纵其他终端节点的本地模型, 使其在测试数据上具有高错误率. 这种攻击也成为无目标攻击, 它使得学习的本地模型不可用, 并最终导致用户无法享用服务. 攻击者知道恶意终端节点的本地训练数据集、本地模型以及训练代码, 但是它不能得知其他终端节点的本地训练数据以及模型. 同时, 攻击者知道中心服务器对于特征图的聚合规则. 例如, 中心服务器会公开聚合规则以提高 FedProto 系统的透明度和信任度. 攻击者可以利用已知得聚合规则和可以任意操作本地特征图来进行攻击, 破坏全局特征图得正确性. 我们假设攻击者控制了 k 个终端节点, 即攻击者可以向 FedProto 系统重注入 k 个恶意终端节点, 或者入侵并控制 k 个诚实终端节点. 尽管恶意终端节点会按照协议执行特定的步骤, 但是却无法保证本地训练数据的正确性以及其上传的本地特征图的正确性. 例如, 恶意终端节点可能会在进行本地模型训练之前将训练数据进行标签随机置乱, 使得特征数据和标签之间的对应关系发生错误. 并且恶意终端节点还可能在本地产生中毒特征图并将其上传至中心服务器, 而这些错误的本地中毒特征图将会严重影响由中心服务器聚合的全局特征图的正确性.

2.2 FMPA 攻击

我们在第 1.3 节详细讨论了 FedProto 方法如何利用特征图来应对终端节点的数据异构和模型异构所带来的

挑战, 但是 FedProto 并未充分关注特征图的正确性. 在 FedProto 中, 终端节点 i 的模型优化损失函数为 $L_i = CE(F(\theta,x),y)+\lambda\sigma_i$. 可以观察到, 该损失函数受到两个部分的影响: 一部分由本地训练数据产生的交叉熵损失 $CE(F(\theta,x),y)$, 另一部分为由全局特征图和本地特征图所计算的均方误差正则项 σ_i (λ 是旨在调整正则项对于 L_i 的影响, 以扩大或缩小其作用). 如果正则项 σ_i 相对于 $CE(F(\theta,x),y)$ 较大, 那么 L_i 将主要受正则项的影响, 模型的优化也主要受正则项影响. 反之, 如果正则项 σ_i 相对于 $CE(F(\theta,x),y)$ 较小, 模型的优化也将主要受交叉熵损失影响.

为此, 我们提出 FMPA 攻击. 恶意终端节点随意操纵本地特征图, 使其变得不可靠, 并将这些篡改后的特征图上传至中心服务器. 如果中毒特征图足够多, 那么中心服务器聚合这些特征图时将会产生错误, 进而导致诚实终端节点本地计算出错误的正则项 σ , 从而干扰模型的优化收敛过程. 为了更清晰的展示 FMPA 攻击, 我们将具体论述本地中毒特征图、全局特征图、正则项以及损失函数的计算方式. 我们假设在参与 FedProto 的终端节点中存在恶意终端节点集合 $k\in U$, 再以标签为 j 的特征数据为例, 恶意终端节点 k 在第 r 轮计算出的标签为 j 的本地中毒特征图为:

$$C^r_{k,j'} = \frac{1}{|D_{k,l}|}\sum_{x_{k,l}\in D_{k,l}} f_k(\theta_k, x_{k,l}) \tag{1}$$

其中, l 是除了标签 j 之外的任意标签. 由于标签被置乱, 所以此时计算出的标签为 j 的特征图 $C^r_{k,j'}$ 是中毒特征图 (其中 j' 表示标签为 j 的本地中毒特征图). 为了增强 FMPA 攻击的效果, 恶意终端节点在将中毒特征图上传至中心服务器之前对其数值进行多倍的扩大操作 $C^r_{k,j'} = \eta\times C^r_{k,j'}(\eta\in\mathbb{R})$, 以使其更有效地影响到诚实终端节点. 通过对中毒特征图进行数值扩大, 恶意终端节点可以改变全局特征图的分布, 从而影响到诚实终端节点在本地计算正则项时的结果. 这种增强的 FMPA 攻击可以导致诚实终端节点在模型优化过程中受到更大的干扰, 从而影响模型的收敛性和性能. 其中, η 为一个大于 1 的实数. 实验中, 不同的数据集对应的 η 值也不同 (MNIST 中我们设置 $\eta=3$, CIFAR10 中 $\eta=5$).

本地中毒特征图参与到全局特征图的聚合中, 将会引起全局特征图的计算偏差, 下面以中心服务器在第 r 轮聚合标签为 j 的全局特征图为例.

$$C^r_{j'} = \sum_{k\in U}\frac{|D_{k,l}|}{N_j}C^r_{k,j'} + \sum_{i\in V, i\notin U}\frac{|D_{i,j}|}{N_j}C^r_{i,j} \tag{2}$$

最终, 对于诚实终端节点 i , 利用第 r 轮中心服务器下发的全局中毒特征图和第 $r+1$ 轮本地特征图计算出正则项 $\sigma_i = \sum_j MSE(C^{r+1}_{i,j}, C^r_{j'})$, 该正则项将作为模型优化损失函数的重要一项. 由于恶意终端节点进行中毒特征图放大的操作, 此时正则项的计算结果将会扩大, 并占据诚实终端节点本地模型优化目标的主体, 进而导致模型优化方向产生偏差, 最终导致模型无法收敛甚至不可用. 第 $r+1$ 轮中, 诚实终端节点 i 的本地目标函数为:

$$\begin{cases} L_i = CE(F_i(\theta_i,x_i),y_i) + \lambda\sum_j MSE(C^{r+1}_{i,j}, C^r_{j'}) \\ C^r_{j'} = \sum_{k\in U}\frac{|D_{k,l}|}{N_j}C^r_{k,j'} + \sum_{i\in V,i\notin U}\frac{|D_{i,j}|}{N_j}C^r_{i,j} \\ C^r_{k,j'} = \eta\times\frac{1}{|D_{k,l}|}\sum_{x_{k,l}\in D_{k,l}} f_k(\theta_k, x_{k,l}) \end{cases} \tag{3}$$

算法 1 详细展示了 FMPA 攻击的基本流程. 程序包括了两个函数, *Local_Update*() 函数在终端节点执行, 其输入为终端节点编号 k 、数据集 D_k , 该函数用于终端节点更新模型同时计算本地特征图. *Global_Proto_Agg*() 函数在中心服务器上执行, 其输入为所有终端节点计算出的本地特征图 $C^r_{1,j}, C^r_{2,j}, \ldots, C^r_{k,j}, \ldots, C^r_{|V|,j}$, 该函数用于加权平均聚合所有本地特征图, 计算出全局特征图. 执行顺序应该是, 终端节点先执行 *Local_Update*() (初始阶段全局特征图初始化为 0), 中心服务器后执行 *Global_Proto_Agg*(). 然后每轮训练交替上述执行顺序.

算法 1. 攻击程序: FMPA 攻击程序伪代码.

输入: 恶意终端节点 k 的本地数据集 D_k. 全体终端节点集合 V, 标签为 j 的各个终端节点的本地特征图 $C^r_{1,j}, C^r_{2,j}, \ldots, C^r_{k,j}, \ldots, C^r_{|V|,j}$;
输出: 恶意终端节点 k 计算本地中毒特征图 $C^r_{k,j'}$, 中心服务器聚合本地特征图形成全局特征图.

1. *Local_Update*(k, C^r_j): /*恶意终端节点执行*/
2. **for** each local epoch **do** /*本地训练轮数*/
3. **for** batch $(x_k, y_k) \in D_k$ **do** /*按照批次获取数据训练模型*/
4. Randomly shuffle the correspondence between x_k and y_k. /*数据标签随机置乱*/
5. Computing $C^r_{k,j'}$ according to Eq. (1). /*计算本地中毒特征图*/
6. $C^r_{k,j'} = \eta * C^r_{k,j'} (\eta \in \mathbb{R})$. /*放大本地中毒特征图*/
7. Executing *Global_Feature_Map_Agg*(), Get C^r_j from server. /*获取全局特征图*/
8. Using $C^r_{k,j'}$ and C^r_j to compute σ_k. /*利用全局特征图和本地特征图计算正则项*/
9. Computing loss function L_k. /*计算本地模型损失函数*/
10. Updating model by back propagation according to L_k. /*更新模型*/
11. **end for**
12. **end for**
13. **return** $C^r_{k,j'}$ /*返回本地特征图*/
14. *Global_Feature_Map_Agg*($C^r_{1,j}, C^r_{2,j}, \ldots, C^r_{k,j}, \ldots, C^r_{|V|,j}$): /*中心服务器执行聚合全局特征图*/
15. **for** each global epoch **do** /*全局训练轮数*/
16. **for** each client *i* in parallel **do** /*多客户端并行执行*/
17. Executing *Local_Update*() and pass $C^r_{i,j}$ to server. /*计算本地特征图并上传*/
18. **end for**
19. Aggregating $C^r_{i,j}$ to $C^r_{j'}$, send it to all user. /*聚合本地特征图得到全局特征图*/
20. **end for**
21. **return** $C^r_{j'}$ /*返回全局特征图*/

2.3 实验配置

1) 数据集: 我们使用两个标准数据集 MNIST[34]、CIFAR10[35]对 FMPA 攻击进行测试, 两个数据集的全部特征数据将会按照非独立同分布的设定分布在所有终端节点上以供训练模型.

MNIST 包含 10 个标签分类, 一共有 60 000 个训练样本和 10 000 个测试样本, 每个类别有 6 000 个训练样本, 1 000 个测试样本, 每个样本都是一个大小为 28×28 的手写数字灰度图片.

FashionMNIST 包含 10 个标签分类, 一共有 60 000 个训练样本和 10 000 个测试样本, 每个类别有 6 000 个训练样本, 1 000 个测试样本, 每个样本都是一个大小为 28×28 的单通道服装灰度图片.

CIFAR10 包含 10 个标签分类, 一共有 50 000 个训练样本和 10 000 个测试样本, 每个类别有 5 000 个训练样本, 1 000 个测试样本, 每个样本都是一个大小为 32×32 的三通道彩色图片.

2) 模型: 对于 MNIST 数据集, 我们使用 CNNMNIST[20]模型进行学习, 该模型一共有 2.2 万参数量. 对于 CIFAR10 数据集, 我们使用 ResNet18[36]模型, 深度为 18 层, 一共 1170 万参数量.

3) 实验细节: 我们实现了 FedProto 的基本配置, 我们用虚拟机模拟了 30 个终端节点作为联邦学习的参与者, 其中编号为 21–30 的终端节点为恶意的投毒终端节点. 我们为每个终端节点分配了非独立同分布的数据[20], 将 MNIST、CIFAR10 中的不同标签数据分别部署在不同的终端节点上. 我们使用 CNNMNIST、ResNet18 模型去对

上述两种数据集进行学习, 学习率设定为经验值 0.001. 为了实现神经网络模型的异构场景, 我们同以前的工作一样[9], 修改了部署在终端节点上的神经网络模型的部分超参数. 在 MNIST 中, 我们将卷积层的输出通道数设置为 18、20 或 22. 在 CIFAR10 中, 我们则为不同终端节点设置不同的卷积层步幅. 我们选择具有 3090Ti 显卡、内存为 24 GB 的高性能服务器进行实验. 所有代码都是 Python 语言和 PyTorch 框架实现, 代码总量 650 行, 并且全部在 GPU 上运行, 特征图中毒攻击实验代码可见 https://github.com/bo-lab520/FeaturePoisonAttack.

2.4 FMPA 攻击实验效果

我们利用 Python 成功实现针对特征图正确性的特征图中毒攻击, 设计消融实验, 通过将恶意终端节点的训练数据进行标签反转并将中毒特征图放大多倍, 同时不断调整恶意终端节点数量以及其投毒频率, 确定中毒特征图对于模型收敛性的影响. 当 FedProto 中存在的中毒特征图最多时, 即恶意终端节点数量为 10 且投毒频率为 1 时, 最高可以将诚实终端节点的模型准确率降低 81.72%. 由此可见 FMPA 可以有效影响参与 FedProto 的终端节点的模型准确率. 表 2 展示了在使用 MNIST、FashionMNIST 和 CIFAR10 数据集进行训练的 FedProto 中, 经过 100 轮训练后, 在 20 个诚实终端节点以及不同数量的恶意投毒终端节点同时参与的情况下, 它们的模型平均准确率以及标准差. 其中, CIFAR10 数据集准确率在没有投毒的情况下准确率低的原因是, 我们将数据集按照非独立同分布规则分为了 30 份, 每个终端节点被分配的训练数据量只有 1666 个. 由于训练集较小, 所以平均准确率较低.

表 2　不同数据集上所有终端节点在第 100 轮时平均准确率对比 (%)

投毒频率	恶意终端数	MNIST	FashionMNIST	CIFAR10
0	0	98.36	82.56	59.23
1/5	2	98.06	82.04	56.73
	4	98.27	81.29	58.39
	6	98.21	81.37	55.57
	8	97.74	80.52	53.53
	10	95.34	78.94	49.58
1/3	2	97.73	81.43	55.16
	4	88.63	73.36	47.21
	6	68.91	57.21	33.83
	8	57.80	51.82	32.09
	10	16.64	33.84	36.79
1	2	17.53	34.68	39.02
	4	17.36	31.41	28.29
	6	17.12	30.76	24.47
	8	16.90	30.07	23.47
	10	16.64	29.17	22.66

表 2 的投毒频率属性中, 数值为 1/5 代表终端节点每 5 轮通信投 1 次毒. 通过表格可以看出, 所有终端节点上部署的模型的准确率和恶意终端节点数量以及投毒频率有关. 当投毒频率为 1/5, 模型准确率几乎不受影响. 当投毒频率为 1/3, 模型准确率随着投毒终端节点的数量增加而降低. 如果 FedProto 中的中毒特征图数量足够大, 终端节点将会因全局特征图的影响而无法收敛. 具体来看, 在 MNIST 数据集上, 当投毒频率为 1 时, 模型的平均准确率迅速降低至 16%–18%, 在 CIFAR10 的数据集上降低至 20%–30%. 由此可以证明 FMPA 攻击可以有效影响 FedProto 的正常运行, 导致参与其的终端节点模型准确率低下, 且中毒特征图数量越大, 模型准确率越低.

表 2 只是展示了诚实终端节点在不同恶意终端数量以及不同投毒频率下的模型平均准确率. 为了可以看出具体的细节, 即诚实终端节点的模型准确率何时开始下降、下降趋势如何、最终又会定格在多少, 我们选择性的使用部分诚实终端节点并绘出其模型准确率变化的折线图. 下面选择性展示了具有代表性的 MNIST 和 CIFAR10 数据集, 恶意终端节点数量 P=2, 6, 10 且 PE=1/5, 1/3, 1 时, 部分终端节点从第 1 轮到第 100 轮的模型准确率变化图.

● MNIST 数据集: 图 1 显示了在恶意终端数量为 2 的情况下, 不同投毒频率 (分别为 1/5、1/3 和 1) 下诚实终端节点和恶意终端节点的模型准确率变化. 当投毒频率为 1/5 和 1/3 时, 由于恶意终端数有 2 个且投毒频率较低,

FedProto 中的中毒特征图数量较少, 对其他诚实终端节点的模型准确率并无影响. 具体的原因是, 由公式 (3) 可知, 由于中毒特征图数量少, 参与模型损失函数计算的正则项 σ 对于模型优化的影响很小, 模型更新主要由训练数据产生的交叉熵损失决定, 所以模型准确率并无出现明显下降.

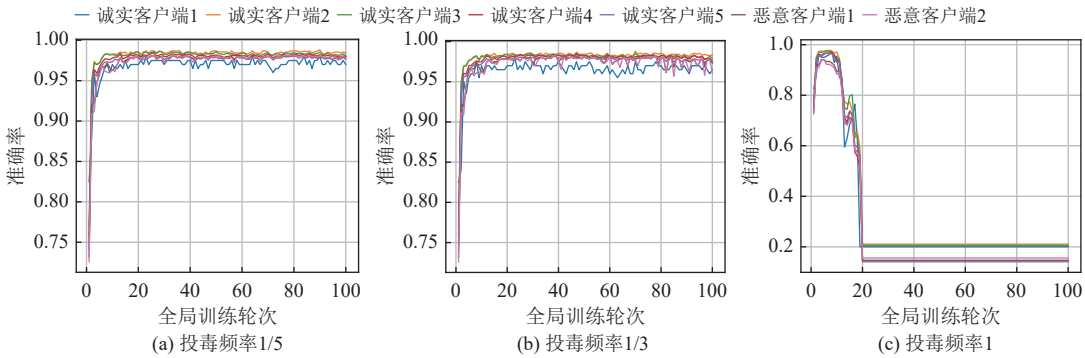


图 1　恶意终端节点数量为 2 时, 不同投毒频率在 MNIST 上的实验结果

然而, 当投毒频率为 1 时, 在每轮通信中恶意终端节点都向 FedProto 注入中毒特征图, 导致第 20 轮通信时模型准确率降至约 15% 的最低点. 原因可以理解为, 由于恶意终端节点投毒频率高, FedProto 在少数轮通信后就含有大量的中毒特征图, 此时正则项 σ 对于模型优化的影响起到关键作用. 又因为 σ 是由错误的全局特征图计算而来, 所以模型准确率下降.

图 2 显示了在恶意终端数量为 6 的情况下, 不同投毒频率 (分别为 1/5、1/3 和 1) 下诚实终端节点和恶意终端节点的模型准确率变化. 考虑到图片篇幅有限以及图片的可读性, 我们只选择其中部分诚实客户端 (5 个) 和恶意客户端 (2 个) 进行准确率变化曲线绘制. 当投毒频率为 1/5, 中毒特征图并没有带来影响, 原因可以参考图 1 的解释. 当投毒频率为 1/3, 到训练轮次 50 后, 终端节点的模型准确率有明显的降低, 最后轮次 100 时, 准确率降低至 70% 左右. 当投毒频率为 1, 由于恶意终端数量的增多且投毒频率更高, 向 FedProto 中投入的中毒特征图数量更大, 所以所有终端节点更早的出现准确率下降的情况, 最终准确率为 15% 左右. 具体原因可以参考图 1 解释.

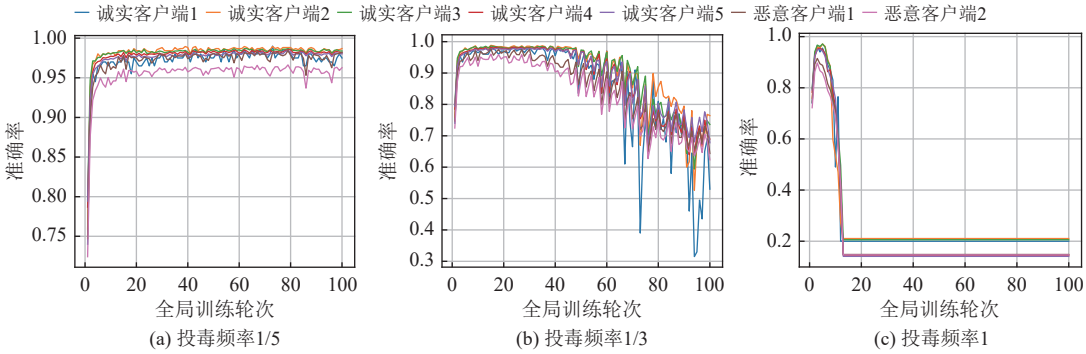


图 2　恶意终端节点数量为 6 时, 不同投毒频率在 MNIST 上的实验结果

图 3 显示了在恶意终端数量为 10 的情况下, 不同投毒频率 (分别为 1/5、1/3 和 1) 下诚实终端节点和恶意终端节点的模型准确率变化, 图 3 及后续实验结果图中只显示部分客户端的原因同图 2.

图 3 准确率变化相比于图 1 和图 2 更为明显. 当投毒频率为 1/5, 中毒特征图对于诚实终端节点的影响并不大, 但是恶意终端节点自身的模型准确率却相对较低. 原因可以解释为, 此时投毒频率较低, 投入 FedProto 的中毒特征图数量较少 (但是比恶意终端数量为 2 和 6 时更多), 全局特征图的计算主要取决于正确的特征图. 所以诚实终端节点利用全局特征图和自身的本地特征图计算得到的正则项 σ 对于模型优化并无太大影响. 相反, 恶意终端节点由于

本地特征图就是错误的, 所以计算的 σ 也是错误的, 此时则会对模型优化产生影响. 当投毒频率为 1/3 时, 由于投毒终端节点数量较大, 所以在训练轮次为 40 时, 模型准确率开始骤降为 15% 左右. 当投毒频率为 1, 由于投毒终端节点数量和投毒频率相较于前几次实验都达到最大值, 所以所有终端节点的模型准确率在 8 轮时就降低至 15%.

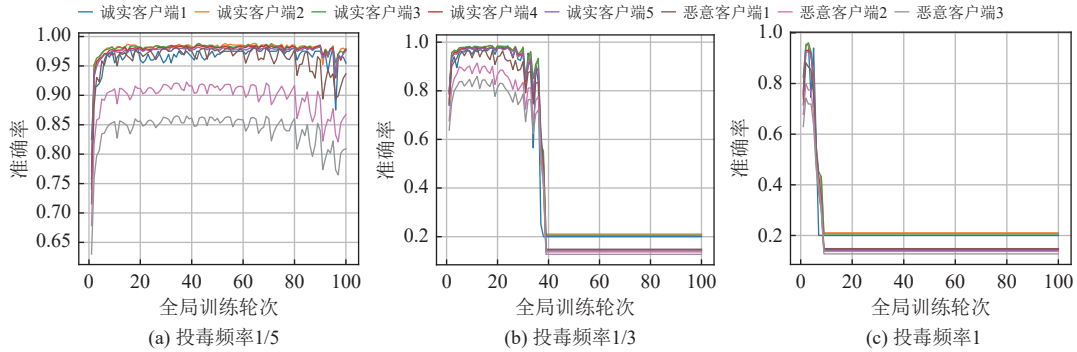


图 3　恶意终端节点数量为 10 时, 不同投毒频率在 MNIST 上的实验结果

● CIFAR10 数据集: 为了证明 FMPA 攻击的普适性, 我同样将 FMPA 攻击应用在 FedProto 中的 CIFAR10 数据集训练 ResNet18 模型的任务上, 并保持上述 MNIST 数据集的实验参数配置. 当恶意终端数量和投毒频率不同时, 模型准确率的变化趋势和 MNIST 数据集上大致一致, 即投入 FedProto 中的中毒特征图数量越多, 模型准确越低.

图 4、图 5 和图 6 显示了在恶意终端数量为 2、6 和 10 的情况下, 不同投毒频率 (分别为 1/5、1/3 和 1) 下诚实终端节点和恶意终端节点的模型准确率变化.

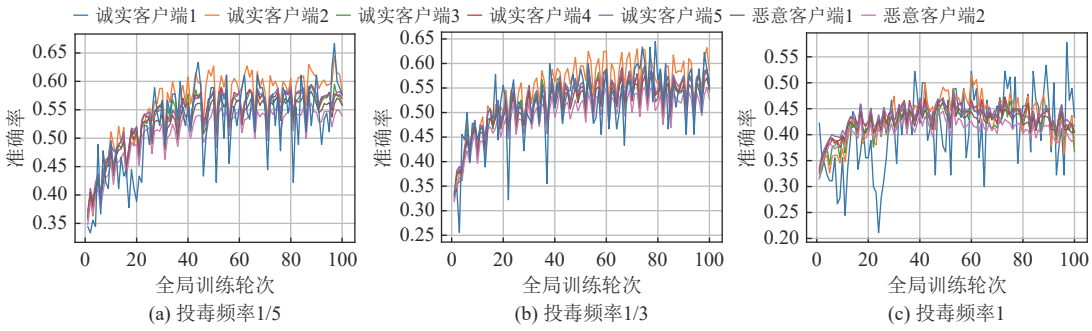


图 4　恶意终端节点数量为 2 时, 不同投毒频率在 CIFAR10 上的实验结果

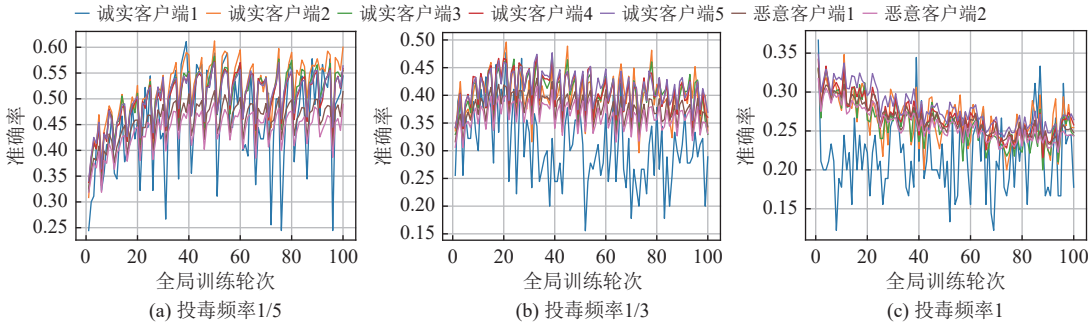


图 5　恶意终端节点数量为 6 时, 不同投毒频率在 CIFAR10 上的实验结果

可以看出, FMPA 攻击可以应用在 CIFAR10 数据集上并使 ResNet18 模型准确率最大降低 36.57% (恶意终端节点数量为 10, 投毒频率为 1). 不同的是, ResNet18 模型在 CIFAR10 数据集上并没有因为中毒特征图而出现梯度

爆炸 (模型的损失函数计算出的损失值过大, 神经网络的参数梯度也将过大导致无法根据此梯度进行模型参数更新. 梯度爆炸的直观表现为, 模型准确率到最低并保持这一最低准确率) 的情况. 而在 MNIST 数据集中, 恶意终端节点数量为 2、6 和 10 且投毒频率为 1 时, 均出现梯度爆炸的情况. 从数据集角度看, MNIST 数据集相对较简单, 包含手写数字的单通道灰度图像. 由于数据集的简单性, 简单模型在训练过程中会更快地适应数据. 当中毒特征图数量变多, 损失值显著增大并产生较大的梯度值, 导致梯度爆炸的问题. 而 CIFAR10 数据集则更为复杂, 包含三通道的彩色图像, 其中每个类别的图像具有更多的变化和细节. 在面对特征图中毒时, 复杂的数据集可以提供更多的梯度平衡, 减少梯度爆炸的风险. 从模型架构角度看, ResNet18 模型采用了残差连接和深度的网络结构, 这些结构有助于缓解梯度爆炸问题. 残差连接允许梯度在网络中更直接地传播, 减少了梯度的累积和放大. 此外, ResNet18 模型相对较深, 一共 18 层, 具有更多的层和参数. 梯度在传播过程中更容易被稀释和抑制, 从而减少了梯度爆炸的发生. 而 CNNMNIST 则没有残差连接且相对较浅 (一共 6 层: 2 层卷积层, 2 层池化层, 2 层全连接层).

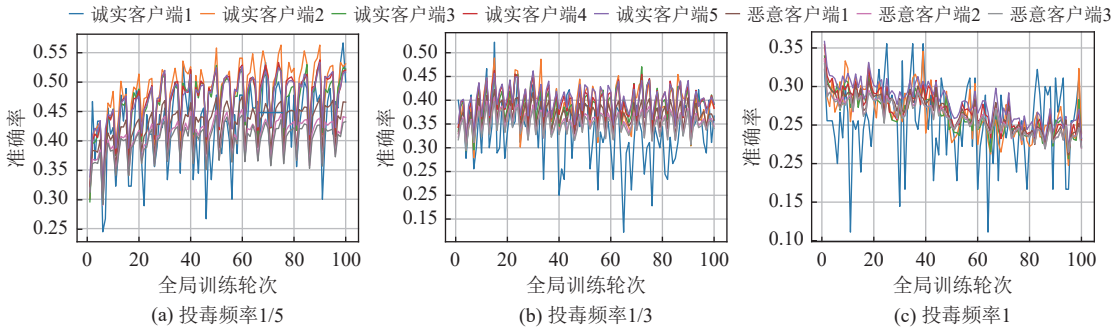


图 6 恶意终端节点数量为 10 时, 不同投毒频率在 CIFAR10 上的实验结果

3 双重防御机制

3.1 防御概述

本节介绍双重防御机制的设计, 其中包括两个阶段的验证, 旨在防止中毒特征图侵入 FedProto 并干扰模型的优化更新. 图 7 展示了 De-FMPA 防御机制的流程.

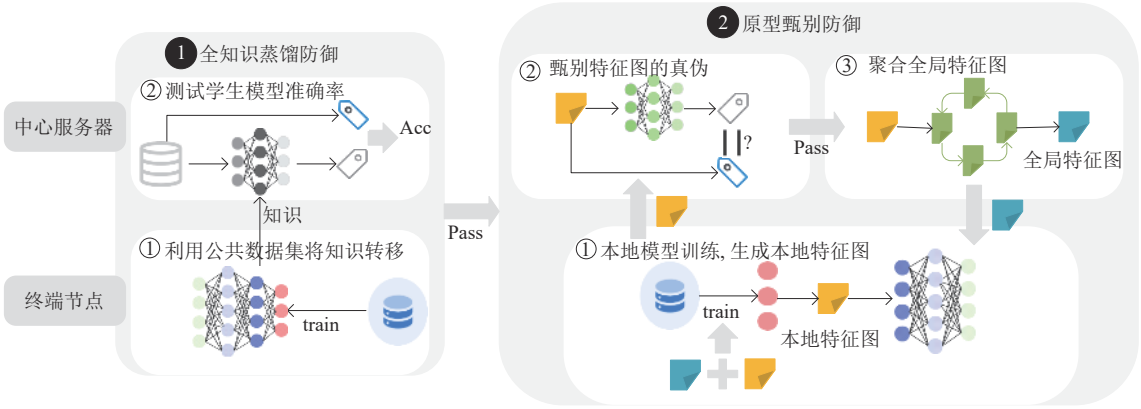


图 7 De-FMPA 防御流程

第 1 阶段, 我们称之为预筛选阶段 (全知识蒸馏防御阶段), 最简单的方法是将终端节点的本地模型参数上传至中心服务器并验证其准确率. 然而, 鉴于传递模型参数可能会泄露用户隐私, 我们放弃了这种传递模型参数的方式. 基于知识蒸馏的预筛选可以将知识从终端节点本地模型转移至中心服务器而无需传递模型参数, 很好地保护

了用户隐私. 本文使用的知识蒸馏技术与传统知识蒸馏所有不同, 我们称之为全知识蒸馏 (详细见第 3.2 节). 它仅将终端节点本地模型的知识转移到中心服务器上的学生模型, 然后中心服务器利用测试数据集验证其可用性. 如果不可用则说明该终端的训练数据不可用 (可能被随机置乱), 则拒绝其加入 FedProto. 第 2 阶段, 利用原型甄别技术以鉴别终端节点上传的本地特征图的正确性. 具体来说, 中心服务器维护一个特征图甄别模型, 它的输入是特征图数据, 输出是特征图对应的标签. 在每一次终端节点传递特征图至中心服务器时, 中心服务器都需要对特征图进行推理. 如果预测标签与终端节点上传的真实标签不一致, 则拒绝该特征图参与聚合, 否则允许参与聚合并利用该特征图对特征图甄别模型进行持续训练.

3.2 全知识蒸馏防御阶段

联邦学习中的数据中毒攻击大部分都是利用混乱的特征数据和标签去训练本地模型, 然后利用错误的本地模型参数去影响其他终端节点的模型准确率. 为了避免这些恶意终端节点加入 FedProto 同时需要保护用户隐私, 我们考虑使用知识蒸馏的方式去判断终端节点的本地训练数据是否被置乱. 如第 1.2 节介绍的, 传统知识蒸馏的学生模型的损失函数存在着两项, 一项是由模型预测标签和真实标签所计算的交叉熵损失, 另一项是由教师模型和学生模型输出的概率分布所计算 *KL* 散度损失. 这种方式中, 尽管学生模型可以学习到教师模型的准确率, 但是也可以学习到用于转移知识的公共数据集的知识. 第 1 阶段只是为了将终端节点本地模型的知识转移至中心服务器的学生模型, 而不包括其他知识.

为此, 本文提出全知识蒸馏技术, 旨在最大限度将本地模型的知识转移到学生模型同时避免公共数据集的知识被学生模型学习. 利用全知识蒸馏技术间接的检测终端节点的本地训练模型的准确率, 而不是将终端节点的模型直接上传至中心服务器, 这种方式很好地保证了终端节点本地模型的机密性以及用户数据的隐私性. 下面利用公式化方式展示全知识蒸馏的计算. 我们利用一个公共数据集, 其中的每一个特征数据都将经过教师模型 (终端节点的本地模型) 和学生模型 (由中心服务器维护) 的推理并输出 *logits*. 全知识蒸馏使用教师模型输出的概率分布 (软标签) 以帮助学生模型进行学习, 其中的软标签由 *Softmax* 函数产生, 具体为 $Softmax(z_i)=\dfrac{\exp(z_i)}{\sum_j \exp(z_j)}$, 其中 z 为 *logits*. 为了防止 *Softmax* 产生的软标签对训练学生模型的损失函数的贡献过小, 知识蒸馏引入了温度 T 的概念, 以提高知识转移效果. 下面是将 T 引入 *Softmax* 后的公式:

$$Softmax(z_i/T)=\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{4}$$

公式 (4) 用来产生教师模型输出的软标签 soft_label, 利用它去监督训练学生模型. 以下是学生模型的损失函数:

$$F_{\text{full\_kd}}=KL\left(\text{soft\_label}, Softmax\left(\frac{logits}{T}\right)\right) \tag{5}$$

另外, 为了充分学习到终端节点的本地模型的知识, 我们分别设置了不同的温度 T (详细情况见实验部分) 去进行全知识蒸馏. 经过比较发现, 当 T 取 0.1 时, 学生模型对终端节点本地模型的学习率达到最高 83.50%.

全知识蒸馏防御的算法伪代码如算法 2. 算法包括了 *Local_Pre-train*() 和 *Full_KD_Defense*() 两个函数, 终端节点加入 FedProto 之前需要先执行 *Local_Pre-train*() 函数, 利用本地数据预训练本地模型. 预训练完成后, 该本地模型作为教师模型指导中心服务器的学生模型进行学习. 中心服务器执行 *Full_KD_Defense*(), 目的是将终端节点的模型的知识转移至学生模型, 随后测试学生模型的准确率, 根据其准确率判断终端节点的训练数据是否可用.

算法 2. 防御程序 1: 全知识蒸馏防御过程.

输入: 恶意终端节点 k 的数据集 D_k. 中心服务器用于知识提取的公共训练数据集 D_{train}, 中心服务器用于测试学生模型的公共测试数据集 D_{test}, KD 中的温度 T, 诚实终端节点的所教出的学生模型的准确率标准 target_acc;
输出: 终端节点的本地模型训练过程是否被恶意投毒.

1. *Local_Pre-train*(D_k): /*终端节点在参与到 FedProto 之前利用本地数据进行预训练*/
2. **for** each local epoch **do** /*本地训练轮次*/
3. **for** batch $(x_k, y_k) \in D_k$ **do** /*按小批次获取训练数据*/
4. Randomly scrambling training data to obtain incorrect data (x_k, y_l) . /*随机置乱数据*/
5. Using (x_k, y_l) to train local model, computing loss. /*计算损失函数*/
6. Updating model by back propagation according to loss. /*反向转播, 更新本地模型*/
7. **end for**
8. **end for**
9. **return** teacher_model /*该本地模型将作为教师模型指导中心服务器的学生模型*/
10. *Full_KD_Defense*(teacher_model): /*中心服务器执行全知识蒸馏*/
11. **for** each full KD train epoch **do** /*全知识训练轮次*/
12. **for** batch $(x, y) \in D_{\text{train}}$ **do** /*按小批次获取训练数据*/
13. y_t = teacher _model.forward(x). /*教师模型预测标签*/
14. soft_label=*Softmax*(y_t / T). /*利用温度参数 T 计算软标签*/
15. y_s =student_model.forward(x). /*学生模型预测输出*/
16. According to soft_label and *Softmax*(y_s / T), compute loss. /*计算损失函数*/
17. Updating student model by loss. /*反向传播损失函数, 更新模型*/
18. **end for**
19. **end for**
20. **for** each full KD test epoch **do** /*利用公共数据集测试学生模型准确率的测试轮数*/
21. **for** batch $(x, y) \in D_{\text{test}}$ **do** /*按小批次获取测试数据*/
22. y_s =student_model.forward(x). /*利用测试数据集进行预测*/
23. acc.append(evaluate (y_s, y)). /*计算测试准确率*/
24. **end for**
25. KD_acc=mean(acc). /*计算平均准确率*/
26. **end for**
27. **if** KD_acc<target_acc **then** /*如果学生模型的平均准确率低于标准准确率*/
28. **return** this terminal is malicious. /*返回结果, 该终端随机置乱训练数据*/
29. **end if**
30. **return** this terminal is honest. /*返回结果, 该终端随机置乱训练数据*/

3.3 特征图甄别防御阶段

本节详细介绍第 2 阶段, 特征图甄别技术. 该技术是为了确定终端节点上传的本地特征图的正确性, 避免错误的本地中毒特征图参与到全局特征图的聚合中. 图 8 展示了该阶段的详细流程.

我们的做法是在中心服务器维护一个特征图甄别模型 (包括 3 层卷积层, 2 层全连接层) 用于甄别终端节点上传的特征图是否正确. 该模型的输入是特征图数据, 输出是该特征图对应的标签. 我们首先利用中心服务器上的公开数据集对该模型进行预训练, 同时在 FedProto 过程, 利用正确的终端节点上传的特征图去持续训练该模型, 以提高特征图甄别模型的精准度.

原型甄别的算法伪代码如算法 3. 该算法包括函数 *Feature_Map_Defense*(), 该函数在中心服务器执行, 在特征图甄别过程中, 如果模型预测标签与真实标签不一致, 则拒绝该特征图参与后续的全局特征图的聚合. 否则接受其加入特征图聚合, 同时利用该特征图去持续的训练特征图甄别模型, 以提高预测准确率.

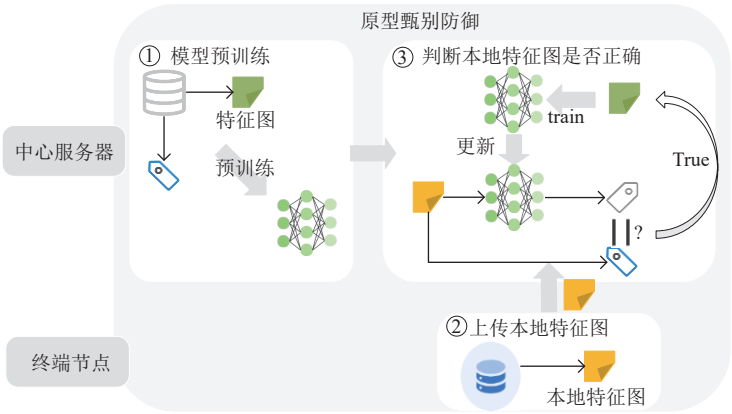


图 8 原型甄别防御流程

算法 3. 防御程序 2: 原型甄别防御过程.

输入: 中心服务器本地维护的特征图甄别模型 feature_map_model, 终端节点传递的特征图 feature_map 和标签 target_label;
输出: 终端节点上传的本地特征图是否正确.

1. *Feature_Map_Defense*(feature_map_model, feature_map, target_label)**:** /*中心服务器执行*/
2. y=feature_map_model.forward(feature_map). /*对终端节点上产的原型进行预测*/
3. **if** y==target_label **then** /*如果预测值与真实标签相同, 则认为该特征图正确*/
4. *Glogal_Feature_Map_Agg*(feature_map). /*计算全局特征图*/
5. **for** each epoch **do** /*迭代多轮*/
6. Using feature_map to train feature_map_model. /*利用此特征图训练持续特征图鉴别模型*/
7. **end for**
8. **elif** y !=target_label **then** /*如果预测值与真实标签不相同, 则认为其为中毒特征图*/
10. Abandon this feature_map. /*拒绝将该 feature_map 作为计算全局特征图的一部分*/
11. **end if**

4 De-FMPA 防御实验效果

本节对 De-FMPA 防御对于 FMPA 攻击的防御效果进行实验. 我们选择使用 MNIST 数据集, 其他相关参数和攻击实验的配置保持一致.

4.1 全知识蒸馏防御实验效果

● 最佳的蒸馏温度: 在预筛选阶段, 我们采用本文提出的全知识蒸馏的方式, 旨在快速筛选出用毒性数据训练本地模型的投毒终端节点, 减小后续特征图甄别防御阶段的计算开销. 为了探索出最佳的用于全知识蒸馏的温度 T, 我们将 T 设置为不同的数值, 发现当 T=0.1 时可以最大限度地提取出教师模型的知识. 在不同温度下, 我们都随机选择 2 个终端节点作为教师模型以帮助训练服务器的学生模型. 表 3 是实验数据, 包含了两个客户端的结果.

表 3 属性中, 教师模型即为终端节点本地模型, 而学生模型部署在服务器上. 由表 3 可以看出, 当 T 设置为 0.1 时, 全知识蒸馏的知识提取效果最好, 在不投毒的情况下最高可以达到 83.88% 的学习率 (学生模型准确率除以教师模型准确率). 我们根据学生模型的准确率去决定教师模型是否中毒, 当学生模型的准确率低于某一个阈值时 (不同数据集上的阈值也不相同, 本次实验中对 MNIST 数据集设置为 60%), 就认为该终端节点是恶意终端节点.

表 3　不同温度下学生模型的学习效果

是否投毒	温度T	教师模型准确率 (%)	学生模型准确率 (%)	学习率 (%)
是	3	33.75/44.37	12.50/22.41	37.04/50.51
	1	46.25/67.50	20.83/20.83	45.04/30.86
	0.1	40.01/39.37	21.87/21.35	**54.67**/**54.23**
	0.01	43.75/53.12	21.61/24.89	49.40/46.86
否	3	98.13/96.87	24.50/25.52	24.97/26.34
	1	95.62/96.87	64.06/79.16	67.01/81.71
	0.1	97.50/98.75	81.78/81.25	**83.88**/**82.28**
	0.01	97.50/98.13	64.58/64.45	66.23/65.68

注: 实验数据包括两个客户端的结果

● 最佳温度下的全知识蒸馏: 我们模拟了 100 个虚拟终端节点, 其中有 60 个为恶意客户端, 他们会在本地进行训练数据置乱, 剩余 40 个为诚实终端节点. 表 4 记录了加入恶意或者诚实终端节点时, 全知识蒸馏防御阶段的鉴别准确率.

表 4　全知识蒸馏鉴别成功率

是否投毒	终端节点数量	鉴别正确数量	鉴别成功率 (%)
是	60	60	100
否	40	37	92.5

恶意终端节点的鉴别是预筛选阶段的主要任务. 表 4 可以看出, 当模拟 60 个恶意终端节点时, 预筛选阶段将他们全部成功识别为诚实终端节点, 准确率为 100%. 由于恶意终端节点的本地模型准确率大多在 10%–50% 之间, 它们蕴含的知识量较少, 所以学生模型的学习率较低 (当 T=0.1 时, 学习率只有 54% 左右), 最终学生模型的准确率将更低 (通常低于 60%), 所以全知识蒸馏对于恶意终端节点的鉴别准确率很高. 当模拟 40 个诚实终端节点, 预筛选阶段可以识别出其中 37 个为恶意终端节点, 准确率为 92.5%. 对于少数鉴别失败的终端节点, 原因可能是, 分配到该终端节点上的数据量较少, 且数据分布与用于全知识蒸馏的公共数据集相差较多, 导致学生模型的学习率较低, 最终学生模型准确率较低.

● 知识蒸馏与全知识蒸馏: 由预备知识可知, 知识蒸馏中学生模型的损失函数包含两项, 即软目标和知识蒸馏损失, 具体损失函数如下:

$$F_{\mathrm{kd}} = \alpha \times CE(y, Softmax(logits)) + (1-\alpha) \times KL\left(\mathrm{soft\_label}, Softmax\left(\frac{logits}{T}\right)\right) \tag{6}$$

其中, y 代表数据的真实标签. 公式 (6) 与全知识蒸馏的损失函数 (公式 (5)) 的区别在于, 学生模型通过上式不仅学习了教师模型的知识 (软目标), 还学习了数据集的知识 (知识蒸馏损失) , 而全知识蒸馏只学习了教师模型的知识. 当知识蒸馏损失项的权值 α 为 0 时, 知识蒸馏转变成全知识蒸馏. 为了能让学生模型更好的体现教师模型, 即通过学生模型的准确率判断教师模型是否中毒, 我们做出如下实验. 我们将温度设置为 0.1, 依次向 FedProto 投入 60 个恶意终端节点, 并分别利用知识蒸馏和全知识蒸馏两种知识转移方式, 将终端节点本地模型知识转移至中心服务器的学生模型. 表 5 体现了全知识蒸馏鉴别恶意终端节点的优势.

表 5　知识蒸馏与全知识蒸馏

方案	α	恶意终端节点数量	鉴别正确数量	鉴别成功率 (%)
知识蒸馏	0.9	60	3	5
	0.5	60	7	11.67
	0.1	60	12	20
全知识蒸馏	0	60	60	100

通过表 5 可以看出, 当知识蒸馏的学生模型损失函数中的超参数 α 的值越低, 恶意终端节点的鉴别成功率就越高, 当 α 为 0 时, 此时损失函数变成全知识蒸馏, 鉴别成功率上升至 100%. 由此可以看出全知识蒸馏在预筛选阶

段的恶意终端节点鉴别方面的优势.

4.2 特征图甄别防御实验效果

为了展现出特征图甄别防御的有效性, 我们做出消融实验. 实验在开启全知识蒸馏防御的同时, 分别设置开启或不开启特征图甄别防御的消融实验, 且每隔 10 轮通信投入一个恶意终端节点, 其投毒频率分别为 1/5, 1/3, 1. 为了充分体现出特征图甄别防御的必要性和有效性, 我们需要设置一些恶意终端节点可以部分参与到 FedProto. 这里投入的恶意终端节点利用干净数据集进行本地训练, 然而却上传中毒的特征图数据. 由于其利用干净数据集训练模型, 恶意终端节点的本地模型可以达到较高的准确率, 能以更大的概率绕过全知识蒸馏防御并参与后续的 FedProto.

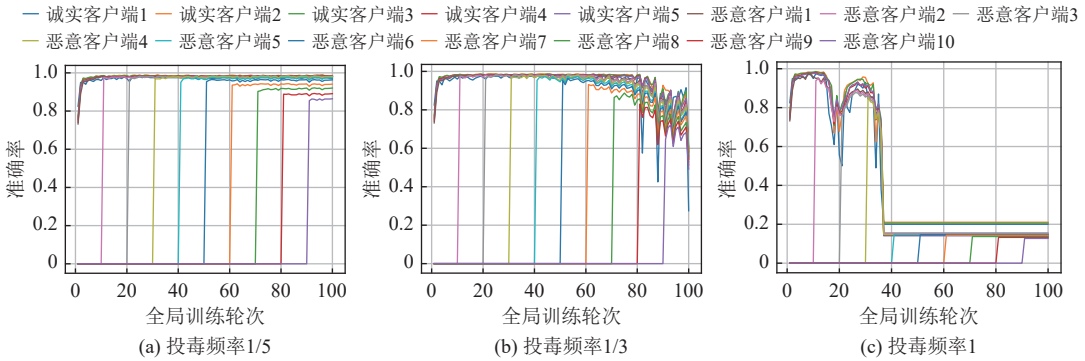
表 6 展示了在全局训练轮次为 100 轮且每隔 10 轮通信投入一个恶意终端节点后时, 所有终端节点在最终通信轮次时的平均准确率. 可以看出, 在不同的投毒频率下, 由于部分恶意终端节点本地模型准确率较高而绕过了全知识蒸馏防御并加入 FedProto 中进行投毒, 开启特征图甄别防御机制后的模型平均准确率总是高于不开启防御机制. 尤其当投毒频率为 1 时, 特征图甄别防御机制在 CIFAR10 和 MNIST 数据集上分别可以将模型平均准确率提高 39.84% 和 80.84%, 充分体现了特征图甄别防御的有效性和必要性.

表 6　是否开启特征图甄别防御的情况下所有终端节点在第 100 轮模型平均准确率 (%)

投毒频率	是否开启特征图甄别防御	CIFAR10	MNIST
1/5	是	67.93	97.80
	否	42.56	97.16
1/3	是	73.91	97.64
	否	35.17	54.93
1	是	75.91	97.48
	否	36.07	16.64

为了清晰的展示特征图甄别防御对于终端节点的模型准确率影响, 下面对于不同数据集投毒频率为 1/5, 1/3 和 1, 分别画出开启防御和不开启防御的准确率变化对比图. 在开启防御机制的图中, 我们只画出诚实终端节点的模型准确率变化图.

● MNIST 数据集: 图 9 展示了在不同投毒频率下, 不开启特征图甄别防御时参与 FedProto 的所有终端节点本地模型的准确率变化. 当投毒频率为 1/5, 其中竖直线代表在当前轮次投入一个恶意终端节点. 由于每 5 轮投毒一次, 投毒频率较低, 所以中毒特征图并没有影响到诚实终端节点模型的准确率. 投毒频率 1/3 时, 不开启特征图甄别防御机制将会明显降低模型准确率. 图 9 显示, 当在 70 轮通信中投入第 7 个恶意终端节点时, 由于投毒频率上升为每 3 轮投毒一次, 此时 FedProto 内部蕴含的大量中毒特征图严重影响了全局特征图的计算, 诚实终端节点的平均模型准确率也开始下降并最终降低至 54.93%. 相比于图 10 中开启特征图甄别防御机制, 终端节点的平均模型准确率降低 42.71%. 当投毒频率为 1 时, FedProto 中在少数几轮就蕴含了大量的中毒特征图.



(a) 投毒频率1/5　(b) 投毒频率1/3　(c) 投毒频率1

图 9　在 MNIST 上, 不开启特征图甄别防御的模型准确率对比图

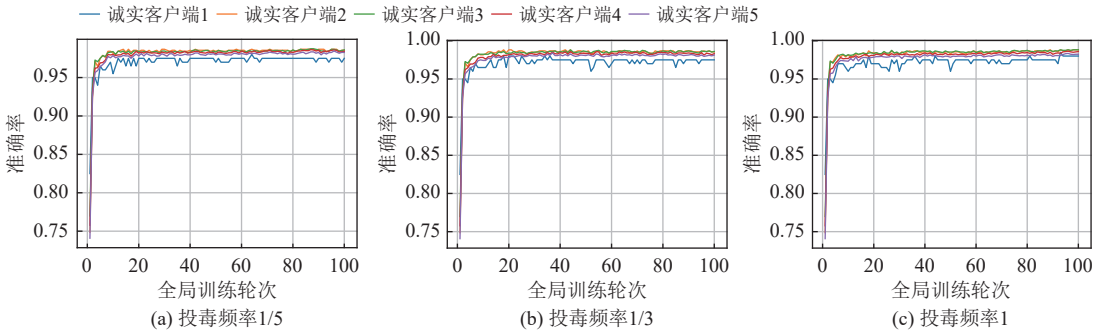


图 10　在 MNIST 上, 开启特征图甄别防御的模型准确率对比图

● CIFAR10 数据集: 我们同样在 CIFRA10 数据集上做出相关实验. 图 11 表示不同投毒频率下不开启特征图甄别防御时, 终端节点本地模型准确率变化图, 图 12 则是开启特征图甄别防御时的准确率变化图. 可以看出, 尽管在 CIFAR10 数据集上, 恶意终端节点上产的中毒特征图没有引起模型梯度爆炸的情况, 但是也在很大程度上影响了其他诚实终端节点本地模型的收敛. 由于 CIFAR10 数据集更加复杂, 本地分类任务计算的损失较大, 为了使特征图中毒攻击生效, 恶意终端节点在上传中毒特征图之前先扩大 10 倍, 使得中毒效率更高, 实验结果更显著. 从图 11 和图 12 可以看出, 当在 FedProto 中应用特征图甄别防御使可以显著提高诚实终端节点的本地模型准确率.

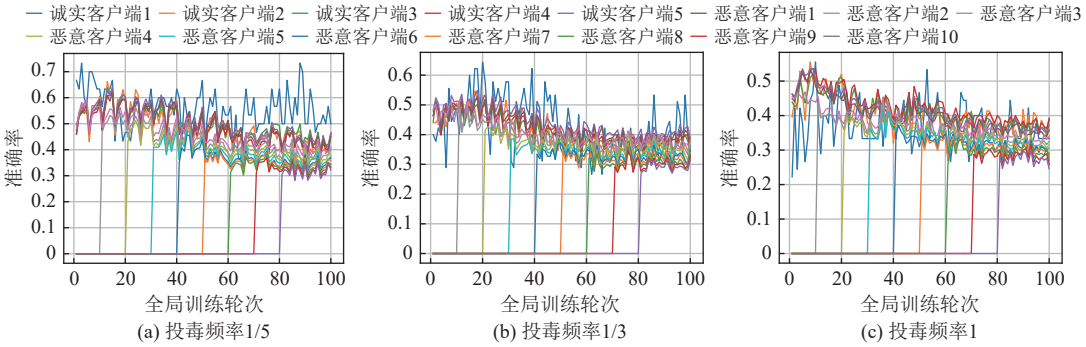


图 11　在 CIFAR10 上, 不开启特征图甄别防御的模型准确率对比图

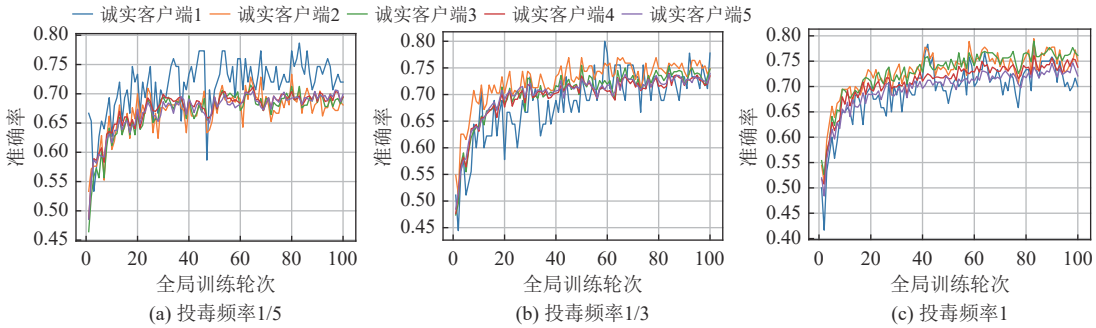


图 12　在 CIFAR10 上, 开启特征图甄别防御的模型准确率对比图

4.3　双重防御机制的必要性

● 全知识蒸馏防御的必要性: 为了减轻特征图甄别防御阶段的计算资源消耗, 我们选择使用全知识蒸馏的防御在终端节点加入 FedProto 之前进行一轮预筛选, 以防止使用中毒数据训练模型的恶意终端节点加入其中, 减小

特征图甄别模型在推理特征图数据时所带来的额外运行时间, 同时还可以直接避免上述恶意终端节点上传中毒特征图以提高模型准确率.

我们假设一共 30 个终端节点参与 FedProto, 其中 20 个为诚实终端节点, 10 个为恶意终端节点, 下表展示了使用全知识蒸馏防御或者不使用时, 终端节点的平均模型准确率. 从表 7 可以看出, 在开启全知识整理防御后, 终端节点的模型平均准确率高于不开启, 尤其当投毒频率为 1 时, 开启全知识蒸馏防御使得模型准确率提高 5.19%. 不开启全知识蒸馏防御模型准确率偏低的原因是, 所有的恶意终端节点都会进入 FedProto 并持续向中心服务器上传中毒特征图. 尽管中心服务器维护一个特征图甄别模型, 但是该模型准确率并不能达到 100%, 会误判少数中毒特征图并允许其加入全局特征图的计算. 如果不经过全知识蒸馏的预筛选, 则所有恶意终端节点都会加入 FedProto. 此时中毒特征图数量也会增大, 由于特征图甄别模型误判而参与全局特征图聚合的中毒特征图也会变多, 则对模型准确率影响也就越大.

另一方面, 如果所有的恶意终端节点不经筛选就全部加入 FedProto, 当它们以高投毒频率向中心服务器上传大量中毒特征图时, 都需要经过特征图甄别模型的推理以鉴别其正确性. 然而, 大量特征图的推理将严重拖慢系统的运行时间, 减慢模型收敛速度, 从表 8 记录了系统 100 轮通信的平均运行时间, 当开启全知识蒸馏防御时系统的平均通信运行时间比不开启全时要降低 4.07 s.

表 7　是否开启全知识蒸馏防御的模型平均准确率

投毒频率	是否开启全知识蒸馏防御	平均准确率 (%)
1/5	是	97.80
	否	97.63
1/3	是	97.64
	否	95.28
1	是	97.48
	否	92.29

表 8　是否开启全知识蒸馏防御的系统平均运行时间

是否开启全知识蒸馏防御	平均一轮通信时间 (s)
是	38.49
否	42.56

● 特征图甄别防御的必要性: 全知识蒸馏防御阶段是为了快速筛选出利用中毒数据训练本地模型的恶意终端节点, 以避免这些恶意终端节点加入 FedProto. 但是该阶段并没有保证参与 FedProto 的终端节点上传至中心服务器的特征图的正确性. 所以我们需要利用特征图甄别防御机制持续性的对本地特征图进行甄别, 以避免中毒特征图影响全局特征图的计算, 进而影响诚实终端节点的本地模型准确率.

5　总结与展望

本文针对通用的异构联邦学习框架 FedProto 提出了特征图中毒攻击, 同时提出结合全知识蒸馏和特征图甄别的两阶段防御机制. FMPA 不同于以往的利用中毒数据诱导错的模型参数, 而是使用中毒特征图使全局特征图产生偏移, 从而间接影响模型的更新优化. 为了应对上述的攻击, 我们设计了 De-FMPA 双重防御机制. 首先预筛选阶段将快速筛选出使用毒性数据训练本地模型的恶意终端节点, 其后第 2 阶段防御主要是检测终端节点上传的特征图是否正确. 实验使用 MNIST、FashionMNIST 和 CIFAR10 数据集验证了 FMPA 攻击和 De-FMPA 防御在 FedProto 上的有效性.

长远来看, 考虑到 FedProto 其实是一种基于特征的联邦学习, 而基于特征的联邦学习普遍都可以克服联邦学习中的数据异构和模型异构两大挑战, 未来将把 FMPA 攻击和 De-FMPA 防御机制扩展至所有基于特征的联邦学习. 此外, 未来如有其他文献针对特征图设计攻击和防御方法, 我们将进一步设计对比实验, 优化本文的方法, 使其更好地服务于临床医疗、无人驾驶等领域.

References:

[1] 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.

[2] Tang LT, Chen ZN, Zhang LF, Wu D. Research progress of privacy issues in federated learning. Ruan Jian Xue Bao/Journal of Software, 2023, 34(1): 197–229 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6411.htm [doi: 10.13328/j.cnki.jos.006411]

[3] Liu YX, Chen H, Liu YH, Li CP. Privacy-preserving techniques in federated learning. Ruan Jian Xue Bao/Journal of Software, 2022, 33(3): 1057–1092 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6446.htm [doi: 10.13328/j.cnki.jos.006446]

[4] Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L. Deep learning with differential privacy. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 308–318. [doi: 10.1145/2976749.2978318]

[5] Wang RJ, Lai JS, Zhang ZY, Li X, Vijayakumar P, Karuppiah M. Privacy-preserving federated learning for Internet of medical things under edge computing. IEEE Journal of Biomedical and Health Informatics, 2023, 27(2): 854–865. [doi: 10.1109/JBHI.2022.3157725]

[6] Wei K, Li J, Ding M, Ma C, Su H, Zhang B, Poor HV. User-level privacy-preserving federated learning: Analysis and performance optimization. IEEE Trans. on Mobile Computing, 2022, 21(9): 3388–3401. [doi: 10.1109/TMC.2021.3056991]

[7] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, Mcmahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacy-preserving machine learning. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security, 2017. 1175–1191. [doi: 10.1145/3133956.3133982]

[8] Karimireddy SP, Kale S, Mohri M, Reddi SJ, Stich SU, Suresh AT. SCAFFOLD: Stochastic controlled averaging for federated learning. In: Proc. of the 37th Int'l Conf. on Machine Learning. PMLR, 2020. 5132–5143.

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

[10] Tan AZ, Yu H, Cui LZ, Yang Q. Towards personalized federated learning. IEEE Trans. on Neural Networks and Learning Systems, 2023, 34(12): 9587–9603. [doi: 10.1109/TNNLS.2022.3160699]

[11] Dinh CT, Tran NH, Nguyen TD. Personalized federated learning with moreau envelopes. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc, 2020. 1796.

[12] Wu Q, He KW, Chen X. Personalized federated learning for intelligent IoT applications: A cloud-edge based framework. IEEE Open Journal of the Computer Society, 2020, 1: 35–44. [doi: 10.1109/OJCS.2020.2993259]

[13] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

[14] Zhang Y, Xiang T, Hospedales TM, Lu HC. Deep mutual learning. In: Proc. of the 2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018. 4320–4328. [doi: 10.1109/CVPR.2018.00454]

[15] Cho JH, Hariharan B. On the efficacy of knowledge distillation. In: Proc. of the 2019 IEEE/CVF Int'l Conf. on Computer Vision. Seoul: IEEE, 2019. 4793–4801. [doi: 10.1109/ICCV.2019.00489]

[16] Chen DF, Mei JP, Zhang HL, Wang C, Feng Y, Chen C. Knowledge distillation with the reused teacher classifier. In: Proc. of the 2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. New Orleans: IEEE, 2022. 11923–11932. [doi: 10.1109/CVPR52688.2022.01163]

[17] Li DL, Wang JP. FedMD: Heterogenous federated learning via model distillation. arXiv:1910.03581, 2019.

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

[19] Guo QS, Wang XJ, Wu YC, Yu ZP, Liang D, Hu XL, Luo P. Online knowledge distillation via collaborative learning. In: Proc. of the 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition. Seattle: IEEE, 2020. 11017–11026. [doi: 10.1109/CVPR42600.2020.01103]

[20] Tan Y, Long GD, Liu L, Zhou TY, Lu QH, Jiang J, Zhang CQ. FedProto: Federated prototype learning across heterogeneous clients. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. AAAI Press, 2022. 8432–8440. [doi: 10.1609/aaai.v36i8.20819]

[21] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436–444. [doi: 10.1038/nature14539]

[22] Ma XD, Zhu J, Lin ZH, Chen SX, Qin YJ. A state-of-the-art survey on solving non-IID data in federated learning. Future Generation Computer Systems, 2022, 135: 244–258 [doi: 10.1016/j.future.2022.05.003]

[23] Chen Z, Yang C, Zhu ML, Peng Z, Yuan YX. Personalized retrogress-resilient federated learning toward imbalanced medical data. IEEE Trans. on Medical Imaging, 2022, 41(12): 3663–3674 [doi: 10.1109/TMI.2022.3192483]

[24] Sun Y, Shen L, Huang T, *et al*. Fedspeed: Larger local interval, less communication round, and higher generalization accuracy. arXiv:2302.10429, 2023.

[25] Koh PW, Steinhardt J, Liang P. Stronger data poisoning attacks break data sanitization defenses. Machine Learning, 2022, 111(1): 1–47. [doi: 10.1007/s10994-021-06119-y]

[26] Nuding F, Mayer R. Data poisoning in sequential and parallel federated learning. In: Proc. of the 2022 ACM on Int'l Workshop on Security and Privacy Analytics. Baltimore: ACM, 2022. 24–34. [doi: 10.1145/3510548.3519372]

[27] Kurita K, Michel P, Neubig G. Weight poisoning attacks on pre-trained models. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, 2020. 2793–2806. [doi: 10.18653/v1/2020.acl-main.249]

[28] Chen XY, Liu C, Li B, Lu K, Song D. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv: 1712.05526, 2017.

[29] Huang H, Mu JM, Gong NZ, Li Q, Liu B, Xu MW. Data poisoning attacks to deep learning based recommender systems. In: Proc. of the 28th Annual Network and Distributed System Security Symp. 2021. [doi: 10.14722/ndss.2021.24525]

[30] Peri N, Gupta N, Huang WR, Fowl L, Zhu C, Feizi S, Goldstein T, Dickerson JP. Deep *k*-NN defense against clean-label data poisoning attacks. In: Proc. of the 2020 European Conf. on Computer Vision. Glasgow: Springer, 2020. 55–70. [doi: 10.1007/978-3-030-66415-2_4]

[31] Chen J, Zhang XX, Zhang R, Wang C, Liu L. De-pois: An attack-agnostic defense against data poisoning attacks. IEEE Trans. on Information Forensics and Security, 2021, 16: 3412–3425. [doi: 10.1109/TIFS.2021.3080522]

[32] Baracaldo N, Chen B, Ludwig H, Safavi JA. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In: Proc. of the 10th ACM Workshop on Artificial Intelligence and Security. Dallas: ACM, 2017. 103–110. [doi: 10.1145/3128572.3140450]

[33] Shah D, Dube P, Chakraborty S, Verma A. Adversarial training in communication constrained federated learning. arXiv:2103.01319, 2021.

[34] Deng L. The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 2012, 29(6): 141–142. [doi: 10.1109/MSP.2012.2211477]

[35] Li HM, Liu HC, Ji XY, Li GQ, Shi LP. CIFAR10-DVS: An event-stream dataset for object classification. Frontiers in Neuroscience, 2017, 11: 309. [doi: 10.3389/fnins.2017.00309]

[36] Wang JB, Pei XK, Wang RJ, Zhang FL, Chen T. Federated semi-supervised learning with tolerant guidance and powerful classifier in edge scenarios. Information Sciences, 2024, 662: 120201. [doi: 10.1016/j.ins.2024.120201]

附中文参考文献:

[2] 汤凌韬, 陈左宁, 张鲁飞, 吴东. 联邦学习中的隐私问题研究进展. 软件学报, 2023, 34(1): 197–229. http://www.jos.org.cn/1000-9825/6411.htm [doi: 10.13328/j.cnki.jos.006411]

[3] 刘艺璇, 陈红, 刘宇涵, 李翠平. 联邦学习中的隐私保护技术. 软件学报, 2022, 33(3): 1057–1092. http://www.jos.org.cn/1000-9825/6446.htm [doi: 10.13328/j.cnki.jos.006446]

王瑞锦(1980－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为联邦学习与聚合安全, 智能计算, 区块链, 知识图谱, 信息安全.

李经伟(1987－), 男, 博士, 研究员, 博士生导师, 主要研究领域为数据安全存储, 联邦学习与聚合安全, 区块链取证.

王金波(2000－), 男, 硕士生, CCF 学生会员, 主要研究领域为联邦学习与聚合安全, 智能计算, 网络与信息安全, 区块链.

李增鹏(1989－), 男, 博士, 副研究员, 博士生导师, CCF 高级会员, 主要研究领域为安全多方计算, 安全认证, 联邦学习与聚合安全.

张凤荔(1963－), 女, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为智能计算, 联邦学习, 网络与信息安全.

陈厅(1987－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件安全, 联邦学习与聚合安全, 区块链, 网络安全.