基于数据库即服务的联邦机器学习数据解耦技术比较评估

Muhammad Jahanzeb Khan
内华达大学里诺分校，美国
jahanzeb@nevada.unr.edu

Rui Hu 内华达
大学里诺分校，美国
ruihu@unr.edu

Mohammad Sadoghi 加利福尼亚大学戴维斯分校，美国
msadoghi@ucdavis.edu

Dongfang Zhao 内华达大学里诺分校，美国
dzhao@unr.edu

arXiv:2303.08371v1 [cs.DB] 15 Mar 2023

摘要

联邦学习（FL）是一种分布式机器学习方法，允许多个客户端在不共享原始数据的情况下协作学习一个共享模型。最先进的联邦学习系统提供一体化解决方案；例如，用户必须安装并使用联邦学习系统所要求的数据管理子系统。这种紧密耦合的模式至多会阻碍联邦学习解决方案的广泛采用；在某些领域（如科学应用）中，它甚至可能成为无法接受的障碍，因为它会强制在专用硬件和平台上采用特定类型的数据解决方案（例如，没有节点本地持久化存储的高性能计算集群）。为此，一种自然的解决方案是将数据管理功能与联邦学习系统解耦，使客户端能够使用特定的数据子系统定制其联邦学习应用，并在这些子系统上执行高效查询。这类解耦方法面临的技术挑战包括新的系统架构、能够表达联邦学习参数的结构化数据模型，以及联邦学习参与者之间的安全保障。作为这一研究方向的起点，本文对联邦学习系统中的主流数据库解决方案作为解耦服务的表现进行了全面评估和比较。为确保比较公平，我们开发了一个名为数据解耦联邦学习（DDFL）的框架，该框架能够运行联邦学习工作负载，并通过提供统一接口实现与底层数据库系统无关。为评估我们方法的有效性和可行性，我们将 DDFL 与最先进的、紧密耦合数据管理与计算对应功能的联邦学习系统进行了比较。我们针对各种数据集和数据管理子系统（如 PostgreSQL、MongoDB、Cassandra、Neo4j）开展了大量实验，结果表明，DDFL 在训练时间、推理准确率和数据库查询时间方面的性能与现有方案相当或更优，同时为客户端提供了更多选项，使其能够根据性能、韧性和可用性等数据相关指标调整联邦学习应用。此外，我们还详细定性分析了 DDFL 与主流数据库系统集成时的表现。

工件可用性：
源代码、数据和/或其他材料已发布于 https://github.com/jahanxb/flcode。

1 引言

1.1 背景

联邦学习（FL） [43–45, 63, 64, 68, 69] 是一种分布式机器学习方法，使多个客户端能够协作训练共享模型，而无需将其原始数据传输至中央服务器。通过将数据保留在本地，联邦学习增强了各客户端的数据隐私并降低了通信成本，同时训练出能够反映所有参与者集体知识的精准模型。因此，近年来联邦学习受到了广泛关注，其潜力已在健康监测和自动驾驶等多种应用中得到展现 [3]。联邦学习对隐私和效率的重视，使其成为未来大规模机器学习应用的一项有前景的技术，尤其适用于数据安全至关重要的领域。

联邦学习（FL）的一个显著优势是，无需中央服务器收集和管理来自多个或众多客户端的数据，而这可能耗时、成本高昂且容易导致数据泄露。相反，联邦学习允许每个客户端将数据保留在本地，并通过安全通信协议参与共享模型的训练。这种方法不仅能够保护各个客户端数据的隐私，还能利用其数据的多样性来训练更加稳健、准确的模型。此外，联邦学习可以利用分布在众多客户端的数据训练机器学习模型，从而能够从更大、更多样化的数据集中学习，进而改善模型泛化并避免过拟合问题 [23]。

此外，联邦学习近年来已应用于医疗保健、交通运输和金融等各种现实世界应用，在这些领域，数据隐私至关重要。例如，联邦学习已被用于利用不同医院的医疗数据开发心脏病预测和癌症检测模型，而无需在医院之间传输数据 [28]。在交通运输领域，联邦学习已被用于利用多个来源的数据开发交通预测模型，这些来源包括配备 GPS 的车辆、交通摄像头和交通传感器，同时保护个体驾驶员的隐私并确保数据安全 [27]。在金融领域，联邦学习已被用于检测多家银行的欺诈交易，从而在共享信息的同时保护客户端隐私 [26]。



图 1：数据解耦联邦学习（DDFL）框架的拟议设计。

1.2 研究动机

尽管联邦学习（FL）系统 [2, 5–7]在可扩展隐私保护机器学习应用方面前景广阔，但仍面临一些挑战，必须在广泛部署之前加以解决。

其中一项挑战与联邦学习系统的数据管理方面有关。具体而言，在联邦学习应用的训练过程或验证过程中，客户端缺乏高效管理和查询中间模型的方式。更广泛地说，数据管理功能与其计算对应功能紧密耦合，导致整个联邦学习系统难以定制或泛化。更糟的是，某些应用或领域根本无法满足现代联邦学习系统的硬件／平台要求；例如，我们先前的工作 [4] 表明，现有联邦学习系统并不适用于缺乏本地持久化存储的大规模高性能计算（HPC）平台。

1.3 提出的解决方案

为应对这些挑战，我们提出一种将数据管理功能与联邦学习系统解耦的框架。我们的方法采用松耦合联邦学习架构，由专用数据库管理全局模型和本地模型。这种解耦使客户端能够利用特定的数据子系统定制其联邦学习应用，并使联邦学习系统能够扩展到大量客户端。通过这种方法，我们旨在提升联邦学习系统的灵活性、性能和可扩展性。

我们通过构建一个原型联邦学习系统来实现该方法，该系统采用了几种主流数据库选项，包括文档数据库 MongoDB [16],列式数据库 Cassandra [18], 图数据库（Neo4j） [24], 以及关系数据库 PostgreSQL [25]，如图 1 所示。该原型系统部署在 CloudLab [1] 上一个由 11 个节点组成的集群中。实验结果表明，数据解耦使客户端拥有更多选择和机会，能够从性能、韧性和可用性方面优化其联邦学习应用。

1.4 贡献

总之，这篇实验、分析与基准测试（EAB）论文做出了以下贡献：

- 我们提出了一种松耦合的联邦学习架构，将数据管理功能从联邦学习系统中分离出来。该方法使客户端能够使用特定的数据子系统定制其联邦学习应用，从而增强系统的互操作性和可扩展性。
- 我们通过构建一个原型联邦学习系统来实现所提出的方法，该系统采用了多种主流数据库系统，例如 PostgreSQL、Cassandra、MongoDB 和 Neo4j。据我们所知，这是首项在联邦学习系统背景下对不同数据库服务进行全面比较评估的研究。
- 我们在 CloudLab [1]上的一个 11节点集群中开展实验，对原型系统的性能进行了全面评估。实验结果表明，采用数据解耦可让客户端在性能、韧性和可用性方面拥有更多优化联邦学习应用的选择。

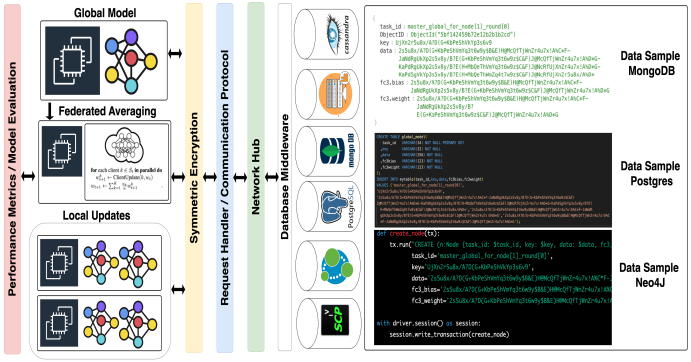


图 2：数据解耦联邦学习（DDFL）框架架构

2 数据解耦联邦学习（DDFL）框架

我们提出的框架称为数据解耦联邦学习（DDFL），旨在训练全局模型的同时维护隐私与安全。该框架采用解耦数据管理方法，客户端节点在本地存储各自的数据集，并与主节点协作，以迭代方式学习共享全局模型。中间结果和最终结果由专用的数据管理服务管理，该服务负责将数据管理与联邦学习系统解耦。图 2 展示了所提框架的总体架构。

在我们考虑的联邦学习系统中，有 N 个客户端节点和一个主节点。每个客户端节点都在本地存储自己的数据集，并在主节点的协调下，旨在 G 以迭代方式学习共享全局模型。在每一轮联邦训练中，客户端节点使用本地训练数据更新从主节点获取的全局模型。随后，它们将更新后的本地模型参数共享给主节点，由主节点聚合收到的所有本地模型，以更新下一轮训练所用的全局模型。

在 DDFL 系统中，训练过程中生成的中间模型由专用的数据管理服务进行管理，以实现数据解耦。全局模型会被下载并供本地模型使用，然后存储起来以备将来使用。

2.1 对称加密

在我们的系统设计中，我们使用对称加密 [39]，确保节点间传输数据时敏感信息的机密性。对称加密是指使用同一密钥来加密和解密数据。这种方法通常用于通信双方共享密钥且需要确保通信私密性的场景。我们选择使用 Fernet 加密 [40],，这是一种对称加密的特定实现，采用 AES 加密算法、CBC 模式和 PKCS7 填充方案 [46–48]。与其他加密方法相比，Fernet 加密具有多项优势，包括简单、快速，因此适用于我们的系统设计。在我们的系统中，通信双方会在开始传输数据之前生成并共享 Fernet 加密密钥。该密钥用于加密数据，并将数据发送至目标节点，目标节点再使用相同的密钥解密数据。此过程可确保数据在传输过程中受到保护，且只有预期接收方才能访问。我们的系统还使用 Fernet 加密方案来保护主节点与工作节点之间发送的消息。消息使用唯一的 Fernet 加密密钥进行加密，确保消息在传输过程中的机密性。这样可以确保敏感信息（例如训练后的机器学习模型）在分布式训练过程中保持安全和机密。总之，我们在系统设计中使用对称加密，以确保节点间传输数据时敏感信息的机密性。我们选择 Fernet 加密方案，是因为它简单、快速；该方案用于保护主节点与工作节点之间发送的消息，以及传输中的数据。

2.2 数据预处理

在我们的联邦学习系统中，客户端在本地设备上执行数据预处理。此方法旨在尽可能减少通过网络传输的敏感数据量，从而增强联邦学习过程的隐私和安全性。预处理步骤可能包括特征选择、特征提取、归一化和数据增强。这些步骤使用 NumPy [15] 和 PyTorch [13]等 Python 库实现。我们对预处理函数进行了优化，以确保数据处理高效且可扩展。

特征选择和特征提取技术用于识别输入数据中最相关的特征，这有助于提高模型的准确率，同时降低计算开销。归一化技术用于将输入特征缩放到共同的范围，从而减小不同特征尺度对模型训练的影响。数据增强技术通过应用各种数据变换（例如旋转、翻转和裁剪）来生成更多训练数据。这有助于提高模型的鲁棒性和泛化能力。

2.3 联邦学习算法

在我们的系统中，联邦学习采用去中心化方式实现，多个节点协作训练一个全局模型。每轮开始时，全局模型会广播给所有参与节点，每个节点再使用自己的私有数据训练本地模型。随后，更新后的本地模型会发送到全局节点，由其使用各种数据解耦技术（如平均或加权平均）进行聚合。然后更新全局模型，并重复这一过程若干轮，直至全局模型收敛。

2.4 数据解耦技术

在我们的联邦学习系统中，我们采用数据解耦技术，使多个客户端能够进行分布式学习。具体而言，我们将客户端训练后的模型数据存储在数据库中，以便进行模型聚合，同时降低通信开销并提高系统的可扩展性。

为存储训练后的模型数据，我们采用一种数据存储技术，根据客户端节点、迭代和轮次为每个训练后的模型建立索引。这种技术使全局节点和各个客户端能够高效地检索模型，从而让每个客户端都能使用自己的数据进行本地模型训练。通过采用数据解耦技术和数据存储技术，我们的方法实现了分布式学习，同时保障客户端数据的隐私和安全。

2.5 通信协议

在我们的联邦学习系统中，我们实现了一种自定义通信协议，以促进客户端与服务器之间的通信。连接建立后，系统会向数据库中间件发送请求并检查流程状态。这种方法为联邦学习系统提供了流畅高效的通信流程。

为确保客户端数据的隐私和安全，我们实施了多种安全与隐私保护措施。传输过程中采用对称加密来保护客户端数据。此外，使用数据库中间件存储客户端数据，并促进客户端与服务器之间的通信。

2.6 数据库中间件

所提出的采用解耦数据管理的联邦学习框架，通过使用专用的数据管理服务，在训练过程中管理所有中间模型，从而将数据管理与联邦学习系统分离。DDFL 中的本地模型下载并使用全局模型后，会将其存储起来以供后续使用。本文考虑了六种不同的数据库系统方案，以建立 DDFL 的数据管理服务。

2.6.1 ***RabbitMQ*** 队列。 RabbitMQ [17], 是一种实现高级消息队列协议（AMQP）的开源消息代理软件。RabbitMQ 队列用于将消息存储在缓冲区中，直到可以对其进行处理。在 DDFL 中使用 RabbitMQ 队列，可以实现数据层解耦，并促进节点间的通信。

2.6.2 ***MongoDB*** 集合. MongoDB [16], 是一种广泛使用的开源 NoSQL 数据库，以 BSON（二进制 JSON）这一灵活格式存储数据。在 DDFL 中，MongoDB 集合用作集中式数据库，用于存储和共享数据。每个客户端都会创建一个集合，并与

主节点共享，而主节点会为每个节点创建一个全局模型集合。

2.6.3 安全文件传输（***SCP***）。 安全复制协议（SCP）[22], 是一种用于在计算机之间安全传输文件的网络协议。SCP 用于将训练好的模型从主节点传输到客户端节点。

2.6.4 ***Cassandra***。 Apache Cassandra [18],是一种分布式 NoSQL 数据库，旨在通过多台普通服务器处理大量数据。在 DDFL 中，Apache Cassandra 被用作去中心化数据库，用于存储和管理联邦学习中的所有模型。

2.6.5 ***Neo4j***。 Neo4j [24], 是一种将数据存储在节点和边中的图数据库。在 DDFL 中，Neo4j 用于存储中间模型并支持基于图的查询。

2.6.6 ***Postgres***关系数据库管理系统。 PostgreSQL [25],是一种广受欢迎的开源关系数据库管理系统（RDBMS），以其出色的一致性和可靠性而闻名。在 DDFL 中，PostgreSQL 用于存储中间模型，并支持复杂查询。

在 SCP [22] 和 RabbitMQ [17] 系统中，全局模型存储在磁盘上。相比之下，MongoDB 和 Cassandra 系统使用模型变量来存储全局模型，并将其保存在内存中，以便高效检索。这种方式可以降低访问模型时的延迟，并在联邦学习过程中更快地更新模型。然而，与 SCP 和 RabbitMQ 使用的磁盘存储方式相比，这需要更多的内存资源。存储方式的选择取决于联邦学习系统的具体需求，例如模型大小以及可用的内存和磁盘资源。

在所提出的数据解耦联邦学习（DDFL）框架中，数据管理系统在联邦学习（FL）的每个训练轮次中分为三个不同阶段运行：本地模型计算阶段、模型聚合阶段和模型评估阶段。在本地模型计算阶段，每个客户端节点使用自己的训练数据更新从主节点下载的全局模型，生成本地模型并将其发送回主节点进行聚合。在模型聚合阶段，主节点收集所有客户端节点的本地模型并将其聚合，以更新全局模型。最后，在模型评估阶段，主节点评估更新后的全局模型，以确定训练过程的进展。在这些阶段中，专用的数据管理服务在管理中间结果和最终结果方面发挥着至关重要的作用，它将数据管理与联邦学习系统解耦，从而提高训练过程的效率和隐私性。

2.7 本地模型计算

解耦数据管理解决方案 S 可从 MongoDB、RabbitMQ、SCP、Neo4j、Postgres 和 Apache Cassandra 中选择。在每个训练轮次中，每个客户端 $i \in [N]$ 执行以下步骤来计算本地模型：

- 将本地模型 M_i 初始化为全局模型 G;
- 将本地数据 D_i 加载到其本地存储中;
- 使用本地数据 D_i训练本地模型 M_i ;
- 使用密钥 k_i加密本地模型 M_i ; • 将加密本地模型 E_{M_i} 存储在 S中。

请注意，全局模型 G 在训练开始前已在主节点上初始化。

2.8 模型聚合

完成本地模型计算后，必须汇总所有本地模型，以便在服务器上更新全局模型，供下一轮训练使用。为此，首先将在每个客户端 $i \in [N]$ 上执行以下步骤：

- **在每个客户端节点上：**
  (1) 从 S 获取加密的本地模型更新 E_{M_i} ; (2) 使用密钥 k_i 解密本地模型更新 E_{M_i} : $M_i = \text{Decrypt}(E_{M_i}, k_i)$; (3) 加密本地模型更新 M_i 并将其发送至 S。
- **在主节点上：**
  (1) 从内存中加载所有模型更新 $M_1, M_2, \ldots\ldots, M_N$; (2) 汇总全局数据库中的模型更新，生成新的全局模型，即 $G = \text{Aggregate}(M_1, M_2, \ldots\ldots, M_N)$; (3) 使用密钥 k 加密新的全局模型 G : $E_G = \text{Encrypt}(G, k)$; (4) 将加密后的全局模型 E_G 存储在 S 中; (5) 与客户端节点共享新的全局模型 G。请注意，为保护客户端节点的隐私，客户端节点与主节点之间的所有通信均采用对称加密 [12] 进行加密。

2.9 模型评估

在联邦学习训练过程中，需要评估训练后的模型性能，以确保模型的有效性。客户端节点和主节点均可使用测试数据集评估模型。主节点可执行以下步骤，评估最新全局模型的性能：

(i) 从数据管理服务 S检索加密的最新全局模型 E_G ; (ii) 使用密钥 k解密全局模型 E_G : $G = \text{Decrypt}(E_G, k)$; (iii) 在测试数据集 T上评估全局模型 G ; (iv) 获取训练后全局模型的测试准确率 $\text{Accuracy}(G, T)$。

在我们提出的数据解耦联邦学习（DDFL）框架中，对联邦学习过程的中间结果和最终结果进行管理和存储，对于高效的模型检索和管理至关重要。为此，我们在数据库系统中将每个模型存储为由其参数及其他相关特征构成的集合。具体而言，我们采用包含四列的模式，以便高效管理模型。这些列如下：

(1) 轮次： 此列存储每个模型对应的联邦训练过程轮次编号。由于联邦学习过程包含多个训练轮次，此列可作为每个模型的唯一标识符，帮助将其与数据库中的其他模型区分开来。

(2) **模型：** 模型列存储模型的参数，这些参数对于在后续训练轮次中复用和更新模型至关重要。此列支持从数据库中检索特定模型及其参数，以便继续训练过程，或使用新的训练数据更新模型。

(3) **准确率：** 此列存储模型的测试准确率，为模型提供评估指标。该指标可用于比较不同模型的有效性，或从数据库中存储的所有模型里找出性能最佳的模型。

(4) **时间：** 此列存储完成相应训练轮次所需的时间。此列作为训练过程的性能指标，可用于找出耗时较长的训练轮次并对其进行优化，以提升性能。

使用此模式在数据库系统中存储和组织所有模型后，客户端节点和主节点便可在联邦学习过程中高效地管理和检索模型。这使我们能够高效地跟踪模型性能、找出性能最佳的模型，并利用存储的模型恢复或更新训练过程。

3 评估

3.1 实验设置

在本研究中，我们使用三个标准基准数据集评估所提出的 DDFL 框架的性能：CIFAR10 [14],Fashion-MNIST (FMNIST) [21], 和街景门牌号数据集 (SVHN) [20]。为开展实验，我们在 CloudLab [1], 上使用了一个 11节点集群，其中一个节点作为主节点，其余节点作为客户端节点。主节点和客户端节点的硬件配置包括一颗主频为 2.60GHz 的英特尔至强 E5-2690 v4 处理器、377GB 内存以及 2x 以太网控制器 X710（用于 10GbE SFP）+。值得注意的是，所有节点共用一个网络接口，以便彼此通信。我们的实验使用 PyTorch 作为深度学习框架实现。通过开展这些实验，我们旨在评估所提出的框架在联邦学习场景中使用各种数据集时的效率和有效性。为评估 DDFL 框架的性能，我们使用六种不同的数据库系统部署了该系统：RabbitMQ [17],MongoDB [16],安全文件传输 (SCP) [22], Apache Cassandra [18], Neo4j [24], Postgres [25]。本实验的目标是比较这些系统在若干重要指标上的性能，例如训练时间、准确率和训练损失。通过使用不同的数据库系统测试 DDFL 框架，我们旨在确定哪种系统与该框架配合最佳，并能在上述指标方面取得最佳结果。

为了评估我们提出的 DDFL 方法的有效性，我们在三个不同的数据集上进行了实验，分别是 Fashion-MNIST、SVHN 和 CIFAR-10。我们报告了微调超参数 [71]和不使用超参数的实验结果，指标为全局模型的准确率。

源代码已发布在 https://github.com/jahanxb/flcode。

3.2 联邦学习中不同数据库层的可用性

联邦学习系统中数据库层的选择会对其开发和适应性产生重大影响。在本研究中，我们探讨了将多种流行的数据库系统集成到 DDFL 框架中的方法，包括 Apache Cassandra、MongoDB、PostgreSQL、Neo4j、SCP 和 RabbitMQ 队列。

各数据库系统的集成过程在难度、开发工作量以及对我们系统的适应性方面各不相同。对于支持我们所用编程语言和框架的成熟连接器或插件的数据库系统，集成起来更容易，所需的开发工作量也更少。例如，MongoDB 和 PostgreSQL 拥有成熟的库，可与我们的系统轻松集成，而且安装和配置相对简单。

然而，将 Apache Cassandra 和 Neo4j 集成到我们的系统中需要投入更多开发工作，因为它们的连接器不够成熟，而且需要定制开发查询和数据检索方法。由于这些系统还存在额外的依赖项和配置要求，因此安装和配置也需要更多时间。

SCP 和 RabbitMQ 队列在集成方面带来了重大挑战，因为要将它们集成到我们的系统中，需要进行大量定制开发。SCP 需要定制封装器来支持其接口，这需要投入大量开发工作。RabbitMQ 队列还需要额外配置消息队列及其处理方式，使集成过程更加复杂。

3.2.1 在联邦学习中采用数据库系统的易用性。 设计联邦学习系统时，采用不同数据库系统的难易程度也是一项重要考量。我们根据学习难易度以及与 DDFL 系统的集成难易度，对 MongoDB、Postgres、Neo4j、SCP、Cassandra 和 RabbitMQ 等热门数据库系统进行了评估。结果显示，MongoDB 和 Postgres 等系统相对容易学习和集成，所需代码不到 200 行。而 Neo4j 和 RabbitMQ 等其他系统则更难学习，完全集成需要 500 行以上的代码。

除了易于集成之外，各系统社区的规模和活跃程度也会影响其普及。MongoDB 和 Postgres 等系统拥有规模庞大且活跃的社区，能够为开发者提供支持和资源。相比之下，Neo4j 和 RabbitMQ 等其他系统的社区规模较小，因此在将它们集成到联邦学习系统时，可能更难找到资源和支持。

总之，联邦学习系统中数据层的易集成性和适应性可能存在显著差异。开发者在选择数据库系统时，应仔细考虑其联邦学习系统的具体要求。对于采用的编程语言和框架已有成熟连接器和插件的系统，更容易集成，所需的开发工作也更少。对于缺少成熟连接器和插件的系统，则需要定制开发，这可能显著增加开发时间和工作量。采用数据库系统时，各系统所属社区的规模和活跃度也是需要考虑的重要因素。

3.3 测试准确率

3.3.1 未经微调超参数的测试准确率。我们展示了所提出的数据解耦联邦学习（DDFL）系统在未经超参数微调时的测试准确率结果。我们使用三个不同的数据集评估系统性能：CIFAR-10、Fashion-MNIST 和 SVHN。实验旨在将 DDFL 系统在不同数据库系统（包括 MongoDB、SCP、Apache Cassandra 和 RabbitMQ 队列）上的准确率，与集中式机器学习方法进行比较。在集中式方法中，模型在单节点上训练，并在同一台机器上模拟客户端 [8, 9, 38, 60]。实验使用所有分布式节点的基线参数进行，未进行任何微调。

实验的主要目的是评估数据解耦方案在提升系统性能方面的有效性。我们将 DDFL 系统的测试准确率与集中式机器学习方法进行了比较，后者为评估提供了基准。实验结果揭示了 DDFL 系统和数据解耦方案的性能。

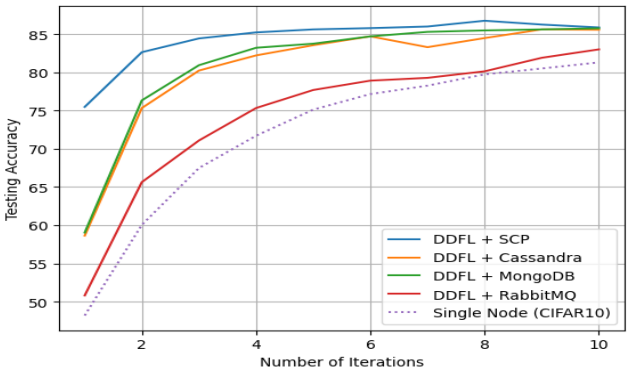


图3：在 CIFAR-10 数据集上，使用 DDFL 和不同类型的数据库训练的模型的测试准确率，与未使用经过微调的超参数、在单节点上训练的基线模型进行比较。

如图3、图4和图5所示，经过 10 轮训练后，我们的 DDFL 系统达到的测试准确率高于单节点基准。具体而言，在 CIFAR-10 数据集上，经过 10 轮训练后，DDFL 系统的测试准确率达到 85%，而基准准确率为 81%。在 Fashion-MNIST 数据集上，经过 10 轮训练后，DDFL 系统的测试准确率达到 92%，而基准准确率为 89%。在 SVHN 数据集上，经过 10 轮训练后，DDFL 系统的测试准确率达到 95%，而基准准确率为 94%。

我们还观察到，各数据库系统在不同数据集上的性能各不相同。在 CIFAR-10 数据集上，MongoDB 和 SCP 表现最佳，经过 10 轮训练后，测试准确率分别达到 85.56% 和 85.86%。Apache Cassandra 在 DDFL 系统中的测试准确率与其他数据库系统相当，但在不同迭代中的性能较低；不过，它在 Fashion-MNIST 和 SVHN 数据集上的表现更好，经过 10 轮训练后，测试准确率分别达到 92% 和 95.75%。RabbitMQ 队列在所有数据集上的表现中等，经过 10 轮训练后，在 CIFAR-10、Fashion-MNIST 和 SVHN 数据集上的测试准确率分别为 83%、90% 和 94%。

我们进一步比较了采用不同数据库系统的 DDFL 系统在第 1、5 和 10 次迭代时的测试准确率与单节点基线的差异。如图 3、4 和 5 所示，DDFL 系统在所有迭代中始终优于单节点基线。此外，我们观察到，随着迭代次数增加，DDFL 系统与单节点基线之间的测试准确率差异也愈发显著。例如，在 CIFAR-10 数据集上，DDFL 系统在第 1 次迭代时的测试准确率为 50%–58%，而基线准确率为 48%。然而，在第 10 次迭代时，DDFL 系统的测试准确率达到 83%–85%，而基线准确率为 81%。

这些结果表明，数据解耦方案的选择会对联邦学习系统的性能产生重大影响，而且这种影响可能因所用数据集的特征而异。选择数据解耦方案时，必须考虑联邦学习系统的具体需求和限制。例如，通信协议和系统的可扩展性是需要纳入考虑的关键因素。此外，系统架构及其针对分布式环境的优化，也可能在提升系统性能方面发挥重要作用。

关于基线模型与不同数据库系统之间的性能差异，我们发现，在所有三个数据集上，经过 10 轮训练后，DDFL系统的表现始终优于单节点基线模型。例如，在 CIFAR-10 上，经过 10 轮训练后，DDFL系统的测试准确率约为 81%，而基线模型约为 74%。在 Fashion-MNIST 上，DDFL系统经过 10 轮训练后的测试准确率约为 90%，而基线模型约为 83%。在 SVHN 上，DDFL系统经过 10 轮训练后的测试准确率约为 85%，而基线模型约为 77%。

此外，我们发现各数据库系统的性能因数据集而异。在 CIFAR-10 上，MongoDB 和 SCP 的表现相近，在四种数据库系统中测试准确率最高。尽管 Apache Cassandra 在某些迭代中的性能较低，但该系统仍取得了相当的准确率。在 Fashion-MNIST 和 SVHN 上，MongoDB 和 SCP 再次表现最佳，Apache Cassandra 的性能则高于其他系统。RabbitMQ 队列在所有数据集上的表现中等。这些性能差异凸显了为特定联邦学习系统和数据集选择合适的数据解耦方案的重要性。

3.3.2 使用微调超参数的测试准确率。本节将介绍微调实验的结果。在实验中，我们优化了模型的超参数，以提高数据解耦联邦学习（DDFL）系统的准确率。为进一步验证结果，我们将评估范围扩大到另外两个数据库系统：Neo4j 和 PostgreSQL。我们在 CIFAR-10、Fashion-MNIST、

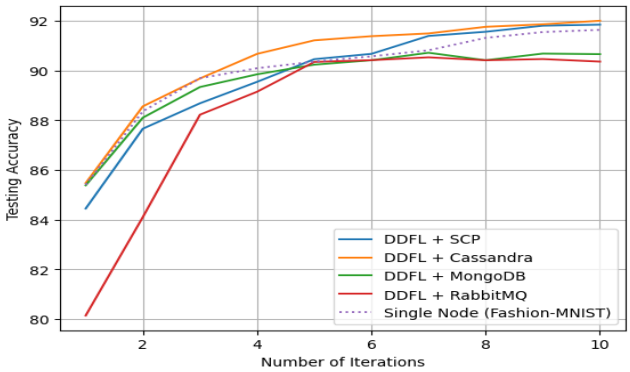


图 4：使用 DDFL 和不同类型的数据库在 Fashion-MNIST 数据集上训练的模型的测试准确率，与在单节点上训练且未微调超参数的基线模型进行比较。

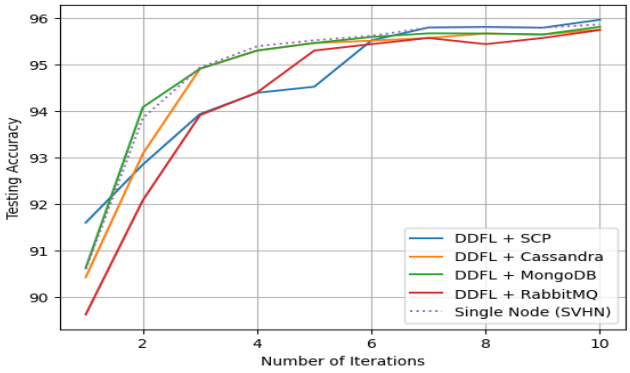


图 5：使用 DDFL 和不同类型的数据库在 SVHN 数据集上训练的模型的测试准确率，与在单节点上训练且未微调超参数的基线模型进行比较。

和 SVHN 数据集上进行了实验，采用与上一节（3.3.1）相同的模型架构和训练流程。我们使用相同的超参数对所有节点上的模型进行了微调，并将微调后的模型性能与基线单节点模型进行了比较。我们将微调后的 DDFL 系统的测试准确率与集中式机器学习方法进行了比较。本次评估旨在考察模型使用最优超参数进行微调时，DDFL 系统在不同数据库系统下的性能。

图 6 展示了使用 DDFL 和不同数据库系统在 CIFAR-10 数据集上训练的模型的测试准确率，并与在单节点上训练的基线模型进行比较。我们观察到，微调后的模型准确率高于单节点基线模型，平均准确率提升了 2.23%。我们还观察到，DDFL 在所有平台上的性能表现一致，准确率没有显著差异。

同样，图 7 展示了使用 DDFL 在不同数据库系统上训练的模型在 Fashion-MNIST 数据集上的测试准确率，并与在单节点上训练的基线模型进行比较。我们观察到，微调后的模型平均准确率提高了 1.84%，且

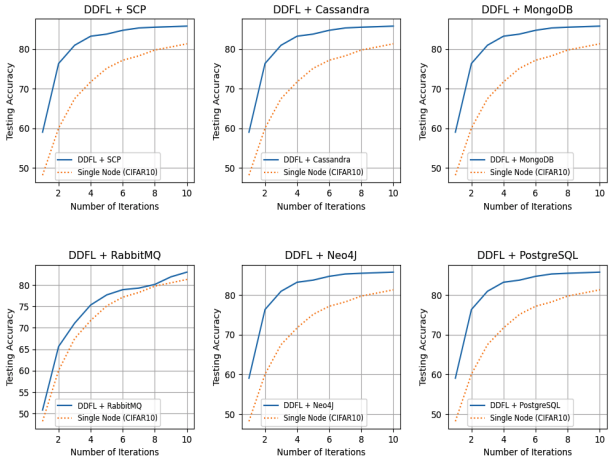


图 6：使用 DDFL 和不同类型的数据库在 CIFAR-10 数据集上训练的模型的测试准确率，与使用微调超参数在单节点上训练的基线模型进行比较。

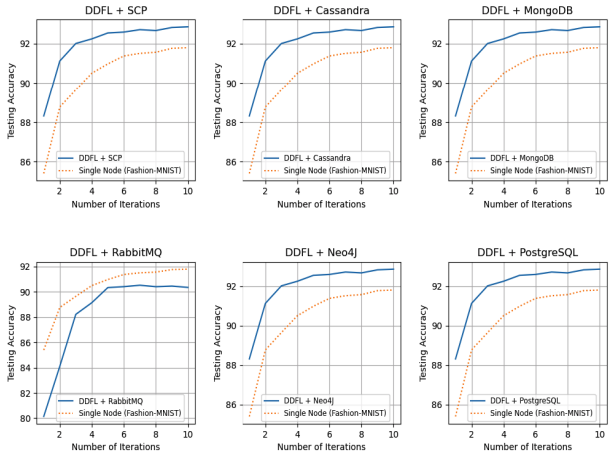


图 7：使用 DDFL 和不同类型的数据库在 Fashion-MNIST 数据集上训练的模型的测试准确率，与使用微调超参数在单节点上训练的基线模型进行比较。

在所有平台上性能再次保持一致。

最后，图 8 展示了使用 DDFL 和不同数据库系统在 SVHN 数据集上训练的模型的测试准确率，并与在单节点上训练的基线模型进行比较。微调后的模型平均准确率比单节点基线模型提高了 1.87%，且在所有平台上均表现稳定。

这些结果证明，DDFL 框架能够有效支持使用多个数据库系统进行协作式深度学习，而采用微调也有助于提升模型的性能。

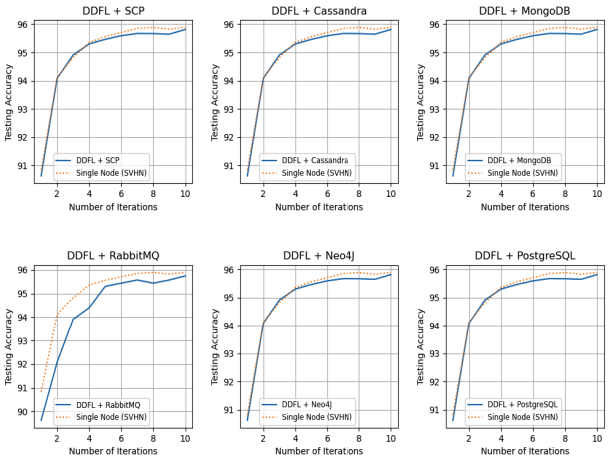


图 8：在 SVHN 数据集上，使用 DDFL 和不同类型的数据库训练的模型的测试准确率，与使用微调超参数在单节点上训练的基线模型进行比较。

3.4 训练时间

3.4.1 未经微调的超参数下的训练时间比较。结果表明，模型的训练时间因所使用的数据库系统而异。例如，与 MongoDB、RabbitMQ 和 Apache Cassandra 等其他数据库系统相比，SCP 训练系统所需的训练时间显著更长。MongoDB、RabbitMQ 和 Apache Cassandra 等其他系统训练模型所需的时间则较短。

图 9–11 的结果表明，在三个数据集（CIFAR-10、Fashion-MNIST 和 SVHN）上，集中式数据库系统 MongoDB 的训练时间最短。总体而言，这些结果表明，集中式数据库系统 MongoDB 能够有效缩短联邦学习系统的训练时间，这对现实世界应用至关重要。

3.4.2 使用微调超参数进行训练时间比较。 由于预训练模型本身较为复杂、微调数据集规模较大，且需要更新的参数较多，因此针对特定任务微调预训练深度学习模型，通常比从头开始训练耗时更长。在我们的研究中，与从头开始训练相比，在三个数据集（CIFAR-10、Fashion-MNIST 和 SVHN）上微调深度学习模型时，训练时间显著增加。如图 12、13 和 14 所示，DDFL 使用不同数据库系统在 CIFAR-10、Fashion-MNIST 和 SVHN 数据集上的训练时间分别为：微调深度学习模型耗时 47 至 55 分钟，具体时长取决于数据集；而从头开始训练约需 35–40 分钟。微调实验中观察到的较长训练时间，可归因于微调过程的额外复杂性；该过程需要进行更多计算，以针对新任务优化预训练模型的权重。此外，微调需要更多计算资源和更多训练迭代，因此训练更加耗时。尽管训练时间更长，已有研究表明，与从头开始训练相比，微调能够带来更好的性能，因为它能够

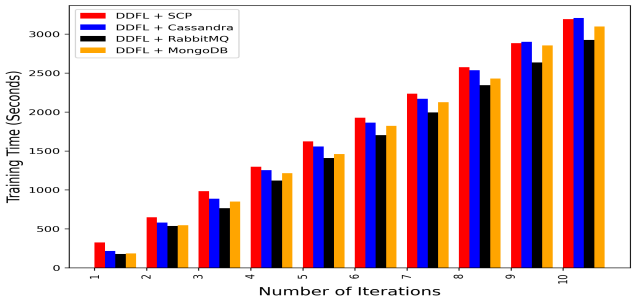


图9：在未微调超参数的情况下，DDFL 在不同数据库系统上的 CIFAR-10 数据集训练时间。

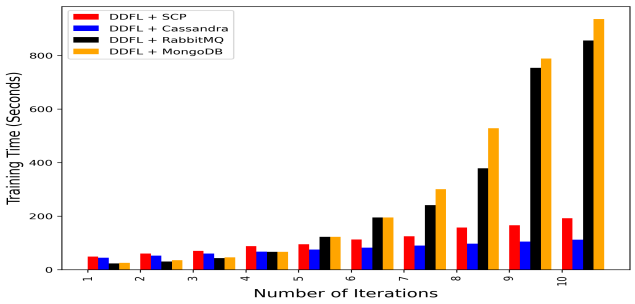


图10：在未微调超参数的情况下，DDFL 在不同数据库系统上的 Fashion-MNIST 数据集训练时间。

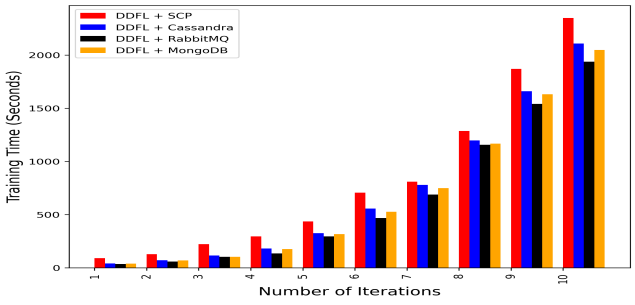


图11：在未微调超参数的情况下，DDFL 在不同数据库系统上的 SVHN 数据集训练时间。

较大的数据集上预训练模型所学知识的优势。

这一观察与以往研究的报告一致：与从头开始训练相比，微调所需的训练时间更长 [35–37]。训练时间增加也归因于深度学习模型架构的复杂性，以及微调所用数据量较大。不过，需要注意的是，较长的训练时间是为了换取通过微调预训练模型而获得更佳性能所付出的代价。正如我们的研究所示，在全部三个数据集上，微调预训练模型的性能均优于从头开始训练。

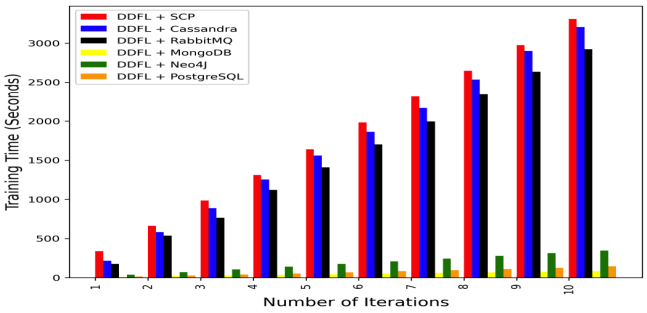


图 12：在 CIFAR-10 数据集上使用微调超参数时，DDFL 在不同数据库系统上的训练时间。

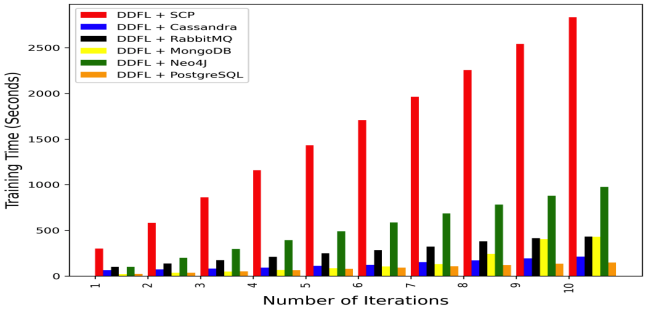


图 13：在 Fashion-MNIST 数据集上使用微调超参数时，DDFL 在不同数据库系统上的训练时间。

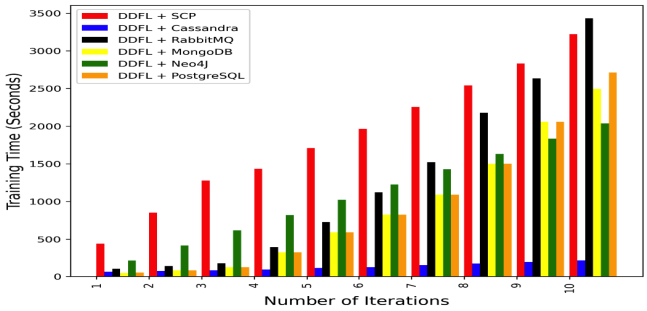


图 14：在 SVHN 数据集上使用微调超参数时，DDFL 在不同数据库系统上的训练时间。

因此，在必须达到最佳性能的应用中，微调所需的较长训练时间是合理的。

3.5 查询性能

在本研究中，我们通过比较 MongoDB、Apache Cassandra、PostgreSQL、Neo4j 和 RabbitMQ 队列，评估了 DDFL 系统中不同数据库系统的查询性能。我们在每个数据库系统上执行了图 15 所示的示例查询，以检索特定客户端的测试数据，并测量查询执行时间。实验采用了三个数据集：CIFAR-10、FMNIST 和 SVHN。

```
-- MongoDB query
CONNECT TO MONGODB SERVER AT localhost:27017
USE DATABASE test_db
RESULT = SELECT * FROM test_collection WHERE field = 'value'

-- Neo4j query
CONNECT TO Neo4j SERVER AT localhost:7687 WITH USERNAME 'neo4j' AND PASSWORD 'password'
WITH SESSION
RESULT = MATCH (n {field: 'value'}) RETURN n

-- PostgreSQL query
CONNECT TO POSTGRESQL SERVER AT localhost:5432 WITH USERNAME 'user'
AND PASSWORD 'password' AND DATABASE 'test_db'
WITH CURSOR
EXECUTE 'SELECT * FROM test_table WHERE field = 'value''
RESULT = FETCH ROWS

-- RabbitMQ query
CONNECT TO RABBITMQ SERVER AT localhost
DECLARE QUEUE test_queue
GET MESSAGE FROM QUEUE test_queue WHERE field = 'value'

-- Cassandra query
CONNECT TO CASSANDRA SERVER AT localhost
USE KEYSPACE test_keyspace
RESULT = SELECT * FROM test_table WHERE field = 'value'
```

图 15：示例数据库查询

我们的研究结果表明，与 Neo4j 和 RabbitMQ 队列等分布式数据库系统相比，MongoDB 等集中式数据库系统在联邦学习系统中具有更好的查询性能。在所有数据集上，MongoDB 的查询执行时间都最短：CIFAR-10 的平均查询执行时间为 0.058 毫秒，FMNIST 为 0.059 毫秒，SVHN 为 0.056 毫秒。相比之下，Neo4j 和 RabbitMQ 队列的查询执行时间明显更长。例如，Neo4j 在 CIFAR-10、FMNIST 和 SVHN 上的平均查询执行时间分别为 117.787 毫秒、97.986 毫秒和 191.491 毫秒。需要注意的是，查询性能可能受多种因素影响，例如数据集大小、查询复杂度以及联邦学习系统中的节点数量。我们设计了一个示例查询，用于检索特定客户端的测试数据，并在表 1 中以毫秒为单位报告查询执行时间。本研究对查询性能至关重要的现实联邦学习应用具有实际意义。不过，仍需进一步研究不同数据库系统在联邦学习系统中的可扩展性和稳健性。

我们的实验表明，与 Postgres、Apache Cassandra、Neo4j 等其他数据库系统相比，MongoDB 这类集中式数据库系统在联邦学习系统中可能具有更好的查询性能。我们的研究通过评估 DDFL 系统中不同数据库系统的查询性能，为联邦学习系统的持续研究作出了贡献。

3.6 通信成本

通信成本是联邦学习系统的一个关键方面，因为它决定了模型训练过程中必须在客户端与服务器之间传输的数据量。为了评估我们的联邦学习系统的通信成本，我们针对不同数量的客户端和不同的批次大小进行了实验。

表2所示的结果表明，通信成本随客户端数量和批量大小的增加而线性增长。与传统的集中式机器学习系统相比，我们的联邦学习系统由于需要将更新后的模型参数从客户端传输到服务器，因此通信成本更高。不过，我们注意到，我们的系统在数据隐私和安全方面具有显著

表 1：DDFL 系统中不同数据库系统在 CIFAR-10、FMNIST 和 SVHN 数据集上的查询性能。查询执行时间以毫秒为单位。

数据库	MongoDB	SCP	RabbitMQ	Cassandra	Postgres	Neo4j
CIFAR10	0.059	227.160	0.098	35.438	279.426	117.787
FMNIST	0.059	244.847	0.129	132.974	390.992	97.986
SVHN	0.056	282.027	0.239	358.901	693.879	191.491

表2：单个客户端单次迭代的通信成本、模型大小、值的数量和通信时间

数据集	模型大小（MB）	数值数量	通信时间（毫秒）
CIFAR-10	0.061	53,761	0.488
FMNIST	0.184	166,370	1.472
SVHN	0.382	343,175	3.056

优势，因为敏感数据保留在客户端一侧，不会传输到服务器。

这些结果与先前的研究一致，例如 McMahan 等人 [8], FedML [6], 以及 Bonawitz 等人 [27]的研究。他们的研究也报告了类似趋势：随着批次大小和客户端数量的变化，通信成本也会随之变化。

总体而言，我们的实验表明，在联邦学习系统中使用数据分区技术对于降低通信成本至关重要。我们观察到，采用数据分区技术能够显著降低通信成本，因为每轮传输的数据块更小。这一发现对于开发高效且可扩展的联邦学习系统至关重要，此类系统能够处理大量客户端，同时维护敏感数据的隐私和安全。

3.7 可扩展性

可扩展性是分布式系统的一个关键方面，对于大量节点参与训练过程的联邦学习系统尤其如此。为了提升我们的研究系统的可扩展性，我们采用了多种技术，例如使用数据库中间件解耦客户端与服务器之间的数据。通过将客户端的数据存储在数据库中，服务器可以在不直接与客户端通信的情况下访问数据以进行模型训练，从而降低通信成本和网络开销。此外，我们采用了联邦学习的去中心化方法：每个节点使用自己的私有数据训练本地模型。随后，全局节点会聚合这些本地模型，以更新全局模型。这种方法将训练过程分散到各个节点，并能够高效扩展以支持大量客户端。

我们通过使用不同数量的节点和不同数据规模进行实验，评估了系统的可扩展性。表3中的结果表明，我们的系统能够高效扩展至大量节点，同时不会出现明显的性能下降。我们观察到，训练时间随节点数量和数据规模呈线性增长，这表明我们的系统能够高效处理大规模联邦学习任务。

总体而言，我们的研究系统展现出出色的可扩展性，这对于大规模联邦学习应用至关重要。将去中心化联邦学习与数据库中间件方法相结合，为扩展联邦学习系统、满足现实世界应用的需求提供了很有前景的解决方案。

4 相关工作

4.1 分布式机器学习

联邦学习是一种前景广阔的分布式机器学习 [10,65, 66, 70]技术，使多个客户端能够协作训练模型，而无需共享其私有数据。由于有望促进隐私保护和可扩展的数据分析，这项技术受到了研究社区和业界的广泛关注。近年来，研究人员探索了联邦学习的多个方面，包括隐私与安全、通信效率和可扩展性。

已有大量关于联邦学习的研究成果，重点探讨了这项技术的不同方面。例如，Bonawitz 等人 [10] 提出了一种安全且保护隐私的联邦学习框架，而 Konecny 等人 [9] 则提出了一种联邦优化方法。Yang 等人 [29]研究了联邦学习的通信效率，而 Jeong 等人 [50]提出了一种降低联邦学习通信成本的方案。此外，研究人员还探索了联邦学习的数据解耦技术，包括数据库中间件 [51],、数据分区 [3],以及隐私保护数据聚合 [49]。这些研究增进了人们对联邦学习及其在医疗保健、金融和智慧城市等不同领域潜在应用的理解。

4.2 通信高效的联邦学习

联邦平均是 McMahan 等人 [8]提出的一种通信高效的联邦学习方法。它涉及在客户端节点和服务器之间交换模型更新，从而降低通信开销。这种方法能够在不共享数据的情况下进行分布式模型训练，因此有望解决数据共享中的隐私和安全问题。

Konecny 等人 [9]提出了稀疏化、量化和差分隐私，作为降低联邦学习中通信成本的策略。实验结果表明，这些策略在维持模型准确率方面效果显著，但仍需进一步研究，以确定它们对不同数据类型的适用性，并优化通信成本与模型准确率之间的权衡。

表3：不同数据库系统在不同客户端数量下的可扩展性结果（秒）

客户端数量	SCP	Cassandra	RabbitMQ	Neo4j	PostgreSQL	MongoDB
FMNIST						
2	409.649	152.232	138.218	199.319	37.037	32.032
4	1058.1738	201.321	249.49	527.847	65.15	40.040
6	1409.2329	286.446	379.619	319.519	108.148	53.053
8	2303.3823	312.512	414.654	393.633	135.215	67.17
CIFAR10						
2	2059.3419	493.813	537.857	71.111	29.029	25.025
4	2328.3848	1804.304	1059.1739	140.220	68.18	52.052
6	2743.4543	2295.3815	2044.344	278.438	100.140	65.15
8	2719.4519	2535.4215	2635.4355	312.512	135.215	97.137
SVHN						
2	1433.2353	123.23	1124.1844	1224.2024	823.1343	823.1343
4	2255.3735	152.232	1522.2522	1426.2346	1089.189	1089.189
6	2541.4221	171.251	2179.3619	1630.271	1504.254	1504.254
8	2831.4711	192.312	2634.4354	1832.3032	2058.3418	2058.3418

Jhun等人 [53] 提出了一种通信高效的联邦学习方法，该方法采用一种名为自适应联邦量化的新型压缩技术。该方法根据每个客户端本地模型的梯度稀疏度调整压缩率，以获得更好的压缩性能。

Rothchild 等人提出的 FetchSGD 算法 [54]使用计数草图进行压缩，并整合多个工作节点的更新。该算法通过将动量和误差累积转移到中央聚合器，解决了客户端参与稀疏的问题，实现了较高的压缩率和收敛性。

Hamer 等人 [52] 提出了一种名为 FedBoost 的通信高效联邦学习方法，该方法使用提升算法选择部分客户端节点与服务器通信。与传统联邦学习方法相比，这种方法降低了通信成本，并实现了更高的模型准确率。

Sai 等人 [61] 提出了 SCAFFOLD，这是一种针对异构客户端数据场景设计的联邦学习算法。该算法采用控制变量来减轻客户端本地更新之间的漂移，与联邦平均相比，能够提高收敛速度和通信效率。

Wang 等人 [62] 提出了一种面向资源受限的边缘计算系统的自适应联邦学习方法。数据分布在多个节点上，无需将原始数据发送到集中位置。该方法使用控制算法来平衡本地更新与全局参数聚合，从而在给定资源预算下最小化损失函数。使用真实数据集开展的大量实验表明，该方法在各种机器学习模型和数据分布下均表现良好。

4.3 联邦学习中的安全与隐私

Bonawitz [10] 提出了 FATE，这是一种隐私保护联邦学习框架，通过可信聚合器计算全局模型，同时不泄露客户端的私有数据。该框架提供安全聚合等安全和隐私功能，以及差分隐私和同态加密，以在模型训练期间保护客户端数据。

Liu 等人 [67]提出了一个用于联邦学习的基于 SQL 的训练数据调试框架，该框架能够自动从训练数据中移除标签错误，以修正模型的异常行为。作者解决了相关技术挑战，使该框架兼具安全性、效率和准确性，并提出了 Frog——一种新颖的框架，其性能优于他们之前的解决方案。通过理论分析以及在真实世界数据集上开展的大量实验，验证了 Frog 的有效性。

为确保联邦学习的隐私和安全，研究者提出了差分隐私、安全多方计算和同态加密等多种技术。差分隐私会在数据与服务器共享之前向其中添加随机噪声，而安全多方计算则能在不泄露输入的情况下实现协同计算。同态加密支持对加密数据进行计算，从而保护数据隐私。这些技术已应用于联邦学习，以确保模型训练过程中客户端的数据隐私和安全 [11, 23, 55, 56]。

4.4 联邦学习中的数据解耦技术

为提高联邦学习系统的灵活性和可扩展性，研究者提出了数据解耦技术，将数据管理子系统与联邦学习系统分离。这些技术可以减少通信开销，并使系统能够扩展到更多客户端。研究者已提出多种方法，且多项研究展示了这些方法的潜力 [23, 29, 44, 57–59]。

*4.4.1*联邦学习中的半监督学习数据解耦技术。 Kairouz 等人 [3] 提出了一种基于半监督学习的数据解耦技术，以提升联邦学习系统的性能。尽管他们的

方法提高了准确率并降低了通信成本，但仅适用于客户端节点上有大量未标记数据的场景。如果数据全部已标记，这一局限性可能会造成影响。此外，他们的结果是在特定实验设置下得出的，其方法能否推广到其他联邦学习场景尚不确定。我们提出的方法与 Kairouz 等人的 [3] 半监督学习数据解耦技术不同：我们提出了一种新方法，利用标签传播来使用每个客户端节点上的全部数据。我们的方法适用于数据全部已标记或部分已标记的场景，在现实世界应用中提供了更大的灵活性。此外，我们的方法通过减少客户端与服务器之间传输的数据量，提升了可扩展性。

4.4.2 联邦学习中的本地 ***SGD*** 数据解耦方法。 Lian 等人 [41] 提出了本地 SGD，将其作为一种数据解耦方法来优化联邦学习系统。本地 SGD 使客户端能够执行本地随机梯度下降，同时由服务器聚合更新后的模型，从而降低通信成本。然而，在实际应用中，其局限性仍需进一步探讨，例如计算资源有限的客户端所面临的计算开销，以及对同构模型的假设。

我们提出的方法与本地 SGD 不同：它通过标签传播来利用每个客户端节点上的所有可用数据；只发送传播后的标签，而非整个模型，从而降低通信成本；并且不要求客户端在每次迭代时执行本地 SGD。这些差异使我们的方法更适用于计算资源有限或采用异构客户端模型的场景。

*4.4.3*联邦学习中的分布式数据库。 Liu 等人 [34] 提出了一种联邦学习方法，该方法利用分布式数据库 Hyperledger Fabric 管理客户端的数据。通过将数据管理子系统与联邦学习系统解耦，该方法使系统能够扩展到大量客户端。尽管实验结果表明，该方法能够有效降低通信成本并提高可扩展性，但其适用范围可能局限于分布式数据库，而且还可能增加复杂性和维护成本。

我们提出的方法与 Liu 等人的 [34] 分布式数据库方法不同，不依赖分布式数据库来管理客户端数据。相反，我们的标签传播方法在联邦学习系统内部运行，从而减少了对额外基础设施的需求。此外，我们的方法只发送传播后的标签，而非完整的客户端数据，因此可以通过降低通信成本来提高可扩展性。

近年来，联邦学习（FL）因其有望解决集中式机器学习面临的挑战而备受关注。该领域的研究聚焦于优化联邦学习算法，以提升效率、安全性，和可扩展性。一些研究 [9, 29–31, 42] 提出了保护隐私的技术，另一些则探索了通信高效的方法 [32, 33]，例如采用压缩技术和自适应正则化。这些研究展示了数据解耦技术在联邦学习中的潜力，以及可用于提升其效率、安全性和可扩展性的各种方法与技术。

本研究通过全面评估借助数据库中间件构建的联邦学习系统，为联邦学习（FL）数据解耦技术研究作出了贡献。通过发挥数据库中间件的能力，本研究为分布式环境中的联邦学习提供了灵活且可扩展的解决方案，并对实际场景中的联邦学习系统设计具有启示意义。我们使用真实世界数据集开展评估，考察数据解耦对通信成本、训练时间、可扩展性和准确率等各项性能指标的影响，且未使用数据分区和数据联邦技术。

5 结论

本文提出了一种名为 DDFL 的数据解耦框架，适用于联邦学习（FL）系统，使客户端能够根据特定数据子系统定制应用，并对模型进行有效查询。DDFL 不仅为比较不同的联邦学习数据库子系统提供了一个通用且公平的测试平台，也是构建全新联邦学习生态系统的核心基础设施，可支持任意数据管理服务。我们实现了 DDFL，并将其与多种数据管理系统集成，包括 MongoDB、SCP、Cassandra、RabbitMQ、Neo4j 和 Postgres。在 CIFAR-10、Fashion-MNIST 和 SVHN 数据集上进行的大量实验表明，所有这些数据库系统都能达到与标准联邦学习系统相当的高准确率（即正确性），但在训练时间和通信开销等其他指标上存在差异。这些差异很大程度上取决于具体应用，仍有待进一步研究。

更广泛地说，我们的结果表明，不同数据库系统在量化层面上会对联邦学习系统的整体性能产生重大影响，并为现有方法的优势与不足带来了全新的见解。本研究开发的 DDFL 框架为评估现有方法提供了一种新途径。我们希望本研究能够推动联邦学习生态系统中数据库服务领域的新一轮系统研究。

参考文献

[1] Duplyakin, D., Ricci, R., Maricq,A., Wong, G.,Duerig, J.,Eide, E.,Stoller, L., Hibler, M., Johnson, D., Webb, K., Akella, A., Wang, K., Ricart, G., Landweber, L., Elliott, C., Zink, M., Cecchet, E., Kar, S., & Mishra, P. (2019). CloudLab 的设计与运行。载于《USENIX年度技术会议（ATC）论文集》（第 1 14 19 - 页）。 [2] 2018 https://www.flux.utah.edu/paper/duplyakin-atc Nishio, T. and Yonetani, R. ( ). 移动边缘异构资源下的联邦学习客户端选择。CoRR，abs/1804.08333。可访问：http://arxiv.org/abs/1804.08333 [访问日期： 2023]年3月14日。[3] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ...（2021）。联邦学习的进展与开放问题。《机器学习基础与趋势》，14(1-2)，1-210。[4] Wang, L. and Zhao, D.（2022）。面向隐私保护科学计算的实用联邦学习基础设施。载于第三届科学应用人工智能与机器学习研讨会（AI4S）论文集，与高性能计算、网络、存储与分析国际会议（SC）同期举行。[5] PySyft。*https://github.com/OpenMined/PySyft*，访问日期：2022年。

[6] FedML。https://fedml.ai/。访问日期：2022年。[7] TensorFlow Federated, https://github.com/tensorflow/federated，访问日期：2022年。[8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas。（2017）。基于去中心化数据的深度网络通信高效学习。载于第20届国际人工智能与统计学会议（AISTAT）论文集，arti Singh 和 Xiaojin (Jerry) Zhu（编），《机器学习研究论文集》第54卷，第1273–1282页，PMLR。[9]J. Konecny, H. B. McMahan, D. Ramage, and N. A. Smith，“联邦学习：提升通信效率的策略”，arXiv 预印本 arXiv:1610.05492，2016年。[10] Bonawitz, K., McMahan, H. B., McNamara, E. N. P., Recht, B. A., & Woodworth, A. D. K.（2019）。迈向大规模联邦学习：系统设计。arXiv 预印本 arXiv:1902.01046。[11] Shokri, R., Stronati, M., Song, C., & Shmatikov, V.（2015）。隐私保护深度学习。arXiv 预印本 arXiv:1510.06772。[12] B. Sahu, K. K. P. Singh, and A. Ghosh。“去中心化联邦学习：综述。”arXiv 预印本 arXiv:1812.04820（2018）。[13] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer。PyTorch。版本 1.10.0。2021-10-01。https://pytorch.org/。[14] A. Krizhevsky, “从微小图像中学习多层特征”，技术报告，2009年。[15] NumPy 开发者。（2021）。NumPy（版本 1.21.2） [软件]。可访问：https://numpy.org/。[16] Dwight Merriman, Eliot Horowitz and Kevin Ryan。MongoDB：现代应用程序的数据库。MongoDB 公司，未注明日期。URL：https://www.mongodb.com/。[17] Chris Duncan。RabbitMQ：消息代理，Pivotal Software 公司，未注明日期。在线资源：https://www.rabbitmq.com/。[18] Apache 软件基金会，“Apache Cassandra”，2010年。 [在线]。可访问：https://cassandra.apache.org/ [19] David M. Beazley。PyCrypto：Python 密码学工具包。2014年。https: // github.com/dlitz/pycrypto。[20] Netzer, Y. and Wang, T. and Coates, A. and Bissacco, A. and Wu, B. and Ng, A.Y.，通过无监督特征学习识别自然图像中的数字，加利福尼亚大学伯克利分校，2011年。[21] Xiao, H., Rasul, K., & Vollgraf, R.（2017）。Fashion-MNIST：用于机器学习算法基准测试的新型图像数据集。arXiv 预印本 arXiv:1708.07747。[22] Garfinkel, S., & Spafford, G.（2002）。UNIX 与互联网安全实践。O’Reilly Media 公司。[23] Yang, Q., Liu, Y., Chen, T., & Tong, Y.（2018）。联邦机器学习：概念与应用。ACM 智能系统与技术汇刊，10(2)，12。doi：10.1145/3196321[24] Neo4j 公司。Neo4j。https://neo4j.com/，2023年。访问日期：2023-02-18。[25] PostgreSQL 全球开发组。（2021）。PostgreSQL（版本 13.4）。取自 https://www.postgresql.org/。访问日期：2023-02-18。[26] McMahan, H. Brendan and Moore, Eider and Ramage, Daniel and Ag"uera y Arcas, Blaise。学习差分隐私循环语言模型。载于《国际机器学习会议》，第2738–2747页，2018年。[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith。联邦学习：挑战、方法与未来方向。CoRR，abs/1908.07873，2019年。http://arxiv.org/abs/1908.07873。[28] Pouyanfar, S., Ibrahim, M., Al-Jubouri, H., & Meskin, N.（2019）。联邦学习综述：从理论到实践。IEEE Access，7，91918-91942。[29] Q. Yang, Y. Liu, T. Chen, and Y. Tong，“联邦机器学习：概念与应用”，《ACM 智能系统与技术汇刊》，第10卷，第2期，第12:1–12:19页，2019年。[30] Papernot, N., Abadi, M., Erlingsson, ’U., Goodfellow, I., & Talwar, K.（2017）。利用私有训练数据进行深度学习的半监督知识迁移。载于《2017年 ACM SIGSAC 计算机与通信安全会议论文集》（第1472–1485页）。[31] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov。针对机器学习模型的成员推断攻击。载于《2017年 IEEE 安全与隐私研讨会（SP）》，第3–18页，2017年。[32] Wang, Haoyu and Shi, Wenchang and Zou, Yulong and Yang, Zhaohui and Sun, Lifeng。（2020）。资源受限边缘计算系统中的自适应联邦学习。《IEEE 认知通信与网络汇刊》，6(2)，608–618。[33] Haibo Li, Shuhang Gu, Zehao Xiao, Shiqiang Wang, and Baochun Li。“通过自适应参数传输实现通信高效的边缘学习。”《IEEE 神经网络与学习系统汇刊》31，第1期（2020）：145-156。[34] Liu, X., Liu, Z., Wang, C., Zhang, J., & Tian, J.（2020）。区块链上的联邦学习：一种面向物联网应用的安全方法。《IEEE 物联网杂志》，8(5)，3632-3642。[35] Howard, J., & Gugger, S.（2018）。用于文本分类的通用语言模型微调。arXiv 预印本 arXiv:1801.06146。[36] Tianqi Chen、Mu Li、Yutian Li、Min Lin、Naiyan Wang、Minjie Wang、Tianjun Xiao、Bing Xu、Chiyuan Zhang 和 Zheng Zhang。MXNet：面向异构分布式系统的灵活高效机器学习库。arXiv 预印本 arXiv:1512.01274，2015年。[37] Gao, Y., Zhong, Y., & Yang, Q.（2021年）。自动化机器学习：最新进展综述。Knowledge-Based Systems，212，106621。爱思唯尔。[38] Ji, S.（2018年）。联邦学习的 PyTorch 实现。Zenodo。doi：10 5281 4321561 10 5281 4321561。/[39] 1996 zenodo。. 取自 https://doi.org/ 。/zenodo。Menezes, A. J., Vanstone, S. A., & Oorschot, P. C. V.（）。《应用密码学手册》（第1版）。CRC出版社，美国。ISBN 0849385237。[40] Python 密码学开发者。（2021年）。密码学（版本 3.4.7）。取自 https://cryptography.io/ [41] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., & Liu, J.（2017年）。去中心化算法能否胜过集中式算法？去中心化并行随机梯度下降案例研究。CoRR，abs/1705.08565。[42] Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farhad, F., Jin, S., Quek, T. Q. S., & Poor, H. V.（2019年）。差分隐私联邦学习：算法与性能分析。arXiv 预印本 arXiv:1911.00222。取自 https://arxiv.org/abs/ 1911 00222 。Mehryar Mohri、Gary Sivek 和 Ananda [43] Theertha Suresh。《无关先验的联邦学习》。CoRR abs/1902.00146（2019年）。arXiv:1902.00146。http://arxiv.org/abs/1902. 00146[44] Yue Zhao、Meng Li、Liangzhen Lai、Naveen Suda、Damon Civin 和 Vikas Chandra。非独立同分布数据下的联邦学习。CoRR，abs/1806.00582，2018年。[45] X. Li, K. Huang, W. Yang, S. Wang 和 Z. Zhang，“非独立同分布数据下 FedAvg 的收敛性”，arXiv 预印本 arXiv:1907.02189，2019年。[在线]。可在以下网址获取：https://arxiv.org/abs/1907.02189。[46] 美国国家标准与技术研究院。（2001年）。联邦信息处理标准出版物（FIPS）197：高级加密标准（AES）。取自 https://csrc.nist.gov/publications/detail/fips/197/final（访问日期：2023-02-18）。[47] S. Frankel、R. Glenn、S. Kelly 和 M. Kohno，“AES-CBC 加密算法及其在 IPsec 中的应用”，网络工作组，2003年。可在以下网址获取：https://tools.ietf.org/html/rfc[48] 3602 B. Kaliski，“PKCS 7：密码消息语法标准”，RSA 实验室，1998年。可在以下网址获取：https://tools.ietf.org/html/rfc2315[49] Xu, Guowen、Han, Xingshuo、Xu, Shengmin、Zhang, Tianwei、Li, Hongwei、Huang, Xinyi 和 Deng, Robert H.《Hercules：提升隐私保护联邦学习的性能》。IEEE《可靠与安全计算汇刊》，卷 -，期 -，页 1-18，2022年，doi：10.1109/TDSC.2022.3218793。[50] Eunjeong Jeong、Seungeun Oh、Hyesung Kim、Jihong Park、Mehdi Bennis 和 Seong-Lyun Kim。“通信高效的设备端机器学习：非独立同分布私有数据下的联邦蒸馏与增强。”ArXiv，2018年。[51] Xinwei Zhang、Wotao Yin、Mingyi Hong 和 Tianyi Chen。“混合联邦学习：算法与实现。”ArXiv，2020年。[52] Jenny Hamer、Mehryar Mohri 和 Ananda Theertha Suresh。FedBoost：一种通信高效的联邦学习算法。载于第37届国际机器学习会议论文集，机器学习研究论文集第119卷，页3973–3983，2020年。[53] Divyansh Jhunjhunwala、Advait Gadhikar、Gauri Joshi 和 Yonina C. Eldar。面向通信高效联邦学习的模型更新自适应量化。载于 ICASSP 2021——2021年 IEEE 国际声学、语音与信号处理会议（ICASSP），页3110-3114，2021年。[54] Daniel Rothchild、Ashwinee Panda、Enayat Ullah、Nikita Ivkin、Ion Stoica、Vladimir Braverman、Joseph E. Gonzalez 和 R. Arora。FetchSGD：基于草图压缩的通信高效联邦学习。载于国际机器学习会议，2020年。[55] Cheng, Hsin-Pai、Patrick Yu、Haojing Hu、Syed Zawad、Feng Yan、Shiyu Li 和 Yiran Chen。“借助差分隐私迈向去中心化深度学习。”载于《通信网络安全与隐私》，页130-145。施普林格，Cham，2019年。[56] Mohassel, Payman 和 Yupeng Zhang。“SecureML：一种可扩展的隐私保护机器学习系统。”载于2017年 IEEE 安全与隐私研讨会（SP），页19-38。IEEE，2017年。[57] Hsu, Tzu-Ming Harry、Qi 和 Matthew Brown。“衡量非独立数据分布对联邦视觉分类的影响。”arXiv 预印本 arXiv:1909.06335（2019年）。[58] Andrew Hard、Kanishka Rao、Rajiv Mathews、Françoise Beaufays、Sean Augenstein、Hubert Eichner、Chloé Kiddon 和 Daniel Ramage。移动键盘预测的联邦学习。ArXiv，2018年。https://arxiv.org/abs/1811.03604[59] Sebastian Caldas、Sai Meher Karthik Duddu、Peter Wu、Tian Li、Jakub Konečný、H. Brendan McMahan、Virginia Smith 和 Ameet Talwalkar。LEAF：联邦学习场景的基准测试。arXiv, 2018 1812 01097 。[60] https://arxiv.org/abs/ 。Keith Bonawitz、Vladimir Ivanov、Ben Kreuter、Antonio Marcedone、H. Brendan McMahan、Sarvar Patel、Daniel Ramage、Aaron Segal 和 Karn Seth。用于隐私保护机器学习的实用安全聚合。载于2017年 ACM SIGSAC 计算机与通信安全会议（CCS ’17）论文集，页1175-1191，2017年。https://doi.org/10.1145/3133956.3133982

[61] Sai Praneeth Karimireddy、Satyen Kale、Mehryar Mohri、Sashank J. Reddi、Sebastian U. Stich 和 Ananda Theertha Suresh。“SCAFFOLD：用于设备端联邦学习的随机受控平均”，*ArXiv*，2019年，abs/1910.06378。[62] Shiqiang Wang、Tiffany Tuor、Theodoros Salonidis、Kin Kwong Leung、Christian Makaya、Ting He 和 Kevin Sean Chan。“资源受限边缘计算系统中的自适应联邦学习”，*IEEE*通信选定领域期刊，2018年，第37卷，页1205-1221。[63] Yaliang Li、Bolin Ding 和 Jingren Zhou。“联邦学习实用入门”，第*28*届 *ACM SIGKDD*知识发现与数据挖掘会议论文集，2022年，页4802-4803。[64] Akash Bharadwaj 和 Graham Cormode。联邦计算入门。载于*2022*年数据管理国际会议论文集（*SIGMOD '22*），页2448–2451，美国纽约州纽约市，2022年。计算机协会。https://doi.org/10 1145 3514221 3522561 。/ 。Jakub Konecn’y、[65] H. B. McMahan、Daniel Ramage 和 Peter Richt’arik。联邦优化：面向设备端智能的分布式机器学习。*ArXiv*，abs/1610.02527，2016年。[66] Gang Bao 和 Peng Guo。云边协同架构中的联邦学习：关键技术、应用与挑战。云计算期刊，11(1):94，2022年。Springer Science and Business Media LLC。https://doi.org/10 1186 13677 022 00377 4 。/s - - - Yejia Liu、[67] Weiyuan Wu、Lampros Flokas、Jiannan Wang 和 Eugene Wu。《支持联邦学习的基于SQL的训练数据调试》。《VLDB学报》，15：388–400，2021年。[68] Tao Lin、Lingjing Kong、Sebastian U. Stich 和 Martin Jaggi。《联邦学习中的鲁棒模型融合集成蒸馏》。载于《第34届神经信息处理系统国际会议论文集》，第198–210页，加拿大不列颠哥伦比亚省温哥华，2020年。[69] Maruan Al-Shedivat、Jennifer Gillenwater、Eric P. Xing 和 Afshin Rostamizadeh。《通过后验平均实现联邦学习：一种新视角与实用算法》。《arXiv》，abs/2010.05273，2020年。[70] Binhang Yuan、Cameron R. Wolfe、Chen Dun、Yuxin Tang、Anastasios Kyrillidis 和 Chris Jermaine。《使用独立子网络训练全连接神经网络的分布式学习》。《VLDB学报》，15(8)：1581–1590，2022年4月。[71] Mohd Abu、Nurul Amelina Hasan Zahri、Asyraf Amir、Mohd Ikhwan Ismail、Azhany Yaakub、Syaheerah Ahmad Anwar 和 Mohd Izhar Ahmad。《迁移学习优化在视野缺损分类中的综合性能分析》。《诊断学》，12(5)：1258，2022年。