# 基于轻量自蒸馏的低成本联邦学习

刘 松[1,3],罗杨宇[2,3],许佳培[1,3],张建忠[2,3*]
(1. 南开大学计算机学院,天津 300350;2. 南开大学网络空间安全学院,天津 300350;
3. 数据与智能系统安全教育部重点实验室,天津 300350)

**摘 要:** 随着边缘计算的发展,深度学习模型的训练越发依赖于大量边缘设备生成的隐私数据. 在此背景下,联邦学习因其突出的隐私保护能力而受到学术界和工业界的广泛瞩目. 在实践中,联邦学习面临着因数据异质性和计算资源受限导致的训练效率低下和模型质量不理想的问题. 本文受知识蒸馏理念的启发,提出一种采用轻量自蒸馏技术的高效联邦学习算法(efficient Federated learning with lightweight Self Knowledge Distillation,FedSKD),该算法首先利用自蒸馏技术挖掘训练过程中的内在知识,以减轻本地模型的过拟合问题并增强其泛化能力,并通过服务端参数聚合将本地模型的泛化能力转移到全局模型,从而提高全局模型质量和收敛速度. 其次,通过动态同步机制,进一步提高全局模型的准确率和训练效率. 实验结果表明,FedSKD算法在非独立同分布数据划分策略下,在降低训练代价的同时,提高了模型准确率和训练效率. 在CIFAR10/100数据集上,与最新的基线算法FedMLD算法相比,FedSKD算法在准确率上取得了平均2%的提升,并降低了平均56%的训练代价.

**关键词:** 联邦学习;自蒸馏;非独立同分布;深度学习;边缘计算
**基金项目:** 天津市科技重大专项与工程(No.18ZXZNGX00200)
**中图分类号:** TP391 **文献标识码:** A **文章编号:** 0372-2112(2025)01-0259-11
**电子学报URL:**http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240325

# Low-Cost Federated Learning Based on Lightweight Self-Distillation

LIU Song[1,3], LUO Yang-yu[2,3], XU Jia-pei[1,3], ZHANG Jian-zhong[2,3*]
(1. *College of Computer Science*, *Nankai University*, *Tianjin* 300350, *China*;
2. *College of Cyber Science*, *Nankai University*, *Tianjin* 300350, *China*;
3. *Ministry of Education Key Laboratory of Data and Intelligent System Security*, *Tianjin* 300350, *China*)

**Abstract:** With the development of edge computing, the training of deep learning models increasingly relies on the privacy data generated by a large number of edge devices. In this context, federated learning has drawn extensive attention from both academia and industry due to its prominent privacy protection capabilities. However, in practice, federated learning faces challenges such as inefficient training and suboptimal model quality due to data heterogeneity and limited computational resources. Inspired by the concept of knowledge distillation, this paper proposes an efficient federated learning algorithm, named efficient federated learning with lightweight self knowledge distillation (FedSKD). This algorithm utilizes lightweight self-distillation techniques to extract intrinsic knowledge during the training process, alleviating local model overfitting and enhancing its generalization capability. Furthermore, it aggregates the generalization capability of local models to a global model through server parameter aggregation, thereby improving the quality and convergence speed of the global model. Additionally, by employing a dynamic synchronization mechanism, it further enhances the accuracy and training efficiency of the global model. Experimental results demonstrate that FedSKD algorithm, under non-identically distributed data partition strategies, enhances model accuracy and training efficiency while reducing computational costs. On the CIFAR10/100, compared to the latest baseline FedMLD, the FedSKD achieved an average 2% improvement in accuracy and reduced the training cost by an average of 56%.

**Key words:** federal learning; self-distillation; non-independent distribution; deep learning; edge computing
**Foundation Item(s):** Technology Research and Development Program of Tianjin (No.18ZXZNGX00200)

收稿日期:2024-04-09;修回日期:2024-11-10;责任编辑:覃怀银
*通讯作者:张建忠

## 1 引言

随着边缘计算的普及与发展,各种类型的边缘设备为深度学习模型的训练提供了丰富的数据资源.然而,传统的集中式训练方式面临着数据隐私泄漏风险,并带来高昂的通信代价[1~3].联邦学习[4]作为一种去中心化的分布式机器学习方式,各参与方在不向其他参与方披露泄露隐私数据的前提下,协作进行模型训练.联邦学习要求参与者进行多轮次的本地模型训练,并定期将本地模型参数上传至服务端以进行参数聚合,生成全局模型.尽管如此,在实际场景中,参与者的本地数据往往存在异质性,呈现非独立同分布(Non-Independent and Identically Distributed,Non-IID),这限制了本地模型的知识获取能力,增加了其过拟合的风险,从而降低了全局模型的性能并提高了其收敛速度[5,6],对资源受限的边缘设备带来了较大的计算开销和通信开销.

针对上述问题,研究者们提出了多种优化算法来缓解数据异质性带来的影响.部分研究[6~10]通过在损失函数后添加近端项,约束本地模型和全局模型在参数空间上的偏差,提高训练效率,但这些方法在模型准确率上表现不佳[11].一些研究在本地训练中引入全局模型[12,13],以减少本地模型和全局模型在更新方向上的差异,然而这种方式增加了本地训练的计算成本,对边缘设备产生了较大的计算负担.还有一些研究[14~18]借助公共数据集或在训练中生成辅助数据集,从根本上降低本地数据集非独立同分布的程度,但是添加额外数据集的方式增加了边缘设备的存储压力和计算成本,而且由于公共数据集需要在训练前手动选择,这降低了算法的普适性和鲁棒性,限制了在真实场景中的部署[19,20].

本文受知识蒸馏方法的启发[21,22],提出了一种基于轻量自蒸馏的高效联邦学习算法(efficient Federated learning with Self Knowledge Distillation,FedSKD),该算法要求参与者在每一轮本地训练过程中,利用上一轮训练的逻辑单元向量进行知识蒸馏,提高知识的传递与映射效率,增强本地模型的泛化能力.服务端的参数聚合进一步将这种泛化能力融合到全局模型中,提升模型质量.不同于计算成本较高的传统知识蒸馏方法[21~24],本文提出的轻量级自蒸馏技术在不增加参与者计算成本的同时,仅需要少量的额外存储,便能显著提升本地模型的知识学习能力.为了进一步提高训练效率,FedSKD算法还采用了动态同步机制,进一步改善全局模型质量.该机制使得参与方上传参数的时间节点不再是完成固定次数的本地训练,在训练初期通过频繁同步缩小本地模型与全局模型的差异,在训练后期减少同步次数以提升模型准确率.这一机制在不增加任何计算和通信代价的前提下,进一步提升了全局模型的质量和训练效率,适合资源受限的边缘计算环境下的联邦学习.本文的贡献如下:

(1)本文提出了一种名为FedSKD的高效联邦学习算法,该算法将轻量级自蒸馏技术与联邦学习进行结合.在本地训练中,轻量级自蒸馏技术将前后批次预测结果的一致性作为知识,使得本地模型学习到更丰富的表征知识,提高了本地模型泛化能力和全局模型的性能,改善了联邦学习在非独立同分布数据场景下所面临的训练效率问题.

(2)为了进一步提高模型准确率,FedSKD引入了动态聚合机制.动态聚合机制通过在训练前期多聚合后期少聚合的方式,在没有引入额外计算资源的情况下,进一步提高了联邦学习的全局模型质量.

(3)本文探索了独立同分布和非独立同分布两种数据分布,在Fashion-MNIST和CIFAR10数据集上的实验结果表明,FedSKD算法的全局模型准确率相比于同为基于知识蒸馏的FedMLD算法,分别提高了约0.5%和2%,在达到相同全局模型准确率时的训练代价分别减少了约12.5%和58%.在CIFAR100数据集上,FedSKD算法和FedMLD算法的全局模型准确率相同,但达到相同全局模型准确率时的训练代价减少了约55%.

## 2 相关工作

### 2.1 联邦学习

为了在保护客户端数据隐私的条件下进行分布式训练,McMahan等人[4]于2017年提出了联邦平均算法FedAvg:服务端在训练开始前将统一的模型架构分发给训练参与方;参与方在本地数据上训练网络模型,并将新的模型参数上传至服务端;服务器平均聚合参与方的模型参数生成全局共享模型;最后,参与方下载全局模型进行新一轮的本地训练.然而在现实场景中,客户端数据存在着Non-IID分布的问题.对此,研究者们提出了多种联邦学习优化方法.一些研究通过改良训练中的损失函数,来减少局部模型和全局模型的差异,从而缓解数据异质性对联邦学习的影响.FedProx[6]在每个客户端上的本地训练的损失函数上添加近端项,以限制每个客户端本地模型更新对全局模型的影响.Scaffold[7]通过减少方差来纠正客户端本地训练的权重更新.FedLC[8]算法提出了一种用于本地训练的细粒度损失函数以缓解本地模型过拟合程度.然而上述方法在全局模型准确率上表现不佳[11].FedDyn[9]算法中每轮的每个客户端使用动态正则化器,以最小化局部模型和全局模型之间的差异.MOON[10]算法将模型对比学习与联邦学习结合,将对比全局模型和本地模型的特征差异加入到损失函数中,纠正本地训练中模型的

更新方向,然而该算法需要存储全局模型以及上一轮通信时的本地模型,并且在每一次训练迭代中,全局模型、本地模型和上一轮本地模型都需进行正向传播,增加了客户端的存储和计算代价,不适合资源受限的边缘设备.

### 2.2　知识蒸馏

知识蒸馏[21]是一种将教师模型的知识传递给学生模型,以提高学生模型的性能和泛化能力的深度学习技术. 知识蒸馏适用于资源受限的场景,因为它能够在减小模型规模的同时提高效率. 现有研究大多集中于如何更有效地将教师模型的知识传递给学生模型[21,23~25]. 然而,训练和推理教师模型通常需要大量的数据和计算资源,在资源受限的设备上难以实现[26]. 为解决这一问题,自蒸馏[21,27~31]应运而生,其通过让模型自身同时充当教师和学生,无需外部教师模型,从而减少了对计算资源的依赖,适合在计算资源有限的场景中应用. 自蒸馏的实现方式主要包括三类:(1)基于数据增强的自蒸馏技术[28,29]通过生成多样化的训练样本,提升了模型的鲁棒性和泛化能力,然而这种方法可能涉及用户隐私数据的获取;(2)基于辅助头的自蒸馏技术[30,31]通过在模型中添加轻量级的辅助网络,增强了模型的特征学习和多任务学习能力,但需要对模型结构进行一定的修改;(3)基于历史知识的自蒸馏技术[22]则依赖模型在训练过程中的历史信息,无需修改模型结构或获取隐私数据,相比于前两类自蒸馏技术更加简单轻量,在提升模型泛化能力的同时,不依赖外部教师模型,特别适用于资源受限的联邦学习中.

### 2.3　联邦知识蒸馏

受到知识蒸馏的启发,一些研究将知识蒸馏和联邦学习结合,用于改善Non-IID数据分布带来的影响. FedMD[14]和MHAT[16]算法借助公共数据集生成的知识指导本地模型训练,以提高本地模型泛化能力,进而提高全局模型收敛速率. FedHKT[17]、FedGKT[19]和DS-FL[20]算法通过在服务端吸收客户端的知识形成新的数据集,并反哺回每个客户端以提高本地模型泛化能力. 然而,借助公共数据集的方式在资源受限的客户端很无法实施,利用本地知识生成新的数据集的方式会导致知识和生成的数据集在参与者和服务端之间频繁交互,提高了通信成本. FedNTD[12]、FedCAD[13]、FedDistill[32]和FedLMD[33]算法利用知识蒸馏技术,在训练过程中使全局模型指导本地模型训练,以缓解Non-IID特性带来的问题,然而与MOON算法类似,客户端需要额外存储全局模型权重并在每次训练迭代中进行两次正向传播,提高了客户端的存储和计算代价. 鉴于以上问题和分析,本文将利用模型训练中的历史知识的自蒸馏与联邦学习进行结合,提出FedSDK算法. 该算法是一种基于自蒸馏的简单高效的联邦学习算法,该算法在每一次训练迭代中只需要一次正向传播,计算代价低,且无需借助公共数据集,无需存储全局模型权重,也不需要生成新的数据集,更加适合参与者资源受限条件下的联邦学习.

## 3　基于自蒸馏的联邦学习算法

### 3.1　联邦知识蒸馏

在介绍算法细节之前,本文首先介绍FedSKD算法的总体框架和相关符号定义. 本文考虑由$N$个边缘设备组成的客户端集合和一个服务端进行协同训练的场景. 其中,客户端集合表示为$\mathcal{U}$,$\mathcal{U}=\{u_1,u_2,\cdots,u_N\}$. 在$\mathcal{U}$中的任意客户端$u_k$拥有本地数据$\mathcal{D}_k=\{\boldsymbol{x}_i,y_i\}(i=1,2,\cdots,n_k)$,其中,向量$\boldsymbol{x}_i$和标量$y_i$分别是第$i$个训练样本的输入和期望输出. FedSKD算法的目标是通过最小化总损失$\mathcal{L}_{\text{total}}$来学习一个通用的全局模型$h(\omega)$,总体目标可以表示为

$$\min \mathcal{L}_{\text{total}}(\omega)=\frac{1}{|\mathcal{D}|}\sum_{k=1}^{N}\left|\mathcal{D}_k\right|\cdot\mathcal{L}_k(\omega) \tag{1}$$

其中,$\omega$为模型参数,$\mathcal{L}_k$为客户端$k$的损失函数. $\mathcal{D}$为总数据集,$|\mathcal{D}|$和$\left|\mathcal{D}_k\right|$分别表示总数据集和客户端$k$本地数据集的大小. 对于每一个参与训练的客户端$k$来说,其目标函数$\mathcal{L}_k$可以表示为

$$\mathcal{L}_k=\mathcal{L}_{\text{CE}}\left(\omega_k\right)+\lambda\cdot\mathcal{L}_{\text{SKD}}\left(\omega_k\right) \tag{2}$$

其中,$\omega_k$表示客户端$k$的本地模型权重,$\lambda_{\text{CE}}$表示常规模型训练得到的交叉熵损失,$\mathcal{L}_{\text{SKD}}$表示通过轻量级自蒸馏技术在训练过程中从本地数据中学习更多知识所产生的损失,用于提高模型的泛化能力,减少本地模型过拟合程度. $\lambda$是超参数,用于决定自蒸馏损失在目标$\mathcal{L}_k$中的比例.

图1展示了FedSKD算法的整体框架. 其左侧部分展示了客户端与服务端之间协同训练的流程. 与经典联邦学习不同的是,在协同训练开始前,服务端会根据动态同步机制,计算每个通信轮次$t$中客户端本地训练次数,机制的具体细节在3.3节中介绍. 在通信轮次$t$中,协同训练主要包含以下4个步骤.

(1) 客户端迭代次数设置:服务端得到在通信轮次为$t$时,客户端执行的本地训练次数$E_t$.

(2) 模型分发:服务端向每个客户端发送当前轮次最新的全局模型参数$\omega^t$以及本地训练次数$E_t$.

(3) 客户端$u_k$接收服务端发送的全局模型$\omega^t$后,利用本地数据$\mathcal{D}_k$使用轻量化自蒸馏技术,训练局部模型$h\left(\omega_k^t\right)$,最小化损失$\mathcal{L}_k$. 在经过$E_t$次本地训练后得到$\omega_k^{t+1}$,客户端计算本地模型的参数更新$\Delta\omega_k^{t+1}=\omega_k^{t+1}-\omega_k^t$,并将其发送至服务端.

(4) 服务端聚合客户端本地模型参数:当所有客户

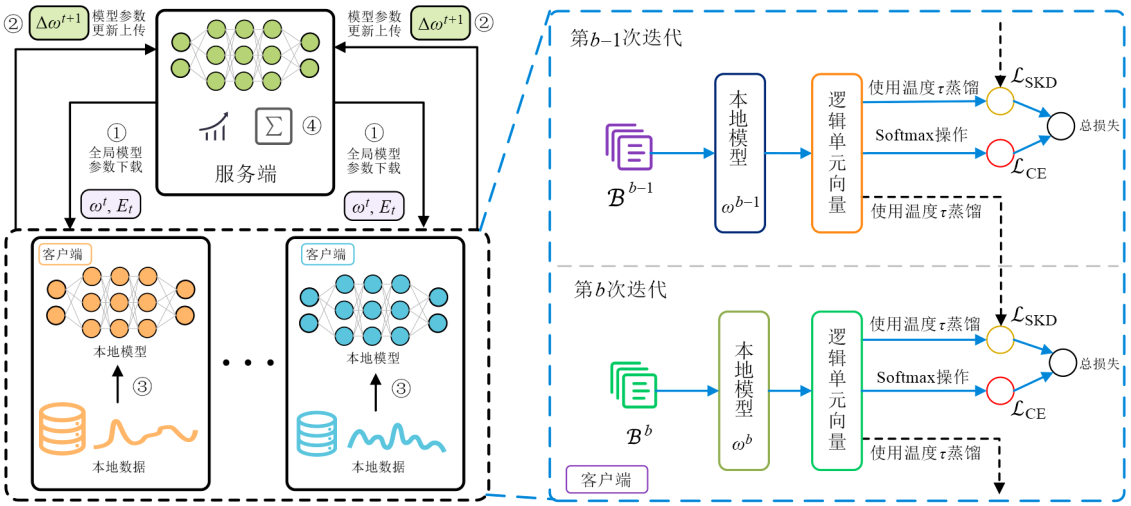



图 1 FedSKD算法整体框架图

端完成本地训练并将参数更新上传至服务端后，服务端根据客户端本地数据的大小，对所有的参数更新进行加权平均得到新的全局模型参数：

$$\omega^{t+1}=\omega^{t}+\frac{|\mathcal{D}_k|}{|\mathcal{D}|}\sum_{k=1}^{N}\Delta\omega_k^{t+1} \tag{3}$$

最后，FedSKD算法重复上述步骤，直到$\mathcal{L}_{\text{total}}(\omega)$最小或达到指定的通信轮次上限$T$. 服务端协同训练流程如算法1所示. 在图1右侧部分展示的客户端本地训练流程，客户端利用前后批次预测结果的一致性作为知识进行自整馏训练，挖掘并学习本地数据集中蕴含的隐藏知识，以增强本地模型对不平衡数据集的泛化能力，缓解过拟合现象. 自蒸馏的训练细节在3.2节中详细说明.

**算法1 服务端协同训练**

**输入**：客户端总数$N$，通信轮次$T$，客户端$k$的本地数据大小$|\mathcal{D}_k|$，全局数据集大小$|\mathcal{D}|$.

**输出**：全局模型$\omega$

1. 初始化全局模型$\omega^0$
2. 计算本地训练迭代次数集合
3. FOR $t=1$ TO $T$ DO
4. 向客户端发送全局模型$\omega^t$和$E_t$
5. FOREACH 每一个被选中的客户端$k$，并行执行
6. $\Delta\omega_k^{t+1}\leftarrow$客户端本地更新$(k,\omega^t,E_t)$
7. END FOREACH
8. $\omega^{t+1}=\omega^{t}+\frac{|\mathcal{D}_k|}{|\mathcal{D}|}\sum_{k=1}^{N}\Delta\omega_k^{t+1}$
9. END FOR
10. RETURN $\omega$

### 3.2 轻量自蒸馏设计

为了以较低的计算成本提高本地模型的泛化能力，本文提出了一种基于前一批次知识的轻量自蒸馏算法，该算法增强了本地模型对本地数据的学习能力，提高了训练效率. 该算法的核心是利用上一轮迭代生成的逻辑单元向量作为教师知识，对当前批次的学习进行指导. 通过将前一次迭代的输出作为软标签，让模型在训练时不仅关注当前批次的真实标签，还要匹配其与前一轮迭代输出之间的差异. 在这一过程中，本地模型同时担任教师和学生的角色：作为教师，它生成逻辑单元向量用于下一批次的训练指导；作为学生，它通过蒸馏知识来从上一批次的数据中挖掘隐藏的知识. 为了更清晰地表达，假设客户端$u_k$的本地数据集$\mathcal{D}_k$有$M$个类别，并在一轮本地训练中需要执行$B$次迭代，在第$b\in B$次迭代中得到批次数据表示为$\mathcal{B}^b=\left(x_i^b,y_i^b\right)\left\{i=1,2,\cdots,\left|\mathcal{B}^b\right|\right\}$，其中$\left|\mathcal{B}^b\right|$是批次的大小. 同时，将第$b$次迭代中的本地模型参数表示为$\omega^b$.

首先，当第$b$次迭代开始时，批次数据$\mathcal{B}^b$会被输入至模型$h\left(\omega^b\right)$进行正向传播，获得对应的逻辑单元向量. 我们将其定义为$\boldsymbol{z}^b=\left[z_1^b,z_2^b,\cdots,z_m^b,\cdots,z_M^b\right]$，$z_m^b$是类别$m$在$\boldsymbol{z}^b$上的分量. 然后，计算批次数据$\mathcal{B}^b$在模型$h\left(\omega^b\right)$上的Softmax预测分布向量，定义模型$h\left(\omega^b\right)$对$\mathcal{B}^b$的Softmax预测分布向量为$\boldsymbol{p}^b=\left[p_1^b,p_2^b,\cdots,p_m^b\cdots,p_M^b\right]$，$p_m^b$是类别$m$的在Softmax预测分布向量上的分量，其公式为

$$p_m^b=\frac{\exp\left(z_m^b\right)}{\sum_{j=1}^{M}\exp j\left(z_j^b\right)} \tag{4}$$

此时，交叉熵损失$\mathcal{L}_{\text{CE}}$可以用如下公式定义：

$$\mathcal{L}_{\text{CE}}=\text{H}\left(\boldsymbol{y}^b,\boldsymbol{p}^b\right) \tag{5}$$

其中，H为交叉熵损失函数，$\boldsymbol{y}^b$为批次$\mathcal{B}^b$的期望输出向量. 在$\mathcal{L}_{\text{CE}}$计算结束后，客户端在本地保存逻辑单元向量$\boldsymbol{z}^b$以指导对下一个批次数据$\mathcal{B}^{b+1}$的训练.

其次，使用批次数据$\mathcal{B}^{b-1}$产生的知识对本批次数

据$\mathcal{B}^b$的训练进行指导. 具体来说,在上一次迭代结束后,上一批次数据$\mathcal{B}^{b-1}$的逻辑单元向量$\boldsymbol{z}^{b-1}$由客户端保存在本地,客户端对$\boldsymbol{z}^{b-1}$进行知识蒸馏,得到教师知识$\boldsymbol{P}^{\tau,b-1}=\left[p_1^{\tau,b-1},p_2^{\tau,b-1},\cdots,p_m^{\tau,b-1},\cdots,p_M^{\tau,b-1}\right]$,其中$p_m^{\tau,b-1}$是类别$m$在教师知识$\boldsymbol{P}^{\tau,b-1}$上的分量,被如下公式定义:

$$p_m^{\tau,b-1}=\frac{\exp\left(z_m^{b-1}/\tau\right)}{\sum_{j=1}^{M}\exp j\left(z_j^{b-1}/\tau\right)} \tag{6}$$

其中,$\tau$是蒸馏温度系数,用于控制知识的平滑程度,较高的温度会导致更均匀的类别概率分布,从而产生与标签平滑类似的正则化效果. 同理,我们将第$b$次迭代得到逻辑单元向量$\boldsymbol{z}^b$进行蒸馏,形成学生知识$\boldsymbol{P}^{\tau,b}$,然后使用KL(Kullback-Leibler)散度来衡量教师知识和学生知识的差异,并将其作为自蒸馏损失:

$$\mathcal{L}_{\mathrm{SKD}}=\tau^2\cdot\mathrm{KL}\left(\boldsymbol{P}^{\tau,b-1}\,\|\,\boldsymbol{P}^{\tau,b}\right) \tag{7}$$

其中,KL散度用于衡量当前批次的输出分布与上一次迭代的输出分布之间的差异. 通过最小化这种差异,模型能够逐渐从历史迭代训练中获得更丰富的特征表示,从而提升其泛化能力. 在这之后,客户端将$\mathcal{L}_{\mathrm{CE}}$和$\mathcal{L}_{\mathrm{SKD}}$组合起来,共同形成总损失$\mathcal{L}$:

$$\mathcal{L}=\mathcal{L}_{\mathrm{CE}}+\lambda\cdot\mathcal{L}_{\mathrm{SKD}} \tag{8}$$

其中,$\lambda$是平衡两个损失的系数,用于控制$\mathcal{L}_{\mathrm{CE}}$和$\mathcal{L}_{\mathrm{SKD}}$在总损失$\mathcal{L}$中的占比.

最后,客户端根据总损失函数$\mathcal{L}$,进行反向传播,更新本地模型参数,并进行下一次迭代训练. 值得注意的是,当客户端进行第一次迭代训练时,无法获得逻辑单元向量$\boldsymbol{z}^{b-1}$,也就无法进行知识蒸馏,此时客户端的总损失$\mathcal{L}$只由$\mathcal{L}_{\mathrm{CE}}$构成. 在上述轻量自蒸馏算法的训练过程中,客户端仅存储上一次迭代得到的逻辑单元向量,存储一批逻辑单元向量只需要很少的额外成本. 从计算成本上来分析,轻量自蒸馏算法在每次迭代中只需一次前向传播和一次反向传播,与正常训练相同,该方法增加的计算开销主要来自KL散度的计算,虽然需要额外的计算资源和内存来存储和处理前一次迭代的输出向量,但是所需的资源是极少的. 因此,基于前一批次知识的轻量自蒸馏算法与标准训练过程相比,计算和存储负担相对轻微,训练成本几乎相同,更适合在资源有限的设备上执行,其训练流程如算法2所示.

**算法2　客户端基于轻量自蒸馏的本地训练**

**输入**:第$t$轮的全局模型参数$\omega^t$,本地训练次数$E_t$

**输出**:本地模型和全局模型参数的差值$\Delta\omega_k^{t+1}$

```
从服务端收到最新的全局模型ω^t和本地训练迭代次数E_t
FOR e=1 TO E_t DO
  LastLogits = None
  FOR b=1 TO B DO
    z^b ← h(ω_k^b)
    Loss = L_CE(z^b, y^b)
    IF LastLogits != None DO
      Teacher=Softmax(LastLogits /τ)
      Student=Softmax(Logits /τ)
      Loss + = λ * L_SKD (Teacher,Student)
    END IF
    Loss.backward()
    LastLogits = z^b
  END FOR
END FOR
向服务端提交Δω_k^{t+1}
```

### 3.3　动态同步机制设计

在传统联邦学习中,客户端在本地经过一定次数的本地训练后,将本地模型参数上传至服务器进行同步,然而这种周期性同步的策略并非最优[34]. 因此,为了进一步提升全局模型的质量,FedSKD算法采用动态同步机制,其策略为训练前期客户端进行少量本地训练和多次同步,训练后期客户端进行大量本地训练和少量同步. 在Non-IID的数据分布下,客户端的本地训练会使本地模型的更新方向偏离全局模型,且本地训练次数越大,偏离程度也就越大. 在训练前期进行少量的本地训练而多次同步,有利于约束本地模型与全局模型的更新方向,改善训练效率. 在训练后期,增加本地训练迭代次数,使得本地模型从本地数据集中学习更多的知识,更有助于全局模型质量上的提高.

定义客户端本地训练迭代次数集合为$\mathcal{I}=\{E_1,E_2,\cdots,E_t,\cdots,E_T\}$,其中,$T$代表通信轮次的上限,$E_t$表示在第$t$通信轮次的本地迭代次数. 客户端本地训练总次数表示为$\mathcal{E}=\sum_{t=1}^{T}E_t$. 本文提出的动态同步机制在计算和通信成本一定的条件下动态调节客户端本地训练迭代次数. 在联邦学习开始前,服务端根据给定$T$和$\mathcal{E}$,得到客户端本地训练迭代次数集合$\mathcal{I}$. 客户端在训练前期进行少量的本地训练,训练后期进行大量的本地训练的策略被表示为

$$E_1\leqslant E_2\leqslant\cdots\leqslant E_T \tag{9}$$

这意味着,客户端在训练开始时的本地训练次数$E_1$最小,随着通信轮次的增加,本地训练次数随之增加,到训练结束本地训练次数$E_T$达到最大. 为了实现上述策略,本文首先计算本地训练次数最大值$E_T$:

$$E_T=\mathrm{floor}\left(\left(\frac{T}{T+\delta}+1\right)\cdot\frac{\mathcal{E}}{T}\right) \tag{10}$$

其中,$\delta$是超参数,用于控制$E_T$的大小,当$\delta$较小时,$E_T$的值越接近$2\frac{\mathcal{E}}{T}$,当$\delta$较大时,$E_T$的值越接近$\frac{\mathcal{E}}{T}$. 随后,本文引入差值$\Delta d$,用于计算其他轮次下的本地训练次数:

$$\Delta d=\frac{2\left(\frac{\mathcal{E}}{T}-E_T\right)}{T-1} \tag{11}$$

其中，差值的$\Delta d$大小受到$E_T$的影响，当$E_T$的值越接近$\frac{\mathcal{E}}{T}$时，$\Delta d$越趋近于0，这意味着整个训练过程中本地训练次数变化不大，接近静态同步机制．当$E_T$的值越接近$2\frac{\mathcal{E}}{T}$时，$\Delta d$越大，整个训练过程中本地训练轮次变化较大．结合$E_T$和$\Delta d$，可以通过下式得到在通信轮次$t$时的本地训练次数$E_t$：

$$E_t=(T-t)\Delta d \tag{12}$$

最后，动态同步机制得到本地训练次数的集合$\mathcal{I}$使用如下公式表示：

$$\mathcal{I}=\left\{E_T+(T-1)\Delta d, E_T+(T-2)\Delta d, \cdots, E_T\right\} \tag{13}$$

在动态同步机制下，客户端本地训练迭代次数集合$\mathcal{I}$是在给定通信轮次上限$T$和客户端本地训练总次数$\mathcal{E}$的条件下计算得出的，其训练成本和传统的周期性同步机制相同，因此动态同步机制没有增加训练成本．

### 3.4 计算代价分析

在本节中，我们将FedSDK算法和对比的基线算法的计算代价进行分析．如文献[29]中所述，本地训练中的主要计算负担来自模型的正向传播和反向传播的．设$\rho_f$和$\rho_b$分别表示模型正向传播和反向传播的操作次数．客户端训练一个批次$\mathcal{B}$所需的计算成本$C$，使用如下公式计算：

$$C=\left(\rho_f+\rho_b\right)|\mathcal{B}| \tag{14}$$

本文对比的基线算法为FedAvg[4]、FedProx[6]、MOON[10]、FedNTD[12]、FedDistill[32]和FedMLD[33]．对于FedSKD算法来说，对于每一个批次的训练，都只需要一次模型的正向传播和一次反向传播操作，因此$\rho_f$和$\rho_b$都为1，同理，FedAvg[4]、FedProx[6]和FedLC[8]算法的$\rho_f$和$\rho_b$也都为1．FedNTD[12]、FedDistill[32]和FedMLD[33]算法引入了知识蒸馏，在训练中全局模型也进行正向传播，导致在一个批次的训练中需进行两次正向传播，一次反向传播，其$\rho_f$和$\rho_b$分别为2和1．MOON[10]算法在本地训练阶段同时考虑上一轮的本地模型与全局模型，因此每训练一个批次数据时需执行三次正向传播和一次反向传播，其$\rho_f$和$\rho_b$分别为3和1．由于上述所有方法的反向传播次数都为1，因此计算成本$C$可以被简化为

$$C=\rho_f|\mathcal{B}| \tag{15}$$

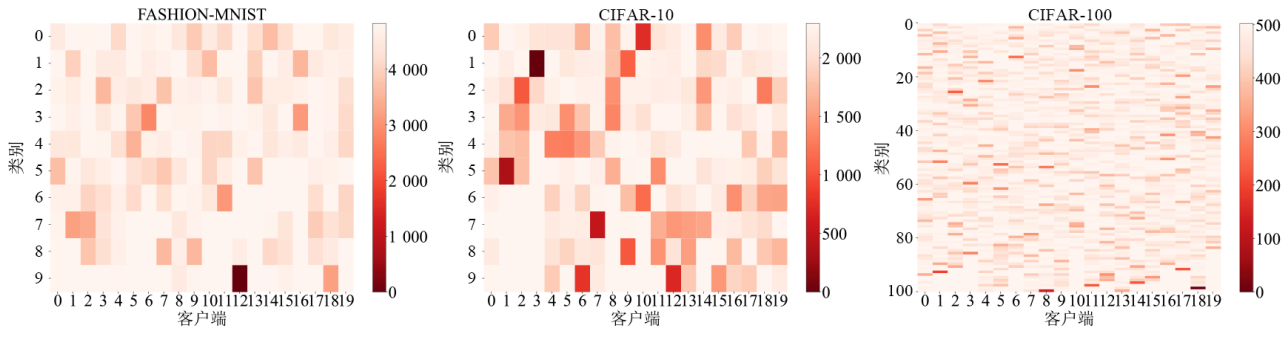
经过上述分析，FedSKD的训练一个批次的计算成本较低，与FedAvg[4]和FedProx[6]算法相同，低于FedNTD[12]、FedDistill[33]、FedMLD[34]和MOON[10]算法．

## 4 性能评估

### 4.1 实验数据集

为了更全面地评估FedSKD算法，本文在三个不同学习难度的数据集上进行了实验，即Fashion-MNIST[35]、CIFAR-10和CIFAR-100[36]．其中，Fashion-MNIST数据集由10个类别的70 000个时尚产品的28×28灰度图像组成，每个类别有7 000张图像．CIFAR-10和CIFAR-100这两个数据集都包含60 000张32×32的彩色图像．CIFAR-10数据集共有10个类别，而CIFAR-100数据集有100个类别．相比于Fashion-MNIST，CIFAR-10和CIFAR-100这两个数据集更复杂，学习的难度更大．

为了模拟客户端数据集的非独立同分布特性，本文使用狄利克雷分布来划分客户端的本地数据集．数据集的分布受狄利克雷系数$\alpha$的影响．当$\alpha$越大时，所得概率分布越趋近于均匀分布，当$\alpha$越小时，生成的数据集倾斜更为显著，更接近非独立同分布．在实验中，本文将$\alpha$设置为0.5，三个数据集的数据分布如图2所示，可以很容易观察到不同类别的数据样本在20个客户端上的分布并不均匀，有些类别的数据样本在某个客户端上的数据很少，甚至为0.




注：颜色条表示数据样本的数量，每个矩形代表一个客户端中特定类别的数据样本数量．

图2 客户端数据集分布热力图，狄利克雷系数为0.5

### 4.2 实验设置

对于全局训练,本文将简单的Fashion-MNIST数据集的通信轮次设置为100,将复杂的CIFAR-10和CIFAR-100数据集的通信轮次设置为200,并将参训客户端数量设置为20. 对于本地训练,在上述设置下,设置每个客户端在一个通信轮次中的本地训练迭代次数为5. 对于本地训练,所有算法的客户端使用SGD优化器,批次大小设置为128. 在设置FedSKD算法的超参数方面,本文对温度$\tau=\{1,2,3,4\}$和超参数$\delta=\{10,50,100,200\}$进行网格搜索,发现在Fashion-MNIST数据集上温度$\tau$设置为4,超参数$\delta$设置为100得到最优结果,在CIFAR-10数据集上温度$\tau$设置为2,超参数$\delta$设置为10的最优结果,在CIFAR-10数据集上温度$\tau$设置为4,超参数$\delta$设置为10得到最优结果. 此外,对于所有数据集,平衡系数$\lambda$设置为1. 有关其他对比算法中的超参数,我们遵循其在论文中推荐的参数.

在模型的设置上,本文选取了两种不同复杂程度的模型作为主干网络进行相应实验. 对于小型数据集Fashion-MNIST,本文使用结构简单的Lenet5作为训练模型. 对于大型数据集CIFAR-10和CIFAR-100,则使用结构相对复杂的ResNet34作为训练模型. 与现有相关工作一致,本文实验采用如下指标:(1)准确率,使用全局模型在测试集上的Top-1准确率来衡量训练后模型的性能,准确率越高,代表算法效果越好;(2)训练效率,使用训练全局模型达到目标精度所需的训练代价,即通信代价和计算代价之和,所需训练代价越小,代表算法的训练效率越高.

为了公平地检验本文提出的FedSKD算法的有效性,本文将其与6个具有代表性的联邦学习算法进行了比较,这6个算法分别为FedAvg[4]、FedProx[6]、MOON[10]、FedNTD[12]、FedDistill[33]和FedMLD[34],其中,后三个算法是基于知识蒸馏的联邦学习算法. 由于FedSDK算法与对比算法在客户端上传模型参数的行为上没有差异,即每轮通信中客户端向服务器发送的模型参数大小几乎相同,因此本文将通信轮次视为通信代价. 在统计训练的计算代价时,本文将所有的客户端的计算代价之和作为算法的计算代价,在该条件下,由于各个算法在训练过程中使用相同的数据集,因此计算代价只和模型的正向传播次数有关. 在3.4节的基础上,将式(15)修改为

$$C=\rho|\mathcal{D}|\cdot E_t \tag{16}$$

其中,$\rho$为训练一次全局数据集所需的模型正向传播次数;$|\mathcal{D}|$为全局数据集大小;$E_t$为第$t$通信轮次的本地迭代次数,即在$t$通信轮次中要训练多少次全局数据集. 为了计算方便,将$|\mathcal{D}|$设置为1,$\rho$的值和3.4节中$\rho_f$的值相同.

### 4.3 准确率分析

表1展示了FedSKD算法与对比算法在全局模型准确率上的性能,其中最优结果以加粗形式突出显示,次优结果则通过下划线标识,其结果揭示了FedSKD算法在全局模型准确率方面超越了所有对比算法,表现出其良好的泛化能力. 具体来说,与FedAvg算法相比,FedSKD算法在Fashion-MNIST、CIFAR-10以及CIFAR-100数据集的准确率分别提升了1.2%、16.74%和13.83%. 虽然FedDistill、FedNTD和FedMLD算法的准确率与FedSKD算法相差较少,甚至在CIFAR-100数据集上,FedMLD算法的准确率略微优于FedSKD算法,但它们的计算代价却约为FedSKD算法的两倍. 此外,随着数据集复杂度的提升,FedSKD算法在准确率提高方面的表现尤为突出,在CIFAR100数据集上,相比于FedAvg算法在模型准确率上获得了1.45倍的提升. 以上分析表明,面对Non-IID分布下的数据,FedSKD算法以较低计算成本,展现出了良好的学习效果,并且对于更为复杂的任务显示出更强的适应性,证明了其在边缘联邦学习场景下拥有良好的实用性和可行性.

**表1** FedSKD算法和对比算法的全局模型Top-1准确率 单位:%

| 对比算法 | Fashion-MNIST | CIFAR-10 | CIFAR-100 |
|---|---|---|---|
| FedAvg | 87.56 | 52.67 | 30.43 |
| FedProx | 88.02 | 50.32 | 29.70 |
| MOON | 87.93 | 61.68 | 31.94 |
| FedDistil | 87.96 | 64.84 | 42.63 |
| FedNTD | 88.39 | 66.14 | 43.20 |
| FedMLD | <u>88.41</u> | <u>67.68</u> | **44.32** |
| FedSKD | **88.74** | **69.41** | <u>44.26</u> |

### 4.4 计算代价分析

表2列出了所有对比算法在实现相同目标准确率时的通信代价、计算代价和训练代价,为方便统计,目标准确率设定为FedAvg算法的最高准确率的向下取整,最优结果以加粗形式显示,次优结果则通过下划线标注,通信代价、计算代价和训练代价按照4.2节中的设定进行计算. 图3描绘了FedSKD算法与各对比算法在准确率上的变化曲线. 从计算代价来看,FedSDK达到目标准确率所需的计算代价在Fashion-MNIST,CIFAR-10以及CIFAR-100数据集上均为最优,相较于计算代价次优的算法分别降低了20%、72%和71%. 从结合通信代价和计算代价的训练代价来看,FedSDK达到目标准确率所需的训练代价也在三个数据集上表现出最优结果,相较于训练代价次优的算法分别降低了17%、58%和55%. 同时,随着数据集复杂度的增加,FedSKD算法在计算代价和训练代价上的相比于FedAvg算法,降低幅度逐渐增大. 上述结果证明,在Non-IID的数据场景下,FedSKD算法不仅拥有相较于其他算法更强的泛化能力,而且其较低的训练代价使其更适合资源受限的边缘计算.

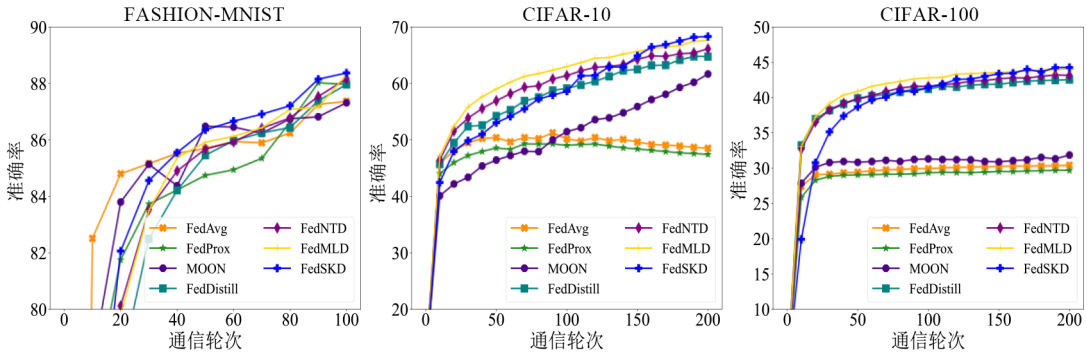



图3 在不同数据集下,FedSKD算法与对比算法的准确率曲线

**表2 FedSKD算法和对比算法在达到相同准确率所需的通信代价,计算代价和训练代价**

| 算法 | Fashion-MNIST | | | CIFAR-10 | | | CIFAR-100 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 准确率=85% | | | 准确率=50% | | | 准确率=30% | | |
| | 通信代价 | 计算代价 | 训练代价 | 通信代价 | 计算代价 | 训练代价 | 通信代价 | 计算代价 | 训练代价 |
| FedAvg[4] | 35 (1.00×) | 175 (1.00×) | 210 (1.00×) | 42 (1.00×) | 210 (1.00×) | 252 (1.00×) | 108 (1.00×) | 540 (1.00×) | 648 (1.00×) |
| FedProx[6] | 61 (1.74×) | 305 (1.74×) | 366 (1.74×) | 114 (2.71×) | 570 (2.71×) | 684 (2.71×) | — | — | — |
| MOON[10] | **25 (0.71×)** | 475 (.13×) | 500 (2.38×) | 90 (2.14×) | 1 350 (6.42×) | 1 440 (5.71×) | 22 (0.20×) | 330 (0.60×) | 352 (0.54×) |
| FedDistill[32] | 50 (1.42×) | 500 (4.26×) | 550 (2.62×) | 24 (0.57×) | 240 (1.71×) | 264 (1.05×) | **10 (0.09×)** | 100 (0.18×) | 110 (0.17×) |
| FedNTD[12] | **25 (0.71×)** | 250 (1.42×) | 275 (1.31×) | 20 (0.47×) | 200 (0.94×) | 220 (0.87×) | **10 (0.09×)** | 100 (0.18×) | 110 (0.17×) |
| FedMLD[33] | 40 (1.33×) | 400 (3.99×) | 440 (2.10×) | **18 (0.42×)** | 180 (0.84×) | 198 (0.79×) | **10 (0.09×)** | 100 (0.18×) | 110 (0.17×) |
| **FedSKD** | 35 (1.00×) | **140 (0.80×)** | **175 (0.83×)** | 32 (0.76×) | **51 (0.24×)** | **83 (0.33×)** | 21 (0.19×) | **29 (0.05×)** | **50 (0.08×)** |

### 4.5 通信代价分析

由表2的结果可知,FedSKD在Fashion-MNIST、CIFAR-10及CIFAR-100数据集上达到目标准确率所需的通信代价均优于多数算法,稍逊于基于知识蒸馏的联邦学习算法.这一表现归因于FedSKD算法采用的动态同步机制,该机制在训练前期减少本地训练次数并增加同步频率,而在训练后期则增加本地训练次数并减少同步频率,从而导致全局模型准确率上升速度在前期略显缓慢.然而,从表2和图3可知,FedSKD算法在中后期保持了较为稳定的准确率上升速度,从而在训练结束后获得了最优的模型准确率.同时,综合通信代价和计算代价来看,FedSKD在达到相同准确率时所使用的训练代价最小.因此,FedSKD算法适用于资源受限的联邦学习场景.

### 4.6 消融实验

FedSKD整体可分为两个部分:轻量自蒸馏框架和动态同步机制.为了证明上述两部分的有效性,本文对其进行了消融实验.在消融实验中,采用Fashion-MNIST、CIFAR-10和CIFAR-100数据集进行实验,实验相关参数与4.3节相同.本文将FedAvg算法作为基线算法,分别与去掉轻量自蒸馏的FedSKD算法(FedSDK w/o SDK)、去掉动态同步机制的FedSKD算法(FedSDK w/o AE)和FedSKD算法进行实验,其结果如图4所示.

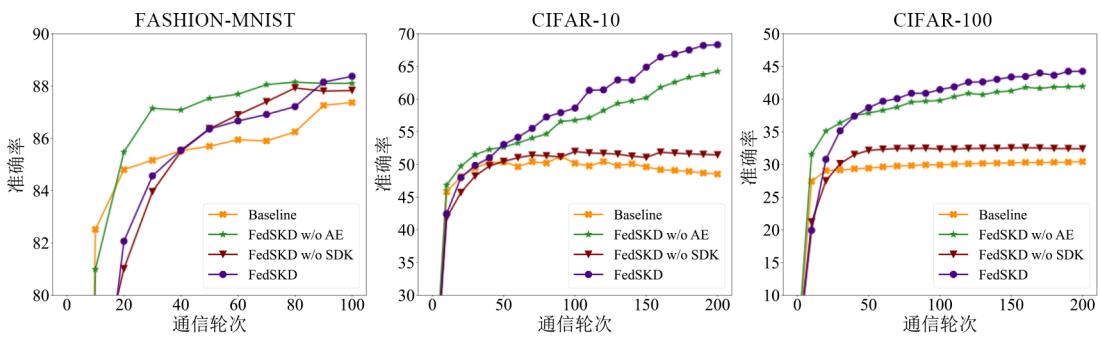



图4 轻量级自蒸馏和动态同步的有效性

首先,分析轻量自蒸馏的有效性. 轻量自蒸馏通过学习本地数据集中的隐藏知识,改善本地模型的泛化能力,进而提高全局模型的质量. 从图4中可以看出,只使用轻量自蒸馏训练的FedSKD w/o AE算法在Fashion-MNIST、CIFAR-10和CIFAR-100数据集上的全局模型准确率相比于基线算法分别提高了约1%、13和11%. 上述结果证明了轻量自蒸馏的有效性.

然后,分析动态同步机制的有效性. 本文提出的动态同步机制要求在客户端在训练前期少训练、多同步,从而以较小的训练代价减少本地模型与全局模型的差异. 然而客户端本地训练轮次的减少会导致训练前期全局模型准确率的增长速度较慢. 动态同步机制要求在客户端在训练后期多训练、少同步,以此进一步提高全局模型准确率. 图4中基线算法和只使用动态同步机制的FedSDK w/o SDK算法和基线算法在三个数据集上的迭代曲线印证了这一观点. FedSDK w/o SDK算法的全局模型准确率在通信轮次较少时低于基线算法,在通信轮次较多时高于基线算法. 到训练结束,FedSDK w/o SDK算法比基线算法的全局模型准确率在三个数据集上分别提高了1%、1%和2%. 最后,同时结合轻量自蒸馏和动态同步机制的FedSKD算法获得了最优模型准确率.

## 5　结论

当前联邦学习方法由于设备资源受限和数据异质性等问题,导致出现部分客户端训练时间长、训练效率低下等问题. 本文提出了一种结合自蒸馏的联邦学习算法FedSKD,该算法由轻量自蒸馏模块和动态同步模块组成. 轻量自蒸馏模块要求在客户端在训练中将上一批次的预测结果作为教师模型的知识,来指导本批次的训练过程,以提高模型的泛化能力并加快全局模型的收敛速度,动态同步模块通过调节客户端本地训练轮次和同步的频率,在不增加任何计算和通信代价的前提下,进一步提升了全局模型的质量. FedSKD算法在训练过程中不需要引入额外模型,也不需要借助额外数据集,是一种简单高效的联邦学习算法. 实验表明,与6种主流联邦学习算法相比,FedSKD算法在非独立同分布数据下以较小的训练代价显著提升了模型准确率,提高了训练效率.

在未来的工作中,我们将首先关注边缘联邦学习场景下,客户端因系统异构性而导致的落伍者问题,设计训练效率更高的联邦学习算法,其次关注如何从模型压缩角度,减少客户端和服务端的通信数据量.

**参考文献**


[1] BROWN T B, MANN B, RYDER N, et al. Language models are few-shot learners[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM, 2020: 1877-1901.

[2] NGUYEN D C, DING M, PATHIRANA P N, et al. Federated learning for industrial Internet of Things in future industries[J]. IEEE Wireless Communications, 2021, 28(6): 192-199.

[3] ABDULRAHMAN S, TOUT H, OULD-SLIMANE H, et al. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond[J]. IEEE Internet of Things Journal, 2020, 8(7): 5476-5497.

[4] MCMAHAN H B, MOORE E, RAMAGE D, et al. Communication-efficient learning of deep networks from decentralized data[J]. Proceedings of Machine Learning Research, 2017, 54: 1273-128.

[5] SINGH N, RUPCHANDANI J, ADHIKARI M. Personalized federated learning for heterogeneous edge device: Self-knowledge distillation approach[J]. IEEE Transactions on Consumer Electronics, 2024, 70(1): 4625-46.

[6] LI T, SAHU A K, ZAHEER M, et al. Federated optimization in heterogeneous networks[J]. Proceedings of Machine Learning and Systems, 2020, 2(3): 429-450.

[7] KARIMIREDDY S P, KALE S, MOHRI M, et al. Scaffold: Stochastic controlled averaging for federated learning[J]. Proceedings of Machine Learning Research, 2020, 119: 5132-514.

[8] ZHANG J, LI Z Q, LI B, et al. Federated learning with label distribution skew via logits calibration[J]. Proceedings of Machine Learning Research, 2020, 162: 26311-26329.

[9] JIN C, CHEN X D, GU Y, et al. FedDyn: A dynamic and efficient federated distillation approach on recommender system[C]//2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS). Piscataway: IEEE, 2023: 786-79.

[10] LI Q B, HE B S, SONG D. Model-contrastive federated learning[C]//2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2021: 10713-10720.

[11] DENG W, CHEN X T, LI X Y, et al. Adaptive federated learning with negative inner product aggregation[J]. IEEE Internet of Things Journal, 2024, 11(4): 6570-6581.

[12] LEE G, JEONG M, SHIN Y, et al. Preservation of the global knowledge by not-true distillation in federated learning[J]. Advances in Neural Information Processing

Systems, 2020, 35(11): 38461-38474.
[13] HE Y T, CHEN Y Q, YANG X D, et al. Class-wise adaptive self distillation for federated learning on non-IID data (student abstract) [J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, 36(11): 12967-12968.
[14] LI D L, WANG J P. FedMD: Heterogenous federated learning via model distillation[EB/OL]. (2019-10-08)[2024-04-09]. https://arxiv.org/abs/1910.03581v1.
[15] HE C Y, ANNAVARAM M, AVESTIMEHR S, et al. Group knowledge transfer[C]//Proceedings of the 34th International Conference on Neural Information Processing Systems. New York: ACM, 2020: 14068-14080.
[16] HU L, YAN H Y, LI L, et al. MHAT: An efficient model-heterogenous aggregation training scheme for federated learning[J]. Information Sciences, 2021, 560: 493-50.
[17] DENG Y H, REN J, TANG C, et al. A hierarchical knowledge transfer framework for heterogeneous federated learning[C]//IEEE INFOCOM 2023 - IEEE Conference on Computer Communications. Piscataway: IEEE, 2023: 1-10.
[18] ITAHARA S, NISHIO T, KODA Y, et al. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data[J]. IEEE Transactions on Mobile Computing, 2020, 22(1): 191-205.
[19] YAO D Z, PAN W N, DAI Y T, et al. FedGKD: Toward heterogeneous federated learning via global knowledge distillation[J]. IEEE Transactions on Computers, 2024, 73(1): 3-17.
[20] ZHANG J, GUO S, GUO J C, et al. Towards data-independent knowledge transfer in model-heterogeneous federated learning[J]. IEEE Transactions on Computers, 2020, 72(10): 2888-2901.
[21] SU T T, ZHANG J S, YU Z Y, et al. STKD: Distilling knowledge from synchronous teaching for efficient model compression[J]. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(12): 10051-10064.
[22] SHEN Y Q, XU L W, YANG Y Z, et al. Self-distillation from the last mini-batch for consistency regularization[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2022: 11933-1194.
[23] HINTON G, VINYALS O, DEAN J. Distilling the knowledge in a neural network[EB/OL]. (2015-04-09)[2024-04-09]. https://arxiv.org/abs/150.02531v1.
[24] LI M Y, LIN J, DING Y Y, et al. GAN compression: Efficient architectures for interactive conditional GANs[C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2020: 5284-5294.
[25] PARK W, KIM D, LU Y, et al. Relational knowledge distillation[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Piscataway: IEEE, 2019: 3962-3971.
[26] 邵仁荣, 刘宇昂, 张伟, 等. 深度学习中知识蒸馏研究综述[J]. 计算机学报, 2022, 45(8): 1638-1673.
SHAO R R, LIU Y A, ZHANG W, et al. A survey of knowledge distillation in deep learning[J]. Chinese Journal of Computers, 2022, 45(8): 1638-1673. (in Chinese)
[27] LEE H, HWANG S J, SHIN J, et al. Self-supervised label augmentation via input transformations[C]//Proceedings of the 37th International Conference on Machine Learning. New York: ACM, 2020: 5714-5724.
[28] XU T B, LIU C L. Data-distortion guided self-distillation for deep neural networks[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 5565-5572.
[29] LONG Z X, MA F Y, SUN B, et al. Diversified branch fusion for self-knowledge distillation[J]. Information Fusion, 20, 90: 12-.
[30] ZHANG L F, BAO C L, MA K S. Self-distillation: Towards efficient and compact neural networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 44(8): 4388-4404.
[31] FURLANELLO T, LIPTON Z, TSCHANNEN M, et al. Born again neural networks[J]. Proceedings of Machine Learning Research, 2018, 80: 1607-1616.
[32] JIANG D L, SHAN C, ZHANG Z H. Federated learning algorithm based on knowledge distillation[C]//2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE). Piscataway: IEEE, 2020: 163-167.
[33] LU J H, LI S K, BAO K X, et al. Federated learning with label-masking distillation[C]//Proceedings of the 31st ACM International Conference on Multimedia. New York: ACM, 2023: 222-230.
[34] ZHANG H, WU T T, CHENG S Y, et al. Aperiodic local SGD: Beyond local SGD[C]//Proceedings of the 51st International Conference on Parallel Processing. New York: ACM, 2023: 1-10.
[35] XIAO H, RASUL K, VOLLGRAF R. Fashion-mnist: A

novel image dataset for benchmarking machine learning algorithms[EB/OL]. (2017-09-15) [2024-04-09]. https://arxiv.org/pdf/1708.07747.

[36] ALEX K. Learning multiple layers of features from tiny images[EB/OL]. (2009-08-08) [2024-04-09]. https://www.cs.toronto.edu/kriz/learning-features-2009-TR.

## 作者简介

**刘　松**　男,1998年10月出生于河北省邢台市. 现为南开大学计算机学院博士研究生. 主要研究方向为边缘计算、联邦学习与5G网络架构.

E-mail: liusong0918@mail.nankai.edu.cn

**罗杨宇**　男,2001年3月出生于福建省漳州市. 现为南开大学网络空间安全学院硕士研究生. 主要研究方向为边缘计算、5G网络架构与算力网络.

E-mail: lyy@mail.nankai.edu.cn

**许佳培**　男,2001年5月生于河北承德市. 现为南开大学计算机学院硕士研究生. 主要研究方向为视频传输与边缘计算.

E-email: 2120230702@nankai.edu.cn

**张建忠**　男,1964年6月出生于河北省石家庄市. 现为南开大学网络空间安全学院教授、博士生导师. 主要研究方向为计算机网络与网络安全.

E-mail: zhangjz@nankai.edu.cn