联邦学习环境下的高效通信方法研究

# 摘要与关键字

# 引言

## 研究背景

在物联网时代，智能手机、平板以及物联网设备无时无刻不产生着大量数据。使用这些数据来训练机器学习模型可以很好地提升用户体验。然而，由于用户数据的隐私性，将这些数据上传到中心服务器再使用传统的机器学习算法进行集中训练是不可行的。出于保护用户隐私的考虑，联邦学习范式应运而生。联邦学习最早由谷歌研究院提出，它允许将用户数据保存在用户设备上，并借助用户设备固有的计算能力来训练一个共享的机器学习模型，在数据不共享的情况下完成联合建模。

联邦学习常用的架构有两种：一种是客户端—服务器架构，另一种是对等网络架构。在客户端—服务器架构中，联邦学习的一般过程是：首先，参数服务器将当前模型发送给各个客户端。然后，客户端使用本地数据训练出一个新模型，并且将脱敏后的模型参数发送到参数服务器。最后，参数服务器在收到模型参数后进行聚合，将聚合后的模型参数分发给客户端。客户端和服务器端重复以上步骤，直到全局模型收敛为止。在对等网络结构中进行联邦学习训练时，参与方之间可以直接通信，不需要第三方的介入，隐私的安全性得到了进一步提高，但需要更多的计算操作进行加密和解密。

然而，不论是哪种联邦学习架构，在进行联邦建模的过程中，参与方之间需要反复交换大量的模型参数，通信频率快、通信数据的总量大。此外，由于绝大多数用户设备是通信带宽有限、电量有限的移动设备，这些资源受限设备上的通信通常是比较慢速且不稳定的，可能会影响到联邦学习任务的正常进行。特别是在全局模型规模较大的情况下，网络带宽的限制和参与方数量过多会加剧联邦学习的通信瓶颈，可能会造成用户设备掉队或退出的问题。因此，通信开销和通信效率成为联邦学习的主要瓶颈之一。在联邦学习环境中，如何降低通信开销、提高通信效率，是近来联邦学习优化算法研究的热点方向。

在联邦学习的实际应用中，对企业来说，联邦学习主要用来解决企业信息化建设中的数据孤岛问题。因此，找到一种高效通信的联邦学习方法，降低通信开销，减轻网络负担，能切实降低企业运营成本，提高企业的经济效益；对用户来说，降低用户设备与参数服务器的通信开销，使得共享模型的训练对用户透明，降低用户设备的关键资源消耗，能更好地提升用户体验。

## 国内外研究现状

通信开销一般包含两层含义：一是通信数据的总量；二是通信的总时耗。本文研究的通信开销特指联邦学习任务中全局模型达到预设的性能指标（如特定精度）所需传输的数据总量和通信轮次。因此，降低通信开销通常可以从减少通信总次数、降低通信频率以及减少单轮通信回合的通信总比特数入手。减少通信总次数主要依靠降低模型更新频率和选择更少的客户端进行通信来完成，而减少单轮通信回合数据量则主要依靠对模型进行压缩，通过降低通信占用的带宽来降低总的通信开销。本文仅讨论服务器端—客户端架构下的联邦学习通信开销的优化。

### 2.1基于降低模型更新频率的优化方法

为了降低联邦学习环境中的模型更新频率，可以考虑以客户端的计算成本来换取客户端与参数服务器之间的通信成本，即增加参与方的计算量或提高并行性来减少训练模型所需的通信次数。

增加参与方的计算量指的是每个参与方在每个通信回合内执行更加复杂的计算。以参与方执行随机梯度下降的优化算法为例，每个参与方执行随机梯度下降的多次迭代以计算权重更新，而不是在每次迭代后向服务器端发送更新后的模型。提高并行性则是引入更多的参与方在每个通信回合之间独立工作，使得全局模型的实际优化次数增加，从而降低全局模型达到收敛的预设精度所需要进行模型聚合的次数。

### 2.1.1增加参与方计算量

FedSGD算法使用随机梯度下降方法作为客户端本地模型的优化算法。它每一轮在各个客户端上进行一次梯度计算，因此，计算效率很高，但是通常需要大量的迭代轮次才能使模型收敛，导致其通信开销很大。

为了降低频繁通信造成的巨大的通信开销，McMahan等提出了联邦平均（FedAvg）算法。FedAvg算法先在客户端上进行多次的随机梯度下降，然后再将本地的模型更新结果发送给服务器。根据FedAvg算法在MNIST数据集和SHAKESPEARE数据集上的实验结果显示，不论是在卷积神经网络还是在长短期记忆网络上，在达到相同的模型精度时，该方法所需要的通信轮数明显少于FedSGD方法。

Alistarh等在McMahan等人的基础上优化了FedAvg算法，通过增加每一轮迭代在每个客户端本地更新参数的计算次数，进一步降低了通信开销。Alistarh等使用卷积神经网络在MNIST数据集上进行测试，实验结果表明：当数据独立同分布时，该算法可以明显降低通信成本；但当数据非独立同分布时，算法对于通信开销降低的效果仍不明显。显然，实际场景中，联邦学习的数据基本都属于非独立同分布，因此FedAvg算法及Alistarh提出的优化算法虽然较FedSGD算法通信成本更低，但应用场景有限，需要进一步探究针对非独立同分布数据的优化算法。

Li等人提出了一种更通用的FedProx算法。该算法并不要求每一次迭代时不同客户端的本地计算次数统一，而在每一轮中，只对一部分客户端进行采样以执行更新，这种算法在数据为非独立同分布时优化效果更明显。因此，FedProx算法可以动态更新不同客户端每一轮需要本地计算的次数，充分利用客户端的计算资源，使得算法更适合客户端数据为非独立同分布的联邦学习场景。

### 2.1.2提高并行性

并行计算分为同步并行和异步并行。对于联邦学习任务，增加客户端的数量能够显著减少整个联邦学习系统的通信时间和单个参与方的通信量。但同步并行计算中存在显著的“短板效应”：当某个参与方出错需要重新计算时，该节点计算所需时间比其他所有节点都要长，然而，由于同步更新模型的要求，其他节点依然需要一直等待该节点完成计算才可进行下一步。为了解决这种“短板效应”，Zhou等从算法框架的角度出发，将通信与训练并行，基于集合分层计算策略、数据补偿机制和NAG（NesterovAcceleratedGradient）算法，提出重叠联邦平均（OverlapFedAvg，Overlap-FedAvg）算法。以FedAvg算法为基线，在非独立同分布的数据场景下分别使用MLP等模型在不同数据集上训练，结果表明，Overlap-FedAvg算法单次迭代训练需要的时间少于普通的FedAvg算法。Overlap-FedAvg算法具有很高的并行度，能够在保持与FedAvg几乎相同的最终精度的前提下，大大加快联邦学习的进程，适用于模型相对较大且客户端的网络连接缓慢或不稳定的场景，对不平衡和非独立同分布的数据具有鲁棒性，可以减少在分散数据上训练深度网络所需的通信轮次。

### 2.2基于模型压缩的优化方法

模型压缩通过降低每轮次模型聚合时客户端占用的通信带宽来降低总的通信开销。压缩方案可以是随机稀疏模式、概率量化、梯度量化、子抽样、低秩等方法的一种或多种组合。根据执行阶段不同，联邦学习任务中的模型压缩可以分为上行链路压缩和下行链路压缩：上行链路压缩是指参与方再上传模型更新之前，对模型更新进行压缩的过程；下行链路压缩是指中央服务器再分发全局模型之前，对模型进行压缩的过程。在不同的执行阶段，具体的压缩方案也有比较大的差别。

JakubKonečný等人为了减少上行链路的通信开销，提出使用结构化更新与轮廓更新两种方式来更新服务器参数。结构化更新通过提前定义上传模型参数的矩阵结构上传模型更新，而轮廓更新是指每次更新的参数需要在客户端本地进行压缩编码，学习完整的更新模型后，进行压缩再发送给服务器。模型最后通过在CIFAR10数据集上的图像识别算法进行验证，实验表明，参与方越多，压缩效果越好。

Caldas等人考虑的是服务器到客户端的下行链路中模型参数的通信优化，通过有损压缩的方式来减少从服务器到客户端需要传递的参数数量。当迭代次数保持不变时，模型压缩方法会在一定程度上降低模型准确率。

### 2.3基于客户端选择的优化方法

在联邦学习中，客户端的数量可能非常大，但由于模型分发和重新上传的带宽相当有限，一般只选取一部分参与方参与训练过程。因此客户选择策略对于联邦学习过程计算效率、通信效率、最终模型的质量以及公平性等至关重要。客户端选择算法需要根据数据集是否IID、是否有用户退出等实际情况选择最优方案。

### 2.3.1数据IID场景下的FedCS算法

Nishio等提出了一种FedCS（FederatedClientSelection）算法，根据累计有效参与值（CumulativeEffectiveParticipation，CEP）选择模型迭代效率最高的客户端进行聚合更新，以此提高整个联邦学习算法的收敛效率，进而降低通信代价；但该算法只有在基础的动态神经网络等典型网络性能较好或数据IID时，精度与通信开销性能较好，对于拓扑结构或参数较为复杂的情况，该方法客户端选择公平性和客户端聚合效率会更低，反而会造成通信次数增加。

### 2.3.2数据non-IID场景下的Hybrid-FL算法

针对FedCS算法只能在数据IID时同时保证高精度和降低通信开销，但数据non-IID时降低通信代价却无法保证高精度的问题，Yoshida等在启发式算法（heuristicalgorithms）的基础上提出了一种Hybrid-FL（HybridFederatedLearning）的协议，该协议可以处理数据non-IID的客户端数据，解决在non-IID数据上FedAvg、FedCS算法精度、准确度等性能不高的问题，文献［39］在数据non-IID场景下仿真边缘计算环境，在CIFAR-10和FashionMNIST数据集上通过执行分类任务进行性能测试，结果表明non-IID数据场景下，为了达到较高准确率时，该方法所需通信代价小于FedAvg、FedCS算法，但Hybrid-FL协议一定程度上增加了通信损耗：服务器需要通过额外的资源请求选择部分客户端，从而在本地建立一种近似独立同分布的数据集用于联邦学习的训练和迭代。因此，下一步可以研究如何综合运用Hybrid-FL和FedCS方法，平衡精度与通信代价之间的关系，在保证高精度的同时使通信开销足够低。

### 2.3.3波动训练环境下E3CS算法

在真实的联邦学习中，被选中的客户往往有机会退出，不会返回经过训练的模型，也不会通知服务器他们的退出，这种情况将会形成一种波动的训练环境。Huang等针对更接近现实的波动的训练环境和非独立同分布的数据，研究了客户端选择问题，提出了FedCS的改进方法E3CS（Exp3-basedClientSelection），这一研究扩展了指数权重算法的应用领域。对该算法进行性能评估时，以随机选择客户端和FedCS为基准组，对EMNIST和CIFAR-10数据集未带标签的图片进行分类，实验结果表明E3CS方法在达到相同最终精度时所需的通信次数更少。随后，Wu等在研究波动环境下的客户选择问题时，为了提高训练收敛速度和最终模型精度，也运用了E3CS随机选择算法，并进一步设计了“公平配额”设置，该方法在减少通信开销的同时能够保证最终模型精度的损失很小。

## 研究内容

### 3.1理论依据

联邦学习本质上是一种分布式机器学习技术或机器学习框架，其目标是在保证在保证数据隐私安全及合法合规的基础上，实现共同建模，提升AI模型的效果。深度学习是机器学习的一个重要分支，因此，联邦学习一般通过训练深度神经网络来完成业务场景的联合建模。

深度神经网络的应用十分广泛。然而为了更强的特征表示能力，网络也变得越来越大。例如，由Krizhevsky设计的AlexNet拥有大约6100万个参数，是LeCun的传统模型（如LeNet-5）的100多倍，更不用说VGG网络这种复杂得多的模型了。越来越多的参数需要很强的存储能力、计算能力、电力等等，这些限制对在移动端的部署构成了巨大挑战，此外，在资源受限的联邦学习环境中，具有超大量参数的模型的聚合和分发更是加大了服务器端和客户端通信的压力。尽管深度神经网络模型需要大量参数保证其表征能力，然而近来有研究指出神经网络中存在大量的冗余。这使得在几乎不带来精度损失的前提下压缩神经网络模型变得可能。

联邦学习场景中的通信链路可以抽象为上行通信链路和下行通信链路。在上行通信链路中，客户端训练好模型后，对训练好的模型进行压缩，将压缩后的模型发送到服务器端执行聚合。在下行链路中，服务器给各个客户端分发聚合后的压缩模型。当全局模型达到收敛的聚合轮次基本保持不变时，每一轮次的上行通信量和下行通信量都得到了大幅度的减少，极大地降低了联邦学习环境中的通信开销。

### 3.2主要研究内容

本文使用FATE框架搭建联邦学习环境，并且将几种经典的剪枝和量化方法集成到FATE框架中，在联邦学习的情景下使用不同的压缩策略对AlexNet网络模型进行了模型压缩，并根据实验结果对压缩效果以及联邦学习聚合效率进行了分析。结果显示，通信效率在模型精度损失很小、收敛速率降低幅度很小的情况下得到了较大提升。

### 3.3研究创新点

1. 模型压缩研究只关注普通场景下的模型压缩，本文在联邦学习的场景下模型压缩联邦学习的场景下。模型压缩在联邦学习场景下需要注意什么问题？做了哪些适应性的改变？
2. 现有的工作都是只使用一种方式进行模型压缩，本文将这些方法进行了有机结合。本文使用联合使用正则化、剪枝以及模型量化的模型压缩策略。正则化诱导稀疏，剪枝产生稀疏，量化进行低精度存储。
3. 关于训练后量化的迁移：训练后量化方法一般是在模型训练完成后，对模型进行量化，使用量化模型进行推理。与常见的训练后量化方法不同本文在模型聚合的过程中对模型进行量化和反量化。客户端在收到量化后的聚合模型后，将其反量化为高精度模型，继续进行本地训练，扬长避短。

## 论文组织结构

本论文共分为5章，每章的组织结构安排和内容如下：

第1章是引言。本章简要介绍了联邦学习的基本知识、联邦学习情景中的通信瓶颈以及提高参数服务器和客户端之间通信效率的意义、国内外对于降低联邦学习过程中通信开销的研究现状、本文的主要贡献等。

第2章是相关工作。本章介绍了从模型压缩角度降低联邦学习通信开销的相关知识以及剪枝和量化的相关方法并提出了其中的问题。

第3章是剪枝算法和量化算法的算法分析与设计。本章在对以前的研究工作进行全面分析的基础上，将经典的模型压缩方法集成到了联邦学习场景中，并做出了适应性的调整。

第4章是联邦学习环境中模型压缩的算法实现。我利用模拟实验验证了该算法具有较低的时间延迟和较少的通信开销。

第5章给出了本文的结论以及未来工作。

# 相关工作

## 随机稀疏

随机稀疏是根据预先设定的稀疏模式，使用稀疏矩阵刻画本地更新的模型。该模式在每一轮中为每个客户端独立生成稀疏矩阵。

Shi等将训练算法与梯度稀疏相结合，提出更灵活的柔性稀疏法（FlexibleSparsification）：允许参与者只上传一小部分具有显著特性的梯度，从而减少每一轮的通信载荷。作者在5G移动设备上进行实验，结果表明该方法能耗更低，适用于资源受限的移动设备，但最终精度略低于FedAvg算法，这也反映了模型压缩的缺点：在降低通信开销的前提下，不可避免地牺牲部分精度，造成最终模型性能下降。

Sattler等基于非独立同分布、不平衡和小规模批次的本地数据，提出一种新型稀疏三元压缩（Spatio-TemporalContext，STC）框架，其中STC通过稀疏化、三元化、错误累积和最佳Golomb编码扩展当前的top-K梯度稀疏化的上行和下行压缩方法，在减少每一通信轮次传输数据量的同时还可以降低通信频率。

## 剪枝

剪枝是模型压缩研究的一个子领域。该类方法通常给出一个关于度量对象重要性的评价指标，将不重要的权重剪枝掉。神经网络模型中的度量对象一般有三种：权重、激活单元以及梯度。

### 3.1以权重为度量对象的剪枝方法

Yang He等关注对卷积层的滤波器进行剪枝。作者经过实验分析，认为基于范数的剪枝标准的有效性依赖于两个需求：滤波器的范数偏差应较大，滤波器的最小范数应很小。前者使得剪枝阈值T的搜索空间足够大，便于寻找合适的剪枝阈值；后者使得剪去的滤波器对推理结果的贡献极小。然而，现实中这两个需求并不能经常满足，因此，作者根据最接近几何平均的滤波器可以由其余滤波器表示这一依据，提出了一种使用几何平均对滤波器进行剪枝的方法FPGM，选择相对贡献最少的滤波器进行剪枝。

Michael等将大的剪枝的模型和小的稠密的模型进行比较，实验结果显示在占用存储空间相同的前提下，前者的精确度更高。

Jian-Hao Luo等提出的ThiNet框架在训练和推理阶段同时加速和压缩CNN模型。Luo将滤波器剪枝建立为一个优化问题，并且根据下一层而非当前层计算的统计信息来对滤波器进行重要性评价。作者通过实验表明，对于紧凑型的网络，ThiNet可以减少一半以上的参数和浮点运算，代价是Top5精度下降约1%。此外原始VGG-16模型可以进一步裁剪为一个非常小的模型，模型大小仅为5.05MB，保持了AlexNet级别的精度，但表现出更强的泛化能力。

### 3.2以激活单元为度量对象的剪枝方法

Jian-Hao Luo等通过一种新颖的滤波器剪枝策略对现有的卷积神经网络模型同时进行加速和压缩。滤波器的重要性由基于熵的方法进行评估，该方法可以减少中间激活单元的数量。同时，作者还提出了一种新的学习调度策略。无论对哪一层进行剪枝，都对网络进行一个或两个epoch的微调，使得性能得到一定的恢复，并且仅当最后一层被剪枝后，对网络进行多个epoch的重新训练。这样极大减少了重新训练时间，并且还防止了剪枝后的网络在早期阶段陷入局部最优。

### 3.3以梯度为度量对象的剪枝方法

Optimal Brain Damage和Optimal Brain Surgeon从损失函数出发寻找对损失影响最小的神经元进行剪枝。具体地，将目标函数进行泰勒展开，根据每个神经元剪枝后损失函数的变化量快速计算出神经元的重要程度，从而指导剪枝。

Lucas等提出了一种称为Fisher剪枝的贪婪剪枝方法。他们将知识蒸馏和Fisher剪枝相结合，生成了一种运行时高效的视线预测架构。这种架构与CAT2000数据集上的最先进网络相比，在保持模型精度的前提下实现了10倍的加速。

Pavlo等使用一阶和二阶泰勒展开来近似卷积神经网络中滤波器的贡献。这两种方法都可以跨任何网络层进行一致的扩展，而无需进行每层敏感性分析，并且可以应用于任何类型的层。

## 量化

量化最初用于数据压缩，能够有效压缩具有数百万参数的大型神经网络。量化一般分为基于聚类的量化、线性量化以及非线性量化。基于聚类的量化在本地计算好模型更新，对权重进行聚类，使用聚类索引来表示模型的参数；线性量化和非线性量化，直接对数据类型的表示范围进行映射，将模型参数量化成低精度的值以降低通信带宽。

通过量化本地计算的模型，将模型参数量化为索引值或低精度的值而非直接上传原始的参数值，能够降低每个通信轮次的通信比特数，从而有效地减小总的通信开销。

Yunchao Gong等对向量量化方法，包括二值化方法、基于k均值聚类的标量量化方法、乘积量化方法以及残差量化方法进行了介绍，在图像检索任务中压缩CNN模型，展示了向量量化方法相较于矩阵分解方法的优势。

Wenlin Chen等提出了一种新型的网络架构——HashedNets，其借鉴随机权重共享的方法，利用低开销的哈希函数随机均匀地将权重映射到哈希桶中进行共享，从而实现对模型的压缩。

Song Han等提出了一种深度压缩方式，其包含剪枝、量化以及哈夫曼编码三个阶段。剪枝阶段只学习重要的连接，减少连接的数量；量化阶段进行权重共享，减少每个连接的比特数；哈夫曼编码利用有效权重分布的有偏性进一步压缩模型。

## 联邦蒸馏

2015年，Hinton等提出知识蒸馏法（KnowledgeDistillation）：先利用大规模数据训练得到一个教师网络，然后将教师网络的知识迁移到学生网络上，使得学生网络获得可与教师网络媲美的性能；并以手写数字识别和语音识别为例，验证了知识蒸馏方法的有效性及模型的泛化能力。而后，Jeong等提出了联邦蒸馏（FederatedDistillation），其原理是只交换局部的模型输出而非交换传统联邦学习采用的模型参数，这些输出的尺寸通常比模型尺寸小得多，因此可以减少通信开销。

Sattler等利用知识蒸馏的协同蒸馏（CooperatedDistillation，CD）的关键原理，提出压缩联邦蒸馏方法（CompressedFederatedDistillation，CFD），可以将实现固定性能目标所需的累积通信量从8570MB减少到0.81MB，相当于通信量减少至原来的0.009%。目前，联邦蒸馏可以大幅减少通信代价，适用于缺少标签的异质数据、异构模型的场景。联邦蒸馏方法也有其局限性：当两个网络模型大小相差太大时，知识蒸馏会失效，此外，交换输出还有可能增加用户隐私泄露的风险。因此，联邦蒸馏的收敛性和应用性研究需要进一步研究。

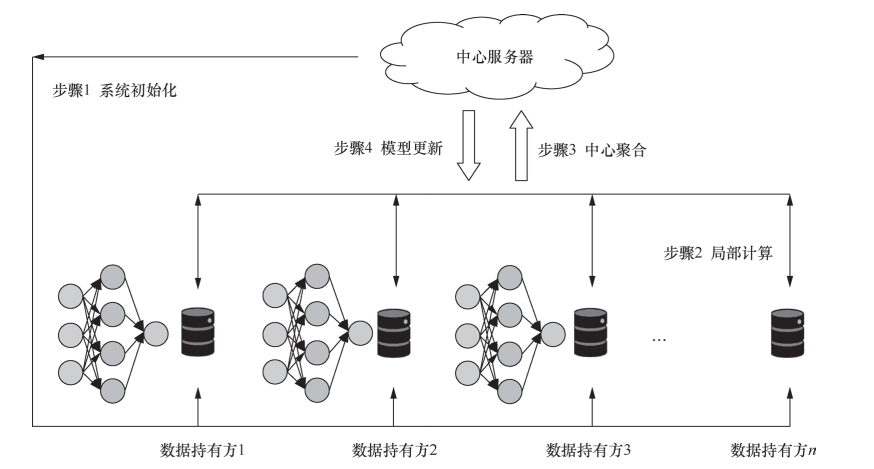
# 联邦学习高效通信方法

## 联邦学习系统（介绍）

### 1.1联邦学习流程

传统的机器学习算法需要用户将源数据上传到高算力的云服务器上集中训练，这种方式导致了数据流向的不可控和敏感数据泄露问题。谷歌研究院在2016年提出联邦学习技术，其本质上是一种借助多个参与方的本地数据、联合训练一个全局模型的分布式机器学习架构，在中央服务器的协调下，多个参与方联合完成机器学习任务，其工作流程描述如下:

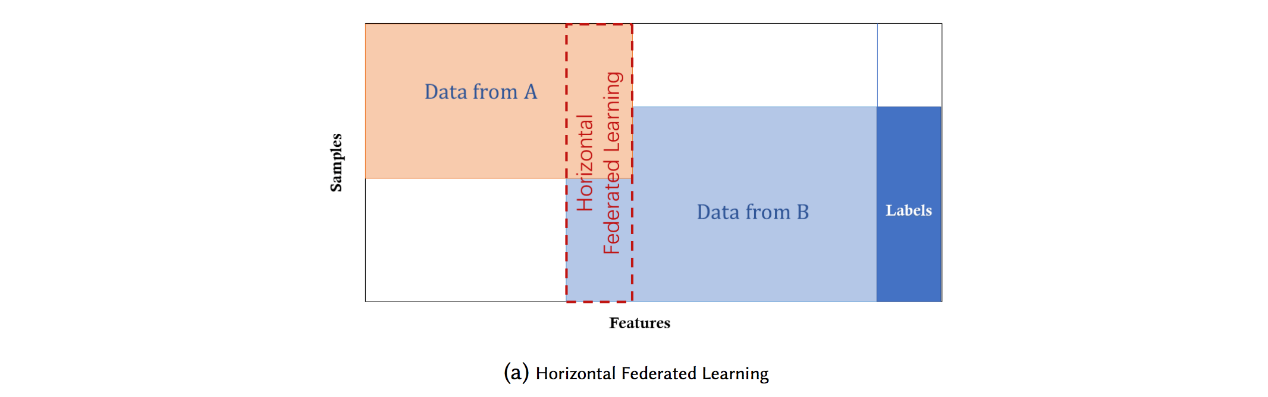
1. 系统初始化：首先由中心服务器发送建模任务，寻求参与客户端。在与其他合作数据持有方达成协议后，各数据持有方进入联合建模过程。由中心服务器向各数据持有方发布初始参数。
2. 局部计算：联合建模任务开启并初始化系统参数后，各数据持有方在本地根据己方数据进行局部计算。当各数据持有方完成局部计算后，将本地局部计算所得模型参数脱敏后进行上传，以用于全局模型的一次更新。
3. 中心聚合：在收到来自多个数据持有方的计算结果后，中心服务器对这些计算值进行聚合操作，在聚合过程中需要同时考虑效率、安全、隐私等方面的问题。
4. 模型更新：中心服务器根据聚合后的结果对全局模型进行一次更新，并将更新后的模型返回给参与建模的数据持有方。数据持有方更新本地模型，并开启下一步局部计算，同时评估更新后的模型性能，当性能足够好时，训练终止，联合建模结束。建立好的全局模型将会被保留在中心服务器端，以进行后续的预测或分类工作。



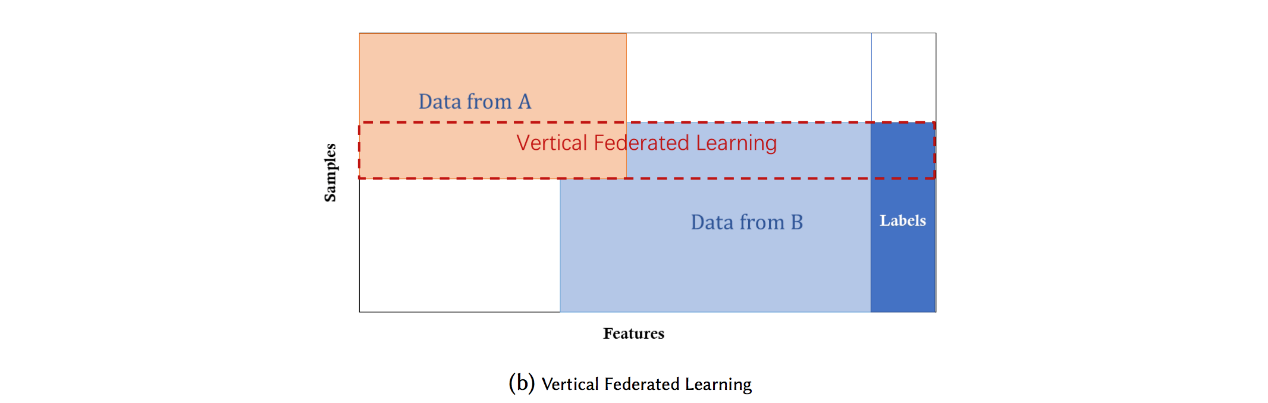
### 1.2联邦学习的分类

根据参与联合建模的客户端的数据集特征信息的不同，可将联邦学习分为横向联邦学习、纵向联邦学习和联邦迁移学习。

横向联邦学习也称为特征对齐的联邦学习（Feature-Aligned Federated Learning），即横向联邦学习的参与者的数据特征是对齐的，如图(a)所示。横向联邦学习适用于参与者的数据特征重叠较多，而样本ID重叠较少的情况，例如，两家不同地区的银行的客户数据。联合多个参与者的具有相同特征的多行样本进行联邦学习，即各个参与者的训练数据是横向划分的，称为横向联邦学习。横向联邦学习使训练样本的总数量增加。



纵向联邦学习也称为样本对齐的联邦学习（Sample-Aligned Federated Learning），即纵向联邦学习的参与者的训练样本是对齐的。纵向联邦学习适用于参与者训练样本ID重叠较多，而数据特征重叠较少的情况，例如，同一地区的银行和电商的共同的客户数据。纵向联邦学习使得训练样本的特征维度增多。

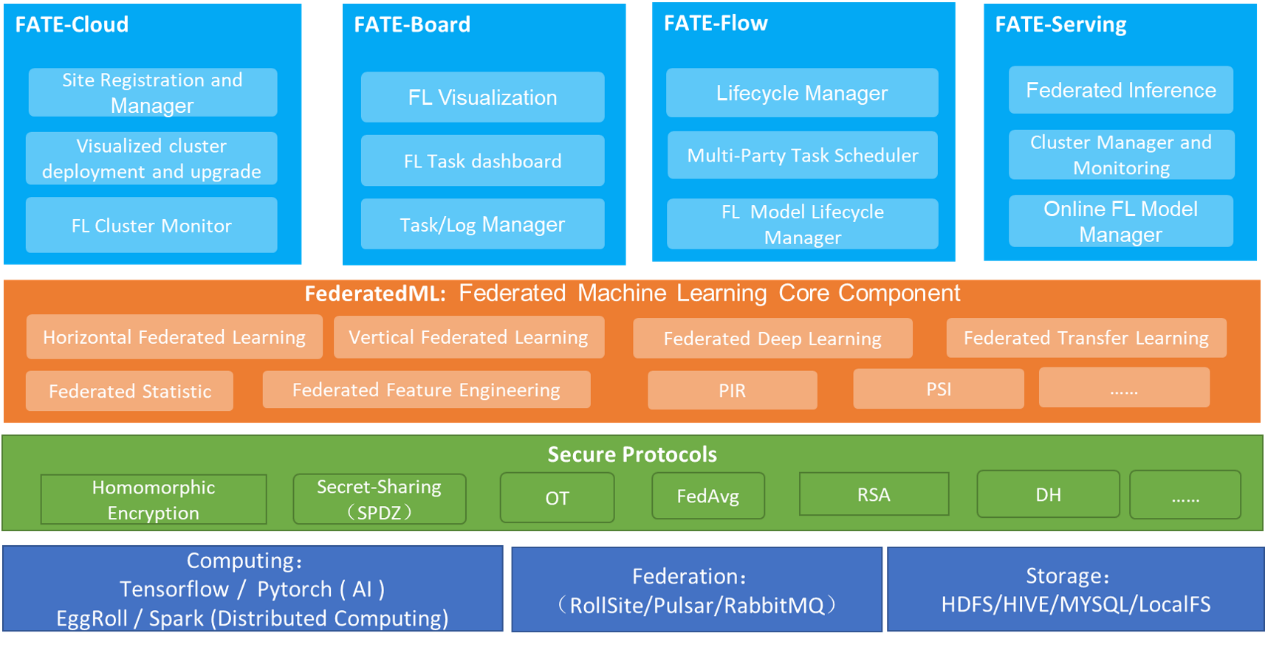


联邦迁移学习的特点是数据集特征、标签信息和样本ID信息都不同。联邦迁移学习被用于解决标签样本少和数据集不足的问题，如中国的电商平台与其他国家银行之间的数据迁移，由于跨部门跨国的数据交流很难实现，通过联邦迁移学习可以很好地解决这类痛点问题。

本文主要关注横向联邦学习的场景。

### 1.3联邦学习框架FATE

FATE（FederatedAITechnologyEnabler）是全球首个联邦学习工业级开源框架，可以在保护数据安全和数据隐私的前提下进行数据协作。FATE项目使用多方安全计算 (MPC) 以及同态加密 (HE) 技术构建底层安全计算协议，以此支持不同种类的机器学习的安全计算，包括逻辑回归、基于树的算法、深度学习和迁移学习等。



关键组件介绍

1. FATE Flow：FATE Flow一个联邦学习端到端全流程的多方联合任务安全调度平台，基于共享状态调度架构和跨数据中心的多方安全通信。在FATE框架中，FATE Flow提供数据接入、任务组件注册、联合作业调度、多方资源协调等生产级的服务能力。
2. FATE Board：FATE Board是联邦学习建模的可视化工具，旨在帮助用户简单而高效地理解模型与探索模型。为了便于理解、跟踪、调试和探索联邦学习建模过程，以及检查、评估和对比各种联邦学习模型。FATEBoard提供一种可视化的方式来探查模型，使得用户可以更有效地调试和改善模型效果。
3. FederatedML：FederatedML是FATE框架运行机器学习任务的核心组件，其中包含许多常见机器学习算法的联邦化实现，所有模块均采用去耦的模块化方法开发，以增强模块的可扩展性。

FATE框架建模场景中的角色

1. Guest：Guest表示数据应用方，在纵向算法中，Guest是有数据标签的一方。
2. Host：Host表示数据提供方。
3. Arbiter：辅助多方完成联合建模，可以认为是标准客户端—服务器架构中的参数服务器。其主要作用是对梯度或模型进行聚合、分发公钥私钥、进行加密解密服务等。
4. 在横向联邦学习场景中，可以认为Guest和Host角色等价。

## 模型压缩算法

### 2.1剪枝算法

神经网络有大量的研究表明，神经网络模型是过度参数化的，即包含许多冗余的逻辑和特征，通过在神经网络参数中诱导稀疏性可以在一定程度上消除冗余。剪枝就是一种在权重和激活单元中诱导稀疏性的常见的方法，它通过只学习重要的连接、减少模型中非零值参数的数量来减少神经网络模型的存储大小。由于联邦学习环境中模型的聚合其实就是模型参数或梯度的聚合，因此，对模型参数进行剪枝能够有效地减少联邦学习环境中模型聚合时的通信开销。

剪枝的对象通常包含权重、偏置以及激活单元。偏置通常是非常小的，并且对一个网络层输出的贡献相对较大，因此一般不对偏置进行剪枝。神经网络一般使用ReLU层作为激活函数：ReLU层会保持激活单元的正值不变，将负值置为0。因此，激活单元在经过ReLU层的作用后，也通常具有良好的稀疏性。而权重的稀疏水平并不高，尽管权重一般都很小，但是通常都是非零值。因此，模型剪枝主要关注的是对模型权重的剪枝。

剪枝的过程一般可以分为三步：首先，训练网络来学习哪些连接是重要的；其次，剪枝掉不重要的连接，将稠密层转成稀疏层；最后，重新训练网络对剩余连接的权重进行微调，使得剩余连接能够弥补移除连接造成的模型精度的损失。其中剪枝和重训练的过程可以进行多次的重复，便于更好地压缩模型。事实上，剪枝和重训练的过程十分类似于哺乳动物幼年期到成年期之间发生的突触消除过程：在幼年期，突触逐渐生成，而在成年的过程中，激活次数很少的突触会逐渐被剪掉，不必要的神经元结构会被去除，最终形成成年期的神经网络结构。

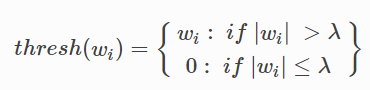
剪枝方法根据剪枝的粒度可以分为非结构化剪枝和结构化剪枝。非结构化剪枝是对单个的权重元素进行剪枝，它也被称作细粒度的剪枝。结构化剪枝是直接对一组元素进行剪枝，以卷积层为例，结构化剪枝以滤波器为单元或者通道为单元进行剪枝。

### 2.1.1非结构化的剪枝

本文实现的非结构化剪枝包括：基于绝对值的剪枝方法、基于稀疏水平的剪枝方法、基于敏感性的剪枝方法、基于稀疏水平的剪枝方法、自动化的剪枝方法、带有拼接操作的剪枝方法；结构化剪枝包括：基于Lp范数的滤波器或通道剪枝方法、基于APoZ的滤波器剪枝方法。

根据以上介绍的剪枝过程可知，剪枝需要一个指标评价神经网络权重的重要性，从而将不重要的网络权重剪枝。最常见的剪枝标准是每个元素的绝对值，即该元素的L1范数。如果一个权重具有较小的L1范数值，则认为该权重对最终结果的贡献比较小，将该权重所代表的连接移除掉。

**基于绝对值的剪枝方法。**该方法对每个权重执行一个thresh(·)函数：

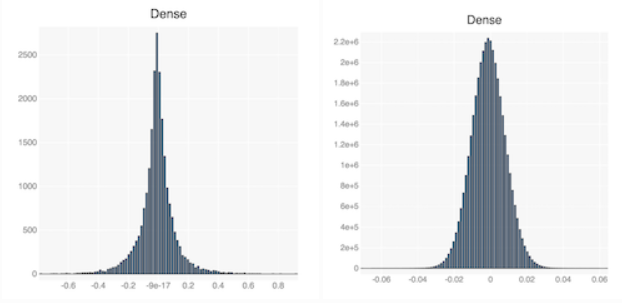


给定一个阈值λ，如果该权重的绝对值小于阈值λ，则对该权重进行剪枝。

**基于稀疏水平的剪枝方法。**通过剪枝诱导稀疏性的难点在于剪枝阈值的确定。基于稀疏水平的剪枝方法通过指定目标稀疏水平而非权重阈值进行剪枝。在指定目标稀疏水平后，该方法对权重进行基于绝对值的升序排序，将绝对值最小的一部分权重置为0，使得权重的稀疏度达到目标稀疏水平。由于目标稀疏水平并不与元素的绝对值耦合，因此，该方法比基于绝对值的剪枝方法要稳定得多。

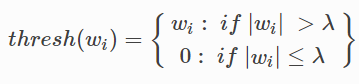
**基于敏感性的剪枝方法。**由于每个层的权重值以及其分布都不相同，因此为每个层确定一个阈值是十分困难的，然而，可以利用卷积层和全连接层的权重都近似呈现出均值为0的高斯分布的事实，根据每一层权重的分布特点分别确定对应层的剪枝阈值。

下面的图显示了TorchVison预训练的Alexnet模型中第一个卷积层和第一个全连接层权重向量的元素分布，可以看出其分布近似于均值为0的高斯分布。



根据卷积层和全连接层权重近似呈均值为0的高斯分布 的事实，使用权重张量的标准差σ作为是不同权重张量的正则因子。例如，如果一个权重张量近似符合均值为0的正态分布，则该张量中大约68%的元素的绝对值小于权重张量的标准差。因此，可以将剪枝阈值设置为s\*σ，表示剪枝掉s\*68%的元素。

该方法可以形式化表示为：



其中λ=s\*σ，其中σ是第l层权重向量的标准差。

敏感度分析可以根据每一层参数张量的剪枝敏感程度对其进行排序，从而指导各个层剪枝阈值的确定。敏感度分析的基本操作是为一个特定的层设立稀疏水平后执行剪枝训练，然后在测试数据集上进行评估，记录剪枝模型的精度。对所有参数层设定多个稀疏水平，分别执行以上的操作，得到每个参数层执行剪枝后的模型精度随稀疏水平的变化情况，从而对剪枝阈值在每个参数层上的确定起到一定的指导作用。

该方法是不稳定的，因为其稀疏水平依赖于权重分布的收敛模式。

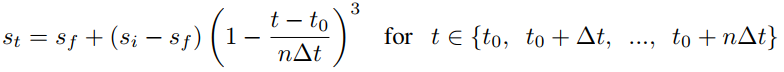
**自动化的剪枝方法。**

基于绝对值的剪枝方法、基于稀疏水平的剪枝方法以及基于敏感度的剪枝方法，在一个训练迭代内的稀疏水平是固定不变的。如果设置的稀疏水平过低，模型的压缩率不高，没有充分挖掘模型的压缩极限；如果设置的稀疏水平过高，由于一次性剪枝掉过多的权重，模型的精度损失可能很大。因此，为了在保证模型精度的同时尽可能地对模型进行压缩，需要一次性精准设置好每个层的稀疏水平。显然，这个要求很难达到。

自动化的剪枝方法能够在一个训练迭代内的多个批次之间逐步增加稀疏水平。该方法在给定初始稀疏水平si和最终稀疏水平sf后，能够比较合理地根据当前训练迭代轮次地的批次数自动计算出当前批次的稀疏水平，使得网络的稀疏水平平滑过渡，尽可能避免剪枝后模型精度骤降的情况出现。

自动化剪枝方法的算法描述如下：

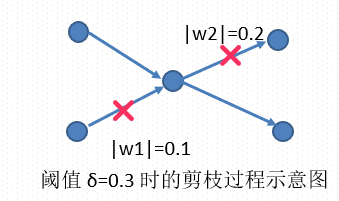
给定：初始稀疏水平si，最终稀疏水平sf，剪枝次数为n，剪枝频率为Δt，当第t次剪枝时的稀疏水平为：



使用上式给定每个批次的稀疏水平，能够在训练的初始阶段对具有较多冗余的网络进行快速剪枝。随着网络中非零权重的数量逐渐减小，稀疏水平的增加也越来越缓慢，当达到最终的稀疏水平时，掩膜权重不会再进行更新。通过设置剪枝间隔Δt，使得网络的精度能从由于剪枝造成的精度损失中恢复。

**带有拼接操作的剪枝方法。**

剪枝方法还可以分为贪婪剪枝和非贪婪剪枝。贪婪剪枝方法使用神经网络连接在某一时刻的重要性评估作为判断连接重要性的唯一指标。然而，由于隐藏神经元之间的相互连接，网络中参数的重要性很难测量，并且一旦剪枝过程开始，参数的重要程度可能会急剧改变。



因此，贪婪剪枝方法可能会导致以下两个主要问题：①造成不可逆的网络损伤：由于贪婪剪枝方法剪掉的连接无法恢复，不正确的剪枝可能造成严重的精度损失，因此通常需要过度降低压缩率来避免模型精度的显著降低。②网络学习很低效，这意味着需要交替执行很多次的剪枝与重训练过程。每个重训练过程包含很多次迭代，这将是非常耗时的。

为了处理以上的问题，带有拼接的剪枝方法使用连续的网络结构维持方法来切断冗余的连接，即将连接的拼接操作加入到剪枝过程中来避免不正确的剪枝。该方法主要包含两个关键操作：剪枝和拼接。剪枝操作通过移除冗余连接对网络模型进行压缩，而拼接操作则是当被剪枝的连接变得重要时，恢复该连接，来弥补不正确剪枝造成的意外损失。剪枝和拼接操作交替进行，动态地将网络连接分成两类，这类似于人类神经系统中兴奋型神经递质和抑制型神经递质的组合。

带有拼接操作的剪枝方法的算法描述如下：

将深度神经网络模型表示为{Wk：0<=k<=C}，其中Wk表示第k层连接的权重矩阵。对于有着p维输入和q维输出的全连接层来说，Wk的维度是pk\*qk；对卷积层来说，将每个核展开后进行连结，表示为Wk矩阵。使用{Wk,Tk：0<=k<=C}一个剪枝的稀疏模型，其中Tk是一个二值矩阵，每个元素表示网络连接的状态。Tk是权重矩阵Wk的掩膜矩阵。

以第k层为例，可以将剪枝后的神经网络模型的训练抽象成解决以下的优化问题：

（1）

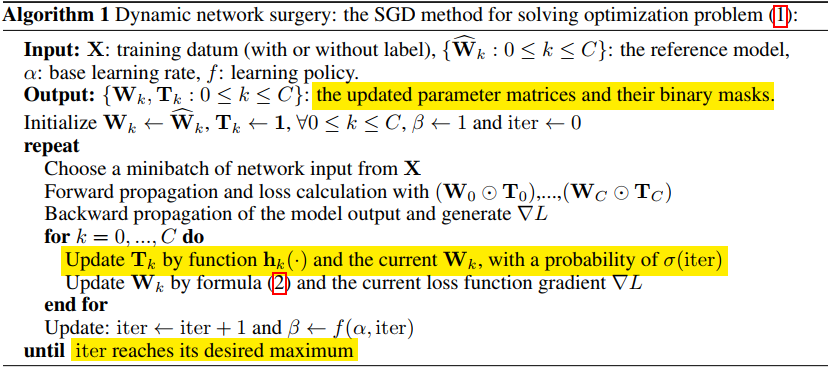
其中L是网络损失函数，表示哈达玛积，I包含Wk中所有的索引项，hk是一个指示函数：如果参数w在当前层是重要的，则hk(w)=1，否则为0。

根据梯度下降法，Wk的更新方式如下：

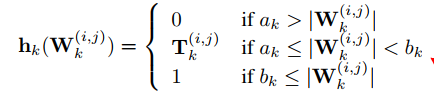
（2）

此处β表示学习率。该方法不仅更新重要的参数，还更新那些对应Tk项为0的参数，从而实现对之前剪枝掉的连接的拼接。偏微分（2）可以通过随机选择小批量样本的链式法则计算，这可以由现有深度学习框架PyTorch或TensorFlow的自动微分机制实现。在一个优化迭代步更新Wk后，使用新的权重矩阵Wk重新计算掩膜矩阵Tk，进而重新计算整个网络的激活单元以及损失函数的梯度。重复以上步骤，即可完成稀疏网络的学习过程。

带有拼接操作的剪枝算法的伪代码：



参数重要性度量影响着网络连接的状态，因此，hk(·)对算法来说很重要。为了增强算法的鲁棒性，可以使用双阈值的方法来就进行更加平滑的剪枝过渡。



其中，ak和bk可以使用敏感度分析方法进行设定。

所提出的算法比标准的反向传播方法更加复杂，可能会出现收敛速度变慢的情况。可以采取以下几个步骤加速收敛：

* 1. 放慢剪枝和拼接的频率。相比于基于优化函数的梯度下降，人为引入的剪枝和拼接操作可能会给模型精度带来较大损伤，因此，剪枝和拼接操作不宜太过频繁进行。在算法中，通过设置概率p=δ(iter)，随机触发Tk的更新。δ函数是单调非递增的并且满足δ(0)=1。在经过足够长的迭代后，概率p可以设置为0。
  2. 分别对卷积层和全连接层进行剪枝。当大量的连接被剪枝时，网络结构变得相当简单，导致损失函数的梯度非常小，引起梯度消失问题，导致收敛速度较慢。

### 2.1.2结构化剪枝

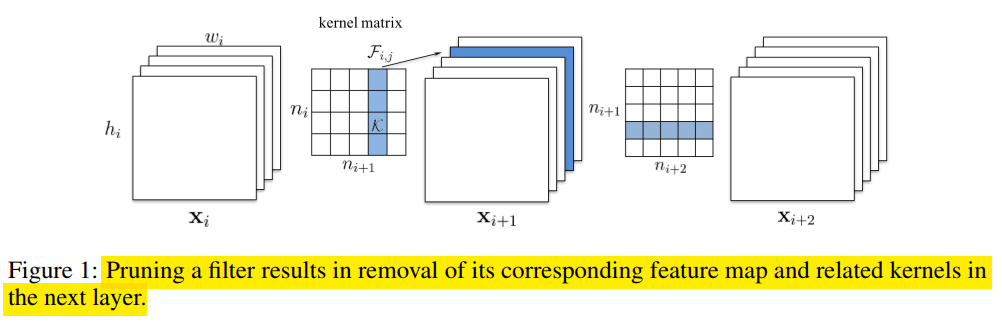
结构化剪枝可以分为对权重的结构化剪枝和对激活单元的结构化剪枝。

**对权重的结构化剪枝**

对权重的非结构化剪枝通过诱导不规则的稀疏性，能够显著减小全连接层的参数数量。但是由于大多数移除的参数来自于全连接层，而全连接层的计算开销相对卷积层来说不高，例如VGG-16中的全连接层拥有整个网络90%的参数量，然后其浮点操作的计算量（FLOP）不足总计算量的1%。因此对权重的非结构化剪枝并不能显著下降整个网络的计算开销。

卷积神经网络在不同的滤波器和特征通道之间通常也有显著的冗余。对于卷积层的剪枝，一般使用结构化的剪枝方法：通过移除网络中的相对不重要的滤波器以及对应的输出特征映射，可以显著降低卷积层的存储开销和计算开销。

使用ni表示第i个卷积层输入通道的数量，hi和wi分别表示输入特征映射的高度和宽度。卷积层通过对ni个输入通道分别使用ni+1个3D滤波器Fi,j∈R(ni\*k\*k)，将输入特征映射xi∈R(ni\*hi\*wi)转换为输出特征映射xi+1∈R(ni+1\*hi+1\*wi+1)，xi+1同时也是下一个卷积层的输入特征映射。所有的滤波器共同组成了核矩阵Fi∈R(ni\*ni+1\*k\*k)。如下图所示，当滤波器Fi,j被剪枝后，它对应的特征映射xi+1,j也会被移除，而且在下一个卷积层作用在该特征映射上的滤波器也会被移除。



在结构化剪枝中，一般也是采用基于绝对值的剪枝方法。即通过计算滤波器中卷积核的权重绝对值的和来衡量滤波器的相对重要性。一般来说，有着更小的卷积核权重的滤波器相比于同层的其他卷积核来说对输出特征映射的贡献更低，因此，认为它是相对不重要的。

在第i个卷积层剪掉m个滤波器的过程如下：

1. 对于每个滤波器Fi,j，计算它的卷积核权重的绝对值之和
2. 使用sj对滤波器进行排序。
3. 减去sj指标最小的m个滤波器以及其对应的输出特征映射。

为了理解每个层的剪枝敏感度，仍然需要对每个层进行独立的剪枝，然后评价剪枝网络在验证集上的模型精度。根据每个层对滤波器剪枝的敏感程度。

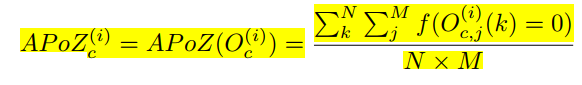
**对激活单元的结构化剪枝。**

不同于对权重的剪枝，该类方法的剪枝对象是激活单元。对于卷积神经网络，无论该网络接收的输入是什么，其很大一部分神经元的输出接近零值。这些神经元的存在会带来很大的存储开销和计算开销，还可能会增加模型过拟合的风险以及网络优化的难度。

以连接为剪枝对象的方法也可能会剪枝掉一些不重要的神经元，但仅当一个神经元的所有入连接和出连接被移除时，该神经元才会被移除。由于大多数冗余发生在全连接层之间以及最后一个卷积层和第一个全连接层之间，而且和一个神经元相连的连接都被移除的情况是罕见的，因此，以连接为剪枝对象的剪枝方法对网络中神经元的剪枝是非常低效的。为了对神经元进行高效的剪枝，使用对激活单元的结构化剪枝方法。

在模型训练过程中，权重的生成和更新是不直接依赖于输入数据的，因此，对模型的参数剪枝可以在网络训练的过程中进行。而激活单元的计算直接依赖于输入数据，这意味着对激活单元的剪枝无法在线完成。因此，该类剪枝方法一般需要在大数据集上分析激活单元的值，再对不重要的神经元进行剪枝。

在对激活单元的结构化剪枝中，对神经元进行评价的常见指标是平均零值占比(APOZ，Average Percentage of Zeros)。定义平均零值占比为经过ReLU层映射后神经元中零激活值的比例。以卷积层为例，令Oc(i)为第i层第c个通道的输出，则第i层第c个神经元（输出特征映射）的ApoZ值定义为：



其中，M是输出特征映射的维度，N为是验证集上样本的数量。

对神经元的结构化剪枝使用APoZ指标来评估网络中每个神经元的重要性。由于神经网络的计算过程一般是乘法-加法-激活，因此，可以认为大多数元素为0的神经元对后续层的输出以及最终结果几乎没有贡献。

使用APoZ评价指标对卷积神经网络的神经元的结构化剪枝方法一般包含三个步骤：首先，使用传统的过程训练网络，然后，在大型验证数据集上运行网络来计算每个神经元的APoZ值。根据设定的APoZ阈值剪掉具有高APoZ值的神经元，同时，该神经元的入连接以及出连接也都被移除。最后，重新训练网络使得剪枝网络的精度得到恢复。

关于剪枝阈值的确定也可以使用敏感度分析。经验上设置为高于APoZ平均值一个标准偏移。假设APoZ值服从高斯分布，这样大概会拒绝16%的ApoZ值较高的神经元。

### 2.2量化算法

量化一般分为基于聚类的量化、线性量化以及非线性量化：前者在本地计算好模型更新后对权重进行聚类，使用聚类索引来表示模型的参数；后者将模型参数量化成低精度的值以降低通信带宽。本文仅对线性量化方法进行讨论。

### 2.2.1线性量化的基本知识

线性量化将连续取值的浮点型模型权重或流经模型的张量数据定点近似为有限个离散值，以达到减少模型尺寸大小、减少模型内存消耗及加快模型推理速度等目的。在深度学习环境中，用于研究和部署的主要的数据格式是32位浮点数，即FLOAT32。然而，由于减少带宽的需要，近来的研究更多地使用低精度的数据类型。

在深度学习模型中，不同层之间的权重和激活单元的分布通常有着显著差异。为了使用整型格式表示这些不同的分布，一般使用放缩因子将张量的范围映射到整型区间。为了降低映射过程中精度的损失，放缩因子通常为浮点数。

关于量化后的计算，需要处理溢出的问题。在深度神经网络模型中，卷积层和全连接层通常将中间结果存储在累加器中。如果对权重、激活单元以及累加器使用相同的比特带宽，由于低比特的整型数据格式能够表示的数据范围和数据精度有限，中间结果会很快溢出。因此，累加器通常使用更高的比特带宽实现。实际中，一般将激活单元和权重量化为8比特整型的数据格式，将偏置以及累加的中间结果量化为32比特整型的数据格式。

**非对称量化模式**

图表, 折线图

描述已自动生成

在非对称量化模式中，将浮点范围的最小值和最大值分别映射到整型范围的最小值和最大值。除了放缩因子之外，该模式还引入零点来实现这种映射，零点也称为量化偏差和偏移量。

使用xf表示原来的浮点张量，xq表示量化后的张量，qx表示放缩因子，zpx表示零点，n表示量化使用的比特数目，则有下式成立：

Word

低可信度描述已自动生成

实际上，通常将使用round(minxfqx)作为zpx的值，这意味着零是由量化范围内的一个整数精确表示。舍入零点实际上对浮点范围内的最小值和最大值进行了微调。

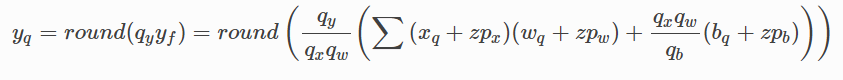
为方便起见，上式使用无符号整数来表示量化范围，即xq∈[0,2^n-1]。也可以使用有符号整数表示，其xq∈[-2^(n-1),2^(n-1)-1]。

使用非对称模式来量化卷积层和全连接层：将输入、输出、权重和偏置分别记为x、y、w和b，则卷积层和全连接层的计算有以下的量化表示：

图形用户界面, 应用程序, 表格

描述已自动生成

因此，有：



在累加求和的过程中，偏置必须进行重新的放缩来匹配累加器的放缩范围。

**对称量化模式**

图表, 折线图

描述已自动生成

对称量化模式中，不将浮点范围的最小值和最大值映射到量化范围，而选择浮点张量的最大值和最小值中绝对值的最大值作为浮点边界。除此之外，在对称量化模式中，没有必要使用零点，因此，需要量化的浮点范围以及量化后的整型范围都是关于零点对称的。

使用与非对称模式中相同的标记，有下式成立：



卷积层和全连接层在对称模式量化下的表示为：

日程表

低可信度描述已自动生成

因此，有：

图片包含 图示

描述已自动生成

**非对称量化模式vs对称量化模式**

可以从简易性和量化范围利用率两种角度探讨对称量化和非对称量化的区别。

1. 量化范围利用率：当使用非对称量化模式时，量化范围是完全利用的。因此，可以精准地将浮点范围地最小值和最大值映射到量化范围的最小值和最大值。而使用对称量化模式时，如果浮点范围的分布偏向某一侧，会导致很大的量化范围被未观测到的浮点范围的值所占据。以ReLU层为例，当激活单元经过ReLU层后，其中的负值会被置为0，这时，如果使用对称模式量化，会浪费一个比特。
2. 另一方面，在卷积层和全连接层中，对称量化模式的实现是相当简单的。在非对称模式中，需要执行和零点相关的一些计算，会带来额外的计算负担。

一般来说，对于权重的量化使用对称量化，而对于激活单元的量化使用非对称量化。对于权重而言，一般认为其数据分布是对称的，因此使用对称量化可以进一步有效降低量化时的计算量，使得数据分布更加符合真实分布。对于激活单元，因为网络中通常使用ReLU或ReLU6作为神经网络模型的激活函数，如果采用对称量化模式会加剧数据的离散程度，并且会造成数据的浪费。

### 2.2.2模型压缩中的量化方案

量化方案主要分为两种：量化感知训练和训练后量化。其中，量化感知训练也被称为在线量化，训练后量化也被称为离线量化。

### 训练后量化

为了实现FP32格式到INT8格式的模型量化，需要根据每个层的每个张量计算放缩因子。最简单的情况是将浮点张量的最小值和最大值分别映射到整型范围的最小值和最大值。当模型训练完成后，权重和偏置的值都已经确定，因此，对于权重和偏置来说，这种映射是很容易的。然而，由于激活单元的值依赖于输入数据，因此，激活单元的边界值的确定是比较困难的。

对于激活单元来说，通常有离线和在线两种方式确定其边界值。离线方式是在模型的训练阶段收集激活单元的统计数据。当部署模型时，根据训练阶段收集到的激活单元的统计数据来计算激活单元的放缩因子。然而，在实际的推理阶段，激活单元的值可能落在观测到的范围之外，在这种情况下，需要对范围外的值进行钳制，可能会导致模型精度的下降。在线方式是在推理阶段动态地计算激活单元的边界值，然而，即时计算可能会产生比较大的计算开销。

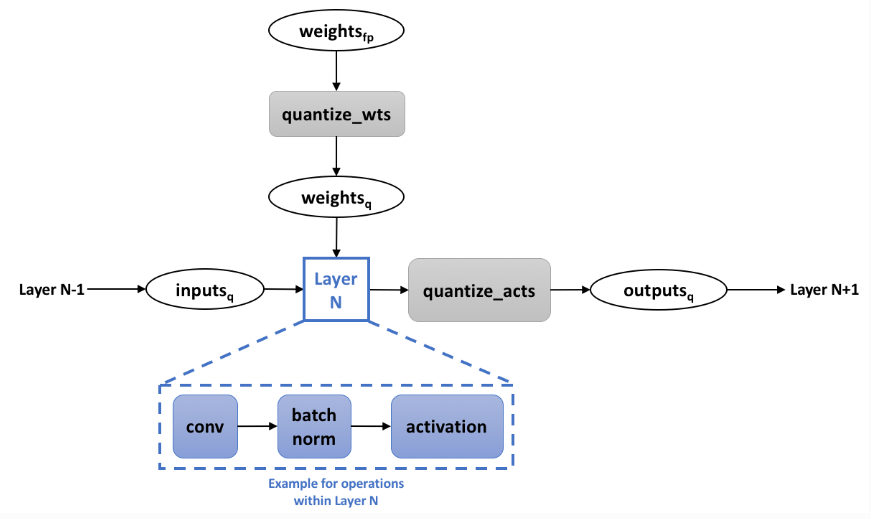
训练后量化的一般过程是：

1. 模型训练完成后，对网络模型的权重进行统计量化，对数据类型进行转换。
2. 当使用量化模型进行预测时，对网络模型中的权重进行反量化操作，使其变成原来的高精度数据类型，再进行正常的推理。

### 量化感知训练

训练后量化在模型训练完成后直接进行量化，这样虽然可以有效地降低模型地参数大小和内存消耗，但往往会带来比较大的精度损失。为了最小化量化所带来的损失，可以将量化操作嵌入到训练过程之中，即在训练过程中对权重和激活单元进行量化。

以第N层的卷积层为例，量化感知训练的示意图如下：



如图所示，在训练过程中模型会维护一个权重的浮点数版本的拷贝，其目的是无损地累积训练过程中梯度的微小变化。当模型训练好之后，模型推理阶段仅使用量化后的权重进行推理。

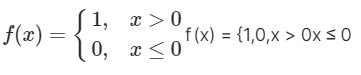
在模型训练的过程中，卷积等操作仍使用高精度的数据类型，因此，量化感知训练有时也称作“模拟量化”。

量化感知训练过程还会在对应层的输入节点和输出节点引入伪量化操作。伪量化操作主要有两个作用：一是统计训练时流经该节点数据的最大最小值，便于在使用端侧转换工具时，转换成端侧需要的格式进行量化。二是参与模型训练的前向传播过程，使优化器感知到伪量化操作带来的模型损失，从而对伪量化操作进行反向传播学习，提高量化后模型的精度。

**直通估计量**

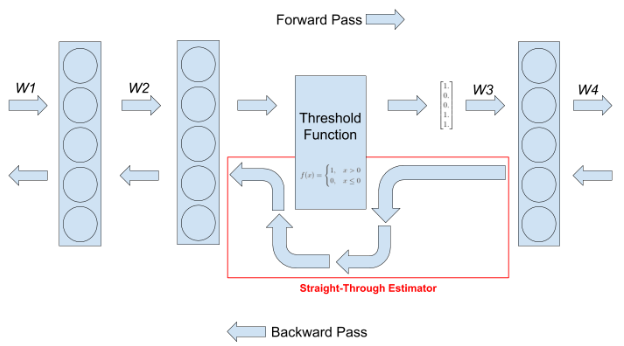
在量化感知训练情景中，一个重要的问题是如何对离散的量化函数进行反向传播。

假设现在需要使用下列函数对一个层的激活单元进行二值化：



显然，以上函数每个点的梯度几乎都为0。因此，使用其梯度将严重阻碍模型参数的学习过程，甚至使模型训练无法进行。对于这类问题，通常的解决方案是使用直通估计量（straight-throughestimator，STE）。

直通估计量是对一个函数的梯度进行估计。具体来说，它忽略阈值函数的导数，并传递传入的梯度。在反向传播阶段，该阈值函数可以认为是一个恒等函数。



## 2.3正则化

### 2.3.1正则化定义

Goodfellow在其著作《Deeplearning》中将正则化定义为：“为了减小学习算法的泛化误差所做的任何修改”。正则项一般作用在损失函数上，即：

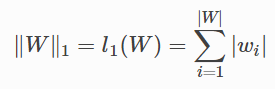


其中，W是网络中所有权重的元素的集合loss(W;x;y)是总训练损失，lossD(W;x;y)是给定数据上目标函数的损失。λR是正则化的强度，它对数据误差以及正则项误差进行平衡。

### 2.3.2稀疏水平和正则化

正则化和一些深度神经网络中稀疏诱导方法有紧密的联系。正则化可以被用来引诱稀疏性。

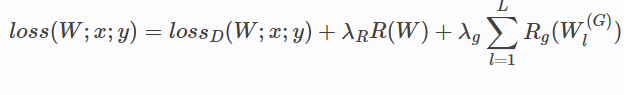
为了诱导元素的稀疏性，可以使用l1范数：



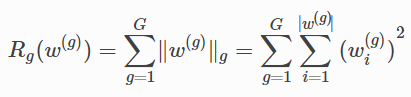
L2范数正则化通过减小比较大的参数来防止过拟合以及提高模型的精度，但是它并不要求参数是零值。而l1范数正则化通过将一些参数元素设置为0，使得模型变得更简单。

### 2.3.3组正则化

组正则化是对整组参数进行惩罚，而不是组内单个元素。将第l层所有的参数组表示为，并且为所有层的所有组添加惩罚项，即：



记组g中的所有权重元素为，则有：



其中且是中元素的数量。也被称为组Lasso正则项。组Lasso正则项会引诱组稀疏性，组稀疏性由于其规整性，能够提高模型的推理速度。

## 算法概要设计

### 联邦学习环境中客户端训练逻辑

在横向联邦学习环境中，客户端的主要任务是在本地数据集上训练模型，并且在聚合轮次将训练好的模型以及该模型在验证数据集上的精度和损失数据提交给服务器端。在服务器端完成模型聚合后，接受服务器端分发的聚合模型，并继续在本地数据集上进行下一轮次的训练。

### 联邦学习环境中服务器端聚合逻辑

横向联邦学习环境中，服务器端的主要任务是接收客户端发送的模型，然后使用模型聚合算法对客户端发送的模型进行聚合，并将聚合后的模型分发给各个客户端。

除此之外，服务器端还需要接收客户端发送的损失数据，根据损失数据判断聚合模型的收敛情况，当模型已经收敛时停止训练，防止过拟合。

### 剪枝算法的概要设计

为了减小模型聚合过程中客户端与服务器端的通信开销，引入模型剪枝算法来提高模型的稀疏性，降低模型的存储占用。

剪枝算法主要作用在模型的训练过程中。当开始一轮epoch训练时，模型压缩调度器会根据配置好的剪枝策略生成权重的掩膜矩阵；当开始每个批次时，模型压缩调度器会对权重矩阵与掩膜矩阵计算哈达玛积，即将需要剪枝的权重置为0。然后在剪枝后的模型上计算输出，并进行模型的更新。

卷积层和全连接层的区别，在哪里提及？

### 量化算法的概要设计

【训练后量化部分】

本文使用的量化算法属于线性量化算法，并且仅作用在聚合阶段，即训练后的量化。在聚合阶段，压缩调度器根据训练阶段收集到的模型中各个层的输入和输出的统计数据如最小值、最大值等，将输入、输出和权重从32位浮点数类型量化到8位整型类型，然后向服务器端发送量化后的模型以及权重的量化参数。服务器端根据接收到的客户端的量化模型以及对应的量化参数对各模型执行反量化的操作，再进行模型的聚合。最后，服务器端对聚合后的模型执行量化操作，并将量化后的模型与量化参数分发给各个客户端。

## 算法详细设计

### 联邦学习环境中客户端算法的详细设计

伪代码

BEGIN:

While(notTerminate-Condation):#当不满足终止条件时，循环

Ifhas\_pruning:　#如果有剪枝策略

Prune()　#对模型进行剪枝

Train()　#训练剪枝后的模型

Valid()　#对训练好的模型进行验证

Ifshould\_aggregate:　　　　　#如果当前轮次需要聚合

SendModel()#给服务器端发送模型

RecvModel()#接收服务器端聚合的模型

END#结束算法

### 联邦学习环境中服务器端算法的详细设计

伪代码

BEGIN:

While(notTerminate-Condation):#当不满足终止条件时，循环

RecvModels()#接收客户端传来的模型

AggregateModels()#对客户端传来的模型进行聚合

SendModel()#分发聚合后的模型

CheckConvergence()#进行收敛性的验证

Ifconverged:　　　　　#如果模型已经收敛

break#退出循环

END#结束算法

### 剪枝算法的详细设计

BEGIN:

sensity=GetSensity()#获取配置好的敏感度

tensor=GetTensor()#获取需要剪枝的张量

threshold=tensor.stddev\*sensity#根据标准差和敏感度计算阈值

mask=CreateMask(threshold)#根据剪枝阈值创建掩膜矩阵

ApplyMask(tensor,mask)#执行掩膜操作，即按元素相乘

END#结束算法

### 量化算法的详细设计

主要函数的伪代码：

BEGIN:

config=GetQuantConfig()#获取量化的配置

model=GetModel()#获取需要量化的模型

ReplaceParamLayer(model,config)#替换模型中的参数层

QuantizeForward(model)#使用量化模型计算输出

END#结束算法

ReplaceParamLayer函数的伪代码：

BEGIN

CalculateScales()#　计算量化的相关参数

QuantWeights()#　对权重进行量化

RegisterSacles()#　将量化参数注册到模块的缓存中

END

### 4.5训练后量化算法的描述、伪代码以及流程框图

1. 算法描述
   1. 创建量化模型
      1. **输入：原模型，哑输入**
      2. **输出：量化后的模型**
      3. 根据需要量化的模型以及dummy\_input获取模型的总结图，得到邻接映射。
      4. 对每一层的统计数据进行处理：如果当前层的下一层为ReLU层，则对该层的输出统计数据进行钳制，即将输出统计数据中的最小值设置为0。
      5. 将nn.Conv2d、nn.Linear等层替换为参数层，参数层的替换过程中根据统计数据计算输入、输出数据的量化参数（scale和zero\_point），根据浮点类型的权重向量计算权重的量化参数，并对权重进行量化。
      6. 将ReLU层替换为恒等层。
      7. 将其余的层替换为伪量化层，对于伪量化层同样计算输出、输出数据的量化参数。
   2. 量化前向传播
      1. **输入：浮点类型的输入、量化后的模型**
      2. **输出：浮点类型的模型输出**
      3. 对输入进行量化，得到低精度的输入
      4. 如果首次前向传播，则计算累加器的量化参数accum\_scale=input\_scale\*w\_scale，否则跳过该步。
      5. 使用计算好的累加器的量化参数对浮点类型的偏置进行量化。
      6. 计算累加器的值accum=wrapped\_model(input\_q)
      7. 如果下一层是relu层，则对累加器的值进行relu操作
      8. 对accum进行解量化，得到浮点类型的模型输出
2. 伪代码
3. 流程框图

### 4.6量化感知训练算法的描述、伪代码以及流程框图

1. 算法描述
   1. 创建量化感知模型
      1. 定义线性量化参数的函数
         1. **输入：高精度类型的参数，量化的元数据**
         2. **输出：量化+解量化后的输出**
         3. 根据给定的模型参数和量化元数据计算量化参数。
         4. 应用量化直通估计量，得到量化处理后的参数。
      2. 定义激活单元的替换函数：引入伪量化层FakeQuantWrapper进行替换。
         1. 训练阶段，使用EMA统计流经该层的数据
         2. 根据统计数据对当前输入进行钳制
         3. 应用量化直通估计量，得到量化处理后的输出。
   2. 模型准备
      1. 预处理容器：将ReLU层替换为伪量化层
      2. 备份浮点版本的参数，创建量化后参数的缓存
      3. 为模型设置输入的伪量化器，当前向传播时，模型输入需要先经过伪量化器的处理。
   3. 使用线性量化参数函数对每个子模块的浮点参数进行量化。
2. 伪代码
3. 流程框图

# 算法的实现与评测

## 算法的实现

### 1.1算法的实现与部署

后量化部分——Prepareamodelforquantization

### 1.2算法实现的结果展示

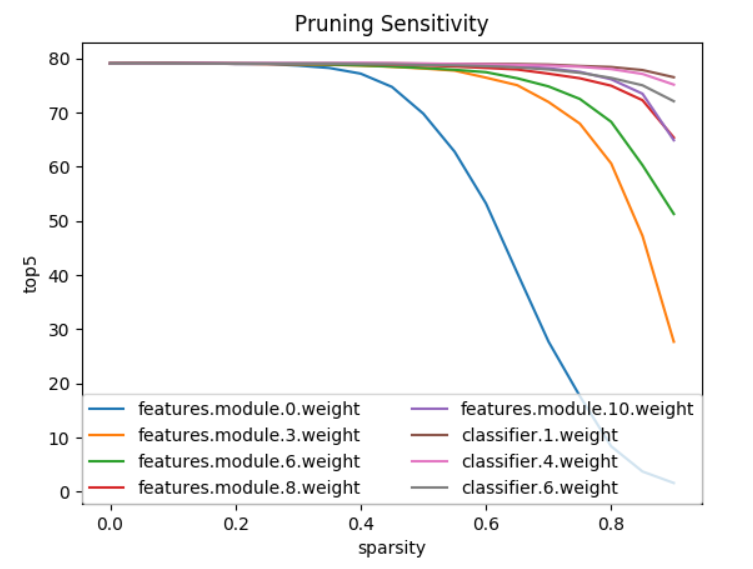
## 2.算法的评测

### 2.1实验环境

AlexNet网络层的介绍

Dropout值的调整

### 2.2 敏感度分析



图表1对AlexNet执行敏感度分析

一个层越敏感，则该层的s值就越小。因此，敏感度分析是一种经验性的方法，其结果给出了s的一组初始值，更优的值还需要通过实验确定。

### 2.3剪枝策略调度

一般的优化器会在训练的初始阶段设置较高的学习率，并在之后的训练过程中对学习率进行衰减。在在对具有很高学习率的网络进行剪枝时，此时网络参数还没有收敛到一个比较好的解，此时进行剪枝，可能会造成比较大的精度损失。因此，一般根据学习率的调度算法来制定剪枝策略。如当训练模型趋于收敛后，对模型进行剪枝，之后再进行一定轮次的训练，使得模型的精度得以恢复。

【对比实验】

1. 开始训练即对模型进行剪枝
2. 当训练50/100个epoch后进行剪枝（具体epoch数根据模型精度变化确定）

### 2.4 Dropout值的调整【SensitivityPruner中】

由于网络中参数量的减小，则dropout的比率也应该进行调整以适应模型容量的变化。

### 2.5整个项目中的调度策略

《PruningFiltersforEfficientConvNets》中采用一次性的剪枝和重训练策略来节省重训练时间。

### 2.2实验数据

### 2.3评测结果分析

# 六、结论

剪枝阈值是剪枝网络的超参数，经验性设置，一般进行敏感度分析来为每个网络层独立地设置剪枝阈值。或者，为了减少超参数数量，所有层使用相同的剪枝阈值，但精度可能不高。

# 参考文献