1 绪论

1.1 背景

近年来，人工智能领域的不断发展，引领着各行各业的革新，机器学习技术作为人工智能领域中的重要技术，其应用面也是越来越广。在人工智能的新时代里，机器学习技术在不断深化应用的过程中，不断影响着人工智能的未来发展方向，同时也遇到了各种各样的难题。

与此同时，“互联网+”概念的不断深化，促使全球的互联网发展迅猛。互联网与计算机技术不断渗透到各行各业，共同将人类带入到全新的互联网时代。

互联网时代带来的是前所未有的海量数据，对于数据的挖掘与应用，是这个时代的命题。面对如此巨大的数据，传统的单端机器学习技术效率堪忧，因此，分布式机器学习技术，应运而生。

在分布式机器学习的发展过程中，有一个重要的挑战出现了，那就是数据的来源。分布式机器学习需要获取到各个不同的端的数据，而在现实生产生活中，数据往往是被某单位独享的，这使得这些业务场景并不适合传统的中心化的模型训练方式，更适合去中心化的模型训练方式。基于此，作为一种去中心化的方法，联邦学习受到了广泛关注。

1.2 研究目的与意义

在互联网日益发展的今天，数据量空前庞大，常规的机器学习技术因其单设备运行无法并行处理等缺点，其处理效率过低，模型过大，甚至几乎无法处理一些超大量的数据等问题的暴露，都使得它无法适用于新型场景。

而在新型的工商业环境下，作为新世纪最重要“资源”之一的数据，本身价值很高，往往掌握在企业等大型机构之中，换言之即算法算力与数据，并不能如同传统方式那样完全结合。因为分布式的机器学习虽然效率较高，不怕大数据的计算，但是因为这种计算都依赖于数据的完全暴露，所以在私密性较高的领域，分布式机器学习也会显得无法发力。

分布式机器学习本质上与传统的机器学习方式并没有太大的区别，其中的差距无非只是模型的分布式计算，数据的分布式处理等，它从本质上来说，只是提高了效率，能够更好地处理更大规模的数据。

而联邦学习是一种去中心化的方法，这一方法将训练数据分布式地保存在移动设备上，中心服务器通过聚合移动设备上本地计算得出的更新来学习一个共享模型。它实现了模型训练与直接访问原始训练数据的解耦，中心服务器只需要利用各个移动设备上本地计算出的一个更新值即可。

联邦学习在解决数据隐私性的问题的同时，也顺带着提高了工作效率，因为在不同的数据集，利用不同的设备进行训练，也意味着分布式地将数据进行了处理，相比较传统的单设备处理，其效率也会有大大提升。

作为最近几年新兴的技术，联邦学习依然具有很大的优化空间和研究价值。

在目前的商业发展情况下，各大公司、各大单位之间的数据隔阂越发严重，因为数据的隔阂，在许多例如医疗等领域，都无法高效地利用人工智能来进行准确地预测与辅助，造成了大量数据资源的浪费。联邦学习恰好适用于这些困难场景，因此对其进行优化，使得它能够更快、更精确、更安全、更稳定地发挥作用，正是迫在眉睫的事。

就目前来说，联邦学习隐含的优化问题包括非独立同分布、不平衡、大规模分布、沟通有限等问题，都亟待解决。本文主要针对机器学习算法等方面的优化，进行研究。

1.3 研究流程与方法

本次研究的主题是，联邦学习算法 FedAvg[1]的评测与传统算法应用于联邦学习过程中的评测与分析。在本文中，我们综合测试了五种算法，包括联邦学习奠基算法 FedAvg，以及由本文简单改写后用于联邦学习的 Federated Adagrad 算法，Federated Adadelta 算法，Federated RMSprop 算法，Federated Adam 算法[2-4]。

事先需要说明的是，对于传统算法用于联邦学习的改写，本文只是使用了较为简单的方法，其准确度很难做到很高，仅仅用于研究和分析算法在联邦学习下的大致特征。

一个典型的联邦学习过程一般包含以下步骤：

一，从客户端中选择一个子集，子集中的每个客户端从服务端下载当前的模型。

二，子集中的各个客户端基于本地数据计算得到一个更新后的模型。

三，子集中的每个客户端将模型更新发送到客户端。

四，服务端聚合客户端所发送的更新，获得一个改善后的模型。

在本次研究中，主要过程有算法学习、算法推演与应用、结果预测、实验环境搭建、测试数据集设计、测试与结果、分析修正与总结。

就目前来说，实验环境的搭建主要通过实验室现有的硬件机器，配合局域网，来模拟出现实情况下的联邦学习环境，并针对此进行优化。

过程中需要使用的开源框架与库主要为 Tensorflow，Docker，Scipy 等，计算机编程语言主要为 Python，数据处理工具主要为 Python 与 Matlab，文档、制表等工具主要为 Microsoft Office 套装与 Latex。

因为联邦学习环境搭建较为复杂，为了达到更加专业的联邦学习环境配置效果，本文的部分实验也会借鉴 Sebastian Caldas 等人所创建的 LEAF 评测系统[5]，从而更快更好地完成工作。

2 联邦学习概述

2.1 研究现状

2.1.1 分布式学习研究现状

分布式计算的出现，给大数据时代的数据分析和人工智能提供了一套可行的解决方案。目前比较著名的云计算平台有 Apache Hadoop 和 Spark 等。基于 Hadoop MapReduce 和 Spark RDD 模型，很多原本受困于算力而无法处理大规模数据的机器学习场景，都可以被较好的解决。后续 Zaharia 等人基于 Spark 提出了一种快速交互式 Hadoop 数据查询架构，性能有了质的飞跃。Huang 等人提出了基于 Spark 的海量遥感数据并行分析算法。在后续的研究中，保证大规模并行机器学习、分布式环境下的资源分配、性能跟踪、任务调度等，都是重要的课题。

在日常工业、商业或教学环境下，目前比较常用的三大分布式计算平台有 Spark、PMLS 和 TensorFlow。

Spark 将模型参数存储在驱动器的节点，通过通信，每次迭代一次，都会更新一次参数。这种方式会带来极大的多余开销，从而使得 Spark 变得不易扩展。

PMLS 节点会存储和更新模型参数同时也会及时响应工作期的请求，工作期会请求来自它们局部的 PS 副本的最新模型参数，并在分配给它们的数据集部分

上执行计算。

TensorFlow 使用节点和边的有向图来表示计算，其需要用户静态声明这种符号计算图，并对该图使用复写和分区，将其分配到机器上进行分布式执行。

基于这三大平台，纽约州立大学布法罗分校计算机教授进行测试，并将分布式机器学习平台归类为三大基本设计方法：基本数据流（Basic Dataflow）、参数服务器模型（Parameter-Server Model）、先进数据流（Advanced Dataflow）。

尽管分布式学习可以很好地优化传统机器学习的效率，支持大数据时代海量数据的挑战，但是无法解决因为数据价值私密性而产生的“数据孤岛”问题，因此，研究者们在分布式学习的基础上，开始尝试在不互相暴露数据的情况下，完成分布式的学习，而这便是联邦学习的诞生。

2.1.2 联邦学习研究现状

联邦学习的研究是近几年兴起的。最初 H.Brendan 等[1]首次提出了联邦学习、联邦优化的概念，并提出了联邦聚合（Federated Averaging）算法，此算法作为联邦学习最基础的算法，开启了联邦学习研究的大门。而广泛接受的关于联邦学习的正式定义，是到 2019 年才正式确定的。Q.Yang 等[6]详细介绍了联邦学习的概念以及最近的研究，重新定义了联邦学习的概念，即不同的数据拥有者在不互相暴露数据给彼此的前提下，协作训练一个联邦模型。并将联邦学习分为三大类：横向联邦学习、纵向联邦学习和联邦迁移学习。在此基础上，联邦学习的算法研究成为了一个热门研究领域，Adrian Nilsson 等[7]基于 MNIST 数据集对三种联邦学习算法进行评估：联邦平均（Federated Averaging）、联邦随机方差递减梯度（Federated Stochastic Variance Reduced Gradient）与 CO-OP；Cong Xie 等[8]提出了一种可以适应非独立同分布数据的异步联邦优化算法 FedAsync，旨在将原本的同步算法异步化，提高学习效率；ShiQiang Wang 等[9]分析了数据非独立同分布情况下的基于梯度下降法的联邦学习的收敛边界。根据这一理论边界提出了一种控制算法，可以在一定的资源条件限制下动态调整全局聚合的频率，较好地权衡局部更新与全局聚合操作，达到最小化损失函数的目的；Konstantin Sozinov 等[10]针对人体活动识别任务建立了两个模型：softmax 回归模型、深度神经网络模型，并分别利用联邦学习与集中式学习的方式训练两个模型，通过实验验证了在人体活动识别任务上，联邦学习可以达到与集中式学习一样的水平。

除了单纯的算法提升，对于异构数据的研究也是联邦学习的一大热门。K.

Sahu 等提出了一种新的联邦学习框架 FedProx，考虑了不同设备中数据的异构性，在 FedAvg 算法的基础上增加了一个正则项，限制 local model 和 global model 参数之间的距离，减小数据异构对算法收敛的影响。同时还提出了一个新的衡量不同设备之间差异性的指标 B-local dissimilarity；V. Smith 等[11]利用多任务学习方法来解决联邦学习中遇到的统计挑战，即数据异构，提出了一种优化算法 MOCHA，并提供了收敛性保证。这种算法还能解决联邦学习所面临的设备异构的挑战，如设备延迟、节点掉线等。Yue Zhao 等[12]指出了 non-IDD 的数据分布下，模型准确率下降的原因在于每个局部模型的权重与全局模型的权重之间的差异，据此文章提出了一种数据共享的策略，即在中心服务器上保存一个均匀分布的共享数据集，并在训练开始阶段由中心服务器利用共享数据集训练一个初始模型。针对联邦学习中出现的公平性问题，Mehray Mohri 等[13]提出了不可知联邦学习（AFL）框架。其中，中心服务器优化时所针对的目标分布是的客户端数据分布的任何可能组合，文章对此都提出了对应的优化算法。

因为需要设备间通信，因此通信方面的研究也不可或缺。考虑到网络连接速率的非对称性，即上行链路通常比下行链路慢得多，Jakub Konecny 等[14]尝试通过减少上行链路的通信代价进而减少联邦学习的整体通信代价；Neel Guha 等[15]提出了一种只需要进行一轮通信的联邦学习算法 One-Shot Federated Learning.主要思路为先在各个设备上独立的完成局部模型的训练，然后从所有的局部模型中依照一定的策略选择出一部分局部模型，将这些模型在中心服务器上进行集成；Sebastian Caldas 等[16]考虑通过减小从中心服务器向客服端的通信量来提高联邦学习的通信效率，提出了一种有损压缩下行通信数据的方法，此外本文还提出了 Federated Dropout 技术，即中心服务器在向客户端传输模型前先对模型进行 dropout，然后将得到的一个子模型传给客户端；客户端基于子模型进行训练，并将相应的参数更新返回给中心服务器；Corentin Hardy 等[17]人展示了在终端设备配置时，参数服务器模型的含义，为了压缩传输流量，作者提出了一种算法 AdaComp，用于压缩服务器上的工人更新（Worker Updates），与标准异步随机梯度下降相比，此算法使得 worker 向服务器发送的数据总量下降 2 个数量级，同时保留准确性；Takayuki Nishio 等[18]人提出了一种新的联邦学习协议，FedCS 协议，可以缓解客户端因计算资源有限、或无线信道条件较差时而带来的低效，同时可以根据资源条件主动管理客户端，其解决了资源限制情况下，客户端的选

择问题，使得服务器可以聚合尽可能多的客户端更新。

隐私保护是联邦学习的重要属性，为了使得数据拥有者的数据能够享有足够的私密性，许多学者对此进行研究。Keith Bonawitz 等[19]在 2016 年发表一篇文章，针对联邦学习提出了一种安全聚合协议，但此文并未完全阐述此协议，而在 2017 年他们又发表一篇类似文章[20]，完善了此安全聚合协议；Stacey Truex 等[21]提出了一种结合差分隐私与安全多方计算的联邦学习方法，在提高准确率的同时保持可证明的隐私保护，防止提取攻击与合谋威胁；Facebook AI 研究所的 Eugene Kharitonov[22]提出 FOLtR-ES 算法，其满足以下多种要求：保护用户隐私，通讯低成本，嘈杂数据环境处理，学习非连续 Ranking 的能力，并证明该算法具有 ϵ 局部差别隐私性（ϵ-local differential privacy），并对其隐私保障方面执行了分析和模拟，还通过实验研究了该算法的优化表现，以及其根据实验表现而做出的一些权衡。

联邦学习的应用研究也层出不穷。Andrew Hard 等[23]利用联邦学习的框架训练 CIFG 语言模型进行下一个单词预测，并与中心化模型下训练的结果进行了比较，说明了联邦学习可以达到与中心化训练相同甚至更优的结果；Fei Chen 等[24]提出了一个用于推荐的联邦元学习框架。在这一框架下，用户信息在算法层面共享，而不是在模型或数据层面；谷歌的 Timothy 等[25]将联邦学习进行商业级别、全球规模的训练，评估并部署出能够在不直接访问底层用户数据的情况下，提高虚拟键盘搜索建议的质量（Virtual keyboard search suggestion quality）的模型；为了处理会有 devices 随机离开或者再次加入训练过程，以及 edge devices 的更新率差距巨大的真实情况， Michael R. Sprague 等[26]人探究联邦学习的同步条件是否可以放宽，提出了一个新的[27]异步联邦学习算法，并且使用最先进的神经卷积模型，将异步联邦学习算法应用到了端对端的地理定位。Keith Bonawitz 等介绍了一个部署在移动手机（安卓）上的联邦学习系统。这一系统主要关注支持同步算法，如 Federated Averaging，解决了很多实际使用中的问题；MuLi 等[28]在这篇文章中详细描述了参数服务器的第三代开源实现，并且着重于系统层面的分布推理。这个系统在应用中保持简洁的同时还可以稳定高效的使用各种算法。这个参数服务器也是第一个可以扩展到工业规模的机器学习系统。

作为一个较新的技术概念，联邦学习的研究依然在进行之中。

2.2 算法简介

本文这些算法中，除了 FedAvg 算法以外，其余都是本文根据传统的人工智能算法而改写得到的用于联邦学习的算法，其算法核心依旧是传统的算法，只是根据联邦学习的需求而进行了适当调整。

2.2.1 Federated Averaging 算法（FedAvg）

此算法是联邦学习中最早最基础的算法之一。此算法基于 Stochastic Gradient Decent 算法（SGD），每轮训练时，将选择一部分客户端，利用局部数据进行训练，获得局部的梯度更新，并将这些局部更新聚集在客户端，进行加权平均，从而获得全局的梯度更新，接着在全局模型上应用全局更新，将更新后的模型参数结果继续返回给客户端。以此循环，来完成训练。

其核心优化算法 SGD 在每次循环都会计算 mini-batch 的梯度，并根据其梯度来对模型参数进行更新，在传统机器学习中，SGD 也算是最常用的方法之一了。

$$g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1}) \tag{2-1}$$

$$\Delta\theta_t = -\eta * g_t \tag{2-2}$$

从公式可见，SGD 非常依赖当前 batch 的梯度，这使得它有诸多缺点，例如学习率的选择很难，收敛容易出现局部最优而非全局最优的情况，但也正是它的奠基，使得后续更多优秀的算法得以出现。

2.2.2 Federated Adagrad 算法（FedAdagrad）

此算法是本文将传统 Adagrad 算法应用于联邦学习的产物。

相比较别的算法，Adagrad 算法对学习率进行了一个限制，

$$n_t = n_{t-1} + g_t^2 \tag{2-3}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{\sum_{r=1}^{t}(g_r)^2+\epsilon}} \tag{2-4}$$

此处 g 根据递推公式，可以在 1 递增到 t 的过程中，形成一个限制项 regularizer，从而完成对学习率的一个限制。其特征在于，在训练刚开始，g 数值小，则 regularizer 数值较大，会将梯度加大。训练中后段，g 数值大，则 regularizer 数值小，会将梯度降低，从而起到限制与约束的作用。

2.2.3 Federated Adadelta 算法（FedAdadelta）

此算法是本文将传统算法 Adadelta 应用于联邦学习的产物。它本质上就是对 Adagrad 算法的拓展。此算法简化了对于梯度平方值的累加，将其规则改为累加大小恒定的项，并且不会将这些项保存，而是通过模糊近似求出大概的平均值。

$$n_t = v * n_{t-1} + (1 - v) * g_t^2 \tag{2-5}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t \tag{2-6}$$

同时为了让整体算法不要依赖全局的学习率，算法设计者还对其进行了近似牛顿迭代法的处理。

$$E|g^2|_t = \rho * E|g^2|_{t-1} + (1 + \rho) * g_t^2 \tag{2-7}$$

$$\Delta\chi_t = -\frac{\sqrt{\sum_{r=1}^{t-1} \Delta\chi_r}}{\sqrt{E|g^2|_t + \epsilon}} \tag{2-8}$$

其特征为，初期直到中期，整体训练速度很理想，接近末端则会在局部最小值出现频繁的上下浮动，后期速度也会减慢。

2.2.4 Federated RMSProp 算法（FedRMSProp）

此算法是本文将传统算法 RMSprop 应用于联邦学习的产物。此算法从某种意义上来说是 Adadelta 算法的特殊情况。通过调整 Adadelta 算法的ρ值到 0.5，再求出其平方和的根，即可获得 RMS（均方根）。

$$\mathrm{RMS}|g|_t = \sqrt{E|g^2|_t + \epsilon} \tag{2-9}$$

$$\Delta\chi_t = -\frac{\eta}{RMS|g|_t} * g_t \tag{2-10}$$

此算法对于全局学习率是有着依赖的，并且与 Adagrad 和 Adadelta 算法都有较大联系，算是 Adagrad 的扩展，其特性位于两者之间。

2.2.5 Federated Adam 算法（FedAdam）

此算法是 RMSprop 算法的衍生，相比较 RMSprop，此算法带有了动量项，利用梯度一阶矩估计和二阶矩估计动态调整每个参数的学习率。其特征与优势，在于校验和修正后，每一次迭代学习率都会有限制固定的变化区间，整体参数会较为平稳可靠。

$$m_t = \mu * m_{t-1} + (1 - \mu) * g_t \tag{2-11}$$

$$n_t = v + n_{t-1} + (1 - v) * g_t^2 \tag{2-12}$$

$$\widehat{m_t} = \frac{m_t}{1 - \mu^t} \tag{2-13}$$

$$\widehat{n_t} = \frac{n_t}{1-v^t} \quad (2\text{-}14)$$

$$\Delta\theta = -\frac{\widehat{m_t}}{\sqrt{\widehat{n_t}}+\epsilon} * \eta \quad (2\text{-}15)$$

从公式可见，对梯度的矩进行直接地估计对内存是没有任何额外的需求的，而且此算法可以根据梯度进行动态的调整。

此算法有着 Adagrad 算法和 RMSprop 算法各自的优势，既能很好地处理稀疏梯度，又能较好地处理非平稳目标，且对内存的需求很低，有着根据不同参数计算不同学习率的设计，适用性很优秀。

3 实验环境搭建

3.1 模拟环境设计

因为实验设备与时间有限，本次实验在同一台电脑上模拟运行。本实验在计算机上模拟总共 176 台客户端，进行客户端层面的分别训练，另外设 1 台中心服务器，用于与客户端交流，获得客户端发送的信息，并且进行模型的更新。

影响联邦学习的因素很多，包括但不限于算法优化程度、通信稳定与效率、隐私加密、客户端设备性能等，在此次的模拟环境中，因为所有模拟客户端与服务器都处于同一台电脑，就不存在网络传输方面的考虑，同时也不存在传输过程中因为加密协议与解密等操作而丧失的训练效率，并且处于同一台电脑之上，我们此次给模拟客户端分配的资源基本都是足够且相差不大的，所以这是一个变量较少，相对较为公平的实验环境，可以更加直观地看出各种算法对于联邦学习的优化效率。

联邦学习中存在同步算法与异步算法[8]的区别，简单来说便是交流发起者的不同——在同步算法中，普遍由服务端统一对每一次迭代后的客户端进行模型参数的收集与处理，每一次的迭代时间和轮次清晰，便于观察和统一管理，但是很显然，因为网络、隐私保护、客户端算力等差别，同步算法需要面临客户端之间完成训练的时间差的困扰，这会浪费一些总体时间。而在异步算法中，服务器往往扮演监听者角色，而客户端在完成一次训练之后，可以异步地向服务器传输信息，从而节约一些原本互相等待的时间，但异步算法如何进行有效的管理，这依然也是一个难题。

为了增加研究的便利与可比性，这些算法包括 FedAvg 和其他传统算法改写而来的联邦学习算法，都是联邦学习意义上的同步算法，这也就意味着，我们对于算法的评价频率与标准是可以统一起来的，可以根据其轮次与准确性的关系来进行比较有意义的评价。

3.2 数据集选择与处理

对于联邦学习数据集的选择，有着几个基本原则[5]：

数据具有自然键控生成过程。

数据的来源是千台到百万台设备之间，保证数据有足够的变化和随机性。

数据的数量点的数目在设备之间存在较大的偏差。

鉴于此，我们选择的数据集是 Federated Extended MNIST（FEMNIST）数据集，它类似于常用的手写数据集 MNIST 的扩展。此数据集也是较为优秀的联邦学习专用数据集，是将 Extended MNIST 数据集进行划分之后得到的。

联邦学习所面临的数据场景[6]与传统机器学习的不同，它的数据特点是 non-i.i.d.，即非独立同分布数据。传统机器学习的数据集处理之后，往往产生的是均匀随机的数据划分，而因为联邦学习面临的实际场景中，各个设备之间的数据状况差别很大，无论是数量、类型、具体数据都会有很大的差别，这也是许多优秀的传统算法在面临联邦学习场景时，效率会降低不少的原因。为了模拟出这种效果，我们也尽力将 non-i.i.d.融入到我们的实验中来。在 i.i.d.采样方案中，每个数据点都拥有相同的可能性被采样，所以从最终结果来看，每个客户端拥有的数据会呈现出基本相同的基础数据分布，而在 non-i.i.d.采样方案中，每个用户的基础数据分布和原始数据是一致的，基于我们在挑选数据集时，关于数据来源必须拥有较大差别，以及数据来源的用户量在千到百万数量级的假设，而我们可以认为，用户之间的数据分布是不同的，所以此采样过程就被认为是 non-i.i.d.过程。

为了保持算法评测的可比性，重复测试是有必要的，本文所测评的所有算法，会在两套事先采样完毕的数据集上，运行三次，并根据结果进行平均后再进行对比。

3.3 结果评价标准

考虑到模拟环境中许多不符合现实场景的设定，例如无网络速度干扰[14]，无隐私相关的干扰[19]，客户端之间算力基本相似等，在此实验条件下，对比任

何算法运行时间等相关的内容，并没有太大的现实意义。但排除了这些干扰因素之后，单纯地测试算法在联邦学习场景下基于训练轮次的训练效果，是比较合适的。

因为本文算法都是同步算法，所以我们的主要评价标准是，同时运行相同的训练轮次（2000），在每固定轮数（20）之间评估一次当前模型的准确率，并最终根据准确率随着训练轮次变化的趋势来做出关于各个算法的评测。

同时为了更好地了解不同客户端之间的跨设备差别，这边还引入了训练准确度前 10%和前 90%的客户端当前的准确度，作为一个评价标准，用于分析客户端之间训练效率的差别。

最后，为了分析在各个算法下，客户端需要承受的计算压力，此处还引入了 FLOPs 指标，主要通过计算所有设备中，FLOPs 值最大的设备，来得知稳定地完成一次训练，需要面临的计算压力。

基于以上指标，本文还尝试了两个可调整的训练参数来更好地分析联邦学习的特点，即每轮训练的客户端数，和训练的 epoch，此指标主要针对成熟的联邦学习算法 FedAvg 进行分析，作为本文的一些额外测试与收获。

以上所有的测试都会在两个事先采样完毕的 non-i.i.d.数据集上先后运行三次，以减少一些偶然性。

4 实验结果

4.1 不同算法准确度对比

下图实验数据是在 178 个客户端，每轮选取 35 个客户端，同时 epoch＝1 的情况下运行 2000 轮后各个算法的实验结果。

图 4-1 算法 FedAvg 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数折线图

图 4-2 算法 FedAdagrad 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数折线图

图 4-3 算法 FedAdadelta 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数趋势图

图 4-4 算法 FedRMSProp 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数趋势图

图 4-5 算法 FedAdam 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数趋势图

图 4-6 五种算法 2000 轮训练后平均准确率对比簇状图

由上图 1 到图 5 可见五种算法整体的准确率的增长趋势，因为整体实验环境本身并非十分精确，所以在分析时，我们主要还是根据结果图的趋势做一些定性和初步定量的分析。

就整体增长趋势来看，FedAdadelta 和 FedAdam 都有着非常优秀的前期训练效率，两者都在轮次不到 250 轮左右，就基本达到了 80%的平均准确率，并且在后续保持一个平稳的增长势头，最终锁定的准确率锁定在 83.8%和 83.9%。而 FedAdam 表现出了最差的训练效率，前期训练效率极低，并且在 2000 轮时期，准确率依然是 64.5%左右。

在稳定性方面，FedAdam、FedAdadelta 和 FedAvg 都表现出了较好的准确性上升趋势的稳定性，并且在跨设备的训练性能上看，FedAdam 和 FedAdadelta 都表现出了较平稳的趋势，前 10%和前 90%之间的差距明显小于其余三组。

4.2 不同算法客户端最大 FLOPs 值对比

经过 2000 轮训练后，各个服务器最大 FLOPs 值如下图 7 所示。

图 4-7 五种算法 2000 轮训练后最大 FLOPs 值对比簇状图

可以看到， FedRMSProp 表现出了较低的 FLOPs 值，而 FedAdam 与 FedAdadelta 则有着较高的 FLOPs 值，可能也与其算法较为复杂有关。

4.3 相同算法不同 epoch 与单轮训练客户端数的对比

此测评算是附加实验，用于探究联邦学习单轮训练客户端数目、Epoch 值，对最终准确度的影响，测评的算法是 FedAvg，在其余条件保持不变的时候，分别测评了单轮训练客户端数目 3，epoch=1 的条件，单轮训练客户端数目 35，epoch=1 的条件，以及单轮训练客户端数目 3，epoch=100 的条件，从而得到以下结果。

图 4-8 算法 FedAvg 在 178 客户端，每轮选取 3 客户端，epoch=1，2000 轮训练的准确率/训练轮数趋势图

图 4-9 算法 FedAvg 在 178 客户端，每轮选取 3 客户端，epoch=100，2000 轮训练的准确率/训练轮数趋势图

图 4-10 算法 FedAvg 在 178 客户端，每轮选取 35 客户端，epoch=1，2000 轮训练的准确率/训练轮数趋势图

如图 8 到图 10 所示，每轮选取客户端的数目与训练准确度曲线的平滑程度有一定的关系，很明显，图 10 算法的准确度上升曲线是最平滑的，而图 8 的曲线中，大量检测点与远离上升趋势，体现了其不稳定性。而在 epoch 增大的过程中，模型前期训练的速度会有显著提升，但就目前的结果来看，这些属性的调整，对训练 2000 轮后的最终准确率并没有产生太大的影响。

图 4-11 算法 FedAvg 在不同单轮选择客户端数目与 epoch 大小下的最大 FLOPs 值簇状图

如图 11 所示，epoch 对于 FLOPs 的影响极大，epoch=100 与 epoch=1 的情况相比，其 FLOPs 值也提升了有 106.7 倍，基本和 epoch 之间比例相近。而单轮训练客户端数目的增加也会对 FLOPs 有提升，其提升效果远不如 epoch 明显。

4.4 结果分析

根据本次实验结果，可做分析如下：

FedAvg 作为较成熟也较为经典的联邦学习算法，表现出了整体比较稳定的训练效率，虽然其核心算法 SGD 相比较别的算法有着较多的缺点，但是可能是因为在改写为联邦学习 FedAvg 的过程中，做了更好的优化，而笔者对别的算法所做的联邦学习适配方面的处理较为简单，所以 FedAvg 最终表现并没有像预期的那么差。

Adagrad 算法在被改写为 FedAdagrad 算法后，呈现出了糟糕的实验结果。根据其传统算法特征应该呈现出来的前期由于 regularizer 增大而带来的梯度增大加速训练效应，并没有体现得很好，可能是因为联邦学习多客户端的训练、中心服务器的更新模式，导致的不稳定性造成的，而后期最终 2000 轮后准确率远远低于别的算法，可能是因为 regularizer 数值减小后，导致梯度降低从而产生了较大的限制与约束作用，

FedAdadelta 算法有着优秀的表现，其传统算法特征中的初期训练很快的特征，也在此次实验中被体现了出来，作为第一个到达 80%准确度的算法最终准确率却并非最高，也体现了它中后期训练速度变慢的特点，并且可以设想，若训练轮次继续扩展，此算法的最终表现对比起来会差一些。

FedRMSProp 算法的表现一般，最终结果也介于 FedAdagrad 和 Adadelta 之间，也算是其特性的体现了。

FedAdam 算法表现优秀，动量项的引入和迭代学习率区间固定的设定，使其整体学习起来非常平稳，可能也是其表现出来前期较为快速的训练速度的原因之一，同时也是准确率前 10%设备与前 90%设备的实时准确率相差最小的原因。同时因为算法较为复杂，也导致了最后 FLOPs 值较大，但是瑕不掩瑜，FedAdam 依然是本次实验中最值得信赖的算法。

5 结论与展望

本文基于 FMMNIST 数据集测评与比较了经典联邦学习算法 FedAvg 与四种由传统算法改写而来的联邦学习算法 FedAdagrad，FedAdadelta，FedRMSProp，FedAdam，比较其准确度与设备最大 FLOPs 值，尝试分析了其算法表现与其算法特征之间地联系，同时也额外测评了以下联邦学习每轮选取客户端数目、epoch 数值大小对整体训练准确率的影响。

从算法本身的表现来说，FedAdam 以其优秀的稳定性和训练效率，在此次测评中，表现最为出色。而 FedAdadelta 位居第二位，同样拥有较好的前期效率和稳定性，但是最终结果和计算量使它位居第一名之后。FedAvg 以其成熟的联邦学习适配性位居第三，而 FedRMSProp 表现一般，FedAdagrad 则表现糟糕，位居第四和第五。

同时根据对于单轮客户端数目与 epoch 大小的评测来看，若要得到较好的联邦学习训练效果，在计算机算力允许的情况下，选取较高 epoch 和较高的单轮客户端数目，会带来更好的训练效果，同样的，这会带来更高的 FLOPs 值，带来更高的计算压力。

本文对于各个传统算法的改写较为简单，其测评结果有可能存在较大的误差，所以后续的工作中，可以对传统算法进行更为精细的改写和适配，从而获取更加优秀的训练效果。

同时，本文的实验环境排除了大量实际联邦学习中会遇到的干扰，如网络状况，隐私保护，设备性能等，使得本文的结果只能具有参考意义，而不具有实际应用意义，后续若想将算法应用于实践之中，可以在更加多干扰的情况下进行测评和改良，保证算法的容错率和稳定性。

再者，本文用到的数据集较为单一，联邦学习需要面临的数据可谓五花八门，保证算法在所有数据集上的优秀表现是必要的，所以后续还可以对各类算法在其余数据集上的表现进行测评。