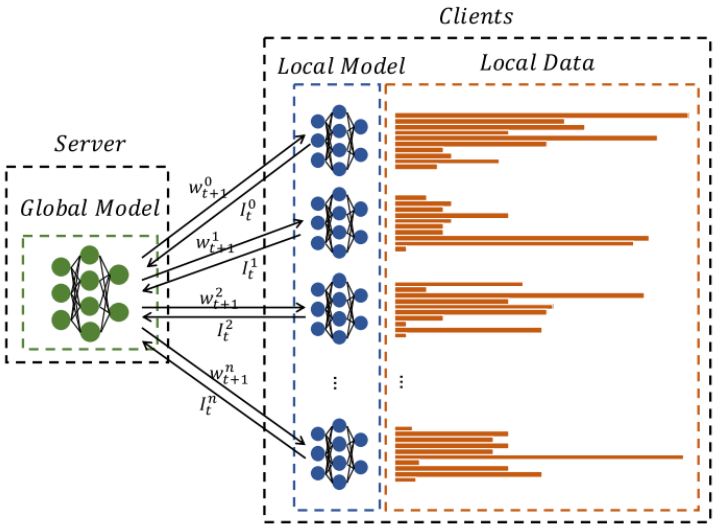
论文分享

题目：Communication-Efficient Federated Learning with Compensated Overlap-FedAvg

引言

FedAvg4 个步骤：
一、中央服务器初始化全局模型并将全局模型推送到每个参与的客户机。
二、客户机根据其本地数据对接收到的全局模型进行训练。之后它们将有利于全局模型训练的基本信息l_t^k（即，大部分是模型权重，但可以是梯度）返回到服务器，其中 k 是客户端的索引，t 分别表示迭代索引。
三、 中央服务器通过所有客户端的信息$\sum k l_t^k$来更新全局模型。
四、重复步骤二和步骤三，直到收敛。



FedAvg 缺点：
如今模型的大小很容易达到数百兆字节甚至千兆字节，再加上联邦学习中的模型通常需要几十万次迭代才能收敛，训练过程中的总体数据传输大小可能会超过 PB。因此，与非隐私保护训练相比，联邦学习通常由于耗时的通信阶段而需要更多的时间来收敛，并且当网络条件相对较差时，它可能会变得更慢。

动机：
由于 FedAvg 每个 Epoch 的同步数据与模型的大小相同，引入了大量的通信开销，从而导致通信效率低下。为此，该文提出了一个将模型训练阶段与模型上传、下载阶段并行的框架 Overlap-FedAvg，使模型上传、下载阶段可以完全覆盖模型训练阶段。与普通的 FedAvg 相

比，Overlap-FedAvg 进一步发展了分层计算策略、数据补偿机制和 Nesterov 梯度加速算法。此外，Overlap-FedAvg 与其他许多压缩方法是正交的，可以将它们一起应用，最大限度地利用集群。

模型设计

Overlap-FedAvg

Overlap-FedAvg workflow：

一、在中央服务器上启动全局模型，并将其推送到每个参与的客户端。

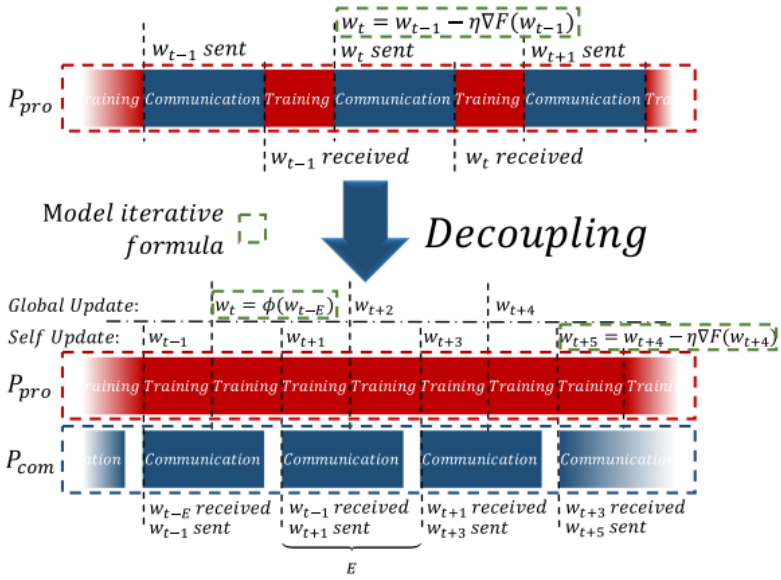
二、每个参与的客户对各自的数据进行本地模型培训的迭代。请注意，这里的 E 不再是需要手动设置的常量超参数，而是一个动态变量，可以适应不断变化的环境。

三、本地训练完成后，客户端立即继续本地训练下一个迭代，同时命令**另一个进程**将其本地模型推送到中央服务器。

四、当中央服务器接收到客户机的改进模型时，它首先**使用泰勒级数展开和 fisher 信息矩阵对模型权重进行补偿，然后使用补偿后的权重计算 nesterov 动量并更新全局模型**。

五、中央服务器将最新的全局模型推送到参与的客户端。

对于体系结构，overlap-FedAvg 和普通 FedAvg 之间的最大区别是第三步，在这一步中，我们放松了链式(当前迭代依赖于上次迭代结果)需求，这意味着中央服务器不能保证从每个客户端接收最新的模型，这不可避免地给我们的设置带来了陈旧性问题。因此，第四步设法解决了陈旧模型权重的问题。



Communication 过程：

在 fed-avg 中的联合学习开始时，每个客户机将初始化两个进程：一个专注于本地模型培训的过程（表示为 Ppro）和一个致力于通信的进程（表示为 Pcom）。

在每次迭代中，Pcom 将在 Ppro 尚未开始训练之前获取模型（即预下载全局模型），在本地模型得到改进后，它还将处理模型上传任务，这样 Ppro 就可以立即继续本地模型训练下一次迭代，而无需关心 communication.

当 Pcom 通信时，Ppro 像普通 Fed-avg 一样，利用基本 SGD 及其本地训练数据来更新本地模型。另一方面，当 Pcom 完成通信并接收到最新的模型时，与普通 FedAvg 不同，普通 FedAvg 只是将加权聚合计算为新的全局模型，而重叠 FedAvg 使用函数ϕ更新全局模型，这将在后一节中详细讨论。重叠 FedAvg 的伪码如算法 2 所示。

Algorithm 2. `Overlap-FedAvg`

Central server do:

Initialization: global model w_0.
for each global iteration $t \in 1, \ldots, iteration$ **do**
  for all each client $k \in 1, \ldots, N$) **do in parallel**
    # Get clients improved model.
    $TrainLocally(k, w_t)$
  end for
  if p_{com} $received\ models$ **then**
    # Update the global model.
    $w_{t+1} = \phi(w_t, p_0, w^0_{t-E}, p_1, w^1_{t-E}, \ldots, p_N, w^N_{t-E})$
  end if
end for

TrainLocally(k, w_0**):**

while p_{com} $is\ communicating$ **do**
  # Do self update
  $w_e \leftarrow w_{e-1} - \eta \nabla F(w_{e-1})$
end while
# Command p_{com} to send latest model
$p\_com\_send(w_E)$

在普通 FedAvg 中使用基本 SGD 进行模型迭代，这同时限制了训练阶段与通信阶段的并行，因为训练阶段需要通信阶段传输的模型权重。为了将训练阶段和通信阶段解耦，Overlap FedAvg 放松了这一限制，并在训练过程中采用了两个模型迭代公式：自更新和全局更新，其中前一个公式在尚未收到全局模型更新所需的模型权重时使用，当成功接收到所需的模型权重时，使用后一种方法。

数据(梯度)补偿机制

Fed-avg 的 w 计算公式

$$w_{t+1} = \underbrace{\sum_{k=0}^{N} p_k w_{t+1}^k = w_t - \overbrace{\eta \sum_{k=0}^{N} p_k \sum_{e=0}^{E-1} \nabla F(w_{t,e}^k)}^{w_{t+1}^k = w_t - \mu \sum_e \nabla F(w_{t,e}^k)},}_{Global\ Model\ Updating} \tag{1}$$

Overlap-FedAvg 的 w 计算公式

$$w_{t+1} = \phi(w_t, p_0, w_t^0, p_1, w_t^1, \ldots, p_N, w_t^N), \tag{2}$$

ϕ函数的目标是生成尽可能在同一时间戳接近等式（1）的结果，等式二可以被分解为

$$w_{t+1} = \phi(w_t, \ldots, p_k, w_t^k, w_{t-1}^k, \ldots, \sum_e \nabla F(w_{t-1,e}^k), \ldots). \tag{3}$$

从公式可以看出 w 计算存在过时问题，作者利用泰勒级数展开和 fisher 信息矩阵来缓解梯度过时问题。

批注 [1]: 数学原理有点复杂，没看明白

Nesterov 梯度加速算法(NAG 算法)

NAG 算法：带有动量的 SGD 算法，简而言之用于加速联邦学习过程
借用 DNN 中的 Nesterov 梯度加速算法，移植到 overlap-fedavg

Since we already had the unbiased estimation of $\nabla F(w_t)$ with $\nabla F(w_{t-1})$, the NAG updating rule in Overlap-FedAvg could be written as

$$\begin{cases} v_{t+1} &= \beta v_t + \nabla F^{ah}(w_{t-1}) \\ & \quad +\beta(\lambda \nabla F(w_{t-1}) \odot \nabla F(w_{t-1}) \odot (w_t - w_{t-1})) \\ w_{t+1} &= w_t - \eta v_{t+1}, \end{cases} \tag{8}$$

实验分析

1. 比较重叠 FedAvg 与普通 FedAvg 的效率

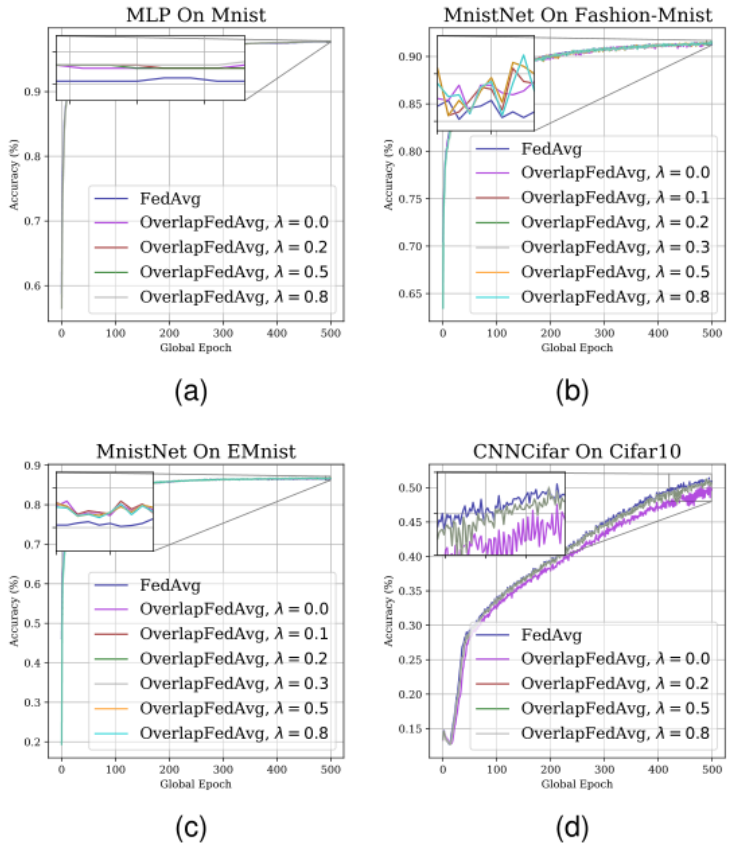


Fig. 5. (a) The accuracy curve of MLP trained on Mnist. (b) The accuracy curve of MnistNet trained on Fashion-Mnist. (c) The accuracy curve of MnistNet trained on EMnist. (d) The accuracy curve of CNNCifar trained on Cifar10.

TABLE 2
Accuracy Comparison Between Vanilla FedAvg and Overlap-FedAvg With Different λ and Fixed $\beta = 0.2$

Model	dataset	η	FedAvg	Overlap-FedAvg			
			Accuracy / PPL	$\lambda = 0.0$ Accuracy / PPL	$\lambda = 0.2$ Accuracy / PPL	$\lambda = 0.5$ Accuracy / PPL	$\lambda = 0.8$ Accuracy / PPL
MLP	Mnist	0.001	0.9711	0.9776	0.9775	0.9775	0.9777
MnistNet	Fmnist	0.001	0.9130	0.9146	0.9145	0.9153	0.9143
MnistNet	EMNist	0.001	0.8661	0.8672	0.8676	0.8673	0.8676
CNNCifar	Cifar10	0.001	0.5088	0.4970	0.5060	0.5058	0.5058
VGG^R	Cifar10	0.0001	0.4321	0.4272	0.4247	0.4248	0.4247
$ResNet^R$	Cifar10	0.0001	0.4356	0.4341	0.4348	0.4345	0.4345
$ResNet^R$	Cifar100	0.0001	0.0895	0.0865	0.0866	0.0866	0.0866
Transformer	Wikitext-2	0.0001	547.067	546.966	546.921	546.920	546.920
				$\beta = 0.0$			

2. 训练速度比较

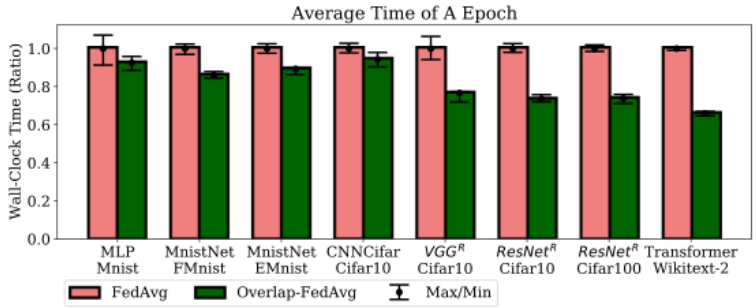


Fig. 8. The average wall-clock time for one iteration of federated learning in MLP on Mnist, MnistNet on FMnist, MnistNet on EMnist, CNNCifar on Cifar10, VGG^R on Cifar10, $ResNet^R$ on Cifar10, $ResNEt^R$ on Cifar100 and Transformer on Wikitext-2. We can see that the Overlap-FedAvg saved more time with the increase of the model size.

TABLE 4
The Average Wall-Clock Time for One Iteration of Federated Learning

Model	dataset	Parameters	Time / Iteration (Second)	
			FedAvg	Overlap-FedAvg
MLP	Mnist	199,210	31.2	**28.85** (↓7.53%)
MnistNet	FMnist	1,199,882	32.96	**28.31** (↓14.11%)
MnistNet	Emnist	1,199,882	47.19	**42.15** (↓10.68%)
CNNCifar	Cifar10	878,538	48.07	**45.33** (↓5.70%)
VGG^R	Cifar10	2,440,394	64.4	**49.33** (↓23.40%)
$ResNet^R$	Cifar10	11,169,162	156.88	**115.31** (↓26.50%)
$ResNet^R$	Cifar100	11,169,162	156.02	**115.3** (↓26.10%)
Transformer	Wikitext-2	13,828,478	133.19	**87.9** (↓34.0%)

总结

1. 利用数学原理和物理知识应用到联邦学习，来补偿自己提出的方案的不足
2. 对于精度方面的实验做的十分详细，列举很多参属下、模型下、集群规模下的不同情况
3. 不足：缺少超参数 λ 的自适应选择方案