

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3日-7月13日

目录

一. 简述.....	1
二. FlexiFed 论文理解.....	2
三. 实验复现过程.....	3
1. 部署远程服务器.....	3
2. 第一次模型训练.....	5
3. 训练结果.....	9
四. 遇到的问题.....	13
1. 文章理解.....	13
2. 实验复现.....	14
五. 总结与思考.....	17

一. 简述

在第一周的学习研究中,我主要在做以下工作:

- 边缘计算领域的研究入门
- FlexiFed 文章的精读理解
- 开展第一阶段的复现任务(实现 FlexiFed 中的三种策略,包括 Basic-Common, Clustered-Common 和 Max-Common,复现在 FlexiFed 框架中用 CIFAR-10 数据集训练 VGG 模型获得的结果)

第一阶段任务完成度目前达到 **40%**,前期精力主要投入在了文献阅读上,后期一直开展实验复现进度会加快很多。

初步探索的路上我也遇到了不少瓶颈,在文章理解上,实验源代码运行上都遇到十分困惑棘手的问题。作为一名科研初学者,也是一名在边缘智能领域的初学者,我还存在许多知识何技能上的不足,请您指正。

二. FlexiFed 论文理解

在反复阅读过 FlexiFed 这篇文章后，我认为它的主要创新点在于真正打破了用户端 ML 模型结构异质化的壁垒，为加快联邦学习效率，促进用户间的知识共享提供了全新的思路，在实际应用上专注与异构结构模型聚合策略的 FlexiFed 相比与以往的 FedAvg, q-FedAvg, FedMA 等策略更加符合异质化结构普遍存在的市场现实，更具有实际应用意义。

下面是我根据对文章理解绘制思维导图：

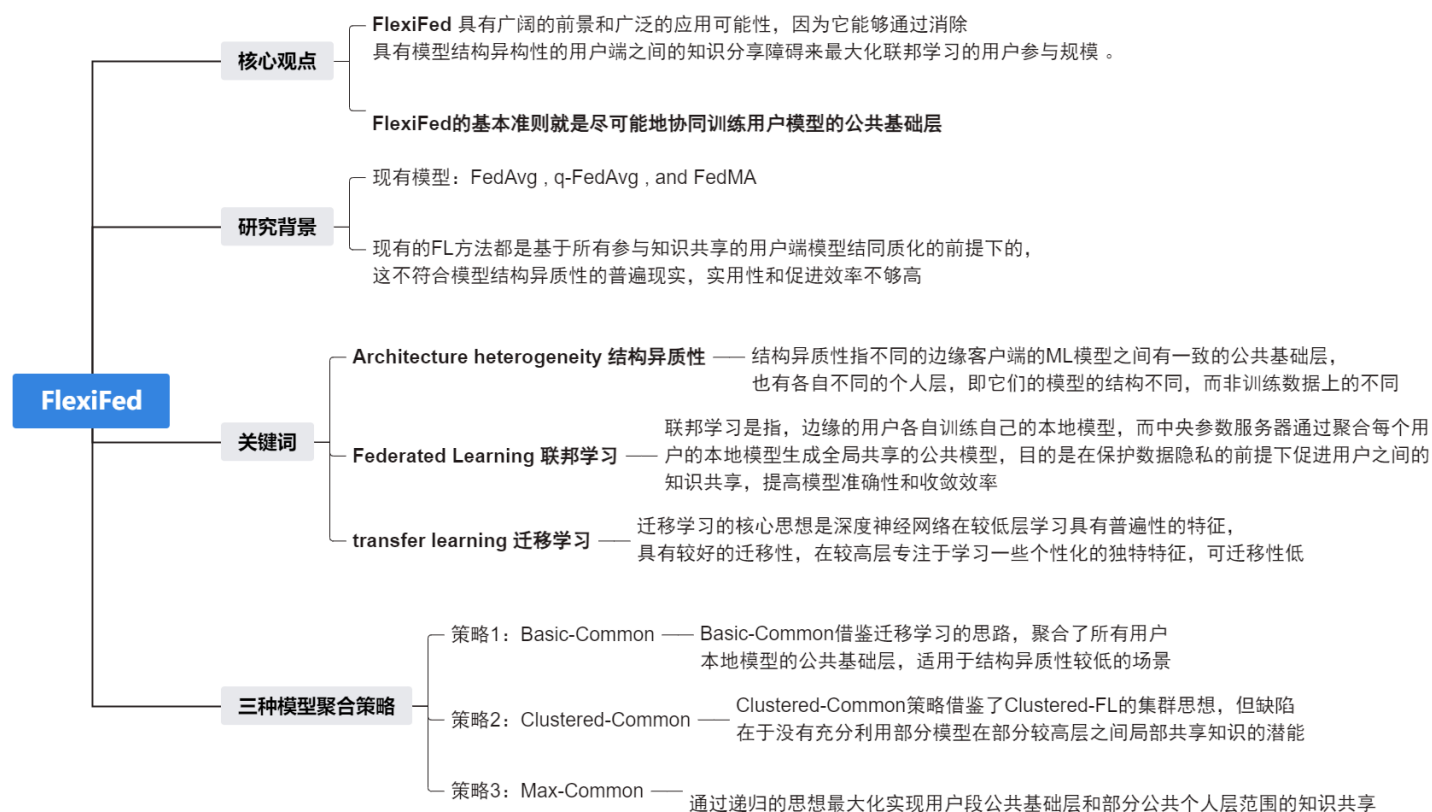


图 1 FlexiFed 思维导图

三. 实验复现过程

1. 部署远程服务器

因为我自己的主机没有独显只能实验 CPU 版本的 pytorch，经测试每一轮 epoch 时间大约 8min，耗时太长，所以选择连接远端服务器进行训练

1.1 获取 SSH 登录指令和密码



图 3.1 在 AutoDL 平台上获取登录指令和密码

1.2 SSH 连接

在 pycharm 上使用 SSH 服务远程连接服务器作为远程解释器。用户名必须选择 root。



图 3.2 正在连接到 SSH 服务器

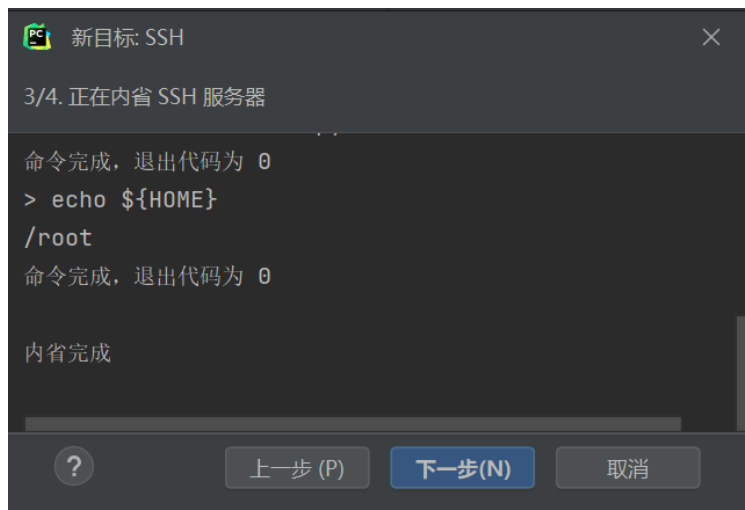


图 3.3 内省 SSH 服务器完成

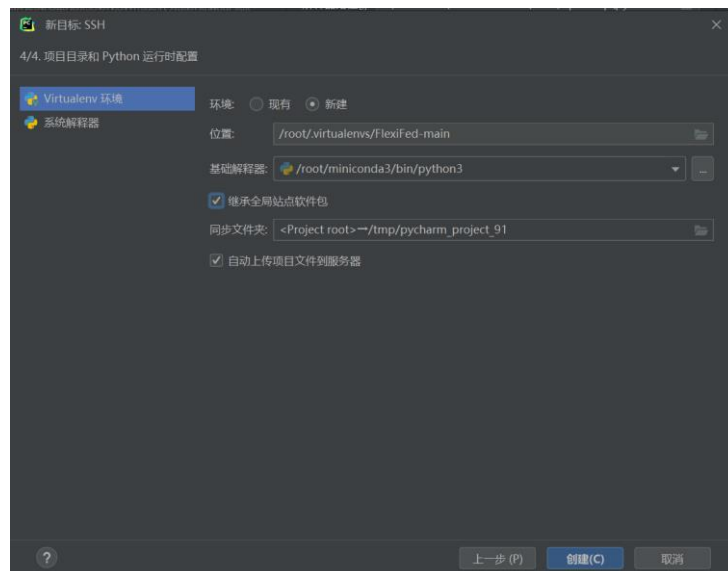


图 3.4 配置远程解释器和同步文件夹

配置完成并测试连接正常后，pycharm 会自动同步本地目录文件和远程段绑定链接的目录文件，与 github 仓库的文件同步类似。

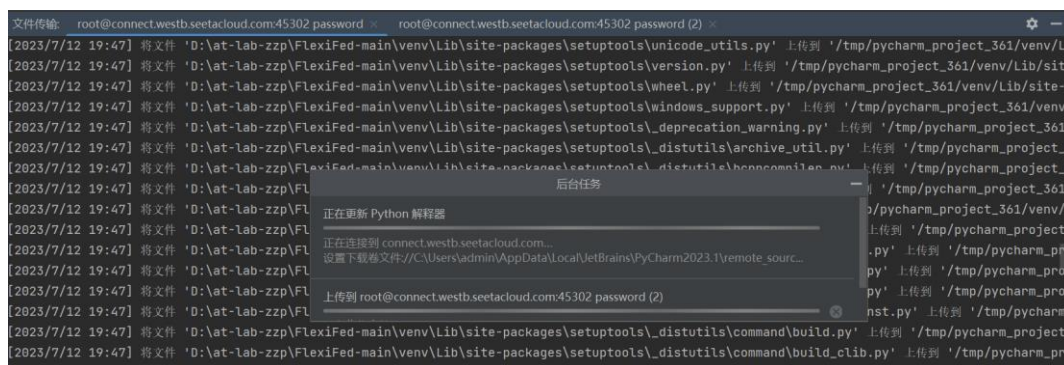


图 3.5 自动同步更新文件

2. 第一次模型训练

2.1 获取论文源码

源码地址: <https://github.com/fio1982/FlexiFed>

在原文 [5 EXPERIMENTS](#) 中给出

2.2 配置环境

AutoDL 云平台中配置的环境是 Pytorch=1.11, 而源码 Requirements 给出的环境依赖为:

```
torch==1.13
```

```
python==3.8
```

需要卸载现有的 torch 并重新下载

使用的 pip 命令为:

```
pip install torch==1.13.1+cu117
```

```
torchvision==0.14.1+cu117 torchaudio==0.13.1--extra-index-url
```

<https://download.pytorch.org/whl/cu117>

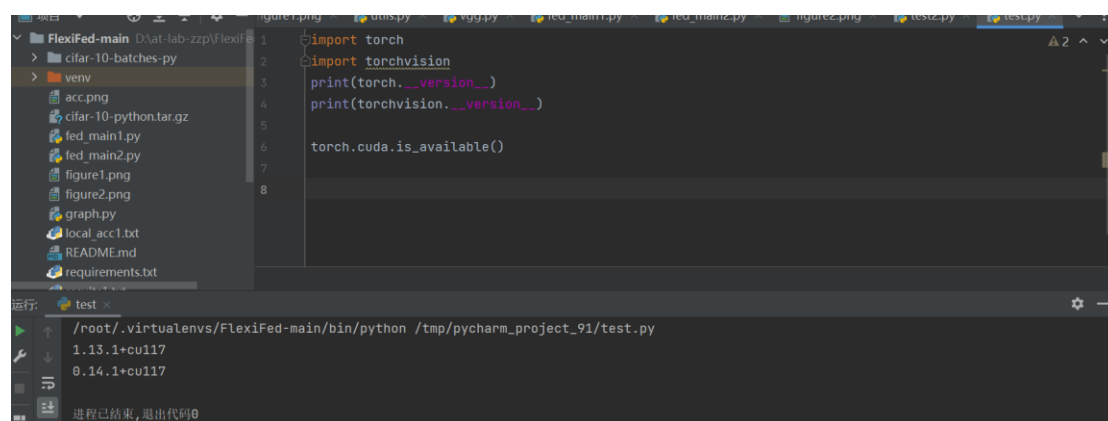


图 3.6 更新后 python 测试当前 torch 版本

2.3 修改源码

现阶段虽然能够理解实验中三种策略的基本思路，但是我对于深度学习的实践经验过于匮乏，源码的代码结构并不能完全看懂并加以完善，目前的修改工作只是运行后基于解释器的报错做出一些 bug 修改。

源码的基本结构为：

- `fed_main.py` 主程序, 包含模型的训练与聚合, `acc` 的记录
- `utils.py` 自定义的函数, 主要有数据集的导入与切割, `basic-common` 与 `clustered-common` 策略的实现, `FedAvg` 方法的实现(已有方法), 用户模型本地训练的实现, `acc` 与 `loss` 的获取
- `vgg.py` VGG 模型家族的实现, 包括有 `vgg-11`, `vgg-13`, `vgg-16`, `vgg-19` 以及各自的 `batch_normalization` (归一化)版本

我跟据编译器报错做出的改动主要有：

- 将 `from utils.utils import *` 改为 `from utils import *`

原因：只有 `utils` 模块, 没有 `utils.utils` 模块, 这段代码的目的应该是导入 `utils.py` 中的所有函数模块

- 将 `from models.vgg import *` 改为 `from vgg import *`

原因：源代码中没有 `models` 模块, 并且在终端使用 `pip` 安装 `models` 时显示失败如图 3.7, 考虑是可能代码编写问题, 其实目的是为了导入 `vgg.py`, 所以改为 `from vgg import *`


```
Preparing metadata (setup.py) ... error
error: subprocess-exited-with-error

* python setup.py egg_info did not run successfully.
| exit code: 1
└─> [8 lines of output]
Traceback (most recent call last):
  File "<string>", line 2, in <module>
  File "<pip-setuptools-caller>", line 34, in <module>
  File "/tmp/pip-install-bz1qubbz/models_0dfc039dc59741598c954ee58bee481d/setup.py", line 25, in <module>
    import models
  File "/tmp/pip-install-bz1qubbz/models_0dfc039dc59741598c954ee58bee481d/models/__init__.py", line 23, in <module>
    from base import *
ModuleNotFoundError: No module named 'base'
[end of output]

note: This error originates from a subprocess, and is likely not a problem with pip.
error: metadata-generation-failed

* Encountered error while generating package metadata.
└─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

图 3.7 models 安装失败截图

2.4 导入数据集

在修改完源码后直接运行时, 程序报错缺少数据集 cifar10, 建议我将

```
datasets.CIFAR10(data_dir, train=True, download=False,
transform=train_transform)
```

中的参数 download 改为 True, 这样就会直接从 url 中下载, 但是下载速度过于缓慢, 预计时间 2h, 所以采用了通过 pycharm 手动上传数据集的方法:

- 1) 找到链接 cifar-10 数据集的 url 下载到本地, 解压得到文件夹 cifar-10-batches-py
- 2) 将 cifar-10-batches-py 文件夹放到本地链接远程服务器的目录下
- 3) 上传到远程服务器同步目录
- 4) 数据集导入成功, 保持 datasets.CIFAR10 中的参数 download=False

2.5 在 screen 中离线训练

当直接在 pycharm 中借助远程服务器开展训练时,我理解中其实是在线训练的方式,每时每刻需要保持我的本地主机和远程服务器的连接训练过程才能有效开展,但是校园网经常半小时断开一次,一旦断开训练进程就会中断,无法重新开始进程,所以通过查阅资料学会了在 screen 中离线训练的方法,如下:

- 1) 打开远程服务器终端,进入同步目录文件夹,如:

```
cd /tmp/pycharm_project_91/
```

- 2) 创建一个带有日志的记录的新 screen

```
screen -L -dmS task1-7-11
```

日志写入.\screenlog.0,可以查看下载编辑

输入 screen -ls 可以查看当前所有 screen 以及各自的 id

- 3) 连接并进入该 screen

```
Screen -r [目标 screen id]
```

- 4) 在该 screen 下执行 python 程序

```
/root/.virtualenvs/FlexiFed-main/bin/python  
/tmp/pycharm_project_91/fed_main1.py
```

执行指令可以通过 pycharm 在线运行时查看到

- 5) 当发生连接断开时,可以重新连接回 screen,且进程不会被中断

```
Screen -r [目标 screen id]
```

3. 训练结果

3.1 任务总结

目前完成了 1 项任务的训练:在 CIFAR-10 数据集下基于 Max-Common 方法训练 VGG 模型,有 vgg11, vgg13, vgg16, vgg19 四个版本,共有 8 个用户端,每两个用户使用一个 vgg 模型版本,其中 max-common 的方法实现和 cifar10 数据集的导入源代码中已经实现.

3.2 数据分析

通过在源代码 fed_main.py 中执行写文件操作,我把运行结果写入了下列文件中:

results1.txt 记录每个 training round 的完成时间和周期性的 acc 记录

```
| Global Training Round : 1 |  
  
2023-07-12 16:14:04  
idxs_users: 0 local_accuracy latest: 0.09 epoch: 0  
idxs_users: 1 local_accuracy latest: 0.1 epoch: 0  
idxs_users: 2 local_accuracy latest: 0.12 epoch: 0  
idxs_users: 3 local_accuracy latest: 0.08 epoch: 0  
idxs_users: 4 local_accuracy latest: 0.09 epoch: 0  
idxs_users: 5 local_accuracy latest: 0.09 epoch: 0  
idxs_users: 6 local_accuracy latest: 0.09 epoch: 0  
idxs_users: 7 local_accuracy latest: 0.09 epoch: 0  
  
| Global Training Round : 2 |  
  
2023-07-12 16:14:19  
  
| Global Training Round : 3 |  
  
2023-07-12 16:14:32
```

图 3.8 results.txt

local_acc1.txt 只记录每 10 个 rounds 时的实时 acc 值

table1.txt 记录每个 rounds 后每个本地用户的 acc 值,用户绘制 acc 图表

screenlog.0 实时记录运行时 screen 的所有输出值,作为运行日志查看

复现任务总目标是为了复现文章中 table1 的内容, 当前进度如下:

数据集	模型	方法	版本			
			V1	V2	V3	V4
CIFAR-10	VGG	Basic-Common				
		Clustered-Common				
		Max-Common	0.72	0.77	0.78	0.8
	ResNet	Basic-Common				
		Clustered-Common				
		Max-Common				
CINIC-10	VGG	Basic-Common				
		Clustered-Common				
		Max-Common				
	ResNet	Basic-Common				
		Clustered-Common				
		Max-Common				
AG NEWS	CharCNN	Basic-Common				
		Clustered-Common				
		Max-Common				
	VDCNN	Basic-Common				
		Clustered-Common				
		Max-Common				
Speech Commands	VGG	Basic-Common				
		Clustered-Common				
		Max-Common				
	ResNet	Basic-Common				
		Clustered-Common				
		Max-Common				

图 3.9 总任务进度表

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

图 3.10 原文实验结果表

从上表可以看出,我复现的实验结果和原文的结果存在差异,V1-V4 四个版本的 VGG 模型结果都低于原文的结果,

我的结果为 0.72, 0.77, 0.78, 0.8,

原文的对应结果为 0.806, 0.852, 0.866, 0.868

(我运行时的 epoch 数也为 502),其中的原因目前还没有探究清楚.

下图 3.11 是我根据自己的结果绘制的四个版本 VGG 模型的 acc 曲线.

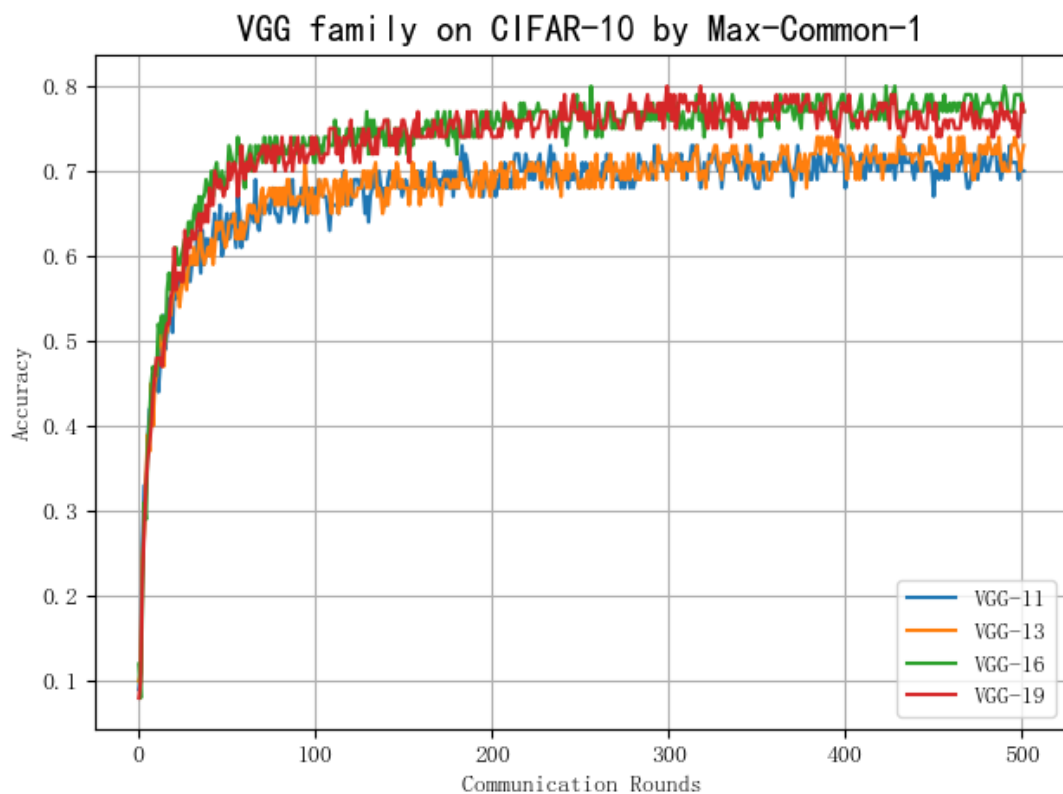


图 3.11 VGG on CIFAR-10 by Max-Common-1

曲线反应出 VGG16 与 VGG19 的准确性和收敛速度由于 VGG11 和 VGG13,但是并不能反应出文章想要表达的 FlexiFed 框架加强的用户之间知识共享的含义,需要进行更多实验.

四. 遇到的问题

1. 文章理解

1.1 basic-common 与 standalone 的比较

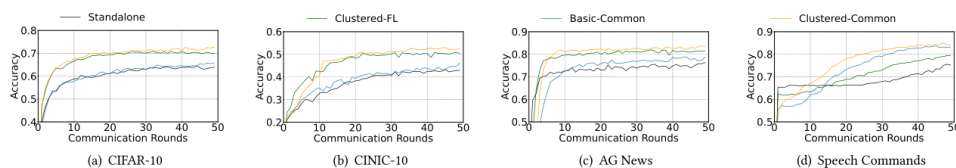


Figure 4: Comparison between Standalone, Clustered-FL, Basic-Common and Clustered-Common in local model convergence across eight clients: (a, b) VGG-{11, 13, 16, 19} trained on CIFAR-10 and CINIC-10 for image classification; (c) VDCNN-{9, 17, 29, 49} trained on AG News for text classification; (d) ResNet-{20, 32, 44, 56} trained on Speech Commands for speech recognition.

图 4.1 原文 figure4

如图 4.1 为原文 figure4, 原文描述如下:

“

We conducted experiments to validate the performance of Basic-Common. The experiment settings can be found in A.5 and the results are shown in Fig. 4. We can see that, compared with Standalone, clients' local models converge much faster to higher model accuracy.

”-from 4.1 Basic-Common Strategy

其中提到说 Fig. 4 能反应出与 Standalone 策略相比, 用户本地模型的准确性和收敛速度大大提高, 上下文可知这里是 Basic-Common 与 Standalone 的比较, 但是从图像观察的结果是, basic-common 与 standalone 相比, 在 CIFAR-10 和 CINIC-10 数据集的表现几乎一致, 在 AG News 上略有小优, 在 Speech Commands 上才有明显差异, 这样的实验结果是否有力地证明了原文的结论呢?

1.2 Clustered-Common 的复杂度分析

“

Here we briefly discuss the time complexity of the clustering process

in Clustered-Common. Given K clients in the FL system. The worse case is that there are no personal layers are the same, the time complexity of the clustering process is $O(K^2)$. The best case is that there are $K-1$ personal layers are the same, its time complexity is $O(K)$.

”-from A.2 Complexity

这段文字分析了 Clustered-Common 策略的时间复杂度, 其中说到最好的情况下是 K 个用户有 $K-1$ 个个人层是完全相同的, 此时复杂度为 $O(K)$, 这个结论是如何推导出来的?

2. 实验复现

2.1 basic-common 报错

第一次实验是, 我在改完调用上的 bug 后直接运行了 `fed_main.py`, 此时聚合策略采用的是 `max-common`, 在代码的最后给出

```
modelAccept = common_max(modelAccept)
```

当我把策略改为 `common_basic`, 即

```
modelAccept, _ = common_basic(modelAccept)
```


在运行到 round 2 时发生报错, 信息如下:

```
| Global Training Round : 2 |  
  
2023-07-13 19:19:17  
Traceback (most recent call last):  
  File "/tmp/pycharm_project_91/fed_main1.py", line 96, in <module>  
    model.load_state_dict(modelAccept[idx])  
  File "/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/module.py", line 1657, in load_state_dict  
    load(self, state_dict)  
  File "/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/module.py", line 1639, in load  
    module._load_from_state_dict(  
  File "/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/module.py", line 1593, in _load_from_state_dict  
    if key.startswith(prefix) and key != extra_state_key:  
AttributeError: 'int' object has no attribute 'startswith'  
  
进程已结束, 退出代码1
```

图 4.2 basic-common 运行报错

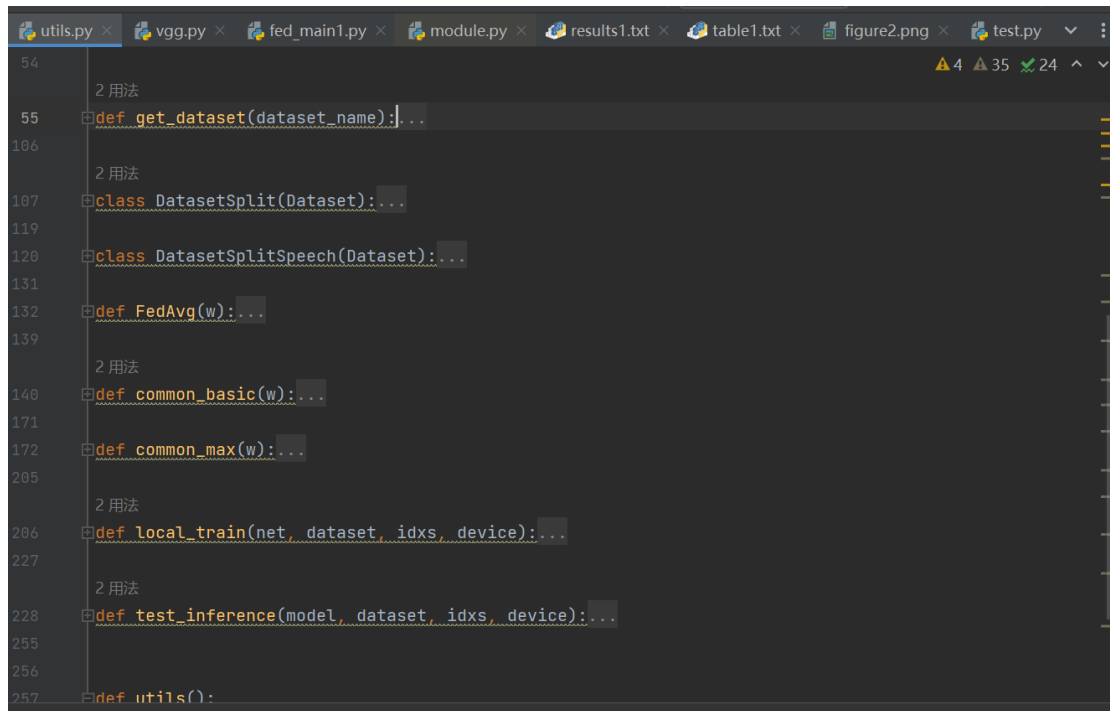
信息显示 int 对象是没有 startswith 类的, 我的推断是:

在 python 中只有 str 类型才有 startswith 类, 说明 state_dict 对象中的某一个 key 没有实际值, 为空, 被默认当成了 int 类型, 进一步反推表明, 在执行 model.load_state_dict 函数时, 传入的模型 modelAccept[idx] 下的参数 state_dict 存在缺省值, modelAccept[idx] 是在整个过程中反复更新的, 除了第一次是使用的直接赋值, 之后的更新都是采用深拷贝方法复制的 localModel 即用户本地训练后的模型参数. 我想到的方法是在 model.load_state_dict 函数中加入参数 strict=False, 采用非严格对应的方法载入参数, 只加载函数参数里有的参数值, 可能避免上述情况.

但是结果是并没有实际效果, 目前该问题尚未解决, 影响了后续采用 basic-common 策略的实验的开展.

2.2 缺少 Clustered-Common 方法的实现

在源代码目录中, 我在 utils.py 里查看到了 Basic-Common 和 Max-Common 方法的实现, 但是没有 Clustered-Common 方法的实现, 只有 FedAvg 的实现, 如图 4.3.



```
54
55 def get_dataset(dataset_name):...
106
107 class DatasetSplit(Dataset):...
119
120 class DatasetSplitSpeech(Dataset):...
131
132 def FedAvg(w):...
139
140 def common_basic(w):...
171
172 def common_max(w):...
205
206 def local_train(net, dataset, idxs, device):...
227
228 def test_inference(model, dataset, idxs, device):...
255
256
257 def utils():
```

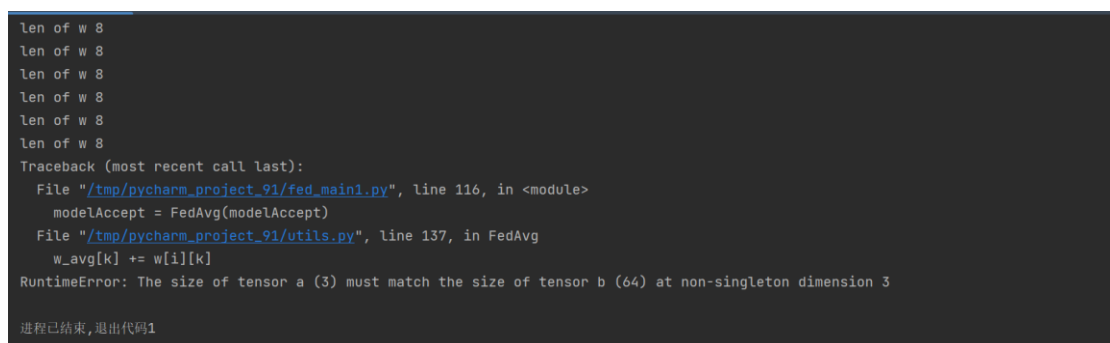
图 4.3 utils.py 文件内容

从原文可知, Clustered-Common 方法是从 FedAvg 中借鉴得来的, Clustered-Common 在公共基础层与 Basic-Common 一致, 在个人层采用的 FedAvg 一样的方法, 两者相近但是还是有差别.

如果直接将 `modelAccept = common_max(modelAccept)`

改为 `modelAccept = FedAvg(modelAccept)`

执行后仍会报错, 显示张量之间不匹配



```
len of w 8
len of w 8
len of w 8
len of w 8
len of w 8
len of w 8
len of w 8
Traceback (most recent call last):
  File "/tmp/pycharm_project_91/fed_main1.py", line 116, in <module>
    modelAccept = FedAvg(modelAccept)
  File "/tmp/pycharm_project_91/utils.py", line 137, in FedAvg
    w_avg[k] += w[i][k]
RuntimeError: The size of tensor a (3) must match the size of tensor b (64) at non-singleton dimension 3
进程已结束,退出代码1
```

图 4.4 直接执行 FedAvg 方法报错

五. 总结与思考

总的来讲, 目前我对于这篇文章的理解还比较浅显, 只是了解它运用了哪三种方法实现了异质结构模型的知识共享, 理解到了伪代码, 但是真正上手实现源码还是出现了非常非常多的问题和疑惑, 主要来自于我对联邦学习这一方向了解不足和对深度学习的一些基本技能掌握不够全面, 对于整个边缘计算, 边缘智能领域都还只是入门都不够的程度.

在研究复现 basic-common 和 clustered-common 方法时, 我以前的缺陷才暴露出来: 曾经的入门科研和课程实验工作时, 我都是"面向调库的机器学习", 搜搜模型-下载依赖-调库-运行-完成, 是搭积木式的完成程序, 所有的核心训练工作都是调用现有的框架函数赋值后一步到位, 自己从来没有深入的实现理解深度神经网络的核心知识. 一到目前复现时, 一旦报错我就会陷入迷茫: FedAvg 的实现应该只是个人层做模型聚合的方法, 实际的 Cluster-Common 应该还需要再加入一段聚合全体公共层的过程, 这一部分的编写我犯了难; 调用 Basic-common 的过程中, 我发现报的错误在 Max-Common 时并没有出现, 比较两者的不同时, 它们的实现源码我还没有完全理解透彻, 一些处理并没有弄清楚它们的用意, 也就不知道该如何处理这个 bug, 所以将论文的思路运用到代码的实现上我还没有掌握这项能力.

希望老师能帮助我指导在文章理解和实验复现中的一些问题, 点拨一条正确的路径, 让我能够在接下来的进程中顺利完成工作, 更重要的是真正入门联邦学习, 边缘智能的研究领域, 为以后的研究生活打下坚实基础.