2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)

2020 IEEE 22nd International Conference on High Performance Computing and Communications | 978-1-7281-7649-9/20/$31.00 ©2020 IEEE | DOI: 10.1109/HPCC-SmartCity-DSS50907.2020.00073

FedMONN: Meta Operation Neural Network for Secure Federated Aggregation

Dan Meng, Hongyu Li, Fan Zhu, Xiaolin Li
AI Institute, Tongdun Technology
{*dan.meng, hongyu.li, fan.zhu, xiaolin.li*}*@tongdun.net*

***Abstract*—Federated learning enables collaborative machine learning among multiple independent participants while preserving data privacy of each participant through model aggregation during training. However, model aggregation still faces potential risks that are associated with indirect leakage, such as parameters. In this paper, we first propose an algorithm called Meta Operation Neural Network (MONN) to perform basic arithmetic operations on encrypted data and generate operation results in a plaintext way. MONN is actually a general neural network composed of an encoder and a meta operation decoder, where both the encryption and meta decryption are lossless. MONN can be applied to federated learning for secure model aggregation. In this way, data privacy can be well preserved not only from malicious attackers but also untrustworthy servers. Experimental results reveal the following three key properties: 1) The proposed MONN based federated aggregation method (denoted as FedMONN) can reach satisfactory performance comparable with non-federated counterparts; 2) FedMONN is more secure than the classic federated averaging and one-time pad aggregation, even the server is not a trustable third party; 3) FedMONN is much efficient than the federated aggregation based on the Paillier homomorphic encryption technique.**

***Keywords*-federated learning, federated aggregation, collaborative machine learning, data privacy**

I. Introduction

Recently, federated learning [1] has become a hot topic in the research of data privacy protection [2]–[4]. Federated learning is proposed for multi-party, and privacy-preserving machine learning. In federated learning, the participants compute an updated model based on their local data and post gradient updates to the server. The server then aggregates these updates (e.g. by averaging) to construct an improved global model. However, the gradient updates may reveal private information during federated aggregation [5]–[8], which still violates the principle of General Data Protection Regulation (GDPR)[1].

Many research efforts have been devoted to seek for more secure federated aggregation methods and technologies in federate learning. A natural approach to preventing information leakage is adding artificial noises, known as differencial privacy (DP) [9]. Existing works on DP based federated aggregation include [6], [10]. In [6], federated DP aggregation algorithm (denoted as FedDP) was proposed to preserve participants' side privacy, and local parameters in each participants were perturbed by adding noises before uploading to the server for federated aggregation. Although DP based federated aggregation is a feasible solution, it has to make the trade-off between privacy level and model accuracy.

Except for DP based federated aggregation methods, the other encryption technologies (i.e. homomorphic encryption (HE), one-time pad (OTP) [11], and encrypted neural network also become a popular research topic. For example, [12] pointed out that federated learning ensures the confidentiality of data by involving HE technology during parameters transmission and federated aggregation process, while Paillier HE [13] (denoted as FedHE) has mostly been used. However, HE based federated aggregation methods are time consuming even using Paillier HE, not to mention using fully HE. [14] proposed an encrypted neural network (ENN) to solve the privacy protection problem for gradient updates transmission and aggregation in federated learning. Nevertheless, ENN is designed only for the averaging operation and is hard to extend to other basic operations or complex operation combinations, because each operation requires a new ENN to be trained from scratch.

In order to provide possibilities of a more secure and efficient federated aggregation method, in this paper, we propose MONN to perform meta operation on encrypted data and generate the result in a plaintext way simultaneously. In the MONN based federated aggregation algorithm (denoted as FedMONN), each participant uses the public encoder to generate encrypted vectors of local model weights respectively. Once the encrypted vectors are transmitted to the server, a meta operation decoding sub-network deployed on the server is responsible to recover the aggregated encrypted vectors. In this way, the original data will never appear both during transmission and aggregation. As a result, the proposed FedMONN can calculate properly without privacy leaking or directly being exposed to the server.

Our major contributions are summarized as follows: 1) A Meta Operation Neural Network is proposed that can perform meta operations on encrypted data without explicitly reconstructing the original data. 2) A novel secure aggregation method FedMONN is designed in real applications, where it can achieve comparable results when compared with non-federated baseline, without sacrificing any model performance. 3) FedMONN is more secure than the classic

[1] https://www.eugdpr.org

978-1-7281-7649-9/20/$31.00 ©2020 IEEE
DOI 10.1109/HPCC-SmartCity-DSS50907.2020.00073

federated averaging and one-time pad aggregation methods, while much efficient than the federated aggregation using traditional Paillier HE.

II. MONN Based Federated Aggregation

In Section II, we first introduce the architecture of meta operation neural network (MONN), and then illustrate the algorithm of FedMONN. We use the term "**meta operation**" to refer six kinds of basic numeric operations, namely add, subtract, multiply, divide, maximum, and minimum. It worth noticing that more complicated operations such as differentiation, integral, power, etc. are also excutable by combing basic operations. We also introduce the concept of MONN, which provides a seamless solution for encrypting raw input and performing meta operations on a ciphertext space. Based on the secure property of MONN, we propose FedMONN for secure federated aggregation.

A. Meta Operation Neural Network

To derive a general framework, we define an encoder-decoder architecture by a tuple $(N, P, Q, N', \mathbb{F}, \mathbb{G}, \mathbb{R}, \mathscr{E}, \mathscr{D}, \Delta)$, where:

- $\mathbb{F}$, $\mathbb{G}$, and $\mathbb{R}$ are sets, indicating the range of input, embedding, and output space respectively. For example, we can set $\mathbb{F} = \mathbb{R} = \{0, 1\}$, $\mathbb{G} = [-1, 1]$.
- N, P, Q, and N' are positive integers, which represents the input and output dimension of the encoder and decoder.
- $\mathscr{E}$ and $\mathscr{D}$ are series of functions mapping input vectors from $\mathbb{F}^N$ to $\mathbb{G}^P$, and the latent vectors from $\mathbb{G}^Q$ to $\mathbb{R}^{N'}$, respectively.
- Δ is a distortion function defined over $\mathbb{R}^{N'}$ and measures the loss between the target and the decoded results.

In practice, we usually select multi-layer perception (MLP) or convolutional neural networks as the mapping functions for $\mathscr{E}$ and $\mathscr{D}$.

The proposed MONN is composed of a shared encoder E and a meta operation decoder $f_{\bar{D}}$, and we adopt a two-stage training scheme to obtain the lossless encoder $E \in \mathscr{E}$ and $f_{\bar{D}} \in \mathscr{D}$. At the first stage, MONN focuses on training a lossless encoder E. With the guide of the raw input, latent representations generated by E is lossless. At the second stage, MONN pays attention to optimize meta operation decoder $f_{\bar{D}}$ on latent representations with fixed E.

In particular, two stages are described as follows:

- ***Stage 1***: Given a set of data $X = \{x_i\}, i = 1, 2, \cdots, m$, in order to get the lossless encryption encoder E, we need the help of a decoder $D \in \mathscr{D}$ satisfying:

$$L_A(X) = \sum_{i=1}^{m} \Delta(x_i, D \circ E(x_i)) \tag{1}$$

  where Δ is the distortion function, and L_A is the overall distortion loss, which is restricted to zero. For distortion function Δ, we simply use binary cross entropy loss denoted as:

$$\Delta = -\sum_{n=1}^{N'} (\hat{z}_n log z_n + (1 - \hat{z}_n) log(1 - z_n)) \tag{2}$$

  where z and $\hat{z}$ are the ground truth and predict vector respectively, N‘ equals to the node number in the last layer of the decoder.

  To achieve lossless encryption, the overall distortion loss L_A must converge to 0. Since L_A is supervised by the original input, the input dimension (N) of the encoder E and output dimension (N') of the decoder D should be the same, which is also suitable to the value range. Consequently, $N = N'$, and $\mathbb{F} = \mathbb{R}$. Besides, the output of the encoder is sent into the decoder, so we have $P = Q$. In our experiments, we set $\mathbb{F} = \{0, 1\}$, $\mathbb{G} \in [-1, 1]$, $N = 64$, $P = 8N$, and use a three layer MLP for E and D.

  In this work, once the lossless encoder E is obtained, D is not used anymore and must be destroyed for privacy protection.

- ***Stage 2***: We then move on to train the meta operation decoder $f_{\bar{D}}$. In other words, given m' training data pairs and each data pair has k input data, then $\forall \bar{x}_i \in \bar{X}, i = 1, 2, \cdots, m'$, and $\bar{x}_i = \{\bar{x}_i^1, \bar{x}_i^2, \cdots, \bar{x}_i^k\}$, the encrypted vectors $E(\bar{x}_i) = [E(\bar{x}_i^1), E(\bar{x}_i^2), \cdots, E(\bar{x}_i^k)]$ are generated with the lossless encoder E obtained in the first stage. In accordance with the definition of **meta operation**, each specified $f_{\bar{D}}$ should directly working out the plaintext result with k encrypted inputs:

$$f_{\bar{D}}(E(\bar{x}_i)) = O(\bar{x}_i) \tag{3}$$

  where $O(\cdot)$ is a specific meta operation conducted on plaintext, such as add, multiply, and maximum.

  Similar to the encoder-decoder architecture $D \circ E$ mentioned in Stage 1, we define a meta operation decoder $f_{\bar{D}} \in \mathscr{D}$ with the pre-trained encoder E obtained in Stage 1 as $f_{\bar{D}}(E(\cdot))$, where the input dimension (Q) of meta operation decoder is k times as the output dimension (P) of the pre-trained encoder, namely $Q = kP$. By concatenating the encrypted vectors, meta operation decoder $f_{\bar{D}}$ transforms the encryption data from $\mathbb{G}^{kP}$ into $\mathbb{R}^{N'}$. As a result, meta operation decoder can be further defined to find $f_{\bar{D}} \in \mathscr{D}$ that minimize the overall distortion function Δ given fixed E:

$$L_M(\bar{X}) = \sum_{i=1}^{m'} \Delta(O(\bar{x}_i), f_{\bar{D}}(E(\bar{x}_i))) \tag{4}$$

  We also use MLP to simulate meta operation decoder $f_{\bar{D}}$. For example, with the specific add meta operation $O(\bar{x}_i) = \sum_{j=1}^{k} \bar{x}_i^j, i = 1, 2, \cdots, m'$ as the optimization objective, $f_{\bar{D}}$ is a lossless since it performs meta

operation on encryption data and outputs exactly the same result of add meta operation $O(\cdot)$ on plaintext.

Table I: Training Time for Different Methods.

Training Methods	Average Training Time (s/epoch)	Average Federated Aggregation Time (s/epoch)
CNN	14.24	-
FedAvg*	12.28	5.16
FedOTP* [11]	25.80	18.68
FedMONN*	42.58	35.46
FedHE* [13]	491.43	484.31

* For federated aggregation methods, we use two participants.

1) Secure Property of MONN: MONN is designed to protect data privacy, and there are at least three points that can validate MONN's secure property. First, even if the encrypted data pairs $E(\bar{x}_i)$ are intercepted by the malicious attackers during transmission procedure, it is still hard to reconstruct the raw data since attackers have no information about the decoder. Second, there is no need to decrypt the raw input data for the meta operation decoder in MONN. As a result, MONN is secure to the untrustworthy third party. Last but not least, meta operation decoder $f_{\bar{D}}$ preserves a many-to-one relationship between raw input data pairs and the output result, which is an irreversible process (e.g. when $\bar{x}_1 = \{3, 5\}$, $\bar{x}_2 = \{2, 6\}$, and $O(\cdot)$ is an add meta operation, then $f_{\bar{D}}(E(\bar{x}_1)) = f_{\bar{D}}(E(\bar{x}_2)) = 8$).

2) Relation to HE: The MONN is in essence a HE technique, because it allows meta operations on encrypted data without exposing sensitive data, and can generate the decryption results which matches the results of the operations as if they had been performed on the plaintext.

The difference lies in that MONN does not need key generation, and the output of MONN is a plaintext result so there is no need to perform the decoding process when compared with HE method. We compare the performance of the most popular federated aggregation methods and our proposed FedMONN in Table I.

B. Secure Aggregation: FedMONN

Suppose k participants try to learn a machine learning model collaboratively with the help of a server, and communication between participants is forbidden. Formally, we denote the classic federated averaging aggregation as FedAvg, and define the weights aggregated by the server as:

$$w = \frac{1}{k}\sum_{j=1}^{k} w^j \tag{5}$$

where w_j is the weights of j^{th} participant. As mentioned in [13], data privacy still can be unwillingly extracted from

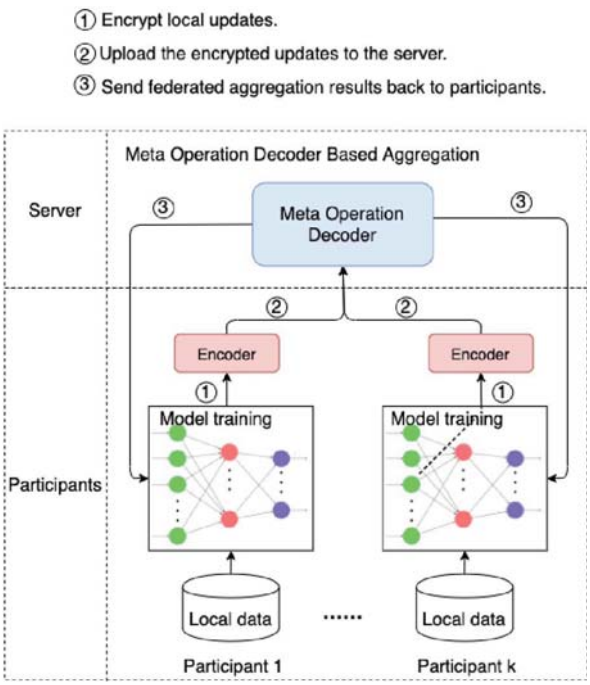


Figure 1: Architecture of FedMONN for federated learning with k participants.

those parameters by FedAvg.

In this paper, by taking advantages of the secure property of MONN, we design a secure federated aggregation method based on MONN, called **FedMONN**, that ensures no leakage of information from any participants once a lossless encryption encoder has been allocated. On the one hand, even if the encrypted parameters are locally intercepted by the malicious attackers unfortunately, it is hard to reconstruct the raw data since attackers have no information about the decoder. On the other hand, the meta operation decoder on the server only calculates the whole updated parameters based on the pre-defined meta operation for aggregation, such as add and average, but will not reconstruct the individual parameters since server has only meta operation decoder and no the decoder for reconstructing raw data. The advantage is that the model updates for a participant are secure even the server is not a trustable third party. We define parameters aggregated by the server using FedMONN as:

$$w = \frac{1}{k} f_{\bar{D}}(E(\bar{w})) \tag{6}$$

where $\bar{w} = \{w^1, w^2, \cdots, w^k\}$, and $f_{\bar{D}}$ is an add meta decoder. MONN is unrelated to federated model, and is an independent training process. As a result, MONN is trained on the server firstly. After L_A converging to 0, the lossless encryption encoder E is distributed to every participant while D is discarded. And after L_M equals to 0, meta operation decoder $f_{\bar{D}}$ is kept on the server. As shown in Fig. 1, each participant is equipped with the shared encoder E, and the

server kept the meta operation $f_{\bar{D}}$.

Then federated learning gets started and participants will be triggered to train a local model for a specific task with their own data. During federated training, local updates are going to be passed to the pre-trained lossless encryption encoder E in order to encrypt them. Once the server receives the encrypted vectors from a certain number, the meta operation decoder $f_{\bar{D}}$ will compute the aggregated gradient updates without explicitly recovering parameters of any participant. The server will synchronously distribute the aggregated meta operation updates to these participants, and then a new round of iteration starts for model training on participants. Algorithm 1 outlines the federated training process including our proposed FedMONN:

Algorithm 1: FedMONN for Federated Learning.

Input: E, $f_{\bar{D}}$, $w_{(0)}$, T
Output: $w_{(T)}$

```
for t = 0 : T do
    Initialization: w^j_(t) = w_(t)
    Local training process:
    for P_j ∈ {P_1, P_2, ··· , P_k} do
        Update local parameters w^j_(t) using local data
        Encrypt local parameters E(w^j_(t)), and upload to server
    end
    Federated aggregation process:
    Update the global parameters w_(t) using Equation (6)
    The server broadcasts global parameters w_(t) back to each participants
    Local testing process:
    for P_j ∈ {P_1, P_2, ··· , P_k} do
        Test FedMONN results w_(t) using local testing dataset
    end
    t ← t + 1
end
```

III. Experimental Results

To estimate the performance of our proposed FedMONN, we simulated participants connecting with a server and trained a convolutional neural network (CNN) model for classification on the MNIST [15] and CIFAR-10 dataset [16]. For both federated and non-federated learning methods, we set the initial learning rate to 0.01, batch size equals to 128, and use SGD as the optimizer. All the experiments done on the machine with Intel Xeon Gold 6130 CPU and Tesla P100.

A. MNIST experiments

We study two ways of partitioning the MNIST data over participants: 1) IID, where the data is shuffled, and then partitioned into three participants (Alice, Bob, and Charile), each receiving 20,000 images, and 2) Non-IID, where Alice, Bob, and Charile kept characters of types 0-3, 4-6, and 7-9 respectively. The CNN model for MNIST dataset consists of two 5×5 convolutional layers[2], a fully connected layers with 50 units and ReLU activation, a dropout layer with dropout rate of 20%, and a final softmax output layer.

For the MNIST classification model learning without federated learning, the accuracy can be as high as 99.40% on the test set, which will act as a baseline in this experiment. For the federated learning framework, each participant (Alice, Bob, and Charile) trained local CNN model and updated the gradients for aggregation after every epoch.[3] But during the test time, all types of character images were pooled together. In this way of data splitting, gradient updates of each participant were indispensable to the aggregated model.

Fig. 2(a) and Fig. 2(b) show the test loss of the participants and server. For both IID and non-IID data distribution, FedMONN converges much quickly and has better generalization ability than any participant, which is capable to classify characters of 0-9. We also list the test accuracy of FedMONN with IID and non-IID data distribution in Fig. 2(c), showing FedMONN converges smoothly and quickly with IID data distribution.

Fig. 2(d) compares the test accuracy of the baseline, and federated aggregation methods (such as FedAvg, FedHE, FedOTP[4], and FedMONN) under IID data distribution. It is observed that the accuracy of FedMONN can reach the comparable accuracy of classical federated aggregation method–FedAvg, and the gap exists in the converge speed may be caused by the different batch data sampled during training time.

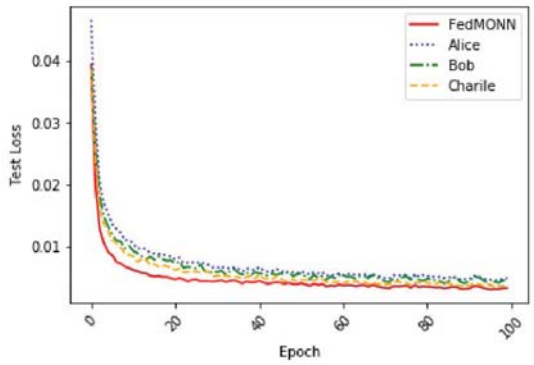
Table I shows the average training and federated aggregation time of the above mentioned four federated aggregation methods on MNIST IID data. The average training time of FedAvg is faster than CNN, benefiting by the decentralization distributed architecture.

Consider factors of model accuracy, training time, and data privacy, federated learning methods can get satisfied results when compared with baseline at the cost of the converge speed. From the aspect of data privacy protection, FedAvg is hardly related to data privacy protection, while FedHE, FedOTP, FedMONN are much safer. However, FedHE achieves the comparable performance at the cost of lots of training time, while FedOTP requires key agreement and exchange among participants, which still faces risks of information leakage to third party. Consequently, FedMONN is the optimal federated aggregation method among these

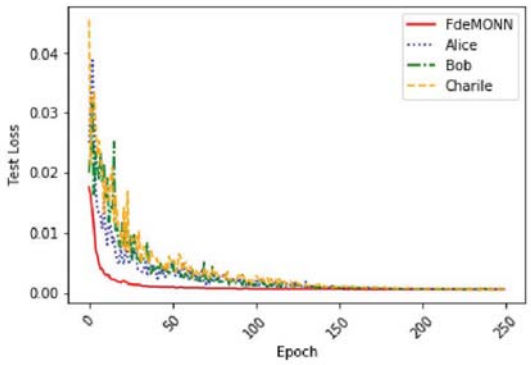
[2]The first with 10 channels, the second with 20 channels, each followed with 2×2 max pooling and ReLU activation.

[3]We can choose whether to encrypt the local gradients before aggregating, and the encrypt methods, e.g. HE, MONN, one-time pad, and etc..

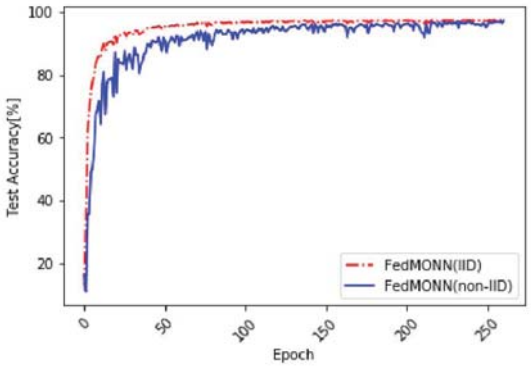
[4]FedOTP is shorted for federated aggregation method using one-time pad [11].



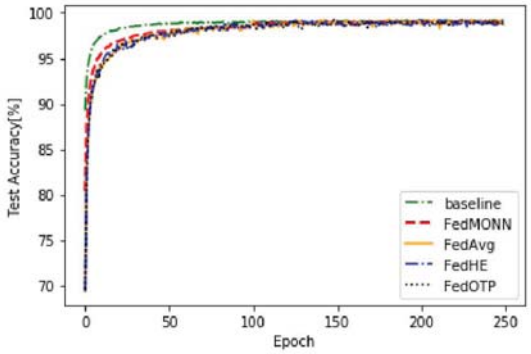
(a) Test loss for MNIST dataset with IID distribution



(b) Test loss for MNIST dataset with non-IID distribution



(c) Test accuracy for MNIST dataset with IID and non-IID distribution



(d) Test accuracy for different federated aggregation methods

Figure 2: Performance of the proposed FedMONN. Fig. 2(a) and Fig. 2(b) are the test loss of three participants and a server on MNIST dataset, with IID and non-IID data distribution. Fig. 2(c) shows the comparison of test accuracy for MNIST IID and non-IID data. Fig. 2(d) is the test accuracy curves for FedAvg, FedDP, FedHE, FedOTP, and FedMONN.

four counterparts.

B. CIFAR experiments

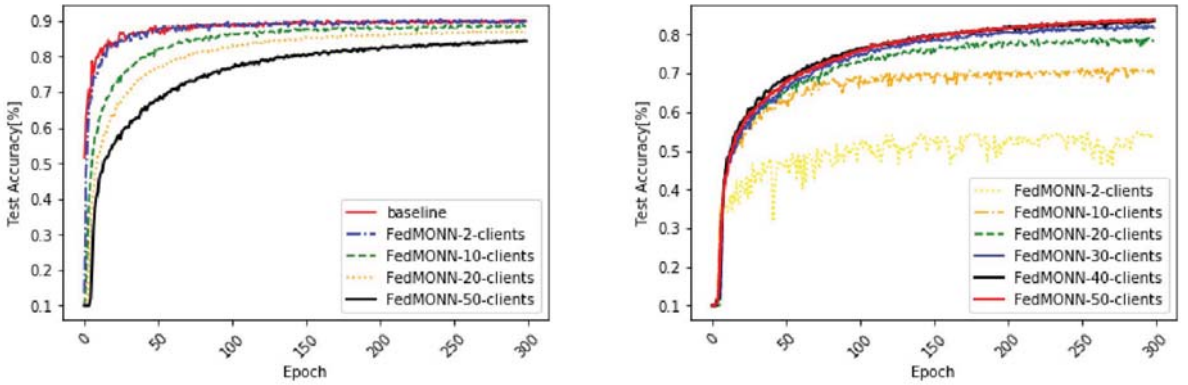
We also ran experiments on the CIFAR-10 dataset to further validate FedMONN. CIFAR-10 dataset consists of 10 classes of 32× 32 images with three RGB channels. There are 50,000 training images and 10,000 testing images. The state-of-the-art approaches for CIFAR-10 dataset have achieved 96.5% [17] accuracy on the testing set. The CNN model for CIFAR-10 classification consists of two convolutional layers[5], two fully connected layers[6], and a final softmax classification layer. Although the CNN model we use in our experiment is simply, it is sufficient for our needs. Since our goal is to evaluate that our proposed FedMONN can achieve comparable performance as non-federated methods, not to achieve the best accuracy on this task. The images are preprocessed as part of the training input pipeline, which consists of randomly cropping the images to 24×24, flipping left-right, and adjusting the contrast.

For the baseline of CIFAR-10 dataset, we train the CNN model on the full training set, reaching a 90.21% testing accuracy after 300 epochs. It is worth noting that participant can join and leave flexibly under federated learning scenario, so we explored the influence of different number of participants in two settings.[7] One case is to participate the training images into 2, 10, 20, and 50 participants, each participant containing 5,000, 1,000, and 2,000 images respectively. In Fig. 3(a), FedMONN with 2 participants can reach comparable accuracy with the baseline counterpart, and FedMONN needs more epochs to converge as the number of participants increased. The other case is that the number of participant varies from 2 to 50, and each participant holds the same training images, which we set 1,000 in our experiments. Experimental results are listed in Fig. 3(b), indicating that with the number

[5]The first with 32 channels, the second with 64 channels, each followed with 2 × 2 max pooling and ReLU activation.

[6]The first with 512 units, the second with 32 units, each followed with ReLU activation.

[7]As there isn't a natural participant-level partitioning of this data, we only considered the IID setting.



(a) Test accuracy with different number of participants

(b) Test accuracy with each participant holds the same number of training samples

Figure 3: Performance of FedMONN with different number of participants.

of participants increasing[8], FedMONN can achieve better performance, and similar phenomena has also been observed in the centralized non-federated training methods.

IV. Conclusion

In this paper, we propose MONN and apply it to federated learning for secure aggregation. MONN is actually a general neural network consisting of an encoder and a meta decoder, which can perform some basic arithmetic operations, such as add, subtract, multiply, divide, maximum, and minimum. Consequently, MONN provides possibilities of more flexible and diverse operation combinations. In MONN based federated aggregation (FedMONN), model updates in each participant are first locally encrypted with the encoder, then the encrypted vectors are transmitted to the third party server, and finally model updates are aggregated with the meta operation decoder on the server. Since the encrypted vectors cannot be decrypted by malicious attackers and the third party server, data privacy is well preserved and thus more secure.

References

[1] H. Li, D. Meng, H. Wang, and X. Li, "Knowledge federation: A unified and hierarchical privacy-preserving AI framework," in *IEEE International Conference on Knowledge Graph (ICKG)*, 2020, pp. 84–91.

[2] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart internet of things systems: a consideration from a privacy perspective," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 55–61, 2018.

[3] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[4] J. Zhou, J. Sun, P. Cong, Z. Liu, X. Zhou, T. Wei, and S. Hu, "Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 745–758, 2020.

[8]The number of training samples involved in the training phrase is also increased.

[5] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM SIGSAC conference on computer and communications security*. ACM, 2015, pp. 1310–1321.

[6] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[7] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: user-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[8] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. Quek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Network*, 2020.

[9] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[10] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: algorithms and performance analysis," *arXiv*, pp. arXiv–1911, 2019.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[12] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[13] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[14] H. Li and T. Han, "An end-to-end encrypted neural network for gradient updates transmission in federated learning," in *Data Compression Conference (DCC)*, 2019, pp. 589–589.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[17] B. Graham, "Fractional max-pooling," *arXiv preprint arXiv:1412.6071*, 2014.