# Towards Efficient and Privacy-Preserving Federated Deep Learning

**Conference Paper** · May 2019

DOI: 10.1109/ICC.2019.8761267

Hongwei Li
University of Electronic Science and Technology of China
**84** PUBLICATIONS **2,714** CITATIONS

Guowen Xu
University of Electronic Science and Technology of China
**44** PUBLICATIONS **694** CITATIONS

# Towards Efficient and Privacy-preserving Federated Deep Learning

Meng Hao* †, Hongwei Li(Corresponding author)* ‡, Guowen Xu* †, Sen Liu* and Haomiao Yang*
* School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
† CETC Big Data Research Institute Co., Ltd., Guiyang 550022, China
‡ Science and Technology on Communication Security Laboratory, Chengdu 610041, China

***Abstract*—Deep learning has been applied in many areas, such as computer vision, natural language processing and emotion analysis. Differing from the traditional deep learning that collects users' data centrally, federated deep learning requires participants to train the networks on private datasets and share the training results, and hence has more gratifying efficiency and stronger security. However, it still presents some privacy issues since adversaries can deduce users' privacy from local outputs, such as gradients. While the problem of private federated deep learning has been an active research issue, the latest research findings are still inadequate in terms of security, accuracy and efficiency. In this paper, we propose an efficient and privacy-preserving federated deep learning protocol based on stochastic gradient descent method by integrating the additively homomorphic encryption with differential privacy. Specifically, users add noises to each local gradients before encrypting them to obtain the optical performance and security. Moreover, our scheme is secure to honest-but-curious server setting even if the cloud server colludes with multiple users. Besides, our scheme supports federated learning for large-scale users scenarios and extensive experiments demonstrate our scheme has high efficiency and high accuracy compared with non-private model.**

***Index Terms*—Federated Deep Learning, Stochastic Gradient Descent, Privacy-preserving, Differential Privacy, Additively Homomorphic Encryption.**

## I. Introduction

Deep neural network has been applied in various application scenarios such as computer vision [1], emotion analysis [2] and natural language processing [3] due to its superior accuracy of prediction for training enormous amount of data. However, the collection and analysis of large-scale data arise potential privacy threats [4] [5]. Consider a situation in real-world, medical institutions often need datasets such as the physiological characteristics of cancer patients for deep learning to obtain the best therapeutic effect [6] [7]. These sensitive data are always leaked intentionally or unintentionally if the users directly submit them to the relevant institutions without any pre-process [8] [9].

In order to solve above challenge, federated deep learning [10] [11]has emerged and drawn widespread attention, where all users just need to upload to the cloud server the local gradients obtained from private training database, and then receive the global parameters from the server. Although it guarantees the privacy of individual data, the updated gradients can still reveal individual information [12] [13] [14]. For tackling this challenge, many scholars have made various of research and proposed some meaningful solutions. Shokri and Shmatikov [15] propose a collaborative deep learning model based on selective stochastic gradient descent, where each user selectively shares part of the local gradients. However, Phong et al. [16] reveal that the above model is not secure under the honest-but-curious server setting, even though the shared gradients have been perturbed utilizing differential privacy. In addition, Phong et al. design a deep learning system via additively homomorphic encryption and the system possesses high confidentiality and high accuracy. However, paillier homomorphic encryption is utilized to bring a large communication and computational overhead. Besides, this protocol can not defend the collusion between the cloud server and multiple users. Bonawitz et al. [17] proposed the first privacy-preserving federated deep learning approach which is robust to users dropping out by utilizing secure multiparty computation(MPC). Unfortunately, the quadratic communication overhead and simultaneous online for users greatly limit its widespread application to large-scale scenarios. Therefore, how to design a scheme to meet the above challenges is still a problem to be solved.

In this paper, we propose an efficient and privacy-preserving federated deep learning scheme based on stochastic gradient descent method by integrating the additively homomorphic encryption with differential privacy. Our contributions are summarized as follows:

- We present an efficient and secure gradients aggregation scheme in federated deep learning with lightweight homomorphic encryption. In this way, our scheme supports federated learning for large-scale applications.
- To prevent the threats from collusion between the cloud server and multiple users, we further use the differential privacy technique based on Laplace mechanism to perturb the user's original local gradients.
- Our proposed protocol can tolerate arbitrary subset of users dropping out under training process with negligible accuracy loss. Besides, extensive experiments demonstrate our scheme has high efficiency and high accuracy compared with non-private model.

The remainder of this paper is organized as follows. In Section II, we outline the system model and threat model. In Section III, we introduce the neural network and discuss

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

the cryptographic primitives. Then we present our scheme in details in Section IV and carry out the security analysis in Section V, respectively. Performance evaluation is depicted in Section VI. Finally, Section VII concludes the paper.

## II. SYSTEM MODEL AND THREAT MODEL

### A. System Model

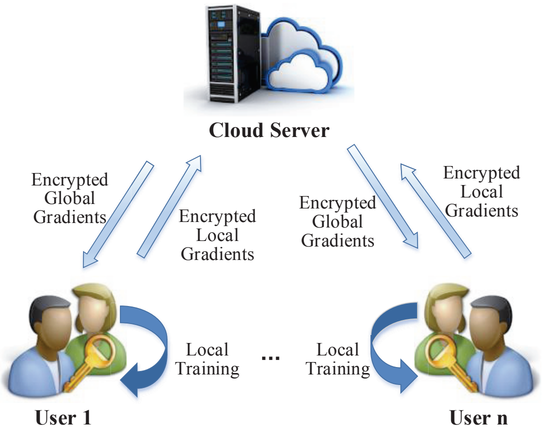

Cloud Server
Encrypted Global Gradients
Encrypted Local Gradients
Encrypted Global Gradients
Encrypted Local Gradients
Local Training
...
Local Training
User 1
User n

Fig. 1: System model

As shown in Fig.1, the system is comprised of two main entities: multiple users and the cloud server. We assume that all users agree on a common model of neural network and common objectives in advance.

- On the users side, they need to upload to the server the local gradients obtained from private training datasets, and then receive the global gradients from the cloud server. For the privacy purpose, each local gradients will be perturbed and encrypted before submitted to the cloud.
- The primary task of the cloud server is to compute the global gradients over encrypted local gradients with its powerful computing power. Afterwards, the global gradients are broadcasted to all users, where our model is robust to a certain amount of users dropping out in the training process. Finally, the entire neural network will be established with the iterative cooperation between the cloud server and multiple users.

### B. Threat Model

The objective of our scheme is to protect individual information from the users in whole training phase, where the cloud server is deemed to be honest-but-curious [18] [19]. More concretely, the cloud server honestly executes data aggregation operation as designed, yet it is also curious to infer sensitive information from users' inputs [20].

On the other hand, our proposed protocol can tolerate the collusion between the cloud server and multiple users, so we demand that the cloud server could not get any useful information from local gradients except the encrypted aggregated result.

## III. PRELIMINARIES

In this section, we introduce the neural network and briefly discuss the cryptographic primitives, which play significant roles in the whole paper.

### A. Neural Network and Federated Deep Learning

As shown in Fig. 2, a fully-connected neural network is comprised of multiple layers( including input layer, multiple hidden layers and output layer) as well as lots of adjustable parameters $\omega$ between neurons in adjacent layers. The neuron in each layer consists of affine transformations and applicable nonlinear activation function. Training a neural network includes two phases called *learning sample sets* and *updating parameters*. Firstly, proper activation function is selected to run neural network for specific learning objectives based on training dataset $D$. Afterwards, according to the above running results, each parameter of network will be updated to improve the learning accuracy. Generally, mini-batch stochastic gradient descent algorithm is commonly used to update the parameters $\omega$ by computing $\omega \leftarrow \omega - \eta \nabla L(D, \omega)$, where $\eta$ is a learning rate and $L(D, \omega)$ represents the loss function to measure the distance between predicted results and actual labels.

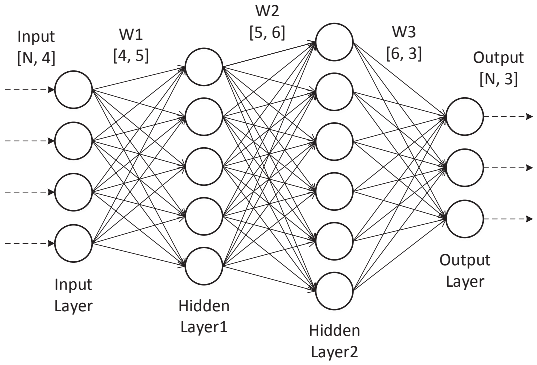


Fig. 2: Fully-connected neural network

In order to improve learning efficiency and provide stronger security for original user's data, federated deep learning was widely exploited. In its framework, multiple users commonly train the networks in a federated, collaborative and distributed manner. More precisely, for one epoch, each user $\mu \in \mathcal{U}$ possesses a private data subset $D_\mu \subseteq D$ and learns them to compute gradients $\nabla L(D_\mu, \omega)$. Subsequently, the cloud server collects local gradients from users and takes the average. Multiple users jointly update the parameters $\omega$ by computing $\omega \leftarrow \omega - \eta \frac{1}{|\mathcal{U}|} \sum_\mu \nabla L(D_\mu, \omega)$.

Note that for convenience of description, the $\mu$'s gradients $\nabla L(D_\mu, \omega)$ is denoted as $G_\mu$ in the following sections.

### B. Additively Homomorphic Encryption

The additively homomorphic encryption guarantees the additivity of multiple ciphertexts, at the same time, the encrypted aggregated result can be successfully decrypted. Therefore, users can outsource encrypted data to the cloud server for processing without revealing privacy.

For example, given two plaintexts $m_1, m_2$, we have

$$\begin{aligned} Enc_{sk}(m_1 + m_2) &= c_1 \oplus c_2 \\ Enc_{sk}(\alpha \cdot m_1) &= \alpha \otimes c_1 \end{aligned} \tag{1}$$

where $\alpha$ is a constant and $c_1$, $c_2$ indicate the ciphertext of $m_1$ and $m_2$ respectively.

### C. $\epsilon$- Differential Privacy and Laplace Mechanism

**Differential Privacy**. An algorithm $\mathcal{A}$ satisfies $\epsilon$-differential privacy($\epsilon$-DP) [21], if for any neighboring datasets $M$ and $\overline{M}$, and for any $T \subset Range(\mathcal{A})$, it holds that

$$Pr[\mathcal{A}(M) \in T] \leq e^{\epsilon} Pr[\mathcal{A}(\overline{M}) \in T] \tag{2}$$

where neighboring datasets represent that two datasets have only one different element and $\epsilon$ called privacy budget represents the current privacy level. In general, the smaller $\epsilon$ provides higher level of privacy protection but lower accuracy.

**Laplace Mechanism** [22]. For a query function $f$, mechanism $f^{'}$ satisfies $\epsilon$-DP if the following formula holds

$$f^{'}(M) = f(M) + Lap(\frac{\Delta f}{\epsilon}) \tag{3}$$

where $Lap(\frac{\Delta f}{\epsilon})$ is sampled from Laplace distribution that satisfies $Pr\left[Lap(\frac{\Delta f}{\epsilon}) = x\right] = \frac{\epsilon}{2\Delta f}e^{\frac{-|x|\epsilon}{\Delta f}}$. $\Delta f$ represents global sensitivity which is defined as the maximum of $\left|\left|f(M) - f(\overline{M})\right|\right|_1$ for neighboring datasets $M$ and $\overline{M}$. In our protocol, function $f$ computes the gradients during one epoch for each user.

## IV. PROPOSED SCHEME

In this section, we propose an efficient and privacy-preserving federated deep learning scheme based on stochastic gradient descent method by integrating the additively homomorphic encryption with differential privacy.

### A. Overview of Gradients Aggregation Scheme

Homomorphic encryption is widely utilized to achieve secure data aggregation in the form of ciphertext. Unfortunately, performing public-key encryption on original data dramatically increases computational and communication overhead [23]. To combat that challenge, we exploit the symmetric additively homomorphic encryption called PPDM [24] due to its excellent efficiency. Moreover, to further improve the security and tolerate users' dropouts, differential privacy technique is utilized for adding calibrated noises to each local gradients before encryption.

As shown in the Fig. 3, for one epoch, each user $\mu$ learns the model with a small batch of local datasets and computes the local gradients $G_\mu$. To protect the gradients' privacy, the user utilizes Laplace mechanism to blind the local gradients and encrypts the blinded gradients with $C_\mu = Enc_{sk}(G_\mu + Lap(\frac{\Delta f}{\epsilon}))$. After receiving all encrypted gradients, the cloud server executes aggregation operation by

$$C_{add} = C_1 \oplus C_2 \oplus \cdots \oplus C_n = Enc_{sk}(\sum_{\mu=1}^{n} G_\mu) \tag{4}$$

where the noises are almost eliminated due to the symmetry of the Laplace distribution.

Ultimately, each user decrypts the encrypted global gradients $C_{add}$ returned from the server by $G_{add} = Dec_{sk}(C_{add}) = \sum_{\mu=1}^{n} G_\mu$ and then updates the parameters $\omega$ with $G_{add}$. In this way, the entire neural network will be established in constant iterative rounds between the cloud server and multiple users.

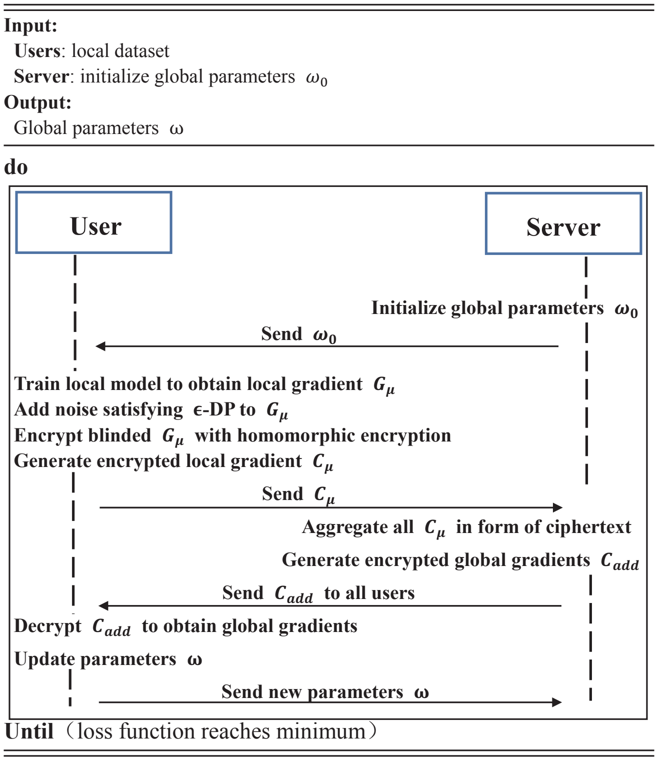

Input:
Users: local dataset
Server: initialize global parameters $\omega_0$
Output:
Global parameters ω
do
User
Server
Initialize global parameters $\omega_0$
Send $\omega_0$
Train local model to obtain local gradient $G_\mu$
Add noise satisfying ϵ-DP to $G_\mu$
Encrypt blinded $G_\mu$ with homomorphic encryption
Generate encrypted local gradient $C_\mu$
Send $C_\mu$
Aggregate all $C_\mu$ in form of ciphertext
Generate encrypted global gradients $C_{add}$
Send $C_{add}$ to all users
Decrypt $C_{add}$ to obtain global gradients
Update parameters ω
Send new parameters ω
Until（loss function reaches minimum）

Fig. 3: High-level view of our scheme

### B. Efficient and Privacy-preserving Federated Deep Learning

**Initialization**. The global training model is copied to each user's device. Meanwhile, the original global parameters $\omega_0$ and the learning rate $\eta$ are initialized by the cloud server. Besides, given the security parameter $\lambda$, the secret key $sk$ is generated and assigned to all users where $sk$ is comprised of two big prime numbers $p, q(|p| = |q| = \lambda)$. The public parameter is $N = pq$.

**Encryption**. To preserve differential privacy of user's information, all users jointly select a same privacy budget $\epsilon$. Here we only consider the user $\mu$ during one epoch. After $\mu$ derives initial parameters $\omega_0$ from server, local training is executed with individual dataset to obtain the local gradients $G_\mu$. For privacy reason, $\mu$ randomly selects noises from the

Laplace distribution $Lap(\frac{\Delta f}{\epsilon})$ to perturb the local gradients:

$$\begin{aligned} G_{\mu,p} &\equiv (G_\mu + Lap(\frac{\Delta f}{\epsilon})) \bmod p \\ G_{\mu,q} &\equiv (G_\mu + Lap(\frac{\Delta f}{\epsilon})) \bmod q. \end{aligned} \tag{5}$$

Differential privacy depends on privacy budget $\epsilon$ as well as sensitivity $\Delta f$. In our protocol, we assume each gradient $0 \leq G \leq 1$ by utilizing Min-Max Normalization, and $\Delta f$ can be set to 1.

Afterwards, the local gradients are encrypted using secret key $p, q$ as follows:

$$C_\mu = q^{-1} q G^p_{\mu,p} + p^{-1} p G^q_{\mu,q} \bmod N \tag{6}$$

where $p^{-1}$, $q^{-1}$ denote the inverses of $p$ and $q$ in $\mathbb{Z}^*_q$ and $\mathbb{Z}^*_p$ respectively and the encrypted local gradients $C_\mu$ will be sent to the cloud server. Moreover, adding noises to the local gradients prevents the threats from collusion between the cloud server and multiple users.

**Secure Aggregation**. Considering the powerful computing ability of the cloud server, after receiving all users' encrypted gradients, the goal of the cloud server is to execute the operation of addition aggregation as below

$$\begin{aligned} C_{add} &= \sum_{\mu=1}^{n} C_\mu \\ &= q^{-1} q \sum_{\mu=1}^{n} G^p_{\mu,p} + p^{-1} p \sum_{\mu=1}^{n} G^q_{\mu,q} \bmod N \\ &= q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^p + p^{-1} p (\sum_{\mu=1}^{n} G_{\mu,q})^q \bmod N. \end{aligned} \tag{7}$$

The above equality is established because

$$\begin{aligned} & q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^p \bmod N \\ &= q^{-1} q ((\sum_{\mu=1}^{n} G^p_{\mu,p}) + p f(G)) \bmod N \\ &= q^{-1} q (\sum_{\mu=1}^{n} G^p_{\mu,p}) \bmod N \end{aligned} \tag{8}$$

where $f(G)$ represents $f(G_{1,p}, \cdots, G_{n,p})$, namely a polynomial composed of the variables $G_{i,p} (i = 1, 2, \cdots, n)$. Finally, the server communicates with all the users to broadcast the encrypted global gradients $C_{add}$.

**Decryption**. After each user receives the encrypted global gradients, the operation of decryption is as follows:

$$\begin{aligned} C_{add} \bmod p &= q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^p + p^{-1} p (\sum_{\mu=1}^{n} G_{\mu,q})^q \bmod p \\ &= q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^p \bmod p \\ &= q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^{p-1} (\sum_{\mu=1}^{n} G_{\mu,p}) \bmod p \\ &= \sum_{\mu=1}^{n} G_{\mu,p} \bmod p \\ &= G_{add,p} \bmod p. \end{aligned} \tag{9}$$

Similarly,

$$\begin{aligned} C_{add} \bmod q &= q^{-1} q (\sum_{\mu=1}^{n} G_{\mu,p})^p + p^{-1} p (\sum_{\mu=1}^{n} G_{\mu,q})^q \bmod q \\ &= \sum_{\mu=1}^{n} G_{\mu,q} \bmod q \\ &= G_{add,q} \bmod q. \end{aligned} \tag{10}$$

According to Euler Theorem and $gcd((\sum_{\mu=1}^{n} G_{\mu,p}), p) = 1$, the above formula holds. Then, the user can recover the global gradients $G_{add}$ by exploiting the Chinese Remainder Theorem as follows:

$$\begin{cases} G_{add} \equiv G_{add,p} \bmod p \\ G_{add} \equiv G_{add,q} \bmod q \end{cases} \tag{11}$$

Calculate this congruence expressions:

$$G_{add} = m_p q G_{add,p} + m_q p G_{add,q} \bmod N \tag{12}$$

where $m_p q \equiv 1 \bmod p$ and $m_q p \equiv 1 \bmod q$ . Computing $m_p, m_q$ is easily performed since the $gcd(p, q)$ equals 1. It is noted that our scheme tolerates arbitrary subset of users dropping out at any time since the number of surviving users is sufficient in real scenarios. Consequently it will almost have no effect on eliminating the noises brought in perturbation process, namely that $G_{add}$ is approximately equal to $\sum_{\mu=1}^{n} G_\mu$.

Ultimately, users update the parameters $\omega$ according to $\omega \leftarrow \omega - \frac{\eta}{N} G_{add}$, where $N$ comes from the cloud sever. Then the system is performed repeatedly until reaching the terminal condition, i.e., the minimum value of the loss function defined previously.

## V. SECURITY ANALYSIS

As shown in previous sections, our main concern is the privacy of individual information from user's local gradients against the cloud server as well as against the cloud server and compromised users. Other privacy issues are not the main issue.

### A. Security against the cloud server

*Theorem*: Our scheme reveals no bit information about individual local gradients, provided that the above additively homomorphic encryption scheme is CPA-secure.

*Proof*: our encryption scheme is based on the hardness of large integer decomposition problem and has been proved to be information-theoretic security in [24]. Consequently, the confidentiality of user's individual local gradients are protected well in our model.

### *B. Security against the cloud server and compromised users*

Considering a real-world situation, an adversary may compromise some users, thereby stealing the private key of the system and the privacy of honest users. To solve this problem, differential privacy is utilized to offer rigorous privacy guarantees.

*Theorem*: The Laplace mechanism $G_{\mu}^{'} = G_{\mu} + Lap(\frac{\Delta G}{\epsilon})$ preserves $\epsilon$-differential privacy of $G_{\mu}$ in gradients perturbation.

*Proof*: Let $\chi$ be the noise injected to $G_{\mu}$ and $\chi \sim Lap(\frac{\Delta G}{\epsilon})$. Let $M$ and $\overline{M}$ be neighboring datasets. We have

$$\begin{aligned} Pr[G_{\mu}^{'}(M) = t] &= Pr[G_{\mu}(M) + \chi = t] \\ &= Pr[\chi = t - G_{\mu}(M)] \\ &= \frac{\epsilon}{2\Delta G}\exp(\frac{-\epsilon|t - G_{\mu}(M)|}{\Delta G}) \end{aligned} \tag{13}$$

Similarly, $Pr[G_{\mu}^{'}(\overline{M}) = t] = \frac{\epsilon}{2\Delta G}\exp(\frac{-\epsilon|t-G_{\mu}(\overline{M})|}{\Delta G})$. Thus,

$$\begin{aligned} \frac{Pr[G_{\mu}^{'}(M) = t]}{Pr[G_{\mu}^{'}(\overline{M}) = t]} &= \frac{\exp(\frac{-\epsilon|t-G_{\mu}(M)|}{\Delta G})}{\exp(\frac{-\epsilon|t-G_{\mu}(\overline{M})|}{\Delta G})} \\ &= \exp(\frac{\epsilon(|t - G_{\mu}(\overline{M})| - |t - G_{\mu}(M)|)}{\Delta G}) \\ &\leq \exp(\frac{|G_{\mu}(\overline{M}) - G_{\mu}(M)|}{\Delta G}) \leq \exp(\epsilon) \end{aligned} \tag{14}$$

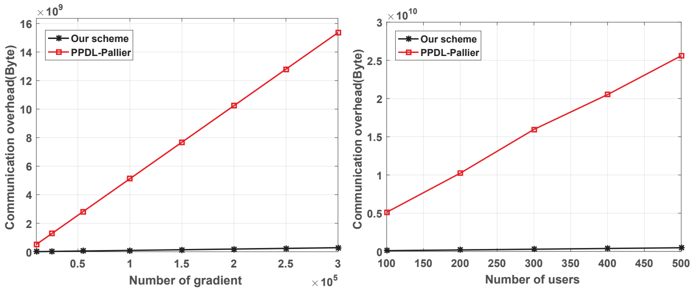
Therefore, gradients perturbation preserves $\epsilon$-DP and our scheme can tolerate the server colluding with any users but not inferring any useful information.

## VI. PERFORMANCE EVALUATION

In this section, we will evaluate our proposed scheme in terms of accuracy and both of the communication and computation overhead by comparing with latest scheme PPDL [16], where paillier encryption is adopted as the underlying structure. The number of users and the size of gradients are considered as the primary parameters in our experiments. Simulation is performed on a Ubuntu 18.04 system with an Intel(R) Xeon(R) CPU E5-2620 v4(2.10GHz) and 16GB of RAM.

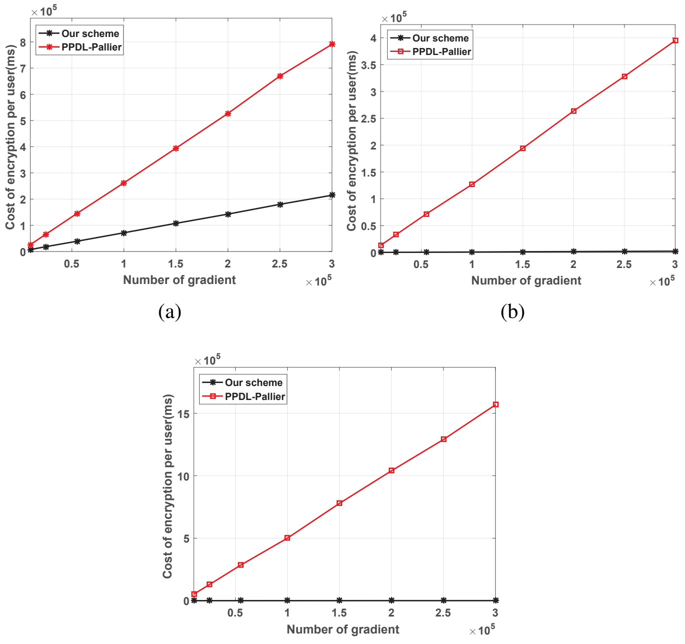
### *A. Communication overhead*

We run the convolutional network using Tensorflow on the MNIST dataset where the gradients come from sixty thousand 28×28 pixel image. Here we assume each user has only single thread, large security parameter $\lambda$ is 512 and the local gradients are presented by 32 bits. Fig. 4 shows the communication overhead of the cloud server under one iteration. Clearly, when the number of users is set to 100, as the number of gradients increases, the overhead of our scheme is about 50 times smaller than [16]. One of the major reasons is the rapid growth of ciphertext volume after utilizing pallier encryption. Similarly, the communication overhead is dramatically lower than [16] as the number of users increases.



(a) (b)

Fig. 4: Communication overhead of the cloud server. (a) For the different number of gradients. (b) For the different number of users.



(a) (b)

(c)

Fig. 5: Computational cost per user. (a) For the different number of gradients at the encryption phase. (b) For the different number of gradients at the aggregation phase. (c) For the different number of gradients at the decryption phase.

### *B. Computational cost*

Then, we discuss computational cost of our proposed scheme in encryption phase, aggregation phase and decryption phase, respectively. As shown in Fig. 5, the cost of encryption is dramatically lower than [16] as the number of gradients increases. Moreover, our cost of aggregation and decryption even seems to be approximately zero since they only contain few multiplication and addition operations. Thus, our proposed scheme can support large-scale user scenarios. For special objects, we can utilize multiple threads for calculations or