On Lightweight Privacy-Preserving Collaborative Learning for Internet-of-Things Objects

Linshan Jiang
School of Computer Science and Engineering
Nanyang Technological University
linshan001@e.ntu.edu.sg

Rui Tan
School of Computer Science and Engineering
Nanyang Technological University
tanrui@ntu.edu.sg

Xin Lou
Advanced Digital Sciences Center
Illinois at Singapore Pte Ltd
lou.xin@adsc-create.edu.sg

Guosheng Lin
School of Computer Science and Engineering
Nanyang Technological University
gslin@ntu.edu.sg

ABSTRACT

The Internet of Things (IoT) will be a main data generation infrastructure for achieving better system intelligence. This paper considers the design and implementation of a practical privacy-preserving collaborative learning scheme, in which a curious learning coordinator trains a better machine learning model based on the data samples contributed by a number of IoT objects, while the confidentiality of the raw forms of the training data is protected against the coordinator. Existing distributed machine learning and data encryption approaches incur significant computation and communication overhead, rendering them ill-suited for resource-constrained IoT objects. We study an approach that applies independent Gaussian random projection at each IoT object to obfuscate data and trains a deep neural network at the coordinator based on the projected data from the IoT objects. This approach introduces light computation overhead to the IoT objects and moves most workload to the coordinator that can have sufficient computing resources. Although the independent projections performed by the IoT objects address the potential collusion between the curious coordinator and some compromised IoT objects, they significantly increase the complexity of the projected data. In this paper, we leverage the superior learning capability of deep learning in capturing sophisticated patterns to maintain good learning performance. Extensive comparative evaluation shows that this approach outperforms other lightweight approaches that apply additive noisification for differential privacy and/or support vector machines for learning in the applications with light data pattern complexities.

CCS CONCEPTS

• **Computer systems organization → Sensor networks**; • **Computing methodologies → Supervised learning**; • **Security and privacy** → *Domain-specific security and privacy architectures.*

KEYWORDS

Internet of Things, collaborative learning, privacy

ACM Reference Format:
Linshan Jiang, Rui Tan, Xin Lou, and Guosheng Lin. 2019. On Lightweight Privacy-Preserving Collaborative Learning for Internet-of-Things Objects. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3302505.3310070

IoTDI '19, April 15–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6283-2/19/04…$15.00
https://doi.org/10.1145/3302505.3310070

1 INTRODUCTION

The recent research advances of machine learning have led to performance breakthroughs of various tasks such as image classification, speech recognition, and language understanding. The drastically increasing amount of data generated by the Internet of Things (IoT) will further foster machine learning performance and enable new applications in various domains. In particular, the *collaborative learning*, which builds a machine learning model (e.g., a supervised classifier) based on the training data contributed by many *participants*, is a desirable and empowering paradigm for smarter IoT systems. By leveraging on the much increased volume of training data and coverage of data patterns, collaborative learning will approach the intelligence of a crowd and improve the learning performance beyond that achieved by any single participant alone. Moreover, a resource-rich learning *coordinator* (e.g., a desktop-class edge device or a cloud computing service) allows the execution of advanced, compute-intensive machine learning algorithms to capture deeper structures in the aggregated data, whereas the participants (e.g., IoT objects) are often resource-constrained and incompetent for intensive computation. By contributing training data, the individual participants will enjoy the improved machine intelligence in return.

However, the data contributed by the participants may contain privacy-sensitive information. On Internet, various online services (e.g., webmail and social networking) generally collect and analyze the user data in the raw forms. In this scheme, the users risk privacy leak due to potential inadvertent actions by the service providers and/or targeted cyber-attacks from the external. This risk has been evidenced by several recent large-scale user privacy leak incidents [7, 34, 38]. Data anonymization can mitigate the concern; but it is inadequate for privacy preservation, because cross correlations among different databases may be used to re-identify

data [33]. Moreover, the correlations between different properties of anonymous individuals (e.g., race, income, political views, etc.) can be exploited to identify an interested user group to target for advertisement and advocacy. In the coming era of IoT with many smart objects penetrating into our private space and time, the current raw data collection approach will only raise large privacy concerns and may potentially violate relevant laws such as the recent General Data Protection Regulation in European Union. Therefore, to be successful, the IoT-driven collaborative learning applications must be privacy-preserving.

Privacy-preserving collaborative learning (PPCL) has received increasing research recently under the enterprise settings, where the participants are entities with rich computing resources. The existing approaches can be broadly classified into two categories. The first category of approaches [9, 23, 31, 36, 40] follows the distributed machine learning (DML) scheme, such that the participants do not need to transmit the training data to the coordinator. The recently proposed *federated learning* [31] is a type of DML. In the second category of approaches [12, 19, 22], each participant applies the homomorphic encryption on the data before being transmitted to the coordinator such that the training and inference can be performed on ciphertexts. However, for resource-constrained IoT objects, these DML and data encryption approaches incur significant and even prohibited compute overhead. The DML will require the participants to execute machine learning algorithms to train local models, which is often too compute-intensive for IoT objects. Moreover, the iterative communication rounds of DML introduce large communication overhead. Currently, the homomorphic encryption algorithms are still too compute-intensive to be realistic for resource-constrained devices (cf. §6). Therefore, these existing approaches are ill-suited or unpractical for the resource-constrained smart objects beneath the IoT edge.

In this paper, we study the design and implementation of a PPCL approach that is lightweight for resource-constrained participants, while keeping privacy-preserving against a honest-but-curious learning coordinator. The coordinator can be a cloud server or a resource-rich edge device, e.g., access points, base stations, network routers, etc. We propose to apply (1) multiplicative *Gaussian random projection* (GRP) at the resource-constrained IoT objects to obfuscate the contributed training data and (2) *deep learning* at the coordinator to address the much increased complexity of the data patterns due to the GRP. Specifically, each participant uses a private, time-invariant but randomly generated Gaussian matrix to project each plaintext training data vector and transmits the result to the coordinator. GRP gives several privacy preservation properties of (1) the computational difficulty for the coordinator to reconstruct the plaintext without knowing the Gaussian matrix [30, 37], and (2) quantifiable plaintext reconstruction error bounds even if the coordinator obtains the Gaussian matrix [30]. From a system perspective, GRP is computationally lightweight and does not increase the data volume. Thus, GRP is a practical privacy protection method suitable for resource-constrained IoT objects. Regarding GRP's impact on the design of the machine learning algorithms, the random projection can be viewed as a process of mapping the original data vectors to some domain in which the data vectors in different classes are less separatable. If the original data vectors are readily separatable (that is, they are features), the inverse of the Gaussian matrix can be considered as a linear feature extraction matrix. With the deep learning's unsupervised feature learning capability, this inverse matrix can be implicitly captured by the trained deep model. Thus, we conjecture that the randomly projected training samples can still be used by the coordinator to build the deep model for classification.

To achieve robustness of the privacy preservation against the collusion between any single participant and the curious learning coordinator, each participant should generate its own Gaussian matrix independently. However, this presents a main challenge on the PPCL system's scalability with respect to the number of participants (denoted by N). Specifically, assuming that the training data samples for each class are horizontally distributed among the participants, the number of data patterns for a class will increase from one in the plaintext domain to N in the projection data domain. This increased pattern complexity is to be addressed by the strong learning capability of deep learning. Thus, in the proposed PPCL approach, most of the computational workload is offloaded to the resourceful coordinator at the edge or in the cloud, unlike the existing DML and homomorphic encryption approaches that introduce significant or prohibitive compute overhead to the smart objects beneath the IoT edge.

To understand the effectiveness of the GRP approach and its scalability with the number of participants and the pattern complexity of the training data, we conduct extensive evaluation to compare GRP with several other lightweight PPCL approaches. The evaluation is based on two example applications with low and moderate pattern complexities, i.e., handwritten digit recognition and spam e-mail detection. The baseline approaches include various combinations between (1) multiplicative GRP versus additive noisification for differential privacy (DP) at the participants, and (2) deep neural networks (DNNs), including the multilayer perceptron (MLP) and convolutional neural network (CNN), versus support vector machines (SVMs) at the coordinator. The results show that, for the two example applications, the proposed GRP-DNN approach can support up to hundreds of participants without sacrificing the learning performance much, whereas the GRP-SVM approach may fail to capture the projected data patterns and the performance of the DP-DNN approach is susceptible to additive noisification. The results of this paper suggest that GRP-DNN is a practical PPCL approach for resource-constrained IoT objects observing data with low- or moderate-complexity patterns.

We also implement GRP-DNN, Crowd-ML [23] (a federated learning approach based on shallow learning), and CryptoNets [19] (a homomorphic encryption approach) on a testbed of 14 IoT devices. Experiments show that, compared with GRP-DNN, Crowd-ML incurs 350x compute overhead and 3.5x communication overhead to each IoT device. Deep federated learning will only incur more compute overhead. CryptoNets incurs 2.6 million times higher compute overhead to the IoT device, compared with GRP.

The remainder of this paper is organized as follows. §2 introduces the background and preliminaries. §3 reviews related work. §4 states the problem and overviews our approach. §5 presents the learning performance evaluation for various lightweight PPCL approaches. §6 presents the benchmark results of GRP-DNN, Crowd-ML, and CryptoNets on the testbed. §7 concludes this paper.

2 BACKGROUND AND PRELIMINARIES

2.1 Supervised Collaborative Learning

Supervised machine learning has two phases, i.e., the learning phase and the classification phase. We now formally describe the collaborative learning scheme. The trained classifier, denoted by $h(\mathbf{x}|\boldsymbol{\theta})$, can classify a d-dimensional data vector $\mathbf{x} \in \mathbb{R}^d$ to be one of a finite number of classes represented by a set C. The learning process determines the parameter $\boldsymbol{\theta}$ based on the training data. Let N denote the number of participants of the collaborative learning. Let $\mathcal{D}_i$ denote a set of M_i training data samples generated by the participant i, i.e., $\mathcal{D}_i = \{(\mathbf{x}_{i,j}, y_{i,j}) | j \in [1, M_i], y_{i,j} \in C\}$, where $\mathbf{x}_{i,j}$ is the training data vector and $y_{i,j}$ is the corresponding class label. For a training data sample $(\mathbf{x}, y)$, denote by $l(h(\mathbf{x}|\boldsymbol{\theta}), y)$ the loss function. The collaborative learning solves the following problem to determine the optimal classifier parameter denoted by $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \frac{1}{M_i} \sum_{j=1}^{M_i} l\left(h\left(\mathbf{x}_{i,j}|\boldsymbol{\theta}\right), y_{i,j}\right) + \lambda \|\boldsymbol{\theta}\|^2, \qquad (1)$$

where the $\lambda\|\boldsymbol{\theta}\|^2$ is a regularization term. With $\boldsymbol{\theta}^*$, the classification for a test data sample $\mathbf{x}$ is to compute $h(\mathbf{x}|\boldsymbol{\theta}^*)$.

A simple approach is to collect all the plaintext training data to the coordinator and solve Eq. (1). However, this approach raises the concern of privacy breach, as the raw training data are generally privacy-sensitive. The problem of solving Eq. (1) without threatening the participants' privacy contained in $\mathcal{D}_i$, $i = 1, \ldots, N$, is called PPCL. Existing approaches to PPCL will be reviewed in §3.

2.2 Random Gaussian Projection (GRP)

This section reviews two properties of GRP. Let $\mathbf{R} \in \mathbb{R}^{k \times d}$ represent a random Gaussian matrix, i.e., each element in $\mathbf{R}$ is drawn independently from the normal distribution $\mathcal{N}(0, \sigma^2)$. GRP has the following two properties [30]:

Property 1. *For data vectors $\mathbf{x}_1$, $\mathbf{x}_2$ and their projections $\mathbf{y}_1 = \frac{1}{\sqrt{k}\sigma}\mathbf{R}\mathbf{x}_1$, $\mathbf{y}_2 = \frac{1}{\sqrt{k}\sigma}\mathbf{R}\mathbf{x}_2$, the dot product and Euclidean distance between $\mathbf{y}_1$ and $\mathbf{y}_2$ are unbiased estimates of those between $\mathbf{x}_1$ and $\mathbf{x}_2$, i.e., $\mathbb{E}\left[\mathbf{y}_1^\intercal \mathbf{y}_2\right] = \mathbf{x}_1^\intercal \mathbf{x}_2$ and $\mathbb{E}\left[\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2\right] = \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$. The estimation error bounds are $\mathrm{Var}[\mathbf{y}_1^\intercal \mathbf{y}_2] \leq \frac{2}{k}$ and $\mathrm{Var}\left[\|\mathbf{y}_1 - \mathbf{y}_2\|_2^2\right] \leq \frac{32}{k}$.*

Property 2. *Given a Gaussian matrix instance $\mathbf{R} \in \mathbb{R}^{k \times d}$ where $k < d$ and the projection $\mathbf{y} = \frac{1}{\sqrt{k}\sigma}\mathbf{R}\mathbf{x}$, the minimum norm estimate of $\mathbf{x}$, denoted by $\hat{\mathbf{x}}$, is an unbiased estimate of $\mathbf{x}$, i.e., $\mathbb{E}[\hat{\mathbf{x}}] = \mathbf{x}$. The estimation error for the ith element of $\mathbf{x}$ is $\mathrm{Var}[x_i] = \frac{2}{k}x_i^2 + \frac{1}{k}\sum_{j,j\neq i} x_j^2$.*

Based on Property 1, the study [30] shows that a trained SVM classifier can be transferred to classify the projected data. In a recent study [45], a random projection layer that can be implemented by GRP is added to an MLP for dimension reduction. Such design is also based on Property 1. However, the studies [30, 45] do not address collaborative learning and privacy.

The estimation error given by Property 2 will be used in the later sections of this paper to measure the degree of privacy protection provided by our proposed approach.

3 RELATED WORK

Existing PPCL approaches can be classified into two categories, i.e., *distributed machine learning* (§3.1) and *training data encryption or obfuscation* (§3.2). §3.3 reviews other related work.

3.1 Distributed Machine Learning (DML)

DML approaches exploit the computing capability of the participants to solve Eq. (1) using some variant of stochastic gradient descent (SGD) in a distributed manner. During the learning process, the training data samples are not transmitted. The studies [23, 31, 32, 40] share the similar idea of exchanging gradients and classifier parameters among the participants, which is coordinated by the coordinator. Specifically, in the Crowd-ML approach [23], a participant checks out the global classifier parameters $\boldsymbol{\theta}$ from the coordinator and computes the gradients using its own training data. Then, the participants transmit the gradients to the coordinator that will update $\boldsymbol{\theta}$. In [40], each participant trains a local deep model using SGD and uploads a selected portion of gradients to the coordinator for combining. Then, each participant downloads a selected portion of the global gradients to update its local deep model. As the exchanged gradients and classifier parameters may still contain privacy, the approaches [23, 40] add random noises to the exchanged values for differential privacy [17]. In the *federated learning* scheme [31], the coordinator periodically pulls the deep models trained by the participants locally based on their training data and returns an average deep model to the participants. In [32], the participant adds random noises to the deep model parameters before being sent to the coordinator for privacy protection in the federated learning process.

However, the above DML approaches have the following limitations. First, the local training introduces computation overhead to the participants. Training a DNN locally may be infeasible for resource-constrained IoT objects. Second, DML approaches often require many iterations for the learning algorithm to converge, which may incur a high volume of data traffic between each participant and the coordinator. In §6, we will show this by comparing the Crowd-ML [23] and our proposed approach. Third, as shown recently in [24], generative adversarial networks can generate prototypical training data samples based on the exchanged gradients and model parameters, weakening the privacy preservation claimed in [31, 40]. In [36] and [9], homomorphic encryption and secure aggregation have been applied to enhance the privacy preservation of the approach in [40] and the federated learning in [31], respectively. With these enhancements, only the encrypted gradients [36] and aggregate model update [9] are revealed to the honest-but-curious coordinator. However, these privacy enhancements further increase the computation overhead of each participant, making it more unsuitable for resource-constrained IoT objects.

3.2 Training Data Encryption/Obfuscation

Different from the DML approaches that transmit classifier's parameters, the approaches [22, 29, 39] transmit the encrypted or obfuscated training data to the coordinator to solve Eq. (1). The approach proposed in this paper also belongs to this category. In the following, we review each of [22, 29, 39] and then discuss our new design to overcome their shortcomings.

In [22], homomorphic encryption is integrated with a Linear Means classifier and Fisher's Linear Discriminant classifier. During both the training and classification phases, the participant transmits the homomorphically encrypted data vector to the coordinator. However, homomorphic encryption results in intensive computation and increased volume of data transmissions (cf. §6). Thus, although the homomorphic encryption approach provides provable confidentiality protection, it is inefficient and unrealizable on many resource-constrained IoT platforms.

To reduce the computation and communication overheads, Liu et al. [29] propose a data obfuscation approach based on random projection. Specifically, the participant i independently generates a random matrix $\mathbf{R}_i$ and transmits the obfuscated training dataset $\{(\mathbf{R}_i\mathbf{x}_{i,j}, y_{i,j})|j \in [1, M_i]\}$ to the coordinator. However, different from Property 1 in §2.2 that requires the same projection matrix, the approach [29] uses distinct projection matrices for different participants and thus no longer preserves the Euclidean distance, i.e., $\|\mathbf{R}_u\mathbf{x}_{u,p} - \mathbf{R}_v\mathbf{x}_{v,q}\| \neq \|\mathbf{x}_{u,p} - \mathbf{x}_{v,q}\|$. This will result in poor training performance for distance-based classifiers, such as k-nearest neighbors and SVM. To address this issue, the study [29] designs a regression phase before the learning phase. Specifically, the coordinator sends a number of *public data vectors* $\{\mathbf{z}_k|k = 1, 2, \ldots\}$ to all participants and the participant i returns the projected data $\{\mathbf{R}_i\mathbf{z}_k|k = 1, 2, \ldots\}$. Based on the original and projected public data vectors, a regress function $f_{uv}(\cdot, \cdot)$ for each participant pair (u, v) is learned such that $f_{uv}(\mathbf{R}_u\mathbf{x}_{u,p}, \mathbf{R}_v\mathbf{x}_{v,q}) \simeq \|\mathbf{x}_{u,p} - \mathbf{x}_{v,q}\|$. As a result, the distance-based classifiers can be trained in the domain of obfuscated data by using the learned regress functions to compute distances during the training phase.

However, the approach [29] has two shortcomings. First, it is only applicable to distance-based classifiers. These conventional classifiers do not scale well with the volume of the training data and the complexity of the data patterns [42]. It is desirable to support the DNNs that give the state-of-the-art learning performance in a range of applications. Second, obfuscating the public data vectors and returning the results may incur known-plaintext attacks and engenders a clear privacy breaching concern. For instance, a proactively curious coordinator may use a public data vector $\mathbf{z}_k = [1, 0, 0, \ldots, 0]^\intercal$ to extract the first column of $\mathbf{R}_i$. Other columns of $\mathbf{R}_i$ can be similarly extracted by using specific public data vectors. Even without using these specific public data vectors, in general, the private random projection matrix $\mathbf{R}_i$ can be estimated using regression analysis based on a number of public data vectors and the corresponding projections.

The study [39] also uses random projection to obfuscate the data vector $\mathbf{x}$ in training and executing a Sparse Representation Classifier. However, all participants use the same random projection matrix, rendering the system vulnerable to the collusion between any single participant and the coordinator.

Different from [39], each participant in our approach uses its own private random project matrix, rendering the collusion futile. Different from [29], our approach uses DNNs and leverages on the deep learning capability to avoid the regression phase that is vulnerable to the known-plaintext attacks. Different from [22] that is too compute-intensive for IoT objects, our approach uses GRP that introduces light computation overhead only.

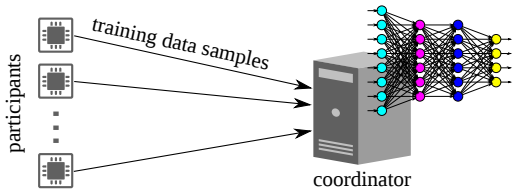


Figure 1: A collaborative learning system.

3.3 Other Related Work

In CryptoNets [19], the computation of each neuron in a neural network trained using plaintext data is performed in the domain of homomorphic encryption. During the classification phase, the participant sends the homomorphically encrypted data to the coordinator for classification. The work [12] extends [19] to support more hidden layers. However, these studies [12, 19] address *privacy-preserving classification outsourcing* (i.e., offloading the classification computation to a honest-but-curious entity), rather than the collaborative learning addressed in this paper. The training in [12, 19] is performed based on plaintext data. Moreover, the homomorphic encryption is too computation intensive for resource-constrained IoT devices, which will be shown in §6.

The *differentially private machine learning* (DPML) [5, 14, 41] builds a classifier that cannot be used to infer the training data. The training of the classifier is based on plaintext data. For DNNs, DPML can be achieved by perturbing the gradients in each iteration of the SGD with additive noises [5, 41]. DPML and PPCL address different problems, i.e., PPCL preserves the privacy of the training data against the honest-but-curious coordinator who builds the classifier, whereas DPML trusts the classifier builder and preserves the privacy of the training data against the curious user of the classifier. Thus, in DPML, the plaintext training dataset is available to the classifier builder; differently, in PPCL, only encrypted or obfuscated training data is made available to the classifier builder (i.e., the learning coordinator).

4 PROBLEM STATEMENT AND APPROACH

In this section, we state the PPCL problem in §4.1 and present the proposed GRP approach in §4.2. §4.3 provides two illustrating examples for insights into understanding the effect of GRP on training DNN-based classifiers. §4.4 discusses two other alternative approaches for lightweight PPCL and their limitations.

4.1 Problem Statement

This section states the problem addressed in this paper. §4.1.1 defines the system model; §4.1.2 defines the threat and privacy models; §4.1.3 discusses several relevant issues.

4.1.1 System model. In this paper, we consider a PPCL system with N resource-constrained *participants* and an honest-but-curious *coordinator* with sufficient computation power. Fig. 1 illustrates the system. During the learning phase, the participants contribute training data samples to build a supervised classifier. As discussed in §2.1, the training dataset $\mathcal{D}_i$ contributed by the participant i consists of M_i data vectors $\{\mathbf{x}_{i,j}|j \in [1, M_i]\}$ and the corresponding class labels $\{y_{i,j}|j \in [1, M_i]\}$. As the learning process is often

compute-intensive, most of the learning computation should be accomplished by the coordinator. In this paper, we focus on addressing the problem of building an effective supervised classifier while protecting certain privacy contained in the data vectors.

4.1.2 Threat and privacy models. The privacy concern regarding the data vectors is primarily due to that the data vectors may contain information beyond the classification objective in question. For example, consider a PPCL system for training a classifier to recognize human body activity (e.g., sitting, walking, climbing stairs, etc). The recognition is based on various body signals (e.g., motion, heart rate, breath rate, etc) that are captured by wearable sensors. However, the raw body signals can also be used to infer health statuses of the participants and even pinpoint the patients of certain diseases. In this paper, we adopt the following threat and privacy models.

Threat model: It consists of the following two aspects:

- *Honest-but-curious coordinator:* We assume that the coordinator will honestly coordinate the collaborative learning process, aiming to train the best supervised classifier. Thus, it will neither tamper with any data collected from or transmitted to the participants. However, the coordinator is curious about the participants' privacy contained in the training data vectors.
- *Potential collusion between participants and coordinator:* We assume that the participants are not trustworthy in that they may collude with the coordinator in finding out other participants' privacy contained in the data vectors. The colluding participants are also honest, i.e., they will faithfully contribute their training data to improve the supervised classifier. The design of the PPCL system should keep the privacy preservation for a participant when any or all other participants are colluding with the coordinator.

Privacy model: The raw form of each data vector is the participant's privacy to be protected. The error in estimating the data raw form by the coordinator can be used as a metric to measure the degree of privacy protection. Data form confidentiality is an immediate and basic privacy requirement in many applications.

4.1.3 Several other issues. In the following, we discuss three issues that are related to privacy protection:

- *Training data anonymization:* We aim to support anonymization of the training data. That is, the coordinator should not expect to know the participant's identity for any received training data sample. Moreover, the coordinator cannot determine whether any two training data samples are from the same participant. To achieve the above strong anonymity, the training data samples can be transmitted in separate sessions via an anonymous communication network [16]. Moreover, the transmissions of the data samples from all participants can be interleaved randomly, such that the coordinator cannot associate the data samples from the same participant by their arrival times. Note that the training data anonymization requirement is not mandatory, because the anonymous communication may incur large overhead for some resource-constrained IoT objects. However, the design of our PPCL approach will not leverage the participants' identities to support data anonymization.
- *Label privacy:* The class labels $\{y_{i,j}|j \in [1, M_i]\}$ may also contain information about the participant. In this paper, we do not consider label privacy because the participant willingly contributes the labeled data vectors and should have no expectation of privacy regarding labels. In practice, several means can be taken to mitigate the concern of label privacy leak. First, the training data anonymization mitigates the concern during the learning phase. Second, during the classification phase, if the participant has sufficient processing capability to perform the classification computation, the coordinator may send the trained model to the participant for local execution. Existing studies have enabled the execution of deep models on personal and low-end devices [25, 47]. Low-power inference chips (e.g., Google's Edge TPU [21]) will further enhance low-end devices' capabilities in executing classification models. Note that the studies [25, 47] and the inference chips are not to support the much more compute-intensive training.
- *Other privacy models:* Differential privacy [17] aim to achieve indistinguishability of different data vectors is another widely used quantifiable privacy definition. However, as discussed in §4.4 and evaluated in §5, the additive noisification implementation of differential privacy is ill-suited for PPCL.

4.2 Gaussian Random Projection Approach

Existing DML and homomorphic encryption approaches incur significant computation and communication overhead due to the many computation/communication rounds and data volume swell. In §6, we will provide benchmark results to show this. Thus, these approaches are not promising for resource-constrained participants. This section describes a GRP-based approach that is computationally lightweight and communication efficient for the participants. The overview of our approach is presented as follows.

At the system initialization, each participant i independently generates a random Gaussian matrix $\mathbf{R}_i \in \mathbb{R}^{k \times d}$, where d is the dimension of the data vector. During the learning phase, the participant i keeps $\mathbf{R}_i$ secret and uses it to project all the training data vectors. The participant i transmits the projected training dataset $\mathcal{D}_i = \{\mathbf{R}_i\mathbf{x}_{i,j}, y_{i,j}|j \in [1, M_i], y_{i,j} \in C\}$ to the coordinator. After collecting all projected training datasets $\mathcal{D}_i$, $i = 1, \ldots, N$, the coordinator applies deep learning algorithms to train the classifier $h(\cdot|\boldsymbol{\theta}^*)$. During the classification phase, the participant i still uses $\mathbf{R}_i$ to project the test data vector $\mathbf{x}$ and obtains the classification result $h(\mathbf{R}_i\mathbf{x}|\boldsymbol{\theta}^*)$. As discussed in §4.1, the classification computation can be carried out at the participant or the coordinator, depending on whether the participant is capable of executing the trained deep model. In our approach, each participant independently generates its random projection matrix to counteract the collusion between participants and coordinator. Now, we explain the two key components of our approach: GRP and deep learning on projected data.

4.2.1 Gaussian random projection. In this work, we adopt Gaussian matrices. Specifically, each element of $\mathbf{R}_i$ is sampled independently from the standard normal distribution [6]. The rationale of choosing Gaussian matrices will be explained in §4.3.2. We set the

row dimension of $\mathbf{R}_i$ smaller than or equal to its column dimension, i.e., $k \leq d$. Thus, the GRP can also compress the data vector. We define the compression ratio as $\rho = d/k$. The understanding regarding the admission of compression into the training data projection is as follows. From the compressive sensing theory [11], a sparse signal can be represented by a small number of linear projections of the original signal and recovered faithfully. Therefore, in the compressively projected data vector, the feature information still exists, provided that the adopted compression ratio is within an analytic bound [11]. In §5, we will evaluate the impact of the compression ratio ρ on the learning performance.

With GRP, if $\mathbf{R}_i$ is kept confidential to the coordinator, it is computationally difficult (practically impossible) for the coordinator to generate a meaningful reconstruction of the original data vector from the projected data vector [30, 37]. Thus, GRP protects the form of the original data. In the worst case where the coordinator obtains $\mathbf{R}_i$, the estimation error given by Property 2 in §2.2 can be used as a measure of privacy protection. Random projection has been used as a lightweight approach to protect data form confidentiality in various contexts [28, 43, 44, 46].

4.2.2 Deep learning on projected data. Feature extraction is a critical step of supervised learning. With the traditional *shallow learning*, the classification system designer needs to handcraft the feature. The emerging deep learning method [26] automates the design of feature extraction by *unsupervised feature learning*, which is often based on a neural network consisting of a large number of parameters. Thus, the deep model is often a tandem of the feature extraction stage and the classification stage. For example, a convolutional neural network (CNN) for image classification consists of convolutional layers and dense layers, which are often considered performing the feature extraction and classification, respectively.

Our approach leverages on the unsupervised feature learning capability of deep learning to address the data distortion introduced by the GRP. We now illustrate this using a simple example system, in which there is only one participant and the projection matrix $\mathbf{R}$ is a square invertible matrix. Moreover, we make the following two assumptions to simplify our discussion. First, we assume that a linear transform $\mathbf{\Psi} \in \mathbb{R}^{f \times d}$ gives effective features of the data vectors, where f is the feature dimension. That is, $\mathbf{f} = \mathbf{\Psi x}$ is an effective representation of the data vector $\mathbf{x}$ for classification. Second, we assume that $\mathbf{\Psi}$ can be learned in the form of a neural network by the unsupervised feature learning. Now, we discuss the impact of the random projection on the unsupervised feature learning. After the projection, the data vector becomes $\mathbf{Rx}$. Moreover, the linear transform $\mathbf{\Psi R}^{-1}$ will be an effective feature extraction method, since $\mathbf{f} = \left(\mathbf{\Psi R}^{-1}\right)(\mathbf{Rx})$. It is reasonable to expect that the unsupervised feature learning can also build a neural network to capture the linear transform $\mathbf{\Psi R}^{-1}$, similar to the unsupervised feature learning to capture the $\mathbf{\Psi}$ based on the plaintext training data $\mathbf{x}$. As a result, the deep model trained using the projected data can still classify future projected data vectors. In §4.3, we will use a numerical example to illustrate this.

The above discussion based on linear features provides a basis for us to understand how the unsupervised feature learning helps address the distortion caused by the GRP. In practice, effective feature extractions are generally non-linear mappings. Neural network-based deep learning has shown strong capability in capturing sophisticated features beyond the above ideal linear features. In this paper, based on multiple datasets, we will investigate the effectiveness of deep learning to address the distortion caused by the GRP.

As discussed earlier, each participant independently generates a Gaussian matrix to counteract the potential collusion between participants and the coordinator. However, this introduces a challenge to deep learning, because the pattern for a class of projected data vectors from N participants will be a composite of N different patterns. Thus, intuitively, a deeper neural network and a larger volume of training data will be needed to well capture the data patterns with increased complexity due to the participants' independence in generating their projection matrices. We note that, the participants' independence also engenders the following possible situation that undermines the learning performance and leads to classification errors: $\mathbf{R}_u\mathbf{x}_u = \mathbf{R}_v\mathbf{x}_v$, where $\mathbf{x}_u$ and $\mathbf{x}_v$ are generated by participants u and v and belong to different classes. However, for high-dimensional data vectors, the probability of the above situation is low. The more complex data patterns due to the independent projection matrix generation will be the major challenge. In this paper, we conduct extensive experiments to assess how well deep learning can scale with the number of participants, compared with the traditional learning approaches.

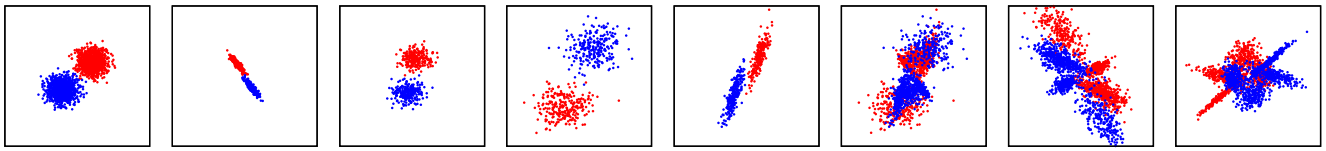
4.3 Illustrating Examples

We use two examples to illustrate the intuitions discussed in §4.2.

4.3.1 A 2-dimensional example. We consider a PPCL system with four participants (i.e., $N = 4$) to build a two-class classifier. The original data vectors in the two classes follow two 2-dimensional Gaussian distributions with means of $[-2, -2]^\intercal$ and $[2, 2]^\intercal$, and the same covariance matrix of $[1, 0; 0, 1]$. Fig. 2(a) shows the plaintext data vectors generated by the four participants. From the figure, the plaintext data vectors of the two classes can be easily separated using a simple hyperplane. Each participant independently generates a Gaussian random matrix. Figs. 2(b)-2(e) show the projected data vectors of each participant. We can see that the patterns of the projected data vectors are different across the participants. Fig. 2(f) shows the mixed projected data vectors received from all participants. Compared with Fig. 2(a), the pattern of the mixed projected data from all participants is highly complex. Moreover, no simple hyperplane can well divide the two classes.

We also generate two other sets of the random projection matrices for all participants. Figs. 2(g) and 2(h) show the mixes of all participants' projected data vectors with the two sets of random projection matrices, respectively. Similarly, the pattern of the mixed projected data from all participants is highly complex.

We construct a classifier based on an MLP with two hidden layers of 30 and 40 rectified linear units (ReLUs), respectively. The input layer admits a 2-dimensional data vector, whereas the output layer consists of two ReLUs. The final classification result is generated using a softmax function based on the output layer's ReLU values. Moreover, we construct an SVM classifier as a baseline approach. We use LIBSVM [13] to implement the classifier. The SVM



(a) Original data (b) Participant 1 (c) Participant 2 (d) Participant 3 (e) Participant 4 (f) Coordinator (g) Coordinator (h) Coordinator

Figure 2: Two-dimensional example. Original data vectors and projected data vectors (red: class 0; blue: class 1). The ranges for the x and y axes are $[-10, 10]$.

classifier uses radial basis function (RBF) kernel with two configurable parameters C and λ. During the training phase, we apply grid search to determine the optimal settings for C and λ.

First, we use disjoint subsets of the original data shown in Fig. 2(a) to train and test the MLP and SVM classifiers. Both classifiers can achieve 99% test accuracy. This shows that the MLP and the SVM are properly designed for the 2-dimensional data vectors.

Then, we use disjoint subsets of the randomly projected data shown in Fig. 2(f) to train and test the MLP and SVM classifiers. Moreover, we also increase the number of participants in the PPCL system. Fig. 3 shows the test accuracy versus the number of participants. We can see that the MLP classifier always outperforms the SVM classifier. Moreover, the test accuracy decreases with the number of participants. This is because, with more participants, the pattern of the projected data becomes more complex, introducing challenges to both MLP and SVM. The test accuracy difference between MLP and SVM increases from 2% to 7%, when the number of participants increases from 4 to 20. This result is also consistent with the understanding that deep learning is more effective in capturing complex patterns than traditional learning.

4.3.2 A 10-dimensional example. Now, we use another example system to understand the effect of deep learning's unsupervised feature learning capability in addressing the data distortion caused by the random projection. This example is a PPCL system with only one participant (i.e., $N = 1$). The original data vectors in two classes follow two 10-dimensional Gaussian distributions, with the $[-2, -2, \ldots, -2]^\intercal$ and $[2, 2, \ldots, 2]^\intercal$ as the respective mean vectors, and the 10-dimensional identity matrix as their identical covariance matrix.

In our discussions in §4.2.2, we assume that the projection matrix $\mathbf{R}$ is invertible and the unsupervised feature learning tend to capture $\mathbf{\Psi R}^{-1}$. As learning algorithms are based on numerical computation on the training data, an ill-conditioned matrix $\mathbf{R}$ will impede efficient fitting of $\mathbf{\Psi R}^{-1}$. We verify this intuition by assessing the learning performance of the single-participant PPCL system using different $\mathbf{R}$ matrices with varying condition numbers. Specifically, by following a method described in [8], the participant generates a random square matrix $\mathbf{R}$ that has a certain condition number value. The condition number is defined as $\|\mathbf{R}\|_F \|\mathbf{R}^+\|_F$ [35], where $\mathbf{R}^+$ denotes the pseudoinverse of $\mathbf{R}$ and $\|\cdot\|_F$ represents the Frobenius norm. Fig. 4 shows the test accuracy of the MLP and SVM classifiers trained using data projected by $\mathbf{R}$ versus the condition number of $\mathbf{R}$. Note that a larger condition number means that the matrix is more ill-conditioned. We can see that the test accuracy decreases with the condition number, consistent with the intuition.

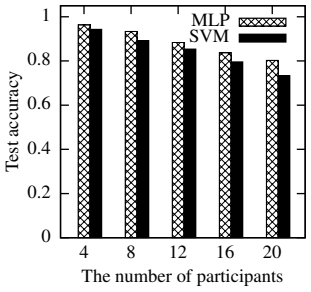


Figure 3: Test accuracy based on projected data vs. the number of participants.

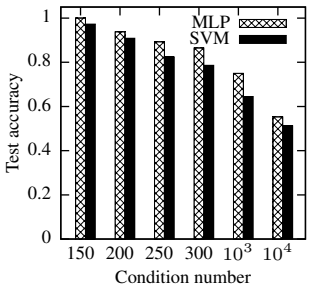


Figure 4: Test accuracy based on projected data vs. the condition number.

The study [15] analyzes the distribution of the condition numbers of Gaussian random matrices. The results show that a Gaussian random matrix is well-conditioned with a high probability. For instance, it is shown in [15] that for a 10×5 Gaussian random matrix, the probability that its condition number is larger than 100 is less than 6×10^{-7}. This is a basis for our choice of using Gaussian random matrices to project data.

4.4 Alternative Approaches and Limitations

This section discusses two alternative approaches to PPCL and their limitations. These two alternatives will be used as the baseline approaches in our comparative performance evaluation in §5.

4.4.1 Non-collaborative learning. If the data anonymity requirement is not enforced, the coordinator can train a separate deep model based on the projected data vectors contributed by each participant. This alternative approach can address the challenge of the complex mixed patterns due to different random projection matrices adopted by different participants as illustrated in §4.3. However, it loses the advantages of collaborative learning, i.e., the increased data volume and pattern coverage. From our evaluation in §5, compared with our proposed approach, despite that this non-collaborative learning approach additionally uses the participant identity information, it yields inferior average accuracy.

4.4.2 Differential privacy. Differential privacy (DP) [17] is a rigorous information-theoretic approach to prevent leak of individual records by statistical queries on a database of these records. The ϵ-DP [17] is formally defined as follows: *A randomized algorithm $\mathcal{A} : \mathbb{D} \to \mathbb{R}^t$ gives ϵ-DP if for all adjacent datasets $D_1 \in \mathbb{D}$ and $D_2 \in \mathbb{D}$ differing on at most one element, and all $S \subseteq Range(\mathcal{A})$,*

$\Pr(\mathcal{A}(D_1) \in S) \leq \exp(\epsilon) \cdot \Pr(\mathcal{A}(D_2) \in S)$. The ϵ, a positive real number, is a measure of privacy loss, i.e., a smaller ϵ implies better privacy. When ϵ is very small, $\Pr(\mathcal{A}(D_1) \in S) \simeq \Pr(\mathcal{A}(D_2) \in S)$ for all $S \subseteq Range(\mathcal{A})$, which means that the query results $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ are almost indistinguishable based on any "test criterion" of S. The indistinguishability between the query results $\mathcal{A}(D_1)$ and $\mathcal{A}(D_2)$ decreases with ϵ. The study [18] develops an approach of adding Laplacian noises to implement ϵ-DP. Specifically, for all function $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^t$, the randomized algorithm $\mathcal{A}(D) = \mathcal{F}(D) + [n_1, n_2, \ldots, n_t]^\intercal$ gives ϵ-DP, where each n_i is drawn independently from a Laplace distribution $\text{Lap}(S(\mathcal{F})/\epsilon)$ and $S(\mathcal{F})$ denotes the global sensitivity of $\mathcal{F}$. Note that $\text{Lap}(\lambda)$ denotes a zero-mean Laplace distribution with a probability density function of $f(x|\lambda) = \frac{1}{2\lambda} e^{\frac{|x|}{\lambda}}$; the global sensitivity is

$$S(\mathcal{F}) = \max_{\forall D' \in \mathbb{D}, \forall D'' \in \mathbb{D}} ||\mathcal{F}(D') - \mathcal{F}(D'')||_1.$$

Essentially, ϵ-DP gives quantifiable indistinguishability of the query results based on different datasets. The ϵ-DP framework has been applied in various privacy preservation problems in machine learning. As discussed in §3.1, the DML approaches to PPCL [23, 40] add random noises to the parameters exchanged between the participants and the coordinator to achieve ϵ-DP. The original parameters can be viewed as deterministic query results of the training data. Adding random noises to the parameters ensures certain levels of indistinguishability between the noise-added parameters based on different training datasets. The achieved ϵ-DP mitigates the privacy concern that the curious coordinator may use the received parameters to infer the existence of particular data vectors in the training dataset. However, these DML approaches [23, 40] incur significant overhead to resource-constrained participants. For PPCL based on resource-constrained participants, an approach to achieving ϵ-DP is to add a Laplacian noise vector to the original data vector $\mathbf{x}$ and then transmit the noise-added data vector to the coordinator for building the classifier. By doing so, certain levels of indistinguishability between the noise-added data vectors based on different original data vectors are achieved.

Additive noisification and multiplicative GRP preserve different forms of privacy. Compared with protecting indistinguishability under the DP framework, we believe that protecting the confidentiality of the raw data form, which can be achieved by GRP, is a more immediate and basic privacy requirement in many applications. The additive noisification, though achieving ϵ-DP, falls short of protecting the confidentiality of the raw data form. Specifically, under the ϵ-DP framework based on zero-mean Laplacian noises, a noise-added data vector can be considered an unbiased estimate of the original data vector with an estimation variance related to ϵ. Thus, the coordinator always has a meaningful (i.e., unbiased) estimate of the raw data. According to Property 2 in §2.2, this only happens to the GRP approach in the worst (and unrealistic) case that the projection matrix is revealed to the coordinator; other than the worst case, the coordinator cannot have a meaningful estimate of the raw data form. In the image classification case studies in §5, we will show that when ϵ is small (i.e., good DP), the contents of the noise-added images can still be interpreted. In contrast, the projected images cannot be interpreted visually at all.

Applying ϵ-DP to PPCL with resource-constrained participants also introduces the following two challenges.

Non-trivial computation overhead: From the DP theory, an independent random noise vector should be generated and added to every data vector $\mathbf{x}$. However, random number generation is often a costly operation due to the use of various mathematical functions. The continuous generation of Laplacian noises will incur non-trivial computation overhead for the resource-constrained participants. Differently, in our approach, the random projection matrix generation is a one-off overhead. The projection to compute $\mathbf{Rx}$ is a lightweight operation consisting of multiplications and additions only. Our previous work [43] has implemented the projection operation on an MSP430-based platform. Moreover, the projection can be sped up if a parallel computing chip (e.g., Google's Edge TPU [21]) is available.

Learning performance degradation: As discussed in §4.2.2, the projection matrix can be implicitly learned by the deep learning algorithms. Differently, the additive Laplacian noises to ensure ϵ-DP can be considered neither a pattern nor an embedding that can be learned by learning algorithms. Thus, the Laplacian noises will only negatively affect the learning performance. Our evaluation in §5 shows that the Laplacian noises for achieving moderate ϵ-DP significantly degrade the learning performance.

From the above discussions and the evaluation results in §5, adding Laplacian noises to the training data for ϵ-DP is not a promising approach to PPCL with resource-constrained participants.

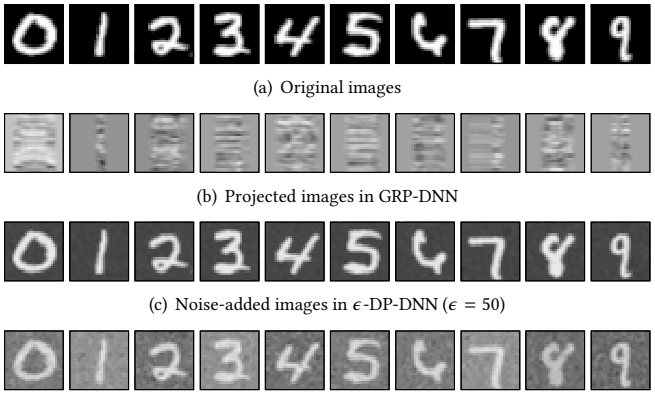
5 PERFORMANCE EVALUATION

In this section, we extensively compare the accuracy achieved by various approaches. The computation and communication overhead of these approaches will be profiled in §6 based on their implementations on a testbed.

5.1 Evaluation Methodology and Datasets

We conduct extensive evaluation to compare several approaches:

- **GRP-DNN:** This is the proposed approach consisting of GRP at the participants and collaborative learning based on a DNN at the coordinator. The design or choice of the DNN model will be application specific. The DNN models and training algorithms are implemented based on PyTorch [2].
- **GRP-SVM:** This baseline approach applies GRP at the participants and trains an SVM-based classifier at the coordinator. The SVM-based classifier is implemented using LIBSVM [13]. The classifier uses RBF kernel with two configurable parameters C and λ. During the training phase, we apply grid search to determine the best settings for C and λ. This grid search is often lengthy in time (e.g., several days).
- **GRP-NCL:** This is the non-collaborative learning (NCL) baseline approach described in §4.4.1. It runs GRP at the participants and trains a separate DNN for each participant at the coordinator. Compared with other approaches, this approach additionally requires the identity of the participant for each training sample.
- **ϵ-DP-DNN:** As described in §4.4.2, this approach implements ϵ-DP by adding Laplacian noise vectors to the data vectors



(a) Original images

(b) Projected images in GRP-DNN

(c) Noise-added images in ϵ-DP-DNN ($\epsilon = 50$)

(d) Noise-added images in ϵ-DP-DNN ($\epsilon = 10$)

Figure 5: Example images from MNIST dataset.

and performs collaborative deep learning based on a DNN at the coordinator.

- **ϵ-DP-SVM:** This approach implements ϵ-DP by adding Laplacian noise vectors to the data vectors and performs collaborative learning based on SVM at the coordinator.
- **CNN, SVM, MLP, ResNet-152:** These are the plain learning approaches based on the CNN, SVM, MLP, and ResNet-152 models, respectively. They do not protect any privacy.

The performance evaluation is performed based on two datasets, i.e., MNIST [27] and spambase [4]. The MNIST dataset consists of 60,000 training samples and 10,000 testing samples. Each sample is a 28×28 grayscale image showing a handwritten digits from 0 to 9. Fig. 5(a) shows an instance of each digit. The spambase dataset consists of 4,601 samples. Each sample consists of (i) a 57-dimensional feature vector that is extracted from an e-mail message and (ii) a class label indicating whether the e-mail message is an unsolicited commercial e-mail. The details of the feature vector can be found in [4]. As the data volume of this spambase dataset is limited, we apply data augmentation to the spambase by adding zero-mean Gaussian noises, resulting in 40,000 training samples and 400 testing samples. We choose these two datasets because the small sizes of the data vectors are commensurate with the limited computing and transmission capabilities of IoT end devices.

Training a spam detector based on user-contributed samples (e.g., e-mails) may cause privacy concerns. Thus, our proposed approach well fits in this case. The choice of the vision-based character recognition task with the MNIST dataset allows us to leverage on the learning capabilities of the deep models that are often designed for image classification. Moreover, by using images as the data vectors, the effect of the distortion caused by noise adding or random projection can be visualized for intuitive understanding. Although the character recognition task is not privacy-sensitive, its results will provide understanding on other image classification-based privacy-sensitive applications, such as collaboratively training a mood classifier using the photos in the album of the users' smartphones.

For a PPCL system with N participants, we divide both the training and testing samples into N disjoint sets evenly. Each set is

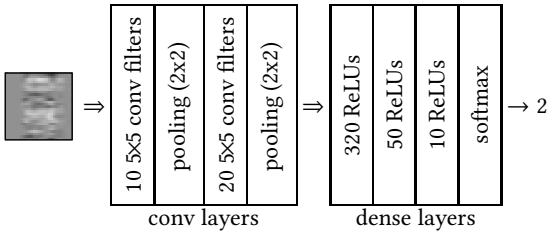


Figure 6: CNN with a projected MNIST image as input.

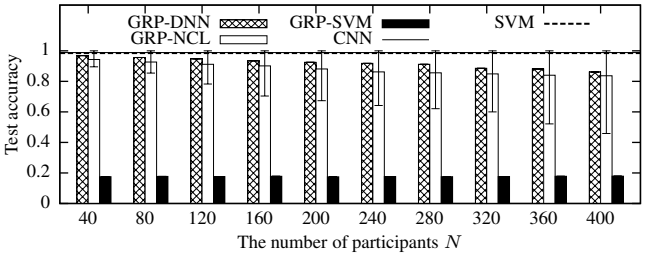


Figure 7: Impact of the number of participants (MNIST). The error bars for GRP-NCL represent min and max.

assigned to a participant. Under GRP-DNN, GRP-SVM, and GRP-NCL, each participant independently generates its random Gaussian matrix as described in §4.2.1 and uses the matrix to project its plaintext data vectors. The deep models and SVM are trained by the coordinator based on the projected or noise-added training data vectors from the participants. The trained deep models and SVM are used to classify the projected or noise-added testing data vectors to measure the test accuracy as the evaluation results.

5.2 Evaluation Results with MNIST Dataset

We design a CNN that is used in the GRP-DNN, GRP-NCL, and ϵ-DP-DNN approaches. The CNN consists of two convolutional layers and three dense layers of ReLUs. We apply max pooling after each convolutional layer to reduce the dimension of data after convolution. The max pooling controls overfitting effectively and improves the CNN's robustness to small spatial distortions in the input image. The last dense layer has ten ReLUs corresponding to the ten classes of MNIST. A softmax function is used to make the classification decision based on the outputs of the last dense layer. Fig. 6 illustrates the design of the CNN. Note that, without random projection, the CNN and the SVM with grid search for kernel parameters can achieve test accuracy of 98.7% and 98.52%. This shows that the CNN and SVM can well capture the patterns of MNIST.

First, we evaluate the impact of the number of participants N on the learning performance of GRP-DNN, GRP-NCL, and GRP-SVM. Fig. 7 shows the results. The two horizontal lines in Fig. 7 represent the test accuracy of the plain CNN and SVM without any privacy protection. The two lines overlap. When N increases from 40 to 400, the test accuracy of GRP-DNN decreases from 96.87% to 86.18%. If N is no greater than 280, GRP-DNN can maintain

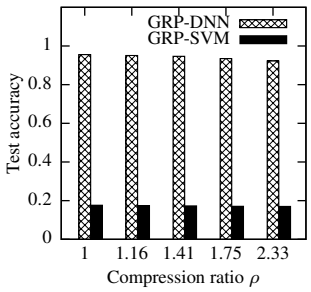


Figure 8: Impact of data compression on learning performance (MNIST, $N = 100$).

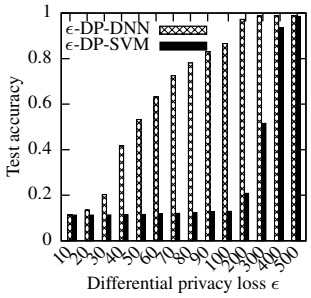


Figure 9: Impact of differential privacy loss on learning performance (MNIST).

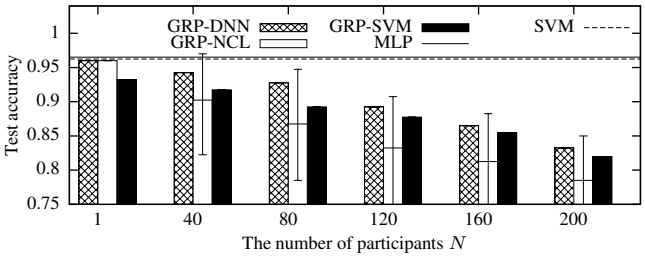


Figure 10: Impact of the number of participants (spambase). The error bars for GRP-NCL represent min and max.

a test accuracy greater than 90%. The drop of accuracy with increased N is consistent with the understanding that distinct random projection matrices increase the pattern complexity of the aggregated data. However, for MNIST data with light pattern complexities, the GRP-DNN approach can support up to 280 IoT objects for a satisfactory classification accuracy of 90%. Under the GRP-NCL approach, the deep models corresponding to the participants have different test accuracy values. The histogram and error bars in Fig. 7 represent the average, minimum, and maximum of the test accuracy values across all trained deep models. Under each setting of N, the maximum test accuracy is 100%. However, the average test accuracy is consistently lower than that of GRP-DNN. This shows that, the GRP-NCL that needs to compromise data anonymity yields inferior average learning performance compared with GRP-DNN. This result shows the advantage of collaborative learning. Lastly, the GRP-SVM approach gives poor test accuracy around 17.5%. This is because no efficient RBF kernels can be found to create proper hyperplanes for classification. This suggests that DNNs are more efficient to cope with the distortions caused by projections.

Second, we evaluate the impact of GRP's data compression on the learning performance. Fig. 8 shows the results when $N = 100$. When the compression ratio increases from 1 (i.e., no compression) to 2.33 (i.e., 43% of data volume is retained), the test accuracy of GRP-DNN decreases from 95.52% to 92.85% only. From our discussion in §4.2.1, the good tolerance of GRP-DNN against data compression is due to the high sparsity of the MNIST images. In contrast, the GRP-SVM approach performs poorly under all compression ratio settings.

Then, we evaluate the impact of adding Laplacian noises to implement ϵ-DP on the learning performance. Fig. 9 shows the test accuracy of ϵ-DP-DNN versus the privacy loss level ϵ. When $\epsilon = 100$ (small Laplacian noises and large differential privacy loss), the ϵ-DP-DNN achieves a test accuracy of 86.6%, lower than those achieved by GRP-DNN when N is up to 400. When $\epsilon = 10$, the performance of ϵ-DP-DNN drops to 11.4%, close to the performance of random guessing. For comparison, we visualize the projected and noise-added images with two ϵ settings in Fig. 5. From Fig. 5(b), we cannot visually interpret the projected images. However, from Figs. 5(c) and 5(d), the noise-added images are easily interpreted when ϵ is down to 10. Note that in our evaluation, we use the same CNN model as shown in Fig. 6 for the GRP-DNN, GRP-NCL, and ϵ-DP-DNN approaches. We do not spend special efforts to improve the CNN design in favor of any approach; we only make sure the CNN fed with the original MNIST images achieves satisfactory performance. The poor performance of ϵ-DP-DNN is consistent with the understanding that the performance of deep learning can be susceptible to small perturbations to the data vectors [48]. There are also systematic approaches to generating adversary examples with small differences from the training samples to yield wrong classification results [10, 20]. Special cares are needed in the deep model design to improve robustness against human-indiscernible perturbations [48]. Significant noises, which are required to achieve good DP protection, are still open challenges to deep learning. Thus, under the ϵ-DP framework, it is challenging to achieve a desirable trade-off between the privacy protection strength and learning performance.

We discussed in §4.4.2 that the additive noisification for ϵ-DP is ineffective in achieving a good trade-off between learning performance and protecting the confidentiality of the raw forms of the training data. Now, we compare the results of GRP-DNN ($N = 1$, $k = d - 1$) and ϵ-DP-DNN. We consider the worst case for GRP-DNN, i.e., the projection matrix $\mathbf{R}$ is revealed to the curious coordinator. From Property 2 in §2.2, the minimum norm estimate of the original data vector by the coordinator will have a per-element variance of about 410 for any MNIST image. Under this setting, GRP-DNN can achieve a test accuracy of 94.82%. To achieve the same per-element variance of 410, the ϵ value adopted by the ϵ-DP-DNN should be 18.89. Under this ϵ setting, the test accuracy of ϵ-DP-DNN is 12.86% only.

Fig. 9 also shows the test accuracy of the ϵ-DP-SVM approach. It performs poorly when $\epsilon \leq 100$. Only when the added noises are very small under the settings of $\epsilon = 400$ and $\epsilon = 500$, this approach can achieve good test accuracy.

5.3 Evaluation Results with Spambase Dataset

We design a 5-layer MLP classifier to detect spams. The numbers of ReLUs in the five layers are 57, 100, 50, 10, and 2, respectively. A softmax function is used lastly to make the final detection decision. Dropout is used during training to suppress overfitting. Without random projection, the MLP and the SVM with grid research for kernel parameters can achieve test accuracy of 96.52% and 96.25%, respectively. This shows that the MLP and SVM can well capture the patterns of spambase.

We evaluate the impact of the number of participants N on the learning performance of GRP-DNN, GRP-NCL, and GRP-SVM. Fig. 10 shows the results. The two horizontal lines in Fig. 10 represent the test accuracy of the plain MLP and SVM without any privacy protection. When N increases from 1 to 200, the test accuracy of GRP-DNN decreases from 96% to 83.25%. If N is no greater 100, GRP-DNN can maintain a test accuracy of about 90%. The average test accuracy of GRP-NCL is about 5% lower than that of the GRP-DNN, because GRP-NCL misses the advantages of collaborative learning. The test accuracy of the GRP-SVM is about 1.25% to 2.75% lower than that of the GRP-DNN. Thus, the GRP-SVM performs satisfactorily for this spambase dataset. The reasons are two-fold. First, in this spambase dataset, the classifiers operate on the e-mail features, rather than the raw data. Second, the RBF kernel is effective in capturing the features. In fact, the nature of this spambase dataset is similar to that of the 2-dimensional and 10-dimensional generated feature datasets used in §4.3, on which the GRP-DNN and GRP-SVM perform similarly.

5.4 Summary and Discussion

We have several observations from the results in §5.2 and §5.3:

- Compared with SVM, deep learning can better adapt to the complexity introduced by the multiplicative projections.
- Although the GRP-NCL approach additionally uses the identities of the participants, it gives inferior performance compared with the collaborative GRP-DNN. This shows the advantage of collaborative learning even with the privacy preservation requirement.
- Compared with GRP-DNN, the additive noisification for ϵ-DP achieves inferior trade-off between learning performance and protecting confidentiality of raw forms of training data.
- GRP-DNN shows promising scalability with the number of participants observing low-complexity data patterns. For the MNIST and spambase datasets, the GRP-DNN can well support 100 participants with a few percents test accuracy drop. For large-scale PPCL systems involving more participants, we envision a two-tier system architecture as follows. The participants are divided into groups. At the first tier, our GRP-DNN is applied within each group; at the second tier, the DML approach is applied among the group coordinators.

6 IMPLEMENTATION AND BENCHMARK

In this section, we measure the overhead of two PPCL approaches (i.e., our GRP-DNN and Crowd-ML [23]) and a privacy-preserving classification outsourcing approach (i.e., CryptoNets [19]) on a testbed of 14 Raspberry Pi 2 Model B nodes [3] and a powerful workstation computer. The Raspberry Pi nodes act as PPCL participants and the workstation acts as the coordinator. They are interconnected using a 24-port network switch. We benchmark these approaches using the MNIST dataset. The training and testing samples are evenly allocated to the participants, resulting in 4,285 training samples and 714 testing samples on each participant. The implementations of the three approaches (GRP-DNN, Crowd-ML, CryptoNets) on the same platform, i.e., Raspberry Pi, allow fair comparisons. The participant part of our GRP-DNN can be implemented on mote-class platforms. Our previous work [43] has implemented Gaussian matrix generation and GRP on the MSP430-based Kmote platform. However, it is difficult/impossible to implement Crowd-ML and CryptoNets on mote-class platforms.

Table 1: The overhead of various approaches.

	Overhead	GRP-DNN	Crowd-ML	CryptoNets
Training	Participant comm. vol.	33.6 MB	117.2 MB	n/a
	Participant compute time	0.96 s	367.24 s	n/a
	Coordinator compute time	928.34 s	1.04 s	n/a
Testing	Participant comm. vol.	5.6 MB	n/a	15.0 MB
	Participant compute time	0.16 s	4.67 s	116 hours
	Coordinator compute time	40.88 s	n/a	

n/a represents "not applicable."

We implement our GRP-DNN approach on the testbed. The compression ratio $\rho = 1$ (i.e., no compression). Table 1 shows the benchmark results. During the training phase, each GRP-DNN participant needs to transmit a total of 33.6 MB projected data. A participant can complete projecting all the 4,285 training images within 0.96 s. The coordinator needs 928.34 s to train the CNN. In our GRP-DNN implementation, the testing phase is performed on the coordinator. During the testing phase, each participant completes projecting all the 714 testing images within 0.16 s and transmits a total of 5.6 MB data to the coordinator. The coordinator needs 40.88 s to classify all projected testing images from the participants. Note that GPU acceleration is not used in this benchmark for GRP-DNN during both the training and testing phases.

The Crowd-ML [23] is a DML approach. In Crowd-ML, a participant checks out the global classifier parameters from the coordinator and computes the gradients using its own training data. Then, the participants transmit the gradients to the coordinator that will update the global classifier parameters. Thus, during the training phase, the participants and the coordinator repeatedly exchange parameters. We apply an existing implementation of Crowd-ML [1] on our testbed. Our measurement shows that, during the training phase, each participant needs to upload and download a total of 117.2 MB data, which is 3.5x of our GRP-DNN. The participant compute time is more than 350x of that under GRP-DNN. Despite the larger volume of data exchanges, Crowd-ML achieves 91.28% test accuracy only, which is lower than the 95.58% test accuracy achieved by GRP-DNN. This is because Crowd-ML uses a simple multiclass logistic classifier, which is inferior compared with the CNN used by GRP-DNN in terms of learning performance. Note that during the testing phase of Crowd-ML, the participants execute their local classifiers. Thus, they do not need to transmit the testing samples to the coordinator for classification.

CryptoNets [19] uses homomorphic encryption algorithm to encrypt a testing sample during the classification phase and transmits the encrypted sample to the coordinator. Then, the coordinator uses a neural network trained with plaintext data to classify the encrypted testing sample. We have implemented the homomorphic encryption part of CrytoNets that runs on the Raspberry Pis. The volume of the 714 encrypted testing images is 15 MB, almost 3x of the data volume generated by random projection. In particular, a Raspberry Pi node takes about 10 minutes and a total of 116 hours

to encrypt an image and all the testing images, respectively. This is 2.6 million times slower than the random projection computation. This result clearly shows that the high computation complexity of the homomorphic encryption makes CryptoNets ill-suited for resource-constrained devices.

7 CONCLUSION

This paper proposes a practical privacy-preserving collaborative learning approach, in which the resource-constrained learning participants apply independent Gaussian projections on their training data vectors and the coordinator applies deep learning to train a classifier based on the projected data vectors. Our approach protects the confidentiality of the raw forms of the training data against the honest-but-curious coordinator. Evaluation using two datasets shows that our approach outperforms various baselines and exhibits promising scalability with respect to the number of participants observing low-complexity data patterns. Benchmark on a testbed shows the practicality and efficiency of our approach.

ACKNOWLEDGMENTS

This research was funded by a Start-up Grant at Nanyang Technological University. We acknowledge the support of NVIDIA Corporation with the donation of two GPUs used in this research. We also acknowledge Dr. Yi Li for constructive discussions and Zhenyu Yan for managing the computation resources used in this paper.

REFERENCES

[1] 2018. Crowd-ML. https://github.com/jihunhamm/Crowd-ML.
[2] 2018. PyTorch. https://pytorch.org/.
[3] 2018. Raspberry Pi 2 Model B. https://bit.ly/1b75SRj.
[4] 2018. Spambase data set. https://archive.ics.uci.edu/ml/datasets/spambase.
[5] M. Abadi, A. Chu, I. Goodfellow, H. McMahan, I. Mironov, K. Talwar, and L. Zhang. 2016. Deep learning with differential privacy. In *Proc. CCS*. ACM, 308–318.
[6] Nir Ailon and Bernard Chazelle. 2009. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing* 39, 1 (2009), 302–322.
[7] Jonathan Berr. 2018. Equifax breach exposed data for 143 million consumers. https://cbsn.ws/2Qc8VOg.
[8] Michel Bierlaire, Ph L Toint, and Daniel Tuyttens. 1991. On iterative algorithms for linear least squares problems with bound constraints. *Linear Algebra Appl.* 143 (1991), 111–143.
[9] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy preserving machine learning. In *Proc. CCS*. ACM, 1175–1191.
[10] Avishek Joey Bose and Parham Aarabi. 2018. Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization. In *Proc. Intl. Workshop Multimedia Signal Process.*
[11] Emmanuel J Candès and Michael B Wakin. 2008. An introduction to compressive sampling. *IEEE Signal Process. Mag.* 25, 2 (2008), 21–30.
[12] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. 2017. Privacy-Preserving Classification on Deep Neural Network. *IACR Cryptology ePrint Archive* 2017 (2017), 35.
[13] Chih-Chung Chang and Chih-Jen Lin. 2018. LIBSVM – a library for support vector machines. https://www.csie.ntu.edu.tw/~cjlin/libsvm/.
[14] Kamalika Chaudhuri and Claire Monteleoni. 2009. Privacy-preserving logistic regression. In *Proc. NIPS*. 289–296.
[15] Zizhong Chen and Jack J Dongarra. 2005. Condition numbers of Gaussian random matrices. *SIAM J. Matrix Anal. Appl.* 27, 3 (2005), 603–620.
[16] George Danezis and Claudia Diaz. 2008. *A survey of anonymous communication channels.* Technical Report. Microsoft Research. MSR-TR-2008-35.
[17] C. Dwork. 2006. Differential privacy. In *Proc. ICALP.*
[18] C. Dwork, F. McSherry, K. Nissim, and A. Smith. 2006. Calibrating noise to sensitivity in private data analysis. *Conf. Theory of Cryptography* (2006), 265–284.
[19] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proc. ICML*. 201–210.
[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *Proc. ICLR.*
[21] Google Cloud. 2018. Edge TPU. https://cloud.google.com/edge-tpu/.
[22] Thore Graepel, Kristin Lauter, and Michael Naehrig. 2012. ML confidential: Machine learning on encrypted data. In *Proc. Intl. Conf. Inf. Security & Cryptology*. Springer, 1–21.
[23] J. Hamm, A. Champion, G. Chen, M. Belkin, and D. Xuan. 2015. Crowd-ML: A Privacy-Preserving Learning Framework for a Crowd of Smart Devices. In *Proc. ICDCS*. IEEE, 11–20.
[24] B. Hitaj, G. Ateniese, and F. Perez-Cruz. 2017. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *Proc. CCS*. ACM, 603–618.
[25] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proc. MobiSys*. ACM, 82–95.
[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.
[27] Yann LeCun, Corinna Corts, and Christopher J.C. Burges. 2018. The MNIST Database of Handwritten Digits. http://yann.lecun.com/exdb/mnist/.
[28] Shancang Li, Li Da Xu, and Xinheng Wang. 2013. Compressed sensing signal and data acquisition in wireless sensor networks and internet of things. *IEEE Trans. Ind. Informat.* 9, 4 (2013), 2177–2186.
[29] Bin Liu, Yurong Jiang, Fei Sha, and Ramesh Govindan. 2012. Cloud-enabled privacy-preserving collaborative learning for mobile sensing. In *Proc. SenSys*. ACM, 57–70.
[30] Kun Liu, Hillol Kargupta, and Jessica Ryan. 2006. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Trans. knowl. Data Eng.* 18, 1 (2006), 92–106.
[31] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.
[32] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning Differentially Private Recurrent Language Models. In *Proc. ICLR.*
[33] Arvind Narayanan and Vitaly Shmatikov. 2006. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105* (2006).
[34] Lindsey O'Donnell. 2018. Zero-Day Flash Exploit Targeting Middle East. https://threatpost.com/zero-day-flash-exploit-targeting-middle-east/132659/.
[35] Christopher C Paige and Michael A Saunders. 1982. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software* 8, 1 (1982), 43–71.
[36] L. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai. 2018. Privacy-Preserving Deep Learning via Additively Homomorphic Encryption. *IEEE Trans. Inf. Forensics Security* 13, 5 (2018).
[37] Yaron Rachlin and Dror Baron. 2008. The secrecy of compressed sensing measurements. In *Proc. Allerton*. IEEE, 813–817.
[38] Reuters. 2018. Facebook critics want regulation, investigation after data misuse. https://reut.rs/2GwKF8p.
[39] Yiran Shen, Chengwen Luo, Dan Yin, Hongkai Wen, Rus Daniela, and Wen Hu. 2018. Privacy-preserving sparse representation classification in cloud-enabled mobile applications. *Comput. Netw.* 133 (2018), 59–72.
[40] R. Shokri and V. Shmatikov. 2015. Privacy-preserving deep learning. In *Proc. CCS*. ACM, 1310–1321.
[41] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *Proc. GlobalSIP*. IEEE, 245–248.
[42] Johan AK Suykens. 2003. *Advances in learning theory: methods, models, and applications*. Vol. 190. IOS Press.
[43] Rui Tan, Sheng-Yuan Chiu, Hoang Hai Nguyen, David KY Yau, and Deokwoo Jung. 2017. A Joint Data Compression and Encryption Approach for Wireless Energy Auditing Networks. *ACM Trans. Sensor Networks* 13, 2 (2017), 9.
[44] Cong Wang, Bingsheng Zhang, Kui Ren, and Janet M Roveda. 2013. Privacy-assured outsourcing of image reconstruction service in cloud. *IEEE Trans. Emerg. Topics Comput.* 1, 1 (2013), 166–177.
[45] Piotr Iwo Wójcik and Marcin Kurdziel. 2018. Training neural networks on high-dimensional data using random projection. *Pattern Anal. Appl.* (2018), 1–11.
[46] Wanli Xue, Chenwen Luo, Guohao Lan, Rajib Rana, Wen Hu, and Aruna Seneviratne. 2017. Kryptein: a compressive-sensing-based encryption scheme for the internet of things. In *Proc. IPSN*. IEEE, 169–180.
[47] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. 2017. DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In *Proc. SenSys*. ACM, 4:1–4:14.
[48] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proc. CVPR*. IEEE, 4480–4488.