生产实习周报小结

任务题目：FlexiFed 论文复现

报告人：周臻鹏

7 月 26 日-8 月 4 日

目录

一. 简述

在第四周的学习研究中,我主要在做以下工作:

- ResNet 4 种模型的实现和在 CIFAR-10，CINIC-10，Speech_Commands 数据集下的训练
- 文本数据集 Ag News 的下载导入和预处理
- CharCNN 4 种模型的实现和在 Ag News 数据集下的训练
- VDCNN 4 种模型的实现和在 Ag News 数据集下的训练
- 对实验过程 acc 和 loss 曲线的观察分析,尝试调整训练中的部分参数以优化训练效果

目前总任务进度达到 **100%**,FlexiFed 论文实验复现任务已全部完成,所有的数据表格图像已汇总.即便过程中对于结果不符合原文效果的部分实验采取了重复实验和调整参数的处理,但总体上我得出的 FlexiFed 框架的表现效果和原文仍有一些差异,对此我做出了一些思考和分析.

经过了前面几周的踩坑和摸索,第四周对于整个框架项目乃至对于 Machine Learning 的工程流程更加熟悉了,整体实验效率大幅度提高.从本质上在完善了 Max-Common,Clustered-Common,Basic-Common 三种策略方法后,剩余的工作只是将其在不同的模型,不同的数据集上举一反三而已.实验的过程中我时常回顾文章原文,发现了自己有部分训练设置不合理的地方,也尝试做了优化比较:如将 clients 个数从 8 提高到 40,准确率会有较小提高;在聚合策略中取消对全连接层的聚合,也能有效提高最终收敛时的准确率.

实验源码 github 地址:

https://github.com/Yogurteer/FlexiFed-repeat-student

二. 实验内容

1. ResNet 的实现

ResNet 是为了解决传统卷积神经网络在层数过大时的退化问题,在网络中加入残差模块,我的理解是以拟合残差的形式取代传统的拟合输出,削弱了网络层输入输入数据相同的主体部分,扩大了输出变化值对权重参数的影响. 如图 2.1

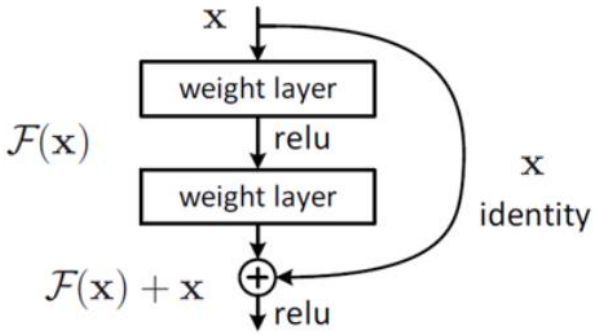

图 2.1 残差模块

图 2.2 所示为 ResNet Family 的主体结构,它们的相似点在于都是 1 层 conv+4 个 block+1 层 fc,残差模块实现在 block 中

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图 2.2 ResNet 主体

根据结构图,以 VGG 模型为母版构建模型即可,我把 block 模块的实现单独封装在 BasicBlock 函数中,构建不同版本的 ResNet 时改变 BasicBlock 的参数即可.

以 ResNet20 为例,网络结构如图 2.3

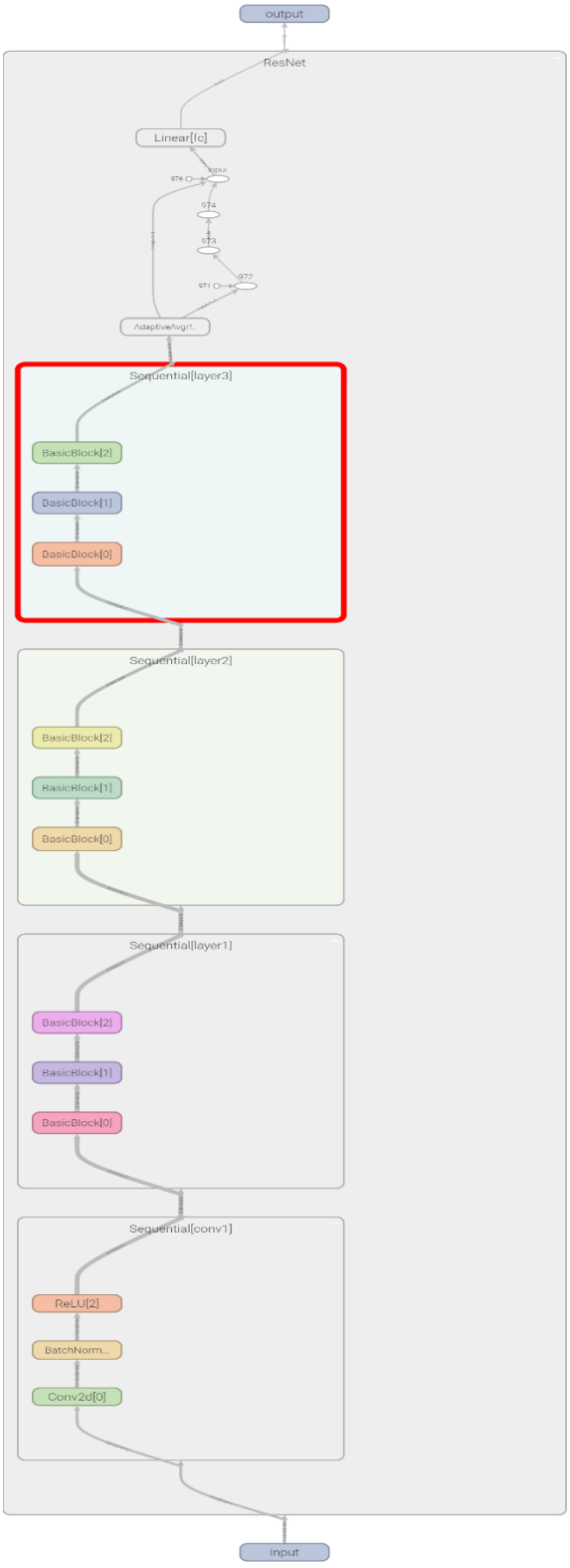

图 2.3 resnet 网络结构图

源代码可见实验代码仓库中的 ResNet.py.

2.Ag News 数据集的导入和预处理

Ag News 数据集为文本类型，每一条数据有三列，第一列为 label，第二列为 title，第三列为 content.

导入:

在我自定义的 Dataset 子类 AG_NewsDataset 类中，数据集的读取通过 csv.reader() 函数实现,共四个类别,每个类别下有 30,000 个训练样本,总计 120,000 个训练样本.

预处理:

切割得到单个样本后,原始数据类型为以字符为元素的 list,估计其字符长度上限为 1024(参考网上文献的数据),以字符为单元,任意一个字符,例如'a', '1','!','&'映射为一个互不相同的 int 数,如果为空则统一映射为 0,由此将一个不定长度文本样本映射为一个 size 固定为(1024)的一维 tensor,便于后续的模型处理.

源代码可见实验代码仓库中的 DelDataset.py.

3.CharCNN 的实现

之前以字符为单位做数据预处理正是为了此处 CharCNN 服务的.传统 CNN 面向的是二维图像 tensor 数据,似乎并不能直接用于一维的文本数据任务.

CharCNN 作为字符级卷积神经网络正是起到了文本字符与卷积神经网络之间的桥梁作用.在我的理解里,version=x(x=3,4,5,6)的 CharCNN 模型的结构为:1 层 Embedding+2 层卷积(卷积后 MaxPool)+x 层卷积(无 Pool)+3 层 Fc

模型结构抽象为下图 2.4:

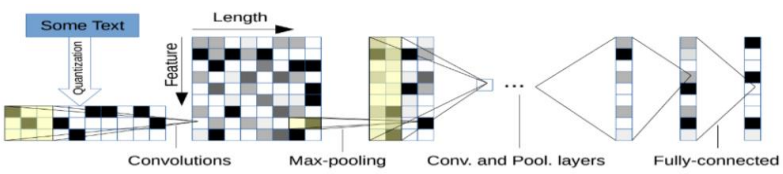

图 2.4 CharCNN 示意图

以 CharCNN5 为例,网络结构如图 2.5:

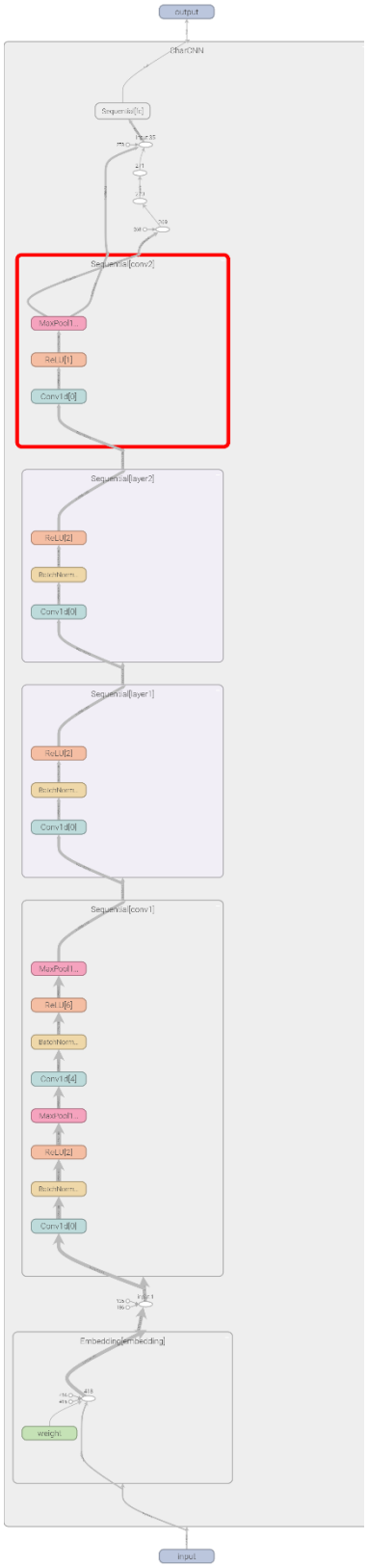

图 2.5 CharCNN5 网络结构图

4.VDCNN 的实现

VDCNN 模型与 CharCNN 模型的相同在于都是以卷积和池化为主要模块,不同在与 VDCNN 的网络深度更深,层数更多.我的理解下 VDCNN 的网络结构为:

1 层 Embedding(输出 16 维)+1 层初始卷积(输出 64 维)+(卷积模块(每个模块 2 次卷积+Batch_Norm+ReLu)+MaxPool)*若干次+1 层 K-Max Pool+3 层 Fc

如图 2.6 表示 VDCNN17 模型的结构,其中卷积模块+MaxPool 的组合重复 8 次,总计 17 层卷积.

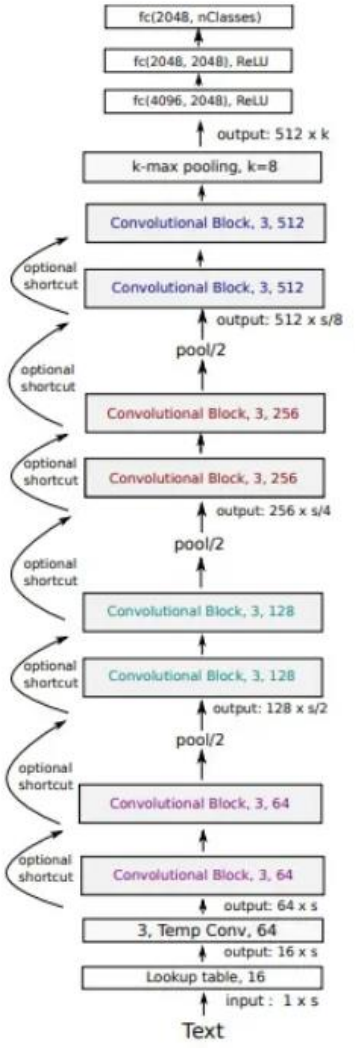

图 2.6 VDCNN17 示意图

网络结构图如图 2.7:

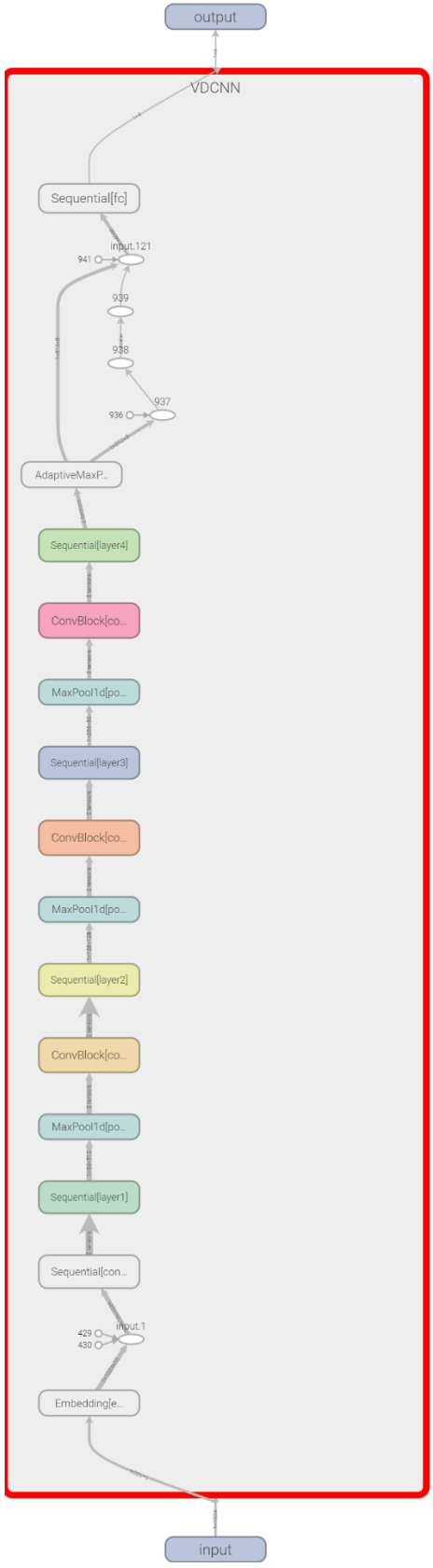

图 2.7 VDCNN17 网络结构图

三. 结果分析

最终实验结果见表 3.1:

Dataset	Models	Scheme	Versions			
			V1	V2	V3	V4
CIFAR-10	VGG	Basic-Common	64.2	64.3	65.1	66.2
		Clustered-Common	71.2	74.7	76.5	79.8
		Max-Common	72.9	77.2	79.3	81.1
	ResNet	Basic-Common	71	71.9	73.6	74.2
		Clustered-Common	78.1	79.1	81.1	81.7
		Max-Common	77.5	79.4	80.9	82.1
CINIC-10	VGG	Basic-Common	43.3	47.1	45.1	42.9
		Clustered-Common	54.2	54.3	56.6	57.5
		Max-Common	55.3	62.1	64.1	64.7
	ResNet	Basic-Common	46.7	46.9	48.1	48.9
		Clustered-Common	54.3	54.8	55.5	56.4
		Max-Common	55.4	55.7	57.1	58.2
AG NEWS	CharCNN	Basic-Common	84.1	85.2	85.6	86.7
		Clustered-Common	84.9	86.1	87.3	87.7
		Max-Common	71.5	84.3	84.8	86.9
	VDCNN	Basic-Common	83.5	83.6	85	85.8
		Clustered-Common	81.3	81.6	82.7	86.2
		Max-Common	86.3	87.8	87.9	89.2
Speech Commands	VGG	Basic-Common	80.2	82.2	83.9	84.1
		Clustered-Common	84.2	84.5	85.3	86.4
		Max-Common	84.9	85.9	86.7	86.9
	ResNet	Basic-Common	82.9	83.4	86.4	87.1
		Clustered-Common	86.9	90.2	91.3	91.3
		Max-Common	86.8	90.1	91.8	92.5

表 3.1 最终实验结果表

整个实验包含 4 个数据集,每个数据采用 2 种模型,每种模型下都分别使用 3 种模型聚合策略联邦学习训练 4 个版本(每次训练 8 个 Client,每 2 个 Client 使用同一个版本的模型),共计 24 次实验,得出 72 项准确率结果.

全部实验的数据汇总表格也可见"FlexiFed 实验复现结果数据汇总.xlsx",其中包括训练失败,结果完全不合理的实验记录.

数据图像的汇总可见"FlexiFed 实验复现结果曲线图汇总.pdf".

下图表 3.2 是原文最终实验结果表:

Dataset	Models	Scheme	Versions			
			V1	V2	V3	V4
CIFAR-10 [24]	VGG [46]	Standalone	64.4	64.8	65.2	65.6
		Clustered-FL	78.2	80.4	80.8	81.2
		Basic-Common	65.2	66.8	67.2	68.2
		Clustered-Common	80.2	82.4	83.2	83.6
		Max-Common	80.6	85.2	86.6	86.8
	ResNet [14]	Standalone	57.2	57.8	58.8	59.2
		Clustered-FL	73.6	74.6	75.2	75.8
		Basic-Common	66.4	67.6	67.8	68.4
		Clustered-Common	78.4	79.2	80.6	80.8
		Max-Common	78.8	79.6	83.4	84.2
CINIC-10 [8]	VGG	Standalone	44.4	45.6	47.4	49.2
		Clustered-FL	58.8	59.8	60.4	60.6
		Basic-Common	48.8	50.8	51.2	51.4
		Clustered-Common	60.8	62.8	63.4	63.8
		Max-Common	61.4	67.6	67.8	68.2
	ResNet	Standalone	46.2	47.2	47.8	49.4
		Clustered-FL	58.2	58.6	59.8	60.4
		Basic-Common	49.8	51.4	51.8	52,2
		Clustered-Common	60.8	61.4	63.4	64.2
		Max-Common	61.6	64.2	66.2	66.6
AG NEWS [65]	CharCNN [65]	Standalone	51.9	53.2	65.1	70.2
		Clustered-FL	76.2	77.7	84.1	84.7
		Basic-Common	84.6	85.6	85.7	86.1
		Clustered-Common	85.2	85.6	86.1	86.3
		Max-Common	85.9	86.5	86.7	87.6
	VDCNN [6]	Standalone	73.2	75.8	77.8	78.6
		Clustered-FL	84.6	85.2	86.2	86.4
		Basic-Common	78.2	79.6	80.4	80.6
		Clustered-Common	84.8	86.6	87.8	88.2
		Max-Common	88.2	89.2	89.6	90.2
Speech Commands [54]	VGG	Standalone	77.5	78.2	80.5	81.5
		Clustered-FL	80.2	81.8	83.4	85.8
		Basic-Common	78.6	80.4	81.2	82.4
		Clustered-Common	82.4	84.6	86.2	86.4
		Max-Common	82.6	86.6	87.4	89.8
	ResNet	Standalone	75.2	78.6	79.4	82.8
		Clustered-FL	79.2	82.0	83.6	84.2
		Basic-Common	83.6	87.6	88.2	89.2
		Clustered-Common	84.8	88.8	89.2	89.8
		Max-Common	85.2	89.6	90.8	91.4

表 3.3 原文实验结果表

1.与原文的比较

整体数据和原文相比,80%以上的数据与原文差异在5%以内,可以认为实验结果基本是合理的.

经统计:

- 52.0%的数据略低于原文实验的结果,主要集中在以VGG为模型的实验中,根据我的推测,VGG模型的实验是我早期训练时得出的结果,在当时没有意识到聚合模型中的全连接层对联邦学习系统的训练反而会起到消极作用,因为每个的模型的个性化特性恰恰体现在它的全连接层,不应参与聚合(原文5.3In-depth Evaluation Table3给出证明);还有一部分略低的数据出现VDCNN模型中,猜测可能是模型太深发生了退化的结果.

Table 3: Model Accuracy with and without FC layers under Max-Common

Model | Dataset	With FC Layers	Without FC Layers
VGG | CIFAR-10	83.55	85.10
VGG | CINIC-10	64.75	66.95
VDCNN | AG News	78.95	89.05
ResNet | Speech Commands	86.75	88.55

图3.1 原文Table 3证明了聚合全连接层的消极作用

- 38%的数据略高于原文实验的结果,主要集中在以Speech_Commands为模型的实验中,根据我的推测,在之前对Speech_Commands的预处理中,我的流程是先将时序音频数据转化问mel频谱图数据,之后再对依次做标准化-归一化处理,使数据集尽可能地接近正态分布(实际上最终还是不符合正态分布),起到了数据增强的作用,然后我将学习率lr从0.01调整为0.001(因为对于当前数据集,以lr=0.01时频繁出现震荡),一定程度上小的lr更有利于求得全局最优解达到更高准确率,加上此时的训练已经取消了对全连接层得聚合,三者相结合使得结果略高于原文.

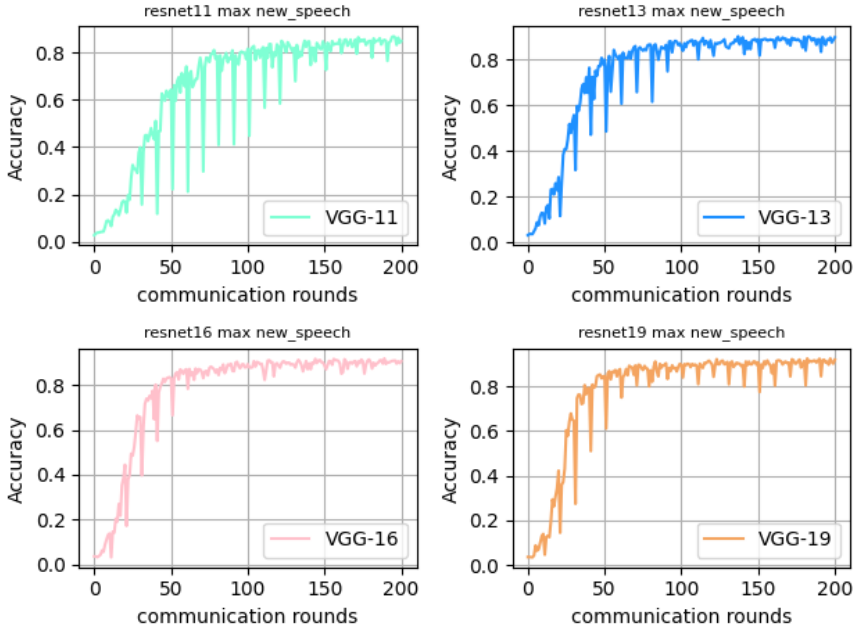

图 3.2 Speech_Commands 在 resnet 模型上使用 Max-Common 的训练结果

- 10%的数据基本不符合理论预期,出入较大,但是我重复 1-2 次实验后仍然无果,目前仍缺乏合理的解决方案.这些结果集中在 ResNet 模型在 CINIC-10 数据集上的训练,案例如图 3.3:

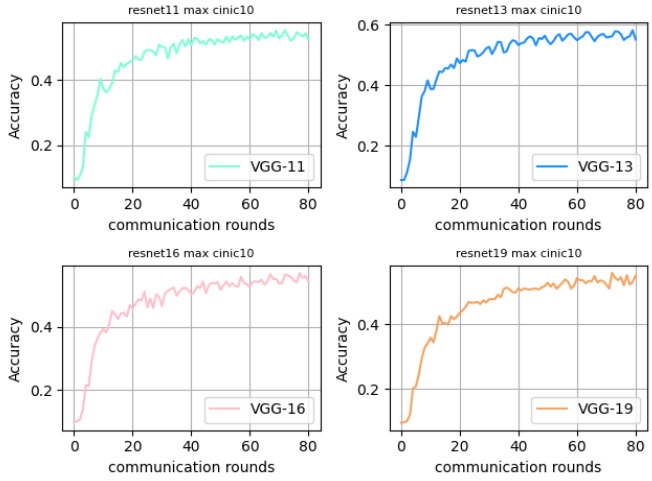

图 3.3 CINIC-10 在 ResNet 上使用 Max-Common 的 acc 曲线

四个版本的模型最终收敛准确率在 55%-59%之间,而原文同场景的准确率在 61%-67%之间,我猜测是因为数据集太大,数据分区时偶然发生了数据分布极度不均,发生了过拟合现象,理由是该案例下 loss 曲线已经下降到了很低的数值.

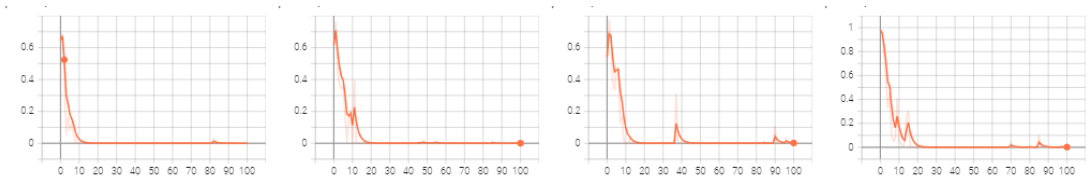

图 3.4 CINIC-10 在 ResNet 上使用 Max-Common 的 loss 曲线

而对于 CINIC-10 数据集以外发生的不合理实验结果,我认为是偶然出现的结果,该场景下的实验次数均不超过 3 次,这种现象也是有可能发生的.

2.不同策略之间的比较

经比较,在同一数据集和同一模型下:

- Max-Common 策略和 Clustered-Common 策略的训练效果明显优于 Basic-Common,证明了聚合较深层的模型参数的确能够加强用户端之间的知识共享,进而提高最终的准确率.
- Max-Common 策略略优于 Clustered 策略的训练效果,证明了 Max-Common 在目前的确能够最大化挖掘 FlexiFed 框架的潜力,极大程度地强化不同模型结构间的知识共享以获得最好的性能.

证明案例见图 3.5:

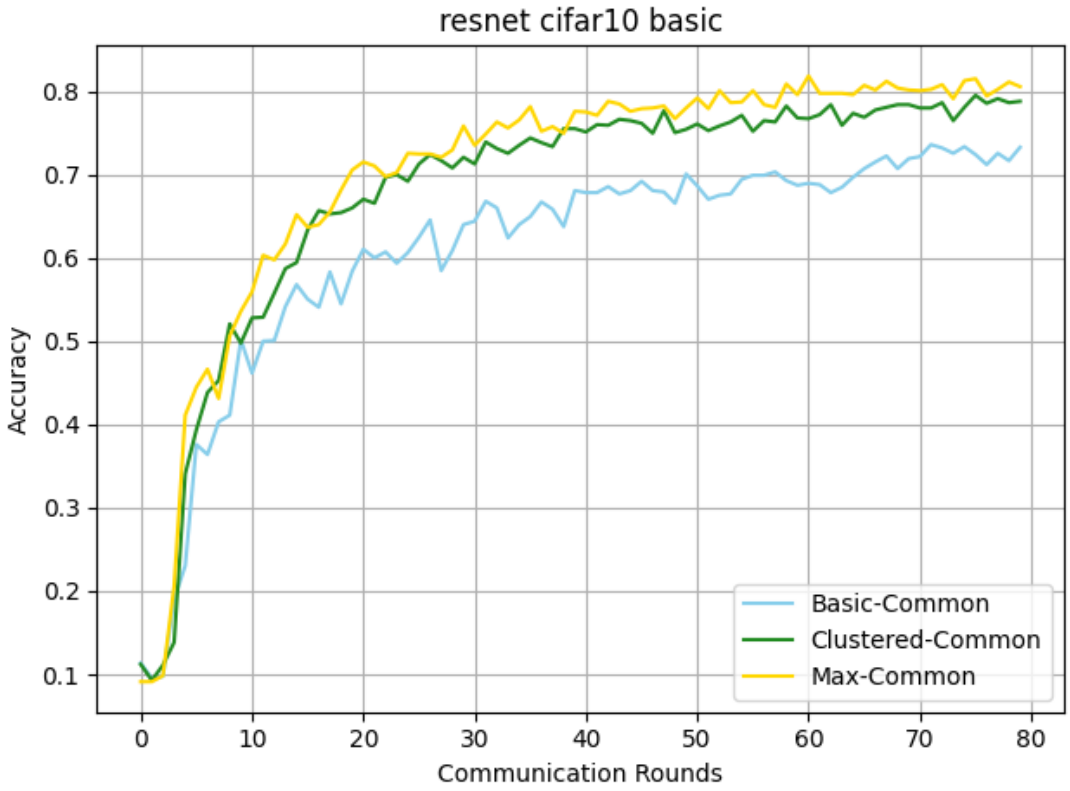

图 3.5 ResNet 模型-CIFAR-10 数据集场景下不同聚合策略的准确率比较

四.后续的展望

本次虽然复现实验设定了一个月的周期,但是我前期花了大量时间部署项目工程和实现源码,再加上不少失败的实验记录,并没有达到最好的效果.针对联邦学习中不同模型结构之间的用户端如何优化的这一问题,我想如果还有更多时间精力我会从以下方面继续研究:

1.与 Basline 的比较

原文中为了证明 FlexiFed 框架的有效性,是将新提出的三种聚合策略与 Clusterd-FL(原文以前最好的方法),Standalone(完全不聚合,单机训练的方法)进行了比较,我也应该控制其他变量,将 5 种策略一起比较才能获得更有说服力的证据.

2.不同 Client 之间的比较

本次实验我犯了一个重大失误,原文中 A.4 描述了实验设置 client 个数为 40,但是我复现实验时一直保持 client 个数为 8,会直接影响到每个用户端的数据规模和整个联邦学习系统的知识共享效果,理论上 client 个数越多,训练效果越好.我应当控制其他变量,设置不同的 client 数量进行比较,从而证明结论.

3.是否聚合全连接层的比较

在训练 VGG 模型阶段,我一直忽略没有取消在 3 种策略中对全连接层的聚合,但是到训练 ResNet,CharCNN 和 VDCNN 模型阶段,我又突然醒悟在策略中取消了对全连接层的聚合.理论上全连接层是每个模型个性化特性的体现,不聚合会得到更好的效果.但是我的实验中无法科学地比较是否聚合全连接层之间的优劣.我应当控制其他变量,设置是否聚合全连接层进行比较,从而证明结论.

4.非独立同分布数据场景下的比较

原文提到过，现实世界中，由于环境和应用程序使用模式的差异，客户的训练数据往往分布有所不同.所以在非独立同分布数据场景下的比较才更有实际意义.但我在复现实验中一直都保持使用独立同分布的数据分区方式.我应该尝试统一使用非独立同分布的数据集分区方式,比较 Clustered-FL 与 Max-Common 的效果以证明 FlexiFed 框架的优越性,比较 Max-Common 和 Clustered-Common 和 Basic-Common 以证明强化不同结构模型之间的知识共享能提高联邦学习效果的理论.

五. 总结与思考

这一个月的复现实验恍惚间就过去了,从一开始的壮志凌云里,到前期的无从下手,再到中期的训练失败屡屡受挫,发现解决方案恍然大悟,再到最后的略感遗憾,意犹未尽,可以说是五味杂陈.只是时间有限,我最后一刻产生的许多想法都来不及去实现,才后悔之前没有学会正确地反复精读文章而漏了很多要点,才明白科学研究中严谨求实,脚踏实地的重要性.

在此期间,我得到的收获远不止一张填表的数据表格而已.我收获了部署 Machine Learning 几乎整个的工程流程,包括数据集的获取导入和预处理;神经网络模型的构建,训练和测试;GPU 服务器的远程部署配置,实例监控管理,多主机单用户的协同,单机多进程的协同;训练结果的可视化呈现,tensorboard 的使用,matplotlib 多类型图表的绘制,训练模型的保存与加载.这些看起来很基础的技能在实验之前我掌握得并不牢固,是实验驱使着我去克服困难,更新知识,丰富自己的技能储备.

我还收获了对联邦学习这一领域的入门认知.FlexiFed 这篇文章是我看的第一篇关于联邦学习的文章(在这之后我才去了解开山之作 FedAvg).第一次阅读时,我没有理解到什么是 Parameter Sever 以及它为什么要去对 Clients 做聚合,在经历了部署实验项目,数据集分区到每个 client,根据文章理解自己编写 Clustered-Common 方法,辨析 Global Training 和 Local Traing 的区别,调试查看每次 Training 之后模型参数的变化,尝试调整训练参数比较效果等等一系列流程后,我才渐渐领悟到 Parameter Sever 聚合模型参数是为了加强不同结构模型之间的知识(模型训练迭代时的权重参数)共享,把每个独立 Client 在同步迭代过程中"学习"到的数据特征融合起来,不至于陷入独立 Client 有限数据集导致的过拟合陷阱,聚合模型之后大家整体的"水平"都会提高,都达到更好的最终准确率.从通俗上来理解就是"异质结构,分头学习,同步交流,部分共享,共同进步"的一个过程,我觉得这体现了求同存异的思想,在 Max-Common 方法中最能够体现.

我觉得自己更大的收获在于体会到了对于边缘智能和联邦学习领域的探究兴趣.在我看来,它们同时涉及了网络安全,机器学习,并行计算等多个领域,有很大的挖掘潜力和应用前景.我想不只是模型性能,准确率和收敛速度的问题需要探索,比如:就像不同服务器的配置训练的速度不一样,实际场景下不同 Client 因为硬件软件等等的不同势必也会有不同的效率和性能，当发起申请的 Client 和 sever 规划的训练参与对象不对等时，sever 要用什么样的方法选择 Clients？还是基于不同 Client 的性能不同，目前 FlexiFed 的框架注定是要所有 Client 同步完成了当前的 Local Training 才会进行模型聚合，那么训练的很慢的 Client 势必就要落后于训练的很快的 Client，导致为了同步出现部分 Clients 等待很久的情况，造成资源时间的浪费，有没有好的方案可以阻止这一现象呢？

感谢老师能够给予这次机会接触这一全新领域，无比期待今年能跟随您的指导做有挑战的研究。如果您对此次实验有任何指正意见，我将感激不尽！