FedDUAP: 使用服务器上的共享数据进行动态更新和自适应修剪的联邦学习

摘要

联邦学习经过多年的发展已经取得了卓越的性能，但计算资源受限和训练效率低下仍然是联邦学习面临的重要问题。为此，在 FedDUAP 这篇文章中，作者开发了一种动态服务器更新算法 (FedDU) 和一种分层自适应模型剪枝方法 (FedAP)，并通过将这两种方法集成在一起提出了一种新的联邦学习框架——FedDUAP。FedDU 利用服务器上的不敏感数据，动态确定服务器更新的最优步骤，以提高全局模型的收敛性和准确性；FedAP 根据多层的不同维度和重要性进行独特的剪枝操作，实现效率与效果的良好平衡。本文利用百度 Paddle 平台对 FedDUAP 方法进行了复现，并参照论文的对比试验安排开展试验，得到与论文相似的整体性能。很好地表明了 FedDU 和 FedAP 对模型训练的积极贡献，两者集成得到的 FedDUAP 方法在准确性、效率、计算成本方面优于基线方法。

关键词：联邦学习；模型剪枝；动态更新

1 引言

联邦学习是一种新兴的分布式机器学习框架，它允许将训练数据分布在移动设备上，并通过聚合本地计算的更新来学习共享模型 [1]。联邦学习可以有效解决数据孤岛和隐私保护问题，在确保数据不离开本地的情况下进行联合训练，建立共享的机器学习模型。

从联邦学习的工作流程可知，中央服务器和各个客户端需要不断地在上下行通信链路交换大量的模型参数，这造成了高昂的通信开销。并且在资源受限的设备上训练模型，庞大的参数量会占用极大的内存和计算资源，会存在局部训练效果不佳和本地设备计算能力不足的问题，甚至无法运行 [2]。因此，如何降低联邦学习过程中的通信开销，提高通信效率，降低边缘设备的计算压力，提高运行效率，成为了联邦学习实际应用中的重中之重。

解决这些问题最常用方法的是模型剪枝，即探索神经网络模型权重中的冗余，在不显著降低模型性能的条件下修剪非关键权重，以此来减小模型大小并加速模型训练。但是，目前主流的压缩方法例如模型剪枝、固定稀疏率的剪枝和一些单指标的通道剪枝方法普遍存在着剪枝过程耗时过长、未充分裁剪模型中的冗余参数、剪枝力度不易控制等不足之处。因此探索一种在联邦学习中行之有效的模型剪枝方法是必要且重要的。

而 FedDUAP [3] 这篇文章旨在利用服务器上的不敏感数据和计算能力，通过动态更新服务器和自适应模型剪枝的方法，进一步提高全局模型的性能，同时考虑服务器数据和设备数据的非独立同分布程度。通过将这两种技术结合起来，构建一个新型联邦学习框架，在保证准确性的同时提高效率，减少计算和通信成本。

本文主要工作概述如下：

- 动态服务器更新 (FedDU)：一种联邦学习算法，用于实现服务器和边缘设备之间的模型更新。通过共享模型参数的更新，边缘设备可以在保护数据隐私的前提下进行模型训练，从而提高整体模型的性能。
- 自适应模型剪枝 (FedAP)：一种自适应模型剪枝方法，用于减少计算和通信成本。与传统的模型修剪方法不同，FedAP 可以根据联邦学习的特点进行模型修剪，从而在保持模型准确性的同时减少计算和通信开销。
- 设计算法对比试验：分别对 FedDU、FedAP 和 FedDUAP 进行评估，与其它方法进行对比实验，以展现出本文方法的优势。结果表明，FedDUAP 在准确性、效率和计算成本方面，在两个典型模型和两个数据集上明显优于目前最先进的方法。

2 相关工作

在联邦学习被提出后，很快对联邦学习框架的探索优化就接踵而至了。在早期的工作中，研究者更多的考虑设备端数据 [4] [5] 在本地更新中的影响，虽然可以从设备端收集数据来假定服务器数据是 IID 的，并使用简单的平均方法将其视为普通设备中的数据 [6]，但服务器上的数据 IID 分布是不现实的。同时，将设备数据上传至服务器可能导致数据隐私和安全问题，而数据传输也将大幅增加通信成本 [7]。目前，对服务器数据基于知识转移使用在如异构模型 [8]、无标记数据 [9]、一次性训练 [10] 等特定场景中，但与其对应的也会产生准确率低或训练时间长等问题。而通过利用服务器数据来选择设备进行训练 [11] 与 FedDU 方法相似相合。

模型剪枝可以显著降低 FL 训练的时间和计算成本 [12]，其分为结构化剪枝和非结构化剪枝。结构化剪枝通过去除掉一些荣冗余的神经元（卷积层中的过滤器）改变模型结构；而非结构化剪枝通过将模型中一些不重要的参数权重设置为 0，并不改变模型结构。非结构化剪枝在保证准确度的情况下可以有效降低通信成本 [13]，但在降低计算成本方面优势有限。结构化剪枝虽然在通信和计算成本上都有建树 [13]，但对神经元的保留操作比较复杂，并且现有研究不能很好的适用联邦学习过程。

3 本文方法

3.1 本文方法概述

FedDUAP 框架由三个主要组件组成：系统模型、动态服务器更新算法（FedDU）和自适应模型剪枝方法（FedAP）。系统模型由一个服务器和 N 个边缘设备组成，服务器性能比设备更强大。FL 训练目标是通过更新全局模型参数 w 来最小化损失函数 $F(w)$。FedDU 算法利用服务器数据和设备数据来更新全局模型。它通过对基于服务器数据计算的随机梯度进行归一化来减少目标异质性，并使用服务器数据对基础方法得到的聚合模型进行进一步更新。在此期间，服务器更新的有效步长根据聚合模型的准确性、数据的 non-IID 程度以及轮数来动态确定。自适应模型剪枝方法根据服务器数据和设备数据的统计信息对模型进行修剪，从而

加速训练过程。通过将联邦学习和自适应模型修剪相结合，FedDUAP 框架在提高模型性能的同时，有效地降低了计算和通信成本，具有较高的实用性和可扩展性。

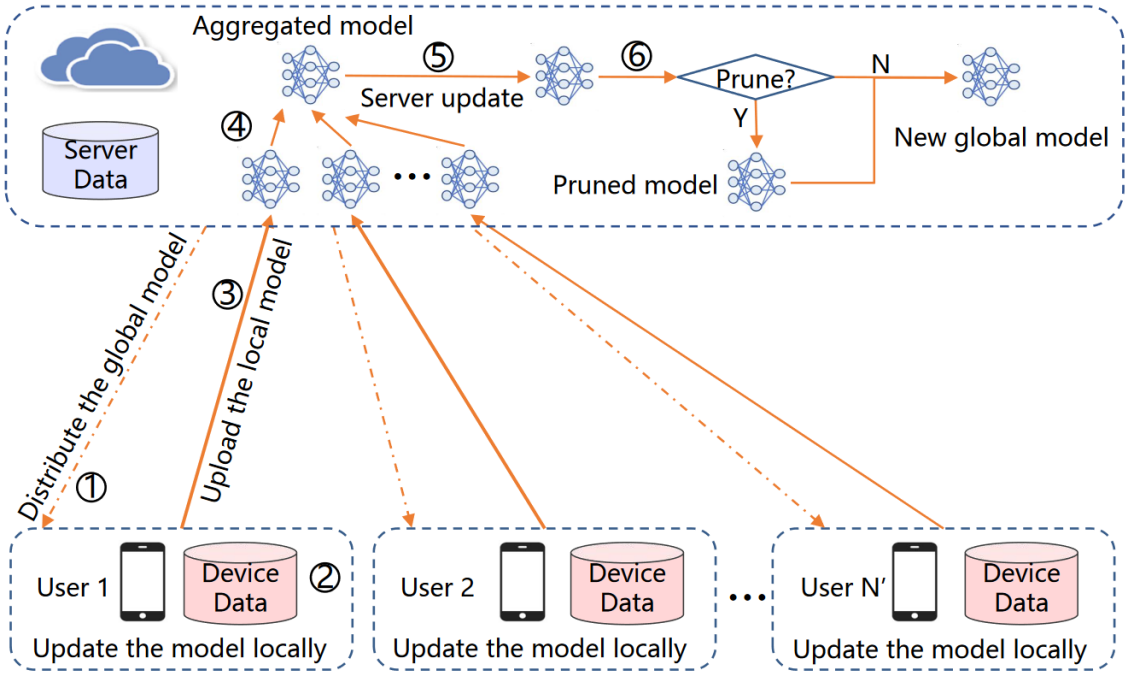

图 1. FedDUAP 框架的训练过程。

如图 1 所示，FedDUAP 进行多个轮次训练，每个轮次由 6 个步骤组成。在步骤①中，服务器随机选择一部分设备来训练全局模型，并将全局模型分发给每个设备。然后，每个设备在步骤②中使用本地数据更新模型。更新后的模型在步骤③中上传到服务器，并在步骤④中使用 FedAvg 算法 [1] 对所有设备上传的模型进行聚合。之后，使用步骤⑤中的服务器数据更新模型（FedDU），详见第 3.3 节。最后，在特定回合中，使用服务器数据和设备数据的统计信息对模型进行剪枝（FedAP），详见 3.4 节。步骤①-④与常规联邦学习步骤相似，同时文章提出使用服务器数据动态更新全局模型（步骤⑤）以提高准确性，并提出一种基于服务器数据和设备数据的自适应模型剪枝方法（步骤⑥）来加速联邦学习的训练。

3.2 整体目标

正如前文所说，FL 训练目标是通过更新全局模型参数 w 来最小化损失函数 $F(w)$。假设设备 k 有本地数据集 $D_k = \{x_{k,j}, y_{k,j}\}_{j=1}^{n_k}$，由 n_k 个样本组成，D_0 表示服务器数据。$x_{k,j}$ 表示设备 k 上的第 j 个输入数据样本，$y_{k.j}$ 是 $x_{k,j}$ 对应的标签。文章将 FL 训练目标表述如下：

$$\min_{w} F(w), \text{with} F(w) \triangleq \frac{1}{n} \sum_{k=1}^{N} n_k F_k(w) \tag{1}$$

其中，w 表示全局模型的参数，$F_k(w) \triangleq \frac{1}{n_k} \sum_{\{x_{k,j}, y_{k,j}\} \in D_k} f(w, x_{k,j}, y_{k,j})$ 表示设备 k 的本地损失函数，损失函数 $f(w, x_{k,j}, y_{k,j})$ 捕获模型在数据对 $\{x_{k,j}, y_{k,j}\}$ 上的误差。

3.3 动态服务器更新 FedDU

FedDU 利用服务器数据动态确定服务器更新的最优步骤，同时考虑服务器数据和设备数据的 non-IID 程度，以提高全局模型的收敛性和准确性。为了减少使用服务器数据生成的随机梯度对聚合模型进行更新可能会受到的客观不一致性的影响，文章对基于服务器数据计算的随机梯度进行了归一化。在公式2中定义 FedDU 中的模型更新：

$$w^t = w^{t-\frac{1}{2}} - \tau_{eff}^{t-1}\eta\bar{g}_0^{(t-1)}(w^{t-\frac{1}{2}}) \tag{2}$$

其中，w^t 表示第 t 轮全局模型的权重，$w^{t-\frac{1}{2}}$ 表示通过 FedAVG 聚合后的全局模型的权重（见3式），τ_{eff}^{t-1} 表示服务器更新的有效步长（见5式），η 表示学习率，$\bar{g}_0^t(\cdot)$ 表示第 t 轮服务器上的归一化随机梯度（见4式）。

$$w^{t-\frac{1}{2}} = \sum_{k\in\mathscr{D}^t}\frac{n_k}{n'}(w^{t-1} - \eta g_k^{t-1}(w^{t-1})) \tag{3}$$

其中，$n' = \sum_{k\in\mathscr{D}^t} n_k$ 是所被选设备的样本总数，$\mathscr{D}^t$ 表示第 t 轮被选设备合集，$g_k^{t-1}(\cdot)$ 是设备 k 使用本地数据计算得到的梯度。

$$\bar{g}_0^{(\iota-1)}(w^{t-\frac{1}{2}}) = \frac{\sum_{i=1}^{\tau} g_0^{(t-1)}(w^{t-\frac{1}{2},i})}{\tau} \tag{4}$$

其中，$w^{t-\frac{1}{2},i}$ 是在第 t 轮次内，服务器按批次进行第 i 次迭代后的聚合模型参数，$g_0^{(t-1)}(\cdot)$ 表示服务器对应的随机梯度，$\tau = \lceil\frac{|n_0|E}{B}\rceil$ 是服务器更新中执行的总迭代次数，E 表示服务器更新轮次，B 表示批次大小。在每一轮中，通过对小批量服务器数据进行采样，使用多次迭代来更新模型。

由于 τ_{eff}^t 对训练过程至关重要，因此根据聚合模型的准确性、数据的 non-IID 程度和轮数动态确定，定义如下式：

$$\tau_{eff}^t = f'(acc^t) * \frac{n_0 * \mathcal{D}(\overline{P'}^t)}{n_0 * \mathcal{D}(\overline{P'}^t) + n' * \mathcal{D}(P_0)} * \mathcal{C} * decay^t * \tau \tag{5}$$

其中，acc^t 是聚合模型（即3所定义）在第 t 轮的准确度（使用不敏感的服务器数据评估），n_0 表示服务器数据的样本数，$\mathcal{D}(\cdot)$ 是 JS 散度（Jensen–Shannon divergence [14]），表示所选择设备数据与服务器数据之间的 non-IID 程度（见附录），$\overline{P'}^t = \frac{\sum_{k\in\mathscr{D}^t} n_k P_k}{\sum_{k\in\mathscr{D}^t} n_k}$ 表示所选设备的所有数据在第 t 轮的分布，P_0 表示服务器数据分布，$decay \in (0,1)$ 用于确保所训练模型最终收敛到公式1的解，$\mathcal{C}$ 是超参数。$f'(acc^t)$ 是基于 acc^t 的函数，在训练之始，acc^t 值相对较小，对应的 $f'(acc^t)$ 值较突出，此时对服务器的高性能及数据加以利用进行模型更新，随着训练的结束，$f'(acc^t)$ 值变小以减小服务器数据的影响。

3.4 自适应模型剪枝 FedAP

FedAP 根据服务器数据和设备数据的 non-IID 程度，对服务器进行独特的剪枝操作，适应多层的不同维度和重要性。请注意，修剪过程是在训练过程中的特定轮次进行的。

首先，使用服务器数据和设备数据计算预期剪枝率。对服务器或设备 k，给定具有初始参数 W_k 的神经网络，经过 T 轮训练后，用 W_k' 表示更新后的参数，并使用 $\Delta = W_k - W_k'$

表示参数更新前后的差异。然后计算损失函数的汉森矩阵（Hessian matrix）$H(W'_k)$ 并对其特征值按升序排序，序列表示为 $\{\lambda_k^m | m \in (1, d_k)\}$, 其中 d_k 表示 $H(W'_k)$ 的秩，m_k 表示特征值的索引。定义一个基本函数 $B_k(\Delta_k) = H(W'_k) - \nabla L(\Delta_k + W'_k)$, 将其利普希茨常数（Lipschitz constant）定义为 $\mathscr{L}_k$。根据 [15] 的工作, 找到第一个满足 $\lambda_(m_{k+1}) - \lambda_(m_k) > 4\mathscr{L}_k$ 的 m_k 以避免精度损失。使用 $p_k^* = \frac{m_k}{d_k}$ 计算预期剪枝率。

其次，由于数据的 non-IID 分布，每个设备计算得到的各自预期剪枝率都有很大差异，因此使用公式 (6) 计算全局预期剪枝率

$$p^* = \sum_{k=0}^{n} \frac{\frac{n_k}{\mathcal{D}(P_k)+\epsilon}}{\sum_{k'=0}^{n} \frac{n_{k'}}{\mathcal{D}(P_{k'})+\epsilon}} * p_k^* \tag{6}$$

其中的 ϵ 是一个很小的值，避免除以 0。设置全局阈值 $\mathscr{V}$，$\mathscr{V}$ 的值与所有参数升序排列的第 $\lfloor R * p^* \rfloor$ 个参数的绝对值相等。然后，将每个卷积层中绝对值低于 $\mathscr{V}$ 的参数数量除以总量来计算每层的剪枝率 p_l^*。

最后，根据 [13] 的工作，使用特征图的秩作为剪枝依据会有很好的效果。计算并使用 $R^l = \{r_l^j | j \in 1, d_l\}$ 表示第 l 层中特征图的秩的升序序列，其中 d_l 表示第 l 层的过滤器数量。保留下 $d_l - \lfloor p_l^* * d_l \rfloor$ 秩所对应的过滤器，并使用保留的过滤器整体替换掉原始模型的过滤器，以此完成剪枝操作。

4 复现细节

4.1 代码说明

文章作者并没有将源码随文章一同公开，但因为其内容的进步性以及方向的相关性吸引着我对其进行研读，通过前文的阐述可以较为清晰的了解这篇文章所做工作对联邦学习模型剪枝方向的贡献。基于此，我在开源网站 GitHub 上搜索并获取到了其他研究者对本文方法的部分实现代码[1]，结合此以开展自己的工作。因服务器可搭载的 Paddle 版本落后于可参考代码的版本，同时由于代码规范问题，导致花了较多时间进行针对版本适配的修改。此外依照参考代码结构完善了 FedAvg 算法的代码并重写了可视化代码。

4.2 实验环境搭建

代码基于百度 Paddle 联邦学习框架实现，使用 gpu==2.3.2、cudatoolkit=11.2 的 paddle 版本, 本地运行在单张 Quadro P5000 服务器显卡上，Python==3.8。

4.3 使用说明

代码使用，命令行形式运行，一个 FedDUAP 使用 ResNet 模型在 non-IID 的 Cifar10 数据集上训练的命令示例如下：

[1]https://github.com/PaddlePaddle/PaddleFL/tree/dcc00c5dff62c3dd0092801f4e9b89d8c0957d3d/python/paddle_fl/mobile/fedDUAP

```
python centralmain.py --iid 0 --share_percent 10 --unequal 0 --dataset "cifar10" --central_train 1 --result_dir "FedDUAP_equal" --epochs 500 --local_bs 10 --local_ep 5 --decay 0.999 --prune_interval 30 --prune_rate 0.6 --share_l 5 --model "resnet"
```

5 实验结果分析

本文参照原文章所做的对比实验进行验证，因实际实验情况不尽相同，对实验设置进行了适应性修改，保证不同的方法除了方法本身的区别外，其它客观实验条件相同。

5.1 实验设置

本文的实验建立了一个由一台服务器和 100 台设备组成的联邦学习系统，每一轮随机选择 10 台设备参与训练。对每种方法分别使用基础 CNN 和 ResNet 两种模型实现，分别在 Cifar10 和 Cifar100 [16] 两个数据集上进行训练，并统一使用 non-IID 数据分布。将原文实验中不同的 p，即表示服务器中数据大小和客户端中数据大小之比统一设定为 10%。由于硬件所造成的时间成本原因，实验统一设置全局运行 500 轮次，本地运行 5 轮次，尽管这会影响到最终结果，但从实验结果来看，就 500 轮以内的对比与原文实验结果相差无几，依然可以很好的展现方法性能。本文所提出的 FedDUAP 方法是包含动态服务器更新算法（FedDU）和分层自适应模型剪枝方法（FedAP）两个模块的集成方法，因此对每一模块进行单独的实验验证也是有意义的。

5.2 对 FedDU 评估

评估 FedDU 时，使用 FedAvg、Data-sharing [17] 和 Hybrid-FL [6] 作为对比方法。Data-sharing 将服务器数据发送给设备，以便设备可以将本地数据与服务器数据混合，形成一个 non-IID 更小的数据集，以提高全局模型的精度。Hybrid-FL 使用 FedAvg 算法将服务器视为普通设备。

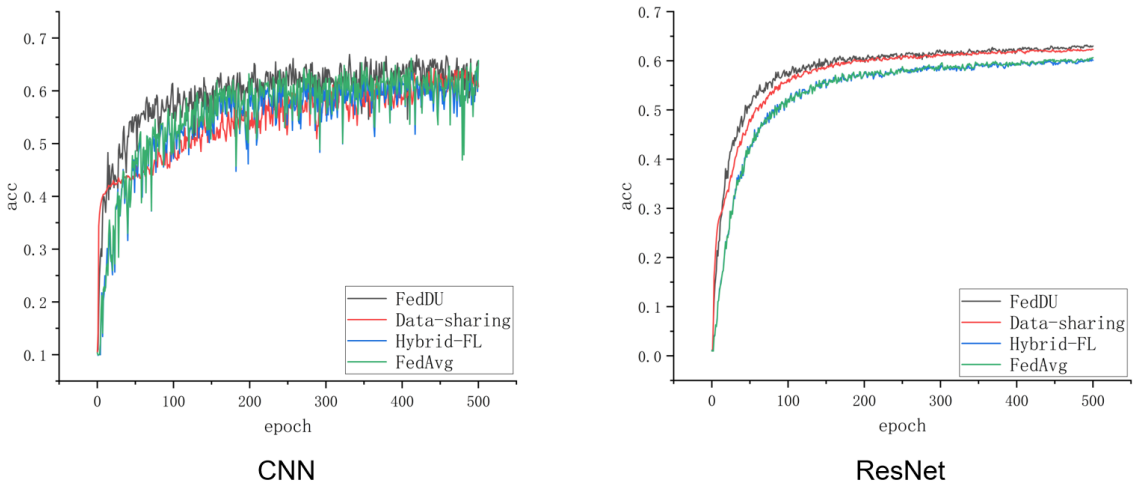

图 2. 对应于 FedDU 的各种模型更新方法的准确性（复现）。

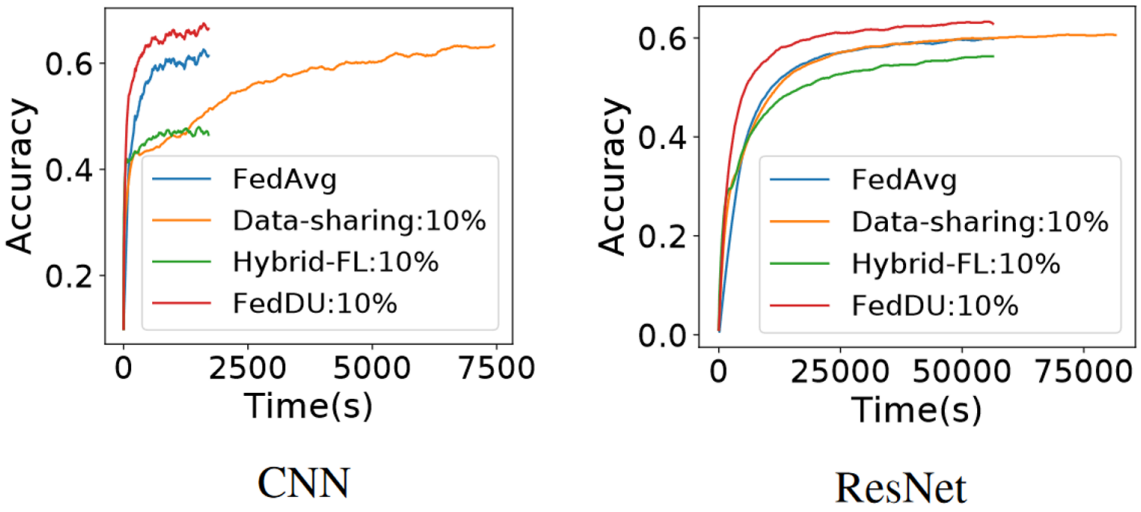

图 3. 对应于 FedDU 的各种模型更新方法的准确性（原文）。

如图2所示，FedDU 在使用 CNN 和 ResNet 模型的准确率都比 FedAvg、Data-sharing 和 Hybrid-FL 更优，并且可以更快地达到一个较好的精度。与 FedDU 相比，Data-sharing 需要将服务器数据传输到设备，这可能会产生重大的通信成本和隐私问题，同时也使得其需要更长的训练时间才能达到 0.6 的准确率。从两张图中也可以看出，使用 ResNet 模型训练的结果比使用 CNN 模型训练更加平滑，波动较小，这很好的说明了 ResNet 模型性能的强悍。这样的结果与原文（图3）近乎吻合，进一步说明本文所提方法的普适性和进步性。

5.3 对 FedAP 评估

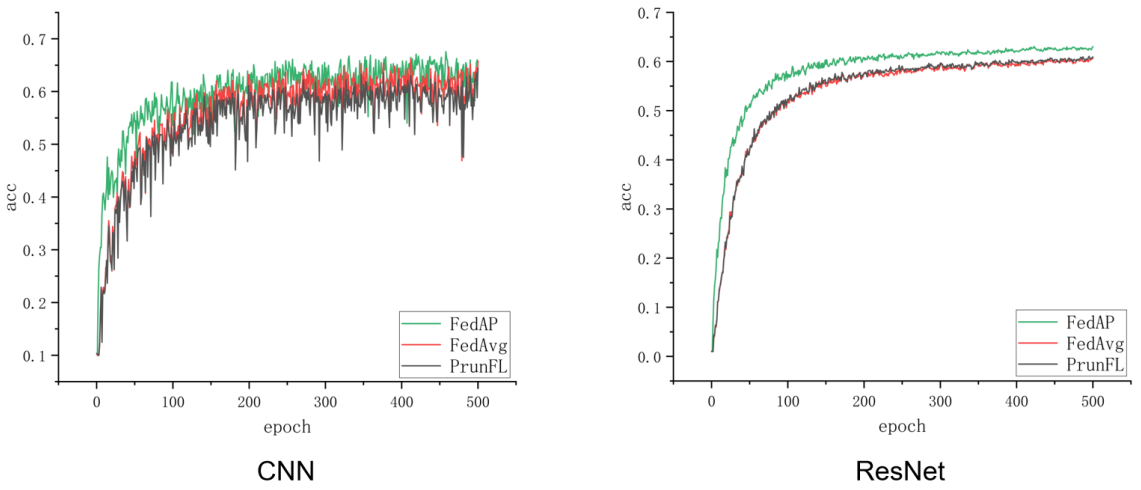

图 4. 对应于 FedAP 和其它基线方法的准确性（复现）。

本节中将 FedAP 方法与 FedAvg 和 PrunFL [12] 剪枝方法进行比较。PrunFL 是一种非结构化剪枝方法。本文的 FedAP 方法为网络的每一层生成自适应剪枝率，可以在有效缩减模型大小、缩小训练时间的同时，达到依然可观的精度。如图4所示，FedAP 的训练过程明显快于 FedAvg 和 PrunFL 到达较高精度，并且在高精度上保持优势。由于非结构化剪枝方法无

法利用通用硬件来加快计算速度，相比之下 FedAP 的计算成本就要小得多。需要说明的是原文中对比的 IMC 方法（图5）在 Paddle 框架下实现所需的硬件环境要求较高，在公用服务器上无法做到针对性环境部署，因此本文暂先搁置 IMC 方法的对比。

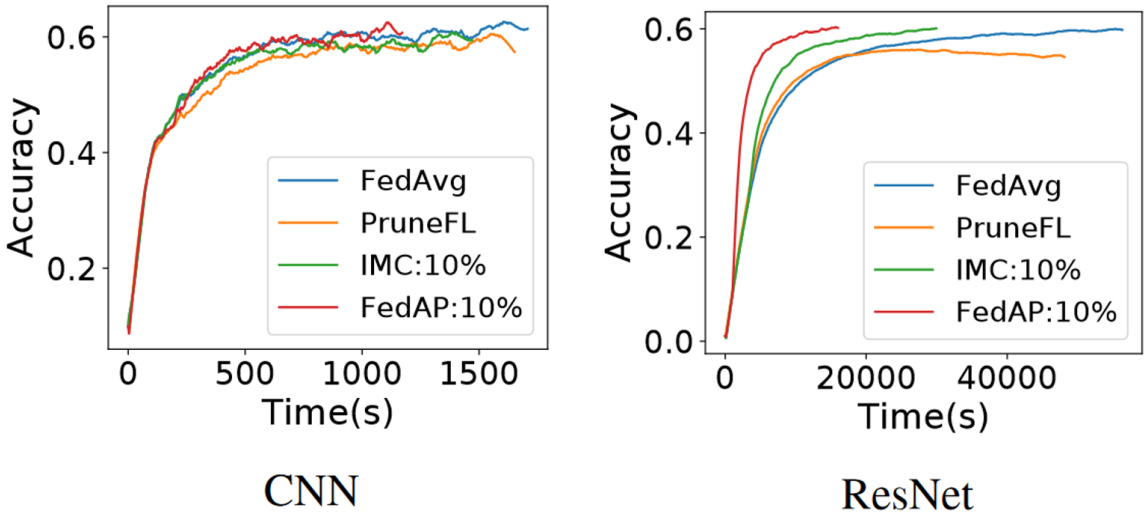

图 5. 对应于 FedAP 和其它基线方法的准确性（原文）。

5.4 对 FedDUAP 的评估

本文的 FedDUAP 方法由 FedDU 和 FedAP 两个子模块组成，在单独评估了 FedDU 和 FedAP 后，本文对 FedDUAP 方法做整体评估。

Model	Method	Accuracy	time
CNN	FedAvg	0.625	108
	Data-sharing	0.638	257
	Hybrid-FL	0.620	222
	PrunFL	0.603	156
	FedDUAP	**0.662**	**37**
ResNet	FedAvg	0.600	397
	Data-sharing	0.620	437
	Hybrid-FL	0.600	790
	PrunFL	0.602	376
	FedDUAP	**0.631**	**87**

表 1. FedDUAP 与其他基线方法对比。其中，time 表示单轮训练所耗平均时间，因公用服务器性能及任务占用程度不同而不同。

如表1所示，与 FedAvg、Data-sharing、Hybrid-FL、和 PruneFL 相比，FedDUAP 的准确率更高。此外，FedDUAP 只需要更短的训练时间就能实现目标精度，比其它方法至少快 2.5 倍。这充分说明由动态服务器更新算法（FedDU）和分层自适应模型剪枝方法（FedAP）两个模块组成的 FedDUAP 联邦学习框架在保证效率的同时，降低通信成本和计算成本上的优势。

6 总结与展望

本文介绍了一种集成了动态服务器更新算法(FedDU)和分层自适应模型剪枝方法(FedAP)的新型联邦学习框架 FedDUAP。其旨在加强资源受限的边缘设备在联邦学习中的贡献，并降低训练过程中的通信成本，使得联邦学习具有更好的效能。同时，本文对 FedDUAP 方法进行了复现，从复现结果来看，可以得到和原文一致的结论，即 FedDU 和 FedAP 在提高准确性和训练速度方面都各有建树，集成的 FedDUAP 在性能上则更上一层楼。

本次的论文复现过程让我对联邦学习和模型剪枝技术有了更加全面的了解，不仅接触了 Paddle 联邦学习框架，也让我了解到了 Data-sharing、Hybrid-FL 等先进的联邦学习算法。但由于可分配时间有限，我对这些方法还没有完全掌握。模型剪枝被公认为是优化联邦学习过程的一个有效手段，有着极高的研究价值，借此复现机会，之后我将继续更加深入的研究联邦学习中的模型剪枝方案，承接 FedDUAP 方法的启发，做出更多的探索。

参考文献

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Bo Tao, Cen Chen, and Huimin Chen. Communication Efficient Federated Learning via Channel-wise Dynamic Pruning. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE.

[3] Hong Zhang, Ji Liu, Juncheng Jia, Yang Zhou, Huaiyu Dai, and Dejing Dou. FedDUAP: Federated Learning with Dynamic Update and Adaptive Pruning Using Shared Data on the Server. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 2776–2782. International Joint Conferences on Artificial Intelligence Organization, July 2022.

[4] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[5] Chendi Zhou, Ji Liu, Juncheng Jia, Jingbo Zhou, Yang Zhou, Huaiyu Dai, and Dejing Dou. Efficient device scheduling with multi-job federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9971–9979, 2022.

[6] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In *ICC 2020-2020 IEEE International Conference On Communications (ICC)*, pages 1–7. IEEE, 2020.

[7] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[8] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020.

[9] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[10] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. *arXiv preprint arXiv:2010.01017*, 2020.

[11] Lokesh Nagalapatti and Ramasuri Narayanam. Game of gradients: Mitigating irrelevant clients in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9046–9054, 2021.

[12] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices, 2022.

[13] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1529–1538, 2020.

[14] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International symposium onInformation theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004.

[15] Zeru Zhang, Jiayin Jin, Zijie Zhang, Yang Zhou, Xin Zhao, Jiaxiang Ren, Ji Liu, Lingfei Wu, Ruoming Jin, and Dejing Dou. Validating the lottery ticket hypothesis with inertial manifold theory. *Advances in neural information processing systems*, 34:30196–30210, 2021.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.