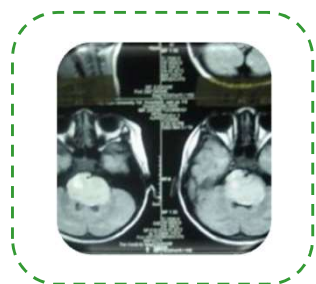


Case V: Distributed Model Reuse: FedPAN

主讲教师：詹德川

Three horizontal bars at the bottom of the slide: a light pink bar, a light red bar, and a light green bar.

然而在很多实际任务场景下，数据呈现“孤岛式”分布：



设备1



设备2

...



设备K

某些数据因为**隐私保护**限制而呈现孤岛分布，例如在医疗、企业、军事领域

某些数据因为**传输限制**而呈现孤岛分布，例如在无线网络情况下的便携式设备



设备1



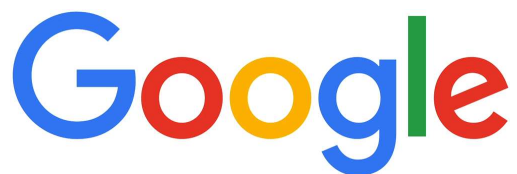
设备2

...



设备K

联邦学习的目的是在数据天然分散的情况下协同多方参与计算，有组织、有协作、有目的地完成传统的机器学习任务

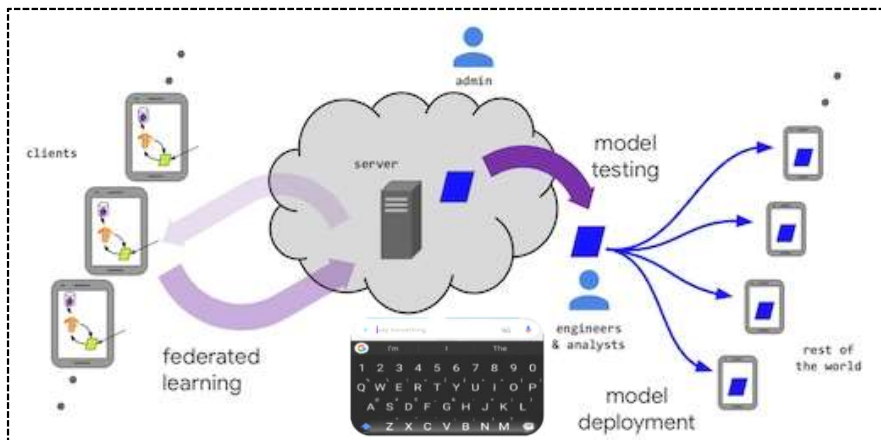


WeBank 微众银行

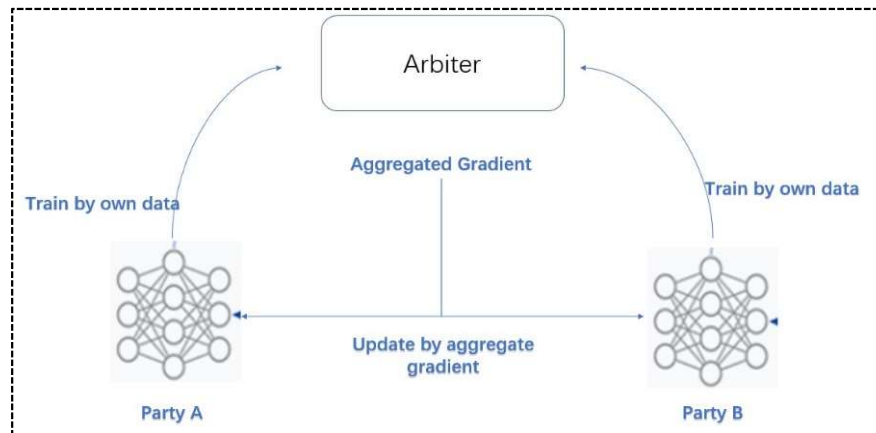
中国平安
PING AN
保险 · 银行 · 投资



腾讯研究院 Tencent Research Institute



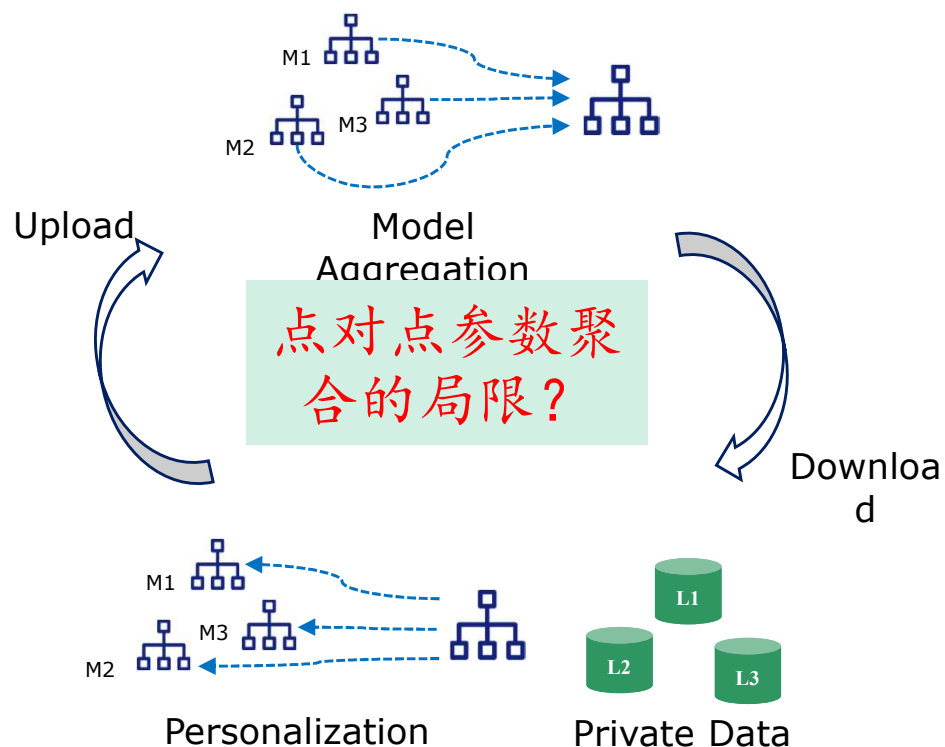
谷歌虚拟键盘**Gboard**应用：上千万边缘设备需要对用户的输入习惯进行建模并用户的下一个输入进行预测



微众银行跨银行反洗钱应用：900多种金融活动特征，单边模式训练只能获得有限洗钱行为数据，联合多方一起训练

局部训练过程：客户端下载全局模型，在本地数据上更新多轮

服务器训练过程：收集客户端更新之后的模型，在服务器执行聚合操作



Algorithm 1 FederatedAveraging. The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

```
initialize  $w_0$ 
for each round  $t = 1, 2, \dots$  do
   $m \leftarrow \max(C \cdot K, 1)$ 
   $S_t \leftarrow$  (random set of  $m$  clients)
  for each client  $k \in S_t$  in parallel do
     $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$ 
   $w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k$ 
```

点对点参数聚合

```
ClientUpdate( $k, w$ ): // Run on client  $k$ 
 $\mathcal{B} \leftarrow$  (split  $\mathcal{P}_k$  into batches of size  $B$ )
for each local epoch  $i$  from 1 to  $E$  do
  for batch  $b \in \mathcal{B}$  do
     $w \leftarrow w - \eta \nabla \ell(w; b)$ 
  return  $w$  to server
```

本地训练

目录

□ 项目背景

□ 痛点难点分析

□ 现有解决方案

□ 提出的解决方案

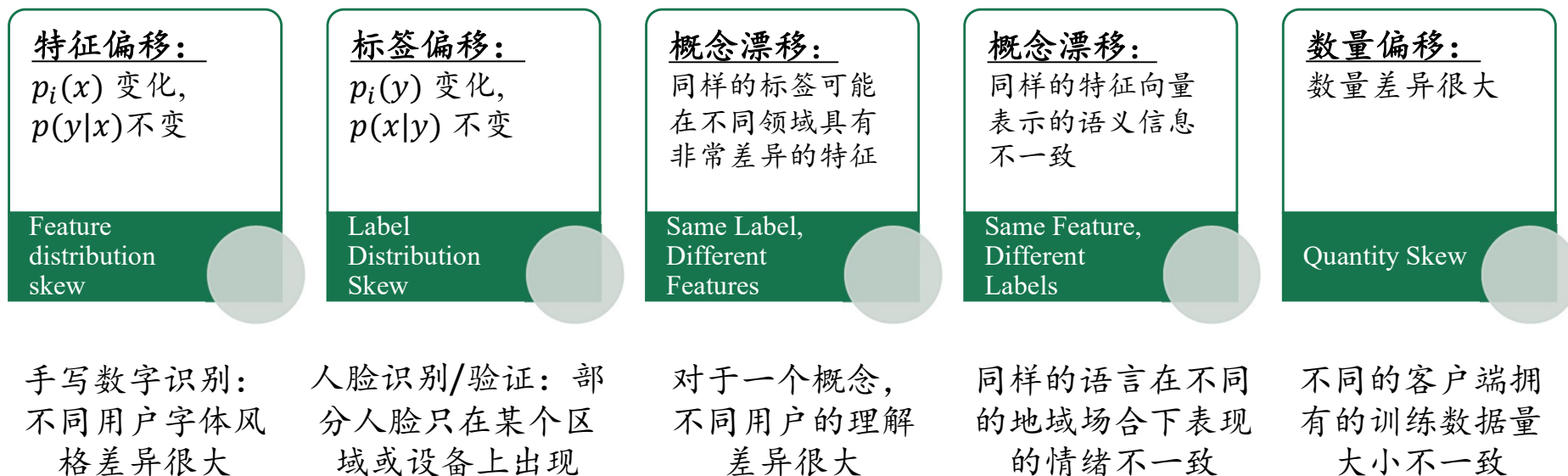
□ 可扩展方向

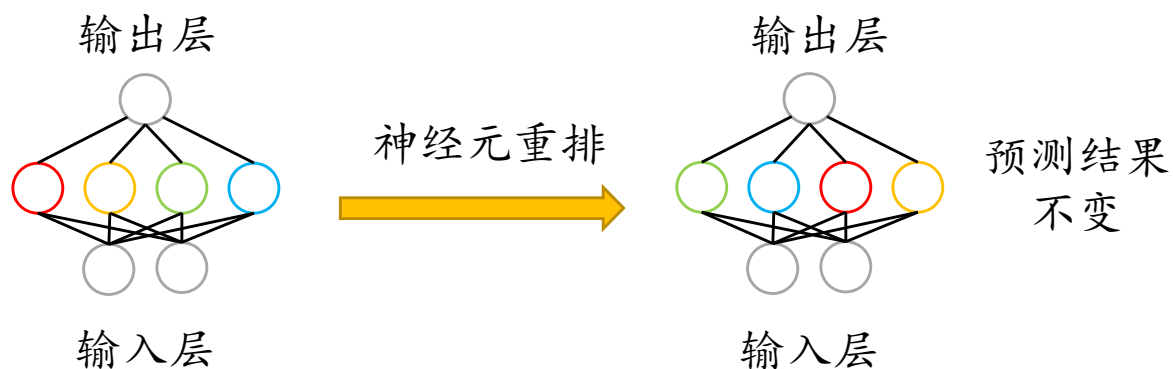
□ 技术落地应用

联邦学习面临的挑战：

- 隐私保护安全：隐私泄露；恶意攻击...
- 系统层级异构：有限传输、计算和存储；不稳定；设备众多...
- 统计层面异构：数据、模型、任务、目标异构...

不同用户、设备等参与方所在的环境是天然异质的，导致联邦学习中的数据是非独立同分布的，即 **Non Independently and Identically Distributed (Non-IID)**



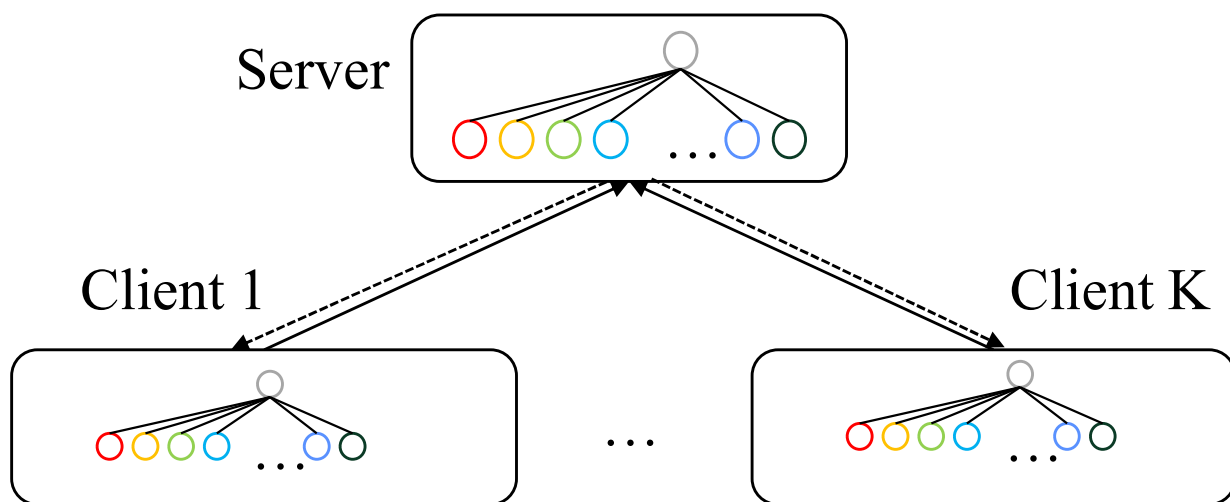


$$y = W_2 f(W_1 x)$$

$$y = W_2 I f(W_1 x)$$

$$I = \Pi \Pi^T$$

Π 是重排矩阵，有很多种重排方式，可以使得神经元位置交换之后输出不发生变化



由于训练的随机性，即便是同一模型在相同数据分布上训练，神经元也会发生错位现象，但不影响最终输出结果

再加上本地数据的NonIID分布，参数之间更难点点对点对齐

实验现象：

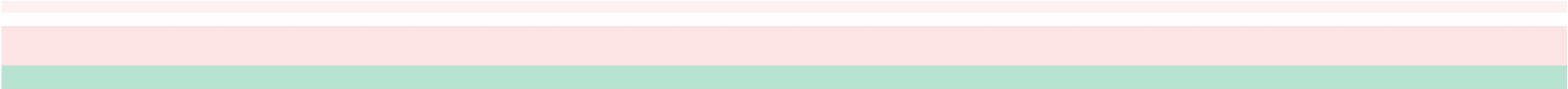
实验数据集：SVHN, CIFAR-10, CIFAR-100

网络结构：VGG11-BN, ResNet20, ResNet20

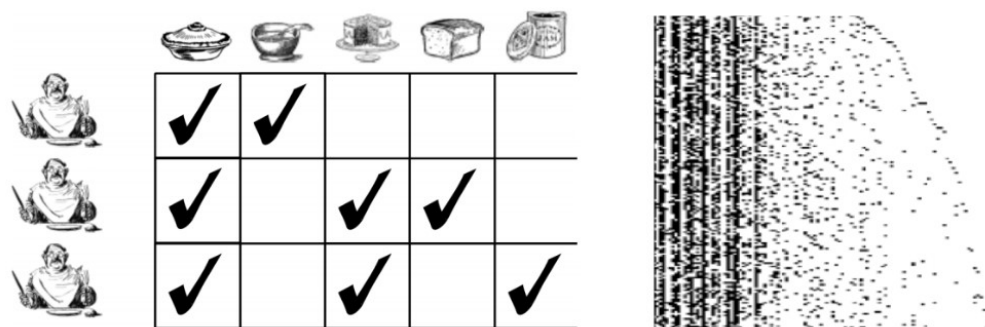
现象描述：FedAvg采用点对点参数平均的方法对客户端模型进行聚合，随着本地数据Non-IID程度的加重（实验数据中由 α 控制，越小代表越不均衡），点对点参数聚合的性能下降非常严重

	数据集中 训练	数据分布式训练 α			
		10.0	1.0	0.5	0.1
SVHN	94.59	61.08	54.26	46.56	17.68
CIFAR-10	92.31	83.77	82.73	82.18	77.42
CIFAR-100	69.10	44.00	34.40	23.92	14.65

目录

- 项目背景
 - 痛点难点分析
 - 现有解决方案**
 - 提出的解决方案
 - 可扩展方向
 - 技术落地应用
- 
- The bottom of the slide features three horizontal bars of equal width. The top bar is light pink, the middle bar is a slightly darker shade of pink, and the bottom bar is a muted green color.

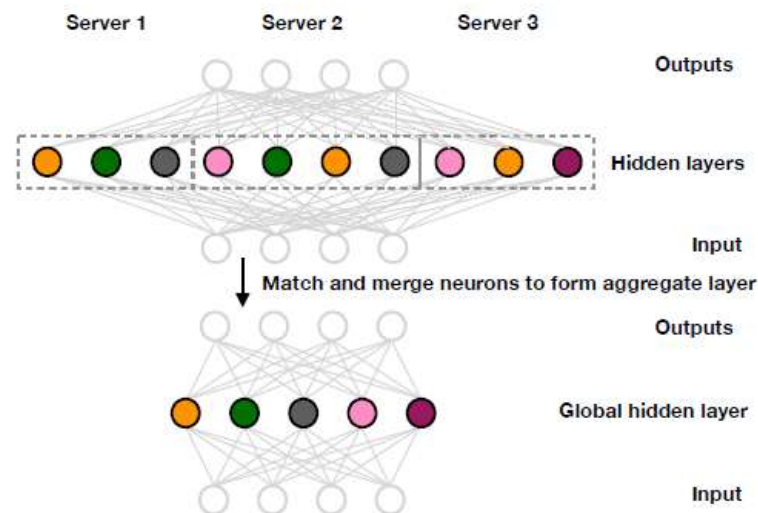
使用中国餐馆过程(Chinese Restaurant Process, CRP)和印度自助餐过程(Indian Buffet Process, IBP)对各个客户端神经元进行匹配, 然后“对号入座”进行聚合



IBP示意图

$$\begin{aligned} \arg \max_{\{\theta_i\}, \{B^j\}} P(\{\theta_i\}, \{B^j\} | \{\mathbf{v}_{jl}\}) \\ \propto P(\{\mathbf{v}_{jl}\} | \{\theta_i\}, \{B^j\}) P(\{B^j\}) P(\{\theta_i\}). \end{aligned} \quad (3)$$

贝叶斯建模过程



Bayesian Nonparametric Federated Learning of Neural Networks. ICML 2019.

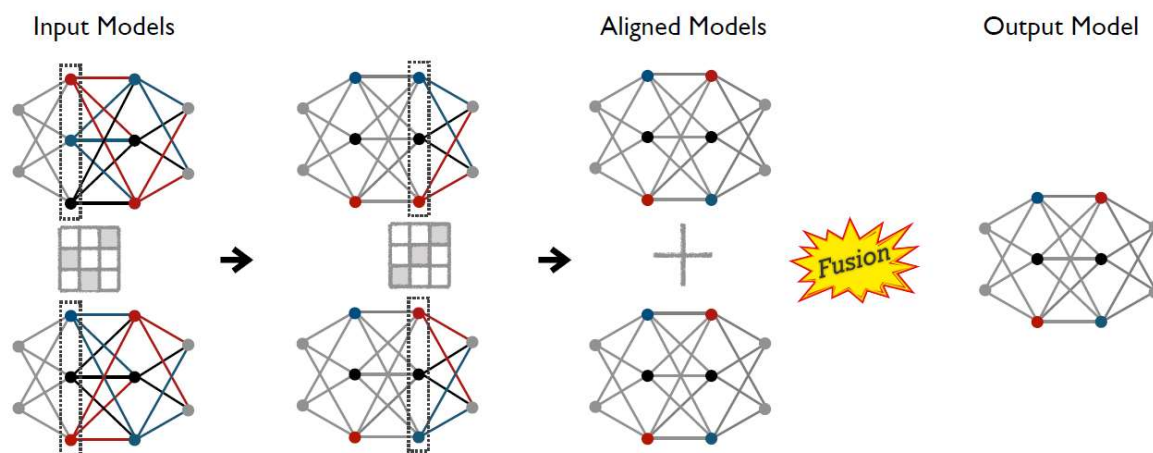
Federated Learning with Matched Averaging. ICLR 2020.

Indian Buffet Process



使用最优传输(Optimal Transport)对神经元对齐

- 联邦学习中神经元发生错位怎么办？
- 如果想聚合不是相同初始化的模型，怎么办？ -- 对齐神经元



Step 1: 获得神经元表示:

- 参数层面：链入参数、链出参数，维度为 d_k 或者 d_{k+1} 或者 $d_k + d_{k+1}$
- 激活值层面：一批数据的输出值，维度为 B

Step 2: 使用Optimal Transport对齐神经元参数，然后聚合

Optimal Transport

最优传输问题最早是由法国数学家蒙日（Monge）于1780年代提出；其解的存在性被俄国数学家Kantorovich证明，由此Kantorovich获得1975年诺贝尔经济奖；法国数学家Brenier建立了最优传输问题和凸函数之间的内在联系。

我们先介绍最优传输问题的数学提法：给定欧式空间中的区域 $U, V \subset \mathbb{R}^n$ ，分别配有概率密度 μ, ν ，总测度相同，

$$\int_U \mu = \int_V \nu。$$

考察一个微分同胚 $f: U \rightarrow V$ ，我们说这一映射是保测度的，如果对于任意一个博莱尔集合 $E \subset V$ ，其原像为 $f^{-1}(E) \subset U$ ，它们测度相同，

$$\int_{f^{-1}(E)} \mu = \int_E \nu。$$

保测度的微分形式如下：假设 U, V 的局部坐标分别是 $(x, y), (u, v)$ ，那么成立方程：

$$\mu(x, y) dx dy = \nu(u, v) du dv = \nu \circ f(x, y) J(x, y) dx dy，$$

这里雅克比矩阵为

$$J(x, y) = \begin{vmatrix} \partial u / \partial x & \partial u / \partial y \\ \partial v / \partial x & \partial v / \partial y \end{vmatrix}。$$

如果映射是保测度的，我们将其记为 $f_{\#} \mu = \nu$ 。

映射 $f: U \rightarrow V$ 的传输代价定义为

$$C(f) := \int_U |p - f(p)|^2 \mu(p)。$$

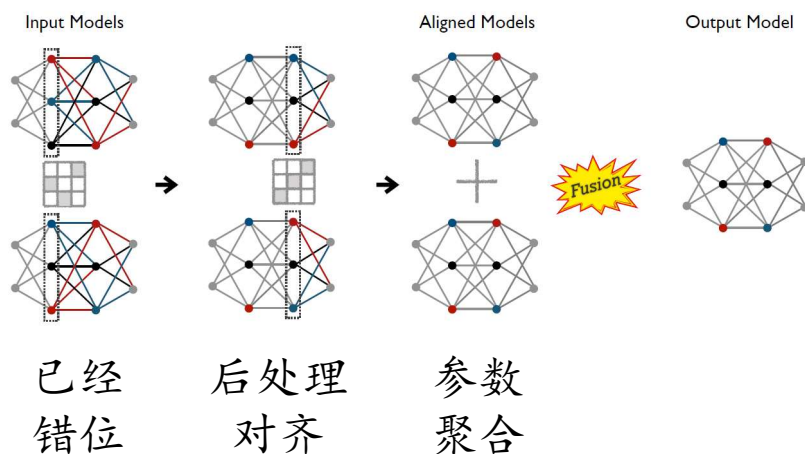
最优传输映射就是所有保测度的映射中，使得传输代价最小者：

$$f_0 := \arg \min_{f_{\#} \mu = \nu} C(f)。$$

我们给出一个粗浅的例子来解释最优传输问题。假设 U 是整个美国领土，概率密度 μ 是美国每英亩土豆年产量， ν 是美国每英亩土豆年消耗率。美国政府需要制定一个土豆运输方案，将土豆由乡村产地运输到城市消耗地，记为 $f: U \rightarrow U$ 。运输方案需要满足供需平衡条件，对于任意一座城市，其土豆年消耗总量等于其供应地土豆年生产总量，换言之，映射 f 是保测度的；同时，土豆运输方案使得运输成本最小。因此，政府所寻求的最优传输方案就是最优传输映射。由此可见为什么康塔洛维奇（Kantorovich）获得诺贝尔经济学奖。

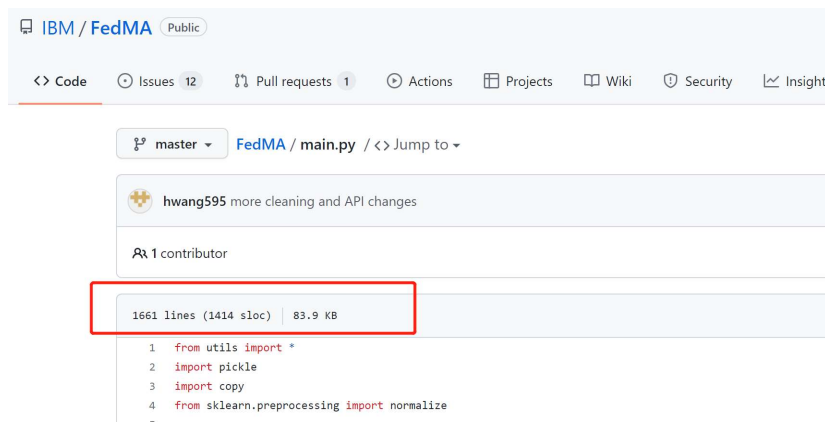
对齐方式:

- 现有方法属于后处理进行对齐的方式，神经元已经在客户端训练过程发生错位，然后在服务器进行聚合——“亡羊补牢，为时未晚”？



对齐代价:

- 现有方法的对齐过程需要求解OT或者最优指派问题，并且需要额外无标记数据，FedMA还需要微调处理，聚合代价高昂——“补牢代价”？



源代码复杂，只是主文件1000+行

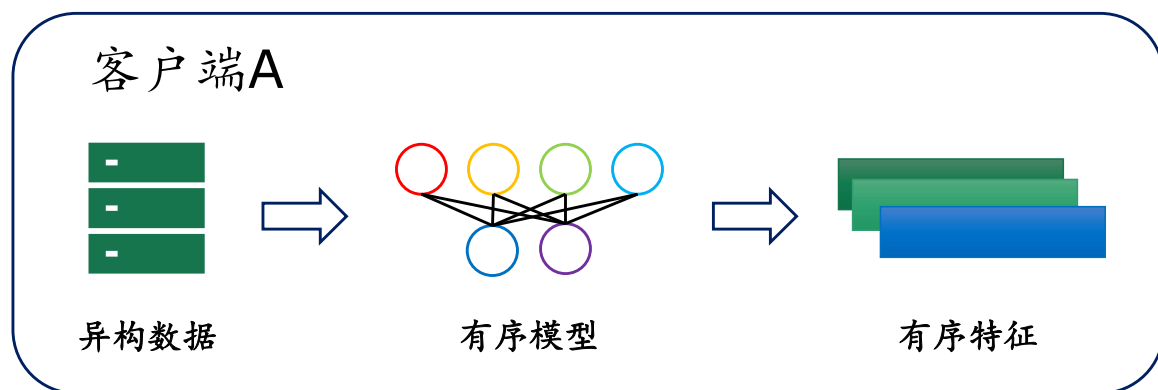
目录

- 项目背景
- 痛点难点分析
- 现有解决方案
- 提出的解决方案**
- 可扩展方向
- 技术落地应用

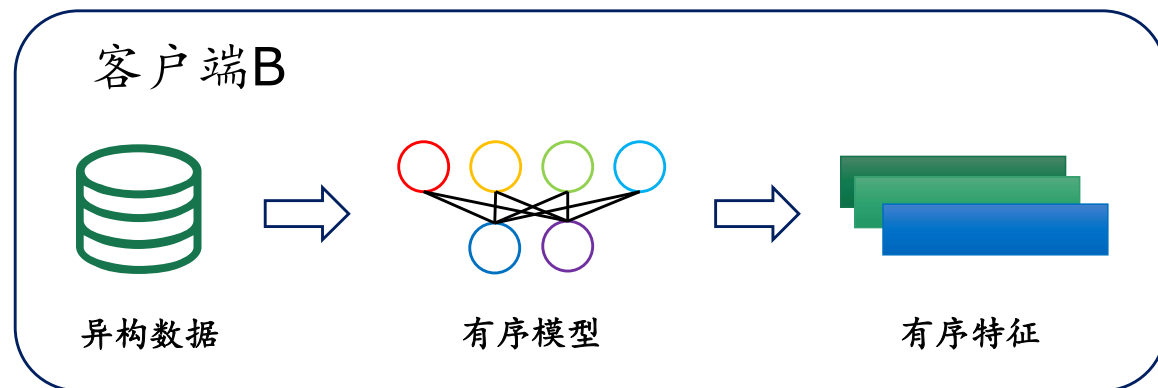
解决思路

有序模型或者有序特征学习

--在客户端就对神经元施加约束，强制不同客户端神经元有序



有序模型更
容易聚合













有哪些技术可以
实现有序模型学
习？

解决思路

学习有序特征的可能性方案：

相关研究查询(关键词)：Ordered, Aligned, ...

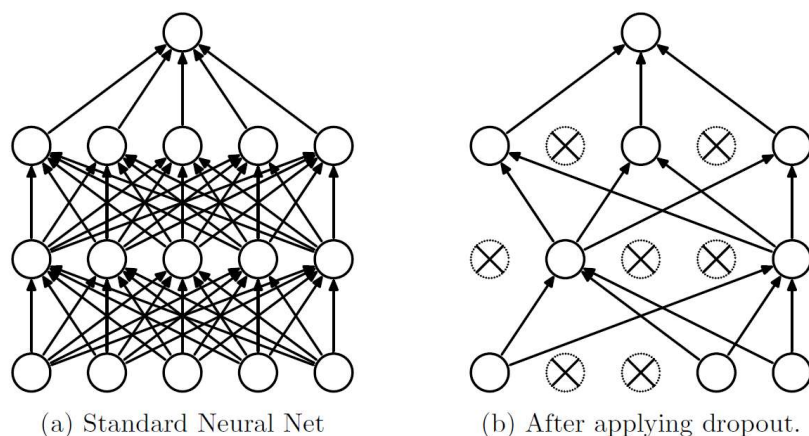
2014

-      Oren Rippel, Michael A. Gelbart, Ryan P. Adams:
Learning Ordered Representations with Nested Dropout. ICML 2014: 1746-1754
-      Oren Rippel, Michael A. Gelbart, Ryan P. Adams:
Learning Ordered Representations with Nested Dropout. CoRR abs/1402.0915 (2014)

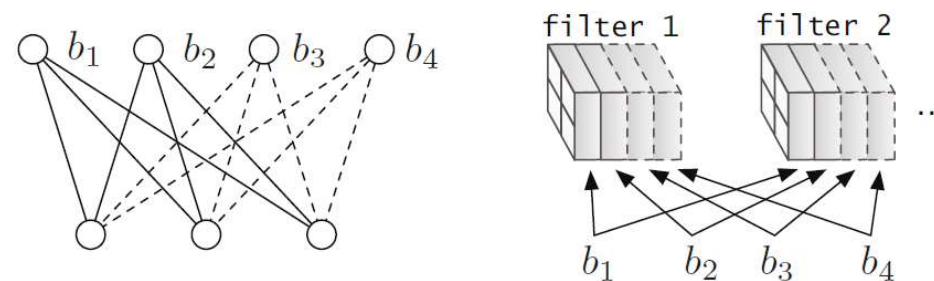
或许Nested Dropout可以实现神经元有序？

尝试方案：Nested Dropout

Nested Dropout原理：



Dropout随机去掉一些结点



Nested Dropout会随机选取一个位置 i ,
然后将最后面的 $K-i$ 个结点去掉

效益：每层前面的结点/卷积核会具有较高的信息量，且信息量随位置依次递减，特别地，选择几何分布作为位置 i 的选取方法： $p(i) = p^i(1 - p)$

优势：想法直观、非常契合联邦聚合过程、新颖，且实现容易，简单易行

尝试方案：Nested Dropout

代码实现：

```
class Dropout(nn.Module):
    def __init__(self, p, training=True):
        super().__init__()
        assert 0 <= p <= 1.0
        self.p = p
        self.keep_prob = 1.0 - p
        self.training = training

    def forward(self, xs):
        if self.training is True:
            if self.p == 1.0:
                return torch.zeros_like(xs)
            else:
                mask = (
                    torch.rand(xs.shape, device=xs.device) < self.keep_prob
                ).float()
                return mask * xs / self.keep_prob
        else:
            return xs

    def train(self, mode):
        self.training = mode
```

Dropout PyTorch实现

```
class NestedDropout(nn.Module):
    def __init__(self, n_unit, loc_ratio, divide=True):
        super().__init__()
        self.n_unit = n_unit
        self.loc = int(n_unit * loc_ratio)
        self.p = induce_geo_prob(n_unit, self.loc)
        self.training = True
        self.divide = divide

        index = np.arange(n_unit)
        self.keep_prob = sps.geom.sf(index, loc=self.loc, p=self.p)

    def forward(self, xs):
        # xs.shape (B, D)
        dim = xs.shape[1]
        indexes = sps.geom.rvs(p=self.p, loc=self.loc, size=(xs.shape[0], ))

        mask = torch.zeros_like(xs)
        for i, ind in enumerate(indexes):
            mask[i, 0:ind + 1] = 1.0

        if self.divide is True:
            keep_prob = torch.FloatTensor(
                list(self.keep_prob[0:dim])
            ).view((1, -1))
        else:
            keep_prob = torch.ones(dim).view((1, -1))

        mask = mask.to(device=xs.device)
        keep_prob = keep_prob.to(device=xs.device)

        if self.training is True:
            return mask * xs / keep_prob
        else:
            return xs

    def train(self, mode):
        self.training = mode
```

Nested Dropout PyTorch实现

尝试方案：Nested Dropout

实验效果：

	MNIST	MNISTM	SVHN	CIFAR-10	CIFAR-100
FedAvg	96.02	46.05	18.25	72.68	44.63
FedAvg + Dropout(0.5)	95.54	54.24	8.78	41.26	23.94
FedAvg + Dropout(0.1)	96.04	59.84	11.21	70.80	46.16
FedAvg + Nested Dropout	95.59	51.53	13.56	67.25	39.30

对比方法：FedAvg，FedAvg和不同概率下的Dropout

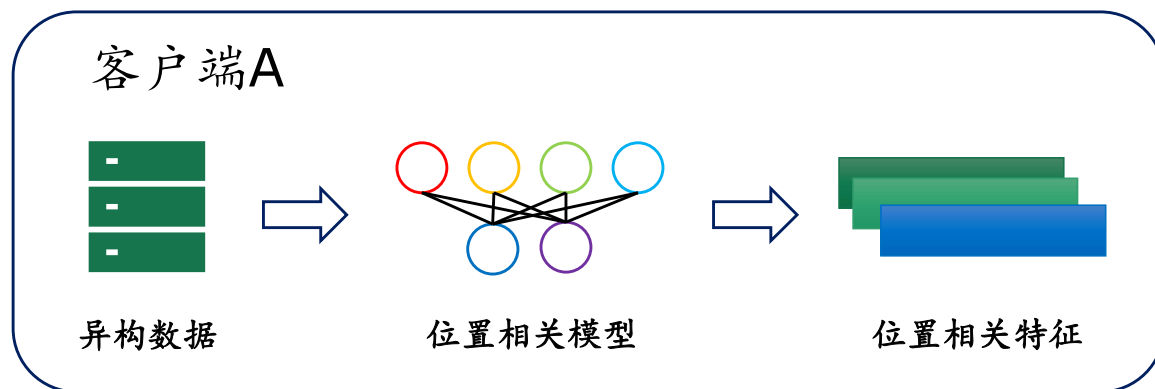
实验分析：

- 部分场景下直接使用Dropout会带来性能提升：客户端本地数据经常数量较少且有偏，使用Dropout可以避免过拟合，因此可以提升性能
- Dropout的比例不能太大：设置为0.5相当于每次训练丢弃的神经元太多，造成训练不稳定
- Nested Dropout效果不好：这种方法强制神经元有序太过“暴力”，没有免费的午餐，强制有序自然会带来性能损失

思路转变

有序 → 位置相关

--强制神经元有序的约束太强，或许只需要做到“位置相关即可”



位置相关模型
也容易聚合

联想到NLP中的位置相关处理技术：

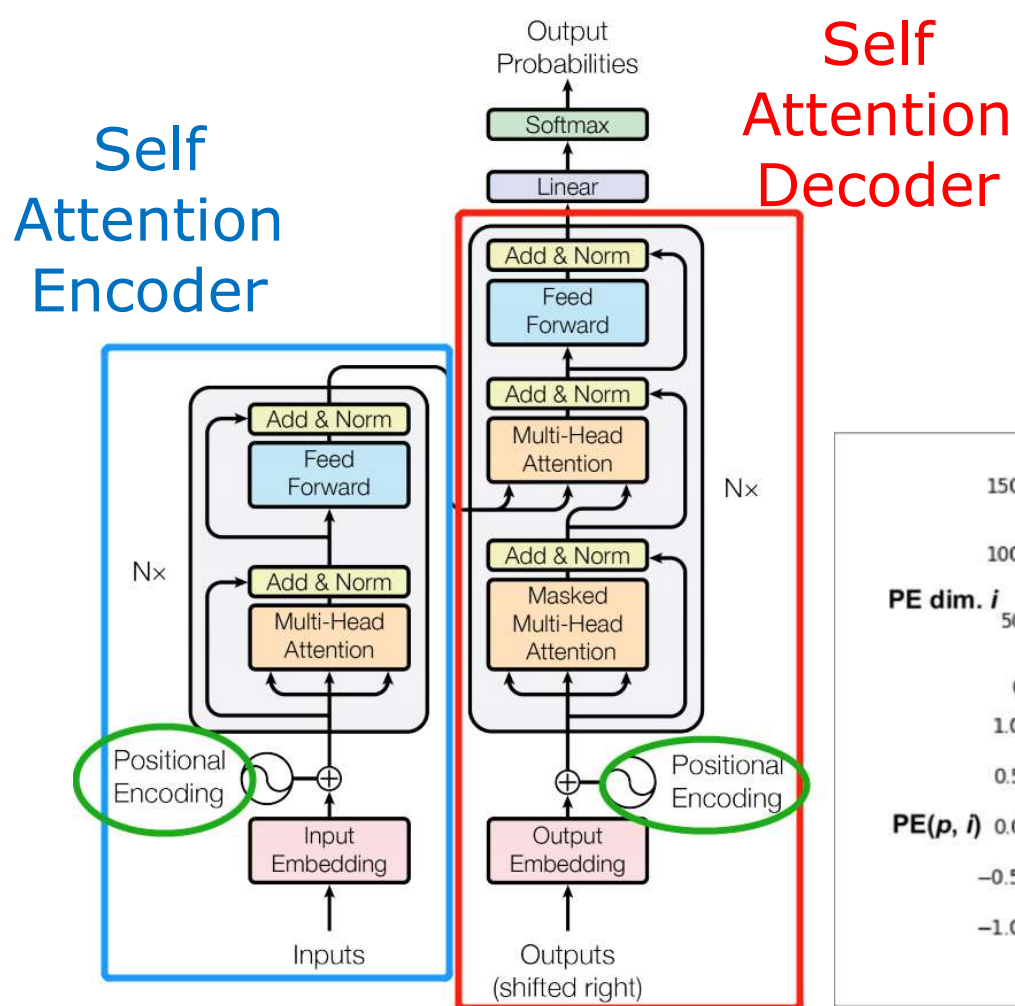
--Transformer中的位置编码

2017

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin:
Attention is All you Need. NIPS 2017: 5998-6008

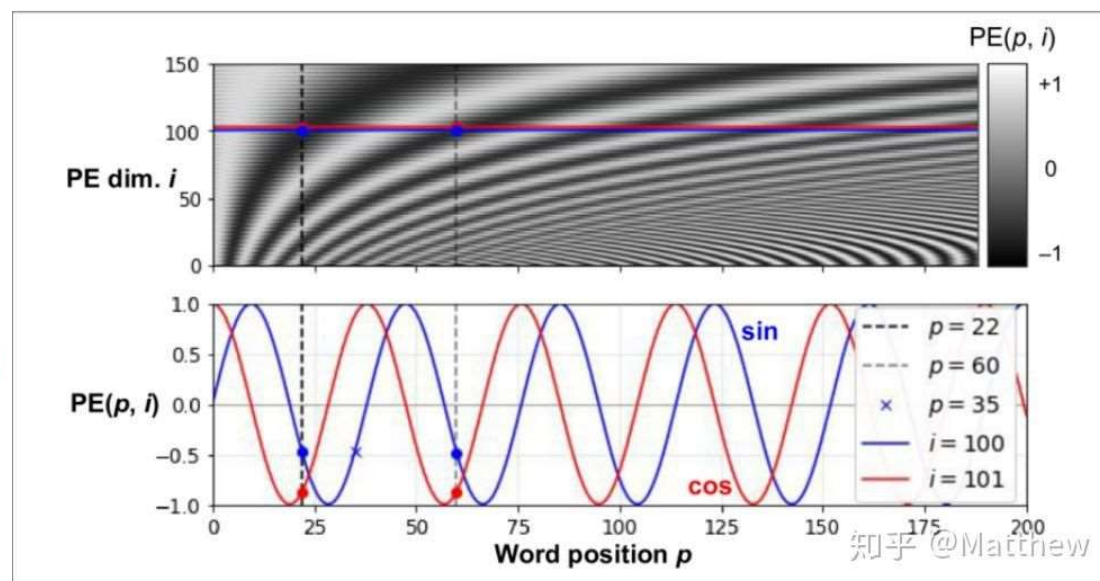
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin:
Attention Is All You Need. CoRR abs/1706.03762 (2017)

Transformer中的位置编码



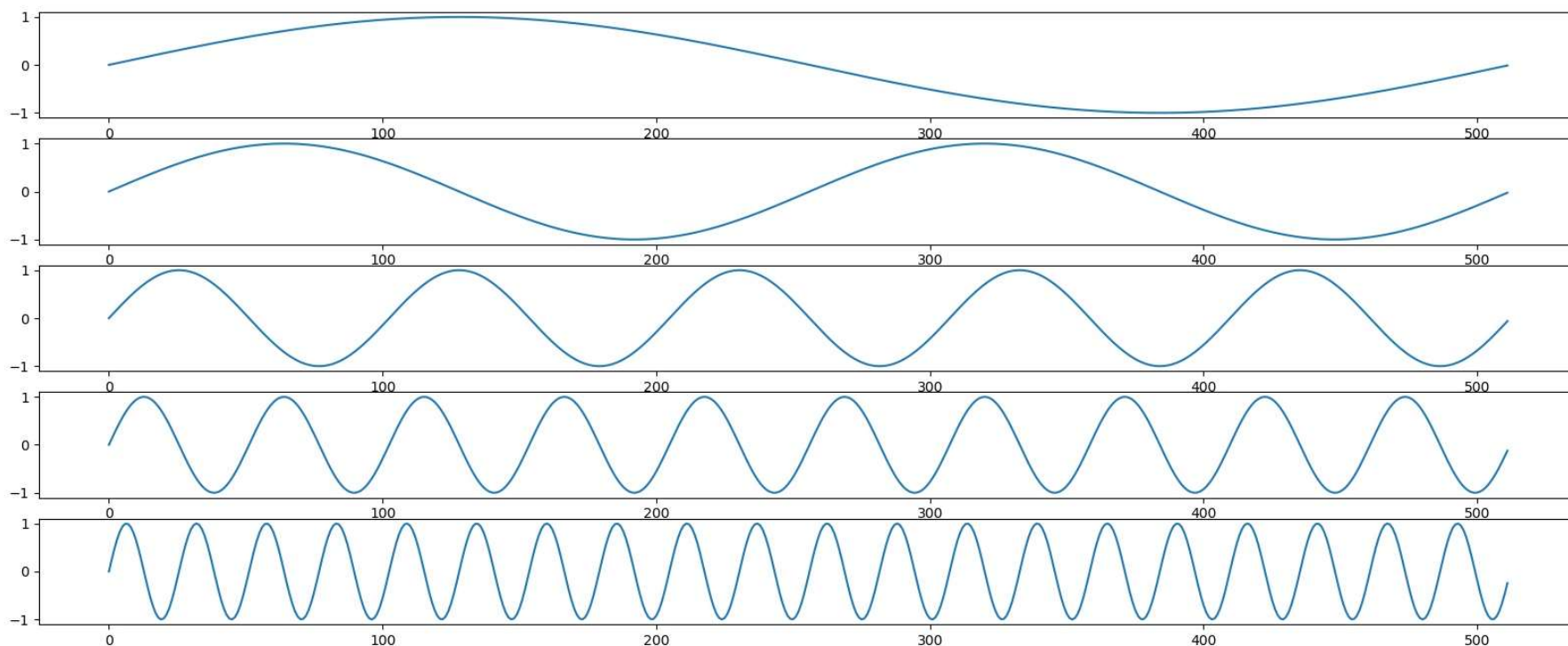
Position Encoding:
Fixed
Shape = (K, d)

$$PE_{(i,2j)} = \sin(i/10000^{2j/d_{model}}),$$
$$PE_{(i,2j+1)} = \cos(i/10000^{2j/d_{model}}),$$



神经网络+位置编码

Position Encoding: 通过位置编码学习位置相关的表示



- 全连接网络：按照神经元位置进行编码
- 卷积神经网络：按照通道位置进行编码

设计PANs

位置相关神经元(Position Aware Neurons, PANs)具体形式:

--可加性和可乘性PANs, 以余弦函数设置位置编码具体值

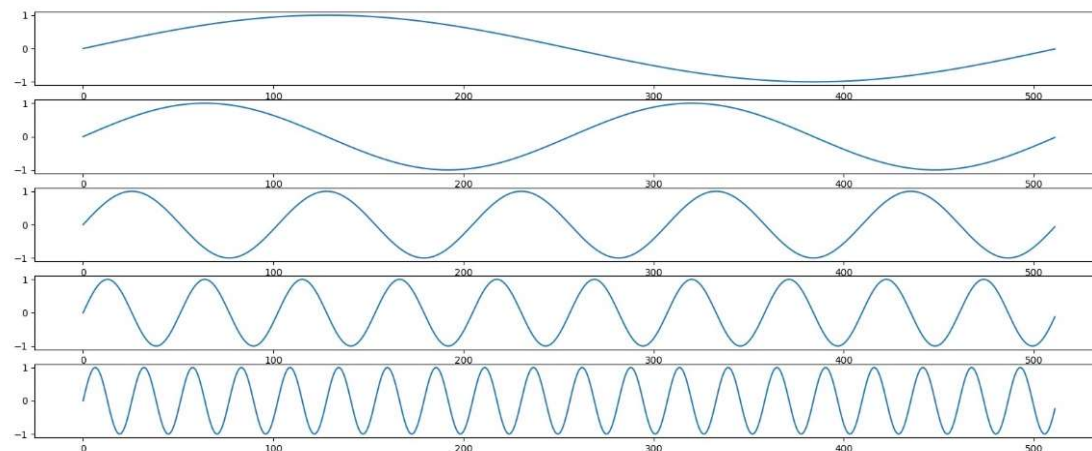
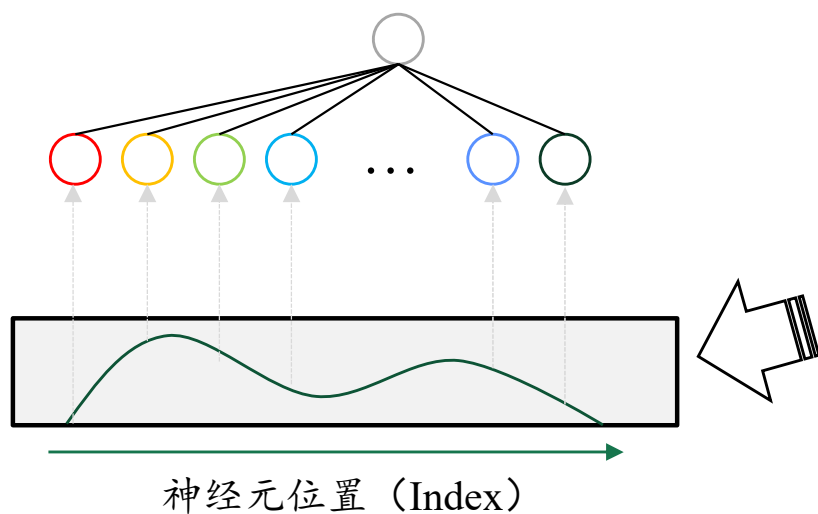
$$\text{PAN}_+ : h_l = f_l(W_l h_{l-1} + b_l + \underline{e_l}),$$

$$\text{PAN}_+ : e_{l,j} = A \sin(2\pi T j / J) \in [-A, A],$$

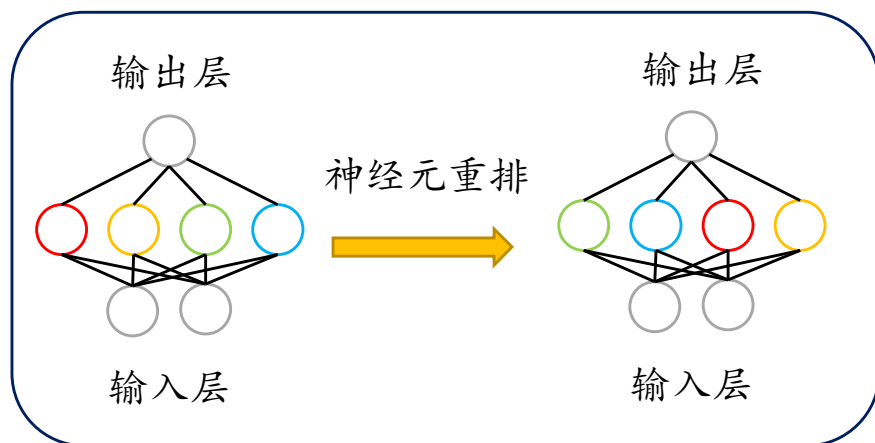
$$\text{PAN}_\circ : h_l = f_l((W_l h_{l-1} + b_l) \circ \underline{e_l}),$$

$$\text{PAN}_\circ : e_{l,j} = 1 + A \sin(2\pi T j / J) \in [1 - A, 1 + A],$$

在神经元输出上加上或者乘以相应的位置编码, 位置编码只和位置有关, 不可学习



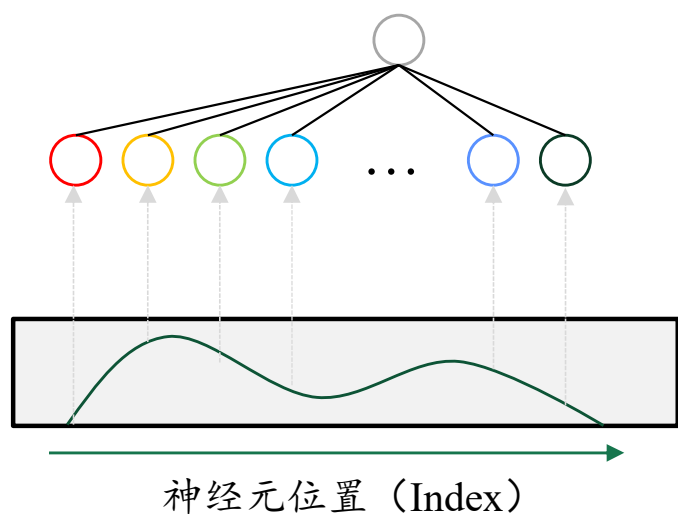
PANs和重排不变性



现象：神经网络的重排不变性 + 异构数据分布

结果：客户端模型对应的神经元可能会错位，参数平均的聚合方式性能很差

→ 如果可以限制神经网络的重排不变性？



解决方案：

每个神经元的输出上乘以或者加上一个位置编码

原来的： $f_j(x) = w_j^T x + b_j$

现在的： $f_j(x) = w_j^T x + b_j + e_j$ or $f_j(x) = (w_j^T x + b_j) * e_j$

预期效果：

--神经网络 不再具有重排不变性

PANs和重排不变性

为什么加入位置编码可以限制神经网络重排不变性：

具有一个隐藏层(L 个神经元)的正常神经网络：

$$y = v^T f(Wx + b), \quad W \in R^{L \times d}, b \in R^L, v \in R^L$$

重排矩阵： $\Pi \in \{0,1\}^{L \times L}$

神经元重排： $y = (\Pi v)^T f(\Pi Wx + \Pi b) = v^T f(Wx + b)$

具有重排不
变性

加了位置编码后的神经网络：

$$y = v^T f((Wx + b) \odot \mathbf{p}), \quad \mathbf{p} \in R^L$$

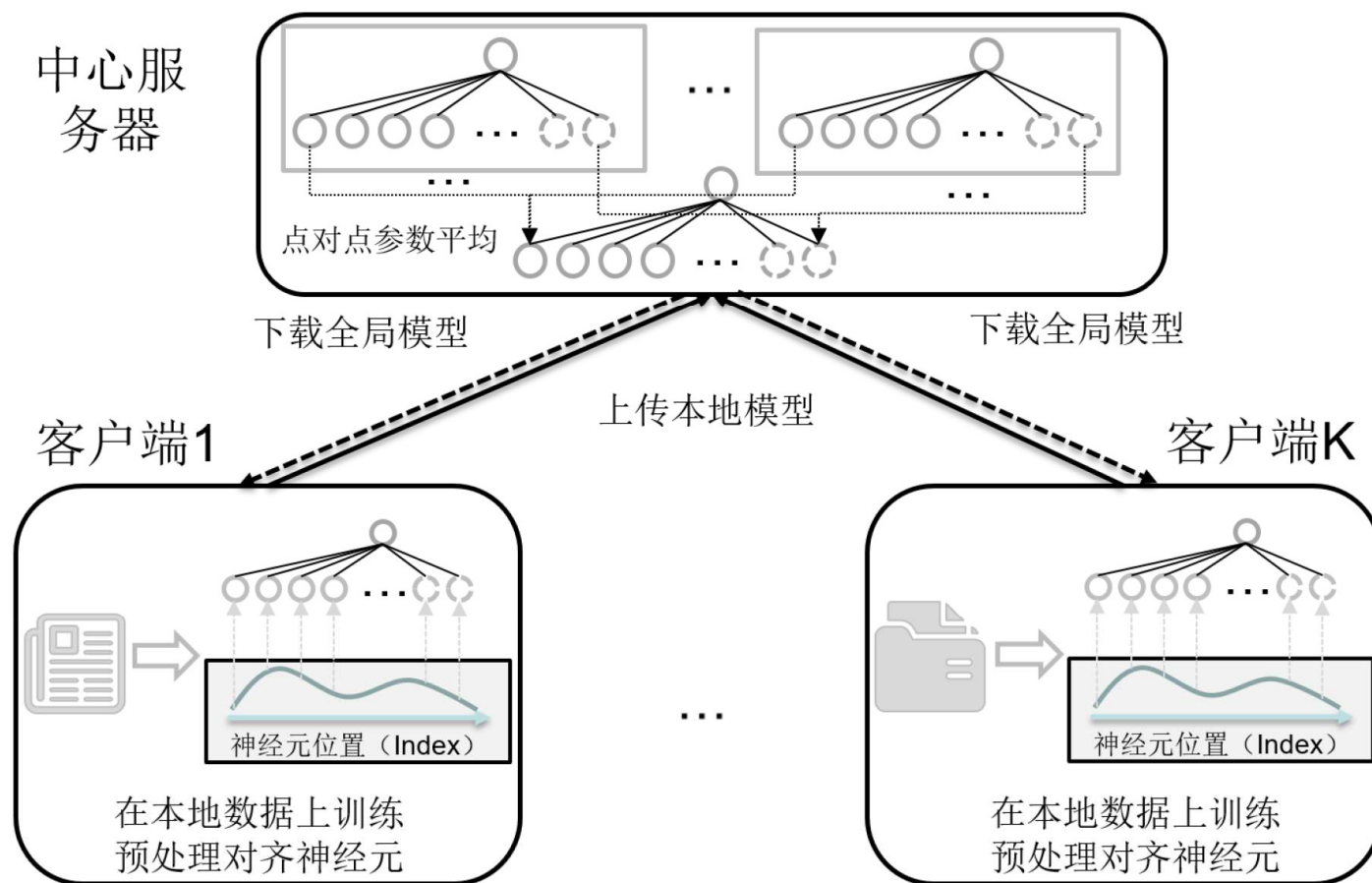
重排之后：

$$\begin{aligned} y &= (\Pi v)^T f((\Pi Wx + \Pi b) \odot p) \\ &\neq (\Pi v)^T f((\Pi Wx + \Pi b) \odot \Pi p) = v^T f((Wx + b) \odot p) \end{aligned}$$

丢失重排不
变性

分布式训练+PANs

效果：本地训练神经元通过位置编码进行预对齐，在中心服务器即可进行有效点对点参数平均



分布式训练+PANs

为什么带有位置编码的神经网络应用到分布式训练会有效：

直观解释：

PANs将神经元和其位置紧密关联，在异构数据情况下，局部模型的神经元会尽量避免错位现象，因为神经元错位之后会使得输出变化，从而导致训练误差变大以及训练不稳定

为了求稳，神经元不会错位

理论解释：

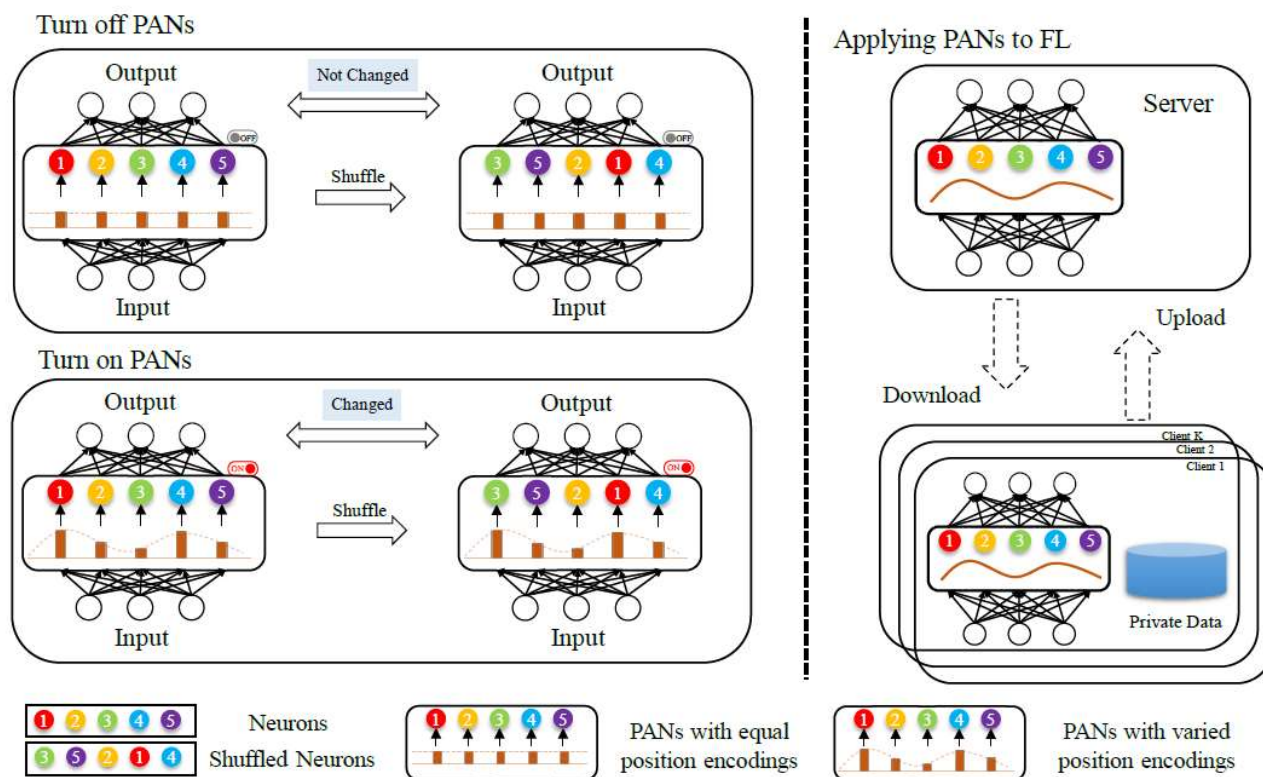
各个客户端前向训练过程有共同的成分，位置编码。梯度更新过程也有一致的成分，位置编码越强，梯度更新方向越一致

$$\text{PAN}_+ : \partial h_l^{(k)} / \partial b_l^{(k)} = D(f_l^{(k)'}) ,$$

$$\text{PAN}_o : \partial h_l^{(k)} / \partial b_l^{(k)} = D(f_l^{(k)'}) D(e_l) ,$$

客户端共享信息：梯度正则化

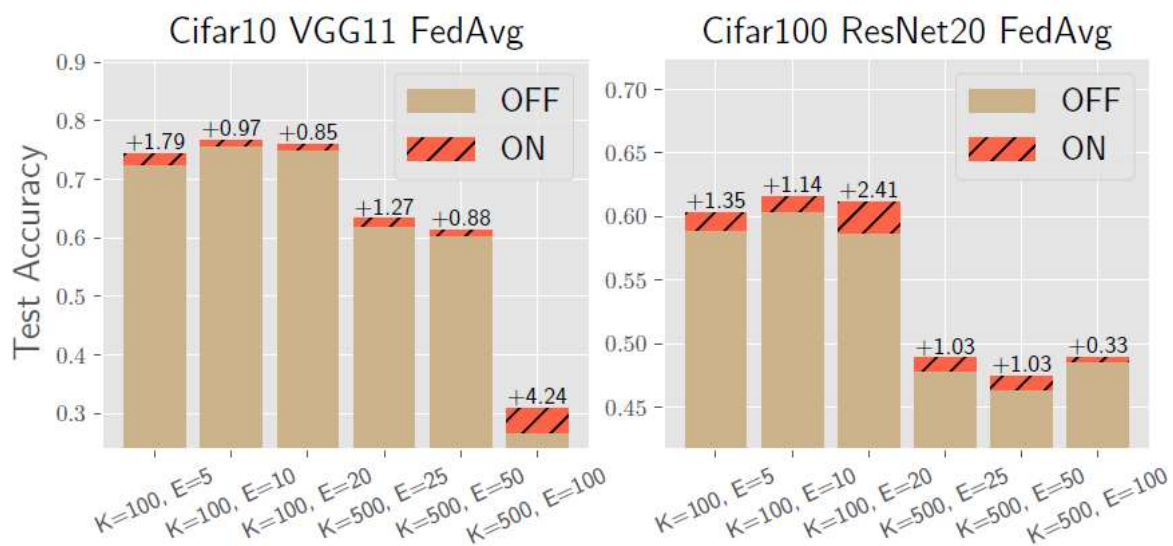
研发算法FedPAN



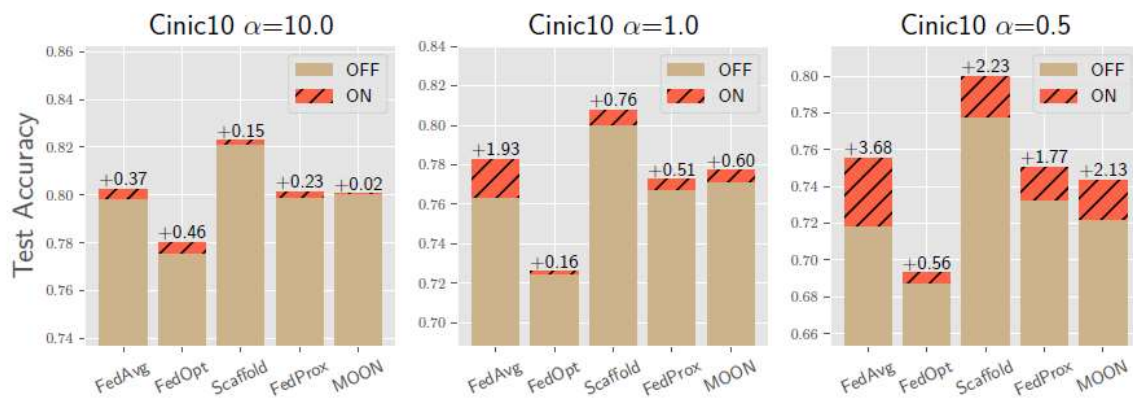
左图：提出“位置相关的神经元” (Position-Aware Neurons, PANs)，将其封装为一个“开关”：打开/关闭PANs，即可禁用/启用神经网络的重排不变性

右图：将PANs应用到分布式训练，客户端上预对齐特征

研发算法FedPAN



在多个数据集、多种设置下均可提升聚合模型的性能(红色部分)

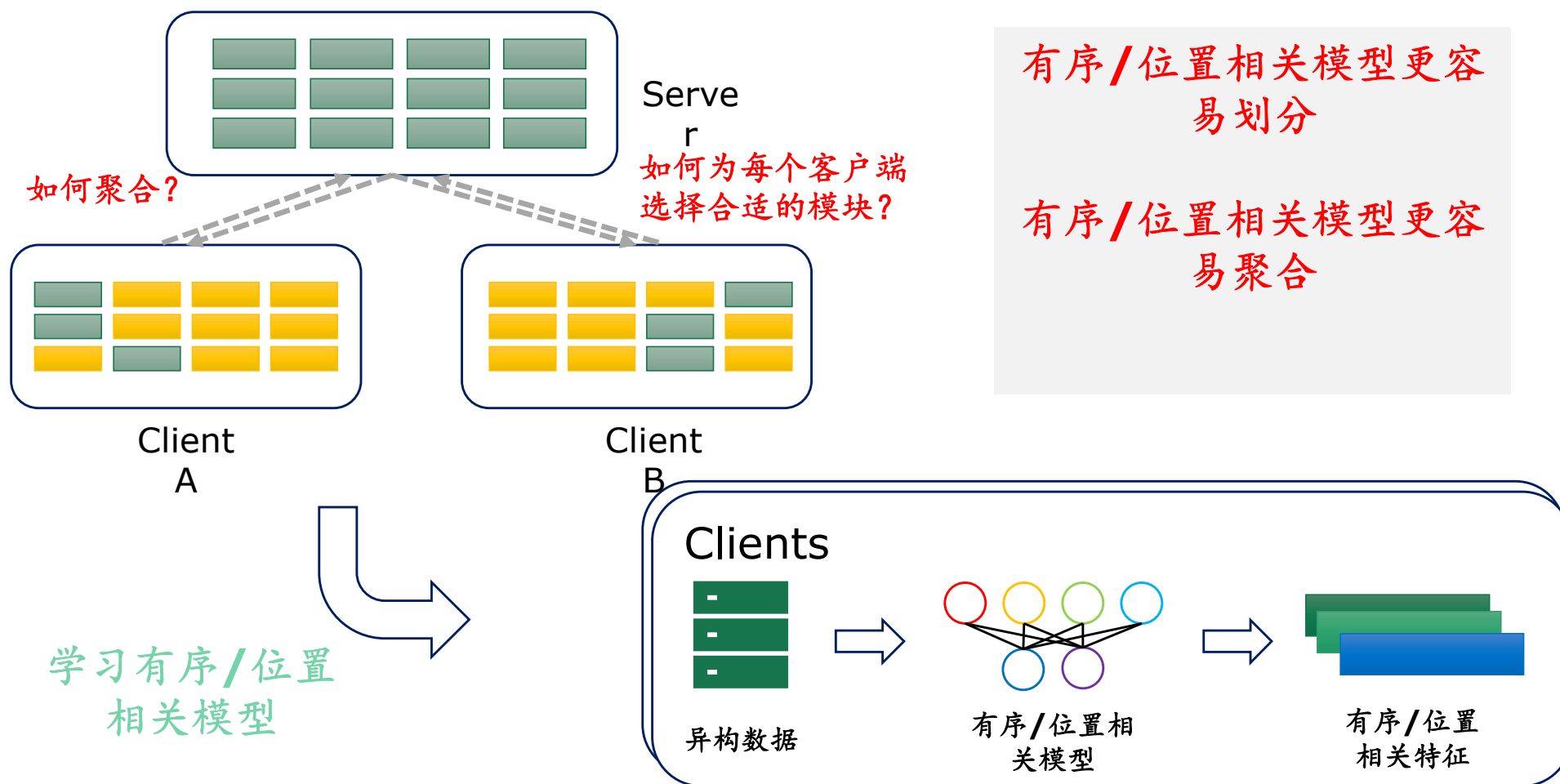


目录

- 项目背景
- 痛点难点分析
- 现有解决方案
- 提出的解决方案
- 可扩展方向**
- 技术落地应用

分布式模块化集成学习

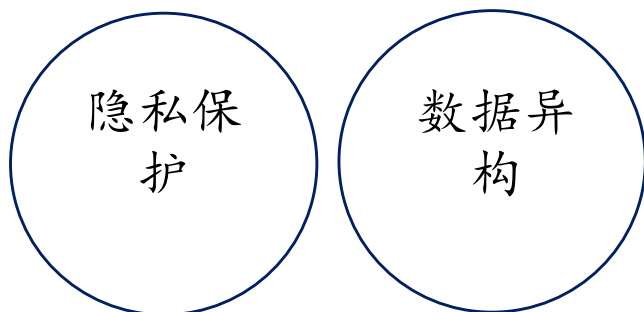
深度网络模块化集成框架：将模型按照纵向、横向切分为多个模块，每个客户端只选择部分模块，如何切分？如何聚合？



目录

- 项目背景
- 痛点难点分析
- 现有解决方案
- 提出的解决方案
- 可扩展方向
- 技术落地应用

技术落地应用



技术研究深化:

- 研究基于位置编码的特征预对齐技术(已和**华为**合作申请专利 + **CVPR 2022**中稿)
- 研究基于有序/位置相关模型的分布式集成学习框架

技术落地

实际应用场景

- ✓ **多APP智能业务感知**: 根据流量包对APP类型进行识别, 但是不能将用户流量包聚集到服务器训练 (已验证)
- ✓ **ONT/OLT应用识别**: 对家庭光猫的用户流量进行应用识别, 需根据以往模型构建新场景下的模型 (待验证)
- ✓ **DC节能模型构建**: 节能模型随时间场景特异, 需根据现有的节能模型推导出合适的新节点模型, 减少成本 (待验证)

多APP智能业务感知背景

智能业务感知(Service Awareness, SA):

-- 通过机器学习从字节流中检测该流量来自的具体应用

问题描述:

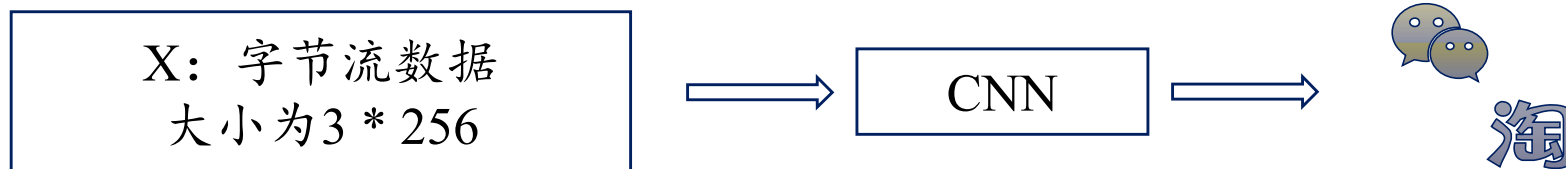
- 单一局点上训练SA模型需要大量的数据支持
- 若要借助多方之间的数据进行合作, 需要保护数据隐私
- 现有数据包括3个局点, SA目标分类是36分类

项目目标:

- 在数据隐私保护的条件下联合三方数据来提升SA模型的性能

SA机器学习问题抽象:

-- 从字节流中检测具体应用的过程使用CNN分类模型进行建模

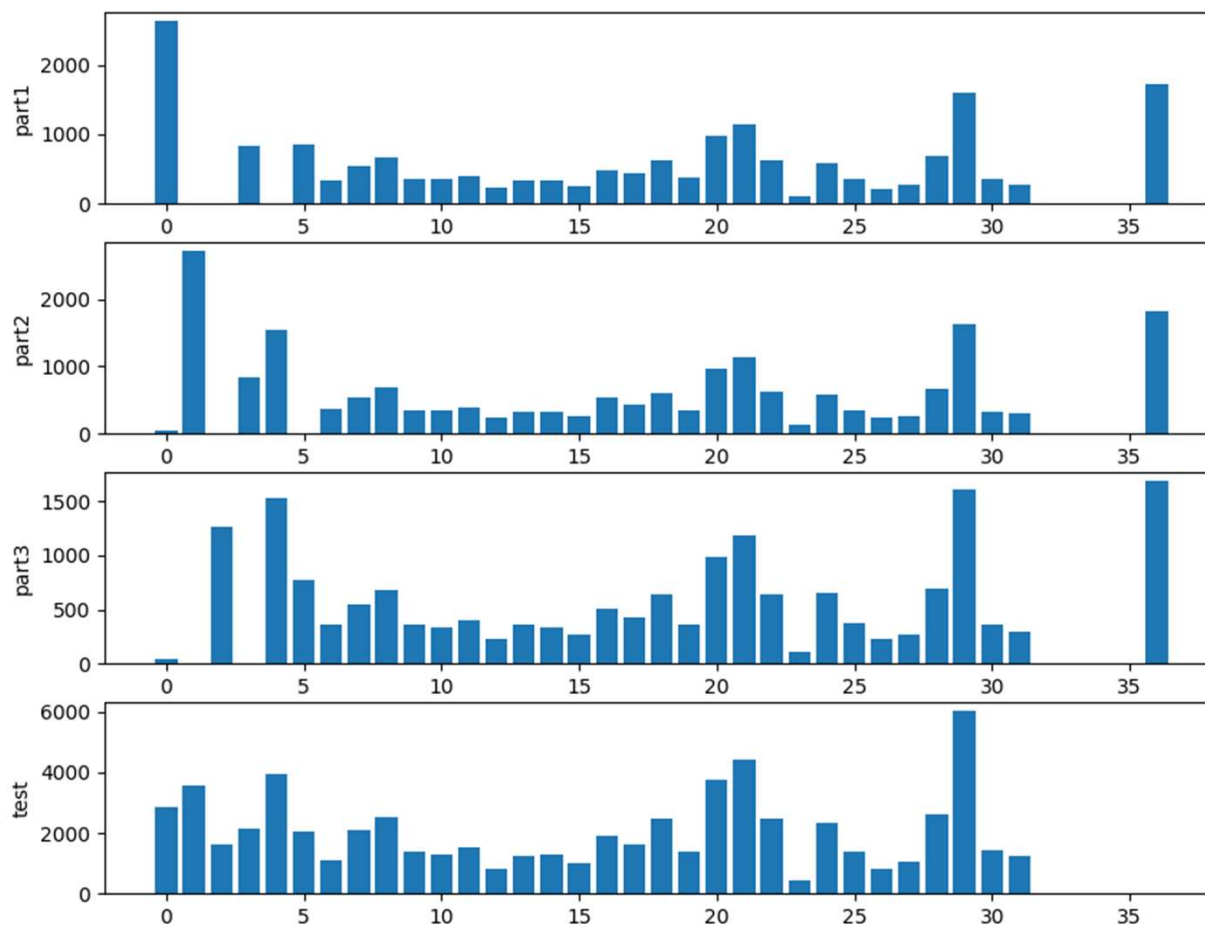


多APP智能业务感知难点

数据分布**NonIID**:

- 标记分布差异
- 特征分布差异

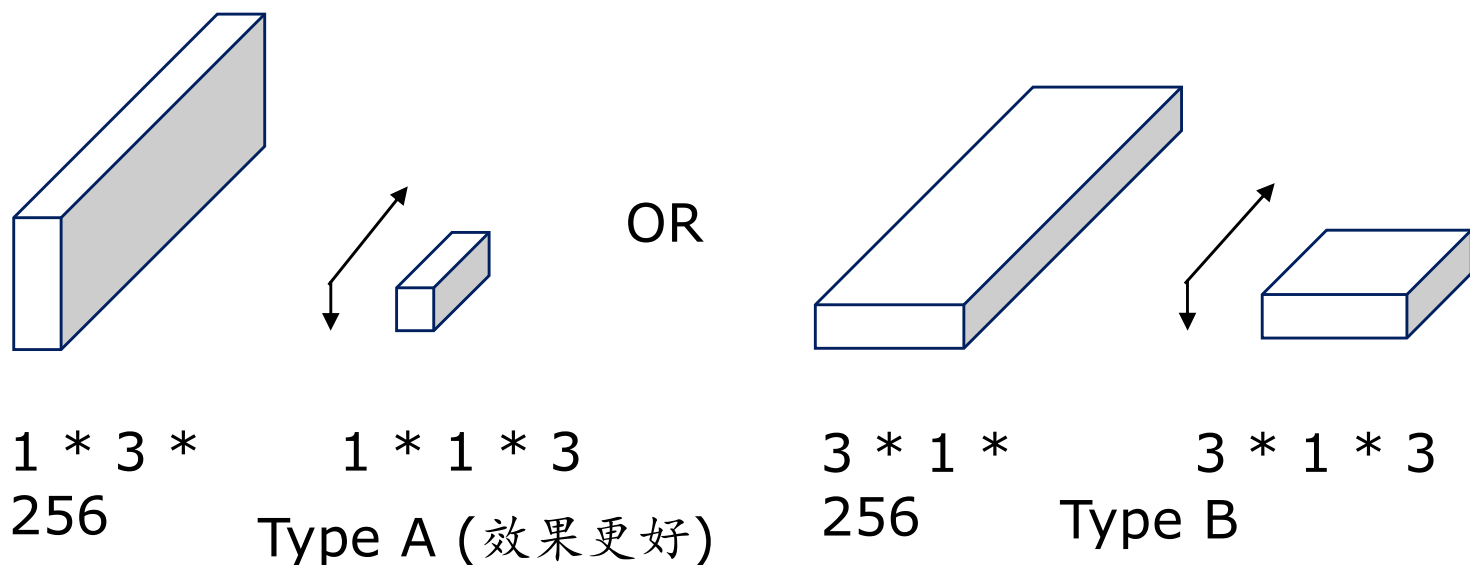
右图展示了三个局点的标记分布，以及全局测试集的标记分布，仅看标记分布就存在很大差异，个别局点上还会有缺失类别，导致直接应用FedAvg会出现模型参数点对点聚合不准的问题



SA卷积网络设计

SA卷积神经网络设计(非常重要):

- 卷积核的设计需要考虑信息流的存储方式
- SA中信息按行存储, 可以理解为3行256列



Model	参数量	运行时间	局点1 ACC	局点2 ACC	局点3 ACC	数据集中ACC
Type A	2, 037, 629	2.51h	0.757	0.803	0.755	0.932
Type B	726, 293	1.89h	0.659	0.742	0.708	0.916

FedPAN应用效果

FedAvg	局点1 ACC	局点2 ACC	局点3 ACC	全局测试集ACC
E=5	0.836	0.834	0.836	0.837
E=10	0.815	0.819	0.816	0.824

FedPAN	局点1 ACC	局点2 ACC	局点3 ACC	全局测试集ACC
E=5	0.873	0.882	0.867	0.893
E=10	0.859	0.858	0.858	0.859

* E代表客户端本地迭代的轮次

验证结果分析：

- 由于数据异构，FedAvg使用点对点参数聚合的模型性能只有0.837
- 采用了FedPAN的结果可以达到0.893，拉近了与数据集中式训练的效果(0.932)的差距

THANKS

