走向公平和隐私保护联盟

深层模型

Lingjuan Lyu，IEEE成员，Jiangshan Yu，Karthik Nandakumar，IEEE高级成员，Li Tong Li，∗

Ma Xingjun，Jiong Jin，成员，IEEE、Han Yu和Kee Siong Ng∗

**摘要**-目前独立的深度学习框架往往导致过度适应和低效用。这个问题可以通过部署一个中央服务器来训练来自各方的联合数据的全局模型的集中式框架，或者利用参数服务器聚合本地模型更新的分布式框架来解决。基于服务器的解决方案容易出现单点故障的问题。在这方面，诸如联合学习（FL）这样的协作学习框架更加健壮。现有的联合学习框架忽略了参与的一个重要方面：公平。所有缔约方都得到了相同的最终模式，而不考虑它们的贡献。为了解决这些问题，我们提出了一个分散公平和隐私保护的深度学习（FPPDL）框架，将公平性纳入联邦深度学习模型。特别地，我们设计了一个本地可信度互评机制来保证公平性，并设计了一个三层洋葱式加密方案来保证准确性和隐私性。与现有的FL范式不同，在FPPDL下，每个参与者都会收到不同版本的FL模型，其表现与其贡献相称。在基准数据集上的实验表明，FPPDL平衡了公平性、隐私性和准确性。它使联合学习生态系统能够发现并孤立低贡献方，从而促进负责任的参与。

**索引术语**-联合学习，隐私保护，深度学习，公平，加密。

F

# 1         简介

|  |
| --- |
| D |

EEP学习已经成为解决图像分类和语音识别等具有挑战性的现实问题的重要技术。经验证据表明，深度学习模型可以从大规模的数据集中受益匪浅[1]。然而，由于数据收集和注释[2]，[3]所需的大量时间和精力，大型数据集并不总是适用于新的域。此外，在大规模数据集上训练复杂的深部网络在计算上是昂贵的，并且在实践中可能不适用于一方。因此，在一组当事人之间以协作的方式进行深度学习的需求很高。

这一趋势的动机是，一方拥有的数据可能非常同质，导致过度拟合，当模型应用于以前看不到的数据时，会对准确性产生负面影响，即泛化能力差。利用多方数据进行深度培训

|  |
| --- |
| •*五十、 吕在新加坡国立大学计算机科学系工作。电子邮件：lyulj@comp.nus.edu.sg。*  •*J、 余教授在澳大利亚克莱顿莫纳什大学信息技术学院工作。E-邮箱：jiangshan.yu@monash.edu。*  •*K、 Nandakumar在IBM新加坡实验室工作，018983。电子邮件：nkarthik@sg.ibm.com。*  •*Y、 李和X.Ma是澳大利亚墨尔本大学计算机与信息系统学院的一员，邮编3010。电子邮件：yitongl4@student.unimelb.edu.au；马兴军@unimelb.edu.au大学.*  •*J、 Jin现供职于澳大利亚墨尔本斯文伯恩科技大学软件与电气工程学院。电子邮件：jiongjin@swin.edu.au。*  •*韩宇是新加坡南洋理工大学计算机科学与工程学院的学生。电子邮件：韩愈@南洋理工大学.*  •*Kee Siong Ng是澳大利亚国立大学软件创新研究所的成员。电子邮件：保持@澳大利亚教育局.*  ∗*通讯作者。* |

模型可以帮助缓解这个问题。然而，由于隐私问题，协作模型培训可能不可行。联合学习（FL）将隐私保护技术融入到协作模型训练中，为这一挑战提供了一个潜在的解决方案[4]。

在当前的联合学习范式[5]中，所有参与者在协作模型培训结束时接受相同的联邦模型，而不管他们的贡献如何。这使得这种范式容易受到搭便车参与者的攻击。例如，几家银行可能会合作建立模型来预测中小企业的信用度。然而，拥有更多数据的大型银行可能不愿意根据高质量的本地数据训练本地模型，因为担心小型银行从共享FL模型中获益并侵蚀其市场份额[4]。如果没有隐私的保障和协作公平的承诺，拥有高质量和大数据集的参与者可能会被阻止加入联合学习，从而对健康的外语学习生态系统的形成产生负面影响。现有的公平性研究主要集中在保护敏感属性或减少参与者之间预测分布的方差[6]，[7]。公平对待联合学习参与者的问题仍然没有解决[4]。

在这篇论文中，我们讨论了公平对待外语参与者的问题，基于他们对构建健康外语生态系统的贡献。我们将所提出的框架称为分散公平和隐私保护深度学习（FPPDL）框架。与现有的工作（如[8]）使用货币奖励来激励良好行为不同，我们提出的解决方案从根本上改变了当前的外语教学模式，因此参与者最终可能不会得到相同的外语学习模式。取而代之的是，他们每个人都将得到一个最终的FL模型，其表现反映了他们对联合会的个人贡献。FPPDL不要求参与者相互信任或任何第三方。它通过区块链技术记录所有操作，包括上传和下载不同的私有人工样本和加密模型更新。FPPDL通过对本地可信度的相互评估，考虑了双方在初始基准测试和隐私保护协作深度学习过程中的相对贡献，实现了协作公平。为了保护隐私，我们提出了一个三层的onionstyle加密方案，以保证准确性和隐私性，而不是以牺牲实用性为代价来利用差异隐私。

据我们所知，本文首次通过调整分配给每个参与者的FL模型的性能水平来实现联合学习中的协作公平。基于两个基准数据集在三个实际环境下的大量实验表明，FPPDL实现了较高的公平性，提供了与现有集中式和分布式深度学习框架相当的精度，并且优于独立的深度学习。

在威胁模型方面，FPPDL采用了一个诚实但奇怪的设置：假设每一方在推断其他方的敏感信息时都很好奇；然而，在操作中，假设它是诚实的。这一设置是合理的，因为参与协作系统的各方的主要动机是获得比没有任何协作的独立模型更好的本地模型。此外，在我们的方案中，各方被视为金融机构或生物医学机构等依法承担责任的组织。然而，在第7节中，我们还讨论了我们的本地可信度互评机制如何帮助防止内部攻击者的某些恶意行为，以及如何抵御外部攻击者。

本文的其余部分安排如下。第二部分回顾了现有的深度学习框架，以及关于隐私保护的协作深度学习和联合学习中的公平性的相关文献，这些都是我们要解决的主要问题。第5节介绍了拟议的FPPDL框架的技术细节。第6节从准确性和公平性方面评估了FPPDL在不同设置下不同SGD框架下的性能，然后在第7节中进行了讨论。第八部分总结全文，并指出未来可能的研究方向。

# 2         相关工作

在这一部分中，我们回顾了有关深度学习框架、隐私保护和联合学习公平性的相关文献，以使我们的研究与现有研究相关联。

## 2.1       深度学习框架概述

一般来说，深度学习框架可以分为以下几类：独立框架；基于服务器的框架，包括集中式框架和分布式框架；分散式框架。特别是在分布式框架和分散框架下，各方都参与了全球或共识模式的改进过程。因此，我们将其称为协作式深度学习框架。表1提供了不同深度学习框架之间的比较。

***独立框架***：缔约方在没有任何合作的情况下，根据其本地培训数据单独培训独立模型（图1（a））。然而，独立的模型可能无法概括为看不见的数据。

***集中框架***：参与者将他们的数据集中到一个集中的服务器中，以训练一个全局模型（图1（b））。这种集中化的框架非常有效，但是它侵犯了数据隐私，因为所有参与者的数据都暴露在服务器上。

***分布式框架***：Dean等人。[9] 首先介绍了分布式深度学习的概念，即各方通过与参数服务器共享本地模型更新来协作训练模型。分布式学习在[5]、[10]、[11]中得到了广泛的研究。

需要注意的是，集中式框架和分布式框架都需要一个中央服务器来协调培训过程，这使得它们容易受到以下问题的影响：（1）参与方策略：出于隐私考虑，双方可能不想将控制权交给不可信的服务器；（2） 单点故障：如果中央服务器发生故障或关闭进行维护，整个网络将停止工作。

***分散框架***：在基于中央服务器的框架中，上述问题可以通过一个分散的框架[12]、[13]、[14]、[15]、[16]、[17]来解决，该框架使各方之间的计算并行化（图1（d））。特别是Kuo等人。[12] 提出了一种保护隐私的机器学习与机器学习相结合的模型保护框架。[15] ，[16]调查了区块链上的隐私保护深度学习。[17] 研究了基于区块链的隐私保护学习中负载共享的公平性问题，它不同于我们工作中的协作公平性问题。具体地说，它们的分散式架构是在一个不太安全的环境下开发的，在这种环境下，一个站点可以访问所有其他站点的模型。[13] ，[14]利用差异隐私在区块链上保护隐私的机器学习。然而，[14]指出，在[13]中提出的算法不能正确地保证隐私保护的性质，因为它们没有考虑重复加性噪声机制的合成。

综上所述，现有的协作框架（分布式或分散式）关注的是如何学习比单个独立模型更精确的全局共识模型。实际上，一些缔约方可能比其他缔约方作出更多贡献，而另一些缔约方则几乎没有贡献，甚至是消极的。究其原因，是不同主体拥有的数据质量不同，在数据采集和存储过程中可能存在不可预测的随机误差。另一方面，缔约方可选择仅将其有限部分数据用于合作模式培训。

## 2.2       隐私保护协作学习

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **第1部分第2部分**DD  D1宽1 D2宽2             **第3部分第4部分**DD  D3 w3 D4 w4  图1：（a）：独立框架。（b） 集中式框架。（c） ：分布式框架。（d） ：分散框架。  表1：比较不同的深度学习框架。   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 框架 | 独立 | 集中式[18]，[19] | 分布式  [10] ，[11]，[20] | [5] 你说， | 分散[12]，  [13] ，[14]，[15]，[16]，  [17] | 去中心化（我们的FPPDL） | | 建筑 | 图1（a） | 图1（b） | 图1（c） | | 图1（d） | 图1（d） | | 全局模型 | 不 | 是的 | 是的 | | 取决于 | 取决于 | | 本地模型 | 是的 | 不 | 是的 | | 是的 | 是的 | | 合作公平 | 不适用 | 不适用 | 不 | | 不 | 是的 | | 质量控制 | 不适用 | 不 | 不 | | 不 | 是的 | |

正如Shokri等人指出的那样。[11] ，集中式深度学习通常会带来许多隐私问题。具体来说，所有敏感的培训数据都会透露给第三方；数据所有者对学习目标没有控制权；学习的模型不能直接提供给数据所有者。为了减轻这些隐私风险，吉拉德·巴赫拉赫等人。[18] 开发密码网来运行同态加密数据的深度学习。然而，密码网络假设神经网络模型是预先训练好的，因此其系统主要用于向用户提供加密输出。相比之下，SecureML[20]通过安全多方计算（SMC）进行隐私保护学习，其中数据所有者需要在初始设置阶段处理、加密和/或秘密共享其数据。SecureML允许数据所有者根据他们的联合数据训练各种模型，而不必透露结果之外的任何信息。然而，这种方法产生了很高的计算和通信成本[21]，[22]。

最相关的工作是分布式选择性随机梯度下降（DSSGD）[11]。为了保护隐私，各方不是明确地共享训练数据，而是根据本地训练数据计算并共享（与PS）其本地模型梯度，同时通过从PS下载最新参数来更新其本地模型。为了进一步减少共享模型更新带来的隐私泄露，每一方都会对本地模型更新添加噪音，以确保每个参数的差异隐私。然而，他们的系统需要一个中央参数服务器来协调训练过程。因此，它面临着基于中心服务器的框架中常见的问题。

集中参数服务器的缺点可以总结如下：

1） 隐私泄露：如[23]所示，即使只有一小部分局部模型更新被发布到PS，局部数据信息也可能泄露给一个诚实但好奇的PS，特别是对于只有一个神经元的局部神经网络，PS可以以不可忽略的概率推断出参与者的真实数据或标签。上述观察结果同样适用于具有交叉熵和平方误差代价函数的一般神经网络。即使对于一般的正则化神经网络，释放的局部梯度仍能揭示真实值。

2） 易受主动对手攻击：大多数分布式学习框架都假设所有参与者都是诚实的。实际上，如果一方被证明是恶意的，它可以通过欺骗随机样本来推断受害方的私人数据信息，从而破坏学习过程。

分布式深度学习的一个特例是联合学习[4]。在佛罗里达州，为了保护个人模型更新的隐私，Bonawitz等人。[24]提出了一种实用的安全聚合协议，该协议在诚实、好奇和活跃的对手环境下是安全的，即使任意选择的用户子集随时退出。特别地，安全多方计算（SMC）被用于以安全的方式从单个用户的设备计算模型参数更新的总和，这是以额外的计算和通信开销为代价的。另一种更有效的方法是使用差异隐私，使服务器能够将定制的噪声添加到加权平均用户更新中，以保证用户级别的隐私[25]。然而，默认的受信任的Google服务器有权清楚地看到所有用户的更新，聚合单个更新并为聚合添加噪声。因此，当服务器不是受信任方时，它们的方法不是首选方法。我们还注意到，当服务器不可信时，加权聚合变得不现实，因为服务器可能不知道用于权重计算的各方的数据大小。相反，提议的FPPDL允许每一方基于本地可信度和共享级别集成其他方的更新。

## 2.3       联合学习的公平性

现有的促进联合学习参与者之间合作公平的方法是基于激励机制的。一般来说，参与者应获得与其贡献相称的报酬。平等分配是平等利润分享的一个例子[26]。在这个方案下，在给定的一轮中可用的总收益在所有参与者中平均分配。根据个人利益分享计划[26]，每个参与者自己对集体的贡献（假设集体只包含）被用来确定他在总收益中的份额。*我我*

工会博弈[27]利润分享方案根据参与者对其前任组成的集体效用的边际贡献（即每个参与者的边际贡献按照加入联合会的相同顺序计算），确定参与者在总收益中的份额。公允价值博弈方案[27]是一种基于边际损失的方案。在这个方案中，参与者在总收益中的份额由参与者离开联盟的顺序决定。Shapley game利润分享方案[27]也是一种基于边际贡献的方案。与工会博弈不同，Shapley博弈旨在消除参与者按不同顺序加入集体的影响，以便更公平地估计他们对集体的边际贡献。因此，它平均每个参与者相对于其他参与者加入集体的所有不同排列的边际贡献。这种方法计算成本很高。

对于基于梯度的联合学习方法，梯度信息可以看作是一种数据类型。然而，在这些情况下，基于输出协议的奖励很难应用，因为相互信息需要多任务设置，而这种情况通常不存在。因此，在这三类方案中，模型改进是为联合学习设计奖励的最相关的方法。目前有两种新兴的联合学习激励方案，主要集中在模型改进上。

[28]中提出了一种为模型更新带来的边际改进付费的方案。这些改进的总和可能导致对贡献的高估。因此，提议的方法还包括一个修正高估问题的模型。该方案确保付款与模型质量改进成比例，这意味着实现目标模型质量水平的预算是可预测的。它还可以确保提前提交模型更新的数据所有者获得更高的奖励。这促使他们甚至在联邦模型培训过程的早期阶段就参与进来。

除了参与者的贡献，[8]还提出了一种基于联合目标优化的方法，将成本和等待时间考虑在内，以便在向FL参与者分配收益时实现额外的公平性。与上述方法不同，本文提出的FPPDL框架没有利用货币收益来实现公平对待FL参与者。相反，它为每个人分配了不同版本的FL模型，其性能与他的贡献相称。这代表了现有联合学习的另一种范式，即所有参与者都会收到相同的最终外语模型。

# 3         准备工作

在本节中，我们将介绍构成所提出的FPPDL框架的关键技术，包括差分隐私、同态加密和区块链。

## 3.1       差别隐私

差分隐私1[29]通过以（i）计算效率，（ii）不允许攻击者恢复原始数据，以及（iii）不会严重影响实用性的方式干扰数据来权衡隐私性和准确性。

**定义1。***随机机制*MD→研发*:带域和范围**-对于所有两个相邻的输入，如果存在差异隐私D、 D*0 ∈D⊆R*在一个记录和任何可测量的输出子集上都是不同的S它认为*

公共关系

*此外*M=0*如果δ.*

差别隐私的形式化定义有两个参数：隐私预算衡量隐私泄露*δ*限制隐私丢失超过的概率。的值在算法重复访问私有数据时累积[30]。

## 3.2       同态加密

同态加密是一种广泛用于以安全方式导出聚合的加密形式。现有的同态加密技术包括完全同态加密、部分同态加密和部分同态加密。完全同态加密可以支持密文的任意计算，但效率较低[31]。另一方面，有些同态加密和部分同态加密只支持有限数量的操作[32]。

然而，所有这些技术通常导致密文比明文更长，从而产生额外的通信成本。为了解决这一问题，我们从流密码[33]中得到启发，开发了高效的同态密文压缩算法，该算法还允许在不同密钥流下加密的密文上进行加法同态运算。更多详情见第5.2.1节。

## 3.3       块链

区块链是一个分散的（即点对点、非中介）系统，由系统中的所有参与者维护。区块链有两种类型，即无许可区块链和许可区块链。使用无许可的区块链，如比特币[34]，参与者可以随时加入和离开，参与者的数量不是预先定义的，也不是固定的。对于许可的区块链（又称联盟区块链），如IBM的Hyperledger结构，参与者需要系统的许可才能加入或离开。参与者的集合通常是预先定义的[35]。

对于参与者相对稳定的应用程序，允许的区块链是首选。它可以作为一个分布式的密钥-值存储，其中需要一个容错（又称拜占庭协议）方案来达成对全局状态的共识。区块链以其透明度、责任性和稳健性而闻名，数据和所有操作都以附件方式记录在区块链上，所有参与者都可以访问。直观地说，联邦深度学习的增量特性使其适合于利用区块链。然而，需要开发一种将区块链与隐私保护深度学习相结合的合理方法。

# 4         FPPDL框架

本节描述了我们提出的去中心化公平和隐私保护深度学习（FPPDL）框架的设计，并研究了区块链作为FPPDL的分散架构。表2列出了本文中使用的符号及其含义，以便于阅读。

表2：符号列表。



|  |  |
| --- | --- |
| 符号 | 意义 |
| *D我*, *米我* | 地方培训数据与地方党建模式*我* |
| *SDi公司* | *µ*党随机抽取的DPGAN样本*我* |
| *p我*, *d我* | 点和梯度下载党的预算*我* |
|  | 党的地方公信力和更新的地方公信力*j我* |
| *用户界面* | 缔约方发布的DPGAN样品数量*我* |
| *dj我* | 释放到参与方的有意义坡度数*j我* |
| *λj* | 共享度*j* |
| *saccj、accj* | 独立和最终模型精度*j* |
| ∆*威斯康星州* | 党的梯度向量*j* |
| ∆*w***˜***吉* | 通过用0填充剩余梯度来屏蔽与参与方共享的参与方的梯度向量*j我* |
| *威斯康星州* | 上一轮参与方参数*我* |
| *w我*0 | 本轮更新了参与方参数*我* |
| *n* | 参与方数量 |
| *c第* | 可信度阈值下限 |
| *C* | 三分之二的缔约方同意的具有上述当地信誉的可信方*cth公司* |
| *乔丹* | 多数标签和参与方的预测标签之间的匹配数*j* |
| （sk00）*我，pk我* | 分别用于签名和验证的方密钥对*我* |
| *基* | 方的密钥流用于第一层三层洋葱式加密*我* |
| *fsk公司* | 新鲜对称加密密钥用于第二层三层洋葱式加密 |
| （滑雪、pki） | 用于解密和加密的第三层三层洋葱式加密的密钥对*我* |
| *Enc公司* | 同态加密 |
| *森克* | 对称密钥加密 |
| *Aenc公司* | 公钥加密 |
| *E* | 每轮本地训练时数 |
| *B、 lr公司* | 本地批大小、本地学习率 |



## 4.1       设计目标

### 4.1.1        隐私保护

在FPPDL中，我们假设双方不信任对方或任何第三方。因此，在没有隐私保护承诺的情况下，当培训一个联合模型时，各方可能不愿意共享他们的信息。在FPPDL中，各方不共享原始数据或模型参数，而是利用差异私有GAN（DPGAN）发布差异私有本地样本，以便在初始基准测试阶段进行相互评估。然后，他们使用提出的三层onionstyle加密方案对共享梯度进行加密，以在协作深度模型训练期间保护隐私。

### 4.1.2        公平

由于我们在这里的重点是根据参与者的贡献向参与者分发最终FL模型的不同变体，因此与我们的目的最相关的公平概念是通过意识实现公平。在这个概念下，在为特定任务定义的相似性度量方面相似的个体应该得到相似的结果[36]。高贡献的党应该得到比低贡献党更多的奖励。此外，我们还明确指出，低贡献方并非恶意的，即他们诚实地遵守协议，以从其他方的数据中获益为目标，但贡献很少，甚至几乎没有贡献或消极贡献。在协作模型培训场景下，我们将协作公平定义为：

**定义2。***合作公平。在协作学习系统中，高贡献方理应获得比低贡献方更好的本地模型。特别地，在IID环境下，公平性可以通过各方贡献与其各自最终模型精度之间的相关系数来量化。*

考虑到这两个目标，我们设计了一个本地可信度互评机制，以加强FPPDL中的公平性，即参与者使用他们的“积分”以“赚取并支付”的方式交换信息。每个参与者的本地可信度和点数通过初始基准测试阶段初始化，并通过保留隐私的协作深层模型培训进行更新。

其基本思想是参与者可以通过向其他参与者提供他们的信息来获得积分。然后，他们可以使用所获得的积分与其他参与者交换信息。因此，鼓励参与者上传更多的样本或梯度来获得更多的分数（只要在共享级别的限制之内），并使用这些点从其他人那里下载更多的梯度。所有交易都被记录为区块链中不可变的交易，从而提供透明度和可审计性。尤其是，FPPDL确保下载和上传过程的公平性，如下所示：

•：由于一方可能对不同的当事方作出不同的贡献，因此该方的信誉可能与不同方的观点不同。因此，每一方记录一个私有的本地可信列表，该列表按可信值的降序排序。在政党的可信度列表中，政党的可信度越高，政党就越有可能从政党下载梯度，因此，政党对政党的奖励就越多。**下载***我j我我jj我*

•：一旦一方收到本地渐变的下载请求，它就可以根据请求方的下载请求和自己的共享级别来确定要发回多少有意义的渐变。**上传**

## 4.2       基于区块链的架构

为了开发FPPDL的去中心化架构，我们使用区块链2.0将隐私保护深度学习算法整合到私有区块链中，该区块链仅对参与方可用。与当前基于服务器的架构相比，FPPDL继承了区块链的peerto-peer架构，允许各方保持模块化，同时与其他方进行互操作。此外，每一方都保持对自己数据的完全控制，而不是将控制权拱手让给中央服务器。此外，区块链提供了自动协调

各方的加入和离开，进一步促进了联盟的独立性和模块化。区块链，没有单点故障，也增强了稳健性。在这里，我们为FPPDL设计了两种类型的区块链，即init区块和作业区块。

**初始化块**初始化各方培训数据有用性的基准测试，作为一组初始事务。init事务包含事务创建者获得的初始点、它提供的DPGAN示例以及将用于验证未来事务的公钥。区块链的genesis块（即第一块）是init块，它根据参与者的相对贡献包含初始点和局部可信性值，如算法1所述。如果任何参与方在以后的更新过程中加入或添加新数据，将创建一个新的init块并将其添加到现有的

块链。

**操作块**包含一组定义上载操作和/或下载操作的事务。所有上传和下载事务都由其创建者使用与init事务中记录的公钥相关联的私钥签名。上载操作承诺数据所有者已将本地模型渐变上载到发送下载请求的一方。下载操作声明一个参与者被提交一个命令，请求其他参与者进行一些本地模型更新。收到下载交易后，区块链矿工验证其签名，检查请求者是否有足够的平衡点来下载请求的梯度数，并将验证的交易记录在操作块中。一旦下载交易记录在区块链中，请求的本地模型梯度将被加密并由所有者上传到可公开访问的存储器中，并使用下载交易中定义的接收者公钥重新加密。

本地模型梯度的隐私通过三层洋葱式加密方案得到保护（见第5.2节）。第一层通过我们提出的基于对称密钥的同态加密算法（算法3）对局部模型梯度进行加密，该算法允许各方在不暴露单个梯度的情况下学习所接收梯度的集合，即当事方的遗忘。第二层和第三层提供了一个标准的混合加密过程：第二层使用新生成的对称密钥重新加密第一层密文，第三层使用请求方的公钥进行加密。通过这种方式，我们可以最小化基于非对称密钥的加密所需的计算成本。上传的加密本地模型梯度（例如，密文的散列值，如图3所示）的承诺将包括在上传事务中。*fsk公司fsk公司我公钥基础设施*

在我们的私有区块链中，只有付费的请求者才能读取明文。其他人可以验证此事务是否已发生，但无法读取明文。当请求者责怪数据上传者时，数据上传者会将明文显示为证据。在这种情况下，申请人将被迫支付罚款，在提交索赔时，如果它被证明是不诚实的索赔。一旦上传交易记录在区块链中，点将自动从请求者转移到上传者。图2和图3分别示出了由区块链存储的初始化和下载事务的示例。对于我们的应用场景，我们期望有一组相对稳定且规模较小的参与者，例如承担法律责任的金融机构，它们属于涉及业务参与者的横向联合学习（HFL）的保护伞[37]。这允许我们采用许可的区块链。

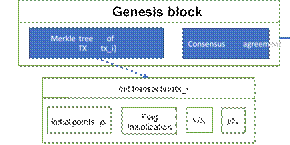


图2：成因块体构造示例。它主要包含两个关键组成部分，一个是以Merkle树的叶子组织起来的一组初始事务；另一个是参与者通过底层共识协议（如PBFT或PoS）达成的共识协议，这是特定于已部署区块链的。init事务中的是参与方的签名验证密钥。*pk键我*0 *我*

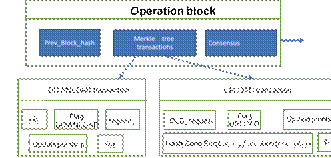


图3：操作块的示例结构。它主要包含三个关键组件，即前一个块的hash值Prev Block hash，一组Merkle树组织的上传/下载事务，以及该块的一致同意。特别地，Prev块散列将当前块链接到前一块，并且上载事务中的请求充当对关联的下载事务的引用。在下载事务中，公钥将在我们的三层onionstyle加密方案的最后一层使用，是该事务的唯一请求ID，将通过DLD请求在相应的上载事务中引用，并且是该事务的签名。，分别指同态加密、对称密钥加密和公钥加密。*公钥基础设施请求西格Enc公司森克Aenc公司*

# 5         FPPDL的实现

本节详细介绍了FPPDL的两阶段实现，以实现公平性和隐私性。其中包括如何算法1初始基准测试

**输入：参与方数量，C=1。，n***n*{}

**输出：当地信誉和各方得分**1： aprior模型预训练：各方基于其本地训练数据训练独立模型和本地DPGAN。2： 共享级初始化：初始化时，party随机选择本地DPGAN生成的人工样本发布给任何一方，共享级别自主确定为=ui/| Di |，其中| Di |为party的本地培训数据大小。*我惯性矩我用户界面jλi我*

3： 局部可信度初始化：Party通过其局部模型对接收到的人工样本进行标记，然后将预测的标签返回给Party。同时，该党还为自己的DPGAN样品贴上标签。然后，党对所有预测的标签进行多数票表决，从而初始化政党的地方公信力*j乔丹我我惯性矩我*，其中是多数标签和party的预测标签之间的匹配数，是party发布的DPGAN样本数。详细说明见第5.1.1节。*乔丹j用户界面我*

4： 地方公信力规范化：

**如果那样的话***c吉<c级第*

党报党为低贡献党*我j*

**结束if**

5： 可信方集：如果大多数参与方报告参与方为低贡献方，区块链将该方从可信方集中移除，所有各方再次运行第4步。6： 下载渐变的点初始化：=λi | |\*（n−1）。*jjC圆周率威斯康星州*



初始化本地可信度值，通过初始基准测试共享级别和点数，以及如何在隐私保护协作深度学习阶段更新本地可信度值和点数，然后量化公平性。两阶段的实现如图4所示。

## 5.1       初始基准

提出的初始对标算法的目的是在协作模型训练开始之前，不需要查看原始数据，而是通过相互评估来评估每个参与者的本地训练数据质量。该算法的工作原理是：每个参与者根据其局部训练数据训练一个DPGAN，生成人工样本。然而，这些生成的样本不会揭示真正敏感的例子，以及数据的真实分布，而只是在DPGAN中使用的适度隐私预算内的一些隐式密度估计。每个参与者根据各自的共享级别发布单独生成的人工样本，而不发布标签。所有其他参与者使用他们预先训练的独立模型对接收到的人工样本进行预测，并将预测的标签发送给生成这些样本的一方。

共享人工DPGAN样品的目的有两个：

1） 在协作学习开始之前获取有关单个模型的先验信息。如果参与者没有合理数量的培训数据来生成一个像样的模型，那么在初始评估阶段，它的表现将很差。因此，其他参与者在与它共享梯度时会非常谨慎。

2） 获得其他参与者数据分布的粗略估计。只有当两个参与者的数据分布不同，但有一定程度的重叠时，他们才能互惠互利。假设两个参与者A和B发布了几乎相同的人工样本，这意味着他们的训练数据分布几乎相同。在这种情况下，B的更新不太可能提高模型A的精度，反之亦然。因此，在随后的通信回合中，A和B应避免从对方下载更新。其他参与者可以选择从A或B下载更新，但不能同时从A或B下载更新。另一方面，假设参与者A和B具有完全不同的数据分布，B的更新不可能提高模型A的精度，反之亦然。因此，在随后的回合中，A和B也应该避免从对方下载更新。此外，假设A的数据分布与所有其他参与者的数据分布不同，所有这些参与者都应尽量避免A。这会自动处理这样的场景：一个诚实的参与者发布一些渐变，而所有其他诚实参与者将非常低的可信度分配给发布者。在这种情况下，发布者的数据分发与其他参与者的数据分发完全不同。因此，降低出版商的信誉是合理的。

接着详细介绍了算法1中初始基准测试的具体步骤，包括：局部可信度初始化、共享级别和点初始化。

### 5.1.1        局部可信度初始化

对于局部可信度初始化，每一方将所有组合标签的多数票与某一方的预测标签进行比较，以评估该方的效果。它依赖于这样一个事实，即所有组合标签的多数投票反映了大多数政党的结果，而政党的预测标签只反映了政党的结果。*jj*

For example, in the case of party initializing local credibility list for other parties, party broadcasts its DPGAN samples to other parties, who label these samples using their pre-trained standalone models, and send the corresponding predicted labels back to party . Meanwhile, party also labels its own artificial samples using its pre-trained standalone model, then combines all parties&apos; predicted labels as a label matrix with total columns, where each column corresponds to one party&apos;s predicted labels. Party *i i ii n i* then initializes the local credibility of party, where *mj* is the number of matches between the majority labels and party &apos;s predicted labels, and is the number of DPGAN samples released by party . Afterwards, party normalizes within [0,1].*jui ii cji*

|  |
| --- |
| st |

If the majority of parties report that the local credibility of one party is lower than the threshold , implying a potentially low-contribution party, it will be banned from the local credibility lists of all parties. Here, is mainly used to detect and isolate the low-contribution party, and it should be agreed by the majority of parties. However, it should not be too small or too large as fairness and accuracy may be affected. If it is too small, it might allow low-contribution party to into the collaborative learning system without being detected and isolated. If it is too large, it might ban most participants from the system. In the following update process, party is more likely to download gradients from more credible participants, while download less, even ignoring those published by less credible parties.*cthcth i*

### 5.1.2        Sharing Level and Points Initialization

Sharing level is denoted by the the upper bound of the number of samples or gradients one party can share with others. Based on the number of artificial samples that party publishes at the beginning, a suitable sharing level of party can be automatically estimated as = ui/|Di|, where is the local training data of party . Points are initialized as follows:*ui i i λi Di i*

*pi* = λi ∗ |*wi*|∗(n − 1) (1)

where is the sharing level of party (i.e., the higher, the more data one party would like to share), || is the number of model parameters, and is the number of parties. The points gained from initial benchmarking will be used to download gradients in the following collaborative learning process, and the number of gradients can downloaded depends on both the local credibility and sharing level of the party from which it is requesting.*λi i win i*

### 5.1.3        Differentially Private GAN (DPGAN)

During initial benchmarking, although each party only releases a small amount of unlabeled samples, it may still disclose privacy of local training data. The approach of generating samples under differential privacy with generative adversarial network (GAN) offers a solution to this problem. Under FPPDL, we train a Differentially Private GAN (DPGAN) by adding tailored noise to the gradients during DPGAN learning [38] at each party.

In the context of a GAN, the discriminator is the only component that accesses the private real data. Therefore, we only need to train the discriminator under differential privacy. The differential privacy guarantee of the entire GAN directly follows because the computations of the generator are simply post-processing from the discriminator. The main idea follows the post-processing property of differential privacy [29], as stated in Lemma 1.

To counter the stability and scalability issues of training DPGAN models, we apply multi-fold optimization strategies, including weight clustering, adaptive clipping and warm starting, which significantly improve both training stability and utility [38]. Unlike PATE [39], where privacy loss is proportional to the amount of data needed to be labeled in public test data, differentially private generator can generate infinite number of samples for the intended analysis, while rigorously guaranteeing -differential privacy of training data. Without loss of generality, we exemplify DPGAN in the context of the improved WGAN framework [40] and let each party generates a total of 1,000 artificial samples. As demonstrated in [38], DPGAN is able to synthesize both grey and RGB image with inception scores fairly close to the real data and samples generated by regular GANs without any privacy protection.

**Lemma 1.** *Let algorithm* A : R→ R be a randomized*n*

*algorithm that is (,δ)-differentially private. Let f* : R → R◦ A : R→ R0 *be an arbitrary randomized mapping. Then f n* 0 *is*

*)-differentially private.*

Meanwhile, it is well-known that larger amount of train-

|  |
| --- |
| **Algorithm 2** Privacy-Preserving Collaborative Deep Learning    **Input: , , , , , , ,** *Ccjipipjdiλjwi*∆*wj*  **Output: updated points , , parameters , and local credibility***p*0*jp*0*iwi*0  1: Trade gradients as per download requests, local credibility, and sharing level; Party points update: In each communication round, party aims to download total = pi gradients from all parties in , while party ∈ C \ i can at most provide × |∆| gradients, one point is consumed/rewarded for each download and upload. Each party updates local model parameters based on the gradients of party ∈ C \ i as follows: for*i di Cj λj wji j*  *i* = (c*i* ∗ *i j* ∆*wj* ), *pj* = p*j* + *i*, *pi* = p*i* − *i*  ∆= ∆, party first chooses meaningful gradients from ∆according to “largest values” criterion: sort gradients in ∆and choose of them, starting from the largest, then masks the remaininggradients with 0 as ∆.*wji wjj dji wji wji dji w***˜***ji*  **end for**  2: Three-layer onion-style encryption: Party follows Algorithm 3 to encrypt the masked gradients ∆with its keystream*j w***˜***ji*  , and re-encrypts the encrypted gradients with a fresh symmetric encryption key as (c,fsk),*c fsk Senc*  the symmetric encryption key of the second layer is encrypted in the third layer by the receiver party &apos;s public key as (fsk,pki). Finally, the two-layer encrypted gradients (c,fsk) and the encrypted fresh symmetric encryption key (fsk,pki) are sent to party ;*ipki AencSencAenci*  3: Parameter update: party uses the paired secret key to decrypt the received encrypted fresh symmetric encryption key as , then uses to decrypt the two-layer encrypted gradients as *i ski fskfsk* , finally decrypts the sum of all the received gradients using homomorphic property and updates local parameters by integrating all its plain gradients ∆as: wi= + ∆+ ∆, where wi is party &apos;s local parameters at previous communication round.*wi* 0 *wiwi wi* P*j*∈C\i*w***˜***jii*  4: Local credibility update: party randomly selects and releases artificial samples to any party for labelling, mutual evaluation is repeated by following Step 3 of Algorithm 1 to calculate local credibility of party at current communication round as. Party updates local credibility of party by integrating its historical credibility as: = 0.2∗cji +0.8∗cij0, where is the local credibility of party at previous communication round.*i ui j j i j cji*0 *cji j*  5: Local credibility normalization:  **ifthen**  party reports party as a low-contribution party*i j*  **end if**  6: Credible party set: If the majority of parties report party as low-contribution, Blockchain removes party from credible*j j* |

  
party set and all parties run Step 5 again.*C*

ing data causes less privacy loss, and allows for more iterations within a moderate privacy budget [30]. Due to the scarcity of training data of each party, data augmentation is exploited to expand local data size of each party to 100 times, which allows DPGAN to generate realistic samples within a moderate privacy budget. In particular, we augment original data with rotation range of 1 and width shift range and height shift range of 0.01.

In our study, we use moments accountant described in [30] to track the spent privacy over the course of training. Our DPGAN is able to generate realistic MNIST samples with and *δ* = 10−5, as shown in Fig. 5. Note that each party can individually train DPGAN and generate massive DPGAN samples offline without affecting collaboration.



Fig. 5: Generated DPGAN samples with using the augmented 60000 MNIST examples of one party who owns 600 original MNIST examples.

## 5.2       Privacy-Preserving Collaborative Deep Learning

Algorithm 2 summarizes the steps for the privacypreserving collaborative deep learning in each communication round, including how to update points as per upload/download, how to preserve privacy of individual model updates using three-layer onion-style encryption followed by parameter and local credibility update, and credible party set maintenance by the Blockchain system. In particular, the gradients download budget of party , i.e., , is closely related with how many points party has in each communication round. More concretely, should not exceed , otherwise, party will not have enough points to pay for the gradients provided by other parties. Moreover, can be dynamically determined based on the existing points in each communication round. For simplicity, we initialize = pi in each communication round, but how many gradients can be downloaded will be dependent on both the local credibility list of the requester and sharing levels of the requested parties, which can be referred to Section 4.1. In the following sections, we will focus on the most important details for parameter update, three-layer onion-style encryption, and local credibility update.*idipi i di pii di pi di*

### 5.2.1        Parameter Update with Homomorphic Encryption

Sharing gradients can prevent direct exposure of the local data, but may indirectly disclose local data information. To further prevent potential privacy leakage from sharing gradients and facilitate gradients aggregation during the collaborative learning process, we use additive homomorphic encryption such that each party can only decrypt the sum of all the received encrypted gradients. Specifically, Vernam cipher or one-time pad (OTP) has been mathematically

proved to be completely secure, which cannot be broken given enough ciphertext and time. Therefore, we use simple and provably secure OTP for additively homomorphic encryption that allows efficient aggregation of encrypted data [41], [42]. The main idea of forming the ciphertext is to combine the keystream with the plaintext digits. Meanwhile, rather than XOR operation typically found in stream ciphers, which is unsecured under the frequency analysis attacks, our encryption scheme uses modular addition (+), and is hence very efficient [41]. The security relies on two important features: (1) the keystream changes from one message to another; and (2) all the operations are performed modulo a large integer [41].*M*

The detailed procedure for homomorphic encryption is presented in Algorithm 3. In practice, if = max(xi), is derived as = 2dlog. All computations in the remainder of this paper are modulo unless otherwise stated. However, all the original floating-point values need to be mapped to the integer domain by using Scaling, Rounding, Unscaling (SRU) algorithm [42]. A pseudorandom keystream can be generated by a secure pseudo random function (PRF) by implementing a secure stream cipher, such as Trivium [43], keyed with each party&apos;s keystream and a unique message ID. For encryption purpose, the secret keys are pre-computed through a trusted setup, which can be performed by a trusted dealer or through a standard SMC protocol.*p M M* 2(p×n)e*M k ki*

For example, a trusted key managing authority can generate these keystreams in each communication round, but the generated keystreams cannot be used more than once. The trusted setup generates non-zero random shares of 0: ∈C= 0, such that each participant ∈ C obtains a keystream . Note that if the Blockchain removes party from the credible party set , a new credible party set should be constructed.P*iki i kij CC*

**Algorithm 3** Homomorphic Encryption Scheme

**Setup**

1: A trusted dealer randomly generates |C| keystreams: ∈ [0,M − 1], such that (mod )= 0, where is a large integer. 2: Party obtains keystream .*k*1*,...,k*|C| P*i*∈C*ki MM i ki*

**Enc(, )***mk*

1: Represent message as integer ∈ [0,M − 1]. 2: Let be a randomly generated keystream, where ∈ [0,M − 1].*m m k k*

3: Compute = Enc(m,k) = m + k.*c*

**Dec(, )***ck*

1: (c,k) = c − k.*Dec*

**AggrDec()***ki*

1: Let = Enc(mj,kj), where ∈ C \ i.*cj j*

2: Party uses −ki = to decrypt the aggregation of other parties as follows: (−ki) = P∈C\i− P∈C\i= P∈C\i.*i* P*j*∈C\i*kj Dec*P*j*∈C\i*cj,jcj jkj jmj*



Model parameter of party is updated as per gradientsencrypted SGD as follows:*i*

*wi*0 = *wi* 

= + ∆+ P∈C\i∆where and correspond to encryption and decryption operations in Algorithm 3, is the local parameters of party at previous round, ∆is the masked gradient vector of party shared with party , where only *wi wi jw***˜˜˜˜***ji Enc Dec wi i wji j idji* gradients are meaningful, i.e., elements of total |∆| *wji*elements are kept intact, while the remaining elements are nullified as 0. The second equality follows the homomorphic addition property, thus participant *i* can get the updated correctly after decryption, without having access to either ∆or ∆. FPPDL ensures party obliviousness by ensuring that each participant knows nothing but the sum of its received gradients in each communication round, and cannot infer any information about other participants&apos; data.*wi*0 *wji wj*

### 5.2.2        Three-layer Onion-style Encryption

However, as all parties need to store different encrypted gradients that are meant to be sent to different parties on Blockchain for commitment, all the encrypted gradients are also accessible to all parties. Applying public-key encryption on top of homomorphic encryption for authentication [42] can address this problem. However, as the released gradient vector is high-dimensional, encrypting gradient vector is both computation and communication expensive.

Therefore, we propose a three-layer onion-style encryption scheme. The first layer protects local model gradients by using symmetric key keystream for homomorphic encryption, as presented in Algorithm 3. The second layer and the third layer are classic hybrid encryption, as used in OpenPGP [44] for instance. In particular, in the second layer, a fresh symmetric encryption key will be generated and used to re-encrypt the ciphertext of the first layer, and then the fresh symmetric key is encrypted by using the receiver&apos;s public key in the third layer. In this way, the encryption of high-dimensional data becomes very effective, and the receiver could be authenticated as well: only the receiver who has the corresponding secret key paired with the public key can decrypt the two-layer encrypted gradients committed on the Blockchain.*kj fsk pki ski pki*

### 5.2.3        Local Credibility Update

Instead of using the standalone models as in the local credibility initialization, during each round of collaborative learning, each party randomly selects and shares a subset of DPGAN samples as per individual sharing level, then calculates the local credibility of other parties based on the returned labels, which are evaluated by using its updated local model at current round. The mutual evaluation follows the same procedure as in Step 3 of Algorithm 1. Finally, local credibility of each party is updated by integrating its historical local credibility as per Step 4 of Algorithm 2. In this way, local credibility of each party can be adaptively updated, reflecting more accurately how one party contributes to different parties during collaborative learning.

## 5.3       Quantification of Fairness

In collaborative learning system, collaborative fairness should be quantified from the point of view of the whole system. In this work, we quantify collaborative fairness through the correlation coefficient between party contributions (i.e., standalone model accuracy which characterizes the learning capability of each party on its own local data, and sharing level, which characterizes the sharing willingness of each party) and party rewards (i.e., final model accuracies of different parties).

Specifically, we take party contributions as the X-axis, which represents the contributions of different parties from the system view. In particular, in Setting 2, we characterize different parties&apos; contributions by their sharing levels and standalone model accuracies, as the party who is less private and has local data with better generalization empirically contributes more. In Setting 1 and Setting 3, we characterize different parties&apos; contributions by their standalone model accuracies, as the party who has local data with better generalization empirically contributes more. Moreover, in Setting 3, the party with more local data typically yields higher standalone model accuracy in IID scenarios. In summary, the X-axis can be expressed by Equation 2, where and denote the sharing level and standalone model accuracy of party respectively:*λj saccj j*

                                                nSetting 2*,*

*x* = *j j saccj saccj*

                                        {sacc1,···,saccn}, Setting 1&3

(2)

Similarly, we take party rewards (i.e., final model accuracies of different parties) as the Y-axis, as expressed by Equation 3, where denotes the final model accuracy of party :*accj j*

*y* = {acc1,···,accn} (3)

As the Y-axis measures local model performance of different parties after collaboration, it is expected to be positively correlated with the X-axis to deliver good fairness. Hence, we formally quantify collaborative fairness in Equation 4:

(4)

where ¯ and ¯ are the sample means of and , and are the corrected standard deviations. The range of fairness is within [-1,1], with higher values implying good fairness. Conversely, negative coefficient implies poor fairness.*xyx ysx sy*

# 6         EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed FPPDL framework by comparing it against the state of the art on real-world datasets.

## 6.1       Datasets

We implement experiments on two benchmark image datasets. The first is the MNIST dataset1 for handwritten digit recognition consisting of 60,000 training examples and 10,000 test examples. Each example is a 32x32 gray-level image [11], with digits locating at the center of the image. The second is the SVHN dataset2 of house numbers obtained from Google&apos;s street view images, which contains 600,000 training examples, from which we use 100,000 for training

1.  http://yann.lecun.com/exdb/mnist/

2.  http://ufldl.stanford.edu/housenumbers/

and 10,000 for testing. Each example is a 32x32 centered image with three channels (RGB). SVHN is more challenging as most of the images are noisy, and contain distractors at the sides. The size of the input layer of neural networks for MNIST and SVHN are 1024 and 3072, respectively. The objective is to classify the input as one of 10 possible digits within [“0”-“9”], thus the size of the output layer is 10. We normalize the training examples by subtracting the average and dividing by the standard deviation of training examples. For reproducibility purposes, our code will be made available here: https://github.com/lingjuanlv/FPPDL.

## 6.2       Baselines

We demonstrate the effectiveness of our proposed FPPDL framework by comparison with the following three frameworks. In all frameworks, stochastic gradient descent (SGD) is applied to each party.

1) framework: which assumes parties train standalone models on local training data without any collaboration. This framework delivers maximum privacy, but minimum utility, because each party is susceptible to falling into local optima when training alone.*Standalone*

2) framework: which allows a trusted server to have access to all participants&apos; data in the clear, and train a global model on the combined data using standard SGD. Hence, it is a privacy-violating framework.*Centralized*

3) framework: which enables parties to train independently and concurrently, and chooses a fraction of parameters to be uploaded at each iteration. In particular, as shown in [11], Distributed Selective SGD (DSSGD) achieves even higher accuracy than the centralized SGD because updating only a small fraction of parameters at each round acts as a regularization technique to avoid overfitting. Hence, we take DSSGD for the analysis of the distributed framework. As DSSGD with round robin parameter exchange protocol results in the highest accuracy [11] and facilitates fairness calculation, we follow the round robin protocol for DSSGD, where participants run SSGD sequentially, each downloads a fraction of the most updated parameters from the server, runs local training, and uploads selected gradients; the next party follows in the fixed order. Gradients are uploaded according to the “largest values” criterion.*Distributed*

## 6.3       Experiment Setup

For local model architecture, we consider two popular neural network architectures: multi-layer perceptron (MLP) and convolutional neural network (CNN), which are the same as in [11]. For local model training, we set the learning rate as 0.001, learning rate decay as 1e-7, and mini-batch size as 1. In addition, to reduce the impact of different initializations and avoid non-convergence, each party is initialized with the same parameter , then local training is run on individual training data to update local model parameter *w*0*wi*. To boost fairness, we let each party individually train 10 epochs before collaborative learning starts. For all experiments, we empirically set the local credibility threshold as via grid search, where |C| is the number of alive parties, i.e., credible parties in the system. Next, we investigate three realistic IID settings as follows:

**Setting 1: Same sharing level, same data size:** in the first case, sharing level of each party is set as 0.1, i.e., each party only releases 10% meaningful gradients during collaboration. For each party, we randomly sample 1% of the entire database as the local training data of each party, i.e., 600 examples for MNIST and 1000 examples for SVHN, this setting is the same as Shokri et al. [11] when the upload rate of each party equals 0.1;

**Setting 2: Different sharing level, same data size:** in the second case, sharing level of each party is randomly sampled from [0.1,0.5], and parties release meaningful gradients as per individual sharing level during collaboration. For each participant, we randomly sample 1% of the entire database as local training data as above.

**Setting 3: Different data size, same sharing level:** in the third case, we simulate the case where different parties have different data size. In particular, for MNIST dataset, we randomly partition total {2400, 9000, 18000, 30000} examples among {4,15,30,50} parties respectively. Similarly, for SVHN dataset, total {4000, 15000, 30000, 50000} examples are randomly partitioned among {4,15,30,50} parties respectively. The sharing level of each party is fixed to 0.1.

**Remark**. In Setting 1 and Setting 2, the purpose of allocating 600 MNIST examples or 1000 SVHN examples for each party is to fairly compare with Shokri et al. [11], in which each party is allocated with 600 MNIST examples or 1000 SVHN examples (small number of local examples to simulate data scarity which necessitates collaboration). Therefore, for MNIST, we simulate the total examples of 2400 (4 parties) up to 30,000 (50 parties). For larger datasets like 300,000 examples, it would require 500 parties, imposing heavy requirement on real deployment, while delivering similar analysis as in Sec. 6.4. We also remark that our Setting 2 and Setting 3 are relatively conservative, by increasing the contribution diversity among parties, for example, sampling sharing level from [0,1] instead of [0,0.5], partitioning data size among parties in a more imbalanced way, our FPPDL can definitely results in higher fairness.

## 6.4       Experimental Results

TABLE 3: Fairness of distributed framework and our FPPDL over MNIST dataset, with different model architectures, different party numbers (P-) and different settings as described in Section 6.3.*k*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Setting 2 | | | | Setting 3 | | | |
| Distributed | | FPPDL | | Distributed | | FPPDL | |
|  | CNN | MLP | CNN | MLP | CNN | MLP | CNN | MLP |
| *P4* | -0.68 | 0.30 | **0.89** | **0.92** | -0.97 | 0.05 | **0.98** | **0.96** |
| *P15* | 0.20 | -0.15 | **0.76** | **0.82** | 0.03 | -0.07 | **0.90** | **0.83** |
| *P30* | -0.02 | 0.02 | **0.79** | **0.85** | 0.13 | 0.01 | **0.75** | **0.63** |
| *P50* | -0.16 | -0.05 | **0.75** | **0.67** | 0.14 | -0.07 | **0.72** | **0.60** |

For collaborative fairness comparison, we only analyze our FPPDL and the distributed framework using DSSGD, neglecting centralized framework and standalone framework, because parties cannot get access to the trained global model in the centralized framework, while parties do not collaborate in the standalone framework. Table 3 and Table 4 TABLE 4: Fairness of distributed framework and our FPPDL over SVHN dataset, with different model architectures, different party numbers (P-) and different settings.*k*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Setting 2 | | | | Setting 3 | | | |
| Distributed | | FPPDL | | Distributed | | FPPDL | |
|  | CNN | MLP | CNN | MLP | CNN | MLP | CNN | MLP |
| *P4* | 0.27 | 0.26 | **0.78** | **0.76** | 0.28 | 0.20 | **0.97** | **0.93** |
| *P15* | 0.16 | 0.19 | **0.77** | **0.71** | -0.13 | 0.16 | **0.87** | **0.88** |
| *P30* | -0.14 | 0.12 | **0.68** | **0.65** | -0.15 | -0.27 | **0.67** | **0.78** |
| *P50* | -0.25 | -0.37 | **0.67** | **0.66** | -0.23 | 0.15 | **0.65** | **0.69** |

list the calculated fairness of the distributed framework and our FPPDL over MNIST and SVHN datasets, with different architectures, different party numbers and different settings, as detailed in Section 6.3. In particular, we omit the results for setting 1 with the same sharing level and same data size, as fairness is a less concerned problem in this setting. All the fairness results for setting 2 and setting 3 are averaged over five trails to reduce the impact of different initialization in each trail.

As is evidenced by the high positive values of fairness, with most of them above 0.5, FPPDL achieves reasonably good fairness, confirming the intuition behind fairness: the party who is less private and has more training data delivers higher accuracy. In contrast, the distributed framework exhibits bad fairness with significantly lower values than that of FPPDL in all cases, and even negative values in some cases, manifesting the lack of fairness in the distributed framework. This is because in the distributed framework, all the participating parties can derive similarly well models, no matter how much one party contributes.

**System-level Convergence**. For accuracy comparison, following [11], we report the best accuracy when running the distributed framework using DSSGD and our FPPDL on MNIST dataset. For DSSGD, we adopted round robin protocol, and set the upload rate as 0.1 (= 0.1) [11], which is equivalent to our Setting 1, we omit the learning curves of DSSGD in Setting 2 and Setting 3 as they approximate the learning curve in Setting 1. Fig. 6 and Fig. 7 present the accuracy trajectories when running different frameworks over MNIST with MLP and CNN architectures. The x-axis corresponds to epochs (communication rounds) (1 round=1 epoch, when the number of local epochs = 1), and y axis corresponds to the maximum accuracy achieved by all parties in each round, hence the curve of our FPPDL is not necessarily associated with a particular party, but it is expected that the highest accuracy is achieved by the most contributive party in our FPPDL, as demonstrated by the individual convergence in Fig. 9, Fig. 10 and Fig. 11.*θu E*

|  |
| --- |
| Fig. 6: System convergence for MNIST MLP. Collaboration involves different number of parties in {4,15,30,50}.    Fig. 7: System convergence for MNIST CNN. Collaboration involves different number of parties in {4,15,30,50}. |

Note that the convergences of the standalone framework in Setting 1 and Setting 2 are the same, as these two settings share the same data shard. It can be observed that FPPDL did not change the overall behavior of convergence in all settings, while achieving comparable accuracy to the non-private frameworks, and delivering both fairness and privacy. We notice that our FPPDL achieves slightly slower convergence rate and more fluctuations (especially in early stages of convergence) compared to the distributed framework, this is partly attributed to the individual training of 10 epochs before collaborative learning starts, as we found that collaboration from the state of 10 epochs of local training results in better fairness than the collaboration from the beginning.

Another important reason is that to strike a good balance between computational efficiency, communication cost and convergence rate, we enforce parties to share their local model updates after each epoch of local training (= 1), where the shared gradients is the average of the gradients over the whole local training data, rather than a single example, a mini-batch or multiple local epochs, which may also affect convergence. We hypothesise that the convergence rate is also closely related with our chosen hyperparameters = 1,E = 1,lr = 0.001 (: number of local training epochs in each communication round; : local batch size; : local learning rate). Better convergence can be achieved by varying the amount of local computation per communication round, local batch size or the learning rate, as indicated in Fig. 8 and Fig. 11 by using B=10, E=5, lr=0.15.*E B EBlr*

**Individual Convergence**. To investigate the impact of our FPPDL on individual convergence, Fig. 9 and Fig. 10 further depict the accuracy trajectory of each party when running Standalone framework and our FPPDL with CNN architecture over MNIST across 100 communication rounds. For the sake of brevity, we only report experimental results obtained for the collaboration among 4 parties and 15 parties in Setting 2 and Setting 3. It can be observed that our FPPDL consistently delivers better accuracy than any standalone model obtained by any individual party, at the cost of slower convergence and more fluctuation. However, most parties can converge within the first 20 rounds, except those with lower standalone accuracy. For example, in Figure 10 (d), party 4 and party 9 encounter higher fluctuations compared with the other parties with higher standalone accuracy. More importantly, these figures confirm that our FPPDL enforces all parties to converge to different local models, which are better than their standalone models without any collaboration, thereby offering fairness as claimed.

To speed up convergence and alleviate fluctuations, we further experiment with larger number of local epochs, larger local batch size, and higher learning rate. As corroborated by Fig. 11, by setting = 10,E = 5,lr = 0.15, each party can converge faster, without affecting both accuracy and fairness. For example, for P15 in Figure 10 (d), it needs 65 communication rounds for all parties to converge using = 1,E = 1,lr = 0.001, while it only needs 50 communication rounds using = 10,E = 5,lr = 0.15 in Figure 11 (d). However, this faster convergence and less fluctuations come at the cost of local computation at each party.*B B B*

|  |
| --- |
| Fig. 8: System convergence for MNIST MLP and CNN using our FPPDL in Setting 2 and Setting 3 (B=10, E=5, lr=0.15).    Fig. 9: Individual convergence for MNIST CNN using Standalone framework and our FPPDL (P4, B=1, E=1, lr=0.001).    Fig. 10: Individual convergence for MNIST CNN using Standalone framework and our FPPDL (P15, B=1, E=1, lr=0.001).    Fig. 11: Individual convergence for MNIST CNN using our FPPDL in P4 and P15 (B=10, E=5, lr=0.15). |

Table 5 provides the accuracy results we obtain when running different frameworks on MNIST dataset of {4,15,30,50} parties for different neural network architectures. For all frameworks, we report the best accuracy the system can achieve across all rounds. In particular, in our FPPDL, fairness enables each party to get a different local model after collaborative learning, and we expect that the most contributive party derives a local model with maximum accuracy approximating the non-private centralized and distributed frameworks. Similarly, Table 6 provides the accuracy on SVHN dataset. For both MNIST and SVHN datasets using CNN and MLP architectures, we show the worst accuracy for standalone SGD (minimum utility, maximum privacy). In particular, FPPDL obtains comparable accuracy (less than 2%) to both the centralized framework and the distributed framework using DSSGD without differential privacy, and consistently achieves higher accuracy than the standalone SGD. For example, as shown in Table 5, for MNIST dataset of 50 parties with CNN model, our FPPDL achieves 98.07%-98.22% test accuracy under different settings, which is higher than the standalone SGD 94.05%, and comparable to 98.83% of the distributed framework using DSSGD without differential privacy, and 98.58% of the centralized framework.

The above fairness results in Table 3 and Table 4, and accuracy results in Table 5 and Table 6 demonstrate that our proposed framework FPPDL achieves reasonable fairness, at the expense of a tiny decrease in model utility.

Moreover, to investigate how fairness and accuracy change with the local credibility threshold , we implement a four-party scenario (P4) under both normal settings*cth*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TABLE 5: MNIST accuracy [%] after 100 communication rounds, achieved by Centralized, Standalone, Distributed (DSSGD without DP, round robin, = 10%) and FPPDL (three settings as described in Section 6.3) frameworks using MLP and CNN architectures. P-indicates there are parties in the experiments.*θu k k*   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Framework |  | M | LP |  |  |  | CNN | |  | | P4 | P15 | P30 | P50 | P4 | P15 |  | P30 | P50 | | *Centralized* | 91.68 | 95.17 | 96.28 | 96.85 | 96.58 | 98.19 |  | 98.52 | 98.58 | | *Distributed* | 91.67 | 95.17 | 96.33 | 97.35 | 96.25 | 98.04 |  | 98.63 | 98.83 | | *Standalone (Setting 1*&*2)* | 87.39 | 88.06 | 88.64 | 88.80 | 93.81 | 93.46 |  | 94.04 | 94.05 | | *Standalone (Setting 3)* | 89.61 | 88.83 | 89.57 | 89.52 | 94.42 | 95.44 |  | 95.11 | 95.45 | | *FPPDL (Setting 1)* | 90.13 | 94.42 | 94.88 | 95.57 | 95.93 | 97.19 |  | 97.62 | 98.07 | | *FPPDL (Setting 2)* | 91.92 | 95.70 | 95.94 | 96.23 | 95.50 | 97.34 |  | 97.84 | 98.14 | | *FPPDL (Setting 3)* | 90.75 | 94.37 | 94.75 | 95.21 | 95.23 | 97.50 |  | 97.82 | 98.22 |   TABLE 6: SVHN accuracy [%] after 100 communication rounds, achieved by Centralized, Standalone, Distributed (DSSGD without DP, round robin, = 10%) and FPPDL (three settings as described in Section 6.3) frameworks using MLP and CNN architectures.*θu*   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Framework |  | M | LP |  |  |  | CNN | |  | | P4 | P15 | P30 | P50 | P4 | P15 |  | P30 | P50 | | *Centralized* | 75.40 | 83.08 | 85.77 | 87.15 | 90.50 | 91.88 |  | 93.42 | 95.44 | | *Distributed* | 78.34 | 85.49 | 87.64 | 89.21 | 91.78 | 93.03 |  | 95.75 | 96.19 | | *Standalone (Setting 1*&*2)* | 57.85 | 58.77 | 57.90 | 59.18 | 80.24 | 80.74 |  | 81.29 | 81.60 | | *Standalone (Setting 3)* | 59.05 | 59.13 | 60.09 | 60.22 | 81.57 | 81.92 |  | 82.06 | 82.31 | | *FPPDL (Setting 1)* | 73.74 | 82.55 | 84.86 | 86.51 | 90.07 | 91.18 |  | 92.74 | 94.83 | | *FPPDL (Setting 2)* | 74.16 | 82.67 | 85.25 | 86.57 | 89.91 | 91.15 |  | 92.59 | 95.18 | | *FPPDL (Setting 3)* | 74.57 | 82.95 | 85.37 | 86.34 | 89.53 | 91.03 |  | 93.13 | 94.89 | |

(Setting 2 and Setting 3 in Section 6.3) and malicious setting

(1 malicious party as indicated in Section 7). As shown in Fig. 12, both fairness and accuracy can keep relatively high values when *cth* is within . In contrast, too small allows even the malicious party to sneak into the collaborative learning system without being detected and isolated, resulting in lower fairness, as manifested by the last figure of Fig. 12. On the contrary, too large *cth* might isolate most participants in the system. For example, will terminate the system within the first 5 rounds during the second stage of collaborative learning, resulting in both lower fairness and accuracy; and *cth* = min{|*C*|−2 1*,*1} will terminate the system after the first stage, and second stage of collaborative learning will never start, thus there is no collaborative fairness. These results validate our hypothesis in Section 5.1.1 and provide empirical support on our chosen in Section 6.3.

**Complexity Analysis**. The main communication cost occurs when each party sends its encrypted gradients to the other (n − 1) parties, resulting in (n − 1) ∗ L ciphertexts, where and are the number of parties and the size of the released gradients (the encrypted symmetric key size is negligible compared with the encrypted gradients). Therefore, our framework is applicable to practical applications to businesses [37], such as biomedical or financial institutions where the number of parties is limited. On the other hand, the main computation cost occurs at each party who needs to train a local DPGAN during initial benchmarking, compute local gradients, and conduct three-layer onion-style encryption during collaborative deep learning. However, all parties can individually train their DPGAN models offline before collaborative deep learning starts, and all parties can individually train local models in parallel, hence deep learning computation cost is not an obstacle for those parties with enough computational power. Moreover, our encryption scheme using stream ciphers and hybrid encryption is relatively efficient, because encrypting a short plaintext (i.e., the symmetric key) requires only one asymmetric operation, while encrypting a longer message (released gradients) would in theory require many asymmetric operations.*n L*

# 7         DISCUSSIONS

**Data Augmentation and Collaboration.** To facilitate credibility initialization, we apply data augmentation to expand local data size to help DPGAN generate reliable samples within a moderate privacy budget. However, data augmentation is intended to increase the amount of training data using information inherent in local training data, and thus improve the generalizability of local model, while not helpful for generalizing to unseen data. In other words, it cannot represent global distribution, and this explains why parties still need collaboration for better utility even after data augmentation. By using DPGAN, it not only preserves privacy of the original data, but also preserves privacy of the augmented data that are similar to the original data.

|  |
| --- |
| Fig. 12: How affects fairness and accuracy in normal and malicious settings. First two figures correspond to setting 2 with sharing level of [0.01,0.1,0.25,0.35] and [0.1,0.2,0.3,0.4], and setting 3 with data shard of [437,980,150,833] among four honest parties. The last figure simulates setting 3 with data shard of [437,980,150,833]) among four honest parties*cth* |

**Fairness and Privacy.** With three-layer onion-style encryption, privacy is better preserved without compromising utility. We ensure fairness from two ways: (i) during initial benchmarking, parties generate DPGAN samples based on their local training data, which are then evaluated by other parties&apos; standalone models to mutually initialize the local credibilities of other parties; and (ii) during collaborative learning process, each party randomly selects and shares a subset of DPGAN samples as per individual sharing level at each round of communication, then updates the local credibility values for other parties who evaluate the received DPGAN samples using their local models at current round. Therefore, local credibility of each party keeps changing, reflecting more accurate relative contribution and thus possessing better fairness. Differentially private training of deep models provides another alternative solution by releasing gradients after each epoch or several epochs of and one more malicious party indicated in Section 7.

.

local training, thus enabling each party to verify the claims of other parties and update their local credibility values as per the received gradients during collaborative learning process. One obstacle is that differentially private models may significantly reduce utility for small values.

**Attacker Prevention.** Although the capability of detecting and isolating malicious parties is not the main focus of this paper, we next discuss how our design can help prevent certain behaviours of inside attacker, and resist the outside attacker as a by-product of FPPDL.

For an inside attacker who is a participant in the decentralized system, we specially consider an interesting case: a free-rider without any data, and we remark that this freerider belongs to the category of low-contribution party. During initialization, this free-rider may choose to send the fake information to other parties. For example, it may randomly sample from 10 classes as predicted labels for the received DPGAN samples, then release them to the corresponding party who publishes these DPGAN samples and requests labels. When the publisher receives the returned random labels from the free-rider and detects that most of them are not aligned with the majority voting, i.e., , then the free-rider will be reported as a “low-contribution” party. If the majority of parties report the free-rider as “lowcontribution”, then the Blockchain rules out the free-rider from the credible party set, and all parties would terminate the collaboration with the free-rider. In this way, such a malicious party is isolated from the beginning, while the collaboration among the remaining parties will not be affected. Even though the free-rider might succeed in initialization somehow, its local credibility would be significantly lower compared with the other honest parties.

To further detect and isolate this malicious party during the collaborative learning process, we repeat mutual evaluation at each round of collaborative learning by using samples generated at the initialization phase, i.e., each party randomly selects and shares a subset of DPGAN samples as per individual sharing level in each round of collaborative learning, then updates the local credibility values of other parties by comparing the majority labels with the received labels output by the local models of other parties in current round of training. Hence, the chance of the survival of the malicious party is significantly reduced, thus it will not dominate the whole system. Note that the lower bound of the acceptable credibility threshold can be agreed by the system requirement. For the outsider attacker like the eavesdropper who aims to steal the exchanged information by eavesdropping on the communication channels among parties, differential privacy used in the first stage and threelayer onion-style encryption applied in the second stage inherently prevent the success of this attack.

We recognize that our current design may not be resistant to all the malicious parties who can arbitrarily deviate from the protocol, sending incorrect and/or arbitrarily chosen messages to honest parties, aborting, omitting messages, and sharing their entire view of the protocol with each other. For example, a malicious party who aims to compromise other parties&apos; local model integrity (prevent other parties from learning reasonable models) may adaptively or alternatively adjust its behaviour by behaving normally during releasing DPGAN samples to avoid being detected and kicked out, while poisoning the second stage by sending random local gradients or local gradients with the embedded backdoor behavior to the requester. However, in this case, this malicious party is unlikely to obtain a reasonable local model or steal any party&apos;s personal information.

Moreover, to prevent the success of the poisoning attack, one potential solution is to let each party repeat local prediction process on its hold-out validation set by using individually aggregated gradients. Each party will release a signal to indicate whether its aggregated gradients can give a reasonable accuracy result, or help improve prediction on local validation set, if majority party report local validation accuracy lower than a threshold, or negative gain on local validation accuracy, then the system terminates to avoid being further poisoned. We leave this open problem to our future work, and our current design is mainly for the business applications, where parties act with legal liabilities.

# 8         CONCLUSIONS AND FUTURE WORK

This paper proposes FPPDL, a decentralized privacypreserving deep learning framework with fairness considerations. Our enhanced framework shows the following properties: (1) it inherently resolves the relevant issues in the server-based frameworks, and investigates Blockchain for decentralization; (2) it makes the first investigation on the research problem of collaborative fairness in deep learning, by introducing a notion of local credibility and transaction points, which are initialized by initial benchmarking, and updated during privacy-preserving collaborative deep learning; (3) it combines Differentially Private GAN (DPGAN) and a three-layer onion-style encryption scheme to guarantee both accuracy and privacy; (4) it provides a viable solution to detect and reduce the impact of low-contribution parties in the system. The experimental results demonstrate that our FPPDL achieves comparable accuracy to both the centralized and distributed selective SGD framework without differential privacy, and always delivers better results than the standalone framework, confirming the applicability of our proposed framework.

A number of avenues for further work are attractive. In particular, we would like to study how to quantify fairness in Non-IID setting, and investigate more malicious behaviours and byzantine or sybil adversary in the decentralized system. We also expect to deploy our system into a wide spectrum of real-world applications.

# ACKNOWLEDGMENTS

This work is supported, in part, by IBM PhD Fellowship; ANU Translational Fellowship; Nanyang Assistant Professorship (NAP); and NTU-WeBank JRI (NWJ-2019-007). The authors would like to thank Prof. Benjamin Rubinstein, Dr. Kumar Bhaskaran, and Prof. Marimuthu Palaniswami for their insightful discussions. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

# REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[2] Y. Wang, W. Liu, X. Ma, J. Bailey, H. Zha, L. Song, and S.-T. Xia, “Iterative learning with open-set noisy labels,” arXiv preprint arXiv:1804.00092, 2018.

[3] X. Ma, Y. Wang, M. E. Houle, S. Zhou, S. M. Erfani, S.-T. Xia, S. Wijewickrema, and J. Bailey, “Dimensionality-driven learning with noisy labels,” arXiv preprint arXiv:1806.02612, 2018.

[4] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, Federated Learning. Morgan & Claypool Publishers, 2019.

[5] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, “Federated learning of deep networks using model averaging,” arXiv preprint arXiv:1602.05629, 2016.

[6] R. Cummings, V. Gupta, D. Kimpara, and J. Morgenstern, “On the compatibility of privacy and fairness,” 2019.

[7] M. Jagielski, M. Kearns, J. Mao, A. Oprea, A. Roth, S. SharifiMalvajerdi, and J. Ullman, “Differentially private fair learning,” arXiv preprint arXiv:1812.02696, 2018.

[8] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang, “A fairness-aware incentive scheme for federated learning,” in Proceedings of the 3rd AAAI/ACM Conference on AI, Ethics, and Society (AIES-20), 2020, pp. 393–399.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao,

A. Senior, P. Tucker, K. Yang, Q. V. Le et al., “Large scale distributed deep networks,” in Advances in neural information processing systems, 2012, pp. 1223–1231.

[10] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, “Parallelized stochastic gradient descent,” in Advances in neural information processing systems, 2010, pp. 2595–2603.

[11] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015, pp. 1310–1321.

[12] L.-M. T-T Kuo, C-N Hsu, “Modelchain: Decentralized privacypreserving healthcare predictive modeling framework on private blockchain networks,” in ONC/NIST Blockchain in Healthcare and Research Workshop, Gaithersburg, MD, September 26-7, 2016.

[13] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, “When machine learning meets blockchain: A decentralized, privacy-preserving and secure design,” in 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018, pp. 1178–1187.

[14] H. Kim, S.-H. Kim, J. Y. Hwang, and C. Seo, “Efficient privacypreserving machine learning for blockchain network,” IEEE Access, vol. 7, pp. 136 481–136 495, 2019.

[15] X. Zhu, H. Li, and Y. Yu, “Blockchain-based privacy preserving deep learning,” in International Conference on Information Security and Cryptology. Springer, 2018, pp. 370–383.

[16] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo,

“Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive,” IEEE Transactions on Dependable and Secure Computing, 2019.

[17] T.-T. Kuo, R. A. Gabriel, and L. Ohno-Machado, “Fair compute loads enabled by blockchain: sharing models by alternating client and server roles,” Journal of the American Medical Informatics Association, vol. 26, no. 5, pp. 392–403, 2019.

[18] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, “Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy,” in International Conference on Machine Learning, 2016, pp. 201–210.

[19] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, “Oblivious multi-party machine learning on trusted processors.” in USENIX Security Symposium, 2016, pp. 619–636.

[20] P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” in Security and Privacy (SP), 2017 IEEE Symposium on. IEEE, 2017, pp. 19–38.

[21] L. Lyu, X. He, Y. W. Law, and M. Palaniswami, “Privacypreserving collaborative deep learning with application to human activity recognition,” in Proceedings of the 2017 ACM Conference on Information and Knowledge Management. ACM, 2017, pp. 1219–

1228.

[22] L. Lyu, J. C. Bezdek, X. He, and J. Jin, “Fog-embedded deep learning for the internet of things,” IEEE Transactions on Industrial Informatics, 2019.

[23] Y. Aono, T. Hayashi, L. Wang, S. Moriai et al., “Privacy-preserving deep learning via additively homomorphic encryption,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1333–1345, 2018.

[24] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 1175–1191.

[25] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, “Learning differentially private recurrent language models,” arXiv preprint arXiv:1710.06963, 2018.

[26] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, “On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing,” IEEE Journal on Selected Areas in Communications, vol. 35, no. 4, pp. 832–847, 2017.

[27] S. Gollapudi, K. Kollias, D. Panigrahi, and V. Pliatsika, “Profit sharing and efficiency in utility games,” in ESA, 2017, pp. 1–16.

[28] A. Richardson, A. Filos-Ratsikas, and B. Faltings, “Rewarding high-quality data via influence functions,” in CoRR, 2019, p. arXiv:1908.11598.

[29] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” Foundations and Trends R , vol. 9, no. 3–4, pp. 211–407, 2014.*in Theoretical Computer Science*

[30] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 308–318.

[31] C. Gentry and D. Boneh, A fully homomorphic encryption scheme. Stanford University Stanford, 2009, vol. 20, no. 09.

[32] I. Damgard, V. Pastro, N. Smart, and S. Zakarias, “Multiparty com-˚ putation from somewhat homomorphic encryption,” in Annual Cryptology Conference. Springer, 2012, pp. 643–662.

[33] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. NayaPlasencia, P. Paillier, and R. Sirdey, “Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression,” Journal of Cryptology, vol. 31, no. 3, pp. 885–916, 2018.

[34] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[35] C. Natoli, J. Yu, V. Gramoli, and P. J. E. Ver´ıssimo, “Deconstructing blockchains: A comprehensive survey on consensus, membership and structure,” CoRR, vol. abs/1908.08316, 2019.

[36] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A survey on bias and fairness in machine learning,” in CoRR, 2019, p. arXiv:1908.09635.

[37] L. Lyu, H. Yu, and Q. Yang, “Threats to federated learning: A survey,” arXiv preprint arXiv:2003.02133, 2020.

[38] X. Zhang, S. Ji, and T. Wang, “Differentially private releasing via deep generative model,” arXiv preprint arXiv:1801.01594, 2018.

[39] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” arXiv preprint arXiv:1610.05755, 2016.

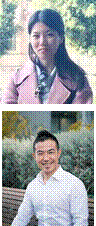
[40] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” arXiv preprint arXiv:1701.07875, 2017.

[41] C. Castelluccia, E. Mykletun, and G. Tsudik, “Efficient aggregation of encrypted data in wireless sensor networks,” in Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005. IEEE, 2005, pp. 109–117.

[42] L. Lyu, K. Nandakumar, B. Rubinstein, J. Jin, J. Bedo, and M. Palaniswami, “PPFA: Privacy preserving fog-enabled aggregation in smart grid,” IEEE Transactions on Industrial Informatics, vol. 14, no. 8, pp. 3733–3744, 2018.

[43] C. De Canniere and B. Preneel, “Trivium,” in New Stream Cipher Designs. Springer, 2008, pp. 244–266.

[44] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, “Openpgp message format,” Tech. Rep., 2007.

**Lingjuan Lyu** (IEEE M&apos;18) is currently a Research Fellow with The Department of Computer Science, National University of Singapore. She received Ph.D. degree from the University of Melbourne. Her current research interests span machine learning, privacy, fairness, and edge intelligence. Her work was supported by an IBM Ph.D. Fellowship.

**Jiangshan Yu** received the Ph.D. degree from the University of Birmingham (UK) in 2016. He is currently Associate Director (Research) at Monash Blockchain Technology Centre at Monash University, Australia. Previously, he was a research associate at SnT, University of Luxembourg (LU). The focus of his research has been on design and analysis of cryptographic protocols, cryptographic key management, blockchain consensus, and ledger-based applications. In particular, Jiangshan&apos;s recent re-

search challenges the soundness of the foundational security models and design principles of existing blockchain systems, where the blockchain ecosystem of hundreds of billions of dollars is based upon. He won numerous prestigious awards, including Dean&apos;s Research Impact Award (2019) and the Chinese Government Award for Outstanding Scholar Abroad (1% worldwide, 2016).

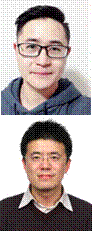
**Karthik Nandakumar** (IEEE SM&apos;02) is a Research Staff Member at IBM Research, Singapore. Prior to joining IBM in 2014, he was a Scientist at Institute for Infocomm Research, A\*STAR, Singapore for more than six years. He received his B.E. degree (2002) from Anna University, Chennai, India, M.S. degrees in Computer Science (2005) and Statistics (2007), and Ph.D. degree in Computer Science (2008) from

Michigan State University, and M.Sc. degree in

Management of Technology (2012) from Na-

tional University of Singapore. His research interests include computer vision, statistical pattern recognition, biometric authentication, image processing, machine learning and blockchain.

**Yitong Li** is currently a Ph.D student in School of Computing and Information Systems, the University of Melbourne. He received B.S. degree from Shanghai Jiao Tong University. His research interests cover privacy and adversarial learning with NLP applications. He has publications in ACL, EMNLP, NAACL, etc.

**Xingjun Ma** is currently a Research Fellow of the University of Melbourne. He received Ph.D. degree from the University of Melbourne, and M.E. degree from Tsinghua University. His research interests cover adversarial machine learning and robust supervised/weakly-supervised learning. He has publications in ICML, ICLR, CVPR, IJCAI, AAAI, ICCV, etc.

**Jiong Jin** (IEEE M&apos;11) received the B.E. degree with First Class Honours in Computer Engineering from Nanyang Technological University, Singapore, in 2006, and the Ph.D. degree in Electrical and Electronic Engineering (EEE) from the University of Melbourne, Australia, in 2011. From 2011 to 2013, he was a Research Fellow in the Department of EEE at the University of Melbourne. He is currently a Senior Lecturer in the School of Software and Electrical Engineering,

Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia. His research interests include network design and optimization, edge computing and distributed systems, robotics and automation, and cyber-physical systems and Internet of Things as well as their applications in smart manufacturing, smart transportation and smart cities.

**Han Yu** received his B.Eng. (Hons) degree and Ph.D. degree from the School of Computer

                                     Science and Engineering (SCSE), Nanyang

Technological University (NTU), Singapore in

2007 and 2014, respectively. He is currently a

Nanyang Assistant Professor (NAP) at SCSE,

NTU. From 2015 to 2018, he held the prestigious

Lee Kuan Yew Post-Doctoral Fellowship (LKY

PDF) at the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY).

His research focuses on the ethics of artificial

intelligence and federated learning. He co-authored the book “Federated Learning” - the first monograph on the topic of federated learning.

**Kee Siong Ng** is an Associate Professor in the newly formed Software Innovation Institute at the Australian National University (ANU), and one of the first two Translational Fellows appointed through ANU&apos;s Entrepreneurial Academic Scheme. He received his PhD degree from the ANU and has more than 15 years of experience in industry and government.