Mobile Networks and Applications (2020) 25:2421–2433
https://doi.org/10.1007/s11036-020-01586-4

FedMEC: Improving Efficiency of Differentially Private Federated Learning via Mobile Edge Computing

Jiale Zhang[1] · Yanchao Zhao[1] · Junyu Wang[1] · Bing Chen[1]

Published online: 18 June 2020

Abstract

Federated learning is a recently proposed paradigm that presents significant advantages in privacy-preserving machine learning services. It enables the deep learning applications on mobile devices, where a deep neural network (DNN) is trained in a decentralized manner among thousands of edge clients. However, directly apply the federated learning algorithm to the mobile edge computing environment will incur unacceptable computation costs in mobile edge devices. Moreover, among the training process, frequent model parameters exchanging between participants and the central server will increase the leakage possibility of the users' sensitive training data. Aiming at reducing the heavy computation cost of DNN training on edge devices while providing strong privacy guarantees, we propose a mobile edge computing enabled federated learning framework, called FedMEC, which integrating model partition technique and differential privacy simultaneously. In FedMEC, the most complex computations can be outsourced to the edge servers by splitting a DNN model into two parts. Furthermore, we apply the differentially private data perturbation method to prevent the privacy leakage from the local model parameters, in which the updates from an edge device to the edge server is perturbed by the Laplace noise. To validate the proposed FedMEC, we conduct a series of experiments on an image classification task under the settings of federated learning. The results demonstrate the effectiveness and practicality of our FedMEC scheme.

Keywords Federated learning · Mobile edge computing · Deep neural network · Model partition · Differential privacy

1 Introduction

With the explosive growth of the smart Internet of Things (IoT) devices, intelligent mobile networking applications have become ubiquitous, which triggers the high demands for the on-device big data analytics. Meanwhile, the cloud-based deep learning services [1], including recommendation systems, health monitoring, language translation, and many others [2–4], call for the efficiency improvement of performing the deep neural networks (DNN) on the mobile devices. However, such a centralized deep learning framework requires the users to outsource their sensitive data to the remote cloud in order to train the corresponding learning models, which arises the significant concerns on privacy as well as the on-device computation resources [5]. To this end, mobile edge computing (MEC) [6] is proposed as a novel distributed computation architecture to provide powerful and real-time data storage and processing capabilities at the edge of the network [7], which can be seen as a cornerstone to bridge the cloud-based learning services and mobile devices [8].

As a deep learning model in mobile edge computing, *Federated learning* (FL) [9, 10] has gradually arisen concerns from academia and industry by considering the data privacy issues in a decentralized learning manner, which can be widely used in numerous emerging scenarios, such as crowdsourced system [12]. The main purpose of

The material in this paper was presented partially at "An Efficient Federated Learning Scheme with Differential Privacy in Mobile Edge Computing", EAI MLICOM 2019 [24]

✉ Bing Chen
cb_china@nuaa.edu.cn

Jiale Zhang
jlzhang@nuaa.edu.cn

Yanchao Zhao
yczhao@nuaa.edu.cn

Junyu Wang
wangjunyu@nuaa.edu.cn

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China

federated learning is to build a joint machine learning model upon localized datasets while providing privacy guarantee [11]. Participants in federated learning act as the data providers to train a local learning model, meanwhile, the server maintains a global model by averaging all the local model parameters (i.e., gradients) generated by randomly selected participants until convergence [13]. The privacy is guaranteed due to the shares between server and participants are only model parameters, which prevent the server from direct access to the private training data.

Although federated learning has several advantages such as privacy-preserving and on-device learning, the scalability and privacy issues are still fatal for this novel paradigm, especially when encounter heavy local training consumes (DNN model) and the possible insider attackers. In this paper, we mainly consider the following two practical issues when applying the federated learning protocol to mobile edge computing architecture: 1) executing whole DNN training phase on the resource-constraint mobile edge devices will introduce incredible computation costs, which means the smart devices cannot afford such a heavy computation requirement in the federated learning protocol; 2) the shares, i.e., local model updates and global model parameters, among central server and participants could directly leak a proportion of private training datasets, meaning that the standard federated learning protocol cannot provide strong privacy guarantee against malicious entities, such as edge and cloud servers.

First of all, there are a very large number of parameters (sometimes hundreds of millions) in a common deep neural network [14, 15], even the forwarding operation of such a huge deep neural network requires significant processing and storage resources. However, the resource-constraint nature of mobile devices implies that the computation capability will soon reach the bottleneck and makes deep learning applications tend to invalidation [16, 17]. Furthermore, the shares (i.e., gradients) in federated learning could leak the sensitive information of users' training data to the untrusted third parties [18, 19]. For example, according to [20], the server in federated learning can easily launch the model inversion attack to obtain parts of training data distributions, and the gradient backward inference described in [21] also enables an adversary to get a fraction of private data from the participants' local updates.

For solving the efficiency problem of DNN computation, model partition technique [22] has been presented to offload the large parts of loosely coupled hidden layers [23] in a DNN model to the third party, which can be used to enable federated learning applications on mobile edge computing environment. On the other hand, aiming at preventing user-side information leakage to the untrusted server, several works have been done from different perspectives [24]. Abadi et al. [25] proposed a privacy-preserving deep learning method to protect user's data privacy by adding the Gaussian perturbation [26] to the clipped gradient. Geyer et al. introduced a user-side differential private federated learning mechanism [27] to protect the shared learning model from revealing each participant's updates. Furthermore, Bonawitz et al. proposed a sum aggregation protocol for high-dimensional data using the secret-sharing method [28]. However, these privacy solutions rely on the presence of a trusted aggregator to perturb the global model parameters and publish the noised parameters to each participant securely. That means the aggregator is able to see each individual's model parameters. Therefore, it is necessary to design a practical mechanism to protect the privacy of each participant against the untrusted third parties in federated learning.

In this paper, we propose an efficient private federated learning scheme in mobile edge computing, named FedMEC, by integrating the model partition technique to the standard federated learning mechanism. Basically, FedMEC can partition the underlying DNN model across the edge server and the mobile edge devices. On the client-side DNN, the users' private training data is fed into the low-level neural network to extract features. On the edge-side DNN, the local model updates are generated by executing the forward and backward propagations on these features. At last, the cloud server in our FedMEC framework aggregates all the received local model updates to improve the current global model until it tends to convergence. In order to preserve training data privacy, we further present a differentially private data perturbation mechanism on the local side, which adds the deliberate perturbations into the features before transmitting to the edge server. In other word, the clients, edge server and the cloud server run the different portion of a federated learning protocol, and the shares among these entities are perturbed by Laplace noise to achieve differential privacy. Our main contribution can be summarized as follows:

- **A framework enabling federated learning in mobile edge computing**: Instead of outsourcing user's training data to the cloud in the conventional cloud-centric machine learning systems, we apply federated learning to mobile edge computing to realize the localized training property. Meanwhile, we design a partitioned learning framework to split the deep neural network into two parts: client-side DNN and edge-side DNN, where the former one maintains a consistent convolutional layer to extracting training data features and the latter one equips the remaining layers to update each participant's parameters.
- **A differentially private data perturbation mechanism**: To protect user-side data privacy against untrusted third parties, we present a differentially

private data perturbation mechanism to perturb the Laplacian random noises to the local training data features extracted by the partitioned convolutional layer, so as to achieve differential privacy and provide rigorous privacy guarantee.
– **Exhaustive experimental evaluation**: We evaluate the proposed FedMEC scheme through a standard image classification task under the settings of federated learning and compare the result with several related methods to demonstrate effectiveness. Further, we also deploy FedMEC on an Android device to test the overhead.

The remainder of this paper is organized as follows. We briefly introduce the basic knowledge of federated learning and differential privacy in Section 2 and the system framework is presented in Section 3. Section 4 provides the detailed construction of the proposed FedMEC scheme and Section 5 discusses the effectiveness of FedMEC by conducting the extensive experimental evaluations. Finally, Section 6 gives the conclusion and future work. The notations used in this paper are listed in the Table 1.

2 Preliminaries

2.1 Federated learning

Federated learning has been explored to provide a more flexible distributed machine learning framework, whose main purpose is to perform joint machine learning model training across multiple mobile devices with private training data respectively. All the participants locally train the global model on their private training data and upload only the model parameters instead of the raw data. Such a localized model training method presents significant advantages in privacy-preserving because the clients do not need to share their private data with any third party. Figure 1 demonstrates the federated learning framework with model average.

Table 1 Notations used in this paper

Notation	Definition
$i \in n$	Total number of participants
x_r	Users' sensitive raw data
$\mathcal{F}_{(x_r)}$	Deep neural network
$\mathcal{L}_{(w_t, x_r)}$	Objective loss function
σ	Noisy scale
$\Delta\mathcal{F}(\Delta f)$	Global sensitive
B	Norm bounding
l	Partitioned layer
x_l	l-th layer output of DNN
I_n	Nullification matrix
x'_r	Nullification result for x_r
x'_l	Nullification result x_l
E	Local epoch
$\mathcal{B}$	Local mini-batch size
η	Local learning rate
$t \in T$	Current communication round
$g_t^{(i)}$	Gradient for user i at t
$\Delta w_t^{(i)}$	Local model update at t
$w_t^{(global)}$	Global model parameters at t

The standard federated learning protocol [13] assumes that all the clients agree on a common learning objective and global model structure. In a certain communication round t, all the sampled participants m_t first download the global model parameters from the central server to update the local model, then the local model is trained locally to generate the local model update $\Delta w_t^{(i)}$ on the private training datasets. After that, the central model collects all the local model updates and aggregates them to improve the current global model until it tends to convergence. The model average step in the central server can be formulated as follow.

$$w_{t+1}^{(global)} = w_t^{(global)} + \frac{1}{m_t}\sum_{i=1}^{m_t} \Delta w_{t+1}^{(i)}, \tag{1}$$

where $w_t^{(global)}$ indicates the global model parameters at the t-th communication round, and $\Delta w_t^{(i)}$ denotes the local model update from the i-th participant at communication round $t+1$.

2.2 Differential privacy

Differential privacy [29] is a rigorous concept that has been widely used in data publication systems by adding randomly noise (e.g., Laplace or Gaussian noises) into datasets, concealing the real results of any data query operation. It is defined in terms of the data query on two adjacent databases $\mathcal{D}$ and $\mathcal{D}'$ where the query results are statistically similar but differing in one data item. In the context of federated learning, differential privacy could be used as the local-side privacy solution to protect the privacy of users' training datasets. The formal definition of ϵ-differential privacy can be described as follow:

Definition 1 (ϵ-differential privacy:) A randomized mechanism $\mathcal{M}: \mathcal{D} \rightarrow \mathcal{R}$ fulfills ϵ-differential privacy for certain non-negative number ϵ, iff for any adjacent input $d \in \mathcal{D}$ and $d' \in \mathcal{D}'$, and any output $S \subseteq \mathcal{R}$, it holds that

$$Pr[\mathcal{M}(d \in \mathcal{D}) \in S] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(d' \in \mathcal{D}') \in S], \tag{2}$$

where ϵ is defined as the privacy budget, which measures the level of privacy guarantee of the randomized mechanism $\mathcal{M}$: the smaller ϵ, the stronger privacy guarantee.

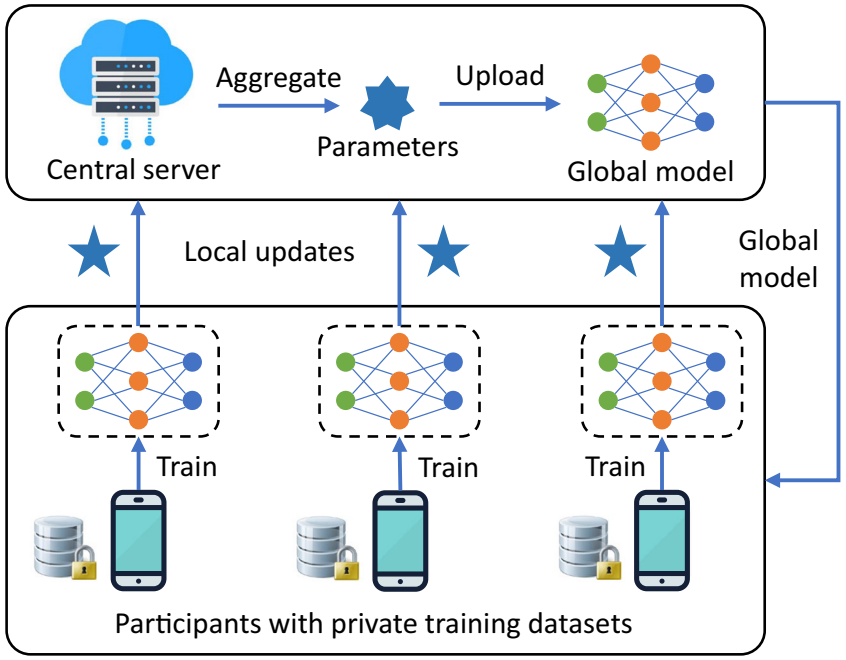


Fig. 1 Federated learning framework

Typically, a deterministic query $f : \mathcal{D} \rightarrow \mathcal{R}$ can simply achieve ϵ-differential privacy approximately by adding the deliberated perturbation to its result. The added perturbation is calibrated to f's *global sensitivity* Δf, which is defined as the maximal value of $||f(d) - f(d')||$ on the adjacent inputs d and d'. In this paper, we use the *Laplace mechanism* [26] to perturb the output data on the client side as follows:

$$\mathcal{M}(d) = f(d) + Lap(\lambda), \lambda = \frac{\Delta f}{\epsilon} \tag{3}$$

where $Lap(\lambda)$ is a random variable sampled from the Laplace distribution, and the scale parameter λ is set to $\frac{\Delta f}{\epsilon}$.

3 System framework

3.1 Federated learning with MEC

In this paper, we consider a MEC framework that can partition the federated learning protocol across the cloud central server, the edge servers, and the edge devices. Assuming all the users voluntarily participate in the federated learning protocol provided by a cloud central server, so as to obtain the desired machine learning services. Meanwhile, these participants try to prevent the training data privacy from being leaked by the malicious entities involved in the whole learning procedure, such as the untrusted cloud central and edge servers. Therefore, we present a three-layer mobile edge computing framework that provides the perfect architecture supportive of federated learning protocol with multiple participants. Specifically, the overall framework is shown in Fig. 2 and the involved entities are as follows.

- **Edge devices**: represent a set of devices (such as smartphones, laptops, smart meters, and so on) owned by the participants. Each edge device is equipped with computation and communication modules, which enable to execute the local training procedure and transfer its local update to the edge server.
- **Edge servers**: are core entities for mobile edge computing architecture. They own more storage and computation resources compared to the edge devices, which usually be deployed at the edge of the network and serves as the computation unit between cloud central and edge devices.
- **Cloud server**: acts as a control center to collect all users' local model updates and execute the federated average algorithm to update the global shared model. After the model average, it assigns the shared model to the edge device, who participates the federated learning protocol.

Among these entities, the edge devices were assumed as the trusted entities, whose goal is to benefit from the intelligent learning services by collaboratively executing the training phases with other participants. However, the third parties, e.g., edge server and cloud server, are honest-but-curious [30]. Specifically, they faithfully execute the federated learning process to compute correctly and send results truthfully. However, they are curious about the privacy contained in the data and attempt to disclose private data [31]. Except with the privacy issues, directly

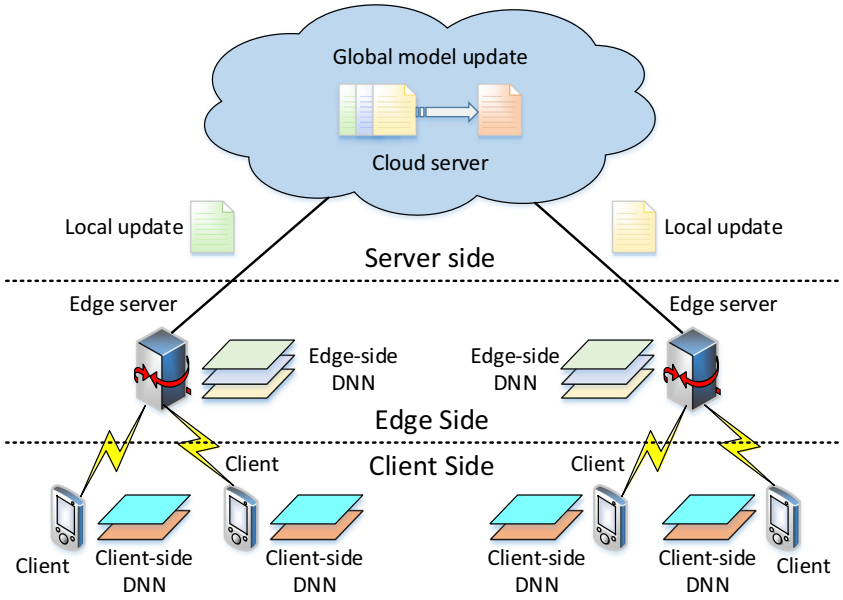


Fig. 2 Federated learning with mobile edge computing

implementing the federated learning protocol in mobile edge computing framework could also face practical concern. That is, the resource-constraint edge devices cannot easily afford the whole local model updating consumes because the heavy computation cost is needed to execute the DNN training procedure. Thus, the main challenge of applying federated learning with mobile edge computing is how to design a valid scheme to reduce the computation overhead on edge devices without broke the federated learning mechanism, while protecting user-side data privacy contained in the original data.

3.2 Overview of FedMEC

To solve the aforementioned challenges, we propose an efficiently private federated learning framework in mobile edge computing, named FedMEC, by considering both performance and privacy issues, where the local DNN model is divided into the client-side part and the edge - side part, so as to reduce the computation cost on the edge devices. The effectiveness of the partition mechanism in DNN architecture lies in the loosely coupled property among multiple insider layers. That is, each hidden layer in DNN can be executed separately by taking the previous layer's output as its input. According to this practical property of DNN, we can simply divide the DNN structure in federated learning across the edge devices and the edge server. In this paper, we consider partitioning the neural network along with the last layer of convolutional layers and all the intermediate results generated by the user-side DNN are hidden from the other entities (i.e., edge server and cloud server).

Figure 3 gives a high-level description of our proposed FedMEC framework. FedMEC relies on the mobile edge computing architecture and splits the whole federated learning protocol across the involved entities. Particularly, the local neural network training phase is divided into two parts: client-side DNN and edge-side DNN. According to the partition mechanism, the convolutional layers of the local DNN are deployed on the client-side while the remaining part (i.e., dense layers or fully-connected layers) will be assigned to the edge server. In this situation, edge

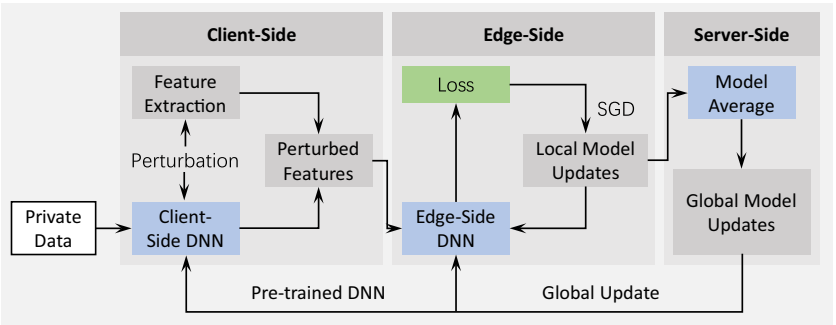


Fig. 3 Overview of proposed FedMEC framework

devices merely undertake the simple and lightweight feature extraction and perturbation. Similar to the settings in [14], the network structure and parameters of the client-side DNN model are frozen and the edge-side DNN can be fine-turned. Besides, the cloud central server in our FedMEC scheme is designed to execute the aggregate and average steps according to the standard federated learning protocol.

Here, the frozen local-side neural network is pre-trained on an auxiliary dataset which shares the same distribution with the localized private training datasets. Accordingly, the global model in the cloud server is initialed by a replica of this well-trained local model. Then the pre-trained global neural network will be partitioned along with the last layer of the convolution layer and the well-trained convolution layer will send to each client for feature extraction. Based on our designed FedMEC framework, the major parts of a whole DNN training procedure can be offloaded to the edge and cloud servers, while mobile edge devices are only needing to execute the simple feature extraction part through a frozen network model. Furthermore, to provide a rigorous privacy guarantee, we apply the differential privacy mechanism in our FedMEC scheme, where the extracted features are perturbed by the deliberated Laplace noise before being transmitted to the edge server. Note that, we do not consider the privacy contained in the label of the contributed training data since we assume that the participant willingly contributes the labeled data to perform supervised learning and should have no expectation on the privacy contained in the labels.

4 Construction of proposed FedMEC

4.1 Deep neural network partition

In this paper, we choose the most popular learning method, i.e., DNN, as the learning framework to construction our federated learning protocol in mobile edge computing scenario. A complete deep neural network structure consists of many loosely coupled hidden layers, which can be partitioned as multiple parts to embed in our mobile edge computing entities, such as the edge devices and the edge server. Thus, partitioning large DNN across mobile devices and edge server is one of the solutions to reduce complex computations on resource-constraint edge devices. Following this property, we can design an efficient deep learning framework which enables federated learning applications on mobile edge computing environments. Here, we apply DNN model with convolutional layers, i.e., convolutional neural network, as our baseline model architecture.

Figure 4 shows a high-level description of our designed DNN partition method. As we know, the convolutional processes in a DNN training procedure take the most complex computation overheads, which will consume plenty of resources of mobile devices. To solve this shortcoming, we split a complete DNN model into two parts: client-side DNN and edge-side DNN, along with the last layer of the convolutional network. In the client-side part, the front portions of a DNN structure (i.e., convolution layers) are deployed to extract features from the raw data. To protect the privacy of the sensitive training data, we add the deliberated perturbations to the outputs of the client-side DNN, so as to guarantee the differential privacy. The edge-side DNN model consists of the remaining portions of the DNN structure (i.e., dense layers) to update the model parameters by executing the forward and backward propagation procedures. Based on our designed DNN partition method, the client-side resource-hungry operations required in the standard federated learning protocol can be significantly reduced [32]. Especially for the computation resources and energy considerations, partitioning solution is attractive to many machine learning service providers, paving the way for federated learning applications on mobile edge devices.

4.2 Differentially private data perturbation

Federated learning protocol is designed for providing basic privacy guarantee for each participants' raw data due

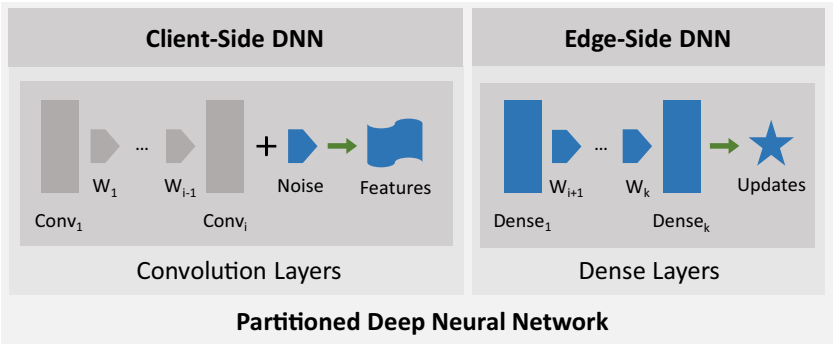


Fig. 4 Partition process on the deep neural network

to its local training property. However, a participant's sensitive data is still possibly leaked to the untrusted third parties, such as edge server and cloud server, even with a small portion of updated parameters (i.e., features and gradients). For examples, according to [20], the server in federated learning can easily launch the model inversion attack to obtain parts of training data distributions, and the gradient backward inference described in [21] also enables an adversary to get a fraction of private data from the participants' local updates. Therefore, it is necessary to design a practical preserving mechanism to protect the privacy of each participant against the untrusted third parties in federated learning.

Differential privacy [26] is a great solution to provide the rigorous privacy guarantee by adding deliberate perturb on the sensitive datasets. However, adding the perturb to the original data directly may lead to significant negative effects about learning performance [33]. Thus, we can perturb the features generated by the convolutional layers of partitioned DNN, so as to preserve the privacy contained in the raw data. In this paper, we solve the aforementioned problem by considering a differentially private data perturbation mechanism which can protect the privacy information contained in the extracted features after executing the client-side DNN.

Following by the work from [14], we consider the deep neural network as a deterministic function $x_l = \mathcal{F}(x_r)$, where x_r represents the private raw data and x_l stands for the l-th layer output of a neural network [34]. For the privacy concern, we applying the differential privacy method to the DNN and further construct our private federated learning protocol in mobile edge computing paradigm. One efficient way to realize the ϵ-differential privacy is to add controlled Laplace noise which is sampled from the Laplace distribution with scale $\Delta\mathcal{F}/\epsilon$ into the output x_l. According to the definition of differential privacy described in Section 2.2, the global sensitivity for a query $f : \mathcal{D} \rightarrow \mathcal{R}$ can be defined as follow:

$$\Delta f = \max_{d \in D, d' \in D'} ||f(d) - f(d')|| \tag{4}$$

However, the biggest challenge here is that the global sensitivity $\Delta\mathcal{F}$ is difficult to quantification in the deep neural network. Directly adding the Laplace perturbations into the output features will destroy the utility of the representations for the future predictions.

To address this problem, we employ the nullification and norm bounding methods to enhance the availability of differential privacy in deep neural networks. Specifically, before a participant starting to extract the features from his sensitive raw data x_r using the pretrained client-side DNN, he firstly performs the nullification operation to mask the high sensitive data items as $x_r' = x_r \odot I_n$, where $\odot$ is the multiplication operation and I_n is the nullification matrix with the same dimension as input sensitive raw data. Besides, the nullification matrix I_n is a random binary matrix (i.e., consisted of 0 and 1) and its structure is determined by a nullification rate μ, meaning that the number of zeros is the supremum of $Sup(n \cdot \mu)$. Apparently, μ has a significant impact on the prediction accuracy which will be discussed in Section 5.

After the nullification operation on the sensitive raw data, each participant needs to run the client-side DNN on x_r' to extract the features as $x_l = \mathcal{F}(x_r')$. Then, we consider the norm bounding method to enforce a certain global sensitivity as follow:

$$x_l' = x_l / \max\left(1, \frac{||x_l||_\infty}{B}\right) \tag{5}$$

where $||x_l||_\infty$ represents the infinite norm of the l-th layer outputs. This formula indicates that x_l' is upper bounded by S, meaning that the sensitivity of x_l can be preserved as long as $||x_l||_\infty \leq B$, whereas it will be scaled by B when $||x_l||_\infty > B$. According to [25], the scaling factor B usually be set as the median of $||x_l||_\infty$. The Laplace perturbation (scaled to B) now is added into the bounded features x_l' to further preserve the privacy as follow:

$$\tilde{x}_l = x_l' + Lap(B/\sigma I) \tag{6}$$

Note that the Laplace noise is added into the final output of the convolutional layers and the parameter $b \in B$ also has high impacts of the model performance. Due to the same network structure for each client-side DNN, we use the same notation $\tilde{x}_l$ to represent the latest perturbed features for all participants.

4.3 FedMEC algorithm

As aforementioned, federated learning allows the clients locally train their model in a distributed manner, and upload its local model update (i.e., gradient) instead of sharing their private data samples to the central server. In our differential private federated learning system, assuming their existing N sampled users agree on a common learning objective and model structure, each edge client owns its private dataset. In each iteration, edge clients download the partitioned deep neural network model to the local and fed their local private dataset into the client-side DNN model to generate the features. Then, they adding the deliberated Laplace noise to the features. After that, all the clients send the perturbed features to the edge server and the edge server will train the edge-side DNN model using those features to further generate the local model updates with the method of stochastic gradient descent (SGD) [10]. At last, those local model updates are aggregated and averaged to jointly optimize the current global model in the cloud server side.

The above steps will be iteratively executed until the global model tends to convergence. The pseudo-code of our overall FedMEC scheme was shown in Algorithm 1.

Algorithm 1 *FedMEC Algorithm.*

procedure CLIENT SIDE
  Nullification operation: $x_r' = x_r \odot I_n$
  Features extraction: $x_l = \mathcal{F}(x_r')$
  Norm bounding: $x_l' = x_l / \max(1, \frac{||x_l||_\infty}{B})$
  Adding perturbation: $\tilde{x}_l = x_l' + Lap(B/\sigma I)$
  Upload $\tilde{x}_i \leftarrow \tilde{x}_l$ to the edge server.
end procedure
procedure EDGE SERVER SIDE
EVENT: Receive perturbed features $\tilde{x}_i$
  Initialize: $w_t^{(i)} \leftarrow w_t^{(global)}$
  for each $E_k \in (1, \cdots, E)$ **do**
    for batch $b \in \mathcal{B}$ **do**
      $g_t^{(i)} = \nabla_{w_t} \mathcal{L}(w_t, \tilde{x}_i)$
      $w_{t+1}^{(i)} = w_t^{(i)} - \eta \cdot g_t^{(i)}$
    end for
    $\Delta w_{t+1}^{(i)} = w_{t+1}^{(i)} - w_t^{(i)}$
  end for
  Upload $\Delta w_{t+1}^{(i)}$ to the cloud server.
end procedure
procedure CLOUD SERVER SIDE
EVENT: Receive local model updates $\Delta w_{t+1}^{(i)}$
  Initialize global parameters $w_t^{(global)}$
  for all $i \in [1, n]$ **do**
    $w_{t+1}^{(global)} = w_t^{(global)} + \frac{1}{n} \sum_{i=1}^{n} \Delta w_{t+1}^{(i)}$
  end for
  Send $w_{t+1}^{(global)}$ to the edge server;
end procedure

According to the standard federated learning protocol [13], after adding the Laplace perturbation on the features extracted from the client-side DNN, all the perturbed features will be fed to the edge-side DNN to further generate the local model update by running the SGD algorithm. For simplicity, we use $\tilde{x}_i$ to represent the i-th participant's update (i.e., participant i's perturbed features), where $i \in [1, n]$. The SGD mechanism is an optimization method to find the parameter w by minimizing the loss function $\mathcal{L}(w, \tilde{x}_i)$. In a certain communication round t, SGD algorithm first compute the gradient $g_t(\tilde{x}_i)$ for any input features $\tilde{x}_i$ as follow:

$$g_t^{(i)} = \nabla_{w_t} \mathcal{L}(w_t, \tilde{x}_i) \tag{7}$$

In terms of gradient descent process during the local training procedure, we utilize the distributed selective SGD (DSSGD) mechanism instead of the conventional SGD algorithm to achieve distributed computation capability. DSSGD splits the weight w_t and the gradient g_t into n parts, namely $w_t = (w_t^1, \cdots, w_t^n)$ and $g_t = (g_t^1, \cdots, g_t^n)$, so the local parameters update rule becomes as follow:

$$w_{t+1}^{(i)} = w_t^{(i)} - \eta \cdot g_t^{(i)} \tag{8}$$

Then the conventional SGD algorithm was executed to calculate the local model update as:

$$\Delta w_{t+1}^{(i)} = w_{t+1}^{(i)} - w_t^{(i)} \tag{9}$$

At last, each edge server sends the local model updates $\Delta w_{t+1}^{(i)}$ to the cloud server to further executing the federated average procedure:

$$w_{t+1}^{(global)} = w_t^{(global)} + \frac{1}{n} \sum_{i=1}^{n} \Delta w_{t+1}^{(i)}, \tag{10}$$

The whole federated learning procedure will be executed iteratively until the global model tends to convergence.

5 Experimental evaluation

In this section, we conduct a series of experiments on an image classification task and a real mobile application system to evaluate the performance and applicability of our proposed FedMEC. We first examine the effectiveness of our differentially private data perturbation method by applying the convolutional denoising autoencoder to visualize the noise and the reconstruction. Then, we verify the model accuracy under different perturbation strengths (μ, b) in terms of a classification task, and evaluate the mean accuracy trends of the global model by fixing one of the perturbation parameters and changing the other one. In addition, the comparison experiments are conducted to estimate the performance of our scheme with other three related works, including no-privacy [13], local-DP [25], and central-DP [27]. At last, in order to verify the applicability of FedMEC, we implement FedMEC on an Android system in a real mobile device to monitor the CPU temperature and CPU frequency.

In our experiments, we run the federated learning protocol on an image classification task to estimate our FedMEC mechanism. The classification task is conducted on a handwritten digital image dataset MNIST, which contains of 60000 training samples and 10000 testing samples ranging from 0 to 9 (i.e., 10 classes) with the same size of 28×28 pixels. For the experimental settings, a general deep neural network is implemented in FedMEC, which includes 3 convolutional layers and 2 dense layers. The kernel size of all three convolutional layers is 3×3 and the stride for these convolutional layers is setting as 2. The activation functions applied in the DNN structure are LReLU. As aforementioned in Section 4, the perturbation strength (μ, b) are the main parameters in our FedMEC scheme, where μ is the nullification rate and b is the

diversity of the Laplace mechanism. The evaluation indexes are the model accuracy of the image classification task and the CPU frequency and temperature of mobile devices.

5.1 Effectiveness of data perturbation

According to FedMEC, the perturbation strength (μ, b) are two critical parameters in the designed differentially private data perturbation method. Thus, we adopt the convolutional denoising autoencoder [35] under the settings of federated learning to visualize the noise and reconstruction, where the perturbation strength is represented by (μ, b). We train our model based on two perturbation strengths ($\mu = 1\%, b = 1$) and ($\mu = 10\%, b = 5$).

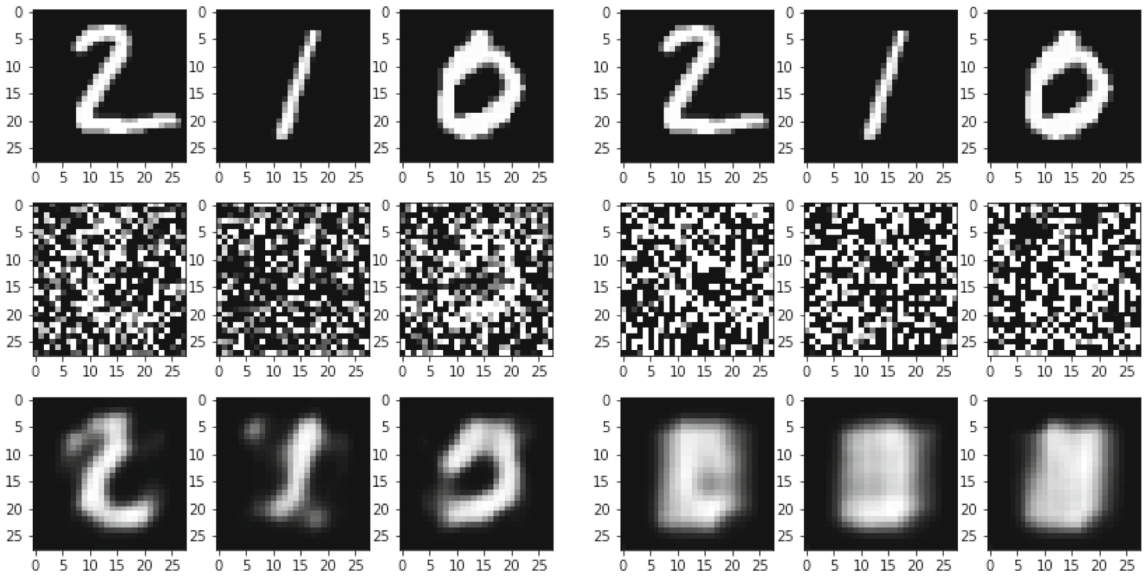
The visualizing noise and reconstruction results are shown in Fig. 5. The three rows in this figure from the top to the bottom represent the original samples, the perturbed samples, and the constructed samples, respectively. According to the perturbation and reconstruction results, we can see that the perturbed digits can be reconstructed to a certain degree at the perturbation strengths of ($\mu = 1\%, b = 1$) as shown in Fig. 5a. However, as shown in Fig. 5b, it is hard to reconstruct the original digital when the perturbation strength reaches ($\mu = 10\%, b = 5$), even the perturbed data is public.

5.2 Performance under different data perturbations

The standard federated learning protocol allows each participant to train their data locally and only updating the model parameters. However, the edge device users may not strictly adopt the perturbation strength specified by the learning services provider, which means edge device users could change their perturbation strength before sending to the edge server. Thus, it is necessary to estimate the impact of our differentially private data perturbation mechanism under different perturbation strength in terms of the global model accuracy. In our experiments, we set two scenarios that the numbers of edge clients n are 100 and 300, and the training is stopped when the communication round reaches 30 and 50 for $n = 100$ and $n = 300$, respectively. The goal of this group of experiments is to estimate the changes of accuracy under two different perturbation strengths ($\mu = 10\%, b = 3$) and ($\mu = 20\%, b = 2$). From the results shown in Fig. 6, we can see that the model can get high accuracy very quickly within several communication rounds in both 100 and 300 clients settings, meaning our FedMEC scheme works well in the settings of federated learning while providing sufficient privacy guarantees. Besides, since the perturbation strengths (μ, b) selected in our experiments are small, the accuracy of the federated model under multiple settings is not too much difference.

5.3 Impact of perturbation strength

We also design a group of experiments to evaluate the model mean accuracy by changing one of the parameters in perturbation strength (μ, b), while keeping another parameter as a fixed value. Here, we consider the mean accuracy for each parameter setting by averaging all the



(a) $(\mu = 1\%, b = 1)$ (b) $(\mu = 10\%, b = 5)$

Fig. 5 Visualization of Noise and Reconstruction

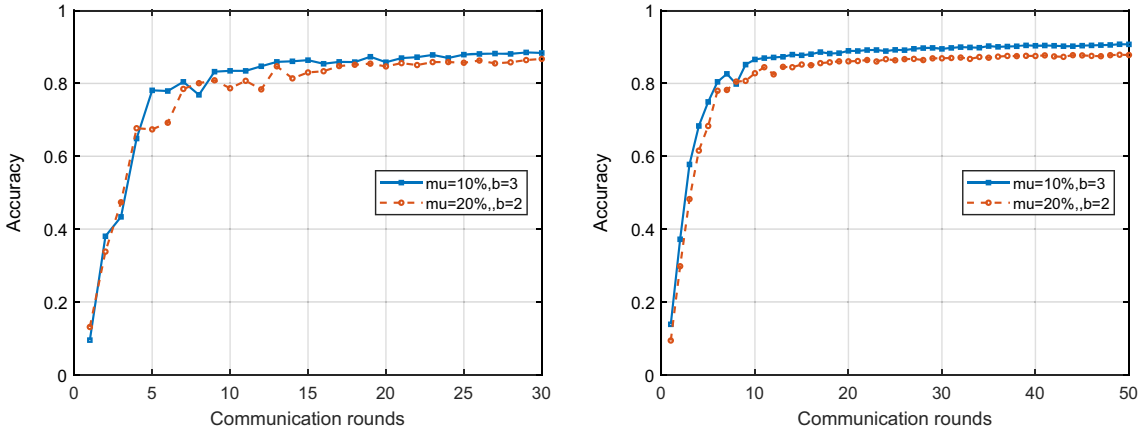


Fig. 6 Classification accuracy on different perturbations

results with 30 and 50 communication rounds for 100 clients and 300 clients. As shown in Fig. 7, our FedMEC scheme can perform more than 85% classification accuracy for all the parameter combinations. Besides, with the gradual increase of perturbation strength, the model accuracy tends to the decreasing trend due to the large perturbation on the features will bring a negative impact in the prediction stage. Moreover, the influence of parameter b is much worse than μ since it is the key factor to determine the differential privacy level. Despite this, the change range of classification accuracy is less than 5%, which shows the stability and validity of our FedMEC scheme.

5.4 Comparison evaluation

To further illustrate the effectiveness, we compare the performance of our proposed FedMEC with other three schemes on an image classification task. The comparison schemes are as follows:

– **No-privacy** [13]: a standard federated learning approach where all the participants train a complete neural network at the local-side and upload the corresponding model updates to the central server.

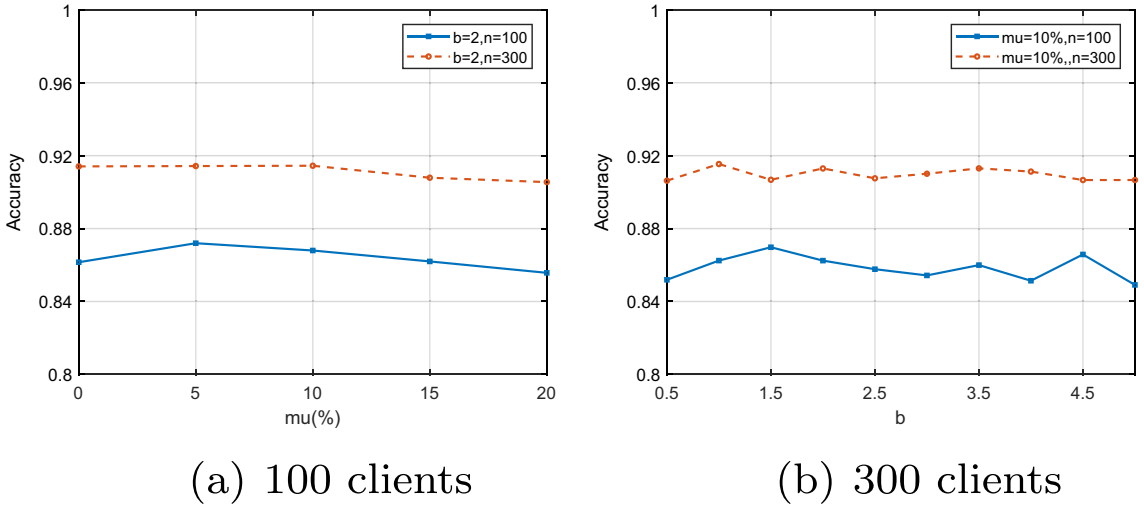


Fig. 7 Mean accuracy on μ and b

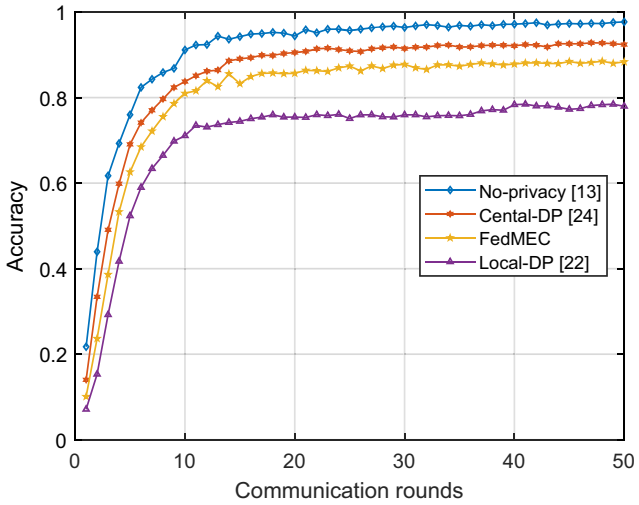


Fig. 8 Accuracy comparison over communication rounds

– **Central-DP** [27]: a randomized mechanism at the server-side to provide differential privacy in the federated learning approach, which a single client's local model updates can be hidden during the aggregation phase.
– **Local-DP** [25]: each client adds enough noise during the local SGD training procedure to protect the training data from being discolored by the untrusted server.

All the four schemes are implemented under the same federated learning settings as described previously. Besides, we set the perturbation strength of our FedMEC scheme as ($\mu = 20\%, b = 2$) and other three schemes as their suggested noisy sizes. Figure 8 shows the results of classification accuracy on 300 clients with 50 communication rounds. Note that "no-privacy" approach is the baseline of standard federated learning approach and it performs the highest classification accuracy (approximately 0.95) in our experiment. Our FedMEC scheme is able to achieve a classification score of about 0.85, which is much higher than the "local-DP" approach (0.75) but little lower than the "central-DP" approach (0.9). The main reason for this phenomenon is that the "central-DP" method can only work in the context of a fully-trusted federated learning scenario, while FedMEC can provide the local-side differentially privacy guarantee even the edge and cloud servers are not trusted.

5.5 Implementation on android

At last, to demonstrate the applicability of our proposed FedMEC framework, we implement the whole scheme in a demo system on HUAWEI MATE20 PRO and a laptop DELL INSPIRON 15. The mobile device is equipped with ARM Cortex-A76@2.6GHz, ARM Cortex-A76@1.92GHz, and ARM Cortex-A55@1.8GHz, and the laptop is fitted out with Intel Core i5-8250U@ 3.4GHz and AMD Radeon-530 Graphics. For the local training samples, we randomly chose 3000 images from MNIST dataset to investigate the CPU status of the mobile device during the client-side DNN training procedure, where all the images are processed consecutively. Similar to [14], we use TensorFlow to generate a deployable model for our Android system and the experiments results are shown in Fig. 9. Specifically, the CPU temperature increases when our demo system executing because the local training procedure consumes

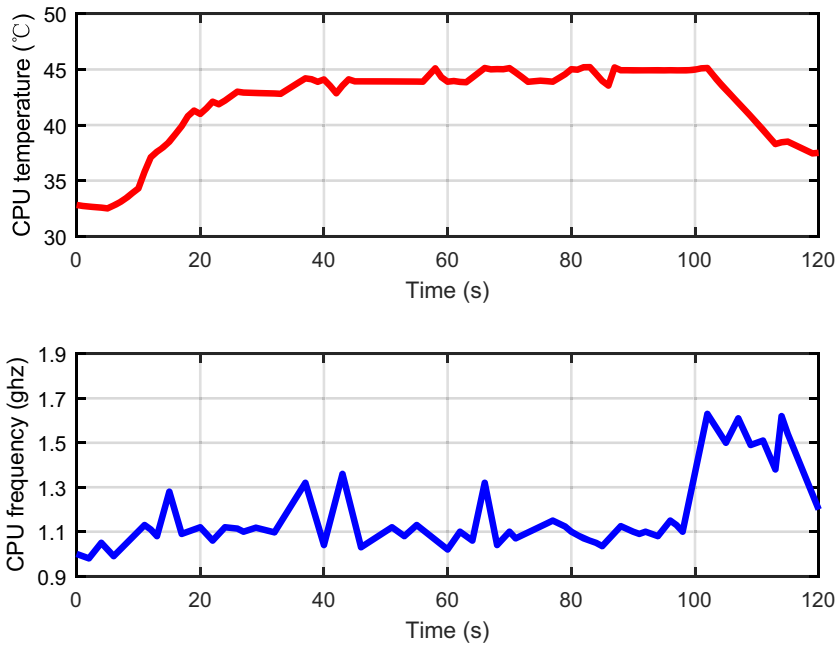


Fig. 9 CPU status during client training procedure

the computation resources of the mobile device. However, it keeps a stable highest temperature values in the interval of 40°C-45°C, which indicates that our FedMEC scheme has a fully acceptable performance in terms of resource consumption. Besides, the trend of CPU frequency further illustrates that the client-side DNN training process only occupies approximately a little more than half of the CPU frequency. Therefore, it is applicable to implement FedMEC on multiple resource-constraint mobile devices.

6 Conclusion

In order to enable highly efficient federated learning protocol and meanwhile protect users' training data privacy, we propose FedMEC in the mobile edge computing environment which can achieve both efficiency and privacy in this paper. The designed FedMEC framework splits a complete deep neural network used in the local training procedure into two parts: client-side DNN and edge-side DNN. A large part of heavy computation works is offloaded to the edge and cloud servers, while the mobile devices merely undertake the lightweight feature extraction part, so as to reduce the computation complexity on the mobile edge devices. Furthermore, to minimize the client-side privacy leakage during the training phase, we introduce a differentially private data perturbation mechanism to perturb the Laplacian random noises to the client-side features before uploading to the edge server. The extensive experimental results on a benchmark dataset demonstrate that our proposed FedMEC scheme can not only achieve high model accuracy but also providing sufficient privacy guarantees. Finally, we also implement FedMEC on a real Android application system to show its applicability and verify the computation overhead. In regards to our further work, we plan to explore the optimal perturbation strength for the differentially private data perturbation mechanism.

Acknowledgment This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802303, in part by the National Natural Science Foundation of China under Grant 61672283 and Grant 61602238, and in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX18_0308.

References

1. Hesamifard E, Takabi H, Ghasemi M, Wright RN (2018) Privacy-preserving machine learning as a service. In: Proc. of 19th Privacy enhancing technologies symposium (PETS), pp 123–142
2. Cao B, Zheng L, Zhang C, Yu PS, Piscitello A, Zulueta J, Ajilore O, Ryan K, Leow AD (2017) DeepMood: Modeling mobile phone typing dynamics for mood detection. In: Proc. of 23rd ACM International conference on knowledge discovery and data mining (SIGKDD), pp 747–755
3. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: A Unified Embedding for Face Recognition and Clustering. In: Proc. of 28th IEEE Conference on computer vision and pattern recognition (CVPR), pp 815–823
4. Zhai X, Guan X, Zhu C, Shu L, Yuan J (2018) Optimization algorithms for Multi-access green communications in internet of things. IEEE Internet of Things J 5(3):1739–1748
5. Zhang J, Chen B, Zhao Y, Cheng X, Hu F (2018) Data security and Privacy-preserving in edge computing paradigm: survey and open issues. IEEE Access 6:18209–18237
6. Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surv & Tut 19(3):1628–1656
7. Zhai XB, Liu X, Zhu C, Zhu K, Chen B (2019) Fast admission control and power optimization with adaptive rates for communication fairness in wireless networks. IEEE Transactions on Mobile Computing, to appear, 2019
8. Li Y, Qi X, Keally M, Ren Z, Zhou G, Xiao D, Deng S (2017) Communication energy modeling and optimization through joint packet size analysis of bsn and wifi networks. IEEE Trans Parall Distr Syst 24(9):1741–1751
9. Konečnỳ J., McMahan HB, Yu FX, Richtárik P., Suresh AT, Bacon D (2016) Federated Learning: Strategies for Improving Communication Efficiency. [Online]. Available: arXiv:1610.05492.
10. Smith V, Chiang C.-K., Sanjabi M, Talwalkar AS (2017) Federated Multi-task learning. In: Proc. of 32nd Annual conference on neural information processing systems (NIPS), pp 4427–4437
11. Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proc. of 22nd ACM Conference on computer and communications security (CCS), pp 1310–1321
12. Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. ACM Trans Intell Syst Technol 10(2):1–19
13. McMahan HB, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Proc. of 20th International conference on artificial intelligence and statistics (AISTATS), pp 1–10
14. Wang J, Zhang J, Bao W, Zhu X, Cao B, Yu PS (2018) Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud. In: Proc. of 24th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp 2407–2416
15. Mao Y, Yi S, Li Q, Feng J, Xu F, Zhong S (2018) Learning from differentially private neural activations with edge computing. In: Proc. of 3rd IEEE/ACM Symposium on edge computing (SEC), pp 90–102
16. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural networks. In: Proc. of 30th Annual conference on neural information processing systems (NIPS), pp 1135–1143
17. Li Y, Hu H, Zhou G (2018) Using data augmentation in continuous authentication on smartphones. IEEE Internet of Things J 6(1):628–640
18. Hayes J, Ohrimenko O (2018) Contamination Attacks and Mitigation in Multi-Party Machine Learning. In: Proc. of 33th annual conference on neural information processing systems (NIPS), pp 6604–6616
19. Li Y, Hu H, Zhou G, Deng S (2018) Sensor-based continuous authentication using cost-effective kernel ridge regression. IEEE Access 6:32554–32565
20. Fredrikson F, Jha S, Ristenpart T (2015) Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures.

In: Proc. of 22th ACM Conference on computer and communications security (CCS), pp 1322–1333
21. Phong LT, Aono Y, Wang T, Hayashi L, Moriai S (2018) Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans Inf Forensics Secur 13(5):1333–1345
22. Osia SA, Shamsabadi AS, Taheri A, Katevas K, Sajadmanesh S, Rabiee HR, Lane ND, Haddadi H (2018) A hybrid deep learning architecture for Privacy-preserving mobile analytics. ACM Trans Knowl Discov Data 1(1):1–21
23. Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, Ranzato M, Senior A, Tucker P, Yang K, Le QV, Ng AY (2012) Large scale distributed deep networks. In: Proc. of 27th Annual conference on neural information processing systems (NIPS), pp 1223–1231
24. Zhang J, Wang J, Zhao Y, Chen B (2019) An Efficient Federated Learning Scheme with differential privacy in mobile edge computing. In: Proc. of 4th EAI International conference on machine learning and intelligent communications (MLICOM), pp 538–550
25. Abadi M, Chu A, Goodfellow I, McMahan HB, Mironov I, Talwar K, Zhang L (2016) Deep Learning with Differential Privacy. In: Proc. of 23th ACM Conference on computer and communications security (CCS), pp 308–318
26. Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends Theor Comput Sci 9(3):211–407
27. Geyer RC, Klein T, Nabi M (2017) Differentially private federated learning: A client level perspective. In: Proc. of 32nd annual conference on neural information processing systems (NIPS), pp 1–7
28. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proc. of 24th ACM conference on computer and communications security (CCS), pp 1175–1191
29. Beimel A, Brenner H, Kasiviswanathan SP, Nissim K (2014) Bounds on the sample complexity for private learning and private data release. Machine Learning 94(3):401–437
30. Li H, Ota K, Dong M (2018) Learning IoT in edge: Deep learning for the internet of things with edge computing. IEEE Network 32(1):96–101
31. Chang S, Li C (2018) Privacy in neural network learning: threats and countermeasures. IEEE Network 32(4):61–67
32. Lane ND, Georgiev P (2015) Can Deep Learning Revolutionize Mobile Sensing? In: Proc. of 16th International workshop on mobile computing systems and applications (HotMobile), pp 117–122
33. Friedman A, Schuster A (2010) Data mining with differential privacy. In: Proc. of 16th ACM International conference on knowledge discovery and data mining (SIGKDD), pp 25–28
34. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural networks 61:85–117
35. Dong C, Loy CC, He K, Tang X (2016) Image Super-resolutio using deep convolutional networks. IEEE Trans Pattern Anal Machine Intell 38(2):295–307