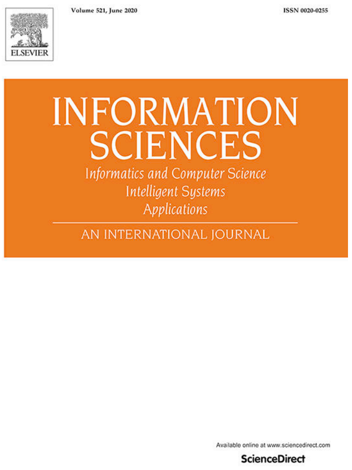


PPCL: Privacy-Preserving Collaborative Learning for Mitigating Indirect Information Leakage

Hongyang Yan, Li Hu, Xiaoyu Xiang, Zheli Liu, Xu Yuan

PII: S0020-0255(20)30974-9
DOI: https://doi.org/10.1016/j.ins.2020.09.064
Reference: INS 15903

To appear in: *Information Sciences*

Accepted Date: 27 September 2020

PPCL: Privacy-Preserving Collaborative Learning for Mitigating Indirect Information Leakage

Hongyang Yan[a], Li Hu[a,*], Xiaoyu Xiang[a], Zheli Liu[b], Xu Yuan[c]

[a]*School of Artificial Intelligence and Blockchain, Guangzhou University, Guangzhou, P.R. China.*
[b] *School of Cyber Science, Nankai University, Tianjin, P.R. China.*
[c] *University of Louisiana at Lafayette, Lafayette, LA, USA.*

Abstract

Collaborative learning and related techniques such as federated learning, allow multiple clients to train a model jointly while keeping their datasets at local. *Secure Aggregation* in most existing works focus on protecting model gradients from the server. However, an dishonest user could still easily get the privacy information from the other users. It remains a challenge to propose an effective solution to prevent information leakage against dishonest users.

To tackle this challenge, we propose a novel and effective privacy-preserving collaborative machine learning scheme, targeting at preventing information leakage agains adversaries. Specifically, we first propose a privacy-preserving network transformation method by utilizing Random-Permutation in Software Guard Extensions(SGX), which protects the model parameters from being inferred by a curious server and dishonest clients. Then, we apply Partial-Random Uploading mechanism to mitigate the information inference through visualizations. To further enhance the efficiency, we introduce network pruning operation and employ it to accelerate the convergence of training. We present the formal security analysis to demonstrate that our proposed scheme can preserve privacy while ensuring the convergence and accuracy of secure aggregation. We conduct experiments to show the performance of our solution in terms of accuracy and efficiency. The experimental results show that the proposed scheme is practical.

Keywords: Collaborative Learning, Privacy-Preserving, Network Transformation, Network Pruning

1. Introduction

Machine learning algorithms have been widely adopted in various multimedia applications, including image classification, face recognition, video retrieval, and many others. Restricted by the available power of local clients, the corresponding tasks are usually outsourced to the centralized and powerful server for

*Corresponding author
Email address: `hu335533@foxmail.com` (Li Hu)

training, which inevitably leads to the loss of control privilege from client's perspectives. Collaborative learning , a promising solution to address this security concern, allows two or more participants to construct a joint model with their own training data. Several architectures have been proposed such as federated learning [1, 2, 3, 4]. It targets to train a global model for multiple participants while retaining each user's local data. Specifically, all participants can upload their respective local models to a central server, who will aggregate them into a global model. The aggregated global model will be then distributed to each participant for training in another round until the convergence of final global model. Comparing to the traditional centralized machine learning paradigms where all users upload all data to a central server, this solution can protect data privacy to some extent.

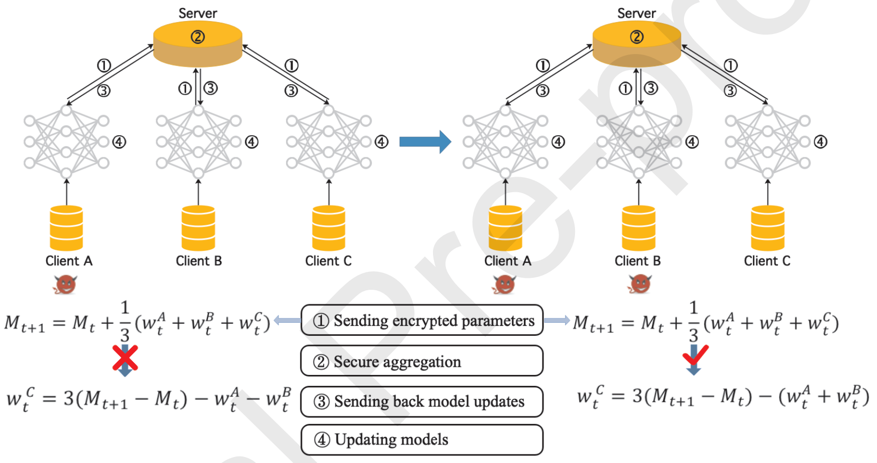

Server
②
①
③
④
Client A
Client B
Client C
$M_{t+1} = M_t + \frac{1}{3}(w_t^A + w_t^B + w_t^C)$
$w_t^C = 3(M_{t+1} - M_t) - w_t^A - w_t^B$
$w_t^C = 3(M_{t+1} - M_t) - (w_t^A + w_t^B)$
① Sending encrypted parameters
② Secure aggregation
③ Sending back model updates
④ Updating models

Figure 1: An example to illustrate the information leakage in collaborative learning.

Model training under the collaborative learning protects the data of each user, by enabling the data stored locally so as to avoid the direct disclosure of private data. However, existing researches [5, 6, 7, 8, 9, 10] have proven that the model parameters uploaded by the distributed users can also lead to partial data disclosure in learning process, which is called as ***Indirect Information Leakage:*** *According to the model parameters uploaded by each client, the raw data information contained in the local client can be inferred.* That is, an attacker (maybe a honest-but-curious server or a malicious client) can directly intercept the model parameters uploaded by each client, and may gain the privacy information of other honest parties. In the medical application scenario where several medical institutions jointly train a disease detection system, the collusion of other parties will lead to serious privacy disclosure of data of one party.

To mitigate such information leakage, secure aggregation proposed by Bonawitz *et al.* [11] ensures that the model updates do not reveal important information to the server. However, it is ineffective for clients if they are dishonest or collusion. For example, Figure 1 illustrates the general framework of three-party collaborative learning system. In this figure, M_{t+1} denotes the $(t+1)^{th}$ aggregated model and w_t^i denotes the t^{th} model parameter of a client $i \in \{A, B, C\}$. According the formula $M_{t+1} = M_t + \frac{1}{3}(w_t^A + w_t^B + w_t^C)$, it is impossible for a malicious client to derive the model parameters w_t^C from model updates. However, if the client A is in collusion with client B, it is vulnerable to infer the contribution input of client C from the formula. More seriously, a dishonest client could obtain model parameters of the other one directly according to the formula in two-party collaborative learning scenario. This leads to the sensitive information leakage of honest clients. Obviously, the existing solutions are defective in this situation. To date, there is no effective solution to prevent such information leakage in collaborative learning.

In this paper, we aim to design a secure privacy-preserving collaborative learning framework to prevent the information leakage tailored for dishonest clients or clients collusion situation. Specifically, we propose a novel network transformation mechanism to alleviate the indirect information leakage by Random-Permutation. Meanwhile, the prominent SGX technology [12, 13] is employed at the server side to secure the model parameters aggregation and network transformation. Afterwards, we utilize a Partial-Random Uploading mechanism [5, 14, 15] to selectively upload the model parameters to the server, against the indirect information disclosure to the clients. Furthermore, to reduce the time cost for training, we continue to employ the neural network pruning operation [16] in our framework to prune the redundant parameters. To maintain the model accuracy, our pruning operation is only applied in the first few rounds of training so that we can reduce the time consumption of model training as much as possible while maintaining the accuracy of neural network model. We give the security analysis both at the server and clients inspired by introducing Random-Permutation and Partial-Random Uploading mechanism, and the probability of a dishonest client successes in infering the privacy information is negligible. We test our scheme on MNIST dataset and the experimental results demonstrate that the proposed scheme does not affect model accuracy.

The main contributions of this work are summarized as follows:

- We proposed a novel privacy-preserving framework for collaborative learning, targeting to alleviate the indirect information leakage for dishonest clients or clients collusion situation. Our solution takes advantage of the prominent technologies, such as Random-Permutation, SGX, Partial-Random Uploading, to effectively mitigate the information leakage to the dishonest clients and curious servers.
- We further employ the network pruning operations to smartly prune the redundant model parameters to make our solution converge fast, which thus significantly improve the computation efficiency.
- We provide the formal security analysis to show that the probability of a dishonest client successes in infering the privacy information of other party is $\frac{1}{(n_i - \beta n_i(1-\gamma_i))!}$, which is negligible.
- We conduct experiments on MNIST dataset to validate our solutions. The experimental results exhibit that our proposed solution achieves a good

performance in terms of accuracy and efficiency.

Organization. The remainder of this paper is organized as follows. Section 2 gives the preliminary knowledge of deep learning, SGX, and network pruning. In Section 3, we present our detailed designs of privacy-preserving collaborative machine learning framework for preventing information leakage. In Section 4, we formally analyze the security of our design. The system evaluation and experimental results are presented in Section 5. Section 6 discusses the related works and Section 7 concludes this paper.

2. Preliminaries

2.1. Deep Learning

A traditional machine learning model can be represented by the function $f(x_i, w) \rightarrow y_i$, where w is the model parameter from a bunch of data $D = \{(x_1, y_1), ..., (x_n, y_n), i = 1, ..., n\}$. Like traditional machine learning, deep learning is also a process of learning features and functions from data, but different from that, it composes of a non-linear mapping layer from input to intermediate hidden state and then to the output. In the deep learning model, each connection between layers has a floating-point weight matrix as the parameters, and these weights are updated in the training. The topology of the connection between layers is related to tasks, which is very important for the model accuracy.

Deep neural networks have oblivious advantages in the processing of high dimensional complex data, such as image recognition [17, 18], prediction-decision [19], speech recognition [20, 21, 22], genetics [23, 24]. In this paper, we mainly focus on the classification network for image recognition using convolutional neural network (CNN). The CNN model consists of layers that process visual information. There are several different types of layers in CNN, which is shown in Figure 2, generally including convolution, pooling, and fully connected layers. The network model receives the image as input and outputs the prediction class of the image.

In neural networks, the gradient of each weight parameter is calculated by forward and back propagation programs. The traditional gradient descent [25] updating method required a traversal of all data for each update, which was not feasible when the data size was large scale. Hence, some updated stochastic gradient descent (SGD) method was proposed, only through a random data (x_n, y_n) to obtain a "gradient", to update model parameters. Through random optimizing loss function on a pair of training data in each iteration, each round parameters update speed is greatly accelerated.

2.2. SGX

Intel SGX [12, 13] is a new extension based on Intel CPU architecture, which aims to ensure the trusted execution environment (TEE) [26, 27, 28] by using a remote trusted hardware. It can create a secure space called "Enclave" to support the private operations. This secure space can protect code and

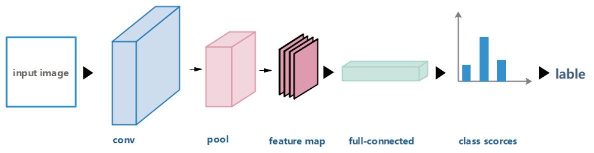


Figure 2: Flowchart of deep learning network.

data not to be tampered or monitored, so as to guarantee their integrity and confidentiality.

When the application program is created, it will generate a certificate according to its own code and data. To create an enclave, it will copy the code and data at the same time during page allocation. After initialization, it can enter the enclave execution mode. Meanwhile, the memory and address mapping protection mechanisms isolate the illegal access.

In this paper, the traditional encryption technology is used to encrypt the model parameters when uploading them to the cloud server. After receiving the model parameters, the trusted secure isolation environment module is created by using the hardware security advantages of Intel SGX at the cloud server. In the secure enclave, the model parameters are decrypted, aggregated, transformed and pruned. The utilization of SGX [29, 30] can reduce the leakage of model parameters uploaded by the client, avoid the direct contact of the curious server, and improves the security, to some extent.

2.3. Network Pruning

Deep learning suffers from the computation efficiency issue, and its complexity increases significantly with the depth. Due to the fact that over parameterization frequently occurs in deep learning, representing many redundant parameters are used to capture only tiny information from the data in training stage, one direction is to remove such parameters that has less contribution for inference so as to accelerate the training procedure. The network pruning [31, 32] is proposed to serve this purpose, by pruning the insignificant parts of the learning model to alleviate the computation cost, which possess the advantages of significantly reducing the amount of calculation and saving power consumption.

The network pruning operation can be formally expressed as follows.

$$\min\ L(D; w) + \lambda \parallel \omega \parallel_0$$

or write parameter clipping as a constrained optimization form:

$$\min\ L(D; w)\ s.t. \parallel \omega \parallel_0 \leq k$$

where D, w, λ, and k denote training dataset, model parameter, regularization factor and constraint value respectively. L_0 norm makes this problem be a combinatorial optimization problem.

At present, the existing works are proposed to address the above optimization problem. In [16], Li *et* al. proposed a method to remove the useless filters and their connections in the whole network, which can greatly reduce the computing cost while retraining to restore the near-original accuracy. Unlike pruning weights, this approach uses the filter as a unit for pruning. Besides, it has demonstrated that model accuracy can be regained by retraining the networks in a certain. Therefore, in this paper we employ this network pruning method in [16] to reduce the time consumption in model training as much as possible while maintaining the accuracy of the neural network model.

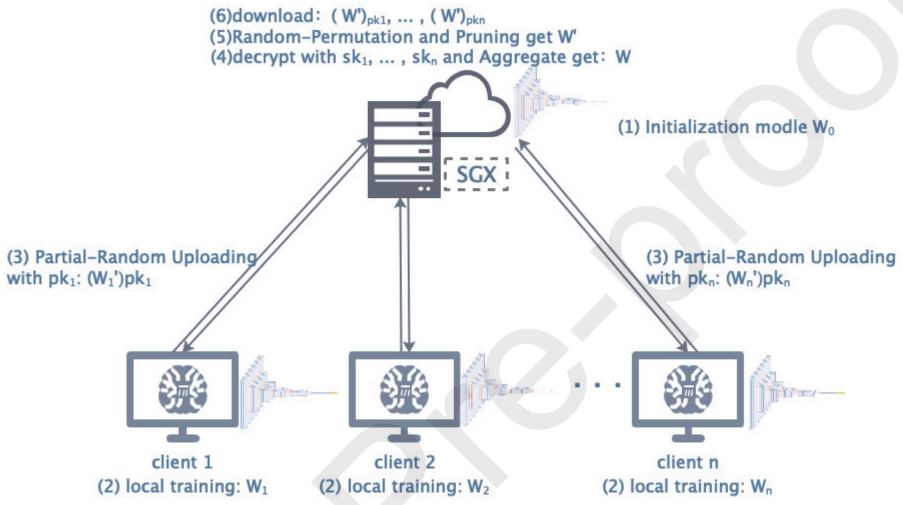

(6)download: (W')pk1, ... , (W')pkn
(5)Random–Permutation and Pruning get W'
(4)decrypt with sk1, ... , skn and Aggregate get: W
(1) Initialization modle W0
SGX
(3) Partial–Random Uploading with pk1: (W1')pk1
(3) Partial–Random Uploading with pkn: (Wn')pkn
client 1
(2) local training: W1
client 2
(2) local training: W2
client n
(2) local training: Wn

Figure 3: An overview of the privacy-preserving collaborative learning framework.

3. Privacy-preserving Collaborative Learning

In this section, we present our design of the privacy-preserving collaborative learning framework. Our design targets to prevent curious server from accessing and speculating on uploaded data and mitigate the ability of dishonest clients to steal information from other participants. In addition, we will achieve the goals of reducing the training time cost and ensuring the accuracy of neural network model.

3.1. System Model

Considering the general collaborative learning framework, the clients jointly train a model with their own datasets where stored locally. Denote W as the global model parameters, D_i as the local dataset of client i. After initializing their model parameters, they update their local models with local data to obtain W_i respectively. Then they update W_i to the server so that compute the global model as $W = \frac{1}{n}\sum_{i=1}^{n} W_i$.

3.2. Attack Model

The goal of our scheme is to protect clients' model parameters during the whole training process and to avoid any disclosure of raw data privacy under the situation of adversaries. The following two threat models are our concerns:

Honest-but-curious Server. The cloud server performs model parameter aggregation, random-permutation and pruning honestly in accordance with the protocol. The curious server has access to the model parameters uploaded by the client, so it may try to infer private information of model parameters corresponding to each client.

Honest-but-curious Participant. The client performs normal training, uploading and other operations in accordance with the protocol. One client may try to infer some useful information of an honest participant through in collusion with other clients.

According to threat models, the proposed system architecture aims to achieve the following privacy protection.

- Honest-but-curious server cannot learn any other information except to perform the necessary operations, i.e., model parameter aggregation, random-permutation and pruning.
- Honest-but-curious participants are unlikely to learn any model parameters from other participants except for normal training and upload operations allowed by the protocol.

In this paper, we assume that secure channels are used in all communications to prevent the middle attacks and trivial snooping attacks.

3.3. Our Design

Assume that each client has a secret key pair (pk_i, sk_i), and the server is a cloud platform embedded with SGX functionality.

3.3.1. Network Initialization

The server initializes CNN model parameter W_0 and sends it to all clients for training and updating in Figure 3. At the same time, it leverages SGX to create an enclave to support the legitimate part of the application program, import the program code and related data into enclave memory, and enter the enclave execution mode.

Each client respectively uploads the private keys sk_i through a secure channel to the enclave in the server. They encrypt the updated local model parameters with public keys pk_i and then upload them to the enclave. In enclave, the parameters are decrypted with their private keys and some relevant operations are then performed. Finally, the updated global parameters are encrypted with public keys pk_i and then distribute $(W')_{pk_i}$ to client respectively.

3.3.2. *Local Training*

For the local training, in each round, assuming t^{th}, the central server sends the global model parameters W^t, updated in the previous round, to all clients. The clients continue to update the model based on the local data and then obtain new model parameters L^{t+1}. In this work, we use the SGD training. Denote $Loss(D, W^t)$ as the loss function of client training under data set D and model parameter W^t. ∇l represents the gradient for $Loss(D, W^t)$. E is denoted as the local training round number, B is a batch size, α is the learning rate of local training. The detailed steps of local training is shown in Algorithm 1.

Algorithm 1 Local Training

```
Input:
    $W^t$ : Global model at round t;
    $D$ : Local dataset;
    $D_{local}$ : Client's local data $D$ split into batches of size B;
Output:
    $L^{t+1}$: Local client training update model;
1: Initialize Local model and loss function $l$;
2: $L^{t+1} \leftarrow W^t$;
3: $l \leftarrow LOSS(D, L^{t+1})$;
4: for epoch $e \in E$ do
5:     for batch $b \in D_{local}$ do
6:         $L^{t+1} \leftarrow L^{t+1} - \alpha \times \nabla l(b, L^{t+1})$;
7:     end for
8: end for
9: return $L^{t+1}$;
```

3.3.3. *Partial-Random Uploading*

In order to mitigate the information leakage on the upload model parameters against honest-but-curious server and participants, we utilize Partial-Random uploading mechanism in client uploading procedure. Selective parameter sharing is effective here, because the basis of modern neural network training relies on random gradient descent algorithm, which can be parallelized and run asynchronously. They are robust to unreliable parameter updates, competitive conditions, participant exits, and so on. By updating a small number of parameters with values obtained from other participants, each participant can avoid local minima in the process of finding the optimal parameters. Parameter sharing can be tuned to control the trade-off between the amount of exchanged information and the accuracy of resulted model.

The detailed description of partial-random uploading mechanism is shown in Algorithm 2. In this algorithm, each client sets the upload ratio to be β, and randomly selects β part of the filters to upload. For the unselected filters, we keep the status of model parameters, and then updates with new globle model.

Algorithm 2 Partial-Random Uploading

```
Input:
    $W^t$ : Global model at round t
    $L^{t+1}$: Local model at round $t+1$
    $\beta$ : The ratio of the parameters selected by the client
Output:
    $U^{t+1}$: Parameters uploaded by the client
1: for each layer $i \in\ L^{t+1}$ $or$ $W^t$ do
2:     set tensor $T^i$ as index sequence of layer $i$ filters
3:     $U_i^{t+1} = L_i^{t+1} * random(T^i, \beta) + W_i^t * (T_i - random(T^i, \beta))$
4: end for
5: return $U^{t+1}$
```

3.3.4. Server Operation

In our design, the server is not only a simple aggregator, but also a transformer and a trimmer. Algorithm 3 shows the process on server-side. Figure 4 shows the results of each operation performed by the server on the aggregate model.

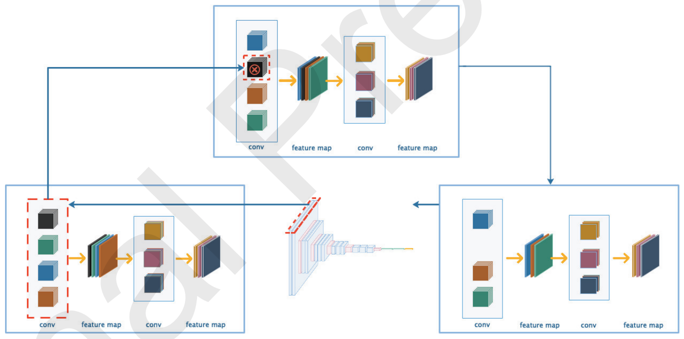


Figure 4: The illustration of server operation.

Aggregator: Its functionality is to aggregate the parameters uploaded by each client. So the global model could be aggregated as $W = \frac{1}{n}\sum_{i=1}^{n} W_i$.

Transformer: For the aggregated parameters, taking the filter as a unit, the filters of each layer are randomly permutated, so as to achieve network transformation, but the accuracy of the model is not affected, and the privacy leakage caused by client collusion can be mitigated. The detailed transformation process as follows.

Let feature map X_i be the input of the i^{th} convolution layer (denoted as F_i) of CNN model, w_i, h_i and n_i be the width, height and channel number of feature map X_i, namely $X_i \in \mathbb{R}^{w_i * h_i * n_i}$. If the number of the i^{th} convolution filter is n_{i+1}, then $F_i \in \mathbb{R}^{k_i * k_i * n_i * n_{i+1}}$. Then feature map X_{i+1} is the output of convolution layer F_i, and $X_{i+1} \in \mathbb{R}^{w_{i+1} * h_{i+1} * n_{i+1}}$. As shown in Figure 5, the column indexes of kernel matrix $f_i \in \mathbb{R}^{n_i * n_{i+1}}$ are randomly disordered, for

Algorithm 3 Server Operation

```
1: initialize $W^0$ - Network model parameters
2: for each round $t = 0, 1, 2, \ldots$ do
3:     for each client $k \in \{1, 2, ..., n\}$in parallel do
4:         $W_k^{t+1} \leftarrow ClientLocalTraining(D_k, W^t)$
5:     end for
6:     $W^{t+1} \leftarrow \frac{1}{n} \sum_{i=1}^{n} W_i^{t+1}$
7:     $W^{t+1} \leftarrow Random - Permutation(W^{t+1})$
8:     if $t \leq 5$ then
9:         $W^{t+1} \leftarrow Pruning(W^{t+1})$
10:    end if
11: end for
12: return $W^{t+1}$
```

example, $(1, 2, 3, 4) \rightarrow (2, 4, 1, 3)$. Then, in order not to affect the output of the model, the same transformation is also needed for the next layer F_{i+1}, the row indexes of kernel matrix f_{i+1} are also disordered, i.e., $f_{i+1} \in \mathbb{R}^{n_{i+1} * n_{i+2}}$. With the random permutation of a convolution layer, the next convolution layers also should be permutated. In this way, the model accuracy will not be affected. The concrete process is shown in Algorithm 4.

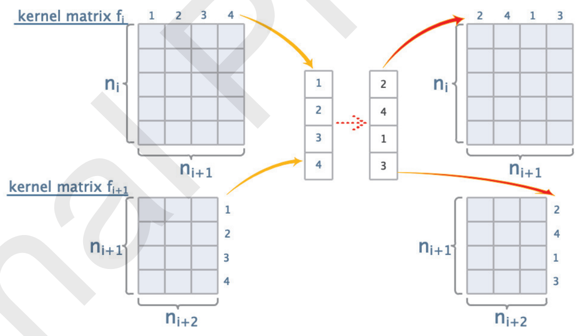


Figure 5: The illustration of Random-Permutation.

Trimmer: We continue to use neural network pruning to minimize the time and memory consumption in collaborative learning scenarios. The training of local model is accelerated by reducing the complexity of model framework so that the convergence of training is promoted. But how to reduce the model architecture is also a big challenge. Here, we adopt the $L1 - norm$ [33] and calculate the corresponding value of the filters in each layer of the aggregated model, while directly subtracting the filters with smaller norm value. This step may degrade the model accuracy. Hence, in order to make a trade-off between efficiency and accuracy, we stop pruning when the accuracy starts to drop largely. Our empirical study have shown that such a pruning strategy is effective to guarantee the model accuracy while accelerating the convergence speed.

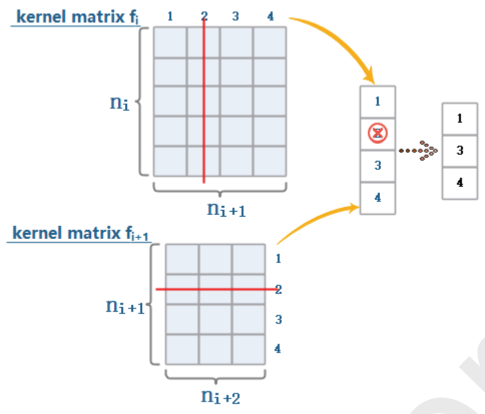


Figure 6: The illustration of pruning.

Algorithm 4 Random-Permutation

```
Input:
    $W$ : Neural network model parameters
Output:
    $RP\_W$ : Random-Permutation neural network model parameters
 1: for each layer $F_i \in W$ $or$ $RP\_W$ do
 2:   kernel matrix $f_i(\in \mathbb{R}^{n_i * n_{i+1}}) \subset W$
 3:   kernel matrix $f'_i(\in \mathbb{R}^{n_i * n_{i+1}}) \subset RP\_W$
 4:   if $i \geq 1$ then
 5:     for $k = 0 \to size(T_i) - 1$ do
 6:       $f'_i[k, :] = f_i[T_i[k], :]$
 7:     end for
 8:   end if
 9:   set tensor $T_{i+1}$ as random index sequence
10:   for each filter $F_{ij} \in F_i$ do
11:     $f'_i[:, j] = f_i[:, T_{i+1}[j]]$
12:   end for
13: end for
14: return $RP\_W$
```

Figure 6 shows an example of how to prune the network. At the i^{th} convolution layer F_i, the kernel matrix $f_i \in \mathbb{R}^{n_i * n_{i+1}}$ is used to sort the norm values calculated by filters. The filter with the lowest norm value is selected to prune. For example in Figure 6, assuming that the second filter has the lowest norm value, it will be cut off directly. For f_i, the second column is deleted. Then, in order not to affect the output of the model, in the kernel matrix $f_{i+1} \in \mathbb{R}^{n_{i+1} * n_{i+2}}$ of next layer F_{i+1}, the second row also should be deleted, so that the pruning operation of a convolution layer in the model framework is completed. Similarly, the subsequent convolutional layers can be pruned in the same way. The detailed pruning process is given in Algorithm 5. Considering the influence of pruning on model accuracy, when we chose to prune 29.8%, the accuracy will not be much affected in terms of empirical studies.

4. Security Analysis

Our scheme targets to mitigate the indirect information leakage in collaborative learning, both at the server and clients.

4.1. Security at the Server

In order to prevent the server from having direct access to the model parameters uploaded by the client, we have adopted protection measures in the uploading phase and the operation phase at the server.

Before uploading the model parameters, each client selects a pair of secret keys and sends the private key secretly to the enclave of SGX. Then the clients encrypt the model parameters with the public key and upload the model parameters to the enclave of the server. In the enclave, the server decrypts the model parameters with the private key and performs relevant operations. After that, it encrypts them with the public key and sends them to the clients. When performing operations on the server, the SGX technology is used to create a secure and trusted environment enclave. Programs and data in the enclave are protected, and the enclave's memory protection and address map protection prohibit unauthorized access. Therefore, the whole process of interaction with the server is secure. The curious server cannot contact the model parameters from the clients, which directly avoids the indirect information leakage at the server.

4.2. Security at The Client Side

Considering that some curious participants would infer the model parameters of a target client through collusion with other participants, we also need to reduce such potential attack as much as possible to ensure the privacy of collaborative learning. We will analyze the CNN model architecture, and analyze how our proposed scheme can mitigate the indirect information disclosure of clients. The security analysis from two aspects: Random-Permutation and Partial-Random Uploading.

Random-Permutation: Suppose that CNN has n convolution layers, and each layer has $n_i(i = 1, 2, ...)$ filters. At the server-side, the aggregated model parameters are randomly permutated in the filter layer. There are a total of $n_i!$ ways to sort the filters in each layer. Hence, the probability of inferring the sorting position of the original filters completely for each layer is $\frac{1}{n_i!}$. If each layer of the model is disordered, the inferring probability of the original sequence in the whole model is $\frac{1}{\prod n_i!}$.

In deep learning, the feature extraction process is visual, that is, the feature maps from each filter are different. According to visual feature map, no matter how many random permutations we make, we can still guess the original position of some filters with obvious features. Here we assume that the filter with part γ_i

will be inferred for analysis due to visualization. Then, the degree of achievable privacy protection through random permutation for each layer is $\frac{1}{[n_i(1-\gamma_i)]!}$

Algorithm 5 Pruning

```
Input:
    $W$ : Neural network model parameters
    $\theta$ : The ratio of pruning
Output:
    $P\_W$ : Neural network model parameters after pruning
 1: $P\_W = W$
 2: for each layer $F_i \in P\_W$ do
 3:     kernel matrix $f_i(\in \mathbb{R}^{n_i * n_{i+1}})$
 4:     if $i > 0$ then
 5:         for $k = 0 \to size(T_i) - 1$ do
 6:             pruning $f_i[T_i[k], :]$
 7:         end for
 8:     end if
 9:     for each filter $F_{ij} \in F_i$ do
10:         $S[j] = L1\_norm(F_{ij})$
11:     end for
12:     pruning $\theta * n_{i+1}$ filters with the smallest $l1\_norm$ values
13:     save the pruning filters index sequence as $T_{i+1}$
14: end for
15: return $P\_W$
```

Partial-Random Uploading: In order to enhance the degree of privacy protection in our scheme, meanwhile to reduce the privacy leakage caused by visualization, we have employed the Partial-Random Uploading mechanism [5] to mitigate this visualization speculation. According to the analysis in [6], there was a certain probability to infer some attribute values of data when uploading the partial gradient [5]. Here, we regard one filter as a neural, thus we use the neural network with only one neuron to analyze. For a d-dimensional input data $x = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$, with a corresponding truth label y, we assume $w_i (1 \leq i \leq d)$ are the weight parameters to be learned; and b is the bias and f is an activation function. The cost function is defined as $J(w, b, x, y) = (h_{w,b}(x) - y)^2$, which presents the distance between the predicted value $h_{w,b}(x) = f(\sum w_i x_i + b)$ and the truth value y. Then, we can get $\eta_k = \frac{\partial J(w,b,x,y)}{\partial w_k}$, $\eta = \frac{\partial J(w,b,x,y)}{\partial b}$, and $\frac{\eta_k}{\eta} = x_k$, which x_k is the component of $x = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$. x_k is completely leaked if η_k and η are shared to the cloud server. If we assume that the threshold value of uploading parameters is β, then the probability that both η_k and η are shared is $\frac{1}{\beta^2}$, which is not negligible.

Therefore, the Partial-Random Uploading mechanism can be introduced to mitigate the visualization speculation. Here we suppose that the threshold value of uploading parameters for each client is β. Because the positions of sharing updating parameters may will be different, we assume that the same position part is τ_i for each layer. Thus there will be $\beta n_i(1 - \tau_i)$ part of parameters can be confirmed by a curious client. There are still a small part of these is unpredictable, which assuming this part is ϵ_i. According to the above analysis, for the model parameter τ_i part uploaded by both parties, part of it can still be inferred due to feature visualization, that is, $\beta n_i \tau_i \gamma_i$. To simplify the analysis of Random-Permutation and Partial-Random Uploading, let's say $\beta n_i(1-\tau_i)\epsilon_i = \beta n_i \tau_i \gamma_i$. The degree of privacy protection of each layer after random partial upload is $\frac{1}{(n_i - \beta n_i(1-\gamma_i))!}$. In other words, when $(n_i - \beta n_i(1-\gamma_i)) > n_i(1-\gamma_i)$, that is $0 < \beta - \gamma_i < \beta\tau_i$. When the parameters of partial random uploading are larger than those that can be inferred by visualization, and the difference between the former two are smaller than the overlapped parameters of partial random uploading. Thus our scheme can enhance the privacy protection.

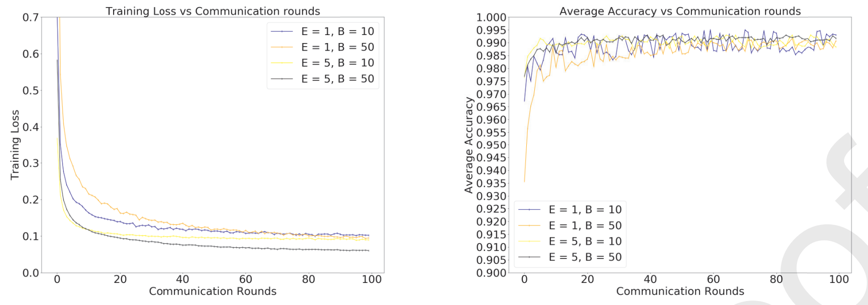
5. Experiment Evaluation

We are motivated by image classification tasks [34]. For these tasks, we first select a small enough set of data, and thoroughly study the super parameters of our algorithm. We conducted experiments on the MNIST digit dataset of recognition task [35] to validate our scheme. MNIST digit dataset [36] consists of 10 types of numbers from 0 to 9. These numbers have been dimensionally standardized and are located in the center of the image, which is a fixed size (28x28 pixels) with values from 0 to 1. There are 50,000 training examples and 10,000 testing examples, we partition MNIST data over clients according to IID settings: the data is shuffled, and then divided into two clients, each receiving 30,000 samples. The model architecture was taken from the Pytorch tutorial [37], which consists of two 5×5 convolution layers (the first with 10 filters, the second with 20 filters, each followed with 2×2 max pooling), two fully connected layers (the first with 50 units, the second with 10 units) and Relu activation, and a final softmax output layer (about 10^6 parameters in total). This model is used to test the effectiveness of applying Random-Permutation and Partial-Random Uploading techniques for simplicity. For testing the effectiveness of pruning technique, we transfer our scheme to a more complicate model, which consists of two 5×5 convolution layers (32 filters and 64 filters respectively), two 3×3 convolution layers (both 128 filters), each followed with 2×2 max pooling, two fully connected layers (128 units, 10 units) and Relu activation, and a final softmax output layer.

5.1. The Settings in Collaborative Learning

As we all know, the value of epoch and batch size in deep learning will affect the convergence and quality of training model. Similarly, in order to better verify our scheme has a good performance, we first choose the epoch and batch size values that are beneficial to our scheme through experiments.

In our experiment, we implement a two-party collaborative learning scenario. Here we choose a local learning rate of 0.01. We use E for epoch, B for batch size, and test the model accuracy when $E = 1, B = 10; E = 5, B = 10; E = 1, B = 50; E = 5, B = 50$. In Figure 7, the loss function converges faster and the convergence value is the smallest when $E = 5, B = 50$. According to the statistics of experimental data, when training reaches 99 rounds, the value of



(a) Traning Loss. (b) Average Accuracy.

Figure 7: The variation trend of Training Loss (left) and Average Accuracy (right) along with Communication rounds.

loss function converges to the lowest value 0.061, the model accuracy achieves 99.16%. Thus, in the following experiments, we select the model parameters with the best performance to validate the proposed scheme.

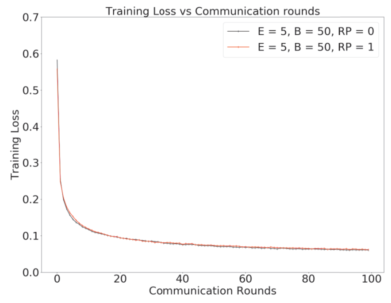
5.2. Results for Our Scheme

The proposed scheme mainly includes Random-Permutation, Partial-Random Uploading and Pruning techniques. In this part, we conduct experiments to verify the availability and accuracy of our proposed scheme.

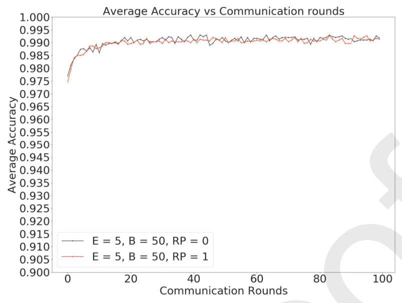
Results for Random-Permutation. In order to demonstrate the model accuracy cannot be affected by random-permutation, we evaluate the model training loss and average accuracy between learning with Random-Permutation and a general collaborative learning. Here we train a CNN model on MNIST dataset with choosing $E = 5$ and $B = 50$.

As shown in Figure 8, we record the training loss and average accuracy for learning with random-permutation and general collaborative learning. Figure 8(a) shows the trend of loss value is consistent. Figure 8(b) indicates the two curves reaches the same average accuracy when communication round is 99. Thus, Random-Permutation operation does not affect model accuracy.

Results for Partial-Random Uploading. In the multi-party federation scenario, the parameter uploading rate of each client plays an important role in influencing the final model performance. However, in order to consider the privacy issue, the parameter uploading rate should be properly selected, which can provide a certain trade-off between accuracy and privacy. Similarly, for the two-party collaborative learning scenario, this Partial-Random Uploading rate should be considered for model accuracy. Here we first conduct experiment to select the suitable uploading rate for our scheme. We set some uploading rate β between 0.1 and 1 for experiments, then select the appropriate uploading rate for trade-off between accuracy and privacy.

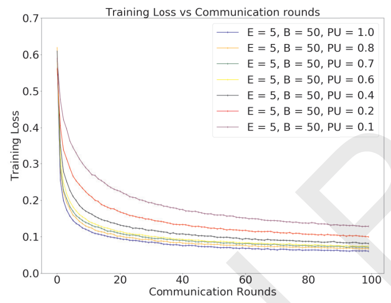


(a) Traning Loss.

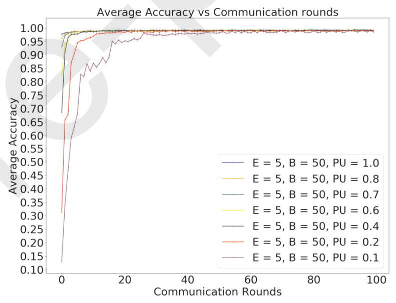


(b) Average Accuracy.

Figure 8: The variation trend of Training Loss (left) and Average Accuracy (right) when E=5, B=50 and RP=1.



(a) Traning Loss.



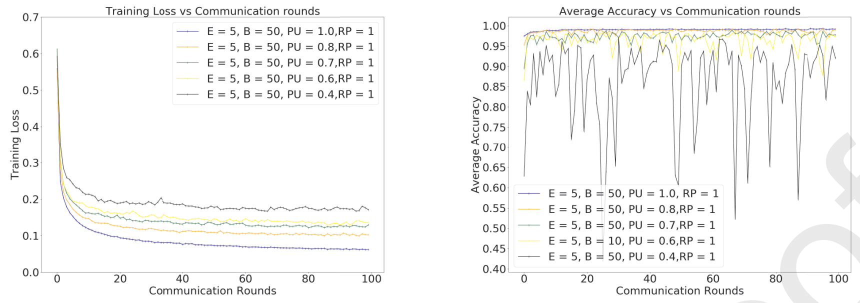
(b) Average Accuracy.

Figure 9: The variation trend of Training Loss (left) and Average Accuracy (right) when E=5, B=50 and upload rate varies.

As shown in the Figure 9(a), under the setting of $E = 5$, $B = 50$, when $\beta = 0.6 \sim 0.8$, the loss value is basically the same as $\beta = 1$. However, when $\beta < 0.6$, the loss function obviously converge slowly, and they don't reach good values. Accordingly, its corresponding test accuracy is low relatively when the communication round is 10, as illustrated in Figure 9(b), the accuracy of $\beta = 1$ has reached 98.55%, while the accuracy of $\beta = 0.4$ is only 86.10%. On the basis of this experimental results, we concluded that when $\beta = 0.6, 0.7, 0.8$, the model accuracy can achieve a good performance.

Figure 10 shows the results with adding both Partial-Random Uploading and Random-Permutation techniques. When $\beta = 0.7, 0.8$, the accuracy is stable. Considering minimizing the number of parameter uploading to mitigate privacy leakage, therefore $\beta = 0.7$ is a suitable choose.

Results for Pruning In order to speed up the training process and reduce the scale of neural network, we use pruning technique in our scheme.



(a) Traning Loss.

(b) Average Accuracy.

Figure 10: The variation trend of Training Loss (left) and Average Accuracy (right) when E=5, B=50, RP=1 and upload rate varies.

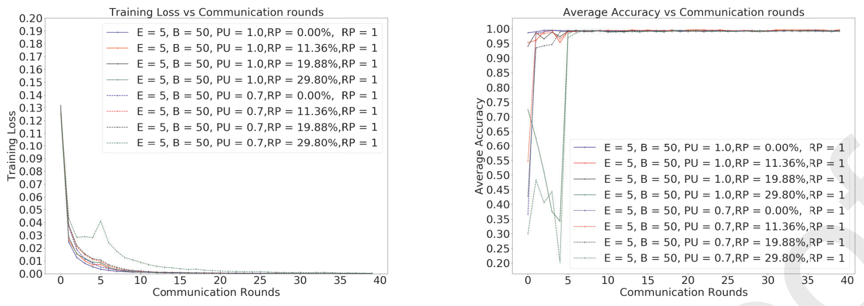
Here we utilize a more complicate model which consists of 4 convolution layers described above to test pruning results. We show pruning results in Table 1 to compare model accuracy and training time with various Upload-Rate (UR) and Prune-Rate (PR). We set $PR = 11.36\%, 19.88\%, 29.80\%$ respectively with $UR = 1.0, 0.9, 0.8, 0.7, 0.6$. We concludes that when $PR = 29.80\%$, the training time is minimum, while the model accuracy is kept. In this experiment, we can infer that a small amount of pruning may result in better model accuracy. However, too much pruning can affect the accuracy of the model, thus we only prune in the first 5 communication rounds.

Because we explained above that $PU = 0.7$ is a suitable choice, here we just shows the comparison of variation trend of training loss and average accuracy between $PU = 1$ and $PU = 0.7$ with various PR in Figure 11. The training loss in Figure 11(a) converges slowly when $PU = 0.7, RP = 29.80\%$. Besides, the model accuracy varies greatly during $0 \sim 5$ communication rounds when $RP = 29.80\%$ in Figure 11(b).

6. Related Work

Collaborative learning aims to train a global model without sharing local data. It has a significant advantage comparing to the traditional data outsourced machine learning. However, although the raw data is under control of the client, the adversary still can infer privacy information about raw data from shared updating parameters. In this section, we introduce the latest related works of collaborative learning in terms of privacy preserving techniques.

Secure Multi-party Computation (SMC) protocol allows each party to obtain the results on the premise of protecting the confidentiality of each party's input. In collaborative learning scenario, when performing aggregation, the cloud server can obtain the aggregated global model without knowing the user's input. Bonawitz *et al.* [11] outlined an approach to advancing privacy-preserving federated learning by leveraging secrete sharing and authenticated



(a) Traning Loss. (b) Average Accuracy.

Figure 11: The variation trend of Training Loss (left) and Average Accuracy (right) when E=5, B=50, RP=1 and pruning rate varies.

encryption techniques. Work [38] proposed a privacy-preserving matrix factorization scheme by combing encryption technique and garble circuit (GC). Xu *et al.* [39] put forward to HybridAlpha, an approach for privacy-preserving federated learning employing an SMC protocol based on functional encryption, and this protocol is resilient to clients dropping out. Although SMC protocols achieve privacy-preserving in collaborative learning, it requires complex computing, especially for most machine learning algorithms such as neural network, which results in implementing inefficiently.

Homomorphic Encryption (HE) is a special encryption technology, which is mainly used to perform some operation on the ciphertext without decryption, so that the result is the same as the result under the plaintext. In the study [40], Graepel *et al.* constructed a privacy-preserving neural network by utilizing fully homomorphic encryption (FHE). However, because of the high complexity, this research is difficult be applied in practice. Authors proposed a secure aggregation method with an additively homomorphic encryption to jointly learning a global model in [6]. Although HE is widely used in machine learning, it still has many shortcomings such as communication overhead and complex computation.

Differential Privacy (DP) provides an another protection strategy for data privacy. In work [5], Shokri and Shmatikov put forward to a deep learning scheme that satisfies DP. In this scheme, users add noise to the training set and parameters during each round. Besides, users uploaded the part of client model parameters to the server for aggregating, reducing the privacy leakage directly. Geyer *et al.* [41] proposed an algorithm for client sided DP privacy preserving federated optimization. Wei *et al.* [42] designed a novel framework based on DP, and give a demonstration that their method can satisfy DP under distinct protection levels by properly adapting different variances of artificial noises. However, the noise introduced by DP will lead to the loss of model accuracy. It is necessary to find a balance between accuracy and privacy, so as to improve the data privacy on the premise of ensuring model accuracy.

However these works still exist privacy leakage when consider the clients

Table 1: Results for Pruning

Upload-Rate (UR)	Prune-Rate (PR)	Accuracy	Time (10^3s)
1.0	0	0.9933	7.318
	11.36%	0.9946	6.833
	19.88%	0.9923	6.535
	29.80%	0.9920	4.223
0.9	0	0.9906	7.341
	11.36%	0.9963	6.863
	19.88%	0.9933	6.475
	29.80%	0.9916	4.229
0.8	0	0.9903	7.253
	11.36%	0.9936	6.860
	19.88%	0.9946	6.475
	29.80%	0.9916	4.229
0.7	0	0.9926	7.282
	11.36%	0.9936	6.857
	19.88%	0.9910	6.555
	29.80%	0.9933	4.223
0.6	0	0.9933	7.181
	11.36%	0.9946	6.870
	19.88%	0.9930	6.545
	29.80%	0.9920	4.205

are dishonest or clients collusion, which motivates our scheme. The proposed scheme in this paper could mitigate information leakage in this situation.

7. Conclusion

This paper proposes PPCL, a novel and effective privacy-preserving collaborative machine learning framework, which aims to prevent information leakage against dishonest clients. Specifically, we first propose a privacy-preserving network transformation method by utilizing Random-Permutation in SGX, which protects the model parameters from being inferred by adversaries. Next, we apply Partial-Random Uploading mechanism to mitigate the information conjecture through visualizations. To further enhance the efficiency, we introduce network pruning operation and employ it to accelerate the convergence of federation training. We give the formal security analysis to demonstrate that our proposed scheme can preserve privacy while ensuring the convergence and accuracy of federation aggregation. We conduct experiments to show the performance of our solution in terms of accuracy and efficiency. The experimental results show that the proposed scheme is practical with little loss of accuracy. As a basis for future research work, we continue focus on privacy-preserving collaborative learning.

References

[1] H. Chang, V. Shejwalkar, R. Shokri, A. Houmansadr, Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer.

[2] J. Konečnỳ, H. B. McMahan, D. Ramage, P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, arXiv preprint arXiv:1610.02527.

[3] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492.

[4] H. B. McMahan, E. Moore, D. Ramage, B. A. y Arcas, Federated learning of deep networks using model averaging.

[5] R. Shokri, V. Shmatikov, Privacy-preserving deep learning, in: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, 2015, pp. 1310–1321.

[6] 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 13 (5) (2017) 1333–1345.

[7] L. Melis, C. Song, E. D. Cristofaro, V. Shmatikov, Exploiting unintended feature leakage in collaborative learning, in: 2019 IEEE Symposium on Security and Privacy (SP), 2019.

[8] L. Zhu, Z. Liu, S. Han, Deep leakage from gradients, in: Advances in Neural Information Processing Systems, 2019, pp. 14747–14756.

[9] Q. Zhao, C. Zhao, S. Cui, S. Jing, Z. Chen, Privatedl: Privacy preserving collaborative deep learning against leakage from gradient sharing, International Journal of Intelligent Systems (1).

[10] J. Geiping, H. Bauermeister, H. Dröge, M. Moeller, Inverting gradients – how easy is it to break privacy in federated learning?

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

[12] V. Costan, S. Devadas, Intel sgx explained., IACR Cryptology ePrint Archive 2016 (086) (2016) 1–118.

[13] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M. L. Stillwell, et al., Scone: Secure linux containers with intel sgx, in: 12th *USENIX* Symposium on Operating Systems Design and Implementation (*OSDI* 16), 2016, pp. 689–703.

[14] L. T. Phong, T. T. Phuong, Privacy-preserving deep learning via weight transmission, IEEE Transactions on Information Forensics and Security 14 (11) (2019) 3003–3015.

[15] J. Yoon, W. Jeong, G. Lee, E. Yang, S. J. Hwang, Federated continual learning with adaptive parameter communication, arXiv: Learning.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710.

[17] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[18] X. Wang, J. Li, J. Li, H. Yan, Multilevel similarity model for high-resolution remote sensing image registration, Information Sciences 505. doi:10.1016/j.ins.2019.07.023.

[19] S.-U. Hassan, H. Waheed, N. R. Aljohani, M. Ali, S. Ventura, F. Herrera, Virtual learning environment to predict withdrawal by leveraging deep learning, International Journal of Intelligent Systems 34 (8) (2019) 1935–1952. doi:10.1002/int.22129.

[20] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 2013, pp. 6645–6649.

[21] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al., Deep speech: Scaling up end-to-end speech recognition, arXiv preprint arXiv:1412.5567.

[22] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal processing magazine 29 (6) (2012) 82–97.

[23] O. Denas, J. Taylor, Deep modeling of gene expression regulation in an erythropoiesis model, in: Representation Learning, ICML Workshop, ACM New York, USA, 2013.

[24] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, et al., The human splicing code reveals new insights into the genetic determinants of disease, Science 347 (6218) (2015) 1254806.

[25] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[26] P. D. I. S. AR, Trusted execution environments intel sgx.

[27] I. Corporation, Intel® software guard extensions (intel® sgx).
URL `https://software.intel.com/sgx`

[28] S. Brenner, C. Wulf, D. Goltzsche, N. Weichbrodt, M. Lorenz, C. Fetzer, P. Pietzuch, R. Kapitza, Securekeeper: confidential zookeeper using intel sgx, in: Proceedings of the 17th International Middleware Conference, 2016, pp. 1–13.

[29] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, J. Li, A training-integrity privacy-preserving federated learning scheme with trusted execution environment, Information Sciences 522 (2020) 69–79.

[30] T. Li, X. Li, X. Zhong, N. Jiang, C. zhi Gao, Communication-efficient outsourced privacy-preserving classification service using trusted processor, Information Sciences 505.

[31] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, W. J. Dally, Eie: efficient inference engine on compressed deep neural network, ACM SIGARCH Computer Architecture News 44 (3) (2016) 243–254.

[32] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1389–1397.

[33] M. Kloft, U. Brefeld, P. Laskov, K.-R. Müller, A. Zien, S. Sonnenburg, Efficient and accurate lp-norm multiple kernel learning, in: Advances in neural information processing systems, 2009, pp. 997–1005.

[34] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Semantic image segmentation with deep convolutional nets and fully connected crfs, arXiv preprint arXiv:1412.7062.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[36] M. Liang, Z. Li, T. Chen, J. Zeng, Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach, IEEE/ACM transactions on computational biology and bioinformatics 12 (4) (2014) 928–937.

[37] T. Contributors, Pytorch documentation, https://pytorch.org/docs/stable/index.html.

[38] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, D. Boneh, Privacy-preserving matrix factorization, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 801–812.

[39] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, H. Ludwig, Hybridalpha: An efficient approach for privacy-preserving federated learning.

[40] T. Graepel, K. Lauter, M. Naehrig, Ml confidential: Machine learning on encrypted data, in: Proceedings of the 15th international conference on Information Security and Cryptology, 2012.

[41] R. C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective.

[42] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. S. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis.