**隐私保护深度学习**

|  |  |
| --- | --- |
| 雷扎·肖克里  德克萨斯大学奥斯汀分校shokri@cs.utexas.edu | 维塔利·什马蒂科夫  康奈尔理工大学shmat@cs.cornell.edu |

# 摘要

基于人工神经网络的深度学习是一种非常流行的建模、分类和识别复杂数据（如图像、语音和文本）的方法。深度学习方法的前所未有的准确性使他们成为互联网上新的基于人工智能的服务的基础。大规模收集用户数据的商业公司是这一趋势的主要受益者，因为深度学习技术的成功与可用于培训的数据量成正比。

深度学习所需的大量数据收集带来了明显的隐私问题。用户的个人、高度敏感的数据，如照片和录音，由收集这些数据的公司无限期保存。用户既不能删除它，也不能限制它的用途。此外，集中保存的数据将受到法律传唤和法外监视。许多数据所有者—例如，可能希望将深度学习方法应用于临床记录的医疗机构—由于隐私和机密性的考虑而无法共享数据，从而从大规模深度学习中获益。

在本文中，我们设计、实现和评估了一个实用的系统，该系统可以让多方共同学习给定目标的精确神经网络模型，而无需共享他们的输入数据集。我们发现现代深度学习中使用的优化算法，即基于随机梯度下降的优化算法，可以被并行化和异步执行。我们的系统允许参与者在自己的数据集上独立训练，并在训练期间有选择地分享他们模型关键参数的一小部分。这在实用性/隐私性权衡方面提供了一个吸引人的地方：参与者在保护各自数据的隐私的同时，仍能从其他参与者的模型中获益，从而提高他们的学习精度，超出仅凭他们自己的输入就可达到的水平。我们证明了我们在基准数据集上保留私有深度学习的准确性。

# 类别和主题描述符

安全和隐私[软件和应用程序安全]：特定于域的安全和隐私体系结构

|  |
| --- |
| 允许免费制作本作品的全部或部分的数字或硬拷贝供个人或课堂使用，前提是复制品不是为了牟利或商业利益而制作或分发的，且副本在第一页上附有本通知和完整的引文。必须尊重作者以外的其他人拥有的本作品的版权。允许使用信用摘要。若要以其他方式复制或重新发布，在服务器上发布或重新分发到列表，则需要事先获得特定的许可和/或收费。从请求权限Permissions@acm.org。  *CCS&apos;15，*2015年10月12日至16日，美国科罗拉多州丹佛市。  版权归所有人/作者所有。授权给ACM的出版权。  ACM 978-1-4503-3832-5/15/10…15.00美元。  内政部：http://dx.doi.org/10.1145/2810103.2813687。 |

关键词

隐私；神经网络；深度学习；梯度下降

# 1        介绍

最近，基于人工神经网络的深度学习方法取得了突破性进展，在语音、图像和文本识别、语言翻译等长期存在的人工智能任务方面取得了突破，苹果利用从用户那里收集的大量培训数据和GPU农场巨大的计算能力，大规模部署深度学习。所得到的模型的前所未有的准确性允许它们被用作许多新的服务和应用的基础，包括准确的语音识别（24）和优于人类的图像识别[26 ]。

虽然深度学习的效用是不可否认的，但同样的训练数据，使它如此成功，也提出了严重的隐私问题。从数百万个人那里集中收集照片、演讲和视频的时机已经成熟，存在隐私风险。首先，收集这些数据的公司会永远保存这些数据；从中收集数据的用户既不能删除数据，也不能控制如何使用这些数据，也不能影响从中获得的信息。第二，图像和语音记录通常包含意外捕捉到的敏感项目，如人脸、车牌、电脑屏幕、其他人说话的声音和周围的噪音[44]等等。第三，公司保存的用户数据会受到传唤和授权，以及国家安全和情报机构的无担保间谍活动。

此外，互联网巨头对从数百万用户收集的“大数据”的垄断导致了他们对从这些数据中学习到的人工智能模型的垄断。用户可以从新的服务中获益，比如强大的图像搜索、语音激活的个人助理和外语网页的机器翻译，但是根据他们的集体数据构建的底层模型仍然是由创建它们的公司专有的。

最后，在许多领域，尤其是与医学相关的领域，法律或法规不允许共享个人数据。因此，生物医学和临床研究人员只能对属于他们自己机构的数据集进行深入学习。众所周知，随着训练数据集的不断扩大和多样化，神经网络模型变得更好。由于在训练他们的模型时不能使用其他机构的数据，研究人员最终可能得到更糟糕的模型。例如，一个单一的医疗机构可能会使用一个非常不准确的单一输入数据。在这种情况下，隐私和保密限制会大大降低实用性。

*我们的贡献。*我们设计、实现和评估了一个实用的协作式深度学习系统，它在实用性和隐私性之间提供了一个有吸引力的折衷方案。我们的系统可以让多个参与者根据自己的输入来学习神经网络模型，而不是共享这些输入，而是从同时学习相似模型的其他参与者那里获益。

我们的关键技术创新是在培训期间选择性地共享模型参数。这种参数共享与随机梯度下降过程中的局部参数更新交织在一起，使得参与者能够从其他参与者的模型中获益，而无需显式共享训练输入。我们的方法独立于为特定任务构造模型的特定算法。因此，在不改变核心协议的情况下，它可以很容易地适应未来神经网络训练的发展。

选择性参数共享是有效的，因为作为现代神经网络训练基础的随机梯度下降算法可以并行和异步运行。它们对不可靠的参数更新、竞赛条件、参与者退出等具有鲁棒性。用从其他参与者那里获得的值更新一小部分参数可以使每个参与者在寻找最优参数的过程中避免局部极小。可以调整参数共享，以控制交换的信息量和结果模型的准确性之间的权衡。

我们用MNIST和SVHN两个数据集作为图像分类算法的基准，对我们的系统进行了实验评估。在我们的系统中，由分布式参与者生成的模型的准确性接近于集中的、侵犯隐私的情况，即一方持有整个数据集并使用它来训练模型。对于MNIST数据集，当参与者共享10%（分别为1%）的参数时，我们获得99.14%的准确率（分别为98.71%）。相比之下，集中的、侵犯隐私的模型的最大正确率为99.17%，非协作模型的最大准确率为93.16%。对于SVHN数据集，当参与者共享10%（1%）的参数时，我们的准确率达到93.12%（89.86%）。相比之下，集中式、违反隐私的模型的最大准确率为92.99%，非协作模型的最大准确率为81.82%。

即使没有额外的保护，我们的系统已经实现了比任何现有的方法更强大的隐私性，并且可以忽略不计的效用损失。我们的系统中唯一的泄漏不是直接暴露所有的训练数据，而是间接的，通过一小部分神经网络参数。为了最小化这种泄漏，我们展示了如何使用稀疏向量技术对参数更新应用差分隐私，从而减轻由于参数选择（即选择共享哪些参数）和共享参数值而造成的隐私损失。然后我们定量地衡量准确性和隐私性之间的权衡。

# 2        相关工作

2.1深度学习

深度学习是从复杂数据中学习非线性特征和函数的过程。深度学习体系结构、算法和应用的调查可以在[5,16]中找到。深度学习在语音识别[23,24,27]、图像识别[30,45]和人脸检测[48]方面表现优于传统技术。一种基于新型整流激活函数的深度学习体系结构声称在从ImageNet数据集识别图像时表现优于人类[26]。

深度学习为分析与癌症[13,22,32]和遗传学[15,56]相关的复杂生物医学数据带来了希望。从隐私角度来看，用于构建这些模型的训练数据尤其敏感，这突出了保护隐私的深层学习方法的必要性。

我们的工作受到了并行化深度学习的最新进展的启发，特别是GPU/CPU集群上的随机梯度下降的并行化[14]，以及神经网络训练过程中分布计算的其他技术[1,39,59]。然而，这些技术并不涉及训练数据的隐私性，它们都假设由一个实体控制训练。

2.2机器学习中的隐私

现有的关于机器学习中隐私保护的文献主要针对传统的机器学习算法，而不是深度学习，并且解决了三个目标：用于学习模型或作为现有模型输入的数据的隐私性、模型的隐私性和模型输出的隐私性。

基于安全多方计算（SMC）的技术可以在多方对其专有输入进行协作机器学习时保护计算的中间步骤。SMC已用于学习决策树[33]、线性回归函数[17]、关联规则[50]、朴素贝叶斯分类器[51]和k-均值聚类[28]。一般来说，SMC技术会带来不小的性能开销，而它们在保护隐私的深度学习中的应用仍然是一个开放的问题。

保护模型隐私的技术包括隐私保护概率推断[38]、隐私保护说话人识别[36]和加密数据计算[3,6,55]。相比之下，我们的目标是合作训练一个神经网络，每个参与者都可以单独使用。

差异隐私[19]是一种流行的隐私保护机器学习方法。它被应用于boosting[21]、主成分分析[10]、线性和逻辑回归[8,57]、支持向量机[41]、风险最小化[9,53]和连续数据处理[43]。最近的结果表明，随机梯度下降的噪声变量在最小化Lipschitz凸函数的有界集上达到了最优误差[4]，而用于防止过拟合的随机“dropout”也加强了简单1层神经网络的隐私保障[29]。据我们所知，以前的工作都没有涉及到使用分布式随机梯度下降的多参与者协作深度学习问题。*`*2

并建议在安全计算中使用多党网络独立训练隐私权。不幸的是，平均神经网络参数不一定会得到更好的模型。

与之前提出的技术不同，我们的系统在协同神经网络训练的背景下实现了所有三个隐私目标：保护训练数据的隐私，使参与者能够控制学习目标以及对各自模型的披露程度，并让他们将共同学习的模型应用于自己的输入，而不暴露输入或输出。与安全多方计算或同态加密等密码技术相比，我们的系统以更低的性能成本实现了这一目标，适合于现代大规模深度学习。

# 3        深度学习

深度学习的目的是从高维数据中提取复杂的特征，并使用它们建立一个将输入与输出（例如，类）关联起来的模型。深度学习体系结构通常被构造成多层网络，因此更多的抽象特征被计算成较低层次特征的非线性函数。我们主要关注有监督的学习，即训练输入被标记为正确的类，但原则上我们的方法也可以用于无监督的、隐私保护的学习。

多层神经网络是最常见的深度学习体系结构。图1显示了具有两个隐藏层的典型神经网络。网络中的每个节点都模拟一个neu-

*是的*1         *是的*2



图1：具有两个隐藏层的神经网络。黑色圆圈表示偏移节点。矩阵包含用于计算每层激活函数的权重。*周k*

罗恩。在一个典型的多层网络中，每个神经元接收前一层神经元的输出加上一个特殊神经元发出的偏置信号，该信号发出1。然后计算其输入的加权平均值，称为总输入。神经元的输出是通过对总输入值应用一个非线性激活函数来计算的。层神经元的输出向量=f（Wk ak−1），其中是激活函数，是确定每个输入信号贡献的权重矩阵。激活函数的示例为双曲正切（z）=（e2z−1）（e2z+1）−1、sigmoid（z）=（1+e−z）−1、整流器（z）=max（0，z）和softplus（z）=log（1+ez）。如果使用神经网络将输入数据分类为有限数量的类（每个类由一个不同的输出神经元表示），最后一层的激活函数通常是一个softmax函数（zj）=ez·（）−1，∀j。在这种情况下，最后一层中每个神经元的输出是输入所属的相对分数或概率去上课。*k***一***kf周fffffj*P*kez公司kjj*

一般来说，在更高层计算的值代表数据的更抽象的特征。第一层由从数据中提取的原始特征组成，例如图像中每个像素的颜色强度或文档中每个单词的频率。最后一层的输出对应于模型产生的抽象答案。如果用神经网络进行分类，这些抽象特征也代表了输入和输出之间的关系。非线性函数和权值矩阵决定了在每一层提取的特征。深度学习的主要挑战是从训练数据中自动学习使神经网络目标最大化的参数（权重矩阵）的值（例如分类精度）。*f*

*利用梯度下降学习网络参数。*神经网络的参数学习是一个非线性优化问题。在有监督学习中，目标函数是神经网络的输出。用来解决这个问题的算法通常是梯度下降法的变种[2]。简单地说，梯度下降从一个随机点（神经网络的一组参数）开始，然后在每一步计算被优化的非线性函数的梯度并更新参数以减小梯度。这个过程会一直持续到算法收敛到局部最优。

在神经网络中，每个权值参数的梯度是通过前馈和反向传播过程来计算的。前馈按顺序计算给定输入数据的网络输出，然后计算误差，即该输出与函数真值之间的差。反向传播将误差传播回网络，并计算每个神经元对总误差的贡献。单个参数的梯度是根据神经元的激活值及其对误差的贡献来计算的。

*随机梯度下降（SGD）。*参数的梯度可以在所有可用数据上取平均值。这种被称为批量梯度下降的算法效率不高，尤其是在大数据集上学习时。随机梯度下降（SGD）是一种极端的简化，它计算整个数据集的极小子集（小批量）上的梯度[58]。在最简单的情况下，对应于最大随机性，在每个优化步骤中随机选择一个数据样本。

设神经网络中所有参数的展平向量，由∀k组成。设误差函数，即目标函数的真值与网络计算输出的差值。可以基于范数或交叉熵[34]。反向传播算法计算中每个参数的偏导数，并更新参数以减小其梯度。参数的随机梯度下降的更新规则是**w w w***周，EE我*2 *E威斯康星州*

（一）

其中是学习率，在小批量中计算。我们将所有可用输入数据的一次完整迭代称为一个历元。*α工程安装我*

请注意，vector中的每个参数都独立于其他参数进行更新。在本文的其余部分中，我们将利用这一特性来设计我们的隐私保护、协作随机梯度下降系统。一些技术自适应地设置学习速率[18]，但仍然保持这种独立性。**w**

# 4        分布式选择性SGD

我们方法的核心是一个分布式的、协作的深度学习协议，它依赖于以下观察结果：（i）梯度下降过程中对不同参数的更新本质上是独立的，（ii）不同的训练数据集对不同的参数有贡献，不同的特征对目标函数的贡献并不相等。我们的选择性随机梯度下降（选择性SGD或SSGD）协议达到了与传统SGD相当的精度，但是在每次学习迭代中更新1个甚至2个数量级的参数。

*选择性参数更新。*选择性参数更新背后的主要直觉是，在SGD过程中，某些参数对神经网络的目标函数贡献更大，因此在给定的训练迭代过程中会经历更大的更新。梯度值取决于训练样本（小批量），并且随样本的不同而变化。此外，输入数据的某些特征比其他特征更为重要，而帮助计算这些特征的参数在学习过程中更为关键，并且经历了更大的变化。

在选择性SGD中，学习者在每次迭代中选择一小部分参数进行更新。这种选择可以是完全随机的，但明智的策略是选择当前值远离其局部最优值的参数，即那些具有较大梯度的参数。对于每个培训样本*我*，计算所有参数的偏导数*威斯康星州*如新加坡元。让我们来看看*Sθ*具有最大值的参数。最后，更新参数向量**w***S*与（1）中的方法相同，因此不在中的参数保持不变。参数选择率是指参数占总参数数的比例。*Sθ*

*分布式协作学习。*分布式选择性SGD假设两个或多个参与者同时独立训练。每轮本地培训结束后，学员-

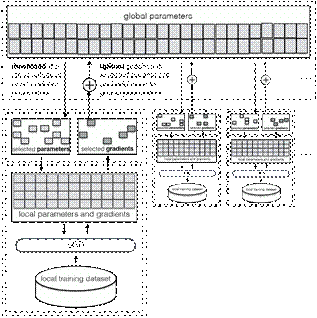


图2：我们的深度学习系统的高层架构。顶部描述了参数服务器的抽象模型，该模型维护参数的全局值。

彼此长期共享他们为某些参数计算的梯度。每个参与者完全控制共享哪些渐变以及共享频率。为给定参数计算的所有梯度之和决定了朝向参数局部最优值的全局下降幅度（“局部”指的是参数值的空间，并不意味着仅限于单个参与者）。因此，参与者可以从彼此的培训数据中获益，而无需实际看到这些数据！-并生成更精确的模型，这些模型仅限于他们自己的训练数据，他们本可以孤立地学习这些模型。

参与者可以直接交换渐变，或者通过可信的中央服务器，甚至可以使用安全的多方计算“不经意地”交换渐变，模拟一个隐藏每次更新来源的可信服务器的功能。出于本讨论的目的，我们假设一个中央服务器的抽象，参与者异步地将渐变上传到该服务器。服务器将所有渐变添加到相应参数的值中。每个参与者从服务器下载一个子集的参数，并使用它们来更新其本地模型。给定参数的下载标准可以是更新的频率或最近度，或者是添加到该参数的渐变移动平均值。

# 5        系统架构

5.1概述

图2说明了我们的协作式深度学习系统的主要组件和协议。我们假设有参与者，每个参与者都有一个可用于培训的本地私有数据集。所有参与者事先就共同的网络架构和共同的学习目标达成一致。我们假设存在一个参数服务器，该服务器负责维护可供各方使用的最新参数值。这个参数服务器是一个抽象，可以由实际服务器实现，也可以由分布式系统仿真。*N*

|  |  |
| --- | --- |
| *α* | 随机梯度下降的学习速率 |
| *θd*, *θ美国* | 选择用于下载和上载的参数的分数 |
| *γ* | 与其他参与者共享的梯度值绑定 |
| *τ* | 梯度选择阈值 |

表1：元参数列表

|  |
| --- |
| 选择初始参数和学习速率。**w**（一）*α*  重复上述步骤，直到获得近似最小值：  1.   下载*θd*××w（）一|参数，并替换相应的本地参数。  2.   在本地数据集上运行SGD并根据（1）更新本地参数。**w**（一）  3.   计算梯度向量∆w（i），这是由于SGD引起的所有局部参数的变化向量。  4.   上传到参数服务器，其中*S*是布景吗  最大×| w（i）|梯度的指数，根据以下标准之一选择：*θu*  •：按∆w（i）对梯度进行排序，并上传其中的分数，从最大值开始。*最大值θu*  •：随机对值高于阈值的梯度进行亚采样。*随机阈值τ*  整个培训的选拔标准是固定的。 |

图3：参与者的DSSGD伪代码。*我*

每个参与者初始化参数，然后在自己的数据集上运行训练。该系统包括一个参数交换协议，允许参与者将所选神经网络参数的梯度上传到参数服务器，并下载每个本地SGD历元的最新参数值。这使得参与者能够（i）独立地收敛到一组参数，并且，关键的是，（ii）避免将这些参数过度拟合到单个参与者的本地训练数据集。一旦网络被训练好，每个参与者就可以在新数据上独立地、私下地评估它，而不必与其他参与者交互。

下面，我们将详细描述系统的所有组件。表1列出了我们系统的元参数。这些参数控制协作学习过程，而不是实际正在学习的神经网络参数。

5.2本地培训

我们假设每个参与者维持一个神经网络参数的局部向量。参数服务器维护一个单独的参数向量。每个参与者可以随机初始化其本地参数，也可以从参数服务器下载其最新值。*我***栈单**（一）（全球）

然后用标准的神经网络训练每个参与者。在当地培训期间，不同参与者之间不需要进行任何协调。它们通过参数服务器间接影响彼此的训练。

图3显示了分布式选择性SGD（DSSGD）算法的伪代码。DSSGD由每个参与者独立运行，每个学习阶段包括五个步骤。首先，参与者从服务器下载一小部分参数，并用下载的值覆盖本地参数。然后，他在本地数据集上进行了一次SGD培训。这种训练可以在一系列的小批量上进行；小批量是随机选择的一组训练数据点的大小。*θd米*

在第三步中，参与者计算∆w（i），即步骤2中所有参数的变化向量，即每个参数*j*，旧值从本地SGD的最新纪元后的新值中减去。我们称之为参数梯度*j*超过一个时期的本地新元[1]∆w（i）值反映了每个参数必须改变多少才能更准确地模拟第h个参与者的本地数据集。这些信息正是其他参与者需要纳入的，以避免过度拟合。*我*

有几种方法可以选择在每个局部纪元结束时共享哪些渐变。参与者需要就标准达成一致，并在整个DSSGD中一致地使用它。我们假设在每个历元中最多可以选择一部分参数进行上传。*θu*

我们考虑两个选择标准。第一种方法是精确地选择值的分数，选取对梯度下降算法有显著贡献的大值。另一种方法是选择大于阈值的随机值子集。由于大于的渐变的数量可能小于参数的分数，因此共享的渐变将更少。这可能会减慢收敛速度，但这个选择标准更接近于我们在使用差分隐私扩展系统时使用的稀疏向量技术（见第7.2节）。*θuττθu*

在上传所选梯度∆w（i）之前，它们的值被截断到[−γ，γ]范围内。为了防止这些值泄漏过多有关训练数据的信息，还可以按照第7节的说明添加随机噪声。简言之，参与者更新∆w（i）与边界（∆w（i），γ），并在上传前添加一些随机噪声。在第7节中，我们将解释如何设置范围和随机性参数，并讨论它们对SGD的影响。5.3参数服务器

参数服务器初始化参数向量，然后处理参与者的上传和下载请求。图4显示了服务器的伪代码。当有人上传梯度时，服务器将上传的∆wj值添加到相应的全局参数中，并为每个参数更新元数据和更新计数器。为了增加最近更新的参数的权重，服务器可以周期性地将计数器乘以一个衰减因子，即：=β·stat。当参与者从服务器获得具有最大值的参数的最新值时，将在下载期间使用这些统计信息。每个参与者通过设置来决定下载这些参数中的哪一部分。**wstat公司**（全球）*国家jβ斯达θd*

5.4为什么分布式选择性SGD有效

我们的分布式SSGD实现了几乎与传统的、侵犯隐私的SGD相同的精度，原因与SGD通常成功的原因相同：学习过程的随机性。在训练过程中用全局参数的子集更新局部参数，提高了局部SGD的随机性。这对于防止本地SGD过度拟合其小型本地数据集起着至关重要的作用。单独训练时，每个参与者都很容易陷入局部最优。用在不同数据集上训练的其他参与者学习的值覆盖本地学习的参数，有助于每个参与者逃避局部最优，并使他们能够探索其他值，从而得到更精确的模型。

我们的分布式SSGD不会对哪些参数需要由其他参与者更新做出任何假设，也不会对更新率做出任何假设。由于更好的计算和吞吐量ca，一些参与者可能会经历更多的更新-

|  |
| --- |
| 选择初始全局参数。**w**（全球）  将“vector”设置为零。**斯达**  事件：参与者上传坡度∆wS。    •对于所有∈S：*j*  –集合（全局）：=w（全局）+∆wj**w**  –设置：=statj+1*国家*  事件：参与者下载参数。*θ*  •排序，设为具有最大值的元素的索引集。**静态***Iθ*  •发送给参与者。**w***我*（全球）*θ*) |

图4：服务器上DSSGD的伪代码。

能力。由于网络错误或其他故障，一些参与者可能无法上载所选参数。由于对参数服务器的异步访问，它们还可能覆盖彼此的更新。种族条件不仅不会削弱我们的分布式SSGD，事实上，它们通过增加随机性来促进它的成功。由于异步参数更新而产生的随机性对于训练精确的深层神经网络是有效的[14]。这也与正则化技术一致，即在训练过程中随机破坏神经元[47]或输入数据[52]，以避免过度拟合。

5.5参数交换协议

DSSGD不假设参与者在上传参数时遵循任何特定的时间表。在我们的评估中，我们考虑了以下场景。

使用循环法，参与者按顺序运行SSGD。每个人从服务器下载最新参数的一小部分，运行本地培训，并上传选定的梯度；下一个参与者按固定顺序跟随。使用随机顺序，参与者以随机顺序下载、学习和上传，但对服务器的访问是原子的，即参与者在读取前锁定服务器，在写入后释放锁定。使用异步时，参与者不协调。当一个参与者正在培训一组参数时，其他参与者可能会在培训结束之前在服务器上更新这些参数。

# 6        评价

6.1数据集和学习目标

我们在深度学习文献中的两个主要数据集上评估了我们的系统。第一个是格式化为32x32个图像的手写数字的MNIST数据集[31]，对其进行标准化，使数字位于图像的中心。数据集[2]由60000个训练样本和10000个测试样本组成。

第二个是从谷歌街景图片中获取的房子编号的SVHN数据集[35]。图像是32x32，有3个浮点数，包含每个像素的RGB颜色信息（我们将其转换为YUV）。每个图像都围绕一个数字居中，但许多图像的侧面都包含一些干扰因素。数据集[3]包含600000个训练图像，其中我们使用100000个训练图像，10000个作为测试示例。表2总结了我们使用的培训和测试示例的数量。

我们通过减去平均值并除以训练集中数据样本的标准差来规范化数据集。神经网络输入层的大小

|  |
| --- |
| nn.顺序{  [输入->（1）->。。。->（7）->输出]  （1） 公司名称：nn.重塑（1024个）  （2） 公司名称：nn.线性（1024->128）  （3） 公司名称：雷鲁  （4） 公司名称：nn.线性（128->64）  （5） 公司名称：nn.ReLU公司  （6） 公司名称：nn.线性（64->10）  （7） 公司名称：nn.LogSoftMax公司  } |

图5:torch7nn中用于MNIST（和SVHN，有3072个输入而不是1024个输入）的MLP架构

|  |
| --- |
| nn.顺序{  [输入->（1）->。。。->（11）->输出]  （1） 公司名称：nn.空间分辨率mm  （2） 公司名称：恩。谭  （3） 公司名称：nn.空间探索  （4） 公司名称：nn.空间分辨率mm  （5） 公司名称：谭恩  （6） 公司名称：nn.空间探索  （7） 公司名称：nn.重塑（256个）  （8） 公司名称：nn.线性（256->200）（9）：恩。谭  （10） 公司名称：nn.线性（200->10）  （11） 公司名称：nn.LogSoftMax公司  } |

图6：用于火炬7 nn中的MNIST的CNN架构

|  |
| --- |
| nn.顺序{  [输入->（1）->。。。->（13）->输出]  （1） 公司名称：nn.空间分辨率mm  （2） 公司名称：恩。谭  （3） 公司名称：nn.顺序{  [输入->（1）->（2）->（3）->输出]  （1） 公司名称：nn.正方形  （2） 公司名称：nn.空间平均池  （3） 公司名称：nn.平方米  }  （4） 公司名称：空间次牵引归一化  （5） 公司名称：nn.空间分辨率mm  （6） 公司名称：恩。谭  （7） 公司名称：nn.顺序{  [输入->（1）->（2）->（3）->输出]  （1） 公司名称：nn.正方形  （2） 公司名称：nn.空间平均池  （3） 公司名称：nn.平方米  }  （8） 公司名称：空间域规范化  （9） 公司名称：nn.重塑（1600）  （10） 公司名称：nn.线性（1600->128）  （11） 公司名称：恩。谭  （12） 公司名称：nn.线性（128->10）  （13） 公司名称：nn.LogSoftMax公司  } |

图7：用于火炬7 nn中SVHN的CNN架构

SVHN分别是1024和3072。学习的目标是将输入分类为10个可能的数字之一，因此输出层的大小为10。

6.2计算框架

我们使用Torch7[11,49]和Torch7 nn包。[4]这一受欢迎的深度学习库已经被Facebook等大型互联网公司使用和扩展

|  |  |  |
| --- | --- | --- |
|  | MNIST公司 | 斯维恩 |
| 火车 | 60,000 | 100,000 |
| 测试 | 10,000 | 10,000 |

表2：训练和测试数据集的大小

|  |  |  |
| --- | --- | --- |
|  | MNIST公司 | 斯维恩 |
| MLP公司 | 140,106 | 402,250 |
| 美国有线电视新闻网 | 105,506 | 313,546 |

表3：神经网络参数数量

6.3神经网络结构

我们使用两种流行的神经网络结构：多层感知器（MLP）和卷积神经网络（CNN）。mlp是一种前馈神经网络结构，每层的神经元与下一层的神经元完全连接。反向传播算法最初是为了学习这些网络的参数[42]。图1是MLP网络的一个示例。cnn是一种特殊的具有稀疏连通性的多层神经网络[31]。cnn广泛应用于图像和视频识别。我们在图5（MLP）和图6和图7（CNN）中提供了我们的网络架构的精确规格，所有这些都是用torch7nn包打印的。图中显示了每层中使用的激活函数（例如，Tanh表示正切双曲，ReLU表示整流函数），以及各层之间的连接。表3总结了参数的数量。

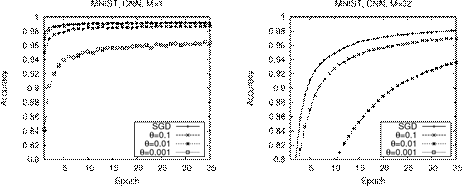
6.4实验装置

我们用三种不同的参数交换协议循环、随机顺序和异步实现了分布式SSGD。随机顺序的性能与循环法非常相似，因此被忽略。我们将所有结果与两个基线情况进行了比较。第一种是在整个数据集上集中SGD。这是一个侵犯隐私的场景，所有的训练数据被汇集到一个数据集中，网络在这个数据集中使用标准的随机梯度下降进行训练。另一种情况是独立的新加坡元。在这种情况下，参与者只根据自己的培训数据进行培训，而不需要任何协作。

我们实现了两个标准来选择要上传到参数服务器的渐变。使用最大值，每个参与者上传上一个本地训练时期的最大绝对值的梯度。使用random With threshold，参与者上传值超过阈值的随机梯度样本。下载时，每个参与者都会选择更新次数最多的参数。其他选择标准，如下载变化最大的参数，也是可行的。

在所有实验中，参数统计的衰减因子（见第5.3节）设置为0.8。我们评估了小批量（1和32）的几个设置，以及衰减率为1e−7的SGD学习率[6]（=0.001和0.01）。我们还将每个DSSGD场景中的参与者数量在30、90和150之间变化。*βαN*

我们用整个数据集的1%随机初始化每个参与者的本地训练数据集，即MNIST场景的600个数据样本和SVHN场景的1000个数据样本。在SSGD中选择共享的参数的分数取{1,0.1,0.01,0.001}中的值，即在训练MLP的MNIST的情况下，{151411402140106}参数（见表3）。要下载的参数的分数通常设置为1。*θuθd*



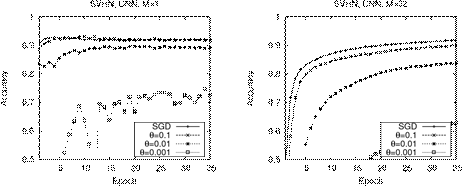


图8：不同小批量规模的SSGD的收敛性。图例显示了在每个梯度下降步骤中选择共享的参数的分数

（有1，SSGD相当于SGD）。

                             0.1新元0.01 0.001独立

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 美国有线电视新闻网 | 0.9917 | 0.9914 | 0.9871 | 0.9645 | 0.9316 |
| 斯文，CNN | 0.9299 | 0.9312 | 0.8986 | 0.7481 | 0.8182 |
|  | 新加坡元 | 0.1 | 0.01 | 0.001 | 独立 |

     970.710.980.980万里拉

     斯维恩，MLP 0.8476 0.8394 0.7833 0.6542 0.5136

表4:SSGD在CNN和MLP网络结构和不同参数共享率下实现的最大精度。结果与独立精度进行了比较。小批量大小为1。

6.5选择性SGD结果

为了证明我们的方法与传统的随机梯度下降方法相比的有效性，我们评估了在MNIST和SVHN数据集上训练卷积神经网络（CNN）时SSGD和SGD的精度。图8比较了SGD和

不同元参数值的SSGD（小批量大小和共享梯度的分数）。一般来说，参与者可以通过对校准数据集（例如，没有隐私影响的公共数据集）的培训来选择元参数的值。

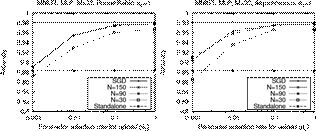
在这些梯度的后面，我们几乎可以用一个小的梯度来确认梯度的精度。此外，有选择参数共享和无选择参数共享的SGD的整体行为是相似的。将mini-batch-size设置为1可以在整个训练过程中获得很高的随机性，收敛速度非常快，但也会导致一些曲线出现波动。图8显示了历元35之前的精确轨迹；除此之外，我们还可以达到表4所示的略高精度。SSGD可以获得比SGD更高的精度，因为在每个历元只更新一小部分参数作为一种正则化技术，通过防止神经网络权值联合“记忆”训练数据（这个概念在[47]中描述）来避免过度拟合。当mini batch size设置为32时，由于在梯度下降过程中对多个训练数据点应用梯度平均值，收敛速度较慢但更平滑。

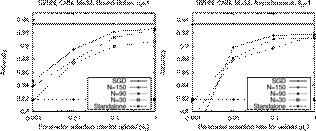
6.6分布式选择性SGD结果

Figure 9 presents the best accuracy we obtain when running DSSGD on MNIST and SVHN for different neural network architectures, parameter exchange protocols, and fractions of shared parameters. The x-axis is the fraction of shared parameters (); the y-axis is the accuracy, i.e., the fraction of correctly classified data samples in the test set. We set the download rate to 1, the learning rate to 0.001, and mini-batch size to 32.*θuθd α*

In each plot, we show the best accuracy for centralized (maximum utility, minimum privacy) and standalone (minimum utility, maximum privacy) SGD. Both are independent of the x-axis since there is no parameter sharing in either. These two scenarios are our baselines. Comparing the accuracy of distributed SSGD with the baselines reflects the tradeoff between utility and privacy. This gap depends on the network architecture and reflects that CNN takes more advantage the training data vs. MLP. Moreover, in our setting, the gap is affected by the complexity of classification and the fact each participant has 1000 data samples in the case of SVHN dataset and 600 data samples in the case of MNIST dataset.

Our results show that any cooperation, even when sharing only 1 percent of parameters, results in higher accuracy than standalone learning. Distributed SSGD using round robin parameter exchange results in the highest accuracy, almost equal to centralized SGD. The reason is its similarity to SSGD (see Figure 8). The price paid





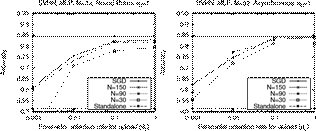


Figure 9: Accuracy of distributed SSGD on the MNIST and SVHN datasets. The legends show the number of participants. “Standalone” means that each participants trains independently on his own data; “SGD” means all training data is pooled for centralized training.

for this accuracy is the speed of learning, which is determined by the slowest participant. The round robin protocol is suitable for scenarios where all participants have similar computation capacity, e.g., biomedical research institutions with dedicated SSGD servers. We do not make any assumptions, however, about how local SGD should run. For example, it can be executed on parallel GPUs to speed up the process. Asynchronous parameter exchange protocol can produce accurate models, too. The key to its success is the inherent randomness and thus high stochasticity of gradient descent, which prevents overfitting. In our implementation, we assumed that each participant may lag behind others and download an outdated set of parameters (those from the previous epoch) with probability 0.5. The promising accuracy of this protocol indicates that DSSGD should work well even with unreliable (e.g., mobile) networks.

            SVHN, CNN, M=1, Round Robin, θ=1 SVHN, CNN, M=1, Round Robin, θ=1, dd

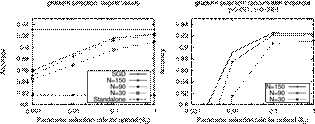


Figure 10: Accuracy of DSSGD for different gradient selection criteria.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

We also observe that the number of participants has a lower impact on accuracy than the percentage of shared parameters. This indicates that distributed SSGD does not require very many participants to boost the accuracy.

The number of participants, the rate of parameter updates, and the parameter exchange schedule all influence the communication cost of distributed SSGD. For example, training an MLP model on MNIST dataset with 90 participants with the parameter upload rate of 10% in round-robin schedule requires the server to support 90 × 14010 × 32 = 38.5 Megabytes of parameter uploads during each epoch. With the parameter download rate of 100%, the server needs to support 385 Megabytes of download during each epoch.

All of the above results were obtained assuming each participant shares his largest gradients with the other participants. The other method is to randomly sample from the gradients whose values are above a threshold. Figure 10 compares the accuracy of DSSGD with these two criteria for both MNIST and SVHN datasets. In the “random with threshold” scenario, we first truncate gradient values ∆w into the [−0.001,0.001] range, then go through them in random order, and upload if abs(∆wj) ≥ τ. The neural network architecture (CNN), learning rate (= 0.001), mini-batch size (= 1), and exchange protocol (Round Robin) are the same in all experiments. In the “random with threshold” scenario, fewer than the fraction of gradients may be uploaded, thus accuracy is sometimes lower. To find an effective value of the threshold , participants need to run DSSGD on a public calibration dataset.*α M θu τ*

Figure 11 shows the convergence of DSSGD for different datasets, learning rates, and number of participants. The upload rate is 0.1, download rate is 1, mini-batch size is 32, the parameter exchange protocol is round robin, and the gradient selection criterion is the largest values. These results show that higher learning rate indeed results in faster convergence to maximum accuracy regardless of the number of participants. Therefore, the distributed and selective nature of DSSGD does not change the overall behavior of the gradient descent algorithm.*θu θd*

# 7        Privacy

|  |
| --- |
| MNIST, CNN, M=32, Round Robin, θ=0.1, θ=1 SVHN, CNN, M=32, Round Robin, θ=0.1, θ=1udud   10                                              20                 30   60    70                          80 10                20                 30   60    70                          80                                        Epoch Epoch  Figure 11: Convergence of DSSGD. The legends show the number of participants and the learning rate .*N α* |

Our system aims to address several privacy threats associated with deep learning. First, in conventional deep learning, all training data is revealed to a third party (typically, the company performing the learning) and individuals who contributed the data do not have any control over it. Their sensitive information may leak to the company itself, to attackers who compromised the company&apos;s data storage, and to law enforcement and intelligence outfits who can access the data via legal and extra-legal means.

Second, in conventional deep learning, data owners have no control over the learning objective (i.e., which model is being trained) and thus no control or even knowledge of what is being inferred from their data. For example, an individual might be willing to share her image for face recognition but not for inferring her location from the background objects.

Third, in conventional deep learning, the learned model is not available directly to data owners. If they want to use it, they must reveal their inputs to the company holding the model, thus exposing them to the same privacy risks as the training data.

Our privacy-preserving deep learning system addresses all of these concerns and aims to protect privacy of the training data, ensure public knowledge of the learning objective, and protect privacy of the data to which the learned model is applied, as well as privacy of the model&apos;s output.

The scenarios we consider—for example, collaborative learning of image recognition models between medical institutions—involve participants who are not actively malicious. Therefore, it is reasonable to assume a “passive” adversary model, in which the participants execute the protocol as designed but may attempt to learn or infer sensitive information from other participants&apos; data.

7.1 Preventing direct leakage

*While training the model.* Unlike conventional deep learning, in our system participants do not reveal their training datasets to anyone, thus ensuring strong privacy of their data. The size and dynamics of local datasets are confidential, and different data samples can be used in each round of SSGD. The participants can also delete their training data at any time.

*While using the model.* All participants learn the model and thus can use it locally and privately, without any communication with other participants and without revealing the input data or the model&apos;s output to anyone. Therefore, in contrast to conventional deep learning, there is absolutely no leakage while using the model.

7.2 Preventing indirect leakage

Participants in our system may indirectly reveal some information about their training datasets via public updates to a fraction of the neural-network parameters during training. Each participant fully controls which gradients to share and may decide not to share particularly sensitive ones. Furthermore, each participant shares only a tiny fraction of his gradients: as we show, even sharing as few as 1% still results in significantly better accuracy than learning just on local data. Even so, we use differential privacy to ensure that parameter updates do not leak too much information about any individual point in the training dataset.

*Differential privacy.* Our application of differential privacy to parameter updates is inspired by recent work on privacy-preserving empirical risk minimization [4]. In a nutshell, a computation is differentially private if the probability of producing a given output does not depend very much on whether a particular data point is included in the input dataset [19]. For any two datasets and differing in a single item and any output of function ,*D D*0 *O f*

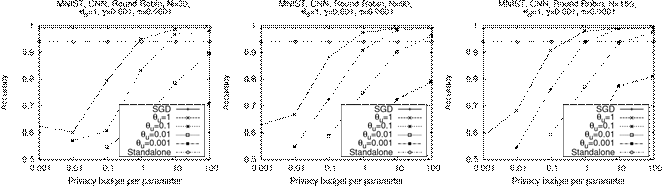
                 Pr*.* (2)

The parameter controls the tradeoff between the accuracy of the differentially private and how much information it leaks.*f*

In our case, computes parameter gradients and selects which of them to share with other participants. There are two sources of potential leakage: how gradients are selected for sharing and the actual values of the shared gradients. To mitigate both types of leakage, we use the sparse vector technique [20, 25] to (i) randomly select a small subset of gradients whose values are above a threshold, and to (ii) share perturbed values of the selected gradients, all under a consistent differentially private mechanism. This is equivalent to releasing the responses to queries whose value is above a publicly known threshold.*f*

Let the total privacy budget for each epoch of DSSGD for each participant. We split this budget into *c* parts, where is the total number of gradients that we can upload at each epoch (i.e., = θu|∆w|). The budget for each potential upload is then divided into two parts. The first will be spent on checking whether the gradient ∆wj(i) of a randomly chosen parameter is above the threshold . The second will be spent on actually releasing (uploading) the gradient if it is above the threshold. We use the Laplacian mechanism to add noise during selection and upload according to the allocated privacy budgets. The noise depends on the privacy budget as well as the sensitivity of the gradient for each parameter. In the following, we assume the same sensitivity ∆f for all parameters, but this is not a requirement, and different parameters may have different sensitivities.*c c j τ*

Figure 12 presents the pseudocode of differentially private DSSGD. To split , we follow [20]. is devoted to the selection, where part of it is spent on random noise *rw* and the other part is spent on random noise *rτ*. The remaining is devoted to the released value. Note that *rτ* is not re-generated after failed threshold checks. This



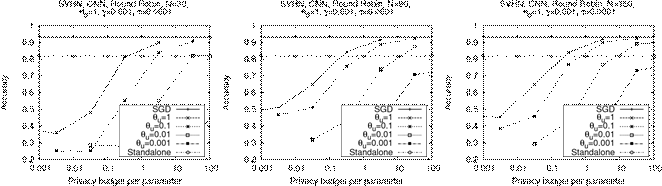


Figure 13: Accuracy of differentially private DSSGD for different datasets, number of participants, fraction of uploaded gradients, and privacy budget. Each subfigure plots the per-parameter privacy budget (i.e., ) versus accuracy. The accuracy of SGD and Standalone are plotted for comparison.

|  |
| --- |
| be the total privacy budget for one epoch of participant running DSSGD, and let ∆f be the sensitivity of each gradient*i*  • Let = θu|∆w| be the maximum number of gradients that can be uploaded in one epoch*c*    1.   Generate fresh random noise ∼ Lap(*rτ*  2.   Randomly select a gradient  3.   Generate fresh random noise ∼ Lap(2*rw*  4.   If abs(bound , then  (a) Generate fresh random noise Lap(  (b) Upload bound to the parameter server  (c) Chargeto the privacy budget  (d) If number of uploaded gradients is equal to , then Halt Else Goto Step 1*c*  5.   Else Goto Step 2 |

Figure 12: Pseudocode of differentially private DSSGD for participant using the sparse vector technique*i*

ensures not only that all shared gradients are differentially private, but also that the privacy “penalty” is not paid for gradients that are too small to be shared with other participants.

*Estimating sensitivity.* The sensitivity of a function determines how much random noise needs to be added to its output to achieve differential privacy. The (global) sensitivity of is*f*

                               ∆f = maxD,D0||f(D) − f(D0)||. (3)

Estimating the true sensitivity of stochastic gradient descent is challenging. Instead, we modify the function so that its output stays within fixed, input-independent bounds and use these bounds to estimate sensitivity: that&apos;s the bound function that enforces a [−γ,γ] range on gradient values that may be shared with other participants (Section 5). This approach may reduce accuracy (although in our case the effect is negligible), but privacy is guaranteed. A similar technique was previously used to enforce privacy of MapReduce computations with untrusted mappers [40].

Limiting the range of values that parameters and gradients can take even improves the training process by helping to avoid overfitting. Some existing regularization techniques already force a bound on the norm of the parameters. Max-norm has been used for collaborative filtering [46] and deep learning [47]. Moreover, gradient values truncated into the [−γ,γ] range indicate the direction and magnitude of moves during gradient descent. Therefore, small values of (implying smaller sensitivity and thus smaller noise and higher accuracy) would influence the learning rate of the algorithm but not whether the optimal solution is achievable. Furthermore, as gradients of multiple participants are aggregated, the gradient descent algorithm can traverse through local optima. We discuss the effect of perturbation on distributed selective SGD below.*γ*

The meta-parameter is set independently of the training data and thus cannot leak any sensitive information. It can be set by training on a calibration dataset with inputs that are similar to the real inputs but are not privacy-sensitive. We then (over-)estimate the sensitivity of our algorithm as 2γ and truncate the uploaded gradients into the [−γ,γ] range. This helps mitigate the detrimental effect of very large noise values on the training process.*γ*

We expect that global sensitivity estimates can be significantly reduced, resulting in higher accuracy, by ensuring that the norm of all gradients is bounded for each update—either globally, or locally, e.g., across all edges leading to a given neural-network node. In fact, the latter kind of norm-bounding is a known regularization technique. We plan to investigate applications of norm-bounding to differentially private deep learning in future work.

7.3 Experimental results

We evaluate the effect of different values of (the differential privacy parameter), (the number of participants), and (the fraction of uploaded gradients) on the accuracy of neural networks trained using distributed selective SGD with differential privacy.*N θu*

Figure 13 shows the results and compares them with standalone learning and centralized SGD. We set the bound to 0.001 and the threshold to 0.0001. As expected, smaller values (i.e., stronger differential privacy guarantees) result in lower accuracy. However, with many participants and when participants share a large fraction of their gradients, the accuracy of differentially private DSSGD is better than the accuracy of standalone training.*γ τ*

7.4 Oblivious parameter server

Regardless of whether the parameter server is trusted, the privacy guarantees of training data separation and differential privacy still hold. However, to prevent a curious server from linking the updates of each participant, it is possible to design a parameter server that is oblivious to uploaders&apos; identities. For example, participants can anonymously authenticate themselves and the gradients they upload [7]. Scalable anonymous communication protocols with provable security can be used to hide participants&apos; identities [12,54].

The independence of parameters from each other in distributed SSGD, which is inherent to the underlying stochastic gradient descent algorithm, also enables a completely distributed implementation of the parameter storage system where each participant is responsible for a random subset of the parameters. We leave the detailed design of this scheme to future work.

# 8        Conclusions

This work is the first step in bringing privacy to a machine learning approach that is revolutionizing AI. We proposed a new distributed training technique, based on selective stochastic gradient descent. Our methodology works for any type of neural network and preserves privacy of participants&apos; training data without sacrificing the accuracy of the resulting models. Therefore, it can help bring the benefits of deep learning to domains where data owners are precluded from sharing their data by confidentiality concerns.

*Acknowledgments.* We are grateful to Adam Smith for explaining how to apply the sparse vector technique and other differential privacy mechanisms in our setting. This work was partially supported by the NSF grants 1223396 and 1408944, NIH grant R01 LM011028-01 from the National Library of Medicine, and Swiss

National Science Foundation postdoctoral fellowship to Reza Shokri.

# 9        References

[1] A. Agarwal, O. Chapelle, M. Dudík, and J. Langford. A reliable effective terascale linear learning system. JMLR, 15(1):1111–1133, 2014.

[2] M. Avriel. Nonlinear Programming: Analysis and Methods. Courier Corporation, 2003.

[3] M. Barni, P. Failla, R. Lazzeretti, A. Sadeghi, and T. Schneider. Privacy-preserving ECG classification with branching programs and neural networks. Trans. Info.

*Forensics and Security*, 6(2):452–468, 2011.

[4] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In FOCS, 2014.

[5] Y. Bengio. Learning deep architectures for AI. Foundations and trends in machine learning, 2(1):1–127, 2009.

[6] J. Bos, K. Lauter, and M. Naehrig. Private predictive analysis on encrypted medical data. J. Biomed. Informatics, 50:234–243, 2014.

[7] J. Camenisch, S. Hohenberger, M. Kohlweiss, A. Lysyanskaya, and M. Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In CCS, 2006.

[8] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In NIPS, 2009.

[9] K. Chaudhuri, C. Monteleoni, and A. Sarwate. Differentially private empirical risk minimization. JMLR, 12:1069–1109, 2011.

[10] K. Chaudhuri, A. Sarwate, and K. Sinha. A near-optimal algorithm for differentially-private principal components. JMLR, 14(1):2905–2943, 2013.

[11] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A Matlab-like environment for machine learning. In BigLearn, NIPS Workshop, 2011.

[12] H. Corrigan-Gibbs and B. Ford. Dissent: Accountable anonymous group messaging. In CCS, 2010.

[13] A. A. Cruz-Roa, J. E. A. Ovalle, A. Madabhushi, and F. A. G. Osorio. A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection. In MICCAI, 2013.

[14] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. Le, et al. Large scale distributed deep networks. In NIPS, 2012.

[15] O. Denas and J. Taylor. Deep modeling of gene expression regulation in an erythropoiesis model. In Representation Learning, ICML Workshop, 2013.

[16] L. Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Trans. Signal and Information Processing, 3, 2014.

[17] W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In SDM, volume 4, pages 222–233, 2004.

[18] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization.

*JMLR*, 12:2121–2159, 2011.

[19] C. Dwork. Differential privacy. In Encyclopedia of Cryptography and Security, pages 338–340. Springer, 2011.

[20] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. Theoretical Computer Science, 9(3-4):211–407, 2013.

[21] C. Dwork, G. Rothblum, and S. Vadhan. Boosting and differential privacy. In FOCS, 2010.

[22] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber. Using deep learning to enhance cancer diagnosis and classification. In WHEALTH, 2013.

[23] A. Graves, A.-R. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In ICASSP, 2013.

[24] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deepspeech: Scaling up end-to-end speech recognition. arXiv:1412.5567, 2014.

[25] M. Hardt and G. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In FOCS, 2010.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. arXiv:1502.01852, 2015.

[27] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups.

*Signal Processing Magazine*, 29(6):82–97, 2012.

[28] G. Jagannathan and R. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In KDD, 2005.

[29] P. Jain, V. Kulkarni, A. Thakurta, and O. Williams. To drop or not to drop: Robustness, consistency and differential privacy properties of dropout. arXiv:1503.02031, 2015.

[30] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.

Gradient-based learning applied to document recognition.

*Proc. of the IEEE*, 86(11):2278–2324, 1998.

[32] M. Liang, Z. Li, T. Chen, and J. Zeng. Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach. Trans. Comput. Biology and Bioinformatics, 12(4):928 – 937, 2015.

[33] Y. Lindell and B. Pinkas. Privacy preserving data mining. In CRYPTO, 2000.

[34] K. P. Murphy. Machine Learning: A Probabilistic Perspective. MIT press, 2012.

[35] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In Deep Learning and Unsupervised Feature Learning, NIPS Workshop, 2011.

[36] M. Pathak and B. Raj. Privacy-preserving speaker verification and identification using gaussian mixture models. Trans. Audio, Speech, and Language Processing, 21(2):397–406, 2013.

[37] M. Pathak, S. Rane, and B. Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In NIPS, 2010.

[38] M. Pathak, S. Rane, W. Sun, and B. Raj. Privacy preserving probabilistic inference with Hidden Markov Models. In ICASSP, 2011.

[39] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In NIPS, 2011.

[40] I. Roy, S. T. Setty, A. Kilzer, V. Shmatikov, and E. Witchel. Airavat: Security and privacy for MapReduce. In NSDI, 2010.

[41] B. Rubinstein, P. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. J. Privacy and Confidentiality, 4(1):4, 2012.

[42] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In Neurocomputing:

*Foundations of research*, pages 673–695. MIT Press, 1988.

[43] A. Sarwate and K. Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. Signal Processing Magazine, 30(5):86–94, 2013.

[44] D. Shultz. When your voice betrays you. Science, 347(6221), 2015.

[45] P. Simard, D. Steinkraus, and J. Platt. Best practices for convolutional neural networks applied to visual document analysis. In Document Analysis and Recognition, 2013.

[46] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In Learning Theory, pages 545–560. Springer, 2005.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 15(1):1929–1958, 2014.

[48] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In CVPR, 2014.

[49] Torch7. A scientific computing framework for LuaJIT

(torch.ch).

[50] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In KDD, 2002.

[51] J. Vaidya, M. Kantarcıoglu, and C. Clifton.˘ Privacy-preserving naive bayes classification. VLDB, 17(4):879–898, 2008.

[52] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In ICML, 2008.

[53] M. Wainwright, M. Jordan, and J. Duchi. Privacy aware learning. In NIPS, 2012.

[54] D. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In OSDI, 2012.

[55] P. Xie, M. Bilenko, T. Finley, R. Gilad-Bachrach, K. Lauter, and M. Naehrig. Crypto-nets: Neural networks over encrypted data. arXiv:1412.6181, 2014.

[56] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, et al. The human splicing code reveals new insights into the genetic determinants of disease.

*Science*, 347(6218), 2015.

[57] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: Regression analysis under differential privacy. VLDB, 5(11):1364–1375, 2012.

[58] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In ICML, 2004.

[59] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola.

Parallelized stochastic gradient descent. In NIPS, 2010.

[[1]](" \l "_ftnref1" \o ") Usually gradient refers to the change in a parameter after a single mini-batch training, but here we generalize it to one epoch of training over several mini-batches.

[[2]](" \l "_ftnref2" \o ") http://yann.lecun.com/exdb/mnist

[[3]](" \l "_ftnref3" \o ") http://ufldl.stanford.edu/housenumbers

[[4]](" \l "_ftnref4" \o ") https://github.com/torch/nn

[[5]](" \l "_ftnref5" \o ") https://github.com/facebook/fblualib

[[6]](" \l "_ftnref6" \o ") The learning rate and its decay rate are applied during local SGD when training over a new mini-batch. The parameter server does not apply it to the uploaded gradients.