Accurate Differentially Private Deep Learning on the Edge

Rui Han, Dong Li, Junyan Ouyang, Chi Harold Liu, *Senior Member, IEEE*, Guoren Wang, *Senior Member, IEEE*, Dapeng Wu, *Fellow, IEEE*, and Lydia Y. Chen, *Senior Member, IEEE*

Abstract—Deep learning (DL) models are increasingly built on federated edge participants holding local data. To enable insight extractions without the risk of information leakage, DL training is usually combined with differential privacy (DP). The core theme is to tradeoff learning accuracy by adding statistically calibrated noises, particularly to local gradients of edge learners, during model training. However, this privacy guarantee unfortunately degrades model accuracy due to edge learners' local noises, and the global noise aggregated at the central server. Existing DP frameworks for edge focus on local noise calibration via gradient clipping techniques, overlooking the heterogeneity and dynamic changes of local gradients, and their aggregated impact on accuracy. In this article, we present a systematical analysis that unveils the influential factors capable of mitigating local and aggregated noises, and design PrivateDL to leverage these factors in noise calibration so as to improve model accuracy while fulfilling privacy guarantee. PrivateDL features on: (i) sampling-based sensitivity estimation for local noise calibration and (ii) combining large batch sizes and critical data identification in global training. We implement PrivateDL on the popular Laplace/Gaussian DP mechanisms and demonstrate its effectiveness using Intel BigDL workloads, i.e., considerably improving model accuracy by up to 5X when comparing against existing DP frameworks.

Index Terms—Deep learning, differential privacy, federated learning, model accuracy

1 Introduction

Deep learning (DL) has gained unquestionable success in many domains (e.g., image classification, video object recognition) and continues to flourish. Today, the emergence of edge computing [1] presents a new paradigm that trains DL models on multiple edge nodes/participants holding local private data. Federated learning is a prevalent framework to support such collaborative learning using datasets distributed across multiple participants (e.g., edge nodes) [2], [3], [4], [5], [6]. With federated learning, edge nodes compute the model updates using decentralized data and contribute to the global model updates, e.g., exchanging the gradients under the orchestration of a central server. However, it's shown that there is still a risk of information leakage when only intermediate model updates are exchanged between edge nodes and the server [7], [8], [9]. Differential privacy [10], [11], [12] is one prevalent technique to protect data privacy by adding statistically calibrated noises at different learning stages based on the estimated risk of privacy loss [2], [13], [14], [15]. The privacy guarantee here is unfortunately at the cost of accuracy losses: lower risks of privacy leakage, but also lower DL accuracies.

Example. Fig. 1 shows a typical example of distributed DL training, which performs global aggregation of gradients across multiple edge nodes in an iterative manner. With differential privacy, each node first computes its local gradients and noise-added gradients, instead of the actual values, and then sends them to a central server. One can see that the DL model accuracy can be degraded first by local noise injected from each local node and then through the aggregation of these noises in the central server. Our evaluations of BigDL jobs on the KubeEdge platform (in Sections 3.2 and 3.3) show that either *local* noises or their *aggregation* lead to more than 50 percent accuracy degradation compared to the one without noises.

Differentially Private DL. The core task of differentially private DL is to determine when, which type and how many noises to add given a privacy budget ϵ, which determines how much information is leaked by a differential privacy mechanism. Conceptually, this budget defines *the theoretical upper bound* $(e^{\varepsilon}-1)$ *of privacy leakage* [16]. That is, a smaller budget denotes a lower bound of privacy leakage and thus requires larger noises to perturb the model learning process. In practice, two types of statistical noises are typically considered: Laplace and Gaussian, and the amount of noise is jointly determined by the *privacy budget* and the *data sensitivity*. The later one measures how sensitive of individual data reacting to noises, and it is defined by the pair-wise norm of gradients in DL training. Essentially, data that has a higher sensitivity needs stronger/larger noises to prevent the privacy leakage.

Challenges of High-Privacy and Accurate DL Training. Edge-based DL training usually requires small privacy budgets to

- *Rui Han, Dong Li, Junyan Ouyang, Chi Harold Liu, and Guoren Wang are with the Beijing Institute of Technology, Beijing 100811, China. E-mail: {hanrui, wanggr}@bit.edu.cn, {1103464147, 912970124}@qq.com, liuchi02@gmail.com.*
- *Dapeng Wu is with the University of Florida, Gainesville, FL 32611 USA. E-mail: dpwu@ufl.edu.*
- *Lydia Y. Chen is with TU Delft, 2628 Delft, The Netherlands. E-mail: Y.Chen-10@tudelft.nl.*

Manuscript received 30 Oct. 2020; revised 2 Mar. 2021; accepted 3 Mar. 2021. Date of publication 8 Mar. 2021; date of current version 19 Mar. 2021. (Corresponding author: Chi Harold Liu.) Recommended for acceptance by Y. Yang. Digital Object Identifier no. 10.1109/TPDS.2021.3064345

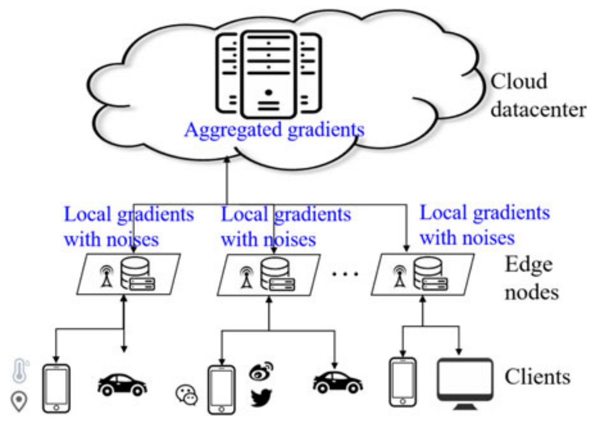
Cloud datacenter
Aggregated gradients
Local gradients with noises
Local gradients with noises
Local gradients with noises
Edge nodes
Clients

Fig. 1. An example of edge-based DL training with differential privacy.

guarantee high privacy levels, at the same time, it requires low noises to mitigate model accuracy degradation. Achieving these objectives gives rise to three major challenges in practice.

First, it is necessary to systematically analyze the influential factors, in addition to privacy budget, of both local and aggregated noises. Specifically, the extent of local noise is mainly determined by the data sensitivity given a privacy budget. Deducing the influential factor of sensitivity, therefore, is the prerequisite to appropriately estimate the sensitivity. Moreover, unveiling the influential factors to minimize the aggregated noise at the central server is another imperative and unsolved challenge in a federated learning setting.

Second, the state-of-the-art of differentially private DL [2], [15], [17], [18] employs gradient clipping techniques that bound the data sensitivity and thus further bound noises added to gradients. However, real-world DL training tasks usually have to tackle with varying models, datasets, and parameters (initial model parameters and hyperparameters in training), which lead to large discrepancies of gradients among different tasks as well as high fluctuations of gradients during a task's iterative training process (for example, the gradients at iteration 10k can be 10000x larger than those at iteration 100 [19]). Hence there is no "one-size-fits-all" best clip bound for different training tasks, and even in a task, the proper bound dynamically changes across training iterations because it is determined by the current range of gradients.

Finally, existing techniques address the question of how to estimate sensitivity for local noises in individual nodes, but do not consider issues relating to the overall model accuracy degraded by aggregated noises in a federated learning setting. Hence how to reduce aggregated noises without increasing the model training overhead is another challenge to be addressed.

Motivated by these challenges, this paper proposes PrivateDL, a novel framework that can reduce local noises by accurately estimating data sensitivity locally without pre-defined clip bounds, while further reducing aggregated noises via virtual batch size amplification with critical set. We specifically consider the federated learning setting, where edge participants train their local models on the local data and exchange gradients via the central server. The design of this framework is based on the systematical analysis that unveils the complex dependency from local noise, to aggregated noise on model accuracy, given a privacy budget. The core feature of PrivateDL thus is to tune factors that can minimize the impact on local and aggregated noises, that is, sensitivity estimation, large batch sizes (to reduce aggregated noises), and training on a subset of critical input data (to improve performance). In detail, we make the following technical contributions:

- We analyze the characteristics of distributed DL model training and deduce the influential factors of local and aggregated noises for both ϵ-differential privacy and (ϵ, δ)-differential privacy noise mechanisms. The experimental evaluations on real DL models and datasets show these factors indeed have significant influences on noises, namely model accuracies (Section 3).
- We design two PrivateDL modules for efficient and accurate model training (Section 4). First, the *sensitivity estimation* module reduces local noises injected in each edge node by dynamically sampling the range of gradients at each iteration, avoiding using the pre-defined gradient clipping bounds. Second, to mitigate model accuracy degradation due to aggregated noises, the *virtual batch size amplification* module increases the batch size to reduce aggregated noises. This module also employs a redundant input data removal technique to decrease the computational costs in gradient calculation.
- We implement PrivateDL on KubeEdge [20], an emerging edge computing platform in the Kubenetes ecosystem [21] and incorporate it with the deep ML algorithms in Intel BigDL [22] and PyTorch [23] (Section 4.4). By applying PrivateDL in both ϵ-differential privacy and (ϵ, δ)-differential privacy noise mechanisms, the comparative experiments against existing clip techniques show: (i) under different DL training settings of *local noises*, PrivateDL increases model accuracy by an average of 411.65 percent, in particular, the accuracy increase is 565.87 percent for the smallest privacy budgets (i.e., the highest privacy level); (ii) under low privacy levels where the accuracy improvement of local noise reduction is small, PrivateDL still improves the model accuracy by an average of 131.88 percent via *aggregated noise* reduction (Section 5).

2 Background

This section first explains the basic concepts of data privacy violation in DL (Section 2.1) and then introduces differential privacy (Section 2.2). We summarize the notations in Table 1.

2.1 Data Privacy and Inference Attacks in DL

DL models are susceptible to various data privacy leakages as they remember information about their training data, both in the model parameters and in the parameter updates during training. Within this context, inference attacks on DL algorithms fall into two major categories [24]: tracing/

TABLE 1
Symbols Description

Symbol	Meaning
K	the number of edge nodes
T	the batch of input data points used at one iteration of training
B	the total number of data points in T, namely the *batch size*
g	a gradient that corresponds to a model parameter
n	the number of model parameters/gradients
Y	a vector $(g_1, g_2, \ldots, g_n)$ of gradients in a DL model of n parameters
$f(.)$	the function applied in a dataset for gradient computation
γ	ratio of input data reduction
ϵ	privacy budget
δ	the parameter in (ϵ, δ)-differential privacy
$Noise(\mu, \sigma)$	a noise distribution with mean value μ and standard deviation σ
$\sigma_{aggregate}$	the standard deviation of aggregated noise
Δ_p	L_p-sensitivity (p = 1 or 2)
C	clip bound

membership attacks that infer if a particular data point was included in the training samples; and reconstruction attacks that infer attributes of data points in the training data.

Membership Attack. This attack can be divided into two types. In the black-box setting, an attacker can only quantify membership information leakage using the prediction outputs of the target model [25]. Hence an attack model is trained to distinguish the target model's behaviour on the entire training data from its behavior on the data without some training samples. In the white-box setting, an attacker can observe the activation functions of the target model, thus either passively observes the model updates or actively influences the training process to exact more information [24].

Reconstruction Attack. This attack learns from the target model about the properties that characterize the entire class [26] or a subset of classes [27], making it possible to construct the representatives of the learned classes. For example, federated learning is designed to support model learning using private training data from different participants. Reconstruction attacks allow an attacker to infer properties of other participants training data [9].

To prevent the above attacks and mask the contribution (privacy information) of any individual participant, differential privacy is one major privacy-preserving technique that introduces uncertainty into the target model. The notion of differential privacy was initially proposed by Dwork *et al.* [28] to bound the probability of data privacy leakage in database by adding Laplacian noises. The differentially private mechanisms were first adopted by the database community, e.g., sublinear query (SuLQ) database models [29] and INQ (Privacy Integrated Queries) systems [14]. The core step of differential private algorithms is to decide the level of injected statistical noises based on the privacy budget and how sensitive the data is to the perturbation. The typical differential private DL algorithms are introduced in the following section.

2.2 Differentially Private DL

Given a DL task, let D and D' denote arbitrary adjacent datasets (that is, they differ in just one record), and let M be a differential private algorithm that adds statistical noises to gradient updates, according to a given *privacy budget* defined by ϵ and *data sensitivity* defined by Δ.

Definition 1: ***(ϵ-Differential Privacy).*** *Let P_M be the domain of all possible outputs of algorithm M and S_M be any subset in P_M. M provides the ϵ-differential privacy preserving if it satisfies the following formula:*

$$Pr[M(D) \in S_M] \leq e^{\epsilon} \cdot Pr[M(D') \in S_M], \tag{1}$$

where budget ϵ denotes the level of privacy leakage. Conceptually, smaller values of ϵ mean lower tolerances to the privacy leakage and hence requiring higher levels of noise perturbations that degrade the model accuracy.

Definition 2: ***((ϵ, δ)-Differential Privacy).*** *Algorithm M provides the (ϵ, δ)-differential privacy preserving if it satisfies the following formula:*

$$Pr[M(D) \in S_M] \leq e^{\epsilon} \cdot Pr[M(D') \in S_M] + \delta, \tag{2}$$

where parameter δ denotes the probability that the standard ϵ-differential privacy is broken. Hence, this definition provides a weaker *privacy guarantee. Note that the parameters ϵ and δ decide the theoretical upper bound of privacy degradation [30]. Specifically, in ϵ-differential privacy, the upper bound is $e^{\epsilon} - 1$. In (ϵ, δ)-differential privacy, the bound is $e^{\epsilon} - 1 + a \times \delta$, where $a \geq 1$ represents the number of algorithms used to produce machine learning models. When applying to DL, the values of these parameters depend on the trained model. For example, our empirical tests show that when setting δ to $1e^{-6}$, ϵ ranges from 0.03 to 1 in LeNet-5, and ranges from 10 to $1e^{3}$ in AlexNet. The parameters smaller than these values inject too much noises and hence considerably decrease model accuracy. The parameters larger than values have negligible impact on model accuracy because little noises are injected.*

Noise Mechanism. In ϵ-differential privacy [15] and (ϵ, δ)-differential privacy [2], the prevalent methods inject noises to sensitive data according to a predetermined distribution, such as the Laplace and Gaussian distributions. These methods, therefore, are termed *Laplace* mechanism [28] and *Gaussian* mechanism, which employ the L_1-norm and L_2-norm to estimate the differences between gradient outcomes with and without noise perturbation.

Definition 3: ***(Laplace Mechanism).*** *The mechanism controls the noise injection using the Laplace distribution and the L_1-norm sensitivity:*

$$M_{Lap}(x, f, \epsilon) = f(x) + Lap\left(\mu = 0, \beta = \frac{\Delta_1}{\epsilon}\right), \tag{3}$$

where μ is the mean of Laplace distribution, and β is the scale parameter calculated with the L_1-sensitivity Δ_1 and the privacy budget ϵ. In Eq. (3), we use $Lap(\beta)$ or $Lap(0, \beta)$ to represent the random value satisfying the Laplace distribution: $p(x) = \frac{1}{2\beta} exp(-\frac{|x-\mu|}{\beta})$.

Definition 4: ***(Gaussian Mechanism).*** *The mechanism controls the noise injection using the Gaussian distribution and* *the L_2-norm sensitivity:*

$$M_{Gaussian}(x, f, \epsilon, \delta) = f(x) + N\left(\mu = 0, \sigma^2 = \frac{2ln(1.25/\delta)(\Delta_2)^2}{\epsilon^2}\right) \tag{4}$$

where μ and σ^2 are the mean and variance of the normal distribution

Definition 5: ***(Sensitivity).*** *The sensitivity of function f over two adjacent datasets D and D' is calculated as the maximal distance between the function's outputs f(D) and f(D'):*

$$\Delta f = max|f(D) - f(D')|, \tag{5}$$

where $|f(D) - f(D')|$ represents the norm distance. In L_1-norm ($||\cdot||_1$) and L_2-norm ($||\cdot||_2$), the sensitivity is calculated as: $\Delta_1 f = \max||f(D) - f(D')||_1$ and $\Delta_2 f = max||f(D) - f(D')||_2$, respectively. Sensitivity estimation is deemed the most computation expensive component of differential privacy algorithms because it requires exhaustive computation on all possible pairs of (D, D'). Several estimation schemes were proposed by the related work [31].

3 Influential Factors of Privacy Guarantee in Differentially Private Model Training

In this section, we first introduce the threat model within the context of distributed DL on edge nodes, and explain how differential privacy preserves privacy by injecting noises into the released gradient updates (Section 3.1). Within this context, we systematically analyze the impact of noises from the perspective of local edge nodes (Section 3.2) and the entire training system (Section 3.3). Using image benchmarks, we show the challenges of achieving high accuracy given the local privacy budget and uncover factors that can minimize the overall accuracy loss and adhere to the privacy guarantee.

3.1 Threat Model in Edge-Based DL

We assume that K participants (where $K \geq 2$) jointly train a DL model in a federated learning setting, in which each participant generates gradients through interactions with her/his local edge node. In a typical model averaging setting [26], model updates correspond to a two-step aggregation at one iteration. First, every participant aggregates the gradients calculated using the local batch of data on the edge node. Second, a central server aggregates the gradients from all K participants. In the threat model, either the server or one of the participants can be the attacker that aims to infer information about other participants' training data during iterative model training. This inference is conducted by inspecting the gradient updates received from the server. For example, attackers can reconstruct the original images from gradient information [32].

Differential privacy addresses the above inference attacks by first applying a differentially private transformation to each participant's gradients and then transferring them to the central server [2], [3], [4]. Specifically, at each iteration, differential privacy adds noises to a participant's gradients according to the pre-specified privacy budget and other model training settings. All these variables may influence the amount of noises injected and thus affect the final model accuracy. In the following two sections, we explain the influential factors of noises at the edge node level (*local noises*) and at the master server level (that is, the *aggregated noises* from all edge nodes).

3.2 Influential Factors of Local Noises Per Edge Node

To study the impact of local privacy budget, we first rewrite the noise obfuscation level as a function of privacy budget (ϵ), data sensitivity (Δ), and the parameters of statistical distribution as following.

In the Laplace mechanism (Definition 3), the noise follows the Laplace distribution:

$$noise \sim Lap\left(\mu = 0, \beta = \frac{\Delta_1}{\epsilon}\right). \tag{6}$$

Similarly, the noise follows the Gaussian distribution in the Gaussian mechanism (Definition 4):

$$noise \sim N\left(\mu = 0, \sigma^2 = \frac{2ln(1.25/\delta)(\Delta_2)^2}{\epsilon^2}\right). \tag{7}$$

Gradient Clipping Techniques. Data sensitivity is defined as the maximum L_1 or L_2 norm of two adjacent gradient vectors. As sensitivity can have a wide range, the prior art proposes to clip the gradient ranges such that the sensitivity is analytically bounded and ease its computation overhead. In traditional database systems, the minimum and maximum values of data points in D and D' (Eq. (5)) are known before calculating sensitivity. However, when applying differential privacy to DL (that is, adding noises to gradients), it is difficult to know these values because the gradients vary across different models, datasets, and training iterations. For such an issue, clip techniques are proposed to enforce a range on gradient values in sensitivity calculation. We now define two prevalent clip techniques.

First, the *clip-by-norm1* technique [15] is designed to calculate the L_1-sensitivity following the Laplace mechanism. It utilizes a fixed, input-independent bound C to bound an original gradient $g_{original}$ in sensitivity estimation:

$$g = clip(-C, C, g_{original}) = \begin{cases} -C & g_{original} \leq -C \\ g_{original} & min_g < g_{original} < C \\ C & g_{original} \geq C \end{cases}. \tag{8}$$

Hence in the Laplace Mechanism, parameter β can be set as: $\beta = \frac{2C}{\epsilon}$.

Second, the *clip-by-norm2* technique [2] scales the original gradient $g_{original}$ to gradient g as follows:

$$g = g_0/max\left(1, \frac{||g_0||_2}{C}\right). \tag{9}$$

Note that this techniques also scales down the noises $(max(1, \frac{||g||_2}{C}))^{-1}$ times.

Sensitivity Estimation in the Clip Techniques. Let $Y = (g_1, g_2, \ldots, g_n)$ and $Y' = (g'_1, g'_2, \ldots, g'_n)$ be two vectors of

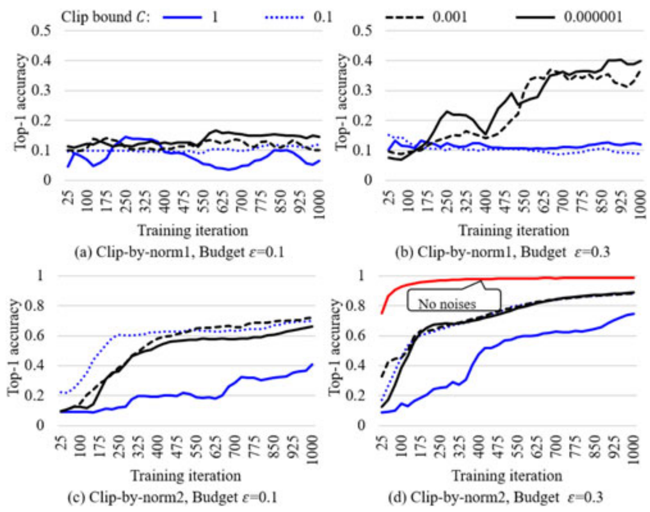

Fig. 2. Evaluations of effects of clip bounds on model accuracy.

gradients, and each g_i (g_i') is independent of others for $1 \leq i \leq n$. According to Eq. (5), the sensitivity in the clip-by-norm1 technique can be estimated as follows:

$$\Delta_1 = max||Y - Y'||_1 = n \times \max_{1 \leq i \leq n}(|g_i - g_i{}'|) = 2nC. \tag{10}$$

Similarly, in the clip-by-norm2 technique, the sensitivity is estimated as follows:

$$\begin{aligned}\Delta_2 &= max(||Y - Y'||_2) \\ &= \sqrt{\max(|g_1 - g_1{}'|)^2 + \ldots + \max(|g_n - g_n{}'|)^2} \\ &= \sqrt{n \times (2C)^2)} \\ &= 2\sqrt{n}C.\end{aligned} \tag{11}$$

Evaluation of Accuracy Degradation Under Different Clip Bounds and Privacy Budgets. We take LeNet-5 and the MNIST dataset [33] as an example to show the impact of *clip bounds* on the overall model accuracy. We tested both clip-by-norm1 and clip-by-norm2 using two privacy budgets (0.1 and 0.3) and four clip bounds (1, 0.1, 0.001, and 0.000001). The evaluation results in Fig. 2 show that: (i) the best clip bound varies when encountering different sampling techniques and budgets. For example, bound 0.000001 achieves the highest accuracy when the budget is 0.3 in the clip-by-norm1 technique (Fig. 2b). In contrast, bound 0.001 is the best one when the budget is 0.1 in the clip-by-norm2 technique (Fig. 2c); (ii) the setting of any bound considerably deteriorates the model accuracy compared to the one without noise injection (Fig. 2d). The average accuracy loss is 53.81 percent when considering all bounds.

Challenges. There is no "one-size-fits-all" best bound in existing clip techniques. The challenge of determining the optimal bound problem is further exacerbated when considering differentiated noise injection mechanisms in different DL models, datasets, and training iterations.

3.3 Influential Factors of Global Aggregated Noises

In this section, we first analytically capture the aggregated accuracy loss from the perspective of aggregating local noises, and then experimentally demonstrate their impact on the overall accuracy. Note that in the *distributed* model training, the *aggregated privacy loss* from multiple edge nodes differs from the sum of privacy losses in the *centralized* training. In the later one, the privacy losses come from different stages of the training and these stages have the *same* privacy barrier, which is the prerequisite for applying the composition theorem [12] in its privacy summarization. However, in the distributed training scenario, each edge node has its *own* privacy barrier and hence the composition theorem is not applicable. To the best of our knowledge, it is not clear how such aggregated obfuscation (noises) will impact the overall accuracy of distributed DL. We thus conduct a quantitative analysis that unveils the factors that influence the extent of aggregated noises and their resulting accuracy loss.

We let batch size per iteration and per node be B. Suppose the noises injected on each node follow the same distribution $Noise(\mu, \sigma)$, and the batch of data points processed by the kth node is $\{t_{i_k}, t_{i_k+1}, \ldots, t_{i_k+b_k-1}\}$. That is, the batch size of this node is b_k and $\sum_{b=1}^{K} b_k = B$. We have: (1) the sum of gradients reported by the kth node is $\sum_{i=i_k}^{i_k+b_k-1} g_i + Noise(\mu, \sigma)$; (2) the final gradient computed in the central server is:

$$g = \frac{1}{B}\left(\sum_{i=1}^{B} g_i + \sum_{k=1}^{K} X_k\right) \qquad X_k \sim Noise(\mu, \sigma). \tag{12}$$

According to the Central-Limit Theorem [34], we have:

$$\sum_{k=1}^{K} X_k \sim N(0, K \times Var(Noise(\mu, \sigma))) = N(\mu, K\sigma^2), \tag{13}$$

therefore, if K is large enough, the formula above can be:

$$g = \frac{1}{B}\left(\sum_{i=1}^{B} g_i + N(\mu, K\sigma^2)\right) = \bar{g} + N\left(\mu, \frac{K\sigma^2}{B^2}\right), \tag{14}$$

where the standard deviation $\sigma_{aggregate}$ is:

$$\sigma_{aggregate} = \frac{\sqrt{K}\sigma}{B}. \tag{15}$$

Hence, when fixing the standard deviation σ of local noise distribution and the edge node number K, the aggregated noise is inversely proportional to the batch size B.

Based on the Eq. (15), it is easy to know the aggregated noise in the Laplace mechanism is:

$$\sigma_{aggregate} = \frac{\sqrt{K}}{B}\left(\frac{\sqrt{2}\Delta_1}{\epsilon}\right). \tag{16}$$

Similarly, the aggregated noise in the Gaussian mechanism is:

$$\sigma_{aggregate} = \frac{\sqrt{K}}{B}\left(\frac{\sqrt{2ln(1.25/\delta)}(\Delta_2)}{\epsilon}\right). \tag{17}$$

Evaluation of Aggregated Noises. We now empirically demonstrate how the different batch sizes affect the aggregated

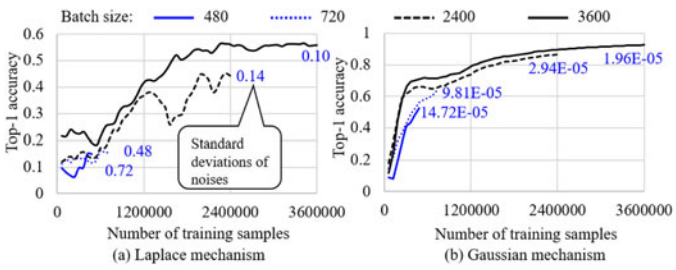

Fig. 3. Evaluations of effects of batch sizes on model accuracy.

noise and the model accuracy. Taking clip bound 0.001 and budget 0.3 (per node and per iteration) as an example, we tested four batch sizes (480, 720, 2,400, and 3,600) and Fig. 3 shows the fluctuating model accuracies across the training iterations. We can see that the batch size considerably affects the model accuracy in each evaluation, in which different numbers of training samples are processed. In particular, batch size 3600 has the highest accuracy and its accuracy loss is only 5.85 percent compared to the one without adding gradient noises. This is because this batch size brings the lowest noises according to Eqs. (16) and (17). Fig. 3 shows that, for batch sizes 480, 720, 2400, and 3600, the standard deviations of aggregated noises are 0.72, 0.48, 0.14, and 0.10, and 14.72E-05, 9.81E-05, 2.94E-05, and 1.96E-05, respectively. The lower noises cause less turbulence to the model and thus bring higher model accuracies.

Challenges. We note that in distributed DL model training, large batch size significantly decreases the aggregated noises in differential privacy. However, real edge nodes such as mobile devices usually have limited resources to process large datasets. Hence how to maintain large batch size while efficiently processing large datasets on resource-constrained edge nodes is another major challenge to be addressed.

4 Accurate Differentially Private DL

In this section, we first describe the design overview of PrivateDL in Section 4.1, following by explaining its unique modules: (i) sensitivity estimation to inject local gradient noises without clip bounds (Section 4.3) and (ii) reducing the aggregated accuracy loss by virtual batch size amplification (Section 4.2). Finally, we introduce our implementation of PrivateDL on KubeEdge, Spark, and PyTorch (Section 4.4).

4.1 Overview of PrivateDL

For a DL model trained across K edge nodes, PrivateDL is designed to reduce the noise perturbation of differential privacy during the model training process. Specifically, at each iteration, it reduces aggregated noises by dynamically amplifying batch sizes according to the ratio of critical input data, and injects less local gradient noises on each edge node based on accurate sampling-based sensitivity estimation. The four steps in an edge node are shown in Fig. 4.

Virtual Batch Size Amplification via Critical Set. In this module, step 1 first amplifies the batch size on the node to reduce the aggregated noise. Step 2 then identifies the critical data points in this batch of data, namely the data points relevant to model parameter updating. In the following

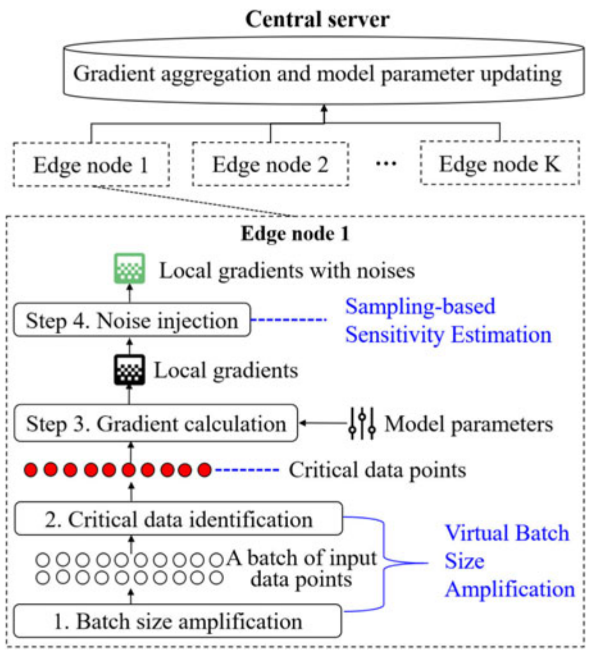
Central server
Gradient aggregation and model parameter updating
Edge node 1
Edge node 2
...
Edge node K
Edge node 1
Local gradients with noises
Step 4. Noise injection
Sampling-based Sensitivity Estimation
Local gradients
Step 3. Gradient calculation
Model parameters
Critical data points
2. Critical data identification
A batch of input data points
Virtual Batch Size Amplification
1. Batch size amplification

Fig. 4. Overall process of PrivateDL.

gradient calculation, step 3 only uses these points to save computational cost, while achieving very similar results to the entire batch of data.

Sampling-Based Sensitivity Estimation. Step 4 adds noises to these gradients according to the differential privacy mechanism. To reduce the noise perturbation, PrivateDL proposes a sampling method that accurately estimates sensitivity according to the latest model parameters and input data at each iteration.

4.2 Virtual Batch Size Amplification Via Critical Set

This module aims to increase the batch size such that the aggregated gradient noise and its impact on the overall model accuracy can be reduced (see Eq. (15)), without exceeding the resource constraints of single edge node. The enabling feature here is to only use the critical data in the amplified dataset in gradient calculation, thus lowering the computational time and resources on each edge node. Given a DL model trained across K edge nodes, Algorithm 1 details the steps of this process. At each iteration, it first amplifies the batch size on node i according to the ratio γ_i of critical input data at the previous iteration (line 6). At the first iteration, this ratio is set to 1 (lines 3 to 5). At other iterations, the original batch size B_i is amplified as:

$$B'_i = \frac{B_i}{\gamma_{i-1}}, 1 \leq i \leq K. \tag{18}$$

This amplification is based on the observation that the ratio gradually changes across iterations (e.g., from 50 to 70 percent in AlexNet), and the ratios of two adjacent iterations are very similar [19]. Hence the algorithm first samples a set T_i of size $\frac{B_i}{\gamma_{i-1}}$ (line 7), and then only uses the critical set whose size is similar to B_i in model training (line 8). Finally, the central server updates the aggregated noises according to the amplified datasets on all K edge nodes (line 11). That is, the standard deviation of aggregated noise (Eq. (15)) is calculated as:

$$\sigma_{aggregate} = \frac{\sqrt{K}\sigma}{\sum_{i=1}^{K} B'_i}. \tag{19}$$

Algorithm 1. Virtual Batch Size Amplification Via Critical Set

```
Require: $i$: the index of an edge node ($1 \le i \le K$);
  $T_i$: the local batch of input data on the $i$th edge node;
  $B_i$: the original batch size on the $i$th node;
  $C_i$: the critical set of input data processed on the $i$th node;
  $\gamma_i$: the ratio of critical input data in the $i$th node;
1.  while (the model training is not converged) do
2.    for ($i$=1; $i \le K$; $i$++) do
3.      if ($i$==1) then
4.        $\gamma_i = 1$;
5.      end if
6.      $B'_i = \frac{B_i}{\gamma_i}$;
7.      Sample a batch of $B'_i$ points in $T_i$; //Batch size amplification
8.      $C_i$= CriticalSet($T_i$); //Identify critical input data
9.      $\gamma_i = \frac{|C_i|}{B'_i}$;
10.   end for
11.   $\sigma_{aggregate} = \frac{\sqrt{K}\sigma}{\sum_{i=1}^{K} B'_i}$
12. end while
```

In Algorithm 1, function CriticalSet() identifies and extracts critical subset from the batch of input data points and it is developed based on SlimML [19] for three reasons. (1) *Low overheads*. SlimML generates aggregated data points to approximate the original input data points. That is, each aggregated data point corresponds to multiple original ones (e.g., 10 ones). The generation and processing of aggregated points, therefore, causes small overheads (less than 5 percent of model training time). (2) *High precision of identification*. By calculating the sum of model parameters' gradients for each aggregated data point, SlimML identifies redundant data points that have negligible effects on model parameter updating. For example, a data point whose cumulative effect is smaller than 1 percent of the total effects. These points' corresponding original data points are removed and the retained input data is critical. (3) *Applicability to SGD-based model training*. SlimML is developed for gradient descent based model training and hence it is applicable for a wide range of DL applications. Note that importance sampling [35] and coreset [36] also employ data points' loss values or gradients to select a subset of importance points to accelerate the training speed. However, this subset is kept unchanged and thus cannot adapt the changing gradients during the iterative training process. In contrast, SlimML calculates data points' gradients at the beginning of each training iteration and dynamically decides the ratio of critical input data (namely the amplified batch size).

4.3 Sampling-Based Sensitivity Estimation

Existing differential privacy techniques suffer from finding proper clipping bounds for gradients, because any bound may become oversize or undersize across the training iterations. To this end, PrivateDL designs a sampling-based module to calculate the data sensitivity of differential privacy without clipping bounds. This module is based on the observation of the 3-sigma rule [37]: for arbitrary distribution of gradients, most values (e.g., at least 88.89 percent) range from $(\mu - 3\sigma)$ to $(\mu + 3\sigma)$. For example, in LeNet and AlexNet, when gradients follow the unimodal distribution, over 95 percent of the values fall into $(\mu - 3\sigma, \mu + 3\sigma)$.

Definition 6: ***(L_p-Sensitivity Δ_p in the Sampling Method).*** *Suppose $\Delta_p = max||Y - Y'||_p$, where Y and Y' are two arbitrary gradient vectors computed by function $f(.)$ and $p \ge 1$. Let $||Y - Y'||_p$ be a random variable whose expectation is $E(||Y - Y'||_p)$ and standard deviation is $\sqrt{Var(||Y - Y'||_p)}$, sensitivity Δ_p is calculated as the maximal value of $||Y - Y'||_p$ according to the 3-sigma rule:*

$$\Delta_p = E(||Y - Y'||_p) + 3\sqrt{Var(||Y - Y'||_p)}, \tag{20}$$

Let n be the number of gradients and each gradient g satisfies the same and arbitrary distribution whose expectation and standard deviation are μ and σ. We deduce the calculation of L_1- and L_2-sensitivity.

Proposition 1: *Assume that two gradients g and g' are independent of each other, we have $E(g - g')^2 = 2\sigma^2$.*

Proof:

$$\begin{aligned} E((g - g')^2) &= Var(g - g') + E^2(g - g') \\ &= Var(g) + Var(g') + (E(g) - E(g'))^2 \\ &= \sigma^2 + \sigma^2 + (\mu - \mu)^2 = 2\sigma^2 \end{aligned} \tag{21}$$

□

Lemma 1: *The L_1-sensitivity $\Delta_1 = (\sqrt{2}n + (3\sqrt{2n}))\sigma$.*

Proof: Let $Y = (g_1, g_2, \ldots, g_n)$ and $Y' = (g'_1, g'_2, \ldots, g'_n)$ be two gradient vectors, where each g_i (g'_i) is independent of each other for $1 \le i \le n$. The L_1-norm utilizes the Manhattan distance between Y and Y': $MD = |g_1 - g'_1| + |g_2 - g'_2| + \ldots + |g_n - g'_n|$. According to Eq. (20) and Proposition 1, we have:

$$\begin{aligned} \Delta_1 &= E(||Y - Y'||_1) + 3\sqrt{Var(||Y - Y'||_1)} \\ &= \sum_{i=1}^{n} E(|g_i - g'_i|) + 3\sqrt{\sum_{i=1}^{n} Var(|g_i - g'_i|)}. \end{aligned} \tag{22}$$

Given that $Var(|g - g'|) = E((g - g')^2) - E^2(|g - g'|) \ge 0$ and $E((g - g')^2 = 2\sigma^2$ (Proposition 1), we have

$$E((g - g')^2) - E^2(|g - g'|) \ge 0 E(|g - g'|) \le \sqrt{2}\sigma \tag{23}$$

By substituting Eq. (23) into Eq. (22),

$$\begin{aligned} \Delta_1 &\le \sum_{i=1}^{n}(\sqrt{2}\sigma) + 3\sqrt{\sum_{i=1}^{n} E((g_i - g'_i)^2))} \\ &= \sqrt{2}n\sigma + 3\sqrt{\sum_{i=1}^{n} 2\sigma^2} \\ &= (\sqrt{2}n + (3\sqrt{2n}))\sigma \end{aligned} \tag{24}$$

□

Lemma 2: *The L_2-sensitivity $\Delta_2 = (\sqrt{20n})\sigma$.*

Proof: Similar to the computation of L_1-sensitivity, the distance between Y and Y' in the L_2-norm can calculated as: $||Y - Y'||_2 = \sqrt{(g_1 - g_1')^2 + (g_2 - g_2')^2 + ... + (g_n - g_n')^2}$. The L_2-sensitivity is calculated as:

$$\begin{aligned}\Delta_2 &= E(||Y - Y'||_2) + 3\sqrt{Var(||Y - Y'||_2)}\\ &= E(||Y - Y'||_2)\\ &\quad + 3\sqrt{E(||Y - Y'||_2^2) - E^2(||Y - Y'||_2)}.\end{aligned} \tag{25}$$

In Eq. (25), $E(||Y - Y'||_2^2) = E(\sum_{i=1}^n (g_i - g_i')^2) = \sum_{i=1}^n E(g_i - g_i')^2 = 2n\sigma^2$ (Proposition 1). According to the arithmetic mean-geometric mean (AM-GM) inequality, we have

$$\begin{aligned}\Delta_2 &= E(||Y - Y'||_2) + 3\sqrt{2n\sigma^2 - E^2(||Y - Y'||_2)}\\ &= \frac{10}{3}\left(\frac{(1/3)3E(||Y - Y'||_2) + 3\sqrt{2n\sigma^2 - E(||Y - Y'||_2)^2}}{(1/3) + 3}\right)\\ &\le \frac{10}{3}\sqrt{\frac{(1/3)(3E(||Y - Y'||_2))^2 + 3(2n\sigma^2 - E(||Y - Y'||_2)^2)}{10/3}}\\ &= (\sqrt{20n})\sigma\end{aligned} \tag{26}$$

At each training iteration, the module calculates the standard deviation σ of gradients on each edge node, then estimates its L_1-norm sensitivity (Eq. (24)) or L_2-norm sensitivity (Eq. (26)). Hence the *estimation error of sensitivity* is determined by the *estimation error of standard deviation* σ in this module. Let $\bar{\sigma}$ be the actual value of σ estimated using the entire gradient space, and S be estimated value of σ using m samples (m is smaller than the number n of model parameters), the estimation error

$$e = 1 - \frac{S}{\bar{\sigma}}. \tag{27}$$

□

Lemma 3: *Given batch size m and confidence level α, let $\bar{\mu}_4$ be the 4th central moment of gradients, the estimation error of standard deviation σ: $e < 1 - \sqrt{1 + \frac{\Phi^{-1}(1-\alpha)\sqrt{\mu_4/\sigma^4 - 1}}{\sqrt{m}}}$*

Proof: Let S^2 be the variance of gradient estimated using m gradients, the expectation and variance of S^2 are:

$$\begin{cases} E(S^2) = \sigma^2 \\ Var(S^2) = \frac{1}{m}(\mu_4 - \frac{m-3}{m-1}\sigma^4) \end{cases}. \tag{28}$$

□

Given that DL models usually have thousands of or even millions of parameters (i.e., gradients), batch size m can be a large value. According to Central-Limit Theorem [34], we can approximate the distribution of S^2 as a normal distribution:

$$S^2 \sim N(\sigma^2, \frac{\mu_4 - \sigma^4}{m}) \tag{29}$$

That is,

$$\frac{\sqrt{m}(S^2 - \sigma^2)}{\sqrt{\mu_4 - \sigma^4}} = \frac{\sqrt{m}((S/\sigma)^2 - 1)}{\sqrt{\mu_4/\sigma^4 - 1}} \sim N(0, 1). \tag{30}$$

Where μ_4/σ^4 represents the kurtosis of the gradient distribution. Hence the distribution of $\frac{S}{\sigma}$ is

$$\tfrac{S}{\sigma} \sim \sqrt{1 + \tfrac{N(0,1)\sqrt{\mu_4/\sigma^4 - 1}}{\sqrt{m}}}. \tag{31}$$

Given confidence level α, the confidence interval of $\frac{S}{\sigma}$ is

$$\left(\sqrt{1 + \tfrac{\Phi^{-1}(1-\alpha)\sqrt{\mu_4/\sigma^4 - 1}}{\sqrt{m}}}, +\infty\right). \tag{32}$$

Eq. (32) indicates that when the sampling size is m, there exists a probability α that $\frac{S}{\sigma}$ is larger than the lower bound of the confidence interval. According to Eq. (27), the estimation error of standard deviation e is smaller than $1 - \sqrt{1 + \frac{\Phi^{-1}(1-\alpha)\sqrt{\mu_4/\sigma^4 - 1}}{\sqrt{m}}}$ given sample size m and confidence level α.

Example of Estimation Error. Based on the above analysis, we consider two factors when applying our sampling based approach in DL training: (1) *confidence level* α: the training process usually consists of hundreds of thousands of iterations, this means we need a sufficiently high confidence level α to guarantee a small estimation error. (2) *Batch size* m: DL models usually have a large number of model parameters (gradients), e.g., 60 k parameters in LeNet-5 and 61.5 million parameters in AlexNet, thus allowing a large batch size m. In this example, we set $\alpha = 99.99999\%$, $m = 10\,k$, the estimation error $e = 3.5\%$ when gradient follows the normal distribution (that is, $\mu_4/\sigma^4 = 3$). Similarity, $e = 2.53\%$ when $\mu_4/\sigma^4 = 2$ and $e = 4.43\%$ when $\mu_4/\sigma^4 = 4$. These results show that our sampling based approach can provide low estimation errors of gradient variance with a sufficiently high probability. In addition, our batch size amplification approach influences the estimation error of gradient variance (namely data sensitivity) from another aspect. In gradient calculation, large batch sizes decrease fluctuations in gradient values and thus may reduce estimation errors.

4.4 Implementation

PrivateDL is implemented using Java and Scala and it is designed for KubeEdge [20], an emerging edge computing platform extended from Kubenetes [21]. Kubenetes is a dominant engine to orchestrate containers and schedule jobs in today's cloud datacenters [38]. Its design is a loosely coupled architecture, which allows components to interact with each other only using API-Server. The current version of Kubernetes only supports nodes with rich computational resources. KubeEdge therefore is designed to manage resources in edge devices with limited resources.

PrivateDL currently targets at DL jobs running in KubeEdge. Fig. 5 demonstrates how the edge and cloud parts of KubeEdge collaboratively completes a training iteration using a list of components. Specifically, in the *edge* part, KubeEdge uses the *MetaManager* component to hold a partition of model parameters and a local input dataset on

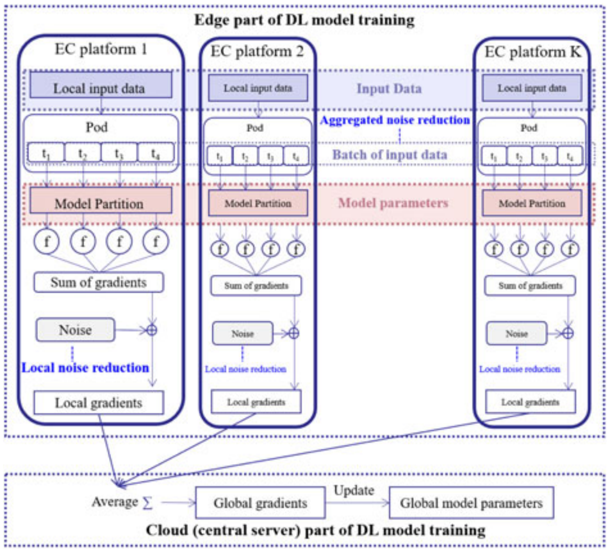

Fig. 5. Implementation of PrivateDL on KubeEdge.

each edge node. At the beginning of each iteration, the input dataset is partitioned into multiple containers (in a *pod*) for parallel execution. Each container executes a parallel DL task, which processes a subset t_i of input data to compute gradients of the model partition. After completing all tasks, the gradients are summarized and injected noises to preserve privacy. Our *sensitivity estimation and virtual batch size amplification* modules are implemented in this process. Finally, the *cloud (central server)* part collects gradients from all K edge nodes. In the collection, the communication between edge nodes and the master node is implemented based on the *EdgeHub* and the *CloudHub* components. The cloud part then calculates the average of the gradients and uses it to update the global model parameters.

We choose to implement the above distributed training process in KubeEdge with three objectives. (1) *Efficient synchronization between cloud and edge parts*. During the model training process, KubeEdge supports the efficient synchronization between the server and edge nodes with message packing and web socket [39]. Specifically, an edge node uses the *EdgeHub* component to report local gradients, the server uses the *CloudHub* component to collect gradients and send back the latest global parameters. (2) *Supporting devices with limited resources*. Inherited from Kubenetes, KubeEdge develops a lightweight version of the *kubelet* component to enable DL tasks running on devices with limited resources (e.g., NVIDIA Jetson). (3) *Applicability to heterogeneous machines*. KubeEdge employs the *EventBus* component to transform different machine/device protocols into a unified one (MQTT [40]). Hence in a DL model training, heterogeneous edge nodes can communicate with each other.

Implemented Noise Mechanisms and Distributed DL Applications. To evaluate the effectiveness of our approach, we incorporated it into two representative noise mechanisms in differential privacy: ϵ-differential privacy [15] (Laplace distribution) and (ϵ, δ)-differential privacy [2] (Gaussian distribution). The original versions of these mechanism only support DL model training in a *standalone* node. To this end, we implemented the distributed version of these noise mechanisms based on Apache Spark [41] (a mainstream platform built upon the MapReduce paradigm and supports in-memory computing using the resilient distributed dataset (RDD) data structure [42]) and PyTorch [23] (an open-source DL library designed for GPUs). Moreover, we incorporated PrivateDL with the DL applications in Intel BigDL [22], a library that provides rich DL applications and supports their training with parameter server paradigm and acceleration techniques [43].

TABLE 2
Comparative Results in Local Noise Reduction With Different Privacy Levels

Model	Norm	Privacy level (budget ϵ)	Accuracy improvement (percentage)
LeNet-5	1	0.1	216.52%
		0.3	272.76%
		0.5	247.10%
		1	240.95%
	2	0.03	48.03%
		0.1	15.47%
		0.3	3.94%
		1	1.50%
AlexNet	1	$5e^5$	383.90%
		$1e^6$	355.79%
		$5e^6$	282.32%
		$1e^7$	270.97%
	2	10	37.66%
		$1e^2$	28.48%
		$5e^2$	23.72%
		$1e^3$	12.97%
ResNet-18	1	$1e^4$	2668.28%
		$5e^4$	2765.87%
		$1e^5$	1755.99%
		$1e^6$	139.90%
	2	1	40.82%
		5	32.37%
		10	24.49%
		50	9.82%

5 Evaluation

Based on the implementation of PrivateDL on KubeEdge, PyTorch, and BigDL, our evaluation on real DL applications and datasets has three objectives. First, we compare PrivateDL against existing gradient clipping techniques to evaluate the proposed sensitivity estimation and its impact on overall accuracy from the perspective of local noise reduction (Section 5.2). Second, we present the comparison results across multiple edge nodes, further highlighting the effectiveness of PrivateDL in reducing aggregated noises (Section 5.3). Finally, we discuss the effectivenss and applicability of our approach (Section 5.4). Tables 2 and 3 summarize the improvement of PrivateDL over the state-of-the-art: how higher accuracy improvement can be achieved under different privacy levels (privacy budgets). We can see that using our approach, the reduction of *local noises* achieves more accuracy improvements for higher privacy levels. Moreover, we tested the low privacy levels when the influence of local noises is small, the results show that the reduction of *aggregated noises* still improves accuracy by an average of 120.37 percent.

TABLE 3
Comparative Results in Aggregated Noise Reduction With Low Privacy Levels (High Budgets)

Model	Norm	Privacy level (budget ϵ)	Accuracy improvement (percentage)
LeNet-5	1	0.3	326.30%
	2	0.3	57.70%
AlexNet	1	$1e^7$	58.01%
	2	$1e^3$	24.59%
ResNet-18	1	$1e^5$	186.68%
	2	50	68.95%

5.1 Experimental Settings

Experimental Environment. The experiments are performed in three types of heterogeneous edge nodes, including one CPU platform, one GPU platform, and one NVIDIA Jetson platform. On the CPU platform, each node is equipped with 18-core Intel E5-2695 v4 processors, and 256 GB of DRAM. One GPU node has 12-core Intel(R) Core(TM) i7-8700K processors, 12-GB TITAN Xp Graphics Card, and 64 GB memory. One Jetson TX2 node has the NVIDIA Pascal framework of 256 CUDA cores, one 2-core Denver processor, one 4-core ARM A57 Complex prcessor, and 8 GB of LPDDR4. All nodes run Linux Ubuntu 18.04. In the Spark cluster, the JDK, Go, Docker, Spark, and KubeEdge versions are 1.8.0.181, 1.12.9, 18.09.7, 2.4.3, and v1.15.3 respectively. In the PyTorch cluster, the python, pytorch, cuda, cudnn, and redis versionts are 3.7.7, 1.6.0, 10.1, 7.6.3, and 4.0.9, respectively.

Tested Workloads and Datasets. We test three DL models (LeNet-5, AlexNet, and ResNet-18) based on the implementation of PrivateDL on Intel BigDL and PyTorch. All models consist of multiple layers, in which the pooling/ReLU layers implement fixed functions, and each convolutional/full connected (FC) layer has multiple neurons ranging from 128 to 4,096. In evaluation, LeNet-5 [44] (60k parameters) and AlexNet [45] (61.5 million parameters [45]), and ResNet-18 [46] are tested using the MNIST dataset [33], the Cifar-10 dataset [47], and the ImageNet32×32 dataset [48], respectively. Both MNIST and Cifar-10 datasets have 60k data points and the ratios of training and testing points are 0.8 and 0.2. ImageNet32×32 has 1.28 million data points (downsampled 32×32 images) from 1000 classes and 50k testing points (50 ones per class).

Baseline Comparisons. To the best of our knowledge, our approach is the first technique that performs sampling-based sensitivity estimation. Hence, we compare against baselines with representative clip techniques developed for L_1-norm and L_2-norm. Specifically, in the clip-by-norm1 technique, the sensitivity is estimated as $\Delta_1 = 2nC$ (Eq. (10)), where n is the number of model parameters and C is the clip bound. In the clip-by-norm2 technique, the sensitivity is estimated as $\Delta_1 = 2\sqrt{n}C$ (Eq. (11)). Hence in evaluation, we set smaller bounds C for models with larger number n of parameters (e.g., AlexNet), and also set larger budgets for the clip-by-norm1 technique, because it injects larger noises for the same values of n and C compared to the clip-by-norm2 technique.

Evaluation Metrics. For a fair comparison, we report the *accuracy* of different techniques under the same training time and budget in differential privacy. The *accuracy* metric is top-1 classification accuracy on the test set: the top 1 predicted class (the one having the highest probability) is the same as the target/actual class label.

5.2 Evaluation of Local Noise Reduction

The effectiveness of PrivateDL is considerably impacted by its ability to find appropriate sigma of gradients in sensitivity estimation (Eqs. (22) and (25)). Given a privacy budget, this estimation decides the intensity of noise obfuscation that thus affects the model accuracy.

Evaluation Scenarios. We test different cases of differentially private training from three aspects: (1) *six model training settings*, including three DL models (LeNet-5, AlexNet, and ResNet-18), and two clip techniques (clip-by-norm1 and clip-by-norm2) that correspond to two noise mechanisms (Laplace and Gaussian). (2) *Six privacy budgets*. Each training setting has a distinct range of budgets, which represent its privacy levels in model training. Note that the model training will not converge for the budget smaller than this range, and the budget larger than this range may have a negligible impact on model accuracy. Specifically, the ranges of budgets are 0.1 to 1, 0.03 to 1, $5e^5$ to $1e^7$, 10 to $1e^3$, $1e^4$ to $1e^6$, 1 to 50 for LeNet-5 (clip-by-norm1), LeNet-5 (clip-by-norm2), AlexNet (clip-by-norm1), and AlexNet (clip-by-norm2), ResNet-18 (clip-by-norm1), and ResNet-18 (clip-by-norm2), respectively. (3) *Three clip bounds*. In existing clip techniques, the clip bound should be manually set according to the values of gradients in model training. Hence each DL model is tested using three bounds: 0.00001, 0.001, 0.1 for LeNet-5, 0.01, 0.03, 0.1 for AlexNet, and 0.1, 0.01, 0.001 for ResNet-18, respectively. Note that we set these values according to our empirical evaluations: for the *upper bound*, we decide its value using a list of steps. The first step starts from a large clip bound and the following steps gradually decrease its value (e.g., the value is reduced by 10 times between two consecutive steps). Given that large bounds incur high noises (Eqs. (10) and (11)) and thus may result in non convergence in model training, each step uses five epoches to test the convergent tendency. The *time complexity* of each epoch is $O(n * i * d)$, where n, i, and d are the number of model parameters, the number of training samples, and the dimensionality of each sample, respectively. We decide the *lower bound* in a similar way. An undersize bound also leads to non convergence in model training because it clips most of the useful gradients and only retains gradients of small values (that is, small impacts on model parameter updating). Suppose five steps are used in either upper or lower bound, the evaluation results show the preprocessing takes 897.5, 310,386.67, and 15,091.7 seconds for LeNet-5, AlexNet, and ResNet-18, respectively. In contrast, our method needs no preprocessing and calculates the standard deviation at each iteration of model training using m samples of gradient. The time complexity of this calculation is $O(m)$ and the evaluation results show it complete within 2 milliseconds.

Model Training Settings. The Adam method [49] is used in model training and the learning rate is set to 0.001 for LeNet-5 and 0.0002 for AlexNet and ResNet-18. Other hyperparameters follow the default values in the BigDL library [22]. At each iteration, the batch size is 2,400 for LeNet-5, 120 for AlexNet, and 256 for ResNet-18. The trainings of LeNet-5, AlexNet, and ResNet-18 take 1,000, 8,340,

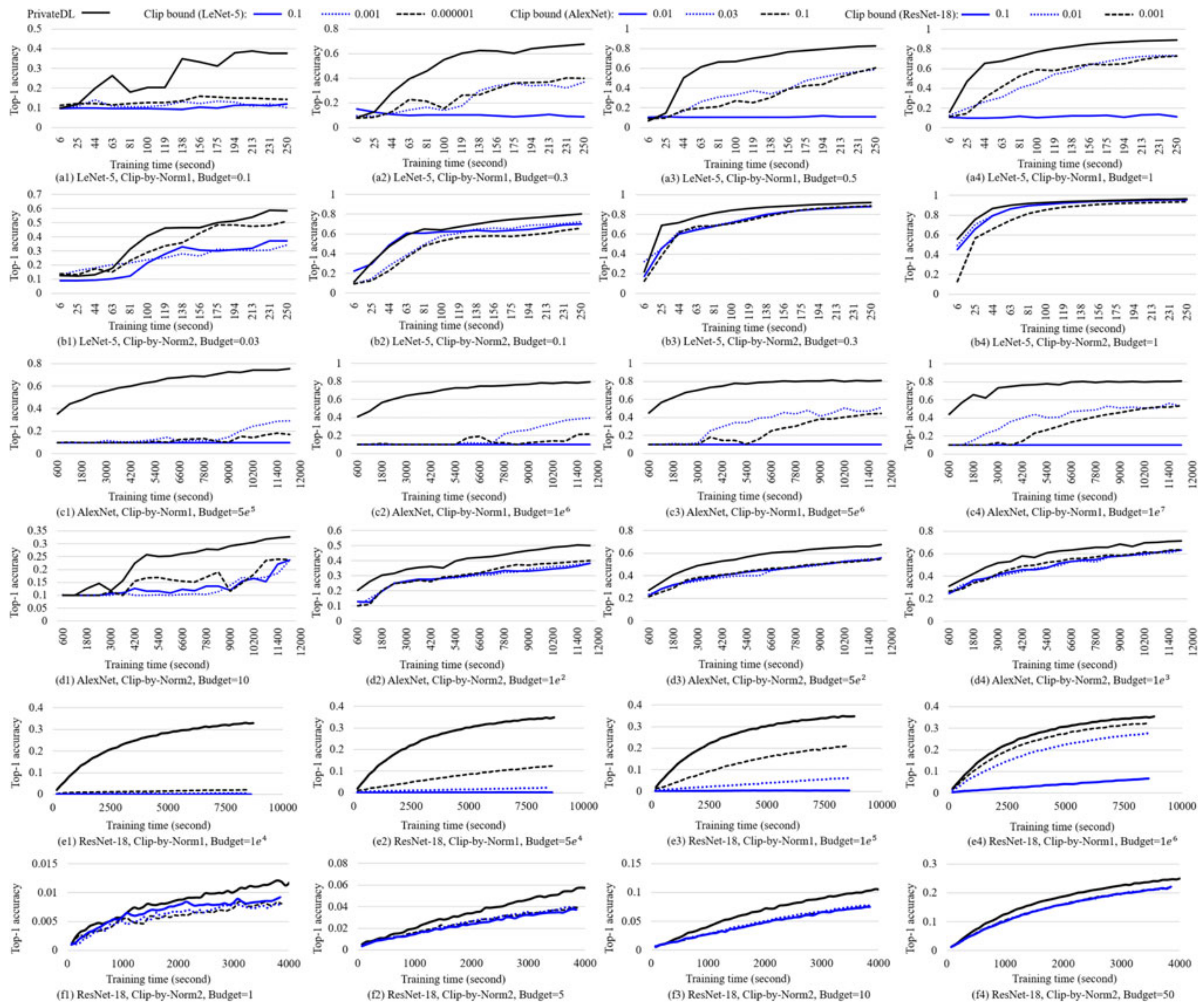

Fig. 6. Comparison of model accuracies between PrivateDL and the clip techniques under different privacy budgets.

and 12,000 iterations to converge, respectively. In comparative evaluations, we use the same hyperparameters and initial model parameters. During training, we test the model accuracy after each epoch and then retrain the model. In LeNet-5, AlexNet, and ResNet-18, the testing and retraining occur every 20, 6000, and 175 seconds, respectively.

Evaluation Results. Fig. 6 shows the model accuracies between PrivateDL and the clip techniques under different experimental settings. We can see that PrivateDL consistently provides higher accuracies because it dynamically estimates the sensitivity according to the latest gradients at each iteration. In contrast, the sensitivity estimation is fixed in the clip techniques (Eqs. (10) and (11)), and there is no ”one-size-fits-all” best bound for different training tasks. For example, bound 0.0001 achieves the highest accuracy among three bounds when the budget is 0.03 in LeNet-5 (Fig. 6b1). However, this bound results in the lowest accuracy when the budget is 0.1 (Fig. 6b2). In addition, a smaller budget represents a higher level of privacy guarantee, but also incurs larger noises and lower model accuracies. Evaluation results show that PrivateDL suffers less from small budgets because it can better estimate the sensitivity by dynamically adapting to the changing gradients during iterative training, and hence injects the lowest noises according to Eqs. (6) and (7). We also observe that model accuracy generally increases across the training itertions. During the test phases, the stability of model accuracy is inversely proportional to the amount of injected noises. For example, Figs. 6a1, 6b1, and 6d1 show that the models with the highest noises have the largest flucactions in test accuracies.

Results. When considering different DL training settings, PrivateDL increases model accuracy by an average of 411.65 percent compared to the existing clip techniques, and the accuracy increase is 565.87 percent for the highest privacy level (the smallest privacy budget).

5.3 Evaluation of Aggregated Noise Reduction

In distributed model training, aggregated noise has two major influential factors: the batch size B that represents the total number of input data points processed across all edge nodes, and the number K of nodes. In this section, we evaluate PrivateDL's effectiveness in reducing aggregated noises with consideration of these factors.

Evaluation of Batch Size B. The evaluation in the previous section shows that the accuracy improvements are small when the privacy budgets are high. Here, we extend this

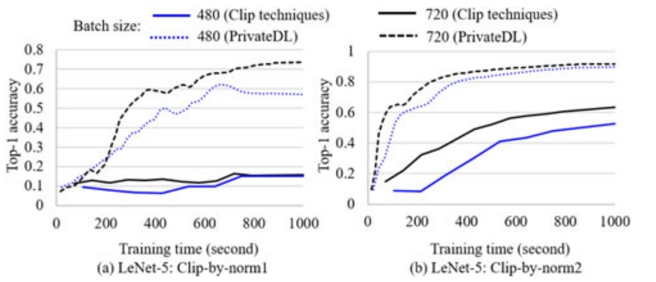

Fig. 7. Comparison of model accuracies in LeNet-5 under different batch sizes.

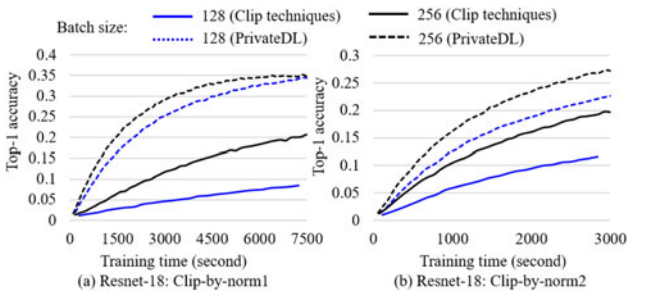

Fig. 9. Comparison of model accuracies in ResNet-18 under different batch sizes.

comparison evaluation to four EC nodes under low privacy levels (high privacy budgets) where local noises are small. Specifically, in LeNet-5, the clip bound is set to 0.001 and the privacy budget is set to 0.3; in AlexNet, the clip bound is set to 0.03, the privacy budget is $1e^7$ for the clip-by-norm1 technique and $1e^3$ for the clip-by-norm2 technique; in ResNet-18, the the clip bound is set to 0.001, the privacy budget is $1e^5$ for the clip-by-norm1 technique and 50 for the clip-by-norm2 technique. For the aggregate noise, we evaluate two different batch sizes for all models: 480, 720 for LeNet-5; 60, 120 for AlexNet; and 128, 256 for ResNet-18. The evaluation of AlexNet has the smallest batch sizes because it has the largest number of parameters and uses the most resources in gradient calculation. In redundant input data removal, each aggregated data point corresponds to 10 original data points and an aggregated data point is removed if its summarized gradient is smaller than 1 percent of the total gradients.

Evaluation Results. Figs. 7, 8, and 9 show the comparing results for three DL models. We can see that in all cases, a larger batch size B brings a higher model accuracy, which verifies the analysis of aggregated noises in Eqs. (16) and (17). Hence for the same batch size, PrivateDL achieves a higher accuracy because it amplifies the batch size before model training. At the same time, our approach removes the redundant data points at the gradient calculation stage to save computation costs. Hence the gradient calculation time in our approach is very similar to that of the clip techniques for two reasons: (1) the aggregated data points are only generated once before training and the generation time is two to three orders of magnitude shorter than that of model training time. At each iteration, the processing time of aggregated data points takes less than 5 percent of the total model training time; (2) the amplified ratio is set according to the removal ratio at the previous iteration such that the redundant input data removal efficiently decreases the required calculations of data points.

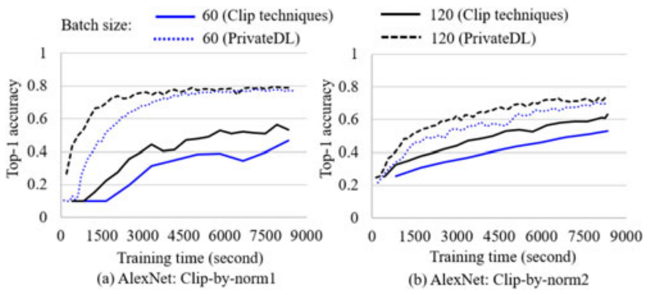

Fig. 8. Comparison of model accuracies in AlexNet under different batch sizes.

Evaluation of Node Number K. In this evaluation, we take AlexNet as an example and test its model training in a cluster of 100 edge nodes (each one has two CPU cores and 10 GB memory). For both clip-by-norm1 and clip-by-norm2 techniques, we tested two privacy budgets and two batch sizes, as shown in Fig. 10. The evaluation results show that in this larger training deployment, the model accuracies in both PrivateDL and the clip techniques are considerably lower than those (Figs. 6c1 to 6d4) in the smaller deployment (four nodes) when setting the same or larger privacy budgets. With larger injected noises, the accuracies in the clip-by-norm1 and clip-by-norm2 techniques start decreasing in the mediate stage of model training. PrivateDL suffers less from such noise perturbation. In particular, it provides sufficient small noises in the L_1-norm (Figs. 10a and 10b) and avoids model accuracy degradation. Figs. 10a and 10b show that when the batch size increases from 400 to 1000, the accuracy improvement is small in PrivateDL. This is becasue in addition to batch size, the accuracy improvement (namely the noise reduction) also depends on privacy budget accoring to Eqs. (16) and (17). The budgets in the L_1-norm are $5e^3$ to e^4 times larger than those of the L_2-norm. Hence both the noises and the decreases in noise (when batch size increases) in the L_1-norm are small. We note that the accuracy improvement will become even smaller when the batch size further increases, because the amount of noise reduction gradually decreases when the batch size increases. In Fig. 3a's example, the noise reduction is 0.24 when batch size increases from 480 to 720, and this reduction is only 0.04 between when batch size increases from 2400 to 3600.

Discussion of Training Time. We can also observe that in Fig. 10's large cluster, the training time is much longer than that of Fig. 6's small cluster. This is because when the node number is small ($K = 4$), each iteration's training time mainly consists of two parts: the gradient calculation time (determined by model size and training sample) and the noise injection time (taking 10 percent of the whole training time). When the node number increases to 100, the synchronization among these nodes becomes the bottleneck of training time, because the standard Bulk Synchronous Parallel (BSP) used here only allows the training to move to the next iteartion when all the nodes complete their trainings. We note that this synchronization is time consuming for large clusters. In the future, we plan to study asynchronous training techniques such as Asynchronous Parallel (ASP) [50] or Stale Synchronous Parallel (SSP) [51] that relaxes BSP by allowing training to proceed to the next iteration once all

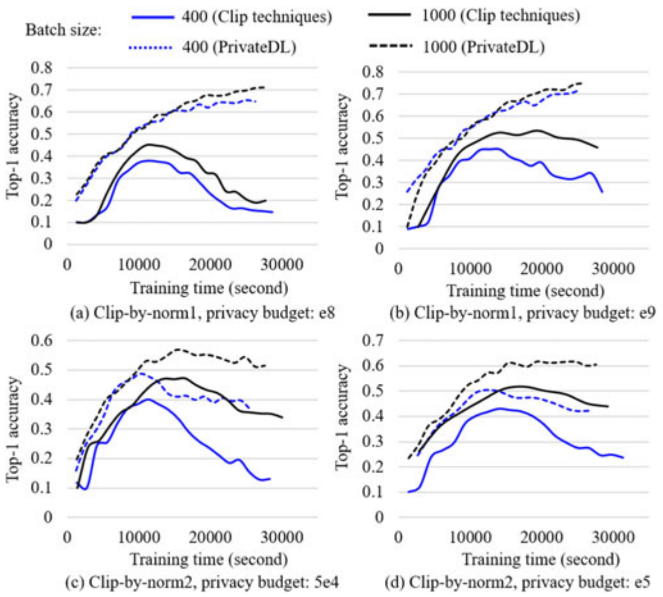

Fig. 10. Comparison of model accuracies in AlexNet under 100 edge nodes.

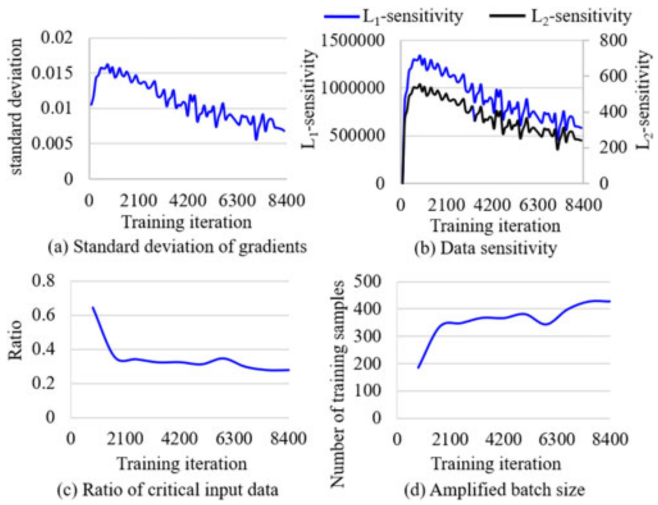

Fig. 11. Discussion of clip bounds and data sensitivity in PrivateDL.

nodes' iterations are within a pre-defined limit with each other. For ASP and SSP, the calculation of aggregated noise (Eqs. (16) and (17)) needs to be revisited according to the K nodes' synchronization setting.

Results. By reducing aggregated noises, PrivateDL increases the model accuracy by an average of 131.88 percent in the evaluations of the LeNet-5, AlexNet, and ResNet-18 workloads.

5.4 Discussions

Differential privacy provides the notion of trade-off between model accuracy and data privacy via noise injection. PrivateDL exploits such a trade-off in a distributed DL training, in which the intensity of noises is mainly influenced by the calculation of sensitivity (Eqs. (6) and (7)). In this section, we first discuss several practical factors that influence the sensitivity calculation. We then discuss the privacy preservation increased with the proposed approach, and the applicability of our approach to a mobile environment.

Discussion of Clip Bounds and Batch Size. In existing clip techniques, data sensitivity is calculated using pre-defined clip bounds (Eqs. (10) and (11)). In contrast, at each iteration of model training, PriviateDL dyanmically calculates the "clip bound" in terms of the standard deviation of gradients and directly uses the standard deviation to compute sensitivity. Fig. 11a shows that the standard deviation gradually decreases across the training iterations, because the value of gradient becomes smaller when the training is close to converge. According to Lemmas 1 and 2, both L_1-sensitivity and L_2-sensitivity are linearly proportional to the standard deviation and thus Fig. 11b shows that both types of sensitivity have a similar trend as the standard deviation. Moreover, Fig. 11c displays the ratios of critical input data during the training process. According to Eq. (18), a smaller ratio brings a larger amplification in batch size, as shown in Fig. 11d, thus achiving more reductions in aggregated noises.

Discussion of Sampling Interval. In PrivateDL, the sampling interval is a key parameter that determines the range of gradients being used to calculate the sensitivity. The variations in the sampling interval therefore affect the noise injection and the mode accuracy during the training. In previous scenarios, the sample interval is set as ($\mu - 3\sigma$, $\mu + 3\sigma$) that cover about 95 percent of the gradients. In this evaluation, we take LeNet-5 as an example and test five sampling intervals, ranging from ($\mu - 1\sigma$, $\mu + 1\sigma$) to ($\mu - 10\sigma$, $\mu + 10\sigma$), following the evaluation settings of the previous sections. The results in Fig. 12 show that smaller intervals bring slightly higher accuracies at early iterations of model training. This is because DL models usually have an even distribution of gradients at early training stages, as shown in Fig. 14a's gradient distribution at iteration 10. At the same time, there are some outliers of gradients. Hence a smaller sampling interval such as ($\mu - 1\sigma, \mu + 1\sigma$) can avoid the disturbance of these outliers in sensitivity calculation. Figs. 14b and 14c also show that at later iterations, most of the gradients belong to a small interval around the mean and there are few outliers. This observation explains the results in Fig. 12 that different sampling intervals have small influences on model accuracy at later training stages, because all intervals cover most of the gradients.

Discussion of Integration With Clip Techniques. We integrate our approach with the two clip techniques for DL [2], [15]. At each iteration, we first define a bound of gradients and then only sample within the bound. In evaluation, five bounds ranging from 0.0000001 to 0.1 are tested and other settings follow the previous sections (the privacy budget is 0.3). Fig. 13 shows the two smallest bounds (0.0000001 and 0.00001) result in the lowest accuracies. This is because the values of gradients range from -0.02 to 0.02 (as shown in Fig. 14),

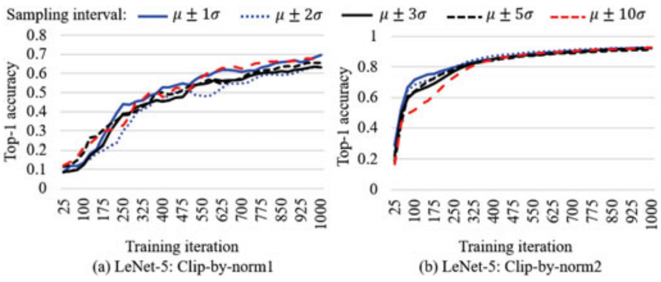

Fig. 12. Discussion of the effectiveness of PrivateDL under different sampling intervals.

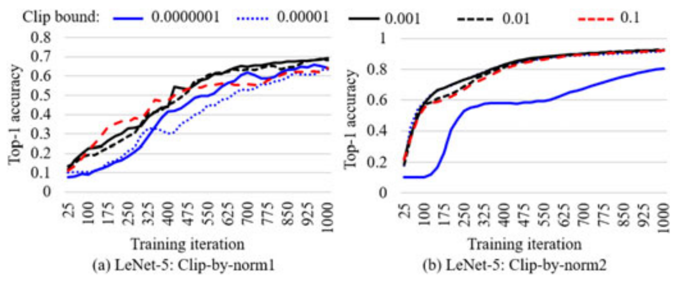

Fig. 13. Discussion of the integration of PrivateDL and clip techniques.

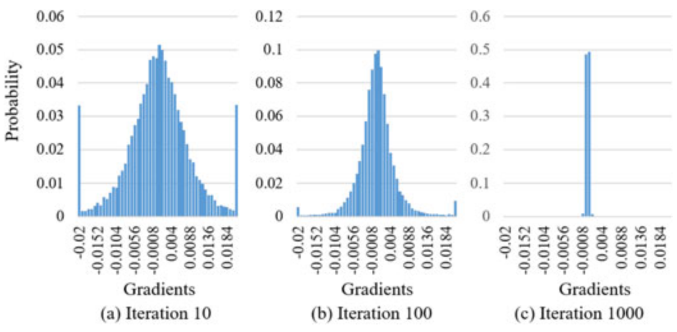

Fig. 14. Distributions of gradients across different iterations of model training in LeNet-5.

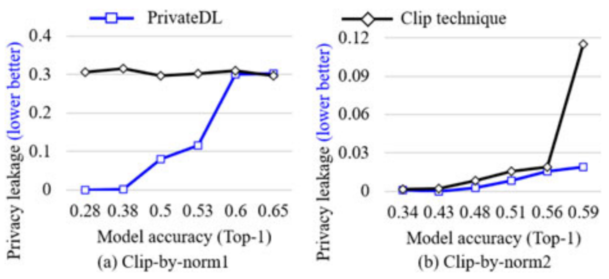

Fig. 15. Discussions of privacy leakages in inference attacks.

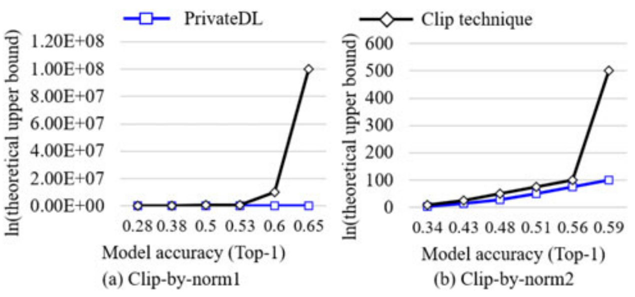

Fig. 16. Discussions of upper bounds of privacy degradation in inference attacks.

and these two bounds restrict the gradients to a much smaller range. In contrast, the largest bound (0.1) has no restrictions on the gradients. The two medium bounds (0.001 and 0.01) achieve the highest accuracy, because they set appropriate ranges for gradients while avoiding the possible outliers in gradients. We note that although setting proper bounds improves accuracy, it is difficult to know which bounds are appropriate in practical before training starts.

Discussion of Privacy Leakages in Inference Attacks. We compare prevacy preservation between PrivateDL and the clip techniques using a prevalent membership inference attack against DL models[1] [25]. In the attack, the attack model (a binary classifier) judges whether a data point belongs to the training sample used by the target model. *Privacy leakage* is used as the evaluation metric, which denotes the difference between the true positive rate (TPR) and the false positive rate (FPR) of the attack model [24]. In evaluation, we compare the privacy leakage of PrivateDL and the clip techniques, given that the target models trained using these techniques have the same model accuracies. Fig. 15 shows that PrivateDL always achieves less privacy leakages compared to the clip techniques, because it maintains the same model accuracies using smaller privacy budgets (i.e., higher privacy levels). In addition, the privacy leakage in the L_1-norm is much larger than that of L_2-norm, because the former mechanism injects much larger noises given the same privacy budget and thus requires much larger privacy budgets for the same model accuracies. In average, our approach reduces privacy leakage by 23.33 times under the same model accuracies. Fig. 16 further compares the theoratical upper bounds of the two techniques. In evaluation, parameter a is set to 1 (one algorithm is used in model training) and the y asix shows the logarithm of upper bounds. We can see that under the same model accuracies, PrivateDL has tigher bounds of privacy degradation than the clip techniques and this result is consistent with the privacy leakages in Fig. 15.

1. https://github.com/spring-epfl/mia

Discussion of Applicability to Mobile Scenarios. We apply PrivateDL in training MobileNet, a compact DL model designed for mobile and enbedded vision systems [52], on the Jetson node. In this evaluation, we tested PrivateDL's effectiveness in reducing local noises using three privacy budgets and three batch sizes for both clip-by-norm1 and clip-by-norm2 techniques. The results in Fig. 17 show that our approach consistently obtains higher model accuracies than the clip techniques under different evaluaion settings and achieves more accuracy improvement when the privacy budeget becomes smaller. Overall, PriviteDL improves model accuracy by 116.97 percent when using the same model training time.

6 Related Work

Within the context of federated learning, numerous efforts are contributed to enhance privacy during the learning process, including security multi-party computation (SMC) [53], homomorphic encryption [54], and differential privacy [10], [11] Specifically, SMC enables multiple participants collaboratively and securely computing a common function without accessing any raw dataset of each other. This techniques has been applied at both training stage [53] and inference stage [55] of DL. In addition, homomorphic encryption aims to make the result computed on encrypted data similar with the one based on raw data. For example, Cryptonets replaces the conventional activation function (e.g., ReLu) with the square function to implement this technique [54]. In contrast, differential privacy is another prevalent approach to prevent privacy leakage during the model training process by adding noises to data or gradients [10], [11]. We now review its application in two major types of infrastructures.

Cloud-Based Machine Learning/DL Applications. Techniques in this category can be divided into interactive and no-interactive ones. The interactive technique is usually integrated with

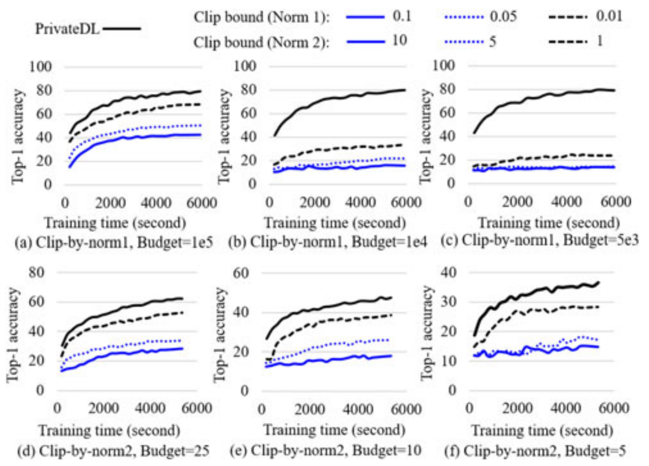

Fig. 17. Discussion of applying PrivateDL in a model environement.

specific algorithms such as recommendation systems [56] and stream processing systems [57], or specific training algorithms such as ID3 decision tree [58]. In contrast, the non-interactive technique can generate a noise-perturbed dataset for different training algorithms [10]. For example, a black box scheme is developed to integrate any SGD-based training algorithm into big data systems such as Hadoop or Spark, and the scheme only adds noises at the end of each iteration [59].

Edge-Based DL Applications. In this scenario, models are trained using data from multiple participants (edge nodes). In a federated learning setting, current differential privacy techniques add noises to gradients before reporting to a third curator [60]. Two techniques are developed for DL applications:

(1) *ϵ-differential privacy* [15]: this technique adds noise according to the Laplace mechanism, and often adopts the Manhattan distance (L_1-norm) to compute the sensitivity. A fixed interval is used in computation: the gradients whose values are outside the interval are replaced with the corresponding value of bound and the gradients falling into the interval are kept unchanged.

(2) *(ϵ, δ)-differential privacy* [2]: this technique clips each gradient in the L_2 norm using an interval bounded by C, then adds noises to the clipped gradient with the Gaussian distribution. At each iteration, it scales down the gradients, whose moduli are greater than the interval, to be of norm C.

Moreover, some approach is proposed to improve the noise injection mechanism of differential privacy [61]. Motivated by the factor that the default mechanism injects equal noises to all training samples, this approach injects more noises to samples that are less relevant to model accuracy, thus improving privacy level with less accuracy degradation [61]. Some other techniques combine differential privacy with homomorphic encryption or block chain. For example, ϵ-differential privacy is combined with an adaptive homomorphic encryption technique to prevent data privacy leakage in the central server [17]. Similarly, the block chain is combined to deal with the threat from both the malicious server and poisoning participants [18].

When applying to DL applications, existing differential privacy techniques rely on pre-specified bounds to clip gradients in noise calibration. Due to the heterogeneous and dynamic nature of gradients, these techniques suffer from setting oversize or undersize bounds, both of which may incur larger accuracy losses (due to larger noises) or worse training performance (due to lower convergency rate). In contrast, our approach needs no pre-defined bounds and thus can adapt to the changing gradients in various training scenarios.

Subsampling in Differential Privacy. Subsample techniques amplify privacy by selecting a subset from the entire input data using random/Poisson sampling with/without placement [62], [63], [64]. Our approach is orthogonal to these techniques such that it first samples more data points at an training iteration because large batch size mitigate noises, and then decreases the training cost by only using the critical points.

7 Conclusion

In this paper, we empirically and analytically show the new challenges of local and global statistical noise calibration for applying differentially privacy on distributed DL. We address the research question of how to improve the overall DL model accuracy while adhering to local privacy constraints and factoring the heterogeneity of training tasks and edge nodes. To such an end, we designed PrivateDL, a novel learning framework that can effectively calibrate local noise via sampling and minimise the impact of global noise by batch size amplification. We also implemented our approach in KubeEdge, and then extensively evaluated and demonstrated its practical effectiveness using Intel BigDL workloads, e.g., an accuracy improvement up to 5X compared to the state-of-the-art. Our future work includes implementing PrivateDL in Tensorflow [65], which provides convenient interfaces for our approach. In Tensorflow 2.2, the eager model supports injection of noises to gradients during model training; tf.GradientTap and tf.keras.optimizers provide interfaces to calculate gradients and update model parameters.

Acknowledgments

This work was supported in part by the National Key Research and Development Plan of China under Grant 2019QY1700.

References

[1] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 63–71.
[2] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Compu. Commun. Secur.*, 2016, pp. 308–318.
[3] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–14.
[4] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2019, *arXiv: 1712.07557*.
[5] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.
[6] L. Lyu *et al.*, "Towards fair and privacy-preserving federated deep models," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2524–2541, Nov. 2020.
[7] 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.
[8] Q. Jia, L. Guo, Z. Jin, and Y. Fang, "Preserving model privacy for machine learning in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 8, pp. 1808–1822, Aug. 2018.

[9] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv: 1912.04977*.
[10] C. Li, P. Zhou, L. Xiong, Q. Wang, and T. Wang, "Differentially private distributed online learning," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1440–1453, Aug. 2018.
[11] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1442–1452.
[12] C. Kamalika and A. D. Sarwate, "Differentially private machine learning: Theory, algorithms, and applications," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017. [Online]. Available: https://www.ece.rutgers.edu/ asarwate/nips2017/
[13] C. Dwork, "Differential privacy," in *Proc. Encyclopedia Cryptography Secur.*, 2011, pp. 338–340.
[14] F. D. McSherry , "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 19–30.
[15] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321.
[16] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *Proc. 28th USENIX Secur. Symp.*, 2019, pp. 1895–1912.
[17] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 5, pp. 1333–1345, May 2018.
[18] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2952332.
[19] R. Han *et al.*, "SlimML: Removing non-critical input data in large-scale iterative machine learning," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.>2019.2951388.
[20] "An open platform to enable edge computing," 2020. [Online]. Available: https://kubeedge.io/en/
[21] "Production-grade container orchestration," 2020. [Online]. Available: https://kubernetes.io
[22] "BigDL: Distributed deep learning on apache spark," 2020. [Online]. Available: https://github.com/intel-analytics/BigDL
[23] "Pytorch," 2020. [Online]. Available: https://pytorch.org/
[24] 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 Symp. Secur. Privacy*, 2019, pp. 739–753.
[25] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 3–18.
[26] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[27] L. Melis, C. Song, E. De Cristofaro , and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 691–706.
[28] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[29] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: The SuLQ framework," in *Proc. 24th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, 2005, pp. 128–138.
[30] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Secur. Foundations Symp.*, 2018, pp. 268–282.
[31] C. Dwork *et al.*, "The algorithmic foundations of differential privacy," *Foundations Trends® Theor. Comput. Sci.*, vol. 9, no. 3–4, pp. 211–407, 2014.
[32] 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 2017.
[33] "MNIST handwritten digit database," [Online]. Available: http://yann.lecun.com/exdb/mnist/
[34] M. Rosenblatt, "A central limit theorem and a strong mixing condition," *Proc. Nat. Acad. Sci. USA*, vol. 42, no. 1, 1956, Art. no. 43.
[35] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2525–2534.
[36] J. Huggins, T. Campbell, and T. Broderick, "Coresets for scalable Bayesian logistic regression," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 4080–4088.
[37] F. Pukelsheim, "The three sigma rule," *Amer. Statistician*, vol. 48, no. 2, pp. 88–91, 1994.
[38] R. Han *et al.*, "Workload-adaptive configuration tuning for hierarchical cloud schedulers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2879–2895, Dec. 2019.
[39] I. Fette and A. Melnikov, "The websocket protocol," RFC 6455, pp. 1–71, 2011.
[40] U. Hunkeler, H. L. Truong, and A. Stanford-Clark , "MQTT-sa publish/subscribe protocol for wireless sensor networks," in *Proc. 3rd Int. Conf. Commun. Syst. Softw. Middleware Workshops*, 2008, pp. 791–798.
[41] M. Zaharia *et al.*, "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.
[42] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation*, 2012, pp. 2–2.
[43] R. Han, C. H. Liu, S. Li, S. Wen, and X. Liu, "Accelerating deep learning systems via critical set identification and model compression," *IEEE Trans. Comput.*, vol. 69, no. 7, pp. 1059–1070, Jul. 2020.
[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[47] "CIFAR-10 database," 2009. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[48] "Downsampled imagenet datasets," 2020. [Online]. Available: http://image-net.org/download-images
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Logic Program.*, 2015, pp. 1–15.
[50] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 2737–2745.
[51] Q. Ho *et al.*, "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. Advances Neural Inf. Process. Syst.*, 2013, pp. 1223–1231.
[52] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv: 1704.04861*.
[53] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proc. 55th Annu. Des. Autom. Conf.*, 2018, Art. no. 2.
[54] R. Gilad-Bachrach , N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[55] J. Liu, M. Juuti, Y. Lu, and N. Asokan, "Oblivious neural network predictions via miniONN transformations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 619–631.
[56] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the netflix prize contenders," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 627–636.
[57] M. Beck, P. Bhatotia, R. Chen, C. Fetzer, T. Strufe *et al.*, "Privapprox: Privacy-preserving stream analytics," in *Proc. USENIX Conf. Usenix Annu. Tech. Conf.*, 2017, pp. 659–672.
[58] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 493–502.
[59] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1307–1322.
[60] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019, Art. no. 12.
[61] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive laplace mechanism: Differential privacy preservation in deep learning," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 385–394.
[62] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Proc. IEEE 55th Annu. Symp. Foundations Comput. Sci.*, 2014, pp. 464–473.

[63] Y. Wang, J. Lei, and S. Fienberg, "Learning with differential privacy: Stability, learnability and the sufficiency and necessity of ERM principle," *J. Mach. Learn. Res.*, vol. 17, no. 183, pp. 1–40, 2016.
[64] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *Proc. Conf. Neural Inf. Process. Syst.*, 2018, pp. 6277–6287.
[65] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, 2016, pp. 265–283.

Rui Han received the MSc degree with honor from Tsinghua University, China, in 2010, and the PhD degree from the Department of Computing, Imperial College London, U.K, in 2014. He is an associate professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include system optimization for cloud data center workloads (in particular highly parallel services, machine learning and deep learning applications). He has more than 40 publications in these areas, including papers at the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Knowledge and Data Engineering*, INFOCOM, and ICDCS.

Dong Li is working toward the master's degree from the School of Computer Science and Technology, Beijing Institute of Technology. His work focuses on security and privacy of computer systems.

Junyan Ouyang is working toward the graduate degree from the School of Computer Science and Technology, Beijing Institute of Technology. His work focuses on optimization of big data system for machine learning and deep learning workloads.

Chi Harold Liu (Senior Member, IEEE) receives the BEng degree from Tsinghua University, China, in 2006, and the PhD degree from Imperial College, U.K, in 2010. He is currently a full professor and vice dean with the School of Computer Science and Technology, Beijing Institute of Technology, China. He is also the director of IBM Mainframe Excellence Center (Beijing), director of IBM Big Data Technology Center. Before moving to academia, he joined IBM Research - China as a staff researcher and project manager, after working as a postdoctoral researcher at Deutsche Telekom Laboratories, Germany, and a visiting scholar at IBM T. J. Watson Research Center, USA. His current research interests include the big data analytics, mobile computing, and deep learning. He received the Distinguished Young Scholar Award in 2013, IBM First Plateau Invention Achievement Award in 2012, and was interviewed by EEWeb.com as the Featured Engineer in 2011. He has published more than 200 prestigious conference and journal papers and owned more than 14 EU/U.S./U.K./China patents, with Google H index 26. He currently serves as the Symposium Chair for IEEE ICC 2020 Next Generation Networking, an area editor for KSII Trans. on Internet and Information Systems and the book editor for six books published by Taylor & Francis Group, USA and China Machinery Press. He also has served as the general chair of IEEE SECON'13 workshop on IoT Networking and Control, IEEE WCNC'12 workshop on IoT Enabling Technologies, and ACM UbiComp'11 Workshop on Networking and Object Memories for IoT. He served as the consultant to Asian Development Bank, Bain & Company, and KPMG, USA, and the peer reviewer for Qatar National Research Foundation, and National Science Foundation, China. He is a fellow of IET, and a fellow of Royal Society of the Arts.

Guoren Wang (Senior Member, IEEE) received the BSc, MSc, and PhD degrees from the Department of Computer Science, Northeastern University, Shenyang, China, in 1988, 1991, and 1996, respectively. He is now a full professor and director of Institute of Data Science and Knowledge Engineering, Department of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He was an assistant president for Northeastern University, China. His research interests include XML data management, query processing and optimization, bioinformatics, high dimensional indexing, parallel database systems, and cloud data management. He has published more than 150 research papers in top conferences and journals like the *IEEE Transactions on Knowledge and Data Engineering*, ICDE, SIGMOD, VLDB, etc.

Dapeng Wu (Fellow, IEEE) received the PhD degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. He is currently a professor with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL. His research interests include networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. He is currently the editor-in-chief of the *IEEE Transactions on Network Science and Engineering*.