

分类号: TP311.5
密 级: 无

单位代码: 10335
学 号: _____

浙江大学

硕士学位论文



中文论文题目: 基于知识蒸馏的联邦学习算法研究与实现
英文论文题目: Research and Implementation of Federated Learning Based on Knowledge Distillation

申请人姓名: _____

指导教师: _____

专业学位类别: 电子信息

专业学位领域: 软件工程

所在学院: 软件学院

论文提交日期 20XX 年 XX 月 XX 日

基于知识蒸馏的联邦学习算法研究与实现



论文作者签名: _____

指导教师签名: _____

论文评阅人 1: _____

评阅人 2: _____

评阅人 3: _____

评阅人 4: _____

评阅人 5: _____

答辩委员会主席: _____

委员 1: _____

委员 2: _____

委员 3: _____

委员 4: _____

委员 5: _____

答辩日期: _____

Research and Implementation of Federated
Learning Based on Knowledge Distillation



Author's signature: _____

Supervisor's signature: _____

Thesis Reviewer 1: _____

Thesis Reviewer 2: _____

Thesis Reviewer 3: _____

Thesis Reviewer 4: _____

Thesis Reviewer 5: _____

Chair: _____
(Committee of oral defence)

Committeeman 1: _____

Committeeman 2: _____

Committeeman 3: _____

Committeeman 4: _____

Committeeman 5: _____

Date of oral defence: _____

浙江大学研究生学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 浙江大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名： 签字日期： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解 浙江大学 有权保留并向国家有关部门或机构送交本论文的复印件和磁盘，允许论文被查阅和借阅。本人授权 浙江大学 可以将学位论文的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

(保密的学位论文在解密后适用本授权书)

学位论文作者签名： 导师签名：

签字日期： 年 月 日 签字日期： 年 月 日

摘要

在数据爆发式增长的时代，人们对数据隐私的重视程度越来越高，不同国家和机构都在出台相关法律法规，以规范数据的管理和使用。如何在保证数据隐私安全的前提下，联合提取数据中的有效价值，是当前学术界和工业界的的最大挑战之一。联邦学习是谷歌在 2016 年首次提出的概念，它允许训练数据分布在参与方本地，在中心服务器的协调下共同训练一个全局模型，实现隐私保护的同时解决数据孤岛问题。常见的联邦学习算法例如 FedAvg，主要是基于中心化的拓扑架构进行论述，中心化联邦学习通常存在拓扑架构不稳定、鲁棒性差、中心节点通信负载高等问题。另外，联邦学习中不同参与方本地数据往往存在非独立同分布的问题，经典的联邦学习算法在数据 Non-IID 的情况下，存在收敛速度慢甚至不收敛、模型精度低等问题。

知识蒸馏是一种高效的模型压缩与知识迁移技术，知识蒸馏结合联邦学习可以在一定程度上解决 Non-IID 带来的问题。本文分析并对比了联邦学习中常见的网络拓扑架构，客户端-服务器架构、对等网络架构下的点对点架构和环形架构，分析了不同架构之间的优劣势。首先，本文在客户端-服务器架构的基础上提出一种基于客户端蒸馏的联邦学习算法——**FedCKD**，在客户端引入知识蒸馏以更好的进行全局共性知识到本地模型的迁移，解决数据非独立同分布带来的问题。然后，本文在环形架构的基础上提出一种基于环形架构的双向蒸馏联邦学习算法——**FedRKD**，在无中心服务器参与的情况下使用双向环形蒸馏进行模型训练，实现去中心化联邦学习以及解决 Non-IID 问题。

本文分别在公共数据集 MNIST、CIFAR10 以及 Non-IID 数据集 VLCS 上对 FedCKD 和 FedRKD 进行了实验测试，对比当前最先进的几类联邦学习算法 FedAvg、FedProx、FedBN 和 MetaFed。在中心化和去中心化架构的基础上，证明了我们的联邦学习算法在非独立同分布的数据上性能表现超越当前的 SOTA 算法。

关键词：联邦学习，知识蒸馏，去中心化，非独立同分布

Abstract

In the era of explosive data growth, people pay more and more attention to data privacy. Different countries and institutions introduce laws and regulations to standardise the use and management of data. One of the biggest challenges for academia and industry today is how to jointly extract valid value from data while ensuring data privacy and security. Federated Learning is a concept first proposed by Google in 2016. It allows training data to be distributed locally among participants, so as to jointly train a global model under the coordination of a central server, thus solving the data island problem and protecting data privacy. Common federated learning algorithms such as FedAvg are mainly discussed based on the centralized topology architecture. However, centralized federated learning usually has some problems, such as unstable topology, poor robustness, and high communication load at the central nodes. In addition, the local data of different participants in federated learning often has the problem of Non-IID. In the case of Non-IID data, the classical federated learning algorithm is slow or even non-convergent in the training phase and has low model accuracy.

Knowledge distillation is an efficient model compression and knowledge transfer technology, thus the Knowledge distillation combined with FL can solve the Non-IID problems. This paper analyzes and compares the common network topology architecture in federated learning, named client/server architecture, point-to-point architecture and ring architecture, and summarizes the advantages and disadvantages between different architectures. Firstly, this paper proposes **a client distillation federated learning algorithm based on client/server architecture**—**FedCKD**. Introducing knowledge distillation on the client side for better transfer of global common knowledge to local models, solving the problem of Non-IID and homogeneous data distribution. Then, the paper proposes **a two-way distillation federated learning algorithm based on the ring architecture**—**FedRKD**, which uses two-way ring distillation for model training without the participation of the central server, enabling decentralized federated learning and solving the Non-IID problems.

We conducted experiments on FedCKD and FedRKD respectively on the public data sets MNIST、CIFAR10 and the Non-IID dataset VLCS. We compared the most advanced FL algorithms like FedAvg, FedProx, FedBN and MetaFed. Based on centralized and decentralized architectures, it is proved that our algorithms outperform the current SOTA algorithms on the Non-IID.

Keywords: Federated Learning, Knowledge Distillation, Decentralized, Non-IID

目录

图目录

表目录

第1章 绪论

1.1 研究背景

随着互联网技术的不断发展，网络覆盖面越来越广，互联网已经成为人们日常生活和工作中不可或缺的一部分，这导致互联网用户规模不断扩大，截止到2022年6月我国互联网用户规模约为10.51亿，人均每周上网时间为29.5个小时。因此互联网每年产生的数据总量越来越大，预计到2025年全球每天将产生463EB的数据。大数据技术能够有效地处理和分析这些数据，为人工智能提供强大的支持。人工智能通过深度学习为主导的技术，可以从大量数据中自动学习和提取有用的信息，并基于这些信息进行预测和决策。因此，人工智能正在以大数据驱动深度学习为主导的新时代，这将为人类社会带来许多积极的变革，过去的十年我们见证了深度学习在人工智能领域的广泛应用，例如自然语言处理、计算机视觉以及推荐系统等领域。

数据作为一种“燃料”，为深度学习提供大量的训练样本，帮助模型提升效果。数据逐渐变成一种资产，服务于企业与组织，并带来巨大的经济效益。随着深度学习落地在各行各业，数据所有权对于企业和用户来说变得越来越重要，企业和用户希望确保自己的数据不会被滥用或泄露，而同时又希望获得对数据的合理使用权。但无论是在深度学习的模型训练、推理和使用阶段，其本身机制都存在着一些数据安全隐患，这些数据安全隐患如果被不法分子有意利用，后果将十分严峻。近年来有许多互联网公司因为泄露用户与系统数据而受到处罚，人们开始更加关注自己的数据信息是否未经许可而被其他商业机构滥用，数据安全和隐私保护已经成为人们日常生活中不可或缺的一部分。

当前各个国家和机构都在出台相应法律法规，以规范数据的使用和管理。2021年开始施行的《个人信息保护法》^[?]是我国第一部专注于保护个人信息的综合性法律，无论是私密还是公开的用户数据，均受到此法保护。欧盟在2018年正式实施《通用数据保护条例》^[?]，是关于个人数据和隐私保护领域重要的法律标杆。瑞士修订的《联邦数据保护法》将于2023年生效，这是瑞士向欧洲通用数据保护条例看齐而修订的一部法律。日本的《个人数据保护法》于2022年

4月正式生效，新法律对旧版做了重大修订，旨在为公民提供个人隐私和数据方面更好的法律保护。

在这样的法律法规环境下，在不同组织机构之间收集和分享数据将受到约束。尤其在金融、医疗、电商等领域，数据的所有者也会极力反对无限制的使用和计算这些数据，这就导致了数据碎片和数据孤岛的形成，使得在不同组织之间进行数据的整合和计算变得非常困难。在遵守现有的法律法规的前提下，通过数据隐私保护技术来保护数据所有者的隐私，同时又满足数据整合与计算的需求，解决数据碎片与数据孤岛的问题，是当前人工智能研究人员面临的重要挑战。

谷歌在2016年提出了联邦学习^[?]，联邦学习旨在通过分布式协同计算训练一个联邦学习模型，在不收集用户本地数据的前提下，达到和传统数据汇集训练几乎一样的效果，其在保护用户数据隐私的同时，解决了数据碎片化和数据孤岛的问题。联邦学习的提出立马引发了学术界和工业界的研究热潮，目前联邦学习正处于高速发展阶段，不断有新的理论知识在丰富和完善，基于联邦学习的应用也越来越广泛，许多新的问题和挑战正在不断产生，因此研究基于联邦学习为基础的隐私保护的算法具有重要的前景和应用价值。

1.2 国内外研究现状

1.2.1 联邦学习

联邦学习是谷歌在2016年发表的论文^[?]中提出了知识蒸馏的概念，它允许训练数据分布在训练参与方本地，在一个中心服务器的协调下共同训练一个全局模型。联邦学习本质上也是分布式机器学习，但在参与方的特性、通信效率、数据特性等方面又与分布式机器学习有所区别^[?]。联邦学习在保证数据隐私安全的同时，又能够达到共同训练模型的目的，其自提出以来就成为人工智能社区里面一个非常活跃的领域，大量相关研究不断发表。

联邦学习作为一种新兴技术，被广泛应用于数据安全与隐私要求比较高的领域，例如智慧医疗、保险金融、智慧城市、智慧政务等。在数据爆发式增长的时代，如何在保证数据隐私的前提下，联合提炼数据中的有效价值，是当前工业界的最大挑战之一。文章^[?]在不共享患者数据的情况下利用联邦学习实现医

疗图像的深度学习建模，解决医疗数据的收集与训练涉及到各种法律、隐私以及数据所有权方面的问题。在自然语言处理领域，谷歌的 Gboard 系统^[?]就利用了联邦学习来提高其预测精度，每一个安卓手机用户都可以在自己的手机上训练一个模型，并将这些模型发送到谷歌中心服务器上进行聚合计算，从而得到一个精度更好的全局模型。这种方法不仅提高了预测精度，还保证了用户隐私的安全。文章^[?]提出了一种基于区块链技术的联邦学习框架，这种框架用来训练全局深度学习模型进行 COVID-19 的 CT 图像预测。各个数据参与方都可以在自己的设备上训练模型，并通过区块链技术来保证数据的安全性和完整性，即使只收集少量的数据，也能够训练出一个准确的全局模型，用来预测 COVID-19 的 CT 图像。

通信效率、数据非独立同分布以及参与方的个性化效果差是联邦学习技术面临的三个主要挑战。联邦学习中不同节点之间频繁通信会对服务器造成较大的压力，常用的优化角度有增加客户端训练轮数、客户端选择、模型压缩等。谷歌的论文提出联邦学习的同时也提出了联邦平均算法 FedAvg^[?]，并在实验层面验证了算法收敛且明显降低通信量，其本质是通过增加客户端计算量的方式以降低通信量。文章^[?]从理论分析的角度证明了联邦平均算法能够收敛且通信量小于随机梯度下降，不过是基于训练数据独立同分布的前提下。数据非独立同分布指的是每个数据参与方的数据分布通常都不相同且互相独立，会对联邦学习模型的训练造成影响。文章^[?]则更进一步，证明 FedAvg 在数据非独立同分布的假设下也能达到收敛且减少通信量，但是在非独立同分布的数据下优化的效果较差且收敛速度慢。FedProx^[?]在 FedAvg 的基础上增加了一个全局模型近似项，是 FedAvg 的一种泛化形式，可以应用于数据非独立同分布场景，不过代价是会降低模型的准确率。文章^[?]提出一种新的联邦优化算法 SCAFFOLD，引入了服务器控制变量和客户端控制变量，这些变量中含有模型更新方向的信息，通过在本地模型的更新公式中添加一个修正项克服梯度差异，解决 FedAvg 在异构数据上表现差的问题。个性化效果差指的是联邦学习模型可能难以适应每个数据参与方的个性化需求。FedBN^[?]通过在客户端本地模型中加入批量归一化层 (BN) 解决联邦学习再数据非独立同分布的情况下特征偏移的问题。另一方面，模型压缩也是解决通信效率的有效方式^[?]，例如通过量化、稀疏化、知识蒸馏等

方式来降低通讯模型的大小，不过数据的压缩也会导致信息的丢失，需要从通信效率和模型压缩之间寻找平衡点。文章^[?]采用一种结构化的模型来更新服务器参数，在每一轮的参数通信过程中减小客户端传递给服务器的模型参数的通信大小，从而减少通信。文章^[?]通过在服务器到客户端发送的全局模型上使用有损压缩且利用联邦 Dropout 的方式允许客户端在全局模型的更小子集上高效地进行本地训练，从而减少了客户端到服务器的通信和本地计算。

安全与隐私是联邦学习技术重点关注的方面。由于联邦学习涉及到多个数据参与方，如何保证数据的安全性和隐私性就成为了一个重要问题。传统的联邦学习通过模型参数传递来进行信息的聚合，文章^[??]提出机器学习模型携带训练数据的信息，梯度和模型参数可以泄露用户隐私。文章^[?]提出可以通过模型逆向攻击的方法，从模型中反推出数据分布的信息，联邦学习中每一轮的迭代都会进行模型参数的传递，则信息的泄露则更加严重。目前联邦学习系统建立在参与方是诚实节点的前提下，当参与方是恶意节点时，数据可能在模型的训练过程中被污染。文章^[?]表明了传统联邦学习的脆弱性，探讨了恶意方实施模型、数据投毒的策略，恶意节点能够在隐藏自身的同时进行模型的投毒攻击，并提出通过验证模型在测试集数据上的准确率、验证模型梯度来检测恶意节点。文章提出 FedTracker^[?]，使用全局水印来验证全局模型的所有权，并将独特的本地指纹嵌入到各个客户端的本地模型，以便于溯源到模型的泄露方。文章^[?]提出联邦模型蒸馏 FedMD 算法，通过未标记公共数据集上来的预测信息来聚合客户端的模型知识。虽然用预测输出取代模型交换很大程度上保护了隐私的泄露，但是预测信息仍然可能导致本地数据的隐私泄露。一种简单的方法是在局部模型的预测中加入随机噪声扰动，不过这需要在隐私保护和性能模型之前取得一个平衡。文章^[?]提出提出了一种 fedMD-NFDP 框架，将无噪声的差分隐私 (NFDP) 机制引入联邦模型蒸馏中，可以保证各方的隐私而不显式地添加任何噪声。文章^[?]提出了一个集成注意力蒸馏框架，通过蒸馏为标记的公共数据集来保护本地数据隐私，框架通过交换客户端的 logits 输出以及注意力映射来大幅降低通信消耗，并引入一种精馏算法，在客户端模型个性化和共识之间达到平衡。

1.2.2 联邦学习与知识蒸馏

Hinton 等人最早在文章^[?]中提出了知识蒸馏的概念，它的核心思想是通过训练一个复杂的教师网络，然后利用教师网络的输出和真实标签来训练一个更小的学生网络。由于学生网络的模型大小和计算复杂度也更小，可以更好地应用到实际场景中。Romero 团队在文章^[?]提出基于特征的蒸馏算法。该项工作主要针对 Hinton 提出的基于目标知识蒸馏的算法进行扩展，允许学生模型不止局限于学习教师模型的目标输出，还可以利用模型的中间特征层进行训练。联邦学习在许多场景中受到越来越多的关注，传统简单的联邦平均算法已经不能满足复杂的现实场景，知识蒸馏作为一种用于模型之间进行知识迁移和压缩的技术，通过引入新的训练过程和模型结构，使得模型的知识可以在其他的模型中重新利用，在保证精度的情况下，让联邦学习模型的训练更加有效。

知识蒸馏通过减少联邦学习中参数或者模型的大小来实现通信成本的压缩，这样做好处是能够减少联邦学习通信传输时所占用的带宽，从而降低通信成本。文章^[?]提出双向蒸馏的方法来进一步地压缩模型传输所占用的带宽。文章^[?]利用基于知识蒸馏的策略，通过利用局部模型上传预测的结果来训练紧凑的全局模型，而不需要上传模型和参数。文章^[?]提出客户端的数据可以通过数据蒸馏的方式，压缩为相对少量的伪样本，并通过传输并使用伪样本训练全局模型，也可以实现通信量的降低。文章^[?]提出了一种自适应的相互蒸馏框架，在每个客户端上相互学习一个学生模型和一个教师模型，其中只有学生模型由不同的客户端共享，并协同更新，并且使用基于正弦值分解的动态梯度逼近方法进一步压缩学生模型，降低通信成本。文章^[?]提出 QuPeD，这种算法通过知识蒸馏聚合不同客户端的知识，并允许客户端使用不同的量化参数和个性化模型进行协作，QuPeD 能够有效地聚合不同客户端的知识，并保证客户端的隐私。文章^[?]提出了 FedH2L 算法，利用相互蒸馏的方式在客户端节点之间传递数据集的预测层来进行训练，让客户端节点之间能够以更加轻量的通信进行协同工作，从而达到更高的训练效果。文章^[?]设立一个公共的未标记数据集，在小规模的数据通信量下，知识蒸馏的方式用局部模型训练全局模型。目前大多数传输预测层来结合知识蒸馏的联邦学习算法中，许多都依赖于代理数据集，文章^[?]提

出一种无数据知识蒸馏方法去联邦学习中存在的异构问题，服务器通过学习一种轻量级生成器，以无数据的方式即成用户信息，然后将其广播给用户，利用知识蒸馏来规范客户端训练。

另一方面，知识蒸馏作为一种更加高效的知识迁移技术，可以在一定程度上解决数据非独立同分布带来的模型训练问题以及在联邦层面实现参与方个性化的问题。文章^[?]设计了一套模型无关的联邦学习框架，利用知识蒸馏的方式解决不同客参与方之间模型和数据异构的问题。文章^[?]使用知识蒸馏将历史的全局模型来指导客户端模型的训练，解决客户端模型训练中存在的偏差问题，并取得了优越的效果。文章^[?]提出了一种基于无监督的双端蒸馏联邦学习框架，该框架在 FedAvg 的基础上使用局部数据蒸馏和基于正则化数据偏移的全局知识蒸馏。文章^[?]开发了聚焦蒸馏和基于注意力的集成技术来平衡提取伪知识，以适应数据的异构场景，隐式地缓解了不同客户端之间的数据非独立同分布问题。文章^[?]提出联邦相互学习，解决数据、对象、模型三个异构问题，这种方式允许客户端协作地训练全局通用模型和定制本地个性化模型，使数据的非独立同分布不再是一个问题，而是服务于本地模型个性化的一种特性。文章^[?]提出 FedFTG，采用一种对抗性蒸馏方法，不断的从客户端模型转移知识，对全局模型进行微调，FedFTG 能够充分利用客户端模型的知识，同时也能够更好地指导全局模型的学习。

1.2.3 去中心化架构下的联邦学习

目前的联邦学习算法通常都是基于中心化的拓扑架构进行开展，这种架构对中心服务器的带宽和网络负载要求非常高，并且容易受到一些因素的影响，如中心服务器宕机、断电或者遭受网络攻击。同时，现实情况中的联邦学习参与方往往是一些网络带宽不高的终端设备，无法满足中心化架构中对于带宽的要求。因此，中心化架构可能不太适用于所有的应用场景，随着对去中心化拓扑结构的联邦学习的关注度的增加，相关的研究也开始逐渐兴起。

在没有中心服务器存在的情况下，串行训练是一种可以用来解决联邦学习问题的方法。这种方法通过将客户端分组进行串行训练，从而减少了对中心服务器的依赖。文章^[?]提出了一种基于动态分组的联邦学习算法 FedFMC。该算

法通过持续更新不同的局部模型和全局模型，来实现分布式训练。文章^[?]提出了一种将客户端分组进行串行训练的联邦学习方法。该方法通过对每个组内的客户端按照编号进行串行训练，并将每组最后一个客户端输出的模型上传至服务端进行聚合，从而提高了联邦学习的效率。然而，这种方法仍然严重依赖于中心节点，并且可能并不适用于所有应用场景。文章^[?]提出了一种基于模型切分的联邦学习方法，它通过使用洪泛协议来传输模型参数，既充分利用了节点之间的带宽，又具有良好的收敛性。这种方法能够提高联邦学习的效率，并且可以适用于多种不同的应用场景。

另外有一些工作，在去中心化拓扑结构中进行点对点通信，文章^[?]提出了 BrainTorrent，这是一个高度动态的点对点通信框架，它通过直接在所有节点之间进行通信，而不依赖中心服务器，来实现动态的点对点通信。然而，这种方式需要极大的通信成本，而且交互的方式比较复杂无序。BrainTorrent 并不是一种无损的方法，并不适用于所有情况。还有一些工作则是从环形拓扑架构开展，例如文章提出 MetaFed^[?]，这是一种分布式机器学习算法，它可以在没有中央服务器的情况下为每个联邦训练个性化模型。这种方法通过将每个联邦视为元节点，并以循环的方式聚合每个联邦的共性知识和个性化知识，从而解决了数据异构和不可靠中央服务器等问题。

1.3 论文研究内容与组织结构

本研究课题为基于知识蒸馏的联邦学习算法研究，旨在不同的联邦拓扑架构下，提出高效的基于知识蒸馏的联邦学习算法，并对算法进行实验验证其有效性，具体的研究内容和主要解决的问题如下：

中心化联邦学习架构的问题。常见的联邦学习算法主要是基于中心化的客户端-服务器拓扑架构(C/S 架构)的前提来进行论述，但中心化联邦学习通常存在网络拓扑架构不稳定，鲁棒性差，中心服务节点通信负载高，易受到内外部攻击等问题。

数据非独立同分布的问题。联邦学习不同客户端之间的数据往往存在非独立同分布问题，经典的联邦学习算法在数据 Non-IID 的条件下，存在收敛速度慢

或者不收敛、模型效果差等问题。经典的联邦学习训练得到的模型往往在参与方上面表现效果差，这与客户端之间数据 Non-IID 有关，与中心服务器直接采用聚合的方式学习所有的知识有关，也与客户端直接替换中心服务下发的全局模型等有关。如何解决数据非独立同分布带来的问题成为了联邦学习研究中的关键之一。

综上所述，我们需要基于以上的问题，提出高效的联邦学习算法，其应该具备这些特点：无需中央服务器的协调，在参与方数据非独立同分布的条件下，联邦学习训练的模型稳定且精度高。

本文的主要结构如下：

第一章，绪论。首先绪论介绍课题的研究背景与意义，指出在数据爆发式增长的年代，各个国家与组织对隐私的规范性要求越来越高，联邦学习作为一种实现数据隐私保护和解决数据孤岛问题的新型技术在深度学习领域具有重要的意义。然后介绍当前联邦学习在国内外的研究现状，分析了联邦学习在隐私保护、联邦通信效率的研究进展与应用，介绍了知识蒸馏结合联邦学习的相关工作以及在去中心化拓扑架构下的联邦学习的解决方案。最后介绍本论文的研究内容、主要解决的问题以及论文各章节的主要内容。

第二章，相关技术与理论介绍。本章节首先介绍了深度学习基本知识与卷积神经网络的概念，以及两种常见的卷积神经网络的模型结构，LeNet 和 AlexNet。再从联邦学习的定义切入，介绍了联邦学习与分布式机器学习的区别，经典的联邦学习算法 FedAvg，以及联邦学习中广泛存在的数据非独立同分布问题。最后本章节介绍了知识蒸馏及其两种算法，基于目标的知识蒸馏和基于特征的知识蒸馏。本章内容是后续算法与实验章节的基础。

第三章，基于知识蒸馏的联邦学习算法。本章节首先分析了基本问题的定义，明确了当前工作存在的问题以及论文算法需要解决的内容。然后介绍了联邦学习中存在的网络拓扑架构，分别是客户端-服务器架构、对等网络架构下的点对点架构和环形架构，对比了几种拓扑架构各自的优劣势。根据不同的拓扑架构，首先，本章在客户端-服务器架构的基础上，提出一种客户端蒸馏的联邦学习算法，在客户端引入知识蒸馏以更好的进行全局知识到本地模型的迁移。其次，本章在环形架构的基础上，提出一种基于环形架构的双向蒸馏联邦学习算

法，在无中心服务器参与的情况下使用双向环形蒸馏进行模型训练，实现去中心化联邦学习，解决数据非独立同分布、节点个性化效果受限等问题。

第四章，基于知识蒸馏的联邦学习算法实验。首先本章节介绍了实验结果的评价标准，准确率和收敛速率的定义。然后介绍了实验环境的配置、对比的不同算法、实验中使用到的模型以及知识蒸馏的设置。其次，本章分别在 VLCS 数据集、MNIST 和 CIFAR10 三种数据集上对 FedCKD 和 FedRKD 进行了实验，对比了 FedAvg、FedProx、FedBN 和 MetaFed 算法，证明了我们算法在非独立同分布数据集上的准确率优于目前 SOTA 的算法。然后，本章节在不同的狄利克雷分布中测试了 FedRKD 和 FedCKD 算法，得出了 FedRKD 算法在不同分布的数据集上表现更加稳定，FedCKD 稳定性也强于其他去中心化算法。最后，本章节还测试了 FedRKD 使用双向环形蒸馏的优势，在算法收敛速率和模型准确率上都远超过使用单向环形蒸馏的方法。

第五章，总结和展望。本章节对基于知识蒸馏的联邦学习算法和实验进行了总结，并对后续的工作进行了展望。

第2章 相关技术与理论介绍

2.1 深度学习

本文章提出的基于联邦学习的知识蒸馏算法和实验均是在深度学习的基础上进行的，因此本小节对深度学习及卷积神经网络做一个简要的介绍。

2.1.1 深度神经网络

深度学习^[?]是一种机器学习技术，通过使用多层深度神经网络，可以让机器学习从大量数据中自动提取特征，并进行相应的预测或决策。深度学习模型可以通过多次迭代进行训练，从而不断提升其表现。深度学习在不同领域的应用中都取得了非常优秀的效果，比如自然语言处理、图像识别、语音识别等。深度神经网络最早的提出可以追溯到 1940 年代对人工神经网络^[?]的研究，2006 年深度神经网络模型在 ImageNet^[?] 图像识别比赛中取得了第一名的好成绩，这一成功标志着深度神经网络已经超越了传统机器学习算法，成为解决实际问题的有效工具，研究人员也在不断探索新的深度神经网络结构和算法，以提高模型的性能和可靠性。

2.1.2 卷积神经网络

卷积神经网络^[?]，简称 CNN，是深度神经中的一种常见网络架构，主要用于图像分类和物体检测等任务。卷积神经网络的架构与深度神经网络类似，由多层神经元组成，但不同于普通的深度神经网络的是，卷积神经网络使用卷积层来提取图像中的特征，并通过池化层来缩小图像的尺寸，以提高计算效率。卷积神经网络根据人类视觉系统处理信息的原理进行设计，可以有效地捕捉图像中的空间和局部特征，并将它们组合成更高层次的特征。通过不断地迭代训练，卷积神经网络能够从大量数据中自动学习到有用的特征，并进行相应的预测或分类。

卷积神经网络通常由多个卷积层、池化层和全连接层组成。卷积层通过使用卷积核（也称为过滤器）来提取图像中的局部特征，并将它们传递给下一层。

池化层通过计算最大值或平均值等方式来降低图像的尺寸，以降低参数量级和防止过拟合。全连接层则将提取的特征组合成更高层次的特征，用于最终的预测或分类。

常见的卷积神经网络有 LeNet、AlexNet、VGGNet、GoogLeNet、ResNet 等，下文将简单介绍之后实验中会使用到的两种 CNN 模型。

2.1.2.1 LeNet

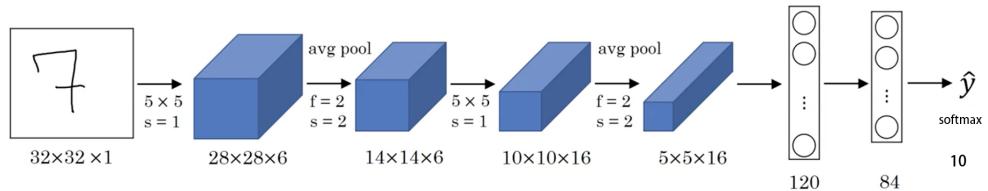


图 2.1 LeNet5-卷积神经网络结构

LeNet^[?] 是一种用于手写数字识别的小型卷积神经网络，如??所示，由 Yann LeCun 提出。它包括两个卷积层、两个池化层和两个全连接层。LeNet 的结构非常简单，但它为深度学习提供了先驱性的工作，能够有效地识别手写数字图像。它的设计思路也成为后来的深度神经网络的基础。

2.1.2.2 AlexNet

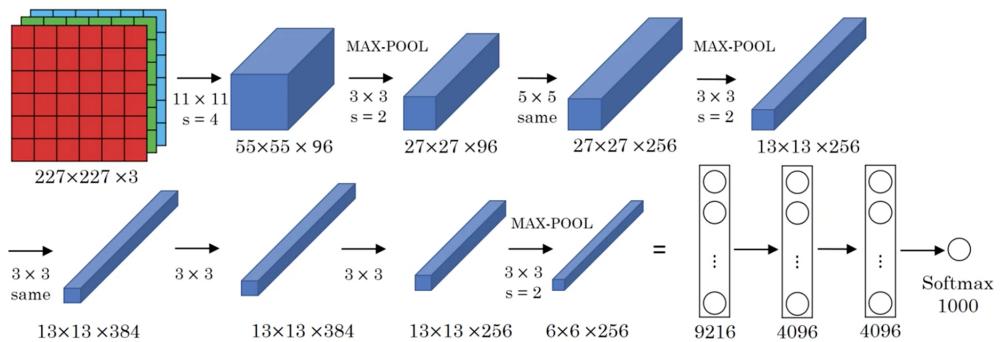


图 2.2 AlexNet-卷积神经网络结构

AlexNet^[?]是一种深度卷积神经网络，如??所示，由 Alex Krizhevsky、Ilya Sutskever 和 Geoffrey Hinton 提出。它包括五个卷积层和三个全连接层，总共包含了 60 万个参数。AlexNet 在 ImageNet 大型图像识别挑战赛中取得了优秀的成绩，并成为深度学习领域的里程碑。它的主要贡献在于使用了 ReLU 激活函数、Dropout 正则化方法和数据扩增技术，从而有效提高了模型的泛化能力。

2.2 联邦学习

2.2.1 联邦学习的定义

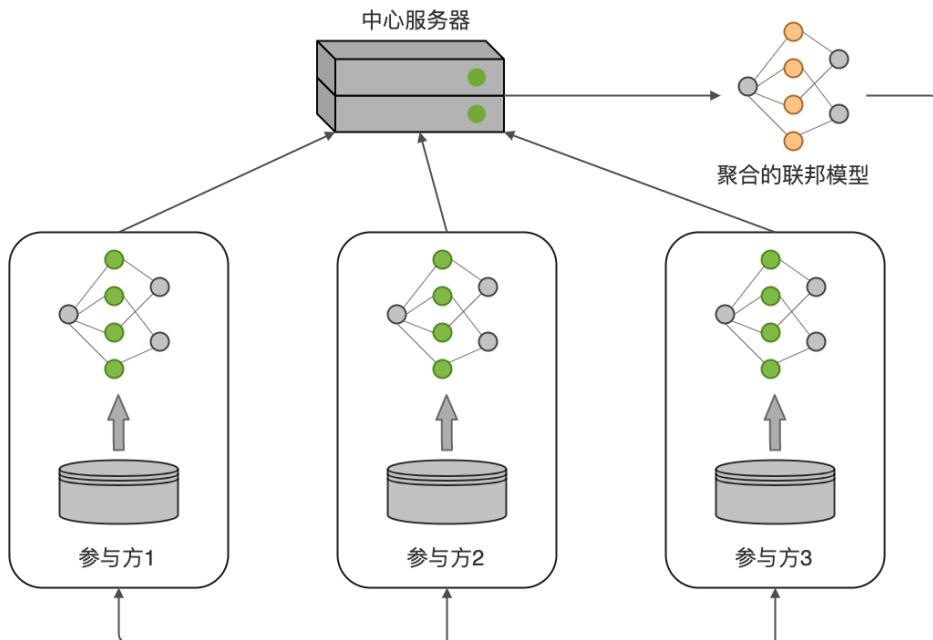


图 2.3 传统联邦学习系统架构——以客户端-服务器架构为例

联邦学习^[?]是谷歌在 2016 年提出的一种技术，旨在各个参与方不互相交换数据的情况下，让多个参与方的数据协同训练一个机器学习模型。如??所示，各个参与方之间会交换模型的参数，并使用这些参数来更新自己的模型。这个过程可以重复进行，直到模型达到满意的精度为止。在模型训练完成之后，参与方可以使用联邦学习得到的全局模型进行推理，即对新数据进行预测或分类。

假设一个联邦学习系统中有 K 个参与方 $\{C_1, C_2, \dots, C_K\}$ ，每个参与方持有

自己的本地数据集 $\{D_1, D_2, \dots, D_K\}$ 。K 个参与方在不共享本地数据集的情况下，协同训练一个全局模型 f_{fed} ，精度为 A_{fed} 。假设传统的机器学习模型是在数据集合并到中心服务器 $D = D_1 \cup D_2 \cup \dots \cup D_K$ 的情况下训练一个模型 f_{sum} ，精度为 A_{sum} ，则我们联邦学习模型具有 δ 的精度损失，如式 (??) 所示。

$$|A_{fed} - A_{sum}| > \delta \quad (2.1)$$

2.2.2 联邦学习与分布式机器学习的区别

表 2.1 联邦学习和分布式机器学习的区别

	联邦学习	分布式机器学习
控制权	用户对数据/设备控制权大	服务器对数据/设备的控制权大
节点状态	设备不稳定且计算性能差异大	设备运行稳定且计算性能一致
通信情况	通信代价远大于计算的代价	通信快速稳定
数据分布	分散存储且固定，数据无法互通可能存在数据的 Non-IID	集中存储不固定可以任意打乱、平衡地分配给所有客户端
节点数量	$1 \sim 10^{10}$	$1 \sim 1000$

联邦学习本质上来说是一种分布式机器学习，但是与分布式机器学习又存在一些区别，如??所示。首先，用户对设备有绝对控制权，用户可以随时控制设备停止参与通信与计算，节点就像是联邦的组成部分，每个联邦都有高度的自治权；而分布式机器学习中，工作节点完全受控于中心服务器。其次，参与联邦学习的节点大多是平板、手机等不稳定设备，不同设备之间的计算性能差异也非常大，节点数量级更大；而分布式机器学习的计算节点大多是在机房中的稳定设备，且计算性能高。最后，参与联邦学习的节点所拥有的数据具有非独立同分布（Non-IID）的特性；而分布式机器学习的数据是随机切片分割到各个工作节点，且是独立同分布的。

2.2.3 联邦学习的分类

文献^[?]根据参与方之间的数据特征空间和数据标签空间的不同分布情况，联邦学习可以分为横向联邦学习、纵向联邦学习。

横向联邦学习指的是参与方之间数据特征空间相同，但标签空间不同的情况。在这种情况下，参与方可以共享相同的特征，但每个参与方都拥有自己的标签，因此可以用来解决不同参与方之间数据标签不同的问题。纵向联邦学习指的是参与方之间数据标签空间相同，但特征空间不同的情况。在这种情况下，参与方可以共享相同的标签，但每个参与方都拥有自己的数据特征，因此可以用来解决不同参与方之间数据特征不同的问题。目前的大多数工作主要以横向联邦学习为主，本文也主要基于横向联邦学习的角度展开论述。

2.2.4 联邦平均算法

联邦平均算法 FedAvg 是文献^[?]提出的，其广泛应用于横向联邦学习当中。假设一共有 K 个客户端，中心服务器初始化模型参数，每轮随机选取至少 1 个至 K 个客户端参与训练，接下来每个被选中的客户端 k 在本地根据服务器下发的本轮（第 t 轮）模型 f_t ，用本地的数据数据 D_k 训练本轮次局部模型 f_{t+1}^k ，并将模型上传至中心服务器。中心服务器将收集来的各客户端的模型，再根据各方样本数量进行加权平均聚合，得到下一轮的模型 f_{t+1} ，执行若干轮次。

算法 2.2.1: 联邦平均算法-服务端

```

    初始化模型  $f_0$ 
    for 每一轮次  $t = 1, 2, \dots$  do
         $m \leftarrow \max(\rho \cdot K, 1)$ 
         $S_t \leftarrow$  (随机选择  $m$  个客户端)
        for 每一个客户端  $k \in S_t$  并行的 do
             $| f_{t+1}^k \leftarrow$  客户端更新  $(k, f_t)$ 
        end
         $f_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} f_{t+1}^k$ 
    end

```

联邦平均算法服务端部分如??所示，其中 f_t^k 代表再客户端 k 上完成第 t 轮

更新后的局部模型， f_t 表示第 t 轮聚合完成的全局模型， ρ 表示每一轮参与计算的客户端的比例， K 代表参与训练的客户端数量， m 代表每一轮次选取的客户端数量， S_t 表示第 t 轮选中的客户端集合， n_k 表示客户端 k 上的样本数量， n 表示所有被选中的客户端的总样本数量。

算法 2.2.2: 联邦平均算法-客户端

```

Input: 客户端  $k$ , 轮次  $t$ , 第  $t$  轮全局模型  $f_t$ 
 $\beta \leftarrow$  (将  $D_k$  分成若干大小为  $B$  的 batch)
for 每个本地执行轮次  $t = 1, 2, \dots, E$  do
    for batch  $b \in \beta$  do
         $| f_t \leftarrow f_t - \eta \bigtriangledown \ell(f_t; b)$ 
    end
end
return 训练完成的  $f_{t+1}^k$  给服务器

```

联邦平均算法客户端部分如??所示。其中 D_k 表示客户端 k 上数据集，其大小为 n_k ； E 为 epochs，表示每个客户端本地数据训练的轮数， B 为 batchsize，表示客户端更新的 batch 大小 ($B = \infty$ 表示 batch 为全部样本，此时就是 full-batch 梯度下降)， β 表示 batch 的集合， \bigtriangledown 为计算梯度， ℓ 为损失函数。

联邦平均算法的本质是通过增加客户端的计算量而减少客户端与服务器之间的通信量。为了增加客户机计算量，可以在中心服务器做聚合（加权平均）操作前在每个客户机上多迭代更新几次，计算量由 ρ 、 B 和 E 三个参数决定。当 $E = 1, B = \infty$ 时，对应的就是 FedSGD，即每一轮客户机一次性将所有本地数据投入训练，更新模型参数。对于一个有着 n_k 个本地样本的客户机 k 来说，每轮的本地更新次数为 $u_k = E \cdot \frac{n_k}{B}$ 。

2.2.5 非独立同分布问题

独立同分布（Independently Identically Distribution, 简称 IID）是指从同一个隐含未知的分布中独立的进行采样，每一组随机采样的变量的概率分布都是相同的，且这些随机变量相互独立。但联邦学习的特性是协同海量的设备及其本地数据进行训练，由于不同场景下的不同设备上数据的分布往往相差较大，即

非同分布的特性；由于受到用户群体和地域等因素的影响，这些设备的数据分布又大多是有关联的，即非独立的特性。因此联邦学习上的不同客户端之间的数据大多满足非独立同分布（Non-IID）的性质。

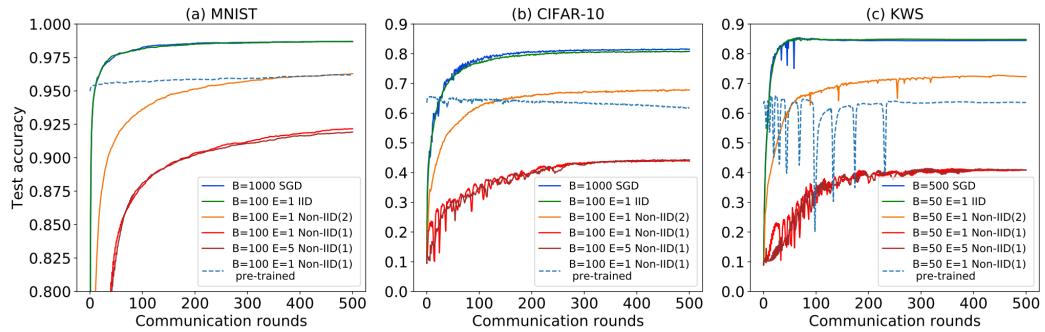


图 2.4 数据非独立同分布条件下 FedAvg 算法性能

文章^[?]对分析了数据 Non-IID 的场景下对联邦学习的影响，并通过实验证明 FedAvg 在 Non-IID 的场景下模型精度低，且存在收敛速度慢或者不收敛等问题，如^{??}所示。因此如何解决数据 Non-IID 的问题成为了联邦学习研究中的关键。

2.3 知识蒸馏

大规模的机器学习和深度学习越来越普遍，训练得到的模型也越来越大，例如 GPT-3 使用 570GB 的文本进行训练，训练所得的模型由 1750 亿个参数组成。虽然训练大型模型有助于获得最优的性能，但是在边缘设备上部署大模型却是不现实的。

知识蒸馏是指将模型的知识从一个大模型或者一组模型转移到一个更小的模型的技术，使得模型可以在性能有限的机器上进行部署，且不会显著的降低模型的性能，本质上说知识蒸馏是一种模型压缩的方法。Geoffrey Hinton^[?]等人在 2015 年发表的论文中将这种从大模型中学习小模型的过程定义为知识蒸馏。

随着深度学习在过去十年的研究与应用，它在语音识别、图像识别和自然语言处理等多个领域有非常突出的应用前景，相关的神经网络模型架构变得越来越复杂，为了在内存和计算能力有限的边缘设备上部署大型的神经网络模型，

知识蒸馏作为一种模型压缩技术同时也被推上了历史的舞台，针对知识蒸馏的大量研究不断发表。

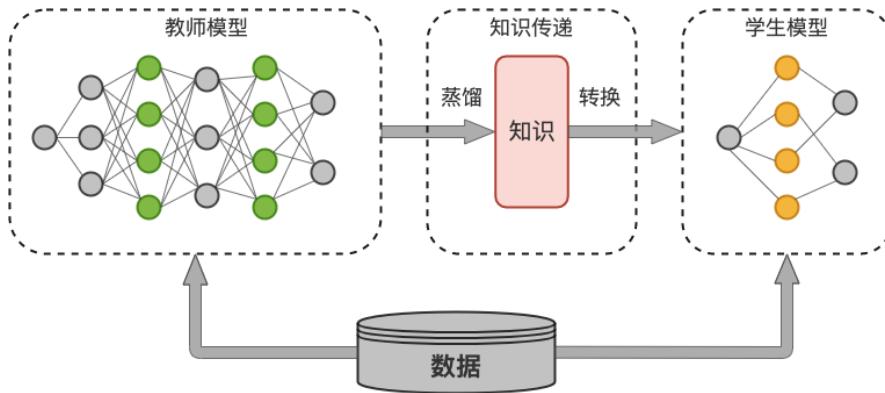


图 2.5 知识蒸馏流程——以教师/学生架构为例

如??所示，在知识蒸馏中，一个结构更小的“学生”模型模仿学习一个结构更大的“老师”模型，并利用“老师”的知识训练获得相似或者更高的精度。在神经网络中，知识通常是指模型的权重和偏置，在一个大型的深度神经网络中，知识的来源具有丰富的多样性。知识蒸馏是对模型的能力进行迁移，典型的知识蒸馏方法使用 logit 作为知识的来源，另外有些方法则关注中间层的权值。根据迁移方法的不一致，可以分为基于目标的蒸馏算法和基于特征的蒸馏算法。

2.3.1 基于目标蒸馏的算法

我们把问题定义限定在分类预测下，分类预测的相同点是模型的输出会有一个 SoftMax 层，SoftMax 的值对应着相应类别出现的概率。在进行蒸馏训练之前，我们已经得到一个泛化能力很强的教师模型，我们可以直接让学生模型去学习教师模型的输出，一种非常直接的迁移方式就是使用 SoftMax 层的输出类别概率作为“软目标”。Hinton 发表的知识蒸馏开山之作^[?]介绍的就是基于目标蒸馏的算法。这种算法主要关注教师模型的最终输出层，利用损失函数计算教师模型的 logit 层和学生模型的 logit 层的蒸馏损失，当这种损失再训练中被最小化时，学生模型就能做出和教师模型一致的预测。

2.3.1.1 硬目标和软目标

传统的深度学习算法的训练过程需要定义一个损失函数，其目标是使预测值尽可能接近真实值，也就是我们训练数据所给的标签。其中真实值也被称作硬目标 (Hard-target)，在原始数据集标注的 one-shot 标签中，正标签为 1，负标签为 0。相对应的软目标 (Soft-target) 则是教师模型的输出层类别概率，每个类别都有相应的概率，其中正标签对应的概率最高。

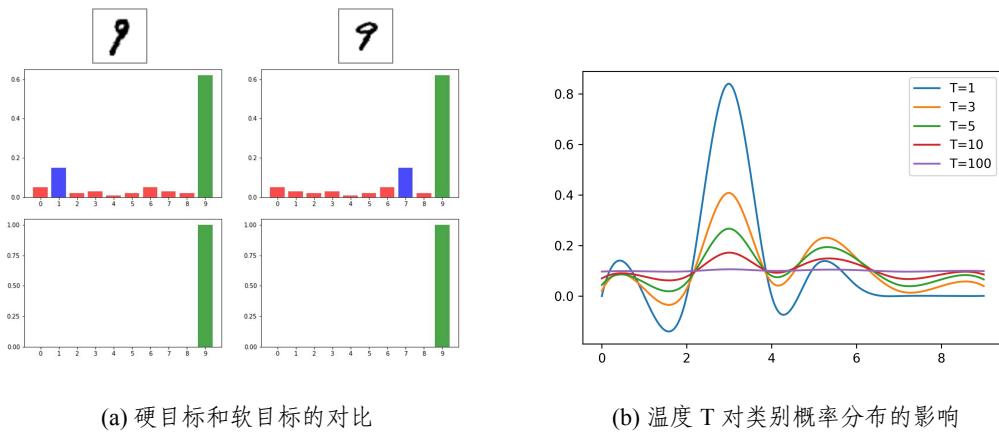


图 2.6 硬目标和软目标的对比/温度 T 对类别概率分布的影响

因为教师模型输出的 softmax 层携带的信息要多于硬目标，老师模型不仅给学生模型提供了正标签的信息，还提供了大量负标签的信息，而这些负标签在硬目标中被当作 0 来统一对待，但负标签其实携带了大量教师模型推演时的信息，比如一些负标签的概率大于另外一些负标签的概率，则表示教师模型在预测时认为测试用例与该负标签有一定的相似性，所以学生模型可以学到更多硬目标以外的知识。

如??所示，例如在手写数据集中做数字识别任务时，假设某个手写体的“9”更加形似于“1”，输出层的输出值中“1”对应的概率会比其他负标签类别的概率高；而另一个手写体“9”则更加形似“7”，则这个输出分配给“7”对应的概率会比其他负标签类别的概率要高。这两个“9”对应的硬目标的分布是相同的，但是它们的软目标的分布却是不同的，由此可见软目标蕴含着比硬目标更多的信息。

同时，使用软目标训练时梯度的方差会更小，可以使用更大的学习率进行训练，所需要的样本也更少。这也说明了通过蒸馏的方法训练出的学生模型相比使用完全相同的训练数据和模型结构且只使用硬目标的训练方法得到的模型，拥有更强的泛化能力。

2.3.1.2 Logits 与温度

Logits. Logits 是一个输入经过神经网络的各种非线性变换，在网络最后的 Softmax 层之前得到的属于各个类别的汇总分值 $[e_1, e_2, \dots, e_k]$ ， e_i 代表第 i 个类别的可能性。因为 e_i 并非最终的概率值，所以一般在得到 Logits 之后会再经过 Softmax 函数进行变换，得到最终的概率值结果 $[p_1, p_2, \dots, p_k]$ ， p_i 代表第 i 个类别的概率值。Softmax 函数会把 Logits 在各类别之间进行概率归一化，使得不同类别满足概率分布，同时它也会放大类别数值之间的差异，使最终的概率得分两极分化，Logits 分值高的类别其概率值会偏大，反之则偏小。使用 Softmax 函数实现 Logits 向概率值转换，原始 Softmax 函数如式 (??) 所示。

$$q_i = \frac{e^i}{\sum_j e^j} \quad (2.2)$$

温度。 但是直接使用 Softmax 的输出作为 soft-target 会有一些额外的问题，当 Softmax 输出的概率分布的熵相对较小时，负类别的 label 几乎等于 0，对 loss 函数的贡献很小，几乎可以忽略不计。所以在原始 Softmax 函数中新增温度 T (temperature)，变更后的 Softmax 函数如式 (??)

$$q_i = \frac{\frac{e^i}{T}}{\sum_j \frac{e^j}{T}} \quad (2.3)$$

当 T=1 时，这个公式就是标准的 Softmax 公式。例如??所示，当温度 T 越高，输出的概率分布越平缓，分布的信息熵越大；当温度 T 越低，输出的概率分布越陡峭，分布信息熵越小。

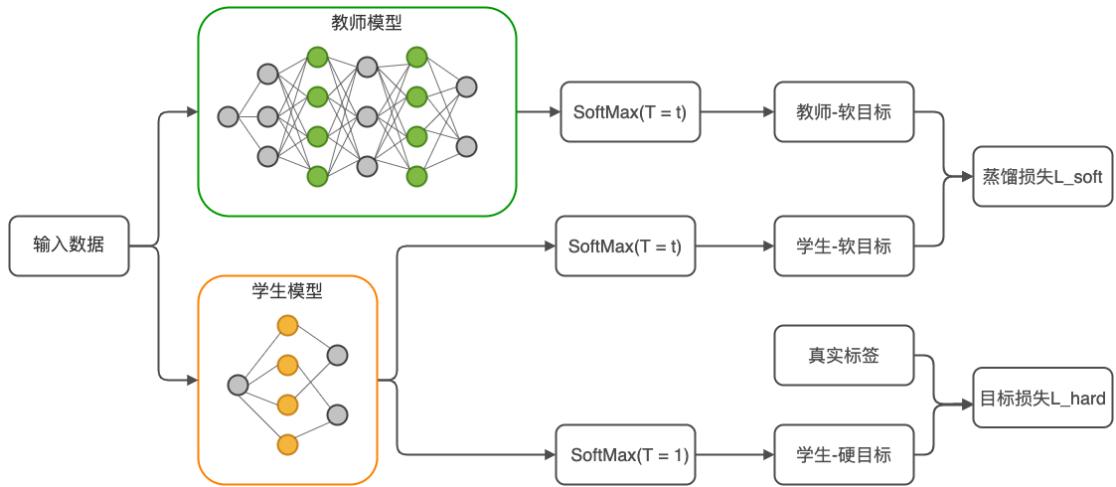


图 2.7 知识蒸馏总体流程图

2.3.1.3 基于目标的知识蒸馏具体算法

基于目标的知识蒸馏总体流程如??所示，当我们训练完教师模型之后，就进入学生模型训练阶段，也就是高温蒸馏阶段。高温蒸馏阶段的目标损失函数由软目标（Soft-target）和硬目标（Hard-target）加权得出，如式 (??) 所示。

$$L = \alpha L_{soft} + \beta L_{hard} \quad (2.4)$$

教师模型和学生模型同时输入训练数据，这里的训练数据可以直接使用训练教师模型的训练数据集。用教师模型在温度 T_{high} 下产生的软目标（Soft-target），与学生模型在相同的温度 T_{high} 下输出的软目标（Soft-target）计算损失，这一部分作为 L_{soft} ，如式 (??) 所示。

$$L_{soft} = - \sum_i^N p_i^T \log(q_i^T) = - \sum_i^N \frac{e^{v_i/T}}{\sum_k^N e^{v_k/T}} \log\left(\frac{e^{z_i/T}}{\sum_k^N e^{z_k/T}}\right) \quad (2.5)$$

其中， p_i^T 是指教师模型在温度等于 T 的条件下 Softmax 输出在第 i 类上的值； q_i^T 是指学生模型在温度等于 T 的条件下 Softmax 输出在第 i 类上的值。 v_i 是指教师模型的 Logits， z_i 是指学生模型的 Logits， N 是指总标签数量。

L_{hard} 为学生模型在 $T = 1$ 的条件下的 Softmax 的输出，其中， c_i 是指在第 i 类上是否为正标签， $c_i \in \{0, 1\}$ ，正标签取 1，负标签取 0。 L_{hard} 公式如式 (??)

所示。

$$L_{hard} = - \sum_i^N c_i \log(q_i^1) = - \sum_i^N c_i \log\left(\frac{e_i^z}{\sum_k^N e_k^z}\right) \quad (2.6)$$

之所以设置 L_{hard} , 是因为教师模型也不是完全正确, 使用 L_{soft} 可以在一定程度上降低教师模型传递错误知识给学生模型的可能性。打个比方, 现实生活中教师的知识虽然远远大于学生, 但也仍然有认知错误的地方, 如果学生在参考教师教授的知识以外, 同时参考标准答案, 可以更加有效地辅助学生学习。

文章^[?]推理得出, α 和 β 是关于 L_{soft} 和 L_{hard} 的权重, 实验发现, 当 L_{hard} 权重较小时, 能产生最好的效果。由于 L_{soft} 贡献的梯度大约为 L_{hard} 的 $\frac{1}{T^2}$, 因此在 L_{soft} 的权重上乘以 T^2 的系数, 这样才能保证 Soft-target 和 Hard-target 贡献的梯度量基本一致。

2.3.2 基于特征蒸馏的算法

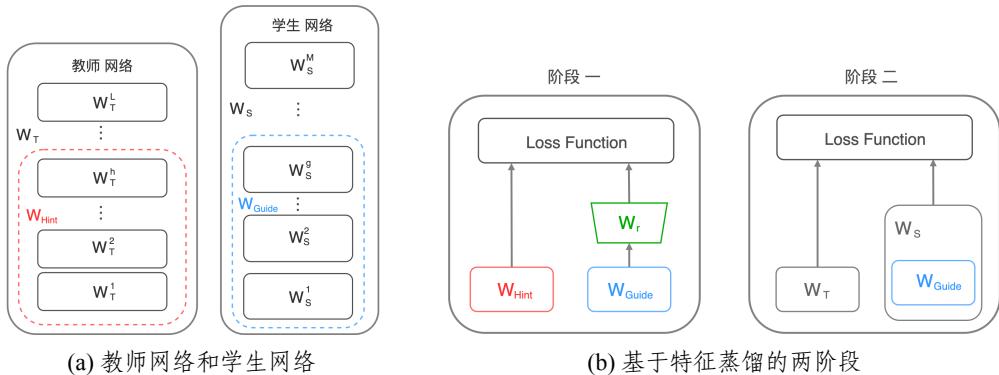


图 2.8 使用特征蒸馏训练学生模型

Romero 团队^[?]提出知识蒸馏的另外一种思路——基于特征的蒸馏算法。该项工作主要针对 Hinton 提出的基于目标知识蒸馏的算法进行扩展, 他们允许学生模型不止局限于学习教师模型的目标输出, 而是从教师网络的中间层提取特征知识, 作为学生网络中间层输出的知识, 在学生网络的中间层连接一个卷积网络做适配, 也就是教师网络的中间层来指导学生网络的训练, 使学生模型的中间特征层逼近教师网络模型对应的中间层。之前的大多数工作都是专注于压

缩教师网络到一个更浅更宽的网络，没有充分利用网络的深度，他们通过利用深度解决模型压缩的问题，提出新方法训练更深更窄的网络 FitNets。

如??所示，选择教师网络的前 h 层作为 W_{Hint} ，从第 1 层到第 h 层的参数对应图中 W_{Hint} 。选择学生网络的前 g 层作为 W_{Guided} ，从第 1 层到第 g 层的参数对应图中的 W_{Guided} ，由 W_{Hint} 指导 W_{Guided} 进行训练。然后进入到以下两个阶段，如??所示。

$$\ell(W_{Guided}, W_r) = \|u_h(x; W_{Hint}) - r(v_g(x; W_{Guided}); W_r)\|_2^2 \quad (2.7)$$

- 阶段一：由于 W_{Hint} 和 W_{Guided} 的特征维度可能不匹配，因此引入卷积层调整器，记做 W_r ，对 W_{Guided} 维度进行调整。然后使用教师网络的 Hint 层预训练学生网络层的 Guided 层，损失函数如式 (??) 所示。其中 u_h 表示教师模型从第 1 层到第 h 层的特征提取器， v_g 表示学生模型从第 1 层到第 g 层的特征提取器， r 表示卷基层调整器。
- 阶段二：由于阶段一的蒸馏没有对应的标签信息，蒸馏粒度不够细，因此他们在阶段二在阶段一的基础上再进行一轮 Hinton 提出的目标蒸馏。

2.4 本章小结

本章综述了深度学习相关知识、深度卷积神经网络的定义、联邦学习相关概念、知识蒸馏相关的概念，作为后面章节的知识铺垫。第一小节主要介绍了深度学习和卷积神经网络的概念，以及两种常见的卷积神经网络的模型结构 LeNet 和 AlexNet。第二小节从联邦学习的定义切入，介绍了联邦学习与分布式机器学习的区别，以及最具有代表性的联邦学习算法 FedAvg，还介绍了联邦学习中广泛存在的数据非独立同分布问题。第三小节介绍了知识蒸馏及其两种实现算法，基于目标的知识蒸馏和基于特征的知识蒸馏，并解释了知识蒸馏中温度、软/硬目标和 Logits 的概念。

第3章 基于知识蒸馏的联邦学习算法

3.1 问题定义

在一个联邦学习的系统中，通常存在 K 个参与方 $\{C_1, C_2, \dots, C_K\}$ ，他们拥有各自的数据，表示为 $\{D_1, D_2, \dots, D_K\}$ ，这些数据可能有着不同的分布（不同参与方之间的数据非独立同分布）。我们的问题限定于所有参与方输入数据空间和输出数据空间完全一致的情况下，即 $\mathcal{X}_i = \mathcal{X}_j$, $\mathcal{Y}_i = \mathcal{Y}_j$, 其中 $i \neq j$ 。其中，每个参与方的数据集 $D_k = \{x_{k,j}, y_{k,j}\}_{j=1}^{n_k}$ 被拆分成三部分，分别为训练数据集 $D_k^{train} = \{x_{k,j}^{train}, y_{k,j}^{train}\}_{j=1}^{n_k^{train}}$ ，验证数据集 $D_k^{valid} = \{x_{k,j}^{valid}, y_{k,j}^{valid}\}_{j=1}^{n_k^{valid}}$ 和测试数据集 $D_k^{test} = \{x_{k,j}^{test}, y_{k,j}^{test}\}_{j=1}^{n_k^{test}}$ ，其中参与方 k 的数据集等于训练集、验证集和测试集的并集 $D_k = D_k^{train} \cup D_k^{valid} \cup D_k^{test}$ ，参与方的总数据量等于训练集、验证集和测试集的数据量相加 $n_k = n_k^{train} + n_k^{valid} + n_k^{test}$ 。

$$\ell_k^{total} = \min \frac{1}{K} \sum_{k=1}^K \frac{1}{n_k^{test}} \sum_{j=1}^{n_k^{test}} \ell(f_k(x_{k,j}^{test}, y_{k,j}^{test})) \quad (3.1)$$

我们的目标是在参与方之间不进行数据传输与交换的情况下（即数据集保存于参与方本地），在每一个参与方 k 上根据本地数据 D_k 联合训练一个最优的联邦学习模型 f_k ，使得模型在所有的测试数据集上的总损失 ℓ_k^{total} 达到最小，如式 (??) 所示。

FedAvg 算法用模型通信代替直接的数据交换以保护数据的隐私性，尽管 **FedAvg** 在许多应用中取得了令人满意的性能，但在联邦参与方数据非独立同分布场景下，性能却不尽如人意。**FedProx**、**FedBN** 等算法通过在客户端模型中添加损失函数项和批量归一化层等方式，允许局部模型和全局模型之间存在差异，一定程度上缓解了数据非独立同分布带来的影响。以上三种算法都是在客户端-服务器这种网络拓扑架构进行通信与计算，但在对隐私保护要求更高、对联邦训练稳定性要求更高的场景下，中心化联邦学习通常存在网络拓扑架构不稳定，中心服务节点通信负载高，易受到内外部攻击等问题。**MetaFed** 是一个环形联邦学习算法，通过自适应环形知识蒸馏的方式，在无中心服务器参与的情况下进行

模型训练方法，但经过实验发现 MetaFed 在非独立同分布的情况下性能表现不稳定，且存在个性化表现差等问题。

常见的联邦学习算法主要是基于中心化的客户端-服务器拓扑架构的前提来进行论述，但中心化联邦学习通常存在网络拓扑架构不稳定，鲁棒性差，中心服务节点通信负载高，易受到内外部攻击等问题。联邦学习不同客户端之间的数据往往存在非独立同分布问题，经典的联邦学习算法在数据 Non-IID 的条件下，存在收敛速度慢或者不收敛、模型效果差等问题，如何解决数据非独立同分布带来的问题成为了联邦学习研究中的关键之一。传统的联邦学习个性化效果受限，模型往往在参与方上面表现效果差，这与客户端之间数据 Non-IID 有关，与中心服务器直接采用聚合的方式学习所有的知识有关，也与客户端直接替换中心服务下发的全局模型等有关，因此解决参与方本地个性化效果差也是联邦学习研究中的关键之一。

综上所述，我们理想的联邦学习算法应该具备如下特点：无需中央服务器的协调，在数据非独立同分布下联邦学习模型评价指标达到优秀。本章节主要内容如下：

- 介绍了联邦学习中存在的网络拓扑架构，客户端-服务器架构、对等网络架构下的点对点架构和环形架构，分析并对比了几种拓扑架构优劣势。
- 在客户端-服务器架构的基础上提出一种基于 C/S 架构的 C 端蒸馏联邦学习算法，在客户端引入知识蒸馏以更好的进行全局知识到本地模型的迁移。
- 在环形架构的基础上提出一种基于环形架构的双向环形蒸馏联邦学习算法，在无中心服务器参与的情况下使用双向环形蒸馏进行模型训练，解决数据非独立同分布导致的问题。本小节对双向交替的环形蒸馏进行了详细的分析，解释了使用双向交替的环形蒸馏的原因。

3.2 联邦学习拓扑结构

联邦学习中存在两种最基本的网络拓扑架构，分别为客户端-服务器架构和对等网络架构。其中，对等网络架构根据传输模式的不同，又可以分为点对点架构和环形架构。

3.2.1 客户端-服务器架构

客户端-服务器架构也称为 C/S 架构或者主从架构，如??所示。在这种架构中，具有 K 个相同结构的客户端，在服务器的协调下共同训练一个全局模型。中心服务器将聚合完成的全局模型下发到各客户端，客户端用全局模型替换本地模型并进行训练，并将训练完成的模型上传到服务器，整个过程持续迭代，直至收敛。

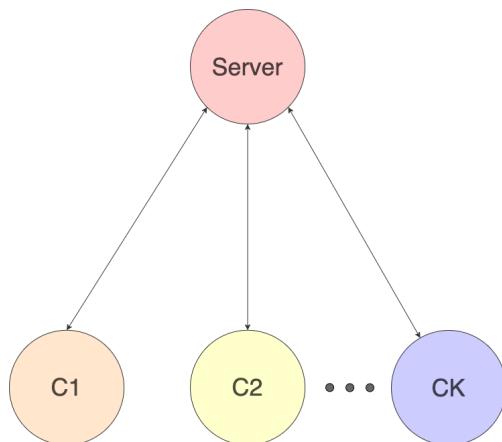


图 3.1 客户端-服务器 (C/S) 架构图

首先，C/S 架构中存在单点故障风险，一旦中心服务器节点被控制，将可能导致系统有遭受攻击的风险。其次，所有客户端与服务器进行并行通信，容易导致中心节点存在通信负载高的问题。因此需要探索一种更加适合联邦学习的架构，有效的提高系统的鲁棒性和保护本地数据，以更好的应对越来越严格的用户隐私和数据安全的监管环境。

3.2.2 对等网络架构

除了客户端-服务器架构以外，还存在一种对等网络架构。在该架构下，不存在中心服务器或者协调方，架构中的 K 个参与方被称作分布式参与方。每一个参与方只负责使用本地数据训练一个机器学习模型，训练方们相互传输模型并进行知识传递。与客户端-服务器架构相比，对等网络架构的明显优势是去中心化，但由于没有中央服务器的协调，每个参与方可能会共同承担更多的通信

消耗。由于中心服务器的缺失，参与方们必须提前协定好发送和接收模型信息的次序，根据参与方传输模式的不同又可以分为点对点架构和环形架构。

点对点架构 点对点架构也称为 P2P (Peer-to-Peer) 架构，如??所示。第 i 个参与方将自身的模型参数发送给参与方 $\{1, 2, \dots, K\} \setminus \{i\}$ 。当第 j 个参与方收到来自第 i 个参与方的模型参数后，它将使用参与方 i 的模型学习其知识，同理第 j 个参与方会将自己的模型发送给参与方 $\{1, 2, \dots, K\} \setminus \{j\}$ 进行学习。这个过程不断重复，直到模型参数收敛或者达到允许的最大训练时间。

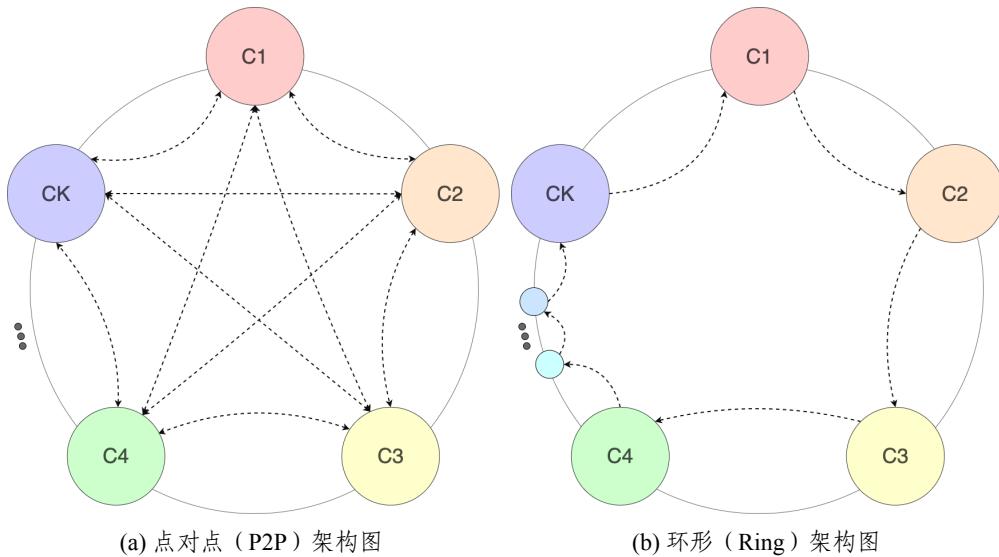


图 3.2 两种对等网络架构图

环形架构 另外一种对等网络架构，称为环形 (Ring) 架构，如??所示。该架构下，参与方们会被组织成一条环形链条，第 1 个参与方（链首）将本地模型发送给它的下游参与方，下游参与方接收来自上游参与方的模型后，使用本地数据集进行模型训练，再将训练后的模型传递给下一个参与方。例如参与方 1 到参与方 2，参与方 2 到参与方 3，参与方 (K-1) 到参与方 K，然后参与方 K 再回到参与方 1，整个过程不断重复，直到模型参数收敛或者达到最大的训练轮次。本质上环形架构可以看作是点对点架构的一种特殊形式。

3.2.3 几种拓扑架构对比

表 3.1 C/S 架构、P2P 架构、环形架构的对比

对比指标	C/S 架构	P2P 架构	环形架构
去中心化	否	是	是
是否单点故障	是	否	否
拓扑复杂度	低	高	中
参与方通信量	小	大	中
服务器通信量	大	/	/

C/S 架构是最简单的一种架构，管理和维护都相对容易，但存在单点故障问题，一旦中心节点出故障，则会导致整个系统不可用，另外所有参与方与服务器进行通信，容易导致中心节点存在通信负载高的问题，通信线路利用率不高。

对等网络架构的明显优势是取消了中央服务器，使得整个拓扑架构更加稳定，不容易受到攻击，也不存在单点故障问题。但 P2P 架构中每个参与方之间存在着模型传输，模型拓扑结构复杂，将导致更大的通信量。环形架构介于 P2P 架构和 C/S 架构之间，在实现去中心化的同时，将拓扑复杂度和通信量维持在一个较低的水平。

基于??的对比情况，本章在 C/S 架构和环形架构的基础上，提出基于知识蒸馏的联邦学习算法。

3.3 基于 C/S 架构的客户端蒸馏联邦学习算法

传统的联邦学习算法例如联邦平均算法（FedAvg），在中心服务器进行模型的聚合，并下发全局模型到各个客户端，参与方直接使用全局模型替代本地模型，这种传统联邦学习算法在数据集非独立同分布条件下，存在模型精度大幅下降，模型收敛慢甚至不收敛等问题。FedProx 算法相对于 FedAvg 来说，在客户端训练过程中增加了损失函数的修正项，使得模型收敛更快效果更好；FedBN

算法通过在客户端局部模型中加入了批量归一化层（BN）解决模型精度低的问题并实现一定程度的模型个性化。总结来说，目前绝大多数算法在客户端获取中心服务器的全局模型之后，几乎都使用全局模型直接替代局部模型或进行微小的调整，这种方式在一定程度上不利于本地模型个性化，且会引起全局模型收敛慢、性能差的问题。

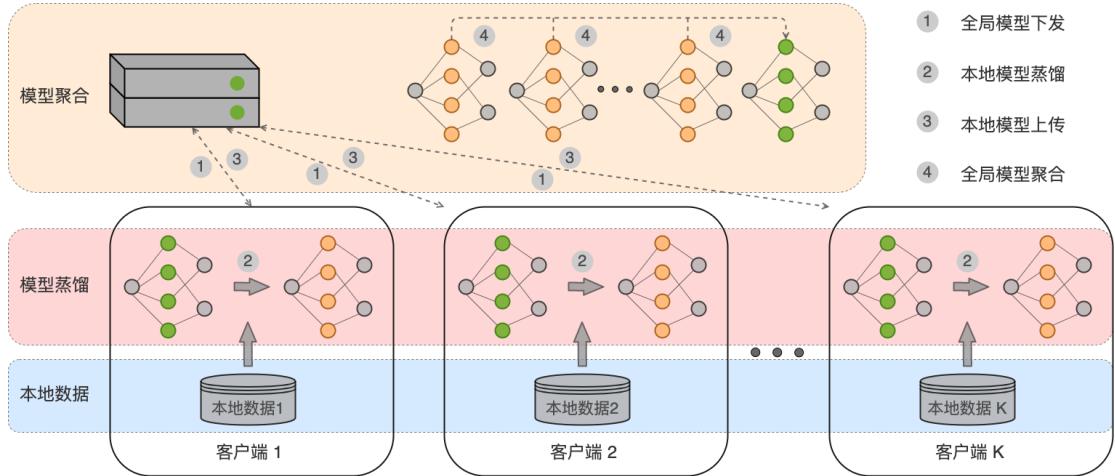


图 3.3 基于 C/S 架构的客户端蒸馏联邦学习算法架构图

知识蒸馏是一种模型无关的知识迁移方式，能有效的在不同模型之间进行知识的迁移，知识蒸馏结合联邦学习可以在一定程度上解决 Non-IID 带来的问题。本小节中我们提出一种基于客户端蒸馏的联邦学习算法——FedCKD，如??所示，在客户端-服务器架构中的客户端部分加入知识蒸馏算法。客户端持有本地数据且互相之间不进行共享，由中心服务器进行模型聚合并下发给客户端，客户端进行模型蒸馏来获取知识。其中橙色模型表示经过客户端模型蒸馏后的模型，绿色模型表示经过中心服务器模型聚合后的模型。

算法按照架构主要划分为两端，FedCKD 服务端部分和 FedCKD 客户端部分，FedCKD 算法的具体描述详见??和??，在每个训练轮次 t 中都会执行以下 4 个步骤：

- 全局模型下发，中心服务器将本轮次聚合完成的全局模型下发给所有客户端，如果是首轮则初始化全局模型并下发。
- 本地模型蒸馏，客户端将本地模型作为学生模型，将中心服务器下发的全

- 局模型作为教师模型，使用客户端本地数据集，进行特征蒸馏。
- 本地模型上传，客户端将蒸馏后的模型上传到中心服务端。
 - 全局模型聚合，服务器接收所有客户端蒸馏后的模型，进行加权平均聚合得到新一轮全局模型。

算法 3.3.1: FedCKD-Server

初始化模型 f_1 ，并广播给所有客户端。

```

for 每一全局模型更新轮次  $t = 1, 2, \dots$  do
    for 每一客户端  $k \in 1, 2, \dots, K$  并行地 do
         $f_{t+1}^k \leftarrow \text{FedCKD-Client}(k, f_t)$ 。
        服务器接收客户端模型参数  $f_{t+1}^k$ 。
    end
    服务器将所有客户端模型加权平均:  $f_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k^{train}}{n^{train}} f_{t+1}^k$ 。
    服务器将聚合后的模型参数  $f_{t+1}$  广播给所有客户端。
end

```

在 FedCKD 服务端部分，算法思想 FedAvg 算法的服务端部分一致。初始化模型 f_1 ，并将模型 f_1 广播给所有的客户端（一共 K 个客户端）。在全局模型更新轮次 t 中，接收每个客户端蒸馏完成的模型 f_{t+1}^k ，并对所有客户端模型进行聚合，即对收到的模型参数使用加权平均： $\sum_{k=1}^K \frac{n_k^{train}}{n^{train}} f_{t+1}^k$ ，得到 $t+1$ 轮次的全局模型 f_{t+1} ，其中 n_k^{train} 表示第 k 个客户端上训练数据的数量， n^{train} 表示 K 个客户端训练数据的总和，最后中心服务器将聚合后的全局模型参数 f_{t+1} 广播给所有客户端，本轮次结束。

在 FedCKD 客户端部分，在客户端模型更新轮次 t 中，客户端 k 接收服务器最新的全局模型 f_t ，计算 f_t^k 在客户端 k 的本地验证数据集 D_k^{valid} 上的精度 acc_t^{valid} 。如果精度 acc_t^{valid} 大于参数 μ_0 ，则表明局部模型 f_t^k 在验证数据集上表现足够好，我们使用特征蒸馏的方式把对客户端 k 有用的知识保留下来，并舍弃对客户端 k 没用的知识。反之，则表明局部模型 f_t^k 还缺少很多全局的知识，则使用全局模型 f_t 替代局部模型 f_t^k ，并使用本地数据集 D_k^{train} 对本地模型 f_t^k 进行训练。最终将训练完成的模型 f_{t+1}^k 发送至服务器，本轮次结束。

$$\ell_{total} = \ell_{class}(f_k; (x_k^{train}, y_k^{train})) + \ell_{dk}(g_{total}; g_{local}; x_k^{train}) \quad (3.2)$$

算法 3.3.2: FedCKD-Client。

Input: 客户端 k, 模型 f_t
 接收服务器第 t 轮全局模型 f_t 。
 计算本地模型 f_t^k 在验证数据集 D_k^{valid} 上的精度 acc_t^{valid} 。
if $acc_t^{valid} > \mu_0$ **then**
 | 使用本地训练数据集 D_k^{train} , 使 f_t 蒸馏 f_t^k , 损失函数为式 (??)。
end
else
 | $f_t^k = f_t$ 。
 | 使用本地训练数据集 D_k^{train} , 训练本地模型 f_t^k , 损失函数为式 (??)。
end
 将训练后的模型 f_{i+1}^k 发送至服务器。

客户端算法中, 蒸馏损失函数如式 (??) 所示, ℓ_{total} 表示总的蒸馏损失, 由 ℓ_{class} 和 ℓ_{dk} 两部分构成。

$$\ell_{class}(f_k; (x_k^{train}, y_k^{train})) = \frac{1}{n_k^{train}} \sum_{j=1}^{n_k^{train}} \ell(f_k(x_{k,j}^{train}, y_{k,j}^{train})) \quad (3.3)$$

ℓ_{class} 表示给定模型以训练数据集的数据和标签计算得到的分类损失, 如式 (??) 所示。 ℓ 表示交叉熵损失函数, f_k 表示客户端 k 的本地模型, x_i^{train} 和 y_i^{train} 分别表示客户端 k 本地的训练数据和标签, n_k^{train} 表示客户端 k 上的训练数据集总数量。

$$\ell_{dk}(g_{total}; g_{local}; x_k^{train}) = \|g_{total}(x_k^{train}) - g_{local}(x_k^{train})\|_2^2 \quad (3.4)$$

ℓ_{dk} 表示特征蒸馏部分的交叉熵损失函数, 如式 (??) 所示。 g 表示特征提取器, g_{total} 表示对全局模型进行特征提取的结果, g_{local} 表示对客户端本地进行特征提取的结果。

当客户端局部模型在本地验证数据集上表现不佳的情况下 (即 $acc_t^{valid} \leq \mu_0$ 时), 使用全局模型直接替换本地模型, 并进行一轮本地模型的训练, 此时 FedCKD 算法退化为 FedAvg 算法; 当局部模型在本地验证数据集表现良好的情况下 (即 $acc_t^{valid} > \mu_0$ 时), 则表明当前局部模型掌握了足够多的本地与全局的知识, 客户端局部模型使用特征蒸馏从全局模型学习到对本客户端有用的知识,

而不是使用全局模型直接替换局部模型，这种方式更加有利于知识在全局模型与局部模型之间的迁移，有利于算法在非独立同分布数据集上更快达到收敛，且性能表现也会更好。

3.4 基于环形架构的双向蒸馏联邦学习算法

3.4.1 算法定义

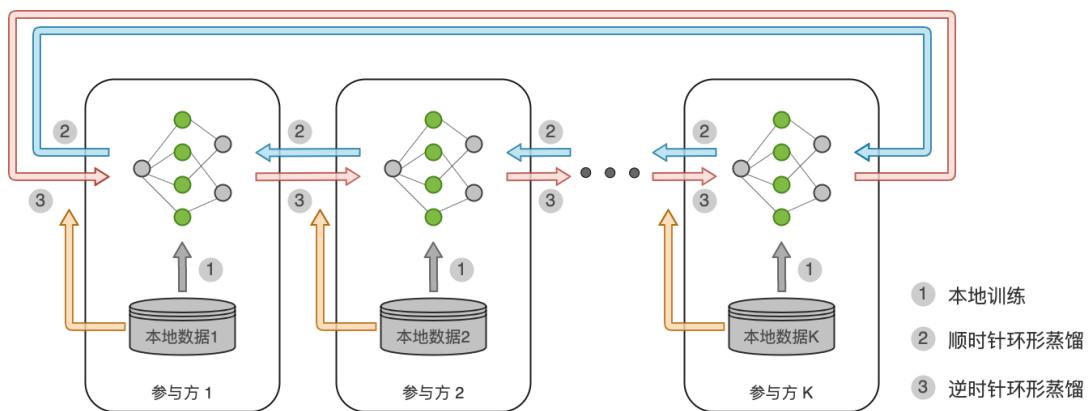


图 3.4 基于环形架构的双向蒸馏联邦学习算法

环形拓扑架构结合知识蒸馏算法，可以在没有中心服务器参与协调的情况下，解决参与方数据非独立同分布带来的影响。本小节中，我们提出一种基于环形架构的双向蒸馏联邦学习算法——FedRKD。如??所示，其中灰色箭头表示本地模型训练，淡蓝色箭头与橙色箭头共同表示顺时针模型传输与蒸馏，粉红色箭头与橙色箭头共同表示逆时针模型传输与蒸馏。FedRKD 算法的主要流程为以下几个步骤：

1. 本地训练，每个参与方使用本地训练数据集并行进行训练。
2. 顺时针环形蒸馏，每个参与方并行地执行，各参与方之间进行模型顺时针环形传输，并通过知识蒸馏的方式进行自适应知识传递。
3. 逆时针环形蒸馏，每个参与方并行地执行，各参与方之间进行模型逆时针环形传输，并通过知识蒸馏的方式进行自适应知识传递。
4. 循环执行第 2、3 步骤，直至迭代次数达到上限。

环形蒸馏部分主要包括模型传输与模型知识蒸馏。

模型传输阶段, 假设联邦中存在 K 个参与方, 则模型传输总共执行 K-1 轮。在每个模型传输轮次中, 每个参与方 j 并行的将本地模型传递给下一个参与方 i, 如果是顺时针环形蒸馏, 则 $i = (j + 1) \% K$; 如果是逆时针环形蒸馏, 则 $i = (j + K - 1) \% K$ 。参与方 i 接收到参与方 j 的模型, 各参与方并行执行知识蒸馏, 这里我们使用基于特征的知识蒸馏。

模型知识蒸馏阶段, 我们设计了一种自适应的基于特征蒸馏的算法, 蒸馏损失如式 (??) 所示。其中 ℓ_{total} 表示总的蒸馏损失, ℓ_{class} 表示分类交叉熵损失, ℓ_{dk} 表示特征蒸馏部分的交叉熵损失。 g 表示特征提取器, g_j 表示对参与方模型 f_j 进行特征提取的结果, g_i 表示对参与方模型 f_i 进行特征提取的结果。 x_i^{train} 和 y_i^{train} 分别表示参与方 i 本地的训练数据和标签。 λ 是权衡全局知识转移和本地数据特征的参数。

$$\ell_{total} = \ell_{class}(f_i; (x_i^{train}, y_i^{train})) + \lambda \ell_{dk}(g_j; g_i; x_i^{train}) \quad (3.5)$$

当参与方 j 的模型在参与方 i 的验证数据集上精度表现不好时, 则蒸馏的权重会被设定为 0 表明此模型不具备参考性, 即 $\lambda = 0$; 当参与方 j 的模型在参与方 i 的验证数据集上精度表现较好时, 则我们自适应地调整 λ 的值来决定全局知识的保留量, 参与方 j 的模型精度比参与方 i 的本地模型精度好的越多, 则其蒸馏权重占比越大, 即对应的 λ 也越大, 具体的参数 λ 设置如式 (??) 所示。

$$\lambda = \lambda_0 \times 10^{\min(1, (acc_j^{valid} - acc_i^{valid}) * 10) - 1} \quad (3.6)$$

算法的具体描述, 如??所示, 在模型本地训练阶段, 每个参与方 k 使用本地数据集 D_k^{train} 进行本地模型训练, 得到模型 f_1^k 。设置 $flag$ 为 $True$, $flag$ 的作用为标志模型传递的方向, $True$ 代表顺时针模型传递, $False$ 代表逆时针模型传递, 设置 λ 为初始值 λ_0 。

环形蒸馏阶段, 每个轮次 t 进行一轮环形蒸馏, 本轮次环形蒸馏的方向由 $flag$ 确定。本轮次环形蒸馏中, 一共会进行 K-1 (K 为联邦中参与方的个数) 次并行的模型传输与蒸馏。对于参与方 j 来说, 根据 $flag$ 来确定模型传递的方向, 得到下

算法 3.4.1: FedRKD

```

for 每一个参与方  $k = 1, 2, \dots, K$  并行的 do
    | 使用本地训练数据集  $D_k^{train}$ , 训练本地模型  $f_1^k$ 。
end
设  $flag \leftarrow True$ 。
设  $\lambda \leftarrow \lambda_0$ 。
for 每一轮次  $t = 1, 2, \dots$  do
    for 每一轮次  $k = 1, 2, \dots, K - 1$  do
        for 每一个参与方  $i = 1, 2, \dots, K$  并行的 do
            if  $flag$  then  $i \leftarrow (j + K - 1) \% K$ 
            else  $i \leftarrow (j + 1) \% K$ 
            参与方  $i$  接收参与方  $j$  的模型  $f_t^j$ 
            计算  $f_t^j$  在验证数据集  $D_i^{valid}$  上的精度  $acc_t^{j,valid}$ 。
            计算  $f_t^i$  在验证数据集  $D_i^{valid}$  上的精度  $acc_t^{i,valid}$ 。
            if  $acc_t^{j,valid} < acc_t^{i,valid}$  then
                | 设置  $\lambda \leftarrow 0$ 。
            end
            else
                | 设置  $\lambda$  为式 (??)。
            end
            使用  $f_t^j$  蒸馏  $f_t^i$ , 数据集为  $D_k^{train}$ , 损失函数为式 (??)。
            训练完毕, 得到模型  $f_{t+1}^k$ , 进入下一轮次。
        end
    end
     $flag \leftarrow !flag$ 
end

```

一个模型传输目标参与方 i , 参与方 i 接收参与方 j 的模型 f_t^j , 计算模型 f_t^j 和模型 f_t^i 在验证数据集 D_i^{valid} 上的精度 $acc_t^{j,valid}$ 和 $acc_t^{i,valid}$ 。如果 $acc_t^{j,valid} < acc_t^{i,valid}$, 则设置 λ 为 0, 否则设置 λ 为式 (??), 使用 f_t^j 蒸馏 f_t^i , 数据集为 D_i^{train} , 损失函数为式 (??), 其中 λ 为动态自适应。

本轮训练完毕, 得到模型 f_{t+1}^k , 并将 $flag$ 至反, 进入到下一轮次的训练, 直到所有训练轮次执行完毕, 程序结束。

3.4.2 双向知识蒸馏

知识蒸馏在不断串行化知识传递过程中，会有一定程度上的知识的丢失，我们称这种现象为知识蒸馏损耗。我们使用 CIFAR10 和 MNIST 数据集训练 0 号 LeNet-5 模型，然后采用知识蒸馏的方式依次对编号 1-100 的 LeNet5 模型进行训练，设置编号为 i 的模型作为教师模型，编号为 $m+1$ 的模型作为学生模型。在整个串行知识蒸馏的过程中，模型的表现如??所示，可以发现在不断进行串行知识蒸馏的过程中，模型的精度会有所下降。通俗来说，“老师”的知识是从“老师的老师”那里学习到，“老师”在学习的过程中并不一定能把所有的知识都学会，同时“老师”身上还包含了从“课本”（即训练数据）中学到的知识，当“老师”再把自己的知识教给“学生”时，同理“学生”也不能学习到“老师”的所有知识，这就导致部分知识在迁移的过程中会逐渐“失传”。

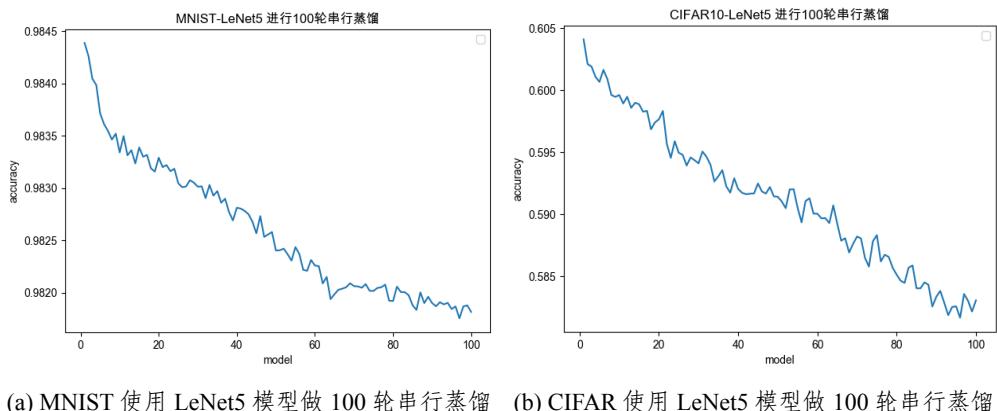


图 3.5 LeNet5 串行知识蒸馏 100 轮的效果

对于有 K 个参与方的环形联邦架构来说，如果只进行单向模型传递与蒸馏，以顺时针环形蒸馏为例，参与方 $k+1$ 的知识传递到参与方 k 需要经过 $K-2$ 个参与方，而参与方 $k-1$ 则不需要经过任何参与方。当 K 的数量庞大的时候，由于知识在蒸馏过程中的损耗，这个过程中会导致参与方 k 的本地模型学习到更多近端（参与方 $k-1$ ）的知识，学习到更少远端（参与方 $k+1$ ）的知识，对于逆时针环形蒸馏也同理。

另外，如果只进行单项模型传递蒸馏，以顺时针环形蒸馏为例，参与方 k 一接收参与方 $k-1$ 的模型。环形蒸馏中总共执行 $K-1$ 次的模型传递与蒸馏，参与

方 $k-1$ 的知识在整个过程中进行 $K-1$ 次蒸馏，不断被强化；而参与方 $k+1$ 的知识对于参与方 k 来说则只进行 1 次蒸馏，这样远端的参与方 $k+1$ 对于参与方 k 的知识传递势必若干于近端参与方 $k-1$ ，对于逆时针环形蒸馏也同理。

因此，远端与近端的参与方在环形蒸馏过程中存在知识传递倾斜的问题，近端的参与方在知识传递的次数和效果上均要好于远端的参与方。

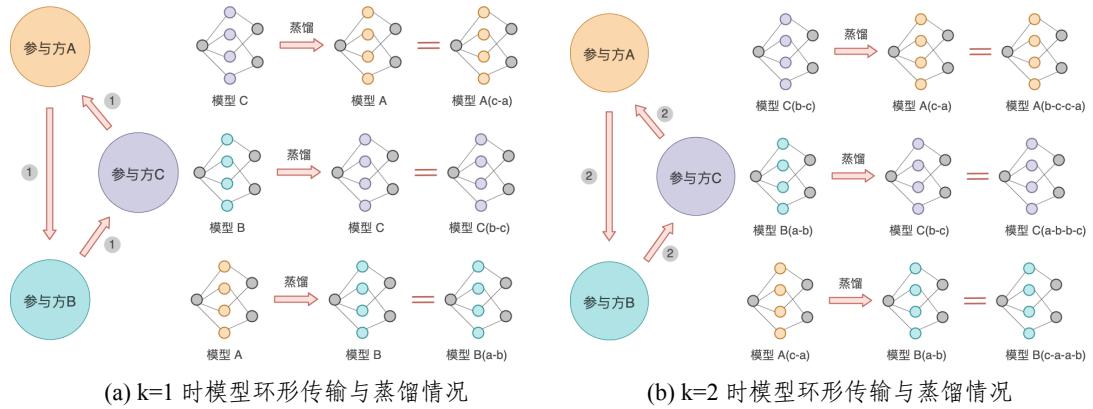


图 3.6 模型环形传输蒸馏演示图（以 $K=3$ 为例）

如??所示，假设联邦中存在 3 个参与方（即 $K=3$ ），分别为参与方 A、参与方 B 和参与方 C。一轮环形蒸馏共需执行 2 次（即 $K-1$ 次）模型传递与蒸馏，假设联邦中各个参与方均执行蒸馏（即 $\lambda > 0$ ）。

当执行第 1 次模型传递与蒸馏时（即 $k=1$ ），如??所示，各参与方上的模型传输与蒸馏情况如下：

- 参与方 A 获取参与方 C 的模型 f_C ，模型 f_C 蒸馏模型 f_A 得到模型 $f_{A(c-a)}$ ；
- 参与方 B 获取参与方 A 的模型 f_A ，模型 f_A 蒸馏模型 f_B 得到模型 $f_{B(a-b)}$ ；
- 参与方 C 获取参与方 B 的模型 f_B ，模型 f_B 蒸馏模型 f_C 得到模型 $f_{C(b-c)}$ 。

当执行第 2 次模型传递与蒸馏时（即 $k=2$ ），如??所示，各参与方上的模型传输与蒸馏情况如下：

- 参与方 A 获取参与方 C 的模型 $f_{C(b-c)}$ ，模型 $f_{C(b-c)}$ 蒸馏模型 $f_{A(c-a)}$ 得到模型 $f_{A(b-c-c-a)}$ ；
- 参与方 B 获取参与方 A 的模型 $f_{A(c-a)}$ ，模型 $f_{A(c-a)}$ 蒸馏模型 $f_{B(a-b)}$ 得到模型 $f_{B(c-a-a-b)}$ ；

- 参与方 C 获取参与方 B 的模型 $f_{B(a-b)}$, 模型 $f_{B(a-b)}$ 蒸馏模型 $f_{C(b-c)}$ 得到模型 $f_{C(a-b-b-c)}$ 。

我们以参与方 B 为例, 在 $k=1$ 时参与方 B 接收了参与方 A 的模型并进行了蒸馏, 获得了参与方 A 的知识, 得到模型 $B(a-b)$; 在 $k=2$ 时参与方 B 接收参与方 A 的模型 $A(c-a)$, 模型 $A(c-a)$ 是经过参与方 C 的模型蒸馏后的参与方 A 的模型, 包含参与方 A 和参与方 C 的知识, 经过蒸馏后得到模型 $B(c-a-a-b)$ 。在一轮环形蒸馏的过程中, 作为远端的参与方 C 相对于近端的参与方 A 来说, 对参与方 B 的知识传递次数更少、传递距离更远, 因此知识迁移的效果会更差。

算法 FedRKD 采用顺时针环形蒸馏与逆时针环形蒸馏交替执行的方式, 消除近端与远端的参与方在环形蒸馏过程中出现的知识传递倾斜的问题, 尽可能均衡环形联邦中不同参与方之间知识共享的效果。

3.5 本章小结

本章主要介绍基于知识蒸馏的联邦学习算法。首先分析了联邦学习系统的问题定义, 明确了当前工作存在的问题以及本论文算法需要解决的问题。本章首先介绍了联邦学习中存在的网络拓扑架构, 分别是客户端-服务器架构、对等网络架构下的点对点架构和环形架构, 对比了几种拓扑架构各自的优劣势。本章在客户端-服务器架构的基础上提出一种基于 C/S 架构的 C 端蒸馏联邦学习算法, 在客户端引入知识蒸馏以更好的进行全局模型知识到本地模型的迁移, 解决数据非独立情况下收敛慢、模型精度低等问题。本章在环形架构的基础上提出一种基于环形架构的双向环形蒸馏联邦学习算法, 在无中心服务器参与的情况下使用双向环形蒸馏进行模型训练, 解决数据非独立同分布带来的问题, 同时本小节对双向交替的环形蒸馏进行了详细的分析。

第4章 基于知识蒸馏的联邦学习算法实验

4.1 实验介绍

4.1.1 实验结果评价标准

联邦学习中，每个参与方都有拥有本地数据集，且各参与方在本地计算资源上训练深度学习模型，模型训练与评估是在每个参与方本地进行的，并且任何一个参与方都无法获取其他参与方的数据集。本地模型性能表示某一参与方在本地测试数据集上计算得到的模型精度。全局模型性能表示所有客户端在测试数据集上计算得到的平均模型精度。由于本文的所有实验均是基于图像分类的任务，因此根据模型精度（即准确率）评价准则进行评估，如式(??)所示。

$$\text{accuracy} = \frac{TP}{TP + FP} \quad (4.1)$$

其中 TP 代表真正例 (True Positive)，表示为预测值与标签值相等的情况，而 FP (False Positive) 代表假正例，表示为预测值与标签值不相等的情况。

实验中对算法收敛速率的定义为模型收敛的快慢，我们可以根据算法训练模型时达到收敛状态的轮次侧面反映收敛速率，算法达到收敛状态时所用的轮次越少，则收敛速率越快，反之则收敛速度越慢。

4.1.2 实验环境配置

实验环境下使用的机器与环境配置情况如??所示。

表 4.1 实验环境配置情况表

项目	参数
系统环境	Ubuntu18.04
软件环境	Python 3.8、Cuda 11.0、PyTorch 1.7.0
处理器	15 核 Intel(R) Xeon(R) Platinum 8358P @ 2.60GHz
显卡	RTX 3090
内存/显存	90GB/24GB

4.1.3 对比算法介绍

我们将 FedCKD 算法和 FedRKD 算法与当前 SOTA 的几种算法进行比较，包括经典的联邦学习方法、针对非独立同分布数据设计的联邦学习方法以及基于环形架构设计的联邦学习算法，这几种算法的具体描述如下：

- **FedAvg^[?]**。服务端直接聚合模型的参数，客户端直接使用全局模型，未进行任何个性化操作。
- **FedProx^[?]**。在 FedAvg 的基础上，客户端训练过程中增加了损失函数的修正项，允许局部模型和全局模型之间存在细微差异。
- **FedBN^[?]**。FedBN 算法通过在客户端局部模型中加入了批量归一化层（BN），保留本地 BN 层，解决联邦学习中数据非独立同分布的问题。
- **metaFed^[?]**。MetaFed 是一个环形联邦学习框架，通过自适应环形知识蒸馏的方式，在无中心服务器参与的情况下提高模型训练精度和个性化学习的方法。

4.1.4 模型与蒸馏设置

以下实验中，我们主要使用到 LeNet-5 模型和 AlexNet 模型，LeNet-5 模型用于处理 CIFAR10 数据集和 MNIST 数据集的图像数据，AlexNet 模型用于处理 VLCS 的图像数据。

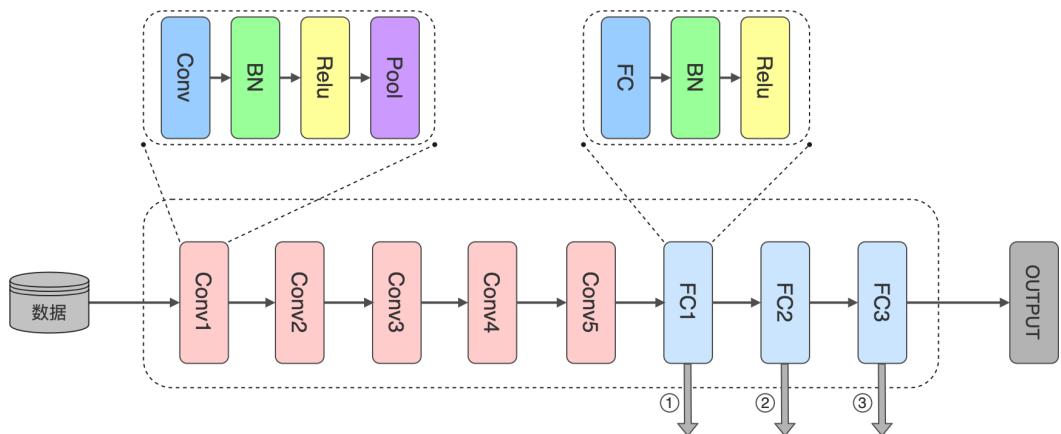


图 4.1 AlexNet 模型特征提取示意图

联邦学习的蒸馏阶段,我们使用的是基于特征的知识蒸馏算法,对于 AlexNet 模型,我们主要提取最后的三层全连接神经网络层作为特征层进行训练,如??所示。

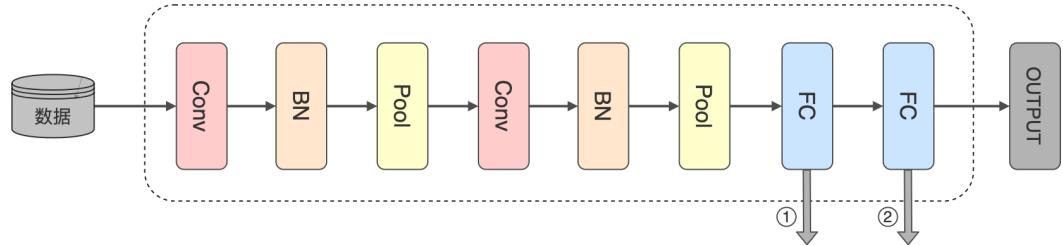


图 4.2 LeNet 模型特征提取示意图

对于 LeNet-5 模型,我们主要提取最后的两层全连接神经网络层作为特征层进行训练,如??所示。

4.2 基于特征偏移数据集的图像分类效果

4.2.1 VLCS 数据集介绍

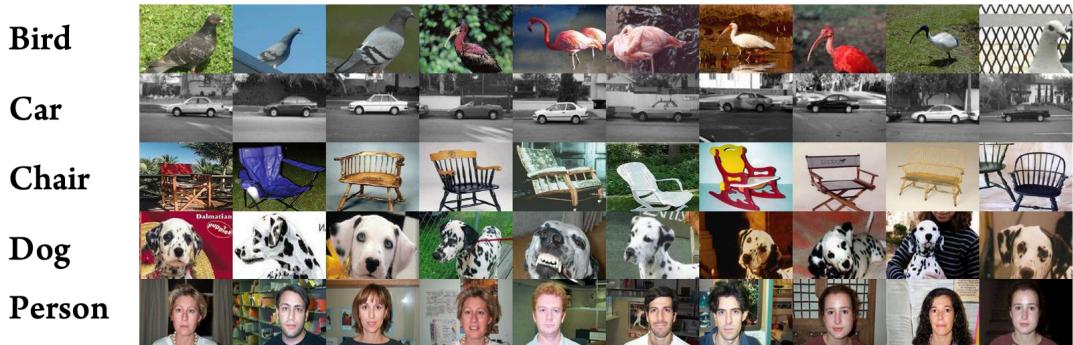


图 4.3 VLCS 数据集样例图

VLCS 是一个著名的特征偏移数据集,由 Chen Fang^[?]等人进行整理,VLCS 包括 4 个图像子数据集 (Caltech101/LabelMe/SUN09/VOC2007),包含 5 个类别 (bird/car/chair/dog/person) 的 10729 张图片,这里抽取了 VLCS 的小部分数据样例,如??所示,我们将每个图像子数据集作为一个参与方来进行联邦训练,总共 4 个参与方。

4.2.2 实验细节设置

实验设置所有的联邦学习算法训练轮次均为 100 轮。由于 VLCS 包含 4 个图像子数据集，联邦学习参与方节点设置为 4，分别代表四个不同的数据集（Caltech101/LabelMe/SUN09/VOC2007），每个参与方节点内部训练迭代次数为 3 轮。由于每个数据集包含太多图像，我们选择 10% 作为训练数据集，10% 作为验证数据集，20% 用于测试数据集。我们使用 AlexNet 模型训练图像，采用三层全连接神经网络作为分类器。每个批次大小为 32，默认学习率设为 0.01，优化器使用 SGD 随机提督下降优化器，损失函数采用交叉熵损失函数。对于 FedCKD，设置参数 $\mu_0=0.5$ ；对于 FedRKD，设置参数 $\lambda_0=1.0$ 。

4.2.3 结果与分析

表 4.2 不同联邦学习算法在 VLCS 上的表现

算法	Caltech101	LabelMe	SUN09	VOC2007	AVG
FedAvg	70.21	56.60	47.26	45.70	54.94
FedProx	78.72	50.57	51.52	46.88	56.92
FedBN	70.92	<u>63.02</u>	<u>55.49</u>	50.15	<u>59.89</u>
MetaFed	74.47	59.62	53.05	49.26	59.10
FedCKD	85.82	58.49	40.55	<u>50.74</u>	58.90
FedRKD	<u>80.98</u>	64.53	56.32	51.34	63.29

展示了 FedAvg、FedProx、FedBN、MetaFed 和 FedCKD、FedRKD 算法在 VLCS 数据集上的表现。其中粗体表示最好的结果，下划线表示次好的结果，我们从这些结果中可以观察到以下几点：

1. FedRKD 算法获得了最佳的平均效果，显著提高（与 FedBN 相比超过 3.4%）。
FedRKD 算法在 3 个参与方节点上都达到了最好的性能，在 1 个参与方节点上达到了次好的性能，证明了我们的算法有效且性能优秀。
2. FedCKD 算法在 Caltech101 上的效果表现最佳，达到了 85.82%；在 VOC2007 上的表现效果次佳，达到了 50.74%。FedCKD 算法的平均效果排名第 4，达

到 58.90%，优于 FedAvg 和 FedProx 但略次于 MetaFed、FedBN 和 FedRKD。

FedCKD 算法在 SUN09 上面表现较差，只有 40.55%，说明 FedCKD 在处理特征偏移的数据集上表现的不太稳定。

3. 平均效果排名，FedRKD>FedBN>MetaFed>FedCKD>FedProx>FedAvg。由于特征偏移的情况下，FedBN 比 FedAvg 有更好的性能，FedProx 的性能可以接受。MetaFed 效果略低于 FedBN，但好于 FedAvg 和 FedProx。

总结来说，FedCKD 效果好于 FedAvg 和 FedProx，这证明了知识蒸馏在客户端用作模型知识迁移，效果好于直接进行模型替换。FedRKD 算法取得了最佳平均效果，证明了我们再环形架构下的知识蒸馏算法在非独立同分布数据集上的性能稳定且优于目前 SOTA 的几类算法。

4.3 基于 MINIST 和 CIFAR10 的图像分类效果

4.3.1 数据集介绍



(a) MNIST 数据集数据样例图

(b) CIFAR10 数据集数据样例图

图 4.4 MNIST 与 CIFAR10 数据集数据样例图

- **MNIST。** MNIST 是一个 10 分类的手写数字数据集，编号从 0 到 9，如??所示。该数据集由 60000 张训练图像和 10000 张测试图像组成，每个类别的图像分别为 6000 和 1000 张，所有图片均为 28*28 的单通道图片。

- **CIFAR10。** CIFAR10 是一个 10 分类的常见图像识别数据集, 如??所示, 由 Krizhevsky^[?]进行整理, CIFAR10 包含 50000 张训练图和 10000 张测试图, 每个类别图像分别为 5000 和 1000 张, 图片均为 32*32 的彩色三通道图片。

4.3.2 实验细节设置

对于 MNIST 数据集和 CIFAR10 数据集上的实验, 设置所有的联邦学习算法训练轮次为 100 轮。两个数据集均采用狄利克雷分布进行数据集划分, 狄利克雷参数 α 设置为 0.1。联邦学习参与方节点设置为 5, 每个节点内部训练迭代次数为 3 轮。模型使用 LeNet-5 来进行图像训练。每个批次大小为 32, 默认学习率设为 0.01, 优化器使用 SGD 随机提督下降优化器, 损失函数采用交叉熵损失函数。对于 FedCKD, 在 MNIST 数据集上设置参数 $\mu_0=0.9$, 在 CIFAR10 数据集上设置参数 $\mu_0=0.5$; 对于 FedRKD, 在 MNIST 数据集和 CIFAR10 数据集上, 设置参数 $\lambda_0=1.0$ 。

4.3.3 结果分析

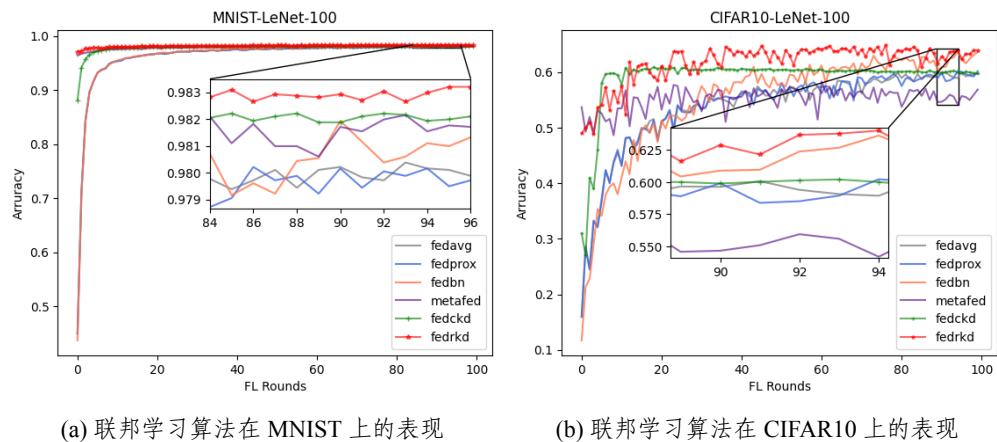


图 4.5 联邦学习算法在 MNIST 和 CIFAR10 数据集上的训练情况

从收敛速率的角度考虑, ??中可以看出 FedRKD 和 FedCKD 相比较 FedAvg、FedProx、FedBN 来说更快达到收敛。FedRKD 和 FedCKD 大概在 10~15 轮次之间达到了收敛, 但其他 FedAvg、FedProx、FedBN 则是在 40~50 轮次之间才达到

收敛。尤其是 FedRKD 的收敛速率更高，是因为 FedRKD 中每个参与方会首先使用本地数据集进行并行训练的原因，而 FedCKD 中没有此步骤，所以相比较 FedRKD 来说收敛速率相对较低，但收敛速率依旧远远超过 FedAvg、FedProx、FedBN。

表 4.3 参与方在 MNIST 数据集上的测试结果表

算法	参与方 1	参与方 2	参与方 3	参与方 4	参与方 5	AVG
FedAvg	97.89	98.12	98.25	97.67	<u>98.25</u>	98.04
FedProx	97.95	98.00	98.23	97.64	98.36	98.04
FedBN	98.28	98.00	98.89	97.87	97.92	98.19
MetaFed	98.03	98.39	98.84	98.25	97.56	98.22
FedCKD	98.15	98.01	<u>99.12</u>	98.07	97.82	<u>98.23</u>
FedRKD	<u>98.25</u>	<u>98.25</u>	99.17	<u>98.17</u>	97.87	98.34

展示了 FedAvg、FedProx、FedBN、MetaFed 和 FedCKD、FedRKD 算法在 MNIST 数据集上的表现。其中粗体表示最好的结果，下划线表示次好的结果，从中可以观察到以下几点：

1. FedRKD 取得了最佳平均准确率，相比较于 FedBN 和 MetaFed 来说，测试准确率提升了 0.15% 和 0.12%。其中 FedRKD 在参与方 3 上取得了最佳准确率，在参与方 1、2 和 4 上取得了次佳准确率。
2. FedCKD 取得了次佳平均准确率，同样作为中心化算法，相比较 FedBN 和 FedAvg 来说，测试准确率提升了 0.04% 和 0.19%。其中 FedCKD 在参与方 3 上取得了次佳准确率。
3. MetaFed 算法在 MNIST 上的表现好于 FedAvg、FedProx、FedBN 三个中心化联邦学习算法，FedBN 相较于 FedAvg 和 FedProx 也有较为明显的提升。

展示了 FedAvg、FedProx、FedBN、MetaFed、FedCKD 和 FedRKD 算法在 CIFAR10 数据集上的表现。其中粗体表示最好的结果，下划线表示次好的结果，从中可以观察到以下几点：

1. FedRKD 取得了最佳平均准确率，相较于 FedBN 和 MetaFed 来说，测试准确率提升了 0.73%，相较于 FedAvg、FedProx 和 MetaFed 有更大幅度的提

表 4.4 参与方在 CIFAR10 数据集上的测试结果表

算法	参与方 1	参与方 2	参与方 3	参与方 4	参与方 5	AVG
FedAvg	65.76	56.24	59.19	61.54	<u>60.63</u>	60.67
FedProx	65.43	55.71	61.77	59.13	60.88	60.58
FedBN	<u>67.23</u>	56.60	69.98	69.30	58.55	<u>64.33</u>
MetaFed	65.87	48.70	65.68	67.58	43.78	58.32
FedCKD	65.01	<u>56.98</u>	66.98	66.81	48.63	60.88
FedRKD	68.23	63.11	<u>68.78</u>	<u>68.74</u>	56.44	65.06

升，分别达到了 4.39%、4.48% 和 6.47%。其中 FedRKD 在参与方 1 和 2 上取得了最佳准确率，在参与方 3 和 4 上取得了次佳准确率。

2. FedCKD 略好于 FedAvg、FedProx 和 MetaFed，测试准确率提升了 0.21%、0.3% 和 2.56%，但表现不如 FedBN。其中 FedCKD 在参与方 2 上取得了次佳准确率。
3. MetaFed 算法在 CIFAR10 上的表现远不及 FedAvg、FedProx、FedBN 三个中心化联邦学习算法，只到达 58.32%，FedBN 相较于 FedAvg 和 FedProx 有较为明显的提升，也超越了 FedCKD 的表现。

总结来说，在经过非独立同分布处理的公共数据集 MNIST 和 CIFAR10 上，FedRKD 和 FedCKD 相较于其他中心化联邦学习算法，收敛速率更高。其中 FedRKD 取得了最佳平均准确率，FedCKD 在 MNIST 上取得了次佳准确率，证明了我们算法在非独立同分布的数据集上性能表现优秀。

4.4 数据集分割方法

4.4.1 狄利克雷分布

在联邦学习中，联邦学习节点之间的数据集通常呈现非独立同分布特点，因此我们使用狄利克雷分布来进行数据集划分。

狄利克雷分布最初由法国数学家狄利克雷提出，它可以用来描述随机事件发生的频率。狄利克雷分布是一种“分布的分布”，其满足 $G \sim DP(\alpha, G_0)$ ，其

中 G_0 表示原始数据集的基础分布， α 是分布的参数， α 越大表示生成的分布接近于均匀， α 越小表示生成的分布则越呈现集中。

我们采取的算法思路是尽量让每一个节点上的标签分布情况尽可能不同，实验中我们按照数据集的标签分布来对样本进行分割。我们设有 K 个节点， N 个标签类别，每个标签类别的样本需要按照不同的比例划分在不同的节点上。我们设二维矩阵 X (X 大小为 $N*K$) 为标签类别的分布矩阵，其行向量表示类别 n 在不同的节点上的概率分布向量，也就是类别 n 的样本划分到不同节点上的比例，该随机向量采用狄利克雷分布。

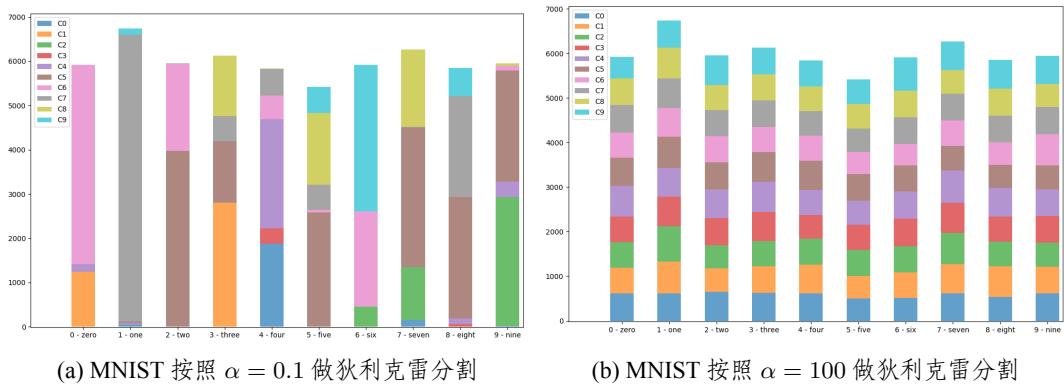


图 4.6 MNIST 按 $\alpha = 0.1$ 和 $\alpha = 100$ 做狄利克雷分割的分布结果

以数据集 MNIST 为例，我们将 MNIST 数据集按照狄利克雷分布进行分割，我们已经知道 MNIST 标签类别为 0~9 的手写数字，假设节点 $K = 10$ ，设置狄利克雷分布函数分布参数 $\alpha_i = 0.1$ 和 $100, i = 1, 2, \dots, K$ ，分割后的得到数据集分布的可视化结果如??所示，我们可以观察到当 $\alpha = 100$ 的数据分布相比于 $\alpha = 0.1$ 的数据分布会更加均匀。

4.4.2 α 对联邦学习算法的影响

我们依旧使用 MNIST 和 CIFAR10 两个数据集进行实验测试，设置所有的联邦学习算法训练轮次为 100 轮。联邦学习参与方节点设置为 5，每个节点内部训练迭代次数为 3 轮。我们使用 LeNet 模型训练图像。每个批次大小为 32，默认学习率设为 0.01，优化器使用 SGD 随机提督下降优化器，损失函数采用交叉熵损失函

数。对于 FedCKD, 在 MNIST 数据集上设置参数 $\mu_0=0.9$, 在 CIFAR10 数据集上设置参数 $\mu_0=0.5$; 对于 FedRKD, 在 MNIST 数据集和 CIFAR10 数据集上, 设置参数 $\lambda_0=1.0$ 。实验中我们将 α 设置为变量, 分别测试 $\alpha = 10^{-2}, 10^{-1}, 10^0, 10^1, 10^2, 10^3$ 条件下, 不同算法训练出来的模型的性能。

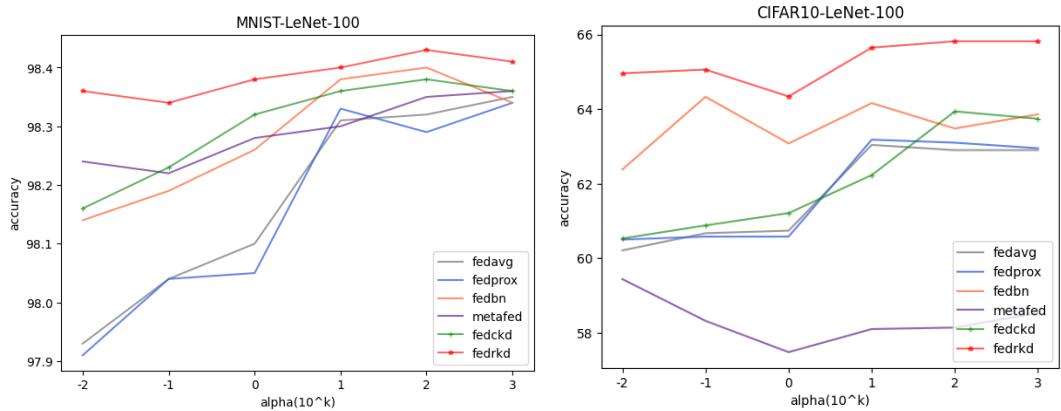


图 4.7 狄利克雷分割中 α 对 FedCKD 和 FedRKD 算法的影响

展示了狄利克雷分布中的参数 α 设定对于 FedCKD 和 FedRKD 算法的影响, 从图中我们可以看出以下几点:

1. FedRKD 随 α 的变动, 模型性能的波动变化不大, 在 MNIST 和 CIFAR10 上模型性能波峰与波谷相差 0.09% 和 0.72%。我们知道 α 越小, 则数据越呈现非独立同分布 (Non-IID), α 越大, 数据越呈现独立同分布 (IID)。FedAvg、FedProx 的波动比较明显, 在 α 较小时性能较差, 随着 α 的提升性能明显提升。FedBN 和 MetaFed 的波动相对较小, 但是从模型的性能上来说总体不及 FedAvg, 尤其是 MetaFed 在 CIFAR10 上的表现较差。这说明 FedRKD 算法更加稳定, 能抵抗数据非独立同分布带来的问题。
2. FedCKD, 在 MNIST 和 CIFAR10 上模型性能波峰与波谷相差 0.22% 和 3.41%, 在 MNIST 数据集上稳定性相较于 FedAvg 和 FedProx 来说有一定的提升, 但总体来说对于数据非独立同分布的抵抗性不如 FedRKD。
总结来说, FedRKD 算法更加稳定, 能抵抗数据非独立同分布带来的问题。FedCKD 稳定性相较于 FedAvg 和 FedProx 来说有一定的提升。

4.5 蒸馏方向对 FedRKD 的影响

4.5.1 实验设置

我们依旧使用 MNIST 和 CIFAR10 两个数据集进行实验测试，设置所有的联邦学习算法训练轮次为 100 轮。联邦学习参与方节点设置为 5，每个节点内部训练迭代次数为 3 轮。我们使用 LeNet 模型训练图像。每个批次大小为 32，默认学习率设为 0.01，优化器使用 SGD 随机提督下降优化器，损失函数采用交叉熵损失函数。设置 FedRKD 参数 $\lambda_0=1.0$ 。

实验中我们将 FedRKD 算法中的蒸馏方向设置为变量，分别测试双向蒸馏、单向蒸馏（顺时针蒸馏、逆时针蒸馏）条件下 FedRKD 算法的性能。单向蒸馏指在每一轮环形蒸馏过程中，模型传输方向不变，一直采用顺时针或逆时针进行模型传输。而双向蒸馏则每进行完一轮环形蒸馏后会改变模型传输的方向，顺时针环形蒸馏与逆时针环形蒸馏交替进行。

4.5.2 结果与分析

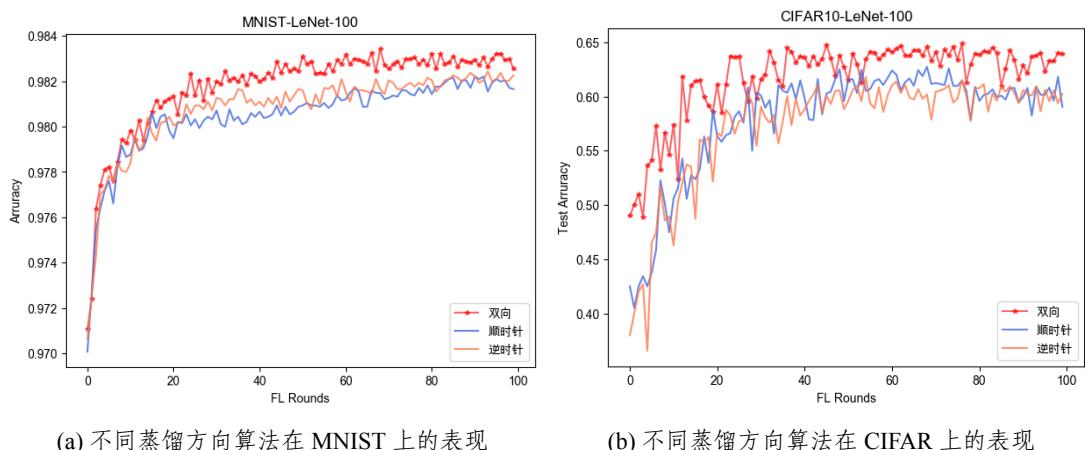


图 4.8 FedRKD 算法不同蒸馏方向对模型训练训练情况的影响

??和??展示了单向和双向环形蒸馏在 MNIST 和 CIFAR10 数据集上的表现。其中粗体表示最好的结果，从中可以观察到以下几点：

表 4.5 FedRKD 算法不同蒸馏方向设置的参与方结果表

数据集	蒸馏方向	1	2	3	4	5	AVG
MNIST	双向	98.25	98.25	99.17	98.17	97.87	98.34
	顺时针	98.24	98.13	98.90	98.10	97.74	98.22
	逆时针	98.20	98.00	99.03	97.98	98.00	98.24
CIFAR	双向	68.23	63.11	68.78	68.74	56.44	65.06
	顺时针	63.99	63.48	61.95	62.49	61.77	62.73
	逆时针	63.14	62.18	57.59	67.18	58.94	61.81

- 从??可以看出，使用双向环形蒸馏的 FedRKD 算法在模型收敛速率上比单向环形蒸馏的 FedRKD 更高。
- 从??中可以看出，FedRKD 使用双向环形蒸馏的模型平均准确度远高于使用单向环形蒸馏的模型平均。在 MNIST 数据集上，使用双向环形蒸馏比顺时针单向环形蒸馏和逆时针单向环形蒸馏高出 0.12% 和 0.10%；在 CIFAR10 数据集上，使用双向环形蒸馏比顺时针环形蒸馏和逆时针单向环形蒸馏高出 2.33% 和 3.25%。

总结来说，使用双向环形蒸馏方法比单向环形蒸馏方法收敛速率更高，模型平均准确率更好。

4.6 本章小结

本章主要介绍基于知识蒸馏的联邦学习算法的实验。首先介绍了模型精度和收敛速率作为实验结果的评价标准，以及实验环境的配置，对比的算法和模型与蒸馏的设置。我们分别在在 VLCS 数据集、MNIST 和 CIFAR10 三种数据集上对 FedCKD 和 FedRKD 进行了实验，对比了 FedAvg、FedProx、FedBN 和 MetaFed 算法，证明了我们算法在非独立同分布数据集上取得较好的效果。然后我们在不同的狄利克雷分布中测试了 FedRKD 和 FedCKD 算法，得出了 FedRKD 算法在不同分布的数据集上表现更加稳定，FedCKD 稳定性也强于其他去中心化算法。再者，我们测试了 FedRKD 使用双向环形蒸馏的优势，在算法收敛速率和模型精度上都远超过使用单向环形蒸馏。

第 5 章 总结和展望

5.1 全文总结

本论文主要研究基于知识蒸馏的联邦学习算法。在数据爆发式增长的时代，人们对数据隐私的重视度越来越高，不同国家和机构都在出台相关法律法规，以规范数据的使用和管理。联邦学习允许训练数据分布在参与方本地，在中心服务器的协调下共同训练一个全局模型，解决数据孤岛的问题并保护数据隐私安全。常见的联邦学习算法主要是基于中心化的客户端-服务器拓扑架构的前提来进行论述，但中心化联邦学习通常存在网络拓扑架构不稳定，中心服务节点通信负载高，易受到内外部攻击等问题。联邦学习不同参与方之间的数据往往存在 Non-IID 的问题，经典的联邦学习算法在数据 Non-IID 的条件下，存在收敛速度慢或者不收敛、模型效果差等问题。知识蒸馏是一种高效的模型压缩与知识迁移技术，知识蒸馏结合联邦学习可以在一定程度上解决 Non-IID 带来的问题。为了解决上述问题，本论文提出了一种基于客户端蒸馏的联邦学习算法——FedCKD 和一种基于环形架构的双向蒸馏联邦学习算法——FedRKD，然后论文对两个算法进行了实验验证。本论文主要工作内容如下：

1. 论文介绍了联邦学习中存在的网络拓扑架构，分别是客户端-服务器架构、对等网络架构下的点对点架构和环形架构，对比了几种拓扑架构各自的优劣势。
2. 论文在客户端-服务器架构的基础上提出一种基于客户端蒸馏的联邦学习算法——FedCKD，在客户端引入知识蒸馏以更好的进行全局共性知识到本地模型的迁移，解决数据非独立同分布带来的问题。论文在环形架构的基础上提出一种基于环形架构的双向蒸馏联邦学习算法——FedRKD，在无中心服务器参与的情况下使用双向环形蒸馏进行模型训练，实现去中心化联邦学习以及解决 Non-IID 问题。
3. 论文在 VLCS 数据集、MNIST 和 CIFAR10 三种数据集上对 FedCKD 和 FedRKD 进行了实验，对比了 FedAvg、FedProx、FedBN 和 MetaFed 算法，证明了两种算法在非独立同分布数据集上取得优秀且稳定的性能。论文在

不同的狄利克雷分布中测试了 FedRKD 和 FedCKD 算法，得出了 FedRKD 算法在不同分布的数据集上表现更加稳定，FedCKD 稳定性也强于其他去中心化算法。论文测试了 FedRKD 使用双向环形蒸馏的优势，在收敛速率和模型精度上都远超过使用单向环形蒸馏。

5.2 后续工作展望

本论文提出的基于知识蒸馏的联邦学习算法仍然有许多地方存在可提升的空间，下面阐述几点本论文的不足之处和后续可以继续研究改进的方向：

1. 目前实验中的两个参数 μ_0 和 λ_0 我们设定为固定值，探究这两个参数设置的最佳值也是一个非常值得研究的问题，希望在后续的研究中继续进行展开对参数最佳值的探索。
2. 在联邦学习中主要瓶颈在于联邦之间的通信量，如何解决降低节点之间的通信量是一个很重要的问题。知识蒸馏可以用于模型压缩，本文中所采用的特征蒸馏方法，主要都是基于相同模型架构之间做知识迁移，而不涉及到模型的压缩。还可以探究在不同联邦之间使用不同的模型架构，采用知识蒸馏的方式进行联邦之间串联，实现不同联邦之间的模型异构，这会是一个非常有意义的研究课题。
3. 隐私保护是联邦学习中另一个非常重要的课题，本文中训练完成的模型在联邦之间通信是不经过任何加密处理的，模型在一定程度上会泄露训练数据的信息，如何结合同态加密、差分隐私等密码学方法，对模型进行加密处理将会是一个非常有意义的研究方向。

作者简历

致 谢

两年多的求是园时光转瞬即逝，遥想 2020 年的 9 月，我怀着忐忑而有好奇的心情踏入浙江大学，很幸运能与灿若星辰的浙大人一起同行。在毕业论文完成之际，我要感谢在我整个写作阶段乃至研究生时光中对我寄予过帮助的每一个人。

首先我要特别感谢的是我的研究生陈奇副教授和王备学长，从论文的选题，开题的论证，实验的设计，论文写作与指导，他们给我提供了良好的环境和悉心的指导。让我从对联邦学习和隐私计算这一领域有了深入的了解，以总结出一套算法流程和专利，并进行实验的验证分析，最终完成论文写作。老师们对待科研的热情和对待工作认真负责的态度，是值得我终身去学习的。

此外还要感谢研究生阶段的舍友和实验室的同门以及在区块链俱乐部认识的小伙伴，是你们的陪伴和相互鼓励，丰富了我研究生阶段的生活，谢谢在求是园中遇到的每一个你们。感谢我在实习阶段遇到的每一位领导和同事，是你们的指导让我慢慢适应了职场的节奏，也给我未来的职业生涯规划许多启示。感谢 MatchUs 团队的小伙伴们，浙里的脱单率没有我们不行！与大家举办的每一期活动都给予了我许多成就感，也更加坚定了我未来要做有价值产品的决心。

最后，我还要感恩我的父母在经济和精神上给予我无条件的支持，有你们我才能够静心学习，并顺利完成研究生学业。

署名

20XX 年 XX 月 XX 日

于西湖畔