[1]Haoxuan Sun

FedCGS: Leverage Feature Compression and Conditional GAN for Privacy Protection and Efficient Communication in Federated Learning

Abstract: - Federated learning is a distributed machine learning technique that can train models from decentralized data, collaboratively building central models and protecting data privacy by passing partial model updates rather than raw data in each iteration of model learning. However, recent work has shown that the information during the exchange between the server and the client is still vulnerable to gradient-based privacy attacks, and traditional privacy protection methods can incur significant costs (such as communication consumption and performance evaluation). Therefore, we need an algorithm that can protect data privacy while reducing communication costs and performance losses. Here, we propose a federated learning framework FedCGS that leverages conditional generation adversarial networks and feature compression in order to achieve privacy protection while maintaining competitive model performance and effectively improving communication efficiency. We divided the local network of each client into feature extractor and public classifier, and kept the extractor local to protect privacy, and sent classifier and client generator to the server for aggregation and update through singular value decomposition, so as to improve the performance of the local network and efficient communication. A lot of experimental data prove that FedCGS can improve the performance of local model better than traditional federation learning, and improve the communication efficiency better than common conditional generation adversarial network. Therefore, we believe that FedCGS has better federal learning potential

Keywords: Federated learning, conditional generation adversarial network, singular value decomposition, privacy security, communication compression

I. INTRODUCTION

With the development of deep learning, many fields have achieved great success. People think that with enough training data, we will eventually get a satisfactory model. In reality, however, a lot of data resides in different clients that are reluctant to share them due to privacy security concerns, which makes it impossible for us to train many aspects of data to fit our deep learning models. To solve these problems, McMahan [1] proposed Federated Learning (FL). It can train the final model by transmitting the parameters required for model updates without sharing all data and protecting privacy.

However, in a recent study, we found that the number of parameter exchanges between the server and client during FL training is frequent, resulting in increasing communication overhead, especially as the local model scales up. Therefore, communication has become one of the main bottlenecks of FL [19, 21]. In addition, as FL continues to advance, studies have found that the classical joint average approach (FedAvg [1]) is still vulnerable to gradient-based privacy attacks [6], that is, FL's privacy protection is no longer sufficient to support today's technology. Subsequently, homomorphic encryption (HE) [4,7] and differential privacy (DP)[5, 27] have been incorporated into FL. HE provides high-level security by encrypting the information exchanged between clients, but its high computing and communication costs lead to its excessive limitations. In addition, DP results in a loss of precision, which degrades the performance of the model. Later, FedCG [2] proposed the use of conditional generation adversarial networks cGAN [11] to ensure privacy security without excessive loss of computing cost and accuracy. However, communication issues remain and the larger scale of local models still cannot be solved.

We propose a new FL method called FedCGS in this paper, which uses cGAN to achieve a higher level of privacy protection. The client local model is divided into feature extractors and classifiers, and the model is trained using cGAN generators and discriminators so that each client generator matches the feature extractor. Finally, moving the generator and classifier to the server instead of the original feature extractor avoids direct contact with the local model. Compared with traditional methods such as FedEx and FedProx [3] that directly transmit local model gradients, it can better resist gradient privacy attacks. At the same time, the singular values of the local generator and classifier are decomposed by the dynamic gradient compression technique, and finally, the server is restored, and the global generator and classifier are aggregated by knowledge distillation [8]. Compared with traditional FL methods, FedCGS still has strong model performance. Compared with FedCG, FedCGS overcomes the high cost of

[1] *Corresponding author: Haoxuan Sun

large-scale local model communication and greatly improves communication efficiency. The novelty of this paper is that it combines cGAN with dynamic gradient compression. A lot of experiments show that compared with traditional FL methods, this method can obtain effective performance and efficient communication. Privacy analysis proves that FedCGS has high protection ability against gradient privacy attacks. Therefore, whether facing large or small models, training purposes can be safely and efficiently achieved, which is the direct advantage of this method

II. RELATED WORK

Federated Learning

Federated learning is an approach to distributed machine learning in which the training of a model is distributed across multiple clients rather than centralized in a single data center. Each client trains the model locally and then sends updated parameters to a central server for integration, which helps protect user privacy. However, as FL has grown, a number of serious challenges have emerged. First, traditional FL methods have been shown to be prone to deep leakage [6, 13, 14] and model inversion [15]. Secondly, when the client data scale becomes large, the required communication overhead cannot be controlled. Therefore, a large number of technologies have been adopted to solve the privacy security and communication efficiency problems of FL. Examples include homomorphic encryption (HE) and differential privacy (DP). However, the balance between privacy security, model performance, loss of accuracy, and communication efficiency is not well controlled.

GAN In Federated Learning

GAN addresses several important issues in FL: One is privacy protection [23], and a major challenge in FL is how to share model updates without compromising user privacy. With GAN, generators generate synthetic data on local devices instead of sharing raw data. This approach avoids the risk of transferring sensitive data to a central server [9, 12], thereby increasing the level of user privacy protection. The second is data scarcity, in FL, participants often have only a small amount of data, which can lead to models being poorly trained on some devices. Using GAN can solve the problem of data scarcity by generating synthetic data on local devices. This generated data can increase the number of local training samples, thereby improving the performance and generalization ability of the model. The data distribution is not uniform, FL involves multiple devices or nodes, and these devices may have different data distribution. Using GAN can help to adjust the model to different data distributions, thereby improving the applicability and performance of the model. Fourth, the model performance is improved [22, 24]. The adversarial training between the GAN generator and discriminator can help improve the model performance. By training the generator on a local device and the discriminator on a central server, the model can be promoted for efficient parameter updating and improvement while protecting data privacy, thus enhancing the expressiveness and robustness of the model.

SVD In Federated Learning

In FL, singular value decomposition (SVD) research results mainly focus on two aspects: First, it can reduce the communication overhead [26], because FL involves multiple clients, each of which may have a large amount of data. In a traditional federated learning setup, participants need to send updated model parameters to a central server for aggregation, which leads to significant communication overhead. Using singular value decomposition, the model parameters can be decomposed and only a small number of parameters are sent or compressed, thus reducing the communication overhead. Second, SVD can be privacy-friendly because it can help model training without sharing raw data [25]. By shredding data on local devices and then sharing only partial or decomposed parameters, you reduce the risk of data breaches and improve privacy protection. To sum up, the two main functions of SVD are the same as the FedCGS method I studied. When we face the problem of a large number of clients and a large amount of data, the combination of SVD and cGAN can effectively achieve the purpose of reducing communication costs and improving privacy security.

III. ALGORITHM STEP

Client Update

In this work, we first set up a central server and N client clients with individual data sets $\{\chi_1, \chi_2, \chi_3, \ldots \chi_n\}$in accordance with traditional Federated learning (FL).These private data sets are taken from different sample spaces.

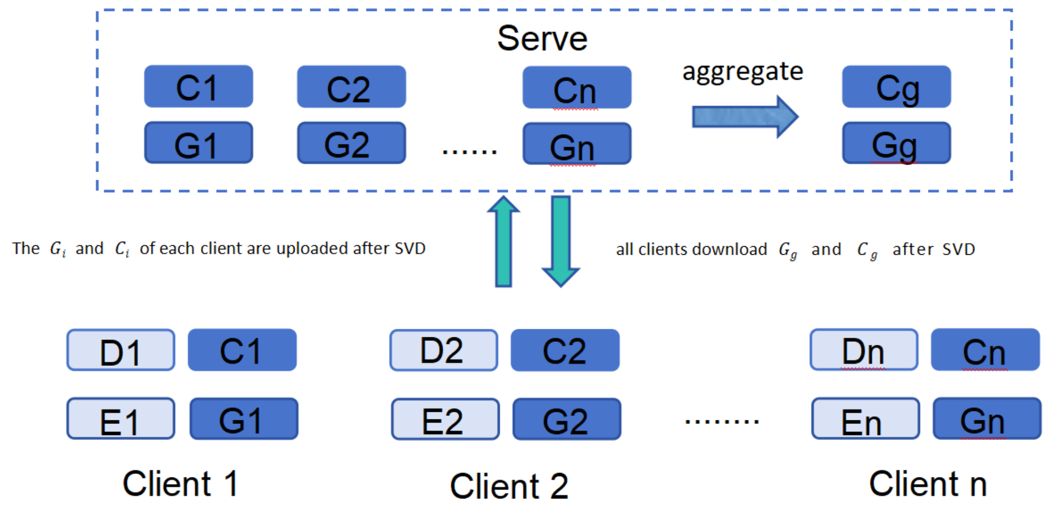
Serve
C1
G1
C2
G2
......
Cn
Gn
aggregate
Cg
Gg
The G_i and C_i of each client are uploaded after SVD
all clients download G_g and C_g after SVD
D1
C1
E1
G1
Client 1
D2
C2
E2
G2
Client 2
....
Dn
Cn
En
Gn
Client n

Figure 1: **Overview of FedCGS.**

We define the parameter of some client i as θ_{E_i,C_i}:=[θ_{E_i}; θ_{C_i}]. It shows that each client i consists of a feature extractor E_i: $\chi_i \rightarrow \check{R}^d$ and a common classifier C_i: $\check{R}^d \rightarrow \check{R}^c$, where d is the feature dimension and c is the class number. Not only that, but each client has a conditional GAN (cGAN), which consists of a generator G_i:Z$\rightarrow \check{R}^d$ and a discriminator D_i: $\check{R}^d \rightarrow \mu$, where Z satisfies the Gaussian distribution and μ is some scalar in the range [0,1]. Each client trains the generator G_i to approximate the extractor E_i and sends G_i（z,y） to the server instead of E_i（x|y）, thus ensuring data privacy.

As shown in Figure 1, in each round of FL, each client i first completes local training and uploals its G_i and C_i to the server, while E_i and D_i remain local. In the process of uploading, we apply the singular value decomposition method to dynamic gradient compression of G_i and C_i. The server then aggregates and uses knowledge distillation to maintain a global generator G_g and a global classifier C_g. Then, the server SVD G_g and C_g and pass it to the client to download the aggregation, replace the original local model, and proceed to the next training.Experiments show that the algorithm can guarantee performance, privacy security and efficient communication at the same time. The following will be a specific analysis of the client and server.

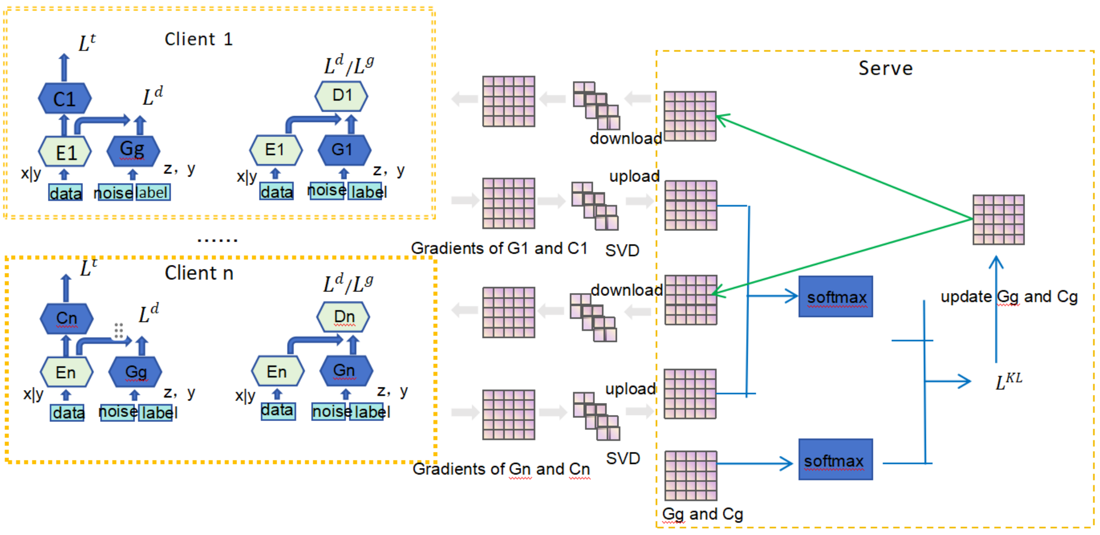

Figure 2: **FedCGS Algorithm flow.**

As can be seen from the Figure 2 above, we define the classification network loss function

$$L^{cs}=\grave{E}_{x,y\sim\chi_i}\Omega(C_i(E_i(x|y,;\theta_{E_i});\theta_{C_i}), y) \qquad (1)$$

Where Ω is the cross entropy function. Client i optimizes by freezing the global generator and minimizing the mean square error loss, receiving the parameters of the global generator to its local extractor. As follows

$$L^{mse}=\grave{E}_{x,y\sim\chi_i}\ \grave{E}_{z\sim Z}||\ E_i(x|y,;\theta_{E_i})\ -\ G_g(z,y;\theta_{G_g})||^2 \qquad (2)$$

Hence our task loss L, which is a combination of L^{cs} and L^{mse}:

$$L^t = L^{cs} + \varepsilon L^{mse} \qquad (3)$$

Algorithm 1 Client Update

```
Require: Clients' datasets $\{\chi_i\}_{i=1}^{n}$, clients' extractors, classifiers, generators,
    and discriminators: $\{E_i(\cdot;\theta_{Ei}), C_i(\cdot;\theta_{Ci}), G_i(\cdot;\theta_{Gi}), D_i(\cdot;\theta_{Di})\}$; global gen-
    erator $G_g(\cdot;\theta_{Gg})$ and global classifier $C_g(\cdot;\theta_{Cg})$; learning rate $\eta_1, \eta_2$; local
    training epoch $T$.
 1: Clients receive $\theta_{\mathbf{G}g}$ and $\theta_{\mathbf{C}g}$ from the server.
 2: for each client $i = 1, \ldots, N$ in parallel do
 3:     $\theta_{\mathbf{C}i} \leftarrow \theta_{\mathbf{C}g}$
 4:     for $t = 1$ to $T$ do
 5:         for all $(x, y) \in clients' datasets$ do
 6:             Sample $z$ from $\mathcal{N}(0, 1)$
 7:             $\theta_{\mathbf{E_i},\mathbf{C_i}} \leftarrow \theta_{\mathbf{E_i},\mathbf{C_i}} - \eta_1 \nabla_{\theta_{E_i,C_i}} \ell^t(\mathbf{x}, y, z)$
 8:         end for
 9:     end for
10:     $\theta_{\mathbf{G}i} \leftarrow \theta_{\mathbf{G}g}$
11:     for $t = 1$ to $T$ do
12:         for all $(x, y) \in clients' datasets$ do
13:             Sample $z$ from $\mathcal{N}(0, 1)$
14:             $\theta_{\mathbf{D_i}} \leftarrow \theta_{\mathbf{D_i}} - \eta_2 \nabla_{\theta_{D_i}} \ell^d(\mathbf{x}, y, z)$
15:             $\theta_{\mathbf{G_i}} \leftarrow \theta_{\mathbf{G_i}} - \eta_2 \nabla_{\theta_{G_i}} \ell^g(\mathbf{y}, z)$
16:         end for
17:     end for
18:     $\theta_{\mathbf{C_i}}, \theta_{\mathbf{G_i}i} \rightarrow \mathbf{U_i} \sum_{\mathbf{i}} \mathbf{V_i}$
19:     Clients upload $\mathbf{U_i} \sum_{\mathbf{i}} \mathbf{V_i}$ to the server.
20: end for
```

We set the ε to slowly increase from 0 to 1 to adjust the balance between the two loss terms.

When generating network updates, each client approximates the output of its local generator to its local extractor. It trains the generator by freezing the parameters of the extractor, and then performs cGAN. Specifically, it sends a small batch of training date (x, y) to E_i to obtain the feature representation ý. Then generates the same batch size Gaussian noise z randomly, and sends (z, y) to the generator G_i to generate the estimated feature representation ÿ. Finally, y and ÿ are sent to the discriminator D_i, the discriminator loss L^d and energy loss L^e are calculated, and the generator θ_{G_i} and discriminator θ_{D_i} are optimized by minimizing these two losses alternately.

$$L^d=\grave{E}_{x,y\sim\chi_i}\grave{E}_{z\sim Z}[\log(1-D_i(E_i(x|y,;\theta_{E_i});\theta_{D_i})) + \log D_i(G_i(z,y;\theta_{G_i});\theta_{D_i}\ )] \qquad (4)$$

$$L^g = \grave{E}_{x,y\sim\chi_i}\grave{E}_{z\sim Z}\log(1-D_i(\ G_i(x|y,;\ \theta_{G_i});\theta_{D_i}) \qquad (5)$$

After the local training is completed, each client uses the SVD method to decompose the gradients of the generator and classifier into smaller matrices and then upload them. The server then aggregate the previous decomposition matrix to reconstruct the local gradient.The aggregated global generator and classifier are further decomposed and distributed to clients for model updates. Specifically, we represent the gradient $g_i \in R^{P\times Q}$ as a matrix with P rows and Q columns (assuming P≥Q). It can be decomposed into the product of three matrices, $g_i \approx U_i \sum_i V_i$, where $U_i \in R^{P\times K}$, $\sum_i \in R^{K\times K}$, $V_i \in R^{K\times Q}$, K is the number of singular values preserved.

If the value of K is PK+K^2+KQ<PQ, then the size of the upload and download gradients can be reduced. In addition, we express the singular values of g_i as $[\alpha_1, \alpha_2, \alpha_3 \ldots\ldots \alpha_Q]$. We use the energy threshold T to determine how many singular values to retain so that we can control the approximation error, as follows:

$$\min_K \frac{\sum_{i=1}^{K} \alpha_i^2}{\sum_{i=1}^{Q} \alpha_i^2} \qquad (6)$$

Then, we propose a dynamic gradient approximation strategy, which can make the model converge better. The function is as follows:

$$T(t) = T_{start} + (T_{end} - T_{start})t,\ t \in [0, 1] \qquad (7)$$

T_{start} and T_{end} are two hyperparameters that can control the start and end values of T. It helps to learn more accurate generator θ_{G_i} and classifier θ_{C_i} models

Server Aggregation

When the server receives the generator $\{G_i\}_{i=1}^{n}$ and classifier $\{C_i\}_{i=1}^{n}$ uploaded by the client, it initializes the parameters of global generator θ_{G_g} and global classifier θ_{C_g} by weighted average method. The small batch training data (z, y) is then sent to all generators using a method of knowledge distillation (KD)[10, 20], where label y is obtained by sampling from the uniform distribution $U(0, c)$, and noise z is obtained from the Gaussian distribution N(0,1).We define two class probability distributions $P_a(y, z)$ and $P_b(y, z)$, where $P_a(y, z)$ is from the client classifier and $P_b(y, z)$) is from the global classifier. Finally,the server optimizes the global classifier θ_{C_g} and generator θ_{G_g} by minimizing the KL deviation of the two class probabilities.

$$P_a(y,z) = \mathrm{softmax}\left(\sum_{i=1}^{n} \frac{|\chi_i|}{\sum_{i=1}^{n} |\chi_i|} C_i(G_i(y, z;\ \theta_{G_i});\ \theta_{C_i})\right) \qquad (8)$$

$$P_b(y,z) = \mathrm{softmax}(C_g(G_g(y, z;\ \theta_{G_g});\ \theta_{C_g})) \qquad (9)$$

$$L^{KL} = \grave{E}_{y\sim U} \grave{E}_{z\sim Z} KL(P_a(y,z), P_b(y,z)) \qquad (10)$$

```
Algorithm 2 Server Aggregation
Require: Clients' datasets $\{\chi_i\}_{i=1}^{n}$, clients' classifiers, generators $\{C_i(\cdot;\theta_{Ci}), G_i(\cdot;\theta_{Gi})\}$; global generator $G_g(\cdot;\theta_{Gg})$ and global classifier $C_g(\cdot;\theta_{Cg})$; learning rate $\eta_3$; training iteration $T$, sample batch size $B$.
1: Serve receives $\{U_i \sum_i V_i\}_{i=1}^{n}$ from all clients.
2: $\{\theta_{Ci}, \theta_{Gi}\}_{i=1}^{n} \leftarrow \{U_i \sum_i V_i\}_{i=1}^{n}$
3: $\{\theta_{Cg}, \theta_{Gg}\} = \sum_{i=1}^{n} \frac{|\chi_i|}{\sum_{i=1}^{n} |\chi_i|} \{\theta_{Ci}, \theta_{Gi}\}$
4: for $t = 1$ to $T$ do
5:     Sample $(z, y)$, where z $\in N(0, 1), y \in U(0, c)$
6:     $\{\theta_{Cg}, \theta_{Gg}\} = \{\theta_{Cg}, \theta_{Gg}\} - \eta_3 \nabla_{\theta_{Cg}, \theta_{Gg}} \ell^{KL}(\mathbf{y}, z)$
7: end for
8: $\{\theta_{Cg}, \theta_{Gg}\} \rightarrow \{U_g \sum_g V_g\}$
9: Server sends $\{U_g \sum_g V_g\}$ to all clients.
```

After the server aggregation update is completed, we use SVD method to decompose the global generator θ_{G_g} and global classifier θ_{C_g} and send them to all clients.

IV. EXPERIMENTS

In this section, we compared FedCGS with FedAvg, FedProx, and FedCG in terms of performance and communication.

Experiment Settings

Model architecture: Using LeNet5 [16], the first 2 convolutional layers of LeNet are used as feature extractors, and the last 3 fully connected layers are used as common classifiers. The size and step size of the convolution kernel in the cGAN are then adjusted to match the output dimensions of the extractor and generator

Dataset: We evaluated client model performance on 2 datasets, namely FMNIST [17] and CIFAR10[18].FMNIST can simulate independent homodistribution (IID).

Baselines: We selected 3 of the existing FL methods as baselines. It includes FedAvg [1], FedProx [3], and FedCG [2], among which FedAvg is the best known FL method. FedProx is also a widely used FL algorithm that allows for more standardized local model training. FedCG algorithm is the use of cGAN and FL fusion, can be a good privacy protection but can not solve the communication problem.

Configuration: We performed 50 global communication rounds and 15 local epochs, with 16 small batches per local epoch. All of them use Adam optimizer with learning rate of 2e-4 and weight attenuation of 1e-4.For FedProx, we chose the best factor in the range {0.001, 0.01, 0.1, 1}. For FMNIST and CIFAR10, let each client randomly extract 2000 images as a local training set.

Results

Performance Evaluation: We first evaluate the performance of FedCGS by comparing the average customer accuracy of four FL algorithms on two datasets. Figure 3 shows the results. As shown in the figure, FedCG achieved the best accuracy in five of the eight different comparisons between the two datasets. The FedCGS algorithm also closely followed the other three FL methods in the FMNIST dataset and slightly underperformed in the CIFAR dataset, but in the worst case, the FedCG algorithm was only 14.1% less accurate than the best case on average. In FMNIST, it is less than 10%.

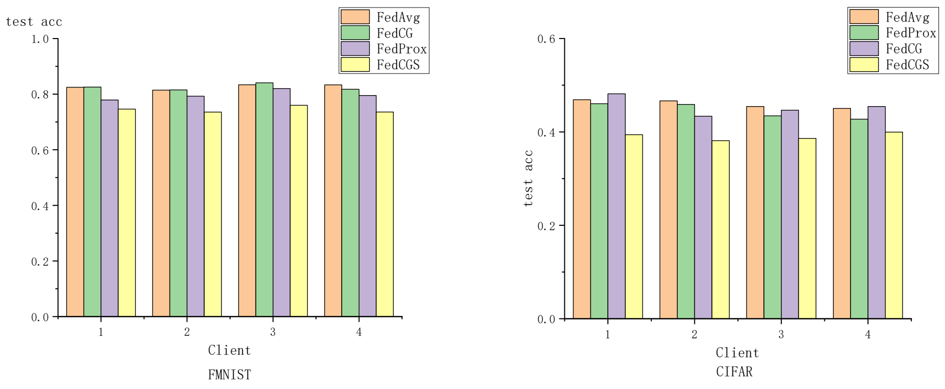

Figure 3: Test set accuracy representation of four FL methods on different clients in two scenarios

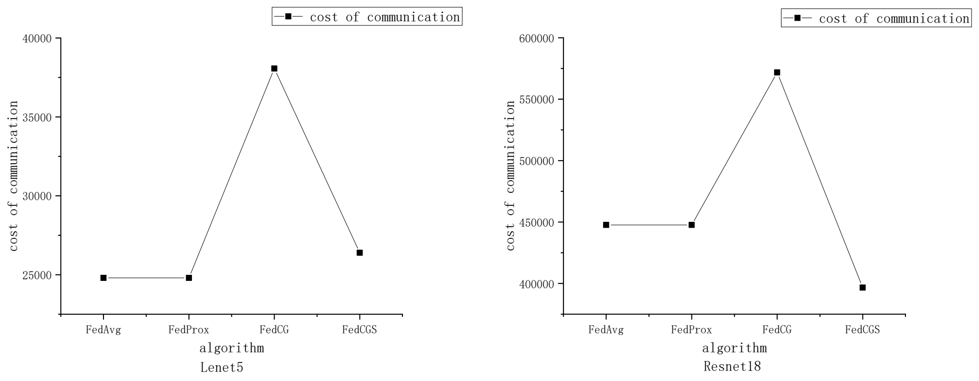

Figure 4: Communication time of four federated learning algorithms

Communication evaluation: We evaluated the communication consumption of four FL methods in Lenet5 and Resnet18 communication networks, including client training time, communication time, and server aggregation time, with little difference in performance evaluation. Considering the conduct of the simulation experiment, we assume that the data transmission rate during communication is 100bit/s, and the results are shown in Figure 4. The figure shows that in the Lenet5 network, the communication cost of the FedCG algorithm is significantly greater than that of the other three algorithms, and the communication capability of FedCGS after compression has been significantly improved. The optimization is about 30.7%. In Resnet18, the communication capability of FedCGS is significantly less than that of the other three methods, and the communication cost is greatly reduced by 30.6% compared with FedCG.

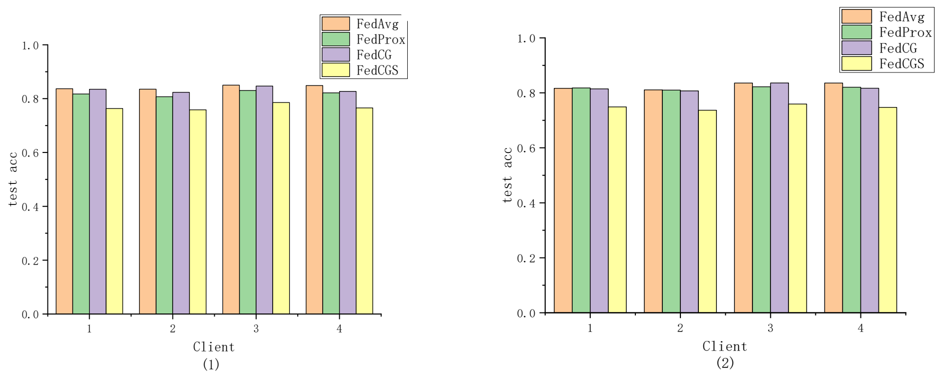

Figure 5: Performance of the four FL methods against heterogeneous data

Isomerism evaluation: Finally, we evaluated the performance of FedCGS in the face of heterogeneous data, thus proving that it has reliable research value and can withstand a variety of external factors. I used unequal random sampling on the FMNIST dataset to select training sets for each client of the four FL methods, to simulate data heterogeneity. The results are shown in Figure 5. The degree of inequality and heterogeneity in (2) is greater than that in (1). The results show that when faced with heterogeneous data, the accuracy of FedCGS's test set is about 7.2% less than that of the other three methods. Therefore, we believe that FedCGS can deal with some heterogeneous data

It can be seen from the experimental results that FedCGS greatly improves the communication efficiency within the acceptable performance loss range, and can maintain the accuracy comparable to the traditional FL method in the face of heterogeneous data. Therefore, FedCGS is a new FL method with research value.

V. CONCLUSION

In this paper, we propose a new FL method called FedCGS, which uses cGAN and SVD to protect data privacy security and ensure excellent model performance and can greatly improve the efficiency of communication, which solves the security and communication problems encountered in federated learning. In FedCGS, each client's local network is divided into a feature extractor and a common classifier, and the extractor is left local to protect privacy. Improve the performance of the client's local network by sharing the client's generator with the server using cGAN. A large number of experiments show that FedCGS has good privacy protection ability, {Phong, 2018 #9}can obtain competitive model performance and efficient communication efficiency. After my analysis, the future FedCGS has a good prospect of federal learning, and we can continue to study in this direction.

REFERENCES

[1] McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In Artificial intelligence and statistics, 1273–1282. PMLR.

[2] Wu, Yz,; Kang, Y.; Luo, Jh.; He, Yq.; and Yang, Q. 2022. FedCG: Leverage Conditional GAN for Protecting Privacy and Maintaining Competitive Performance in Federated Learning. arxiv preprint arXiv:2111.08211

[3] Li, T,; Kumar Sahu, A,; Zaheer, M,; Sanjabi, M.; Talwalkar, A,; Smith, V. 2020 Federated Optimization in Heterogeneous Networks. arxiv preprint arXiv:1812.06127

[4] Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S.; et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security, 13(5): 1333–1345.

[5] Kang, J., Li, Y., Zhao, Y., & Li, Z. (2021). Federated Learning with Differential Privacy: Algorithms and Implementation. IEEE Transactions on Information Forensics and Security, 16, 2713-2728

[6] Geiping, J.; Bauermeister, H.; Droge, H.; and Moeller, M. 2020. Inverting Gradients How easy is it to break privacy in federated learning? In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., Advances in Neural Information Processing Systems, volume 33, 16937–16947. Curran Associates, Inc.

[7] Wang, T., Liu, Q., Wang, H., & Wang, W. (2021). Federated Learning with Homomorphic Encryption: Algorithms and Implementation. IEEE Transactions on Information Forensics and Security, 16, 2882-2897.

[8] Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In NIPS 2014 Deep Learning Workshop.

[9] Hitaj, B.; Ateniese, G.; and Perez-Cruz, F. 2017. Deep models under the GAN: information leakage from collaborative deep learning. In Proceedings of the 2017 ACMSIGSAC Conference on Computer and Communications Security, 603–618.

[10] Wu, C., Wu, F., Lingjuan, L., Huang, Y. & Xie, X. Communication-efficient federated learning via knowledge distillation. https://doi.org/10.5281/zenodo.6383473 (2022).

[11] Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

[12] Wang, Z.; Song, M.; Zhang, Z.; Song, Y.; Wang, Q.; and Qi, H. 2019b. Beyond inferring class representatives: Userlevel privacy leakage from federated learning. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications, 2512–2520. IEEE

[13] Zhao, B.; Mopuri, K. R.; and Bilen, H. 2020. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610.

[14] Zhu, L.; and Han, S. 2020. Deep leakage from gradients. In Federated Learning, 17–31. Springer.

[15] Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 1322–1333.

[16] LeCun, Y.; et al. 2015. LeNet-5, convolutional neural networks. URL: http://yann. lecun. com/exdb/lenet, 20(5): 14.

[17] Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.

[18] Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

[19] Nori, M. K., Yun, S. & Kim, I.-M. Fast federated learning by balancingcommunication trade-offs. IEEE Trans. Commun. 69, 5168–5182 (2021).

[20] Seo, H., Park, J., Oh, S., Bennis, M. & Kim, S.-L. Federated knowledgedistillation. Preprint at arXiv:2011.02367 (2020).

[21] Rothchild, D. et al. Fetchsgd: communication-efficient federated learning withsketching. In ICML, 8253–8265 (PMLR, 2020).

[22] Li, Y., Kairouz, P., Chen, H., & Sankar, L. (2021). On the Convergence Theory of Federated Learning with Generative Models. arXiv preprint arXiv:2110.11468.

[23] Jeong, Y., & Park, S. (2021). Federated Generative Adversarial Learning with Privacy-Preserving Gradient Aggregation. arXiv preprint arXiv:2110.01484.

[24] Sun, J.,Huang, K.,Lyu, Y., & Gong, Z. (2021).Secure and Communication-Efficient Federated Learning with Federated Generative Adversarial Networks. arXiv preprint arXiv:2108.07707.

[25] Yao, S., Li, Y., Kang, J., Huang, Y., & Zhao, Y. (2021). Federated Singular Value Decomposition with Unbiased Global Updates. arXiv preprint arXiv:2102.05900.

[26] Agarwal, N., Goel, T., & Phan, N. (2021). Adaptive Dimensionality Reduction for Secure and Communication-Efficient Federated Learning.

[27] Aono, Y., Hayashi, T., Wang, H., & Yamamoto, R. (2021). Privacy-preserving federated learning with differential privacy: algorithms and performance analysis. IEEE Transactions on Information Forensics and Security, 16, 2923-2938