PADL: Privacy-Aware and Asynchronous Deep Learning for IoT Applications

Xiaoyuan Liu, *Student Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Guowen Xu, *Student Member, IEEE*, Sen Liu, *Student Member, IEEE*, Zhe Liu, *Senior Member, IEEE*, and Rongxing Lu, *Senior Member, IEEE*

***Abstract*—As a promising data-driven technology, deep learning has been widely employed in a variety of Internet-of-Things (IoT) applications. Examples include automated navigation, telemedicine, and smart home. To protect the data privacy of deep-learning-based IoT applications, a few privacy-preserving approaches have also been exploited, designed, and implemented in various scenarios. However, state-of-the-art works are still defective in accuracy, efficiency, and functionality. In this article, we propose the privacy-aware and asynchronous deep-learning-assisted IoT applications (PADL), a privacy-aware and asynchronous deep learning framework that enables multiple data collecting sites to collaboratively train deep neural networks (DNNs), while keeping the confidentiality of private data to each other. Specifically, we first design a layerwise importance propagation (LIP) algorithm to quantify the importance of the model's weights held by each site. Then, we present the customized perturbation mechanism, a precise combination of the LIP algorithm and differential privacy mechanism, which helps to make optimal tradeoffs between the availability and privacy of local models. Furthermore, to fully use the computing resources of all sites, for the first time, we propose an advanced asynchronous optimization (AAO) protocol to perform global updates without waiting. Theoretical analysis shows that the PADL is robust to extreme collusion even with only one reliable site while supporting lock-free optimization. Finally, extensive experiments conducted on real-world data sets using TensorFlow library show that the PADL outperforms the existing systems in terms of efficiency and prediction accuracy.**

***Index Terms*—Deep learning, differential privacy, Internet of Things (IoT), privacy.**

Manuscript received November 18, 2019; revised February 28, 2020; accepted March 1, 2020. Date of publication March 17, 2020; date of current version August 12, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802300 and Grant 2017YFB0802000; in part by the National Natural Science Foundation of China under Grant 61802051, Grant 61772121, Grant 61728102, Grant 61972094, and Grant 61472065; in part by the Peng Cheng Laboratory Project of Guangdong Province under Grant PCL2018KP004; and in part by the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201804. *(Corresponding author: Hongwei Li.)*

Xiaoyuan Liu, Hongwei Li, Guowen Xu, and Sen Liu are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: xiaoyuan.l@foxmail.com; hongweili@uestc.edu.cn; guowen.xu@foxmail.com; 893551724@qq.com).

Zhe Liu is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China (e-mail: zhe.liu@nuaa.edu.cn).

Rongxing Lu is with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada (e-mail: rlu1@unb.ca).

Digital Object Identifier 10.1109/JIOT.2020.2981379

I. Introduction

With the rapid development of the Internet and traditional telecommunication networks, abundant of terminal devices are accessing the network and generate massive amounts of data every day [1]–[3]. Meanwhile, with the advantage of intelligent decisions driven by big data, deep-learning-based technologies have been widely applied to assist the Internet-of-Things (IoT) applications [4]. In particular, deep-learning-based IoT applications have recently demonstrated superior performance in various fields, e.g., smart city [5], [6], smart home [7], personalized healthcare system [8], and autonomous vehicles (AVs) [9].

However, for training a deep neural network (DNN), a large amount of data should be collected from various IoT devices, which usually includes sensitive information about users. Under such circumstances, it is obvious that the data owners will lose control of their data after "sharing" them. Interest-driven adversaries could induce irreversible damage to users with private information. For instance, deep learning models can predict patients' health status based on physiological data, such as pulse, temperature, and blood pressure (BP) collected by wearable IoT devices [10]–[13]. The leakage of these private data may result in huge economic losses to data owners, even endanger their lives [14]. In the field of AVs, a deep-learning-based intelligent decision system may be interfered with malicious adversaries, who can acquire the users' location privacy. The result will cause life-threatening traffic safety problems and bring anxiety to society [15]. Therefore, there is no doubt that protecting the privacy of users' data is a fundamental issue in the process of deep-learning-assisted IoT applications.

To address such privacy concerns [16]–[22], a few studies focusing on privacy-preserving deep learning have been proposed, the technologies can be categorized into the following three types, i.e., federated learning [23], [24], encryption-based technologies [25]–[28], and differential privacy [29], [30]. Federated learning refers to that multiple participants collaboratively learn a unified model, and the local gradients uploaded by each participant are aggregated through the synchronous aggregation rules of the cloud server [31]. However, Hitaj *et al.* [32] suggested that attackers can still recover users' private data through local gradients. Moreover, encryption-based technologies, such as homomorphic encryption and secret sharing, usually require additional overhead

to implement. Specifically, homomorphic encryption can be used to perform calculations in the ciphertext domain [33]. Additional encryption and decryption calculations are essential. Hence, homomorphic encryption is not suitable for high-dimensional data and large-scale participants. The other is secret sharing, which requires multiple rounds of communication between participants. In this mode, participants should keep online, which does not apply to resources-limited devices in IoT applications. Compared with encryption-based technologies, differential privacy has low communication and computation costs [34]. Nevertheless, there is no way to provide privacy protection for free. Achieving differential privacy always requires noise, which drops the accuracy of the system. Therefore, when traditional differential privacy meets deep learning, the utility–privacy tradeoff ought to be made.

In addition to the methodological shortcomings discussed above, none of the existing schemes take into consideration asynchronous optimization, which generally comprises asynchronous learning and asynchronous updates. Implementing asynchronous optimization usually divides the training into multiple submodules, which are performed in parallel by multiple entities. Compared with federated learning that only supports asynchronous learning, asynchronous optimization is featured with the following advantages:

1) markedly improves the efficiency of the training due to the asynchronous updates;
2) strikingly increases the accuracy of the model due to the proportional increase in fresh data per unit time;
3) effectively solves the problem of data and update loss during the training process.

Also, it significantly increases the robustness of the system, because traditional deep learning requires training data to be concentrated in one data center, which can lead to more serious losses in the case of a single-point failure of the data center, while only a portion of the training data is leaked in an asynchronously optimized system.

From the above, it is meaningful and urgent to design a privacy preserving, lightweight solution while supporting asynchronous optimization for IoT applications. In this article, we propose the privacy-aware and asynchronous deep-learning-assisted IoT applications (PADL) for IoT applications, the first privacy-aware deep learning framework while supporting asynchronous optimization. Specifically, before uploading the local model to the cloud server, each data collection site calculates the importance of each weight by the layerwise importance propagation (LIP) algorithm. Then, they adaptively perturb the weights for protecting the privacy of training data. Finally, an advanced asynchronous optimization (AAO) protocol is designed to orderly process global updates. Specifically, the contributions of the PADL can be summarized as follows.

1) We design an LIP algorithm, that combines with differential privacy to form a novel perturbation mechanism. Compared to the traditional differential privacy mechanism, our approach offers amazing advantages in terms of prediction accuracy.
2) We propose the first AAO protocol, which achieves completely lock-free and noninteractive optimization in high-throughput IoT applications. Moreover, this kind of deployment can also significantly improve prediction accuracy.
3) Security analysis proves that it is intractable to reverse the training data even in the extreme case of the collusion with only one reliable site in our PADL. Besides, extensive experiments conducted on real-world data sets show that the PADL outperforms the existing systems in terms of efficiency and prediction accuracy.

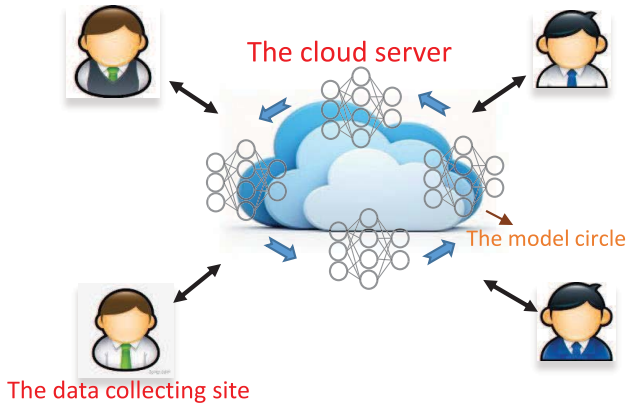


Fig. 1. System model: the cloud server holds a circle of M deep learning models. Multiple data collecting sites train model over local training data.

The remainder of this article is organized as follows. In Section II, we start by providing the problem statement and prerequisites. Subsequently, Section III introduces our novel PADL in detail and Section IV carries out the properties analysis. What is more, Section V includes experimental analysis in the performance and efficiency of this article. The related works are summarized in Section VI. Finally, we conclude this article in Section VII.

II. Problem Statement and Preliminaries

In this section, we first outline the system architecture, and introduce the problem in our scenario. After that, we review the main concepts of deep learning and differential privacy.

A. System Architecture

As shown in Fig. 1, our system model consists of two main components: 1) data collecting sites and 2) cloud server.

1) *Data Collecting Sites:* Each data collecting site holds a replica of DNN and a private database. They train their own models over the local database, as well as share the local models for global updates.
2) *Cloud Server:* The cloud server works as an "exchanger," who receives the perturbed models from the data collecting sites and returns another model to them.

B. Threat Model and Privacy Requirements

In this article, we consider that both entities, i.e., the cloud and the data collecting sites, are "honest-but-curious," which means that they correctly execute the protocols while keeping curious to users' private data. Besides, both of them may collude to eavesdrop particular users' data privacy. From the knowledge of the adversary perspective, the data collecting

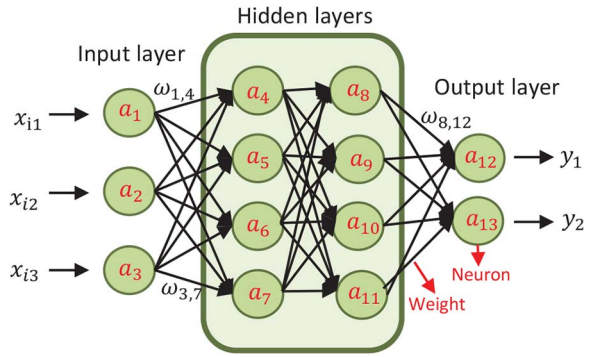


Fig. 2. DNN: DNN composes of the input layer, hidden layers, and output layer.

sites and the cloud server do not only have access to the trained model but may also have the full knowledge of the training mechanism.

Under the above threat model, we provide a rigorous privacy guarantee to the training data of the data collecting sites while still ensuring that each site can benefit from other sites' models. Besides, asynchronous sites-server communication not only contributes to efficiency processing real-time data stream in IoT applications but also provides an additional gain in privacy, i.e., even the server colludes with $n-1$ out of n sites, the remaining site' private information still cannot be eavesdropped.

C. Deep Learning and Deep Neural Network

DNNs, as the infrastructure of deep learning models, are extremely effective for solving many complex deep learning tasks [35]. As shown in Fig. 2, the structure of DNN consists of one input layer, two hidden layers, and one output layer, and each layer is connected by the weights ω. In this article, we focus on the representative convolutional neural network (CNN), which can efficiently process high-dimensional data.

Our model focuses on a supervised setting, where the training data consist of a feature vector and corresponding data label, i.e., $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^v$ are v-dimensional feature vector. Each neuron receives the output of neurons in the previous layer along with the weights. Assuming that the neurons in the input layer receive 3-D training data $x_i = \{x_{i1}, x_{i2}, x_{i3}\}$, the weights are $\omega = \{\omega_{1,4}, \omega_{2,4}, \omega_{3,4}\}$ between neuron a_4 and the neurons in the input layer. The neuron a_4 outputs $\mathcal{F}(\sum_{j=1}^{j=3} \omega_{j,4} * x_j)$, where $\mathcal{F}()$ is an activation function, e.g., $\text{sigmoid}(x) = [1/(1+e^{-x})]$. The outputs of the model refer to the outputs of the neurons on the output layer, expressed as $\boldsymbol{\mathcal{F}}(x_i, \omega)$. The loss function ℓ is defined to penalize the difference of the model output and the true label, e.g., $\ell(\boldsymbol{\mathcal{F}}(x_i, \omega), y_i) = (\boldsymbol{\mathcal{F}}(x_i, \omega) - y_i)^2$.

Subsequently, we adopt the mini-batch stochastic gradient descent algorithm to optimize the model, which is one of the most popular ways to iteratively adjust the weights for minimizing the loss ℓ. Specifically, for the rth iteration, the partial derivatives $\nabla\mathcal{L}(\mathbb{D}, \omega^r)$ of the loss function to the weights, also called gradients, are calculated. Along with the opposite direction of the gradients, the weights ω^r are updated as follows:

$$\omega^{r+1} = \omega^r - \eta \times \nabla\mathcal{L}\big(\mathbb{D}, \omega^r\big)$$

where η represents the learning rate. For efficiency reasons, the more common practice is to randomly select mini-batch subset of data $\mathbb{B}^r$ ($\mathbb{B}^r \in \mathbb{D}$) to estimate $\nabla\mathcal{L}(\mathbb{D}, \omega^r)$ for each iteration, as follows:

$$\overline{\nabla\mathcal{L}(\mathbb{B}^r, \omega^r)} = \frac{1}{|\mathbb{B}^r|} \sum_{(x_i, y_i) \in \mathbb{B}^r} \nabla\ell\big(\boldsymbol{\mathcal{F}}\big(x_i, \omega^r\big), y_i\big).$$

D. Differential Privacy

Dwork [34] designed differential privacy which is a probabilistic mechanism to provide theoretically provable privacy guarantee.

Definition 1 [(ε, δ)-Differential Privacy]: Given any neighboring data sets $\mathbb{D}_1$ and $\mathbb{D}_2$, which differ by at most one record, a randomized algorithm Γ preserves (ε, δ)-differential privacy if

$$\forall Y \subseteq \text{Range}(\Gamma) : \Pr[\Gamma(\mathbb{D}_1) \in Y] \leq e^{\varepsilon} \times \Pr[\Gamma(\mathbb{D}_2) \in Y] + \delta$$

where $\text{Range}(\Gamma)$ represents all possible outputs of the algorithm Γ. ε denotes the privacy budget which restricts the privacy guarantee level of algorithm Γ. δ ($\delta \geq 0$) is the failure probability that allows differential privacy to fail. A tighter bound of differential privacy is achieved when $\delta = 0$. We limit the failure possibility of differential privacy to 0 for obtaining a strictly strong privacy guarantee. Therefore, we use ε-differential privacy in this article.

The sequential composition and parallel composition of differential privacy are very useful in the design of multistep mechanisms [36], as defined in Theorems 1 and 2.

Theorem 1 (Sequential Composition): Given algorithm Γ_1, $\Gamma_2, \ldots, \Gamma_k$ that satisfy ε_1-differential privacy, ε_2-differential privacy, $\ldots$, ε_k-differential privacy, respectively, we have $\Gamma(\mathbb{D}) = \langle\Gamma_1(\mathbb{D}), \Gamma_2(\mathbb{D}), \ldots, \Gamma_k(\mathbb{D})\rangle$ satisfies $(\sum_{i=1}^{k} \varepsilon_i)$-differential privacy.

Theorem 2 (Parallel Composition): Given algorithm Γ_1, $\Gamma_2, \ldots, \Gamma_k$ that satisfy ε_1-differential privacy, ε_2-differential privacy, $\ldots$, ε_k-differential privacy, respectively. $\mathbb{D}_1$, $\mathbb{D}_2, \ldots, \mathbb{D}_k$ are the deterministic partitioning of data set $\mathbb{D}$, $\Gamma(\mathbb{D}) = \langle\Gamma_1(\mathbb{D}_1), \Gamma_2(\mathbb{D}_2), \ldots, \Gamma_k(\mathbb{D}_k)\rangle$ will provide $(\max_{i\in[1,k]} \varepsilon_i)$-differential privacy.

The Laplace mechanism is the main technology to implement ε-differential privacy, which perturbs the true value by adding noise sampled from the Laplace distribution

$$\widehat{\Gamma(\mathbb{D})} = \Gamma(\mathbb{D}) + \text{Laplace}\left(\frac{\Delta f}{\varepsilon}\right)$$

where $\text{Laplace}(\Delta f/\varepsilon)$ is a random value sampled from Laplace distribution, whose mean is 0 and the scale is $(\Delta f/\varepsilon)$. The noise is proportional to the sensitivity Δf of the algorithm Γ, where $\Delta f = \max_{\mathbb{D}_1, \mathbb{D}_2} \|\Gamma(\mathbb{D}_1) - \Gamma(\mathbb{D}_2)\|_1$ is the maximal difference of the algorithm $\Gamma()$ over the neighboring data sets $\mathbb{D}_1$ and $\mathbb{D}_2$. After the perturbation, the probability of the algorithm $\widehat{\Gamma()}$ outputting Y over $\mathbb{D}_1$ and $\mathbb{D}_2$ is $(1/2\lambda)e^{[(-|Y-\Gamma(\mathbb{D}_1)|)/\lambda]}$

TABLE I
PARAMETERS

Symbol	Description
$\mathbb{D}$	Data set
$\mathbb{B}$	Subset of data set
x	Vector of data feature
y	Data label
ω	Weight
$\widehat{\omega}$	Weight after perturbing
η	Learning rate
r	Training termination symbol
$\mathcal{I}$	Importance vector
γ	The total number of gradients in model
ε	Privacy budget
Δf	Sensitivity
∇g	The truncated gradient
α	The norm bound of gradients

and $(1/2\lambda)e^{[(-|Y-\Gamma(\mathbb{D}_2)|)/\lambda]}$, respectively, where $\lambda = (\Delta f/\varepsilon)$. The ratio of the probability is

$$\ln \frac{e^{\frac{-|Y-\Gamma(\mathbb{D}_1)|}{\lambda}}}{e^{\frac{-|Y-\Gamma(\mathbb{D}_2)|}{\lambda}}} \leq \frac{|\Gamma(\mathbb{D}_1) - \Gamma(\mathbb{D}_2)|}{\lambda} \leq \varepsilon$$

which is also known as privacy loss. We can see that a smaller privacy budget ε represents a higher level of privacy protection, and *vice versa*.

III. Proposed Scheme

In this section, we give a detailed description of the PADL from the perspective of the data collection site and the cloud server.

A. Overview

The goal of the PADL is to efficiently train over real-time data stream in IoT applications while providing privacy guarantees to local private data.

To achieve that, we adopt differential privacy to protect the privacy of data for efficiency reasons. To address the shortcomings of differential privacy: the reduction in accuracy, we propose an LIP algorithm to calculate the importance of model weights on the output. Then, we adaptively perturb the model's weight values by combining differential privacy and the LIP algorithm. Besides, based on the nature of IoT applications: high-throughput data and devices with varying computing power, we design an AAO protocol to orderly process global updates.

For ease of reference, Table I lists the symbols appeared in this article and their descriptions.

B. Data Collecting Sites

In IoT applications, each data collecting site continuously takes over and processes real-time data from a large number of different IoT sensors, and then feeds them back to the current task, i.e., to train high-accurate large DNN models over local data at low latency. Meanwhile, the data collection sites require not only the ability to run deep learning training missions but also the ability to support bidirectional communication with the cloud server for uploading and downloading the parameters. Also, for privacy and efficiency reasons, the data collection site provides privacy protection to the parameters of the models and training data via differential privacy. However, traditional practices to directly add Laplace noise inevitably sacrifice the accuracy of the system.

To solve the problem of accuracy degradation caused by differential privacy, we propose a customized perturbation mechanism to improve the model accuracy at the same level of privacy. First, before uploading the local model to the cloud server, we quantify the importance of weights in the current model by the LIP algorithm. Then, we creatively combine the differential privacy mechanism with the LIP algorithm in an application-specific manner, which filters out superfluous noise to provide higher prediction accuracy at the same privacy budget. Specifically, according to the importance of weights, less noise will be injected into the weights for improving accuracy when the importance is high. While there is lower importance, the more noise will be allocated for protecting privacy.

1) Layerwise Importance Propagation Algorithm: By default, the weights have varying importance to the model. In the following phase, the parameter $\omega_{p,q}$ denotes the weight between neurons a_p and a_q in adjacent layers, and the element $\mathcal{I}_{p,q}$ of the importance vector $\mathcal{I}$ represents the importance of the weight $\omega_{p,q}$. We calculate the importance of neurons as intermediate values for conveniently calculating the importance vector $\mathcal{I}$. In addition, assuming that the training termination symbol for each data collection site is r.

The LIP algorithm consists of the *preparation phase* and *quantification phase* as follows.

Preparation Phase:

1) Each data collection site downloads a new model from the cloud server, and trains it over local database until r iterations. Particularly, the mini-batch stochastic gradient descent method [37] is also used to optimize the model as many state-of-the-art works [38], [39]. In this phase, we make some minor adjustments, that clips the l_2 norm of each gradient [40]. Each data collecting site sets a norm bound α of gradients. Then, they bound the gradients as follows:

$$\nabla g_{[\omega_{p,q}]} \leftarrow \overline{\nabla \mathcal{L}\left(\mathbb{B}, \omega_{p,q}\right)} / \max\left(1, \frac{\left\|\overline{\nabla \mathcal{L}\left(\mathbb{B}, \omega_{p,q}\right)}\right\|_2}{\alpha}\right). \tag{1}$$

The weight $\omega_{p,q}$ is updated until r iterations as follows:

$$\omega_{p,q}^{t+1} = \omega_{p,q}^{t} - \eta \times \nabla g_{\left[\omega_{p,q}^{t}\right]} \tag{2}$$

where $t \in [0, r]$.

2) Each data collecting site initializes the importance vector $\mathcal{I}$ to zero vector, which is a γ-dimensional vector, where γ is the total number of gradients.

Algorithm 1 Pseudocode of the LIP Algorithm

Input: Given "new" model ω, private database: $\mathbb{D}$, where $\langle x_i, y_i \rangle \in \mathbb{D}$.

Output: The importance vector $\mathcal{I}$.

for each round $t = 1, 2, ..., r$ **do**
  $\mathbb{B}^t \leftarrow$ (Random set of $\mathbb{D}$)
  $\omega_{p,q}^{t+1} = \omega_{p,q}^{t} - \eta \times \nabla g_{[\omega_{p,q}^{t}]}$
end for
if neuron a_p is in the output layer: **then**
  $i_{a_p}^{l_o} \leftarrow \boldsymbol{\mathcal{F}}_{a_p}(x_i, \omega)$
end if
for each layer **do**
  if $\sum_{a_p \in l_{h-1}} |a_p w_{p,q}| = 0$ **then**
    $\mathcal{I}_{p,q} \leftarrow 0$
  else
    $\mathcal{I}_{p,q} \leftarrow \frac{|a_p w_{p,q}|}{\sum_{a_p \in l_{h-1}} |a_p w_{p,q}|} i_{a_q}^{l_h}$
  end if
  $i_{a_p}^{l_{h-1}} = \sum_{a_q \in l_h} \mathcal{I}_{p,q}$
end for
Return the importance vector $\mathcal{I}$

Quantification Phase:

1) Starting from the neuron a_p in the output layer l_o, whose importance i_{a_p} is equal to its output value as follows:

$$i_{a_p}^{l_o} = \boldsymbol{\mathcal{F}}_{a_p}(x_i, \omega). \tag{3}$$

2) For the weight $\omega_{p,q}$ in adjacent layers, e.g., the $(h-1)$th and the hth layers, whose importance $\mathcal{I}_{p,q}$ is as follows:

$$\mathcal{I}_{p,q} = \begin{cases} \frac{|a_p \omega_{p,q}|}{\sum_{a_p \in l_{h-1}} |a_p \omega_{p,q}|} i_{a_q}^{l_h} & \sum_{a_p \in l_{h-1}} |a_p \omega_{p,q}| \neq 0 \\ 0 & \sum_{a_p \in l_{h-1}} |a_p \omega_{p,q}| = 0 \end{cases} \tag{4}$$

where a_p in the formula represents the output value of the neuron a_p.

3) Except the output layer, for the neuron a_p in the $(h-1)$th layer, whose importance i_{a_p} is as follows:

$$i_{a_p}^{l_{h-1}} = \sum_{a_q \in l_h} \mathcal{I}_{p,q} \tag{5}$$

4) Repeat steps 2) and 3) until $h = 1$.

When $h = 1$, all the weights in the model have been calculated layer by layer. The pseudocodes of the LIP algorithm is given as Algorithm 1.

2) Customized Perturbation Mechanism: By creatively combining the differential privacy mechanism with the LIP algorithm in an application-specific manner, we provide a novel customized perturbation mechanism. This mechanism can mitigate the negative effect brought by differential privacy on the system accuracy, so that the utility of the model is dramatically improved.

First, we normalize the importance vector by

$$\tilde{\mathcal{I}}_{p,q} \leftarrow \frac{\mathcal{I}_{p,q} - \mathcal{I}_{\min}}{2(\mathcal{I}_{\max} - \mathcal{I}_{\min})} + 0.5 \tag{6}$$

which limits $\mathcal{I}_{p,q}$ into the interval [0.5, 1.0], where $\mathcal{I}_{\min}$ and $\mathcal{I}_{\max}$ are the minimum and maximum values in the importance vector $\mathcal{I}$, respectively. We introduce privacy budget $\varepsilon_{p,q}$ for each weight as follows:

$$\varepsilon_{p,q} \leftarrow \frac{\tilde{\mathcal{I}}_{p,q}}{\sum \tilde{\mathcal{I}}_{p,q}} \times \varepsilon_T \tag{7}$$

where ε_T represents the total privacy budget. $\varepsilon_{p,q}$ can be considered as a shuffle of the Laplace noise, which transfers the noise from more important items to fewer ones.

Then, before submitting the whole model to the cloud server, each weight is perturbed by noise as follows:

$$\widehat{\omega_{p,q}^{r}} = \omega_{p,q}^{r} + \text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right) \tag{8}$$

where the sensitivity $\Delta f = 2\alpha \times \eta$.

3) Request Model: After uploading the current training model, the data collecting site requests a new training model from the cloud server for the next training.

Algorithm 2 Pseudocode of the AAO Protocol

for each data collecting site **do**
  Receive a new model
  if the model m (mod M) in cloud have never been initialized **then**
    Replace the model m (mod M)
    Inform the data collecting site to continue to train with the current model
  else
    Replace the model $m + 1$ (mod M) in cloud
    Return the model m (mod M) for model request
  end if
end for
Prediction: average the weights of M models

C. Cloud Server

In this article, we introduce an AAO protocol to achieve lock-free and asynchronous optimization. To be specific, the cloud deploys a circle of M deep learning models labeled as 0, $1, \ldots, M-1$, to orderly process the updates. Pihur *et al.* [41] designed a randomly "draw" and "discard" scheme to process the updates, however, which results in the updates to be lost. It is worth noting that the AAO protocol ensures that no updates loss occurs. Algorithm 2 presents a detailed algorithm.

1) Advanced Asynchronous Optimization Protocol: The data collecting sites upload the trained model accompanied by requesting a new training model for the next round. When the model in the server has not been initialized, it is directly replaced with the model submitted by the data collecting sites, and the data collecting site is informed to continue to train with the current model. After M models have been completely replaced once, the server returns the previous updated model labeled as m, where $m \in [0, M-1]$. With that, the model labeled as $m+1$ in the cloud is replaced by the model uploaded by the data collecting sites, where $(m+1) \in [0, M-1]$. After each interaction with the data collecting sites, the server performs $m \leftarrow m + 1 (\text{mod } M)$.

In the AAO protocol, every update can be utilized with no data loss. This seemingly simple strategy has an essential improvement in performance, efficiency, and privacy, which is analyzed and proved in Section IV.

2) Model Prediction: The server averages the weights of all deep learning models in the server to build a temporary prediction model, and always saves an optimal prediction model for prediction tasks. Specifically, first, the server calculates a temporary prediction model every M interactions with the data collecting sites. Then, the server verifies the prediction accuracy of each temporary model over the validation data set, and always saves the model with the highest prediction accuracy for prediction tasks.

In order to ensure high prediction accuracy, a supplementary condition should be added, which is $M\%2 = 0$. We consider it the synergy effect between noises.

IV. Properties Analysis

In this section, we describe the characteristics of the PADL, which make it more practical and effective in IoT applications.

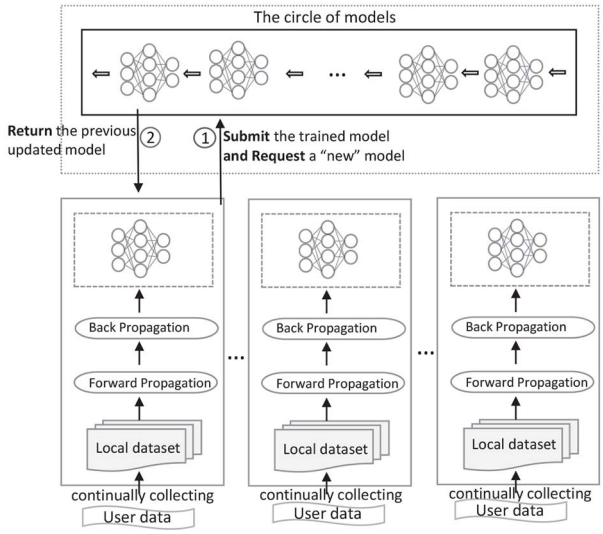


Fig. 3. AAO protocol: the top describes the circle of deep learning models in the server, and the bottoms are the training process for the data collecting sites.

A. Asynchronous Optimization

The foremost innovation of our approach comes from completely asynchronous property. The cloud server holds the circle of M models to process thousands of updates per second. In this article, the AAO protocol is very simple to implement and does not require any locking mechanism to pause the training. We explain from two aspects of *asynchronous learning* and *asynchronous update*.

1) Asynchronous Learning: The asynchronous learning can be simply thought that multiple copies of the model are trained in stand-alone data collecting sites in parallel. In this article, each data collecting site holds a replica of the DNN model, as well as a portion of training data. With the current model parameters and the private database, they train the local model independently. There is no doubt that our solution implements asynchronous learning.

2) Asynchronous Update: As shown in Fig. 3, at the beginning of the interaction between each data collecting site and the cloud server, the data collecting sites upload the trained model parameters, and require a new model for the next training. The cloud replaces the model labeled as $m+1$ with the one uploaded by data collecting site, and returns the model labeled as m for the model request for $m \in [0, M-1]$. Pihur *et al.* [41] experimentally proved that when the number of the models M in the server and the number of data collecting sites N satisfy: $M^2 = N$, asynchronous update can be completely realized. At this time, N data collecting sites can obtain the global parameters without any waiting.

Compared with the existing solutions, the AAO protocol has the following advantages.

1) Due to the lock-free mechanism, the data collecting sites can spend more time in the training deep learning model instead of waiting for averaging gradients.
2) Our solution dramatically increases data throughput for the learning tasks.
3) Each data collecting site can complete a single training with only one round of interaction with the server. Hence, each data collecting site gets fresh data stream from other sites faster than federated learning.
4) Computing power and resources can be fully utilized for each data collecting site.
5) There are no data or update loss in the PADL, which greatly increases the availability and reliability of the system.

B. Privacy

In this article, only the privacy of training data is considered. We acknowledge that the integrity and availability of the scheme will be compromised when malicious adversaries conduct security attacks [16]–[19], such as a poisoning attack [18] or an inference attack [42]. Even so, we still claim that our approach provides a rigorous privacy guarantee to the training data, and the local model theoretically satisfies ε-differential privacy. What is more, the PADL still is robustness, even in the case of extreme collusion, where only one honest and legitimate data collecting site exists.

1) Differential Privacy: According to the result of the LIP algorithm, i.e., the importance vector I, we adaptively inject Laplace noise into the model parameters before sending them to the server, which makes the local model satisfies ε-differential privacy.

Lemma 1: Assuming that two neighboring data sets $\mathbb{D}_1$ and $\mathbb{D}_2$ which only differ in one record. After r iterations, our model parameters are $\omega_{p,q}^r = \omega_{p,q}^{r-1} - \eta \times \nabla g_{[\omega_{p,q}^{r-1}]}$. Before sending them to the cloud server, let

$$\widehat{\omega_{p,q}^r} = \omega_{p,q}^r + \text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right)$$

where the sensitivity $\Delta f = 2\alpha \times \eta$. Then, the local model satisfies ε-differential privacy.

Proof: Given the bounded gradient $\nabla g_{[\omega]}$, which is limited to the interval $[-\alpha, \alpha]$. The weight $\omega_{p,q}$ is updated with the gradient $\nabla g_{[\omega_{p,q}]}$ based on (2). The sensitivity Δf can be calculated as follows:

$$\begin{aligned}\Delta f &= \max_{\mathbb{D}_1,\mathbb{D}_2} \left\| \omega_{p,q}(\mathbb{D}_1) - \omega_{p,q}(\mathbb{D}_2) \right\|_1 \\ &= 2 \times \eta \times \max \left\| \nabla g_{[\omega_{p,q}]} \right\|_1 \\ &= 2\alpha \times \eta.\end{aligned}$$

Then, we have

$$\begin{aligned}\frac{\Pr\left(\widehat{\omega_{p,q}(\mathbb{D}_1)} = Y\right)}{\Pr\left(\widehat{\omega_{p,q}(\mathbb{D}_2)} = Y\right)} &= \frac{\Pr\left(\omega_{p,q}(\mathbb{D}_1) + \text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right) = Y\right)}{\Pr\left(\omega_{p,q}(\mathbb{D}_2) + \text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right) = Y\right)} \\ &= \frac{\Pr\left(\text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right) = Y - \omega_{p,q}(\mathbb{D}_1)\right)}{\Pr\left(\text{Laplace}\left(\frac{\Delta f}{\varepsilon_{p,q}}\right) = Y - \omega_{p,q}(\mathbb{D}_2)\right)} \\ &= \frac{\frac{\varepsilon_{p,q}}{2\times\Delta f} \times e^{\frac{-\left|Y-\omega_{p,q}(\mathbb{D}_1)\right|\times\varepsilon_{p,q}}{\Delta f}}}{\frac{\varepsilon_{p,q}}{2\times\Delta f} \times e^{\frac{-\left|Y-\omega_{p,q}(\mathbb{D}_2)\right|\times\varepsilon_{p,q}}{\Delta f}}} \\ &\le e^{\frac{\varepsilon_{p,q}\times\left|\omega_{p,q}(\mathbb{D}_1)-\omega_{p,q}(\mathbb{D}_2)\right|}{\Delta f}} \\ &\le e^{\varepsilon_{p,q}}.\end{aligned}$$

Consequently, we get the knowledge that the weight $\omega_{p,q}$ $(p, q \in [0, \gamma - 1])$ satisfies $\varepsilon_{p,q}$-differential privacy. As mentioned in Section II-D, the composition property of differential privacy, the entire model satisfies ε_T-differential privacy. ■

2) Against Honest-But-Curious Cloud Server: The cloud server is viewed as honest-but-curious: on the one hand, it faithfully and correctly executes the protocols in the system. On the other hand, it is curious to users' private data thus violating the intent of the legal data collecting sites.

Differential privacy is a powerful privacy concept, which can be used to limit the disclosure of private information of records stored in a database. When it is used to protect privacy for deep learning, the purpose usually is to maintain the privacy of training data. In this article, each model in the cloud server satisfies ε_m-differential privacy, where $m \in [0, M-1]$. The parameter ε_m represents the privacy budget of the mth model in the circle, which is replaced by the local model preserving differential privacy. The private database of each data collecting site can be considered as a partition of all training data. According to the parallel composition of differential privacy discussed in Section II-D, the circle in the server satisfies $\max_{m\in[0,M-1]}\{\varepsilon_m\}$-differential privacy.

To sum up, our solution holds the privacy property that honest-but-curious server learning nothing about training data.

3) Against the Collusion Between Malicious Participants: In this setting, a malicious adversary compromises multiple participants in the protocol, they aim to eavesdrop the private information of particular participants. Nevertheless, nothing they can obtain about other sites.

Since the data collecting sites asynchronously train the models, the interaction with the server cannot be accurately predicted. For instance, the model requested by the malicious parties cannot be determined from which data collecting sites, and other information such as the interaction time with the server, or which model in the server is replaced. Any data collecting sites can appear in the following steps, and maybe repeat more than once, or never participate in. The global model update steps are as follows:

$$\begin{aligned}\omega_0^{\text{Server}} &\leftarrow \widehat{\omega_0^r} = \omega_0^r + \text{Laplace}\left(\frac{\Delta f_0}{\varepsilon_0}\right) \\ &\vdots \\ \omega_m^{\text{Server}} &\leftarrow \widehat{\omega_m^r} = \omega_m^r + \text{Laplace}\left(\frac{\Delta f_m}{\varepsilon_m}\right) \\ &\vdots \\ \omega_{M-1}^{\text{Server}} &\leftarrow \widehat{\omega_{M-1}^r} = \omega_{M-1}^r + \text{Laplace}\left(\frac{\Delta f_{M-1}}{\varepsilon_{M-1}}\right) \\ \omega_0^{\text{Server}} &\leftarrow \widehat{\omega_M^r} = \omega_M^r + \text{Laplace}\left(\frac{\Delta f_M}{\varepsilon_M}\right)\end{aligned}$$

where ω_m^{Server} is the parameters of the mth model in the server $(m \in [0, M-1])$. Each data collecting site sets the norm bound α of gradients independently so that the gradients in the local model are limited to different intervals in practical applications. When data collecting site P_x sets $\alpha_x = 1$, the sensitivity is $\Delta f_x = 2 \times \eta_x$.

Hitaj *et al.* [32] suggested that the gradients could also lead to the leakage of sensitive data via the generative adversarial networks (GANs). Whereas, on account of the asynchronous property of the PADL, it is difficult to obtain the prior model parameters for the adversary. Within the capability range, they only can obtain the perturbed model parameters. When the privacy budget is set to 1.0, the prediction accuracy of the model reaches about 11.87% according to Section V-B. Intuitively, the gradients keep "confidential" for any data collecting sites and the cloud. Besides, the differential privacy mechanism also ensures that the adversary cannot obtain additional information.

In general, when multiple malicious parties collude to eavesdrop the private information of particular data collecting sites, this article is still robust.

4) Against the Extreme Collusion Between Malicious Participants and the Cloud Server: In the worst case, only one data collecting site is reliable, while all the other $n-1$ sites collude with the cloud server.

In this setting, the behavior of the legitimate data collecting site can be detected or inferred. Assume that the reliable site P_1 obtains the "initial" model parameters $\omega_{\text{initial}}^{\text{Server}}$ from the server. After multiple interactions, the cloud finally obtains perturbed model parameters $\omega_{\text{final}}^{\text{Server}}$. From the perspective of malicious parties, the behaviors of P_1 are as follows:

$$\begin{aligned}\omega_1^0 &\leftarrow \omega_{\text{initial}}^{\text{Server}} \\ \omega_1^1 &= \omega_1^0 - \eta_1 \times \nabla g_1^0 \\ &\vdots \\ \omega_1^r &= \omega_1^{r-1} - \eta_1 \times \nabla g_1^{r-1}\end{aligned}$$

TABLE II
DETAILS INFORMATION ABOUT TASKS

Task	Dataset	# of Labels	# of Training Example	# of Test Example	Model Architecture	# of Parameters
Handwritten Digits Recognition	MNIST	10	60,000	10,000	2 Conv + 2 FC	0.43 million
Images Recognition	CIFAR-10	10	50,000	10,000	ResNet-50	23.7 million

$$\widehat{\omega_1^r} = \omega_1^r + \text{Laplace}\left(\frac{\Delta f_1}{\varepsilon_1}\right)$$
$$\omega_{\text{final}}^{\text{Server}} \leftarrow \widehat{\omega_1^r}$$

where the learning rate η_1 will be different for each iteration if the data collecting site P_1 dynamically fine-tunes, where $t \in [0, r]$ denotes the tth iteration. The process in the frame is executed locally and is not visible to the malicious party. What the latter can easily grasp is the initial training model $\omega_{\text{initial}}^{\text{Server}}$ and the trained model $\omega_{\text{final}}^{\text{Server}}$, as follows:

$$\omega_{\text{initial}}^{\text{Server}} - \omega_{\text{final}}^{\text{Server}} = \sum_{t\in[0,r]}\left(\eta_1 \times \nabla g_1^t\right) + \text{Laplace}\left(\frac{\Delta f_1}{\varepsilon_1}\right). \quad (9)$$

The right-hand side of (9) is the sum of the product of the gradient and the learning rate for all rounds, also including the Laplace noise in this article. Accordingly, the malicious adversaries decompose perturbed weights parameters, which is equivalent to the subset sum problem.

In conclusion, our solution is still robustness even with only one reliable data collecting site, unless the subset sum problem in dynamic programming is solved.

C. Cumulative Privacy Loss

Iterated learning with differential privacy always causes cumulative noise, which increases the risk of privacy leakage. We adopt Rényi differential privacy (RDP) [43], a relaxation of differential privacy, to analyze cumulative privacy loss [44]. Specifically, let Γ be an adaptive composition of k mechanisms all satisfying ε-differential privacy, given two neighboring data sets: 1) $\mathbb{D}_1$ and 2) $\mathbb{D}_2$, for $\forall Y \subseteq \text{Range}(\Gamma)$, we have

$$\Pr[\Gamma(\mathbb{D}_1) \in Y] \leq e^{\left(2\varepsilon\sqrt{k\log 1/\Pr[\Gamma(\mathbb{D}_2)\in Y]}\right)} \times \Pr[\Gamma(\mathbb{D}_2) \in Y]$$

where adaptive composition means Γ_κ taking as input the same data set and the output of $\Gamma_{\kappa-1}$.

V. PERFORMANCE EVALUATION

In this section, we conduct our experiments to evaluate the performance of the PADL, in the context of multiple classification applications.

A. Experimental Setup

We evaluate the performance of the PADL by two tasks: 1) handwritten digits recognition (MNIST) [45] and 2) images recognition (CIFAR-10) [46]. The detailed information about data sets and model architecture of each task is described as follows. More details about the tasks are also listed in Table II.

1) *Handwritten Digits Recognition (MNIST):* This task is to recognize the handwritten digits from 0 to 9 on the MNIST data set, which consists of 60 000 training examples and 10 000 test examples. The input size of each example is $28 \times 28 \times 1$. The model architecture in this task is guided by the TensorFlow CNN tutorial [47], which consists of two convolutional layers with 5×5 convolutions, and two fully connected layers with 512 neurons and 10 neurons, respectively.
2) *Images Recognition (CIFAR-10):* The task is performed on the CIFAR-10 data set, which includes 50 000 training examples and 10 000 test examples. The goal of this task is to recognize ten different real objects in the real world, such as airplane, bird, truck, and so on, which are not only noisy but also have different proportions and features. The model architecture in this task is the ResNet-50 model.

All data collection sites set the batch size to 128, using tf.train.AdamOptimizer to control the learning rate. Besides, we run our experiments on a workstation running Lenovo server, which equipped with Intel Xeon E5-2620 2.10-GHz CPU, 256 SSD, 16-GB RAM, 1-TB mechanical hard disk.

B. Accuracy of the Local Model

In this section, we discuss the accuracy of the PADL. One of the drawbacks of differential privacy is that it always needs accuracy–privacy tradeoffs [48]. However, we creatively propose the LIP algorithm, which effectively mitigates the pitfall of differential privacy and improves the accuracy of the system given a privacy budget. In this experiment, we compare the effect of the LIP algorithm on model accuracy. When the LIP algorithm is not used, each data collection site perturbs local model parameters through the traditional differential privacy mechanism.

In the task of handwritten digits recognition, we randomly select a trained model with an accuracy of 0.9813 to test the effect of the LIP algorithm on the model accuracy. As shown in the left of Table III, the results of experiments turn out that the LIP algorithm can greatly improve the accuracy of the model. More specifically, when $\varepsilon = 1.0$, the accuracy of the model is about 0.1641 with the LIP algorithm, while it is about 0.1187 without the LIP algorithm. The proportion of promotion is more than 21.83%. Besides, the prediction accuracy of the model significantly improves with the increase of privacy budget ε, i.e., $\varepsilon = 0.5/1.0/5.0/10.0$.

In the task of images recognition, we choose the local model with an accuracy of 0.8732 to evaluate the performance of the LIP algorithm. As shown in the right of Table III, even in the deeper model architecture of the second task, the LIP algorithm we presented still shows excellent performance.

After receiving a "random" model from the server, the data collecting site continuously optimizes with the new model parameters over the local database. We can observe from Fig. 4 that the recovery period of the new model with different

TABLE III
PERFORMANCE OF THE LIP ALGORITHM

Task	Privacy Budget	with LIP algorithm?		Increasing Rate
		YES	NO	
Handwritten Digits Recognition (MNIST)	0.5	0.1058	0.0827	21.83%
	1.0	0.1641	0.1187	27.67%
	5.0	0.8065	0.2854	64.61%
	10.0	0.9577	0.6632	30.75%

Task	Privacy Budget	with LIP algorithm?		Increasing Rate
		YES	NO	
Images Recognition (CIFAR-10)	0.5	0.0932	0.0659	29.33%
	1.0	0.1789	0.1206	32.56%
	5.0	0.6693	0.3501	47.69%
	10.0	0.8511	0.5239	38.44%

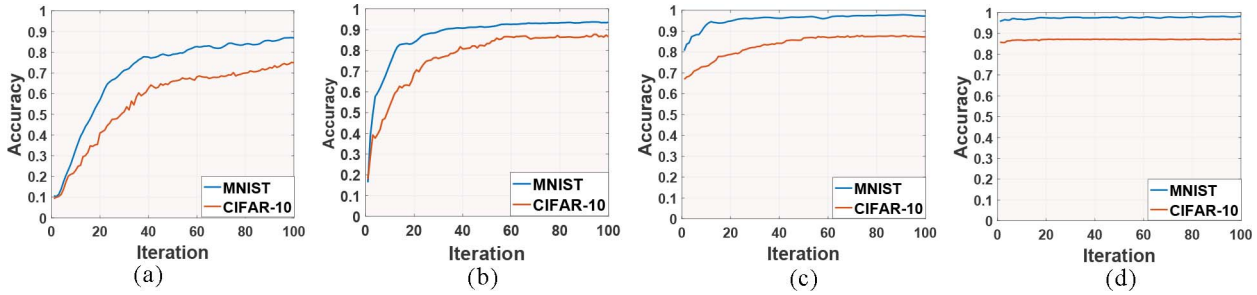


Fig. 4. Accuracy of the PADL. We compare the recovery periods required for the retraining of data collecting sites. Given four different privacy budgets $\varepsilon = 0.5/1.0/5.0/10.0$, we observe the model accuracy of the tasks of handwritten digits recognition (MNIST) and images recognition (CIFAR-10) within 100 iterations.

privacy budgets, i.e., $\varepsilon = 0.5/1.0/5.0/10.0$. The recovery period is defined as the time required to achieve the retrained model accuracy of more than 0.90 or 0.80 for the tasks of handwritten digits recognition (MNIST) and images recognition (CIFAR-10), respectively. Fig. 4(a) shows that the model satisfying ($\varepsilon = 0.5$)-differential privacy has an initial accuracy of almost 0.10 for the task of handwritten digits recognition (MNIST) [0.09 for the task of images recognition (CIFAR-10)], whose recovery period is more than 100 iterations. The recovery period of the model with privacy budget $\varepsilon = 1.0$ is about 30 iterations for the task of handwritten digits recognition (MNIST) [40 for the task of images recognition (CIFAR-10)], as shown in Fig. 4(b). What is more, when the privacy budget of the initial model is $\varepsilon = 10.0$, no recovery period is required as shown in Fig. 4(d). It is owing to that the Laplace noise with $\varepsilon = 10.0$ is enough small and has a little negative effect on the prediction accuracy. It is worth mentioning that the lower the accuracy of the initial model, the longer the recovery period is.

Overall, with the help of the LIP algorithm, we significantly improve the prediction accuracy and effectively shorten the recovery period of the model. The purpose of the latter is to put more fresh data into the training as soon as possible, as well as to improve the information update speed for the task.

C. Accuracy of Prediction

In our PADL, the cloud server deploys multiple model storage spaces, described as M models. As described before, for the prediction task, our solution is to average the weights of the M models to form a temporary prediction model. While the other common prediction schemes are averaging prediction results, randomly selecting a prediction model, and collecting gradients as traditionally distributed deep learning done. We assess the performance of four prediction strategies in six different numbers of models deployed in the server, i.e., $M = 1/10/20/30/50/100$. All schemes are given four different privacy budgets, i.e., $\varepsilon = 0.5/1.0/5.0/10.0$. It is worth mentioning that the perturbed gradients collected by the server also satisfy differential privacy given the same privacy budget as follows:

$$\widehat{\nabla g_{[\omega^r]}} = \nabla g_{[\omega^r]} + \text{Laplace}\left(\frac{\Delta f_g}{\varepsilon}\right)$$

where the sensitivity of the gradient is $\Delta f_g = 2 \times \alpha$.

Comparing with the above three schemes, our approach exhibits extremely high prediction accuracy as shown in Figs. 5 and 6. When the server has only one model ($M = 1$), there is no doubt that averaging weights, averaging prediction results, and randomly selecting a prediction model have the same result. Nevertheless, the accuracy of the solution of collecting gradients is not satisfactory as shown in Figs. 5 and 6. When the number of models deployed in the server exceeds 1, i.e., $M > 1$, our strategy of averaging weights has a more obvious advantage. In particular, when the privacy budget ε is taken at 1.0, the accuracy of our solution is over 0.9186 in the task of handwritten digits recognition (MNIST) as described in Fig. 5, and about 0.8616 in the task of images recognition (CIFAR-10) as shown in Fig. 6.

In general, the solution of averaging weights not only contributes to stable estimation but also helps to neutralize the negative effects generated by noises, which results in higher prediction accuracy.

D. Asynchronous Optimization

1) Performance of Advanced Asynchronous Optimization Protocol: In traditionally distributed deep learning, the server collects the gradients of participants, and calculates the weighted average of the gradients for global update. Finally, the aggregated global gradient update value is broadcast to all participants. In this section, we only take the example of handwritten digits recognition (MNIST). We consider the following three special examples to explain the advantage of asynchronous optimization.

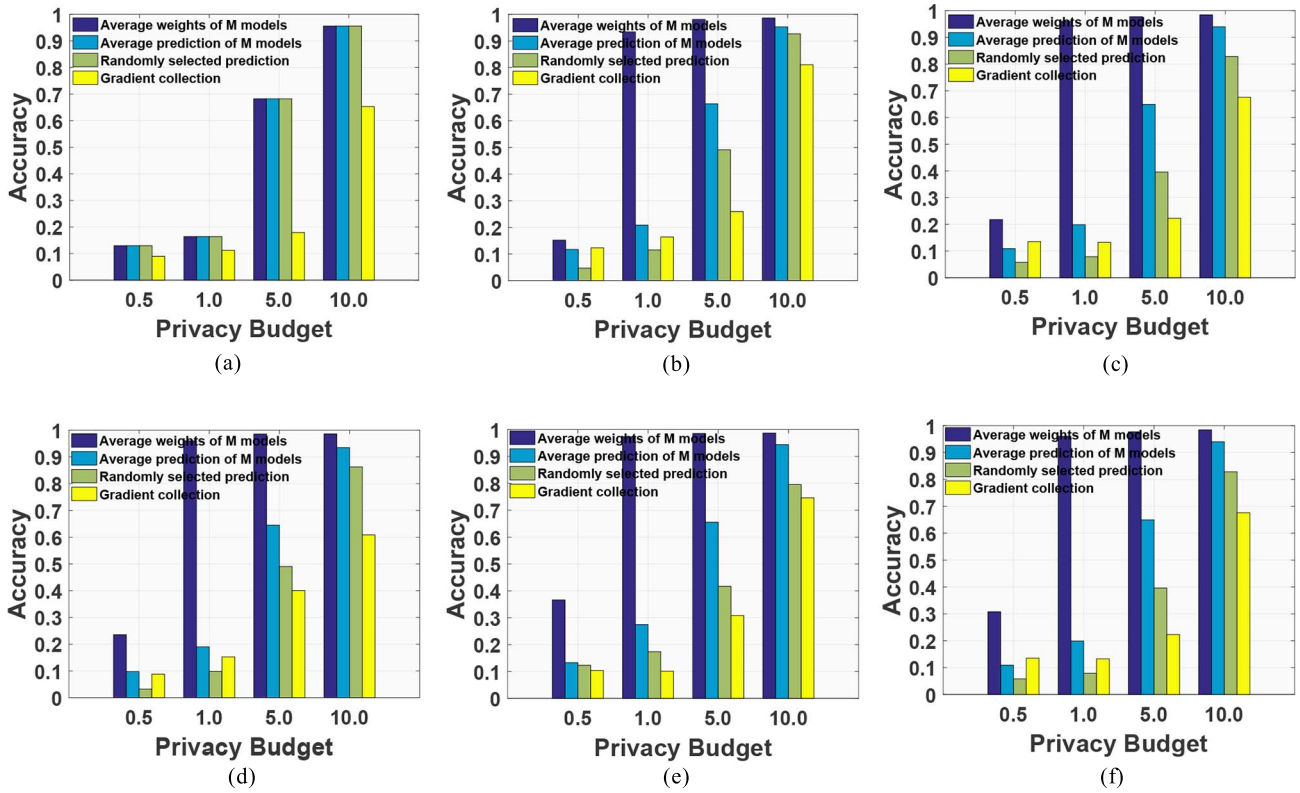


Fig. 5. Prediction accuracy of handwritten digits recognition (MNIST). (a) $M = 1$. (b) $M = 10$. (c) $M = 20$. (d) $M = 30$. (e) $M = 50$. (f) $M = 100$.

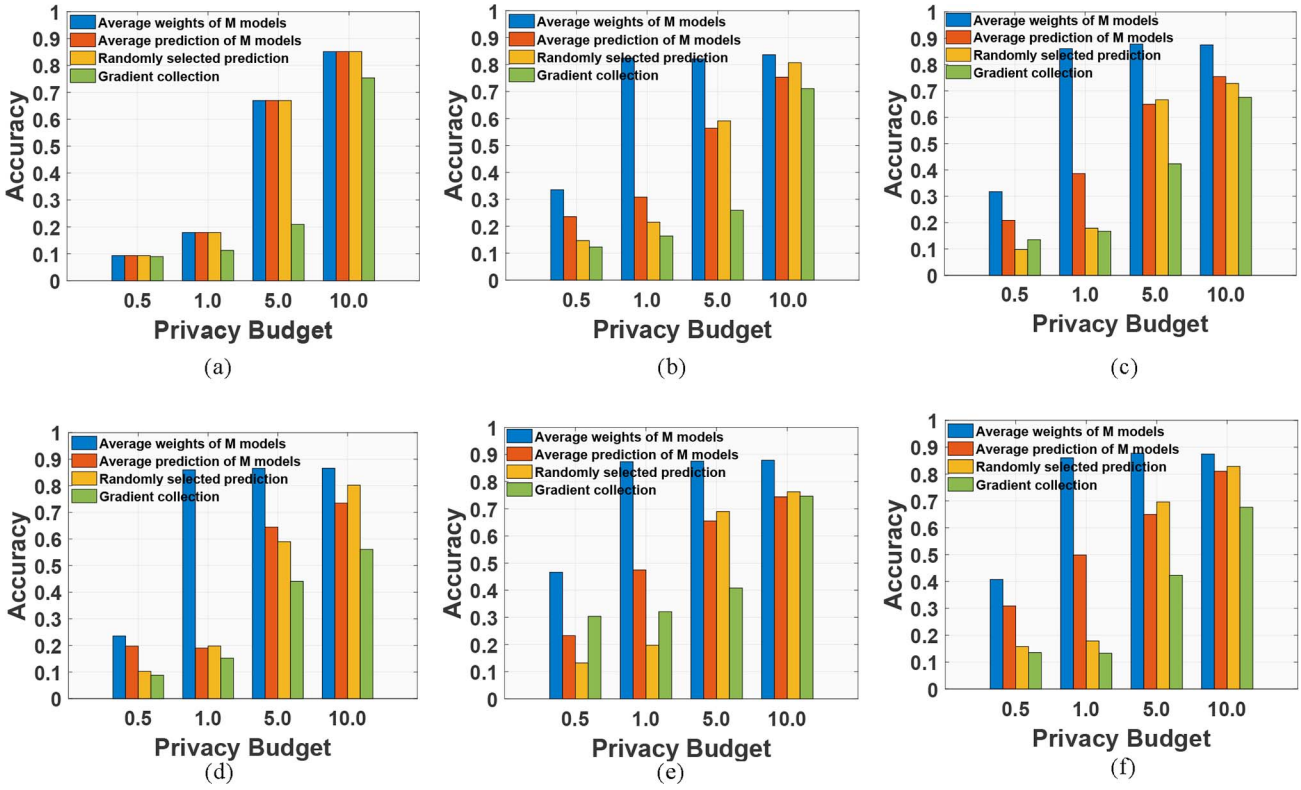


Fig. 6. Prediction accuracy of images recognition (CIFAR-10). (a) $M = 1$. (b) $M = 10$. (c) $M = 20$. (d) $M = 30$. (e) $M = 50$. (f) $M = 100$.

Special Case:

Case 1: Usually, there are many possible causes for different computing power between participants. It is assumed that among the ten participants, the first participant has extremely high computing power, while the tenth participant is the last one

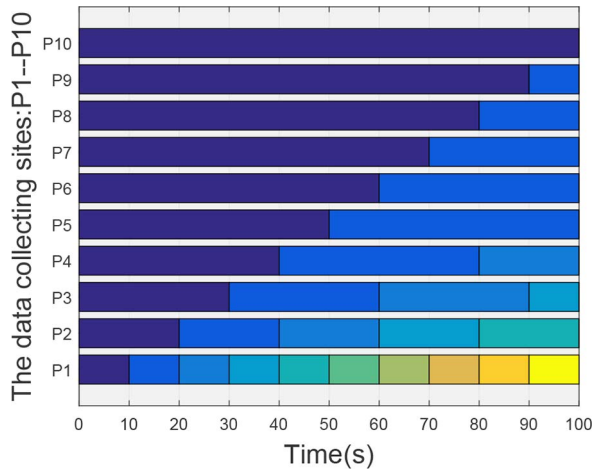


Fig. 7. AAO protocol for the data collecting sites with different computing power: the computing power of P_1 is ten times than that of P_{10}, where each interval represents a complete training session.

in the rank of computing power. After submitting the gradients, the first participant will wait until the other nine participants finish training. That results in wasting the computing power of the first participant and delaying the training. Being aware that in practice, data are always continuously generated.

Case 2: Assume that the server needs to collect the gradients from ten participants, and it has collected the gradients from nine participants. However, all the remaining participants are out of training. Under such circumstances, the server will be waiting for the tenth participant, and the whole training process will be stagnant.

Case 3: The server takes measures on the participant's drop out or the difference in computing power: if the node is over a certain time, the server updates with the collected gradients. It is assumed that over the time limit, only one participant's gradients are collected. Under such circumstances, the server will broadcast the gradients to all participants, which will result in the leakage of personal privacy [32].

Case 2 leads to training stagnation, case 3 brings about privacy threat. Here, we focus on the negative impact brought by case 1. As described in Fig. 7, we make a comparison with the different computing power among ten sites. The data collecting site P_1 has the strongest computing power, and the running time includes training one epoch and interactions before the next training: $T_{P_1} = T_{\text{One,Training}} + T_{\text{Upload}} + T_{\text{Download}} = 10$ s, ..., $T_{P_{10}} = T_{\text{One,Training}} + T_{\text{Upload}} + T_{\text{Download}} = 100$ s.

Assume that a basic data volume is 60 000, training one epoch adds 10% fresh data. Comparing with traditional deep learning, as shown in Table IV, P_1 needs to wait until the last second after 10 s, with 0 fresh data. While in the AAO protocol, P_1 immediately conducts the next training after the first submitting, and continues to train nine epochs in 100 s. After ten epochs, a single data collecting site brings 81 475 fresh data into models. Considering ten sites in 100 s, the total amount of fresh data brought by $P_1 - P_{10}$ is $81\,475 + 27\,846 + 12\,600 + 6000 + 6000 = 133\,921$. Referring to Fig. 8, given the privacy budget $\varepsilon = 1.0$, the accuracy is increasing in 100 s due to the ongoing joining fresh data. At the 70 and 90 s, the prediction accuracy drops a little. We consider this phenomenon when the number of models is singular, the cancellation ability of noises is lower than even numbers.

In summary, the AAO protocol is more practical and advanced in practical IoT applications.

TABLE IV
PERFORMANCE OF ASYNCHRONOUS OPTIMIZATION

	Traditional Optimization		Asynchronous Optimization	
	Frequency(100s)	Fresh Data	Frequency(100s)	Fresh Data
P_1	1	0	10	81,475
P_2	1	0	5	27,846
P_3	1	0	3	12,600
P_4	1	0	2	6,000
P_5	1	0	2	6,000
P_6	1	0	1	0
P_7	1	0	1	0
P_8	1	0	1	0
P_9	1	0	1	0
P_{10}	1	0	1	0

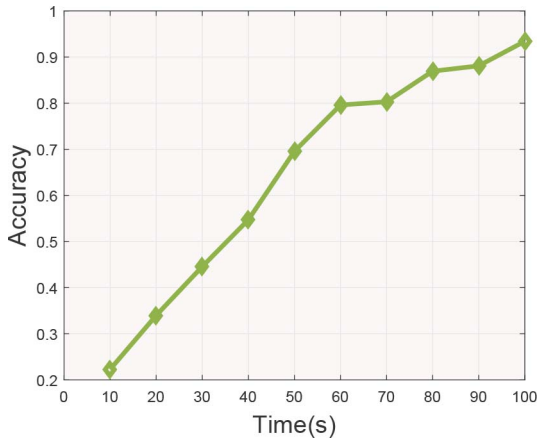


Fig. 8. Accuracy of asynchronous optimization (privacy budget $\varepsilon = 1.0$).

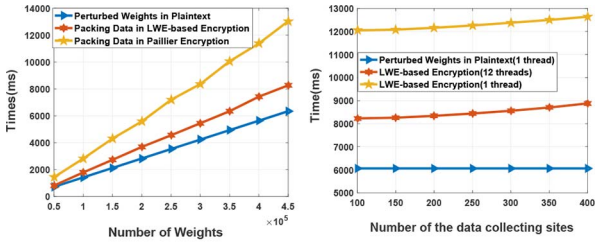
E. Efficiency

In this section, we discuss the computation overhead and communication overhead of the PADL.

As described before, in addition to the normal deep learning training in the PADL, we propose the LIP algorithm, which combines with differential privacy to efficiently provide privacy protection. We compare with the privacy-preserving deep learning system in combination with additively homomorphic encryption [49], which encrypts the gradients of the local model via Paillier encryption or LWE-based encryption after local calculation. Then, they adopt the form of packing ciphertext for reducing communication overhead, where real numbers in ciphertext are represented by 32 bits, while 16 bits in plaintext. For fairness, the experimental configuration and model architectures are the same as ours. Besides, the number of data collection sites $N = 50$. Note that since the number of training examples in the task of handwritten digits recognition is 60 000, each data collection site randomly holds $60\,000/50 = 1200$ examples. Similarly, each data collection

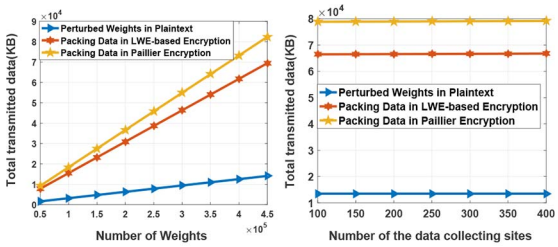
TABLE V
PROPORTION OF ADDITIONAL COMPUTATION OVERHEAD

Task	PADL	Paillier-based	LWE-based
Handwritten Digits Recognition (MNIST)	0.15%	49.21%	22.04%
Images Recognition (CIFAR-10)	0.04%	3.38%	0.98%



(a) (b)

Fig. 9. Computation cost for each data collecting site. (a) Computation cost with different number of weights. (b) Computation cost with different number of the data collecting sites.



(a) (b)

Fig. 10. Communication cost for each data collecting site. (a) Communication cost with different number of weights. (b) Communication cost with different number of data collecting sits.

TABLE VI
COMMUNICATION OVERHEAD AT EACH UPLOAD/DOWNLOAD

Task	PADL	Paillier-based	LWE-based
Handwritten Digits Recognition (MNIST)	0.82 MB	5.04 MB	4.25 MB
Images Recognition (CIFAR-10)	47.40 MB	310.94 MB	237.95 MB
Increased Factors	/	6.36×	5.07×

site randomly holds almost 1 000 examples in the task of images recognition (all used in [49]).

1) Computation Overhead: In the PADL, each data collection site is only required to perform additional customized perturbations. In our experimental environment, for the task of handwritten digits recognition, the perturbed model parameters in plaintext are sent in around 8 ms via 1 Gb/s communication channel, i.e., $T_{\text{Upload}} = 8$ ms. The data collecting site downloads a new model from the server, which also takes 8 ms, i.e., $T_{\text{Download}} = 8$ ms, and trains 200 iterations taking $T_{200,\text{Training}} = 30.30$ ms/iteration $\times$ 200 iterations $= 6060.00$ ms. Besides, the average time of performing perturbation is 0.091 ms. Hence, additional privacy protection mechanism accounts for 0.1498% of each round of training.

Table V details the proportion of the additional computation overhead of the different methods in the total training process. For deeper model architecture, training takes longer, which is related to many factors such as the complexity of the network structure. While the computation overhead of encryption and decryption only increases linearly with the number of model parameters, so the proportion of privacy protection mechanisms in images recognition tasks is reduced. Besides, with the deepening of the network architecture, the ratio of the overhead of the PADL is also decreasing.

Fig. 9 depicted the computation cost of each data collecting site with a different number of weights/data collection sites for the task of handwritten digits recognition. For the images recognition task, the result is the same, so let us just take MNIST as an example. We can observe that the computation overhead in the PADL is much lower than that of homomorphic encryption. Note that the computation cost of the method of LWE-based encryption with 12 threads of computation is higher than our solution with 1 thread as Fig. 9(b) shown.

Overall, our scheme has negligible additional computation overhead, which is very friendly to data collection points with limited computing power.

2) Communication Overhead: As listed in Table II, the model architecture of the handwritten digits recognition task is commonly used four-layer DNN, which has a total of 0.43 million parameters, while 23.7 million parameters in the ResNet-50. In the PADL, the data collection sites send the perturbed model parameters in plaintext to the cloud server, while in [49], the parameters are sent by the form of ciphertext. For the task of handwritten digits recognition, the communication overhead of the PADL at each upload or download is

$$\frac{0.43 \times 10^6 \times 16}{8 \times 10^6} \approx 0.82 \text{ MB}.$$

While the communication overhead of the Paillier encryption with the packing technology at each upload or download is 5.04 MB, and 4.25 MB for LWE-based encryption with the packing technology. The communication cost of the two encryption methods is 6.36× and 5.07× higher than ours. Table VI shows more information about the communication overhead at each upload/download.

Fig. 10 shows the comparison of the communication cost of the PADL and the solution of homomorphic encryption. With the increase in the number of weights/data collection sites, the communication overhead of our solution is significantly low. For the images recognition task, the result is the same, so let us just take MNIST as an example.

Therefore, our scheme is also excellent in terms of communication cost.

VI. RELATED WORK

In this section, we introduce the existing works for the data privacy of IoT applications. Besides, since we focus on deep-learning-based IoT applications in this article, we also review the related works from the perspective of deep learning.

A. Data Privacy in IoT Applications

The most popular IoT architecture consists of three layers: 1) sensor/perception; 2) middleware; and 3) application layer [50]. The security and privacy issues in each layer should be taken into account and addressed [51]. For instance,

Gope *et al.* [6] designed a lightweight and privacy-preserving RFID-based authentication scheme for the perception layer of IoT-based infrastructure. Elmisery *et al.* [52] built a cognitive-based middleware for private IoT-enabled data mashup to serve a centralized environmental monitoring service. As for the application layer, the solution to address security and privacy issues should be application specific. For example, in the personalized healthcare systems (PHS), enormous amounts of vital signs and wearable sensor data, such as BP, body temperature, electrocardiograms (ECG), and oxygen saturation are collected. They are transmitted into static or mobile electronic devices with heterogeneous computing and storage capability, such as laptops, smartphones, and medical terminals [8], [53]. Son *et al.* [54] proposed a dynamic access control model for preserving the privacy of personal healthcare data in a cloud environment. Another example is AVs, which typically equipped with varieties of rich sensor-based devices, such as a global position system (GPS), LiDAR, and cameras, to perceive their surrounding environment. The data are fed into the computing system in the AV for controlling AV [55]. Yu *et al.* [56] utilized encryption and obfuscation techniques to prevent the information of automobiles from sniffing.

B. Privacy-Preserving Deep-Learning-Based IoT Applications

The vast majority of deep-learning-based IoT applications focus on the actual performance of the application. For instance, Garg *et al.* [57]–[59] proposed a hybrid deep-learning-based model for anomaly detection. Jiang *et al.* [60] put forward deep-learning-based multichannel intelligent attack detection. However, on the one hand, data privacy is critical for both IoT applications and deep learning. On the other hand, deep-learning-based IoT applications are popping up. Therefore, it is necessary to pay close attention to privacy-preserving IoT applications from the perspective of deep learning.

There are a few existing privacy-preserving IoT applications from the perspective of deep learning, which mainly involves three underlying technologies: 1) federated learning [23], [24]; 2) encryption-based technologies [26]–[28]; and 3) differential privacy [29], [30]. For instance, Jiang *et al.* [24] developed a federated learning framework for the IoT Edge devices to protect data privacy. However, federated learning is achieved at the expense of high communication overhead. Moreover, Melis *et al.* [61] have proved that a malicious participant can infer sensitive information from the "shared" gradients in federated learning. By exploiting the encryption-based technology, Li *et al.* [28] proposed a privacy-preserving data aggregation scheme for mobile-edge computing-assisted IoT applications. Unfortunately, most existing encryption-based technologies always involve expensive overhead in computation, which leads to system performance degradation greatly. Xu *et al.* [30] designed a deep inference framework-based edge computing with local differential privacy for mobile data analytics, which has low communication and computation costs. However, their method suffers from a loss of accuracy.

In this article, we propose the PADL, which is a privacy-preserving deep learning framework for IoT applications. Moreover, our solution is the first to implement deep learning with asynchronous optimization. Theoretical analysis shows that it also is robustness to extreme collusion even with only one reliable site. Compared with previous works, the PADL can perform deep learning training tasks efficiently, while ensuring data privacy in IoT applications.

VII. Conclusion

The traditional deep learning methods assisted IoT applications are vulnerable to privacy risks and data breaches. In this article, we have presented a feasible solution (PADL) to protect the privacy of training data and handle large data streams in IoT applications. Properties analysis demonstrates that the PADL has the characteristic of asynchronous optimization, and the privacy property against the extreme collusion attack. Experiments performed on the real-world data sets describe the practical performance of our proposed strategy. In future work, we will take into consideration the data integrity and availability of the scheme at security attacks.

References

[1] J. A. Stanković, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Mar. 2014.

[2] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in Internet of Things: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 1–27, Nov. 2017.

[3] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3854–3865, Apr. 2019.

[4] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[6] P. Gope, R. Amin, S. H. Islam, N. Kumar, and V. K. Bhalla, "Lightweight and privacy-preserving RFID authentication scheme for distributed IoT infrastructure with secure localization services for smart city environment," *Future Gener. Comput. Syst.*, vol. 83, pp. 629–637, Jun. 2018.

[7] E. Park, Y. Cho, J. Han, and S. J. Kwon, "Comprehensive approaches to user acceptance of Internet of Things in a smart home environment," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2342–2350, Dec. 2017.

[8] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *J. Biomed. Informat.*, vol. 69, pp. 218–229, May 2017.

[9] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. ACM Int. Conf. Softw. Eng. (ICSE)*, Jun. 2018, pp. 303–314.

[10] R. Poplin *et al.*, "Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning," *Nat. Biomed. Eng.*, vol. 2, no. 3, p. 158, Feb. 2018.

[11] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 3, pp. 312–325, Feb. 2016.

[12] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, Jan. 2018.

[13] A. Rachedi and A. Benslimane, "Multi-objective optimization for security and QoS adaptation in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.

[14] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1606–1616, Jun. 2018.

[15] M. Amoozadeh *et al.*, "Security vulnerabilities of connected vehicle streams and their impact on cooperative driving," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 126–132, Jun. 2015.

[16] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Security Privacy*, May 2019, pp. 739–753.

[17] Y. Ji, X. Zhang, S. Ji, X. Luo, and T. Wang, "Model-reuse attacks on deep learning systems," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Oct. 2018, pp. 349–363.

[18] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Security Privacy*, Jul. 2018, pp. 19–35.

[19] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *Proc. IEEE Security Privacy*, Jul. 2018, pp. 36–52.

[20] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 1, pp. 127–138, Dec. 2015.

[21] G. Xu, H. Li, H. Ren, K. Yang, and R. H. Deng, "Data security issues in deep learning: Attacks, countermeasures, and opportunities," *IEEE Commun. Mag.*, vol. 57, no. 11, pp. 116–122, Nov. 2019.

[22] A. Rachedi and A. Benslimane, "Security and pseudo-anonymity with a cluster-based approach for MANET," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Nov. 2008, pp. 1–6.

[23] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, Jul. 2019.

[24] L. Jiang, X. Lou, R. Tan, and J. Zhao, "Differentially private collaborative learning for the IoT edge," in *Proc. Eur. Conf. Workshop Wireless Sensor Netw. (EWSN)*, Feb. 2019, pp. 341–346.

[25] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Trans. Ind. Informat.*, early access, doi: 10.1109/TII.2019.2945367.

[26] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 870–885, Aug. 2019.

[27] H. Li, D. Liu, Y. Dai, T. H. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 97–109, Dec. 2018.

[28] X. Li, S. Liu, F. Wu, S. Kumari, and J. J. P. C. Rodrigues, "Privacy preserving data aggregation scheme for mobile edge computing assisted IoT applications," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4755–4763, Jul. 2019.

[29] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," Feb. 2018. [Online]. Available: arXiv:1802.08908.

[30] C. Xu, J. Ren, L. She, Y. Zhang, Z. Qin, and K. Ren, "EdgeSanitizer: Locally differentially private deep inference at the edge for mobile data analytics," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5140–5151, Feb. 2019.

[31] K. Bonawitz *et al.*, "Towards federated learning at scale: System design," 2019. [Online]. Available: arXiv:1902.01046.

[32] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, Nov. 2017, pp. 603–618.

[33] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. ACM Symp. Theory Comput.*, May 2009, pp. 169–178.

[34] C. Dwork, "Differential privacy: A survey of results," in *Proc. Theory Appl. Models Comput. (TAMC)*, Apr. 2008, pp. 1–19.

[35] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[36] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 4037–4049, Mar. 2017.

[37] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, Sep. 2010, pp. 177–186.

[38] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM Comput. Commun. Security (CCS)*, Oct. 2017, pp. 1175–1191.

[39] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *Proc. IEEE Security Privacy*, Nov. 2017, pp. 19–38.

[40] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM Comput. Commun. Security (CCS)*, Oct. 2016, pp. 308–318.

[41] V. Pihur *et al.*, "'Differentially-private draw and discard' machine learning," 2018. [Online]. Available: arXiv:1807.04369.

[42] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Security Privacy*, Jun. 2017, pp. 3–18.

[43] I. Mironov, "Rényi differential privacy," in *Proc. IEEE Comput. Security Found. Symp. (CSF)*, Aug. 2017, pp. 263–275.

[44] Y.-X. Wang, B. Balle, and S. Kasiviswanathan, "Subsampled Rényi differential privacy and analytical moments accountant," Dec. 2018. [Online]. Available: arXiv:1808.00087.

[45] Y. LeCun, C. Cortes, and C. Burges. (2010). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist

[46] A. Krizhevsky, V. Nair, and G. Hinton. (2014). *The CIFAR-10 Dataset*. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html

[47] TensorFlow. (2018). *TensorFlow Tutorials*. [Online]. Available: https://www.tensorflow.org/tutorials

[48] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. USENIX Security*, Aug. 2019, pp. 1895–1912.

[49] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, May 2018.

[50] A. Tewari and B. Gupta, "Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework," *Future Gener. Comput. Syst.*, to be published.

[51] V. Adat and B. Gupta, "Security in Internet of Things: Issues, challenges, taxonomy, and architecture," *Telecommun. Syst.*, vol. 67, no. 3, pp. 423–441, Jun. 2018.

[52] A. M. Elmisery, M. Sertovic, and B. B. Gupta, "Cognitive privacy middleware for deep learning mashup in environmental IoT," *IEEE Access*, vol. 6, pp. 8029–8041, 2017.

[53] J. Qi, P. Yang, G. Min, O. Amft, F. Dong, and L. Xu, "Advanced Internet of Things for personalised healthcare systems: A survey," *Pervasive Mobile Comput.*, vol. 41, pp. 132–149, Oct. 2017.

[54] J. Son, J.-D. Kim, H.-S. Na, and D.-K. Baik, "Dynamic access control model for privacy preserving personalized healthcare in cloud environment," *Technol. Health Care*, vol. 24, no. s1, pp. 123–129, 2016.

[55] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proc. IEEE*, vol. 108, no. 2, pp. 357–372, Nov. 2019.

[56] L. Yu, J. Deng, R. R. Brooks, and S. B. Yun, "Automobile ECU design to avoid data tampering," in *Proc. ACM Cyber Inf. Security Res. Conf. (CISRC)*, Apr. 2015, pp. 1–4.

[57] S. Garg, K. Kaur, N. Kumar, and J. J. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 566–578, Jan. 2019.

[58] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, and R. Ranjan, "A hybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 3, pp. 924–935, Jul. 2019.

[59] S. Garg, K. Kaur, G. Kaddoum, S. H. Ahmed, and D. N. K. Jayakody, "SDN-based secure and privacy-preserving scheme for vehicular networks: A 5G perspective," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8421–8434, May 2019.

[60] F. Jiang *et al.*, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE Trans. Sustain. Comput.*, early access, doi: 10.1109/TSUSC.2018.2793284.

[61] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Security Privacy*, Sep. 2019, pp. 691–706.

Xiaoyuan Liu (Student Member, IEEE) received the B.S. degree from the School of Control and Computer Engineering, North China Electric Power University, Beijing, China, in 2016. She is currently pursuing the master's degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

Her research interests include cryptography, and the privacy issues in deep learning.

Hongwei Li (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in June 2008.

He is currently the Head and a Professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China. He worked as a Postdoctoral Fellow with the University of Waterloo, Waterloo, ON, Canada, from October 2011 to October 2012. His research interests include network security and applied cryptography.

Prof. Li is the Distinguished Lecturer of the IEEE Vehicular Technology Society.

Guowen Xu (Student Member, IEEE) received the B.S. degree in information and computing science from Anhui University of Architecture, Hefei, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include cryptography, searchable encryption, and the privacy issues in deep learning.

Sen Liu (Student Member, IEEE) received the B.S. degree in information security from Guizhou University, Guiyang, China, in 2017. He is currently pursuing the master's degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include cryptography and searchable encryption.

Zhe Liu (Senior Member, IEEE) received the B.S. and M.S. degrees from Shandong University, Jinan, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Laboratory of Algorithmics, Cryptology and Security, University of Luxembourg, Luxembourg City, Luxembourg, in 2015.

He was a Researcher with SnT, University of Luxembourg. He is a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He has coauthored more than 70 research peer-reviewed journal and conference papers. His research interests include computer arithmetic and information security.

Prof. Liu received the prestigious FNR Awards 2016—Outstanding Ph.D. Thesis Award for his contributions in cryptographic engineering on IoT devices.

Rongxing Lu (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2012.

He worked as an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from April 2013 to August 2016. He is currently an Associate Professor with the Faculty of Computer Science (FCS), University of New Brunswick (UNB), Fredericton, NB, Canada. He worked as a Postdoctoral Fellow with the University of Waterloo from May 2012 to April 2013.

Dr. Lu was awarded the most prestigious Governor General's Gold Medal; and won the 8th IEEE Communications Society (ComSoc) Asia–Pacific Outstanding Young Researcher Award in 2013. He is the winner of the 2016–2017 Excellence in Teaching Award, FCS, UNB. He is presently a Senior Member of the IEEE Communications Society. He currently serves as the Vice-Chair (Publication) of IEEE ComSoc CIS-TC.