HD-FL: A Federated Learning Privacy Preservation Model Based on Homomorphic Encryption

Siyu Li[1] and Xuebin Ma[1]

[1] Wireless Networking and Mobile Computing Laboratory, Inner Mongolia University, Hohhot 010000, China
lisiyu@mail.imu.edu.cn
csmaxuebin@imu.edu.cn

Abstract. In recent years, federated learning has been widely used in the field of machine learning, and the privacy leakage problem has become more and more serious. The privacy-preserving techniques of federated learning are still immature. This paper proposes the HD-FL model for the privacy leakage problem of federated learning, which combines differential privacy with homomorphic encryption. Homomorphic encryption is used to encrypt the gradient parameters, and the global ciphertext is calculated to update the gradient parameters with differential privacy. HD-FL can prevent inference of the exchanged model information during the training process while ensuring that the generated models also have acceptable prediction accuracy. The global model update allows the client to prevent inference threats and generate highly accurate models without sacrificing privacy. Moreover, this model can be used to train various machine learning methods, which are experimentally validated with three different datasets, and the results show that this approach outperforms current solutions in terms of security and accuracy.

Keywords: Federated Learning, Homomorphic Encryption, Differential Privacy.

1 Introduction

In federated learning, the participating clients are given three different roles: inputting the raw data, building the model, and making queries. Attacks on federated learning can occur in any process of data distribution, model training, or model inference where there is private information about the users [1]. However, most researchers might ignore the protection of parameters in the federated learning process, resulting in an attacker being able to use the obtained parameters to infer sensitive information about the users. Although only the model information is shared with other clients in federated learning, the training data can be inferred using the model parameters, and additional information can be obtained after analytical reconstruction if the data structure is known. Therefore, studying privacy-preserving models under federated learning is of outstanding significance for protecting users' private information and is among the top priorities in terms of privacy protection.

Most of the current federated learning privacy protection methods use homomorphic encryption or differential privacy respectively, but it is not safe to use only one of them. When differential privacy is used alone, the client may add too much noise to reduce the accuracy of the original model, while homomorphic encryption alone is also subject to joint inference attacks on the client side.

In this paper, we combine homomorphic encryption with differential privacy to design a federated learning privacy-preserving framework HD-FL to protect privacy information during federated learning by using the ciphertext computation capability of homomorphic encryption and the privacy-preserving capability of differential privacy techniques. The work in this paper can be summarized as follows.

(1) The HD-FL federated learning privacy-preserving model uses homomorphic cryptography combined with differential privacy mechanisms to prevent joint inference attacks by multiple parties.

(2) In the model training process, an average aggregation strategy is proposed to provide privacy protection for federated learning while improving model accuracy.

The rest of this paper is organized as follows. Section 2 introduces the work related to privacy protection for federated learning. Section 3 introduces some relevant fundamentals. Section 4 describes the HD-FL process in detail. Section 5 provides an analytical summary of the experimental results; finally, Section 6 is the conclusion of this paper.

2 Related Work

Although federated learning privacy preservation has received extensive attention from researchers, existing research approaches are limited to the use of homomorphic cryptography or differential privacy techniques, which may have security issues when used alone. Therefore, we focus on the current state of research on federated learning privacy preservation in this part.

As one of the main methods for privacy-preserving federated learning, gradient sparsification is used to satisfy privacy requirements by filtering gradients. Zhu et al. [2] proposed to filter the gradient parameters of the model participating in the training client. Some of the gradient parameters that pass the screening are set to 0 and uploaded to the server for aggregation for training. It is verified that this method can effectively stop the attacks of malicious parties. However, in practice for the modification of fixed scale gradient parameters, the malicious party can choose to attack a specified feature, and there are privacy protection limitations. Therefore, more researchers have focused on how to provide more secure privacy protection for federation learning through other privacy-preserving techniques. Federated learning based on privacy-preserving techniques is divided into two types. One type is the perturbation class [3]. Abadi [4] proposed a deep learning based differential privacy DPSGD algorithm in 2016, which introduces differential privacy-preserving techniques in the training process of deep learning. Applying DPSGD to federation learning, the client-side model parameters are added to the noise. The client-side privacy is protected but the problem of noise accumulation occurs during server aggregation, which affects the accuracy of the global model. Geyer et al. [5] proposed a client-side differential privacy-preserving joint optimization algorithm. In each round of communication, the clients are randomly selected to compute the gradient two-paradigm and the median is chosen as the criterion to normalize the gradient parameters to the shared gradient. Noise is added to the shared gradient. This approach reduces the nuisance gradient by normalization but does not provide strict privacy constraints on the gradient parameters of the clients.

Another research approach is based on homomorphic encryption. For example, Nikolaenko et al. [6-7] proposed secure computational protocols that encrypt neural networks, one based on linear branching procedures and the other relying on neural networks. Although there is no decrease in accuracy, the encrypted neural network has a huge computational overhead, resulting in significant time consumption. Aono et al. [8] proposed a privacy-preserving deep learning system and referenced it in the federation learning process to encrypt the client's gradient parameters during training. The client is randomly selected to upload the parameter ciphertext and asynchronous gradient parameter updates are used. This method guarantees the security of local data. However, compared to the federated average algorithm [9], the asynchronous gradient update method has poor convergence because some nodes with lower speeds provide outdated or even incorrect gradient directions. Zhang et al. [10] proposed a federated learning network model that supports data privacy preservation. By encrypting the model weight parameters, the ciphertext of the parameters is uploaded and the global model parameters are computationally updated at the server. Although the use of a homomorphic encryption algorithm can resist some plaintext attacks and does not leak any important data and model information from the client side to the server side. However, it is still possible to simulate a model similar to the original model through multiple update processes and face joint inference attacks from multiple clients.

In summary, it can be found that the existing federated learning privacy-preserving models mainly focus on using privacy-preserving techniques alone, while models that use a combination of privacy-preserving techniques are missing. Therefore, this paper combines the homomorphic encryption-based federated learning privacy-preserving model with differential privacy. Compared with other models, the combined use of privacy-preserving techniques can avoid inference attacks. In addition, it can also improve the model accuracy of updates.

3 PRELIMINARIES

3.1 Homomorphic Encryption

Homomorphic encryption is a secure encryption method proposed by Rivest et al. [11] in 1978, which encrypts the initial data to generate ciphertext data through an encryption function and performs direct computation on the ciphertext data without decryption, and the result is obtained is the same as the result obtained during plaintext processing after decryption. Compared with traditional encryption algorithms, homomorphic encryption does not require frequent encryption and decryption operations between the server and the client, and the overhead of communication and computing resources can be reduced to a great extent. The data is transmitted in ciphertext, and no one else knows the content of the data so some operations can be performed on the encrypted private data in an untrusted environment to prevent illegal theft or tampering with the data. Homomorphic encryption allows analysis and computation of encrypted data, which improves the availability and security of the data and provides the necessary privacy protections for the data, while also protecting the private information.

Let M denote the plaintext space and C denote the ciphertext space, the encryption algorithm can be called homomorphic when the conditions of Equation (1) are satisfied as follows.

$$\forall m_1, m_2 \in M, Dec_{sk}\left(Enc_{pk}(m_1) \odot Enc_{pk}(m_2)\right) = m_1 \odot m_2 \tag{1}$$

where $\odot$ denotes some type of operator in the ciphertext space, the operation $\odot$ of the ciphertext $Enc_{pk}(m_1)$ and $Enc_{pk}(m_2)$ is equal to or can be calculated directly from the plaintext m_1 and m_2 by the same operation without intermediate decryption.

3.2 Differential Privacy

Differential privacy is a privacy protection technology based on data disturbance. It disturbs sensitive data by adding noise but keeps the properties of data to be published unchanged. Differential privacy is defined as follows.

ε-Differential Privacy [12]: If a random algorithm A satisfies ε-differential privacy, for all inputs d,d' differing in at most one attribute value, and for all sets of possible outputs $S(S \in range(A)$, we have

$$Pr[A(D) \in S] \leq e^{\varepsilon} \times Pr[A(D') \in S] \tag{2}$$

where ε is a privacy parameter, which indicates the degree of privacy protection A smaller ε indicates a greater degree of perturbation to the output and a higher degree of privacy protection [13], and the privacy budget is proportional to the availability.

Laplace Mechanism: For any function f:$D \to R$, if algorithm A satisfies Equation (3) then algorithm A satisfies ε-differential privacy.

$$A(D) = f(D) + \left(Lap_1\left(\frac{\Delta f}{\varepsilon}\right), \ldots, Lap_D\left(\frac{\Delta f}{\varepsilon}\right)\right) \tag{3}$$

the above equation $Lap_i(\frac{\Delta f}{\varepsilon})$ Laplacian variables are independently and uniformly distributed samples with the scale parameter of $\frac{\Delta f}{\varepsilon}$. In this paper, the Laplace mechanism is used to add noise to the training process.

In addition, differential privacy has the following key features that we can take advantage of when implementing HD-FL.

Theorem 1. Post-processing [14]. Any calculations of the differential privacy mechanism output will not increase privacy loss. In this paper, since the parameters of the generative model guarantee the differential privacy of the data, it is safe to generate the data after the training process.

Theorem 2. Parallel composability [15]. If each disjoint subset of database D_i satisfies ε_i-differential privacy under algorithm M_i, then $D = \sum_{i=1}^{n} D_i$ also satisfies ε-differential privacy under algorithm M_i.

4 HD-FL

In order to guarantee the privacy security of clients participating in federated learning, this paper designs a homomorphic encryption-based privacy-preserving model for federated learning (homomorphic encryption differential privacy federated learning, HD-FL), and the model consists of two parts.

We add the differential privacy to the client side to obtain the gradient model parameters with differential privacy properties, encrypt them and upload them to the server side, which updates them using the optimized average aggregation model. The general flow is shown in Figure 1.

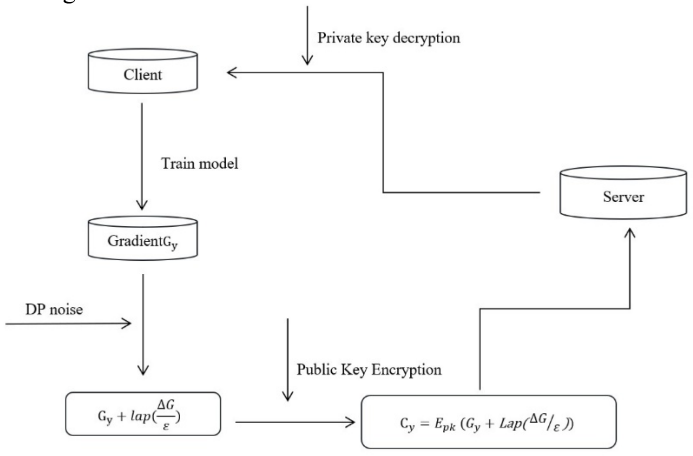

Private key decryption
Client
Train model
Server
GradientG_y
DP noise
Public Key Encryption
$G_y + lap(\frac{\Delta G}{\varepsilon})$
$C_y = E_{pk}\,(G_y + Lap(^{\Delta G}/_{\varepsilon}))$

Fig. 1. General Flow.

4.1 Client Side

The clients involved in training first determine the clients that perform the initialization of the model, and the clients that perform the initialization build the local neural network model and generate the gradient parameters and bias term parameters in the model using the random number algorithm. Then, the parameters are determined using the Paillier homomorphic encryption algorithm, and the public key pk and private key sk, other parameters p, q, and γ are generated according to the definition of the Paillier, and the public and private keys are assigned to all clients and kept secret from the server side. Then this client encrypts the gradient parameters of the neural network model using the private key of the homomorphic encryption Paillier algorithm and uploads the gradient ciphertext parameters to the server; the server broadcasts the gradient ciphertext parameters uploaded by this client down to other clients participating in training; the clients participating in training receive the gradient ciphertext parameters and decrypt the gradient ciphertext parameters using the private key; the decrypted parameters are updated. The decrypted parameters are loaded into their local model after updating the weights to complete the initialization process.

Select two p, q two large elements and calculate $n = p \times q$, $\gamma = lcm(p-1, q-1)$, where $lcm()$ represents the least common multiple. Define $l(u) = \frac{u-1}{n}$ function, and randomly select an integer g less than n^2. g and n satisfy $\gcd(l(g^{\gamma} mod\, n^2), n) = 1$, where gcd represents the greatest common divisor, and the greatest common divisor is sought to ensure that the two primes are of equal length. The public key pk is (n, g) and the private key sk is γ.

The global gradient parameters downloaded by the clients involved in training are decrypted and updated model weights are loaded onto the local model, and the local data are trained with the global model. After completing one learning, in order to protect the privacy of different clients, the client sets the privacy budget ε. In each epoch for client y, y receives the initial parameters from the server side and then decrypts the updated weights, trains the local client to obtain the local gradient parameters, and for privacy purposes, client y adds differential privacy noise to the gradient parameters conforming to the Laplace distribution, assuming that the size of the gradient parameters is between 0-1 and the sensitivity Δf is set to 1.

$$G_y = G_y + Lap\left(\frac{\Delta f}{\varepsilon}\right) \tag{4}$$

The gradient parameters are cropped to avoid gradient explosion according to the algorithm of noise addition followed by cropping proposed by Lin [16], and the gradient parametrization can be within the threshold value even if the noise added is too large. Finally, the gradient is encrypted by the key generated in the key generation stage, given that the plaintext m is an integer less than n, an integer r less than n^2 and mutually prime with n^2 is randomly selected, and the ciphertext is calculated using the public key pk.

$$C_y = g^{G_y} \times \gamma^n \, mod\, n^2 \tag{5}$$

Finally, the encrypted local gradient C_y is sent to the server for a global encrypted gradient parameter update.

The key steps of the client side algorithm are given in Algorithm 1. The global gradient parameters are obtained from the server and then decrypted to update the local model weight parameters (lines 3-4); the model gradient is computed in each iteration (lines 5-10); Laplace noise is added to the gradient and the gradient is cropped (lines 11-12), and finally, the noisy gradient is encrypted and sent to the server side (line 13).

Algorithm 1: Federated Learning Privacy-Preserving Client-side Algorithm

```
Update on the client side(k, [[$G_t$]]):
(Clients k, ∀k=1, 2, …, k execute in parallel)
1: Get [[$C_{t+1}$]] from the server, decrypt [[$G_{t+1}$]] to get the latest
2: Update model parameters locally $W_{t+1} \leftarrow W_t - lG_{t+1}$
3: for $i \in S$ do
4:   batches ←divide the local dataset $D_k$ into batches of size $M$
5:   get local parameters from previous iteration
6:     for $b \in \frac{n_k}{M}$ do
7:       compute batch gradients $G_{t+1}^{(b)}$
8:     end for
9: end for
10: Get local parameter updates $G_{t+1}^{(k)}$=$G_{B,S}^{(k)}$
```

```
11: $G_{t+1}^{(k)} \leftarrow G_{t+1}^{(k)} + Lap\left(\frac{\Delta G}{\varepsilon}\right)$
12: $G_{t+1}^{(k)} \leftarrow G_{t+1}^{(k)} / max\ (1, \frac{||G_{t+1}^{(k)}||_2}{C})$
13: Homomorphic encryption on $G_{t+1}^{(k)}$ to get $\left[\left[G_{t+1}^{(k)}\right]\right], \left[\left[G_{t+1}^{(k)}\right]\right]$ send to server
```

After each client receives the updated global gradient, the decryption operation is performed, and the decryption calculation operation is as follows.

$$G_{global} = \frac{l\left(C_{global}{}^{\gamma}\ mod\ n^2\right)}{l(g^{\gamma}\ mod\ n^2)}\ mod\ n \tag{6}$$

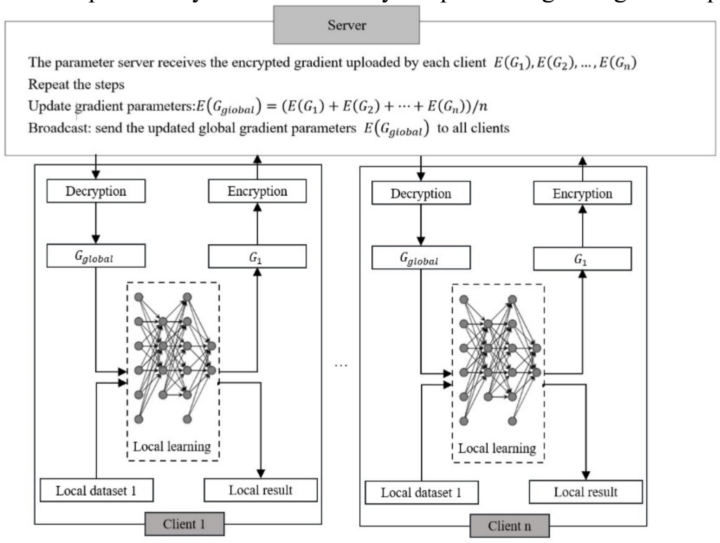
The model weight parameters are updated according to the decrypted gradient parameters Equation (7), where l is the hyperparameter learning rate.

$$w \leftarrow w - l\ G_{global} \tag{7}$$

Finally, the weight parameters are loaded on the local model.

4.2 Server Side

After receiving the encrypted gradients from all users, the server side performs the following gradient averaging aggregation operation, as shown in Figure 2, without decryption on the server side. The server can calculate the average of the gradient ciphertext parameters uploaded by the client directly to update the global gradient parameters.



Server

The parameter server receives the encrypted gradient uploaded by each client $E(G_1), E(G_2), \ldots, E(G_n)$
Repeat the steps
Update gradient parameters: $E(G_{global}) = (E(G_1) + E(G_2) + \cdots + E(G_n))/n$
Broadcast: send the updated global gradient parameters $E(G_{global})$ to all clients

Fig. 2. Secure Aggregation Process.

Finally, the server side sends down the updated ciphertext gradient parameters to the client, and the key algorithm is shown in Algorithm 2.

Algorithm 2: Federated Learning Privacy-Preserving Server-side Algorithm

```
Execute on the server side:
1: for $t = 1,2,3 \ldots$ do
2:     the client determines the number of $C_t$(Determine the set of randomly picked clients)
3:     for Client $k \in C_t$, do
```

```
4:     local update parameters： $[[G_{t+1}^{(k)}]]$ ←Client update(k,$[[G_t]]$)
5:     send the updated parameter $[[G_{t+1}^{(k)}]]$to the server side
6:   **end for**
7:   the server side aggregates the received parameters, $[[G_{t+1}]] \leftarrow \sum_{k=1}^{k} \frac{1}{n} [[G_{t+1}^{(k)}]]$
     (These are all ciphertext operations, and for readability, the mathematical notation for addition and multiplication is reused here to represent calculations based on homomorphic encryption)
8:   the server side broadcasts the aggregated parameter $[[G_{t+1}]]$ to all clients
9: **end for**
```

4.3 Security Analysis

According to the above training process of the homomorphic encryption-based federated learning privacy-preserving model, on the server side. The intermediate data obtained by the client and the server are shown in Table 1. As can be seen from the table, during the training process of the federated learning privacy-preserving model, each client obtains the global model gradient parameters by decryption and cannot obtain other parameters of the client, such as model weight parameters W, gradient G, prediction results and loss values. meanwhile, the server obtains the local model gradient of the client by encrypting the ciphertext parameter $Enc(G_y)$ and the model gradient parameter $Enc(G_{global})$ computed in ciphertext, while decryption of the parameter data is not possible without the decryption private key on the server side. During the federated learning training process, the server-side and each client transmits the ciphertext of the gradient parameters, and even if the attacker intercepts the gradient information, it is impossible to decrypt the ciphertext.

Table 1. Client and server data information.

Name	Client	Server
Intermediate data	$Enc(G_{global})$	$Enc(G_{global})$
	Prediction results	$Enc(G_{y,1})$
	Loss	$Enc(G_{y,2})$
	W_i	$\cdots$
	G_i	$Enc(G_{y,n})$

On the client side, considering the reality that inference attacks may jeopardize client information security when there are fewer clients or jointly between clients. Therefore, to solve this problem, differential privacy is utilized to provide privacy guarantees. It is proved that the gradient perturbation $G'_y = G_y + Lap\left(\frac{\Delta G}{\varepsilon}\right)$ under Laplace mechanism is consistent with ε-differential privacy.

Proof: Let λ denote the noise λ~Lap$\left(\frac{\Delta G}{\varepsilon}\right)$ added to the gradient G_y that conforms to the Laplace distribution, and D and D' be the neighboring data sets.

$$Pr[G'_y(D) = t] = Pr[G_y(D) + \lambda = t]$$

$$= Pr[\lambda = t - G_y(D)] = \frac{\varepsilon}{2\Delta G} e^{\left(\frac{-\varepsilon|t-G_y(D)|}{\Delta G}\right)} \tag{8}$$

Thus,

$$\frac{\Pr[G'_y(D)=t]}{\Pr[G'_y(D')=t]}=\frac{\frac{\varepsilon}{2\Delta G}e^{\left(\frac{-\varepsilon|t-G_y(D)|}{\Delta G}\right)}}{\frac{\varepsilon}{2\Delta G}e^{\left(\frac{-\varepsilon|t-G_y(D')|}{\Delta G}\right)}}\leq e^{\varepsilon} \tag{9}$$

It is shown that gradient perturbation is consistent with ε-differential privacy, and the proposed method in this paper can tolerate collusion among clients, but other clients cannot infer any useful information, thus protecting client privacy.

The number of encryptions and decryptions is independent of the data set and related to the model parameters in the model training process. The number of setting clients proposed in this paper is n; the number of iterations is $epoch$; the gradient parameter of the model is g; the encryption is E; the decryption is D; the time complexity of the encryption is $O(epoch \times n \times g \times E)$ and similar to encryption, the time complexity of decryption is $O(epoch \times n \times g \times D)$. If the size of a ciphertext is c, the communication overhead is $T(epoch \times n \times g \times c)$, while the communication of the federated learning training process has both upload and downlink processes, so the communication overhead is twice that of other machine learning methods.

5 Experimental results

5.1 Experimental Design

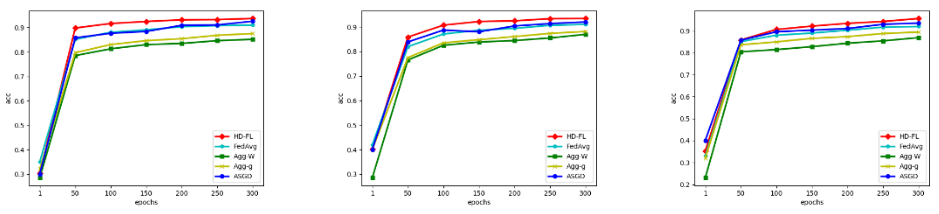
For objective and realistic data, the experimental parameters are set identically and three datasets are used to compare the experiments. We split the dataset and distribute it to each client. In this way, each client has only a portion of the dataset, thus ensuring that it is difficult for a single client to obtain a high-precision classification model from local data. We set up three sets of federated learning experiments with different numbers of clients. The first dataset is MNIST, consisting of handwritten digital images; the second dataset is the Fashion-MNIST dataset, containing images of goods; and the last dataset is the colored items dataset CIFAR-10. All of the above datasets can be found on the website. Since CNN has better image feature extraction capability, the client chooses CNN convolutional neural network as the training model. The experiments pass through the first convolution layer, then through the second maximum pooling layer, then through the third convolution layer, the fourth maximum pooling layer, and finally through the fifth fully connected layer. The model hyperparameters are set as shown in Table 2.

Table 2. Hyperparameter setting.

Options	Kernel	Strides	Padding	Activation function
Conv2D	5	1	2	Relu
Conv2D	5	1	2	Relu
Optimizer	GradientDescent(Learning rate=0.01)			
Batchsize	32			

5.2 Experimental analysis

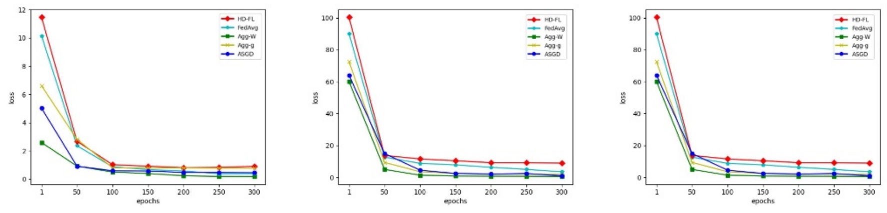
This subsection shows the performance of our experiments. Figure 3 shows the experimental performance of HD-FL compared with other models. The number of epoch set is 300, selecting 10 clients involved in training randomly. All clients achieved over ninety percent accuracy on all three datasets. The ASGD model is closer to HD-FL. The accuracy is improved than other models, but there is a delay between different clients, and the local model of each client is probably not the latest global model. Each client obeys the server to obtain the global model, resulting in inconsistent and unstable global models obtained by each client, as shown in Figure 3(b). The pace of each client varies greatly, which affects the convergence.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Fig. 3. Accuracy of model.

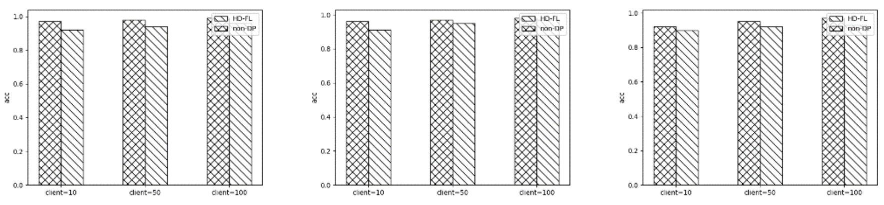
Figure 4 shows the loss values for HD-FL, where the HD-FL fold is smoother and has a lower loss value in the same case. The magnification shows that the fluctuations of ASGD are more pronounced in the case of similar loss values due to the larger initial epoch value of the overall fold change. The size of the batchsize setting affects the fluctuations of the curve, and the overall trend is downward. The loss curve is decreasing, indicating that a gradient descent process is underway and the learning rate is appropriate. The curve gradually becomes smoother, the loss variation does not drop to 10 in all three datasets below, between 1 epoch and 50 epoch. The loss of HD-FL is lower. The global parameter values of the aggregation have a cumulative effect and the loss is correspondingly higher in the experiments.



(a) MNIST (b) Fashion-MNIST (c) CIFAR-10

Fig. 4. Loss of model.

As shown in Figure 5, this paper adds perturbation comparison experiments with different number of clients on three datasets. The number of clients is set to 10, 50 and 100. The experiments compare the difference between HD-FL model and noiseless non-DP model by setting the privacy budget ε = 10. The results show that the accuracy can be improved by about 2%-5% from 10 to 100 clients. As the number of clients increases the accuracy rises, and even with the addition of differential privacy noise, the accuracy can be improved by 1%-5%. Without differential privacy, the accuracy is 95% for a number of clients of 10. The accuracy of the HD-FL model achieves similar accuracy only for a number of clients of 100. Suggesting the same performance because essentially noise-free federal learning can be achieved while increasing the number of clients in training. Compared to the MNIST dataset and the Fashion-MNIST dataset, the CIFAR-10 dataset requires more clients to achieve similar accuracy. This suggests that for complex tasks with larger neural network models, more local data and more clients are required to obtain better data performance for perturbations.



(a)MNIST (b)Fashion-MNIST (c)CIFAR-10

Fig. 5. Impact of model.

In a word, it can be found that HD-FL is more accurate than the other models through comparison experiments. HD-FL outperforms other models because it reduces the gradient information loss and most of the noise from multiple clients is eliminated due to the symmetry of the Laplace mechanism. Therefore, our model with differential privacy can maintain high accuracy.

6 Conclusion

This paper proposes a homomorphic encryption-based privacy-preserving model for federated learning named HD-FL, which combines homomorphic encryption with differential privacy. HD-FL can both protect sensitive data of model parameters from being leaked by inference and meet the needs of models in machine learning tasks in various domains. However, there may not be only one server in practical applications, but multiple servers working together is also a new research hotspot. Therefore, applying HD-FL to multiple servers that do not collude with each other is also one of the main works in the future.

Acknowledgment
This paper is supported by the Science and Technology Program of Inner Mongolia Autonomous Region (Grant No. 2019GG116).

References

1. Yang Q, Liu Y, Chen T, et al. Federated Machine Learning: Concept and Applications[J]. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2):1-19.

2. Zhu L, Liu Z, Han S. Deep leakage from gradients[J]. Advances in Neural Information Processing Systems, 2019, 32.
3. Lin Y, Bao L Y, Li Z M, et al. Differential privacy protection over deep learning: An investigation of its impacted factors[J]. Computers & Security, 2020, 99: 102061.
4. Abadi M, Chu A, Goodfellow I, et al. Deep learning with differential privacy[C]//Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016: 308-318.
5. Geyer R C, Klein T, Nabi M. Differentially private federated learning: A client level perspective[J]. arXiv preprint arXiv:1712.07557, 2017.
6. Khazbak Y, Tan T, Cao G. MLGuard: Mitigating poisoning attacks in privacy preserving distributed collaborative learning[C]//2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, 2020: 1-9.
7. Bhowmick A, Duchi J, Freudiger J, et al. Protection against reconstruction and its applications in private federated learning[J]. arXiv preprint arXiv:1812.00984, 2018.
8. Aono Y, Hayashi T, Wang L, et al. Privacy-preserving deep learning via additively homomorphic encryption[J]. IEEE Transactions on Information Forensics and Security, 2017, 13(5): 1333-1345.
9. McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
10. Zhang Z, Fu Y, Gao T G. Research on Federated Deep Neural Network Model Supporting Data Privacy Protection [J/OL]. Journal of Automation: 1-14.DOI:10.16383/jaas.c200236.
11. Rivest R L, Shamir A, Adleman L M. A method for obtaining digital signatures and public key cryptosystems[M]. Routledge, 2019.
12. Dwork C. A firm foundation for private data analysis[J]. Communications of the ACM, 2011, 54(1): 86-95.
13. Cheu A, Smith A, Ullman J, et al. Distributed differential privacy via shuffling[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Cham, 2019: 375-403.
14. Chuan X Z, Yi S, De G W. Federated learning with Gaussian differential privacy[C]//Proceedings of the 2020 2nd International Conference on Robotics, Intelligent Control and Artificial Intelligence. 2020: 296-301.
15. Bu Z, Dong J, Long Q, et al. Deep learning with gaussian differential privacy[J]. Harvard data science review, 2020, 2020(23).
16. Lin Y, Bao L Y, Li Z M, et al. Differential privacy protection over deep learning: An investigation of its impacted factors[J]. Computers & Security, 2020, 99: 102061.
17. Dwork C, Kenthapadi K, McSherry F, et al. Our data, ourselves: Privacy via distributed noise generation[C].
18. Dwork C, McSherry F, Nissim K, et al. Calibrating noise to sensitivity in private data analysis[C]. Theory of cryptography conference. Springer, Berlin, Heidelberg, 2006: 265-284.
19. McSherry F, Talwar K. Mechanism design via differential privacy[C]//48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07). IEEE, 2007: 94-103.
20. Dwork C, Roth A. The algorithmic foundations of differential privacy[J]. Found. Trends Theor. Comput. Sci., 2014, 9(3-4): 211-407.
21. Lecun Y, Cortes C. The MNIST database of handwritten digits. [Online]. Available: http://yann.lecun.com/exdb/mnist/, retrieved on March 28, 2020.
22. Alex Krizhevsky. The CIFAR-10 dataset. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html, retrieved on March 28, 2020.1.
23. Han xiao, The Fashion-MNIST dataset. Available: https://github.com/zalandoresearch/fashion-mnist.