An Efficient Federated Learning Scheme with Differential Privacy in Mobile Edge Computing

Jiale Zhang(✉), Junyu Wang, Yanchao Zhao, and Bing Chen

College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
{jlzhang,wangjunyu,yczhao,cb_china}@nuaa.edu.cn

Abstract. In this paper, we consider a mobile edge computing (MEC) system that multiple users participate in the federated learning protocol by jointly training a deep neural network (DNN) with their private training datasets. The main challenges of applying federated learning to MEC are: (1) it incurs tremendous computational cost by carrying out the deep neural network training phase on the resource-constraint mobile edge devices; (2) existing literature demonstrates that the parameters of a DNN trained on a dataset can be exploited to partially reconstruct the training samples in original dataset. To address the aforementioned issues, we introduce an efficiently private federated learning scheme in mobile edge computing, named FedMEC, with model partition technique and differential privacy method in this work. The experimental results demonstrate that our proposed FedMEC scheme can achieve high model accuracy under different perturbation strengths.

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

1 Introduction

Nowadays the Internet of Things (IoT) devices, such as smartphones, cameras, and medical tools, have shown explosive growth and became nearly ubiquitous. As a distributed intelligent computation architecture, mobile edge computing [1] shows the powerful real-time and on-devices data processing capability, which achieved great success in numerous networking applications. Along with edge computing, on-device deep learning has turned into a universal and indispensable service [2], including recommendation systems, language translation, security surveillance, and health monitoring. However, such intelligent computation

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 in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX18_0308.

X. B. Zhai et al. (Eds.): MLICOM 2019, LNICST 294, pp. 538–550, 2019.
https://doi.org/10.1007/978-3-030-32388-2_46

scenario rely on the users to outsource their sensitive data to the cloud in order to carry out deep learning services, which causes a number of privacy concerns and resources impacts for the smartphone users [3,4].

Federated learning [5,6] is a recent concept which enables training a deep learning model across thousands of participants in a collaborative manner. It allows the users locally train their model in a distributed manner, and upload its local model update, i.e., parameters of gradient and weight, instead of sharing their private data samples to the central server. Participants in federated learning act as the data provider to train a local deep model, and the server maintains a global model by averaging local model parameters (i.e., gradients) generated by randomly selected participants until it tends to convergence [7]. One biggest achievement for federated learning is the corresponding model average algorithm [8], which can benefit from a wide range of non-IID and unbalanced data distribution among diversity participants.

It seems that federated learning is a promising approach to provide on-device deep learning services on mobile edge computing architecture while protecting user-side data privacy. However, we notice that applying the federated learning approach to mobile edge computing environment would face two practical issues:

- It presents tremendous computational cost by carrying out the deep neural network training phase on the resource-constraint mobile edge devices, meaning that the mobile devices cannot afford such heavy computation processing required in federated learning approach [9–11];
- The parameters of a DNN trained on a dataset can still be exploited to partially reconstruction the training examples in that dataset, which means the conventional federated learning mechanism cannot provide strong privacy guarantee against malicious entities, such as edge and cloud servers [12,13].

To address the above problems, we propose an efficiently private federated learning scheme in mobile edge computing, named FedMEC, based on model partition technique and differential privacy method. The main contributions can be summarized as follows:

- We design a flexible framework which enabling federated learning in the mobile edge computing environment based on the model partitioned technique, reducing the computation overhead on the mobile devices. Specifically, the FedMEC framework partitions a deep neural network into two parts: the client-side DNN and edge-side DNN, so the most complex computations can be outsourced to the edge server.
- We also propose a differentially private data perturbation mechanism on the clients-side to prevent the privacy leakage from the local model parameters. In particular, the edge clients and edge server run the different portion of a deep neural network, and the updates from an edge device to the edge server is perturbed by Laplace noise to achieve differential privacy.

The rest of this paper are organized as follows. In Sect. 2, we briefly introduce the basic knowledge of federated learning and differential privacy. The system

framework is presented in Sect. 3, and the construction of proposed FedMEC scheme is detailed in Sect. 4. Extensive experimental evaluation is conducted in Sect. 5. Finally, Sect. 6 gives the conclusion and future work.

2 Preliminaries

2.1 Federated Learning

Federated learning was firstly proposed by Google [8] which aims to build a distributed machine learning models based on massive distribution datasets across multiple devices. Compared to the conventional centralized training method, participants in the federated learning system can locally train a global model using their private data and upload the model update in form of gradients. Such a localized model training method presents significant advantages in privacy preserving because the clients do not need to share their private data to any third party.

During the federated learning, all the clients agree on a common learning objective and model structure. Assuming that m_t is a fraction of sampled participants who own the different private dataset. In a certain communication round t, each client downloads the global model parameters from the server, then the model is trained locally to generate the local model update $\Delta w_{t+1}^{(i)}$ using its own private dataset. Finally, each participant sends the resulting updates back to the server, where the updates are averaged by the central server to obtain a new joint global model:

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

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

2.2 Differential Privacy

Differential privacy [14] provides a rigorous privacy guarantee for randomized algorithms on aggregated sensitive datasets. 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. 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.

3 System Framework

3.1 Federated Learning with MEC

In this section, we present a mobile edge computing structure for federated learning tasks as shown in Fig. 1. Assume a scenario where all the edge devices intend to obtain desired machine learning services from a cloud central server. At the same time, these users try to prevent the leakage of any private information to the cloud server by executing the federated learning protocol. In this situation, we consider a three-layer mobile edge computing framework that provides the perfect architecture supportive of federated learning protocol with multiple participants. Specifically, the entities involved in our framework including the edge devices, the edge servers, and a cloud central server.

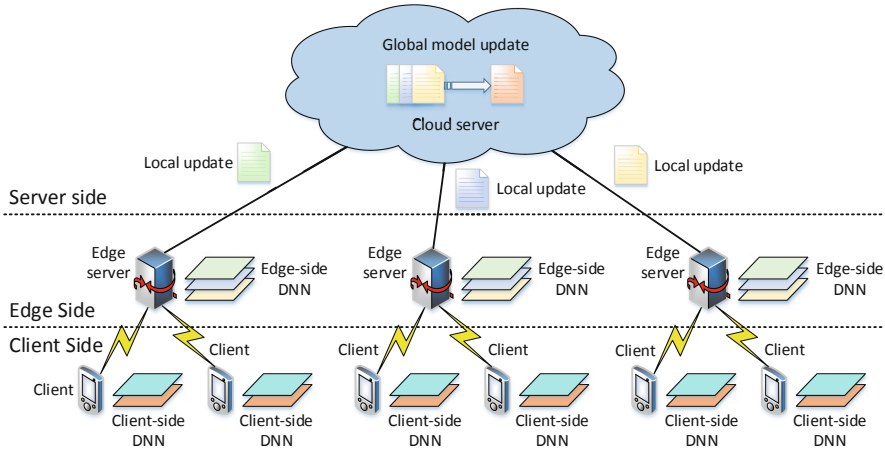


Fig. 1. Federated learning with mobile edge computing

Specifically, implementing the federated learning framework in mobile edge computing faced on two practical issues. Firstly, carrying out the DNN training phase on the mobile devices will definitely present incredible computational cost, while the terminals connected to the mobile edge computing system are usually resource-constraint devices. Secondly, we must consider the user's privacy contained in the outsourced data (or features) due to the edge server and cloud server may not be trusted. 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 challenge, we consider to partition the neural network along the last layer of convolutional layers and all the intermediate results generated by the user-side DNN are hidden from the other entities. 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.

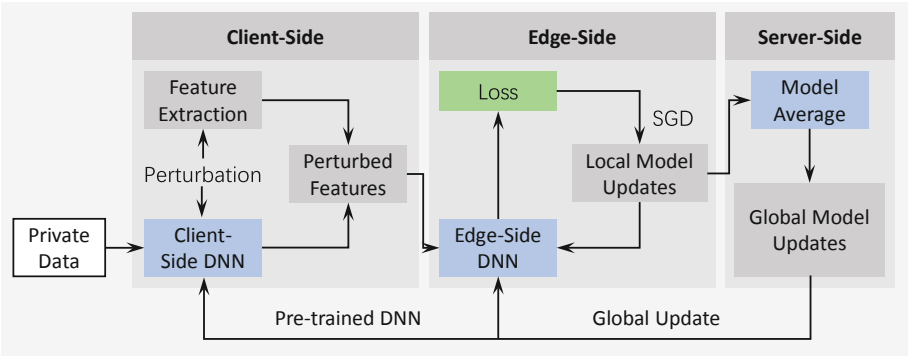


Fig. 2. Overview of proposed FedMEC framework

The overview of FedMEC is presented in Fig. 2. FedMEC relies on the mobile edge computing environment and divides the whole federated learning process into three parts: client-side part, edge-side part, and server-side part. The client-side neural network is assigned by the cloud serve whose network structure and parameters are frozen and the edge-side DNN is fine-turned, the biggest difference between our work and [9] is the iteratively model updates will be aggregated and averaged in the cloud server. In this situation, edge devices merely undertake the simple and lightweight feature extraction and perturbation.

In order to guarantee the performance of the frozen neural network in the client side, we use the public data which has the similar distribution with private data as the auxiliary information dataset to pretrain a deep neural network as an initialized global model in cloud side. Then the pretrained global neural network will be partitioned along the last layer of the convolution layer. Later, the well-trained convolution layer will send to each client for feature extraction. Based on our three-layer federated learning architecture with mobile edge computing, all the resource-hungry tasks are offloaded to the edge servers and cloud center while mobile edge devices merely undertake the simple feature extraction through a local neural network assigned by the cloud center. At last, for the privacy concerns, we perturb the results computed from the original data before being transmitted to the edge server to protect the privacy contained in the raw data.

4 Efficient Federated Learning with Differential Privacy

4.1 Deep Neural Network Partition

In the deep neural network partition strategy, we set the pivot on the last layer of the conversational layers and separate a large DNN into two parts: client-side DNN and edge-side DNN. Specifically, the client-side DNN forms the front portions of a DNN structure (i.e., convolution layers) which are deployed on edge devices to extract features from the raw data. Note that the client-side network is pretrained by the cloud server and the structure and parameters are frozen during the whole training phase in federated learning procedure. The edge-side DNN containing the remaining portions of the DNN network (i.e., dense layers) to update the model parameters by executing the forward and backward propagation procedures. The whole partition process on the deep neural network is illustrated in Fig. 3.

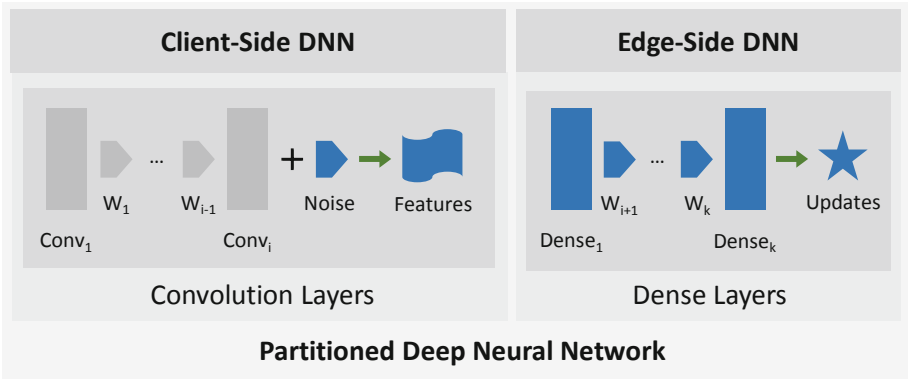


Fig. 3. Partition process on the deep neural network

Therefore, based on our DNN partition mechanism, the complex computation operations on the client side can be greatly reduced. As the experiment shown in [15], the partitioned mechanism can perform lightweight resource consumption when a part of the DNN is offloaded to the third party. In addition to resource and energy considerations, partitioning solutions are attractive to deep 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 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 [12], 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 [13] 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 [14] 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. 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 [9], 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. 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 adding 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 Sect. 2.2, the global sensitivity for a query $f : \mathcal{D} \rightarrow \mathcal{R}$ can be defined as follow:

$$\Delta f = \max_{d \in D, d' \in D'} ||f(d) - f(d')|| \tag{3}$$

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 masking 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 dimensions 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 Sect. 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:

$$x_l' = x_l / \max(1, \frac{||x_l||_\infty}{B}) \tag{4}$$

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 [16], 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:

$$\tilde{x}_l = x_l' + Lap(B/\sigma I) \tag{5}$$

Note that the Laplace noise is added into the final output of the convolutional layers. 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 Differentially Private Federated Learning

According to the standard federated learning protocol [8], 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:

$$g_t^{(i)} = \nabla_{w_t} \mathcal{L}(w_t, \tilde{x}_i) \tag{6}$$

To achieving distributed computation capability, we adopt the distributed selective stochastic gradient descent (DSSGD) mechanism instead of the conventional SGD algorithm into the federated learning procedure. 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:

$$w_{t+1}^{(i)} = w_t^{(i)} - \eta \cdot g_t^{(i)} \tag{7}$$

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

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

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:

$$w_{t+1}^{(global)} = w_t^{(global)} + \frac{1}{n} \sum_{i=1}^{n} \Delta w_{t+1}^{(i)}, \tag{9}$$

The whole federated learning procedure will be executed iteratively until the global model $w_t^{(global)}$ tends to convergence.

5 Experimental Evaluation

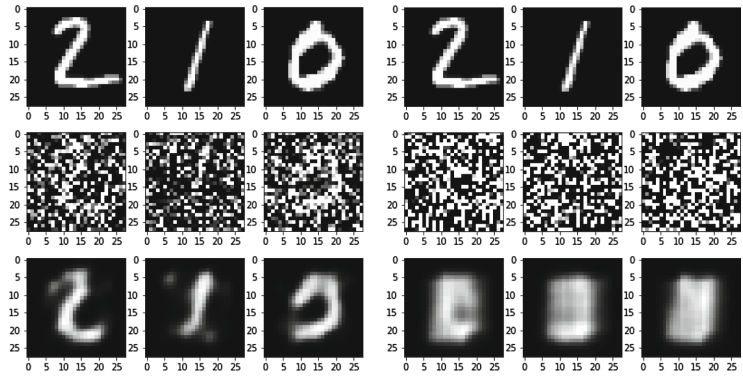
5.1 Dataset and Experiment Setup

Dataset: MNIST (Modified National Institute of Standards and Technology) is one of the popular benchmark datasets which is commonly used in training and testing of deep learning related research fields. The MNIST dataset contains 70000 handwritten grayscale digits images ranging from 0 to 9 (i.e., 10 classes). Each image is with the size of 28×28 pixels, and the whole MNIST dataset is divided into the 60000 training records and 10000 testing data records.

Experiment Setup: In order to estimate our proposed FedMEC algorithm, we run the federated learning protocol on an image classification task. We use the Convolutional Neural Network (CNN) based architecture to construct the classifier in our FedMEC system. The deep neural network structure for MNIST dataset consists of 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 set as 2. In particular, the activation functions applied in the neural network structure is LReLU. As aforementioned in Sect. 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. According to these two parameters, we test the effectiveness of our differentially private data perturbation method by applying the convolutional denoising autoencoder [17] under different perturbation strength. Then, we give a general experimental evaluation under the setting of $\mu = 10\%$ and $b = 3$ to demonstrate the accuracy of our FedMEC scheme. Furthermore, we also test the changes in accuracy when pre-assign different perturbation strengths to the edge clients.

5.2 Experimental Results

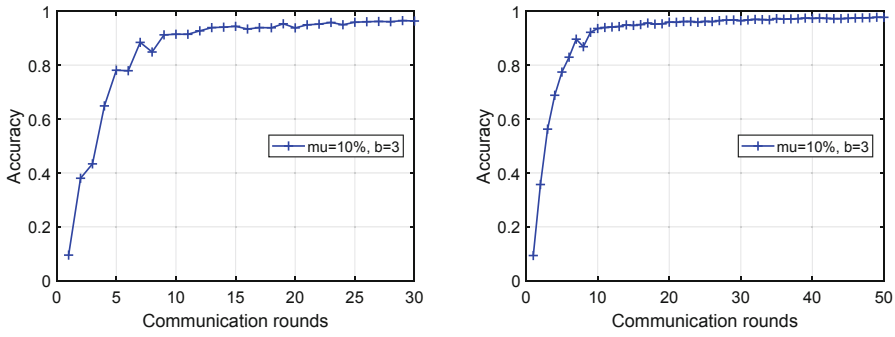
Effectiveness of Data Perturbation: To evaluate the effectiveness of our differentially private data perturbation mechanism, we adopt the convolutional denoising autoencoder under the settings of federated learning to visualize the noise and reconstruction, which the perturbation strength is represented by (μ, b). We train our model based on two perturbation strengths $(\mu = 1\%, b = 1)$ and $(\mu = 10\%, b = 5)$. Figure 4 shows the results of visualizing noise and reconstruction. The first row is the real samples from MNIST dataset and the second row shows the perturbed results under two perturbation strengths by using our differentially private data perturbation mechanism. The last row represents the reconstructed samples based on the convolution denoising autoencoder. 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. 4(a). However, as shown in Fig. 4(b), it is hard to reconstruct the original digital when the perturbation strength reaches $(\mu = 10\%, b = 5)$, even the perturbed data is public.



(a) (1%,1) (b) (10%,5)

Fig. 4. Visualization of noise and reconstruction

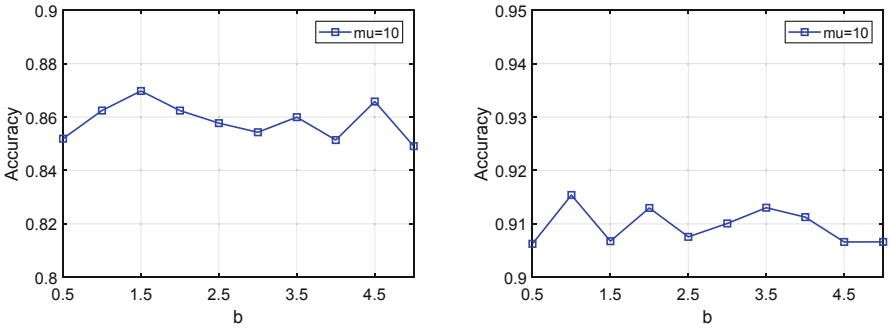
Impact of Data Perturbation: As we know, federated learning allows each participant to training their data locally and only updating the parameters. In this situation, edge device users could change their perturbation strength before sending to the edge server. Thus, we estimate the impact of our differentially private data perturbation mechanism under different perturbation strength on the model accuracy, meaning that the client-side DNN will be trained by the pre-assigned perturbation strength. 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 100 clients and 300 clients, respectively.



(a) (100 clients) (b) (300 clients)

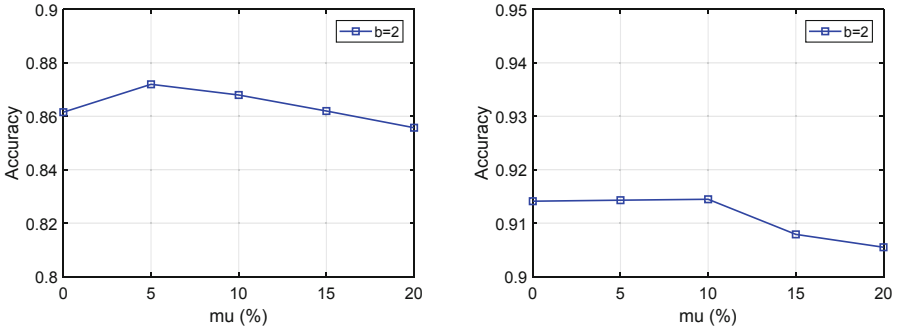
Fig. 5. Accuracy for $\mu = 10\%$ and $b = 3$.

The goal of our first group of experiments is to estimate the changes of accuracy under strength ($\mu = 10\%, b = 3$). From the results shown in Fig. 5,



(a) (100 clients) (b) (300 clients)

Fig. 6. Effect of b.



(a) (100 clients) (b) (300 clients)

Fig. 7. Effect of μ.

we can see that the model can get high accuracy very quickly within several communication rounds in both 100 clients and 300 clients settings, meaning our FedMEC scheme works well in the settings of federated learning while providing sufficient privacy guarantees. We also design a group of experiments to evaluate the global model 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 results with 30 and 50 communication rounds for 100 clients and 300 clients. As shown in Figs. 6 and 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. Despite this, the change range of classification accuracy is less than 5%, which shows the stability and validity of our FedMEC scheme.

6 Conclusion

In this work, we proposed the FedMEC framework which enables highly efficient federated learning service on the mobile edge computing environment. To reduce the computation complexity on the mobile edge devices, we designed a new framework based on the model partition technique to split a deep neural network into two parts, where the most part of heavy computation works can be offloaded to the edge server. Besides, we also presented 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 demonstrated that our proposed FedMEC scheme can achieve high model accuracy while providing sufficient privacy guarantees.

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, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20160805, and in part by the Postgraduate Research & Practice Innovation Program of Jiangsu Province under Grant KYCX18_0308.

References

1. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. **19**(3), 1628–1656 (2017)
2. Hesamifard, E., Takabi, H., Ghasemi, M., Wright, R.N.: Privacy-preserving machine learning as a service. In: Proceedings of 19th Privacy Enhancing Technologies Symposium, PETS, Barcelona, Spain, July 2018, pp. 123–142 (2018)
3. Zhang, Q., Yang, L.T., Chen, Z.: Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans. Comput. **65**(5), 1351–1362 (2016)
4. Zhang, J., Chen, B., Zhao, Y., Cheng, X., Hu, F.: Data security and privacy-preserving in edge computing paradigm: survey and open issues. IEEE Access **6**, 18209–18237 (2018)
5. Smith, V., Chiang, C.-K., Sanjabi, M., Talwalkar, A.S.: Federated multi-task learning. In: Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NIPS, Long Beach, CA, USA, December 2017, pp. 4427–4437 (2017)
6. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)
7. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM Conference on Computer and Communications Security, CCS, Denver, Colorado, USA, October 2008, pp. 1310–1321 (2015)
8. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Agüera y Arcas, B.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS, Fort Lauderadale, Florida, USA, April 2017, pp. 1–10 (2017)

9. Wang, J., Zhang, J., Bao, W., Zhu, X., Cao, B., Yu, P.S.: Not just privacy: improving performance of private deep learning in mobile cloud. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, London, United Kingdom, August 2018, pp. 2407–2416 (2018)
10. Mao, Y., Yi, S., Li, Q., Feng, J., Xu, F., Zhong, S.: Learning from differentially private neural activations with edge computing. In: Proceedings of the 3rd IEEE/ACM Symposium on Edge Computing, SEC, Seattle, WA, USA, October 2018, pp. 90–102 (2018)
11. Osia, S.A., et al.: A hybrid deep learning architecture for privacy-preserving mobile analytics. ACM Trans. Knowl. Discov. Data **1**(1), 1–21 (2018)
12. Fredrikson, F., Jha, S., Ristenpart, T.: Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22th ACM Conference on Computer and Communications Security, CCS, Denver, Colorado, USA, October 2015, pp. 1322–1333 (2015)
13. Phong, L.T., Aono, Y., Hayashi, T., Wang, L., Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. IEEE Trans. Inf. Forensics Secur. **13**(5), 1333–1345 (2018)
14. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3), 211–407 (2014)
15. Lane, N.D., Georgiev, P.: Can deep learning revolutionize mobile sensing? In: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile, Santa Fe, New Mexico, USA, February 2015, pp. 117–122 (2015)
16. Abadi, M., et al.: Deep learning with differential privacy. In: Proceedings of the 23th ACM Conference on Computer and Communications Security, CCS, Vienna, Austria, October 2016, pp. 308–318 (2016)
17. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 295–307 (2016)