Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6596925, 16 pages
https://doi.org/10.1155/2022/6596925

Research Article

A Data Augmentation Method for Vertical Federated Learning

JianFei Zhang and YuChen Jiang

Changchun University of Science and Technology, Changchun 130000, China

Correspondence should be addressed to JianFei Zhang; jfzhang@cust.edu.cn

Received 29 September 2021; Accepted 7 December 2021; Published 24 January 2022

Academic Editor: Amr Tolba

Federated learning (FL) enables various organizations to jointly train one single model without revealing their private data to each other. The FL can be classified as horizontal federated learning (HFL) and vertical federated learning (VFL) according to the distribution of overlap samples and overlap features in the dataset. VFL allows various organizations to share machine learning based on the overlap samples, each one of which has the same identity. However, VFL suffers from insufficient number of overlap samples among all participants. Hence, the shortage of overlap data results in a worse performance of the global model. In this article, we propose a data augmentation method, FedDA, which is based on the generative adversarial network (GAN) to increase the number of training data. We generate more overlap data by learning the features of finite overlap data and many locally existing nonoverlap data, which expand the availability for training the overlap dataset. A series of experiments were executed on both MNIST and CIFAR-10. The results show that FedDA can efficiently utilize nonoverlap samples to enhance the effect of the data augmentation. It can generate high-quality overlap samples and expand the set of overlap samples. Thus, when the VFL is short of overlap samples, FedDA can provide abundant training data to improve the performance of the VFL model.

1. Introduction

Machine learning is used to explore the hidden information from a large volume of existing data, and obviously, it is tedious that those data are from a single participant. Diversified data from different institutions are need to be used for model training. In the early years, the parties were involved in the joint machine learning carried out by gathering data on a trusted server. The data privacy protection, however, has been gradually realized, and the corresponding legal provisions have been formulated. Since privacy data are hardly obtained, various institutions and organizations have become isolated data islands.

In order to resolve the above problems, Google proposed an innovative machine learning framework called federated learning in 2016 [1]. It effectively helps multiple organizations to train a global model with privacy data without violating users' privacy protection, data security, and government regulations. Since then, as an efficient approach, federated learning has become a hot research topic, and it is applied in many fields, such as finance [2], healthcare [3], urban computing [4], Internet of Things [5], and blockchain [6].

Generally, federated learning is classified into horizontal federated learning and vertical federated learning. Horizontal federated learning is usually applied to the scenarios where the datasets of participants have nearly the same feature space but different sample identity spaces. Vertical federated learning, on the other hand, is used in the scenarios where the datasets of participants have nearly the same sample identity space but different feature spaces. Overlap data refers to the data of common users in different institutions. Normally, the amount of overlap data is not very large. Lots of research focus on how to establish a vertical federated learning model. However, they usually can only use the overlap data between participants. When the amount of overlap data between multiple organizations is scarce, performing vertical federated learning will undoubtedly produce a terrible model effect. In order to improve the effect of VFL, we hope to enhance the overlap data by increasing the number of available samples.

In summary, our contributions are as follows:

(1) We design a novel federated data augmentation method for vertical federated learning, namely, FedDA, to expand the number of available samples

(2) We proposed to use adversarial generative networks for data augmentation in vertical federated learning

(3) We conducted a range of experiments on FedDA to prove its effectiveness and studied the quality of generated data by FedDA under different data distributions on two different typical datasets of MNIST and CIFAR-10

2. Related Work

Currently, the major research of VFL is mainly focused on modifying the traditional machine learning method to the federated learning mode and optimizing the existing federated learning algorithm. Hardy et al. use entity analysis [7] and homomorphic encryption [8] to learn vertically partitioned data and provide encryption [9]. Yang et al. proposed a vertical federated logistic regression implementation method based on the centralized federated learning framework. This method collects and calculates the encryption gradient of the participants through the server and assists the participants to update the parameters [10]. Yang et al. believe that it is difficult to find a third-party assistance model that can be trusted by both parties in real life. Besides, the third party will increase the risk of data leakage and the overall complexity of the system. Therefore, they believe that a third party is unnecessary for federated learning framework, and they implement vertical federated logistic regression under the framework of distributed federated learning [11]. For the traditional boosting tree model, Liu et al. proposed a random forest implementation method based on the vertical federated learning framework. In the process of joint modeling, each participant is regarded as a tree, and the structure of each tree will be stored in the central server and individual data holders. The privacy of data is guaranteed by each data holder on holding decentralized node information that matches its own features. This random forest-based federated learning method effectively reduces the communication frequency of each tree during prediction and improves the communication efficiency to a certain extent [12]. Cheng et al. proposed a decentralized vertical federated learning framework based on gradient boosting decision trees. Compared to other federated boosting tree algorithms, it has no loss on accuracy [13]. Li et al. study a practical federated environment with relaxed privacy constraints. In this environment, a dishonest party might obtain some information about the other parties' data, but it is still impossible for the dishonest party to derive the actual raw data of other parties. It is worth noting that their approach can significantly improve the predictive accuracy and achieve comparable accuracy to the original GBDT with the data from all parties [14].

As a popular generative model, the generative adversarial network (GAN) is widely used in many fields [15]. Therefore, more and more researchers have begun to study the application of GAN to federated learning. The team of Rajagopal and Nirmala proposed a novel collaborative imaginary deep learning architecture called Federated AI Imagination, which lets a team of two or more come together to imagine and envision ideas and synergies well with each other's likes [16]. In order to improve the performance of GAN in the target task, it is essential to collect as many images as possible from different sources. Rasouli et al. proposes a method for training a GAN across distributed sources of nonindependent-and-identically distributed data sources subject to communication and privacy [17]. It also provides a further theoretical basis and improved experiments based on Federated AI Imagination. Due to various factors, the images collected by different types of devices may have distinctive deviations. Fan et al. proposed a GAN learning scheme based on a federated learning framework and studied training in a federal environment [18]. This method attempts to distribute generative models to different customers and then merges the models trained in each customer into a unified and general model in the center. It helps solve the problem of data limitation in actual situations. Although the GAN is directly applied in many federated learning solutions to boost the performance of GANs in target tasks, we utilize the GAN to design a data augmentation model to enhance the training effects of federated learning.

3. The Proposed Approach

3.1. Problem Definition. Although in VFL we can align samples according to the identity and privacy protection policies, it is hard to match every sample between participants. Figure 1 shows the relation of samples among different participants. Without loss of generality, we assume that there are two participants, party A and party B.

Datasets of party A and party B can be denoted as $D^A \coloneqq \{(x_i^A, y_i^A)\}_{i=1}^{N^A}$, and $D^B \coloneqq \{(x_i^B)\}_{i=1}^{N^B}$, respectively, where x_i^A, $x_i^B \in R^a$. x_i^A is the i^{th} sample of A, x_i^B is the i^{th} sample of B, and y_i^A is the label of x_i^A. N^A and N^B are the sum number of samples in datasets A and B, respectively.

After aligning the samples, each dataset can be divided into overlap parts and nonoverlap parts. The overlap part is a subset of samples with the same identity as other participates. The nonoverlap part is the remaining samples. Herein, the overlap part of dataset is denoted by D_o, and the nonoverlap part of set is denoted by D_n. The corresponding part of the nonoverlap part is called the missing part which is denoted by D_m. Let $D^A = \{(D_o^A, D_n^A)\}$ represent the dataset A, where D_o^A and D_n^A are the overlap part and nonoverlap part of A, respectively. Similarly, the dataset of B can be represented as $D^B = \{D_o^B, D_n^B\}$, where D_o^B and D_n^B are the overlap part and nonoverlap part of B, respectively.

In this paper, we assume that there exists a finite overlap set $D_o \coloneqq \{x_o^B, x_o^A, y_o^A\}_{i=1}^{N_o}$ between the two parties, where party A has the partition $D_o^A \coloneqq \{x_o^A, y_o^A\}_{i=1}^{N_o}$ and party B has the partition $D_o^B \coloneqq \{x_o^B\}_{i=1}^{N_o}$. x_o^A is the overlap sample of part A, y_o^A is the label of x_o^A, and x_o^B is the overlap sample of part

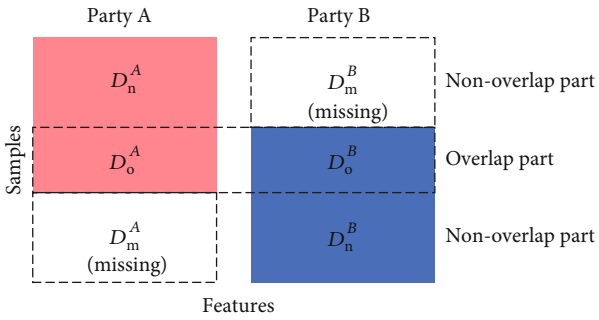


Figure 1: Virtual view of the dataset in vertical federated learning.

B. Besides, we denote the nonoverlap dataset of A as $D_n^A := \{x_n^A, y_n^A\}_{j=1}^{N_n^A}$, whereas x_n^A is the nonoverlap sample of part A. Likewise, we denote the nonoverlap dataset of B as $D_n^B := \{x_n^B\}_{z=1}^{N_n^B}$, whereas x_n^B is the nonoverlap sample of part B.

Generally, a vertical federal machine learning model would be trained on overlap samples, D_o, which is only a small subset of the entire dataset of participants. Each participant still has a large amount of unused nonoverlap data. In order to better improve the overall performance of the training model, we use data augmentation in the nonoverlap sample part.

3.2. Federated Data Augmentation Model. GAN is a type of neural network used for unsupervised and semisupervised learning tasks, such as generating images, improving picture resolution, and retrieving pictures of specific patterns. Compared to other generative models, GAN has a lower computational cost and fewer restrictions. Therefore, based on GAN, we proposed a vertical federated data augmentation method (FedDA); it uses the nonoverlap data to enlarge the set of overlap data. FedDA helps vertical federated learning to achieve better results even if the number of overlap samples is insufficient. The traditional GAN model has many shortcomings, such as the gradient of the generator is unstable or even disappears, the diversity of generated samples is insufficient, and the model convergence speed is slow. However, WGAN-GP can handle the above problems well [19]. So we use WGAN-GP as the basic model.

$$L(G) = -E_{x\sim p_g}[D(x)], \tag{1}$$

$$L(G) = -E_{x\sim p_{\mathrm{data}}}[D(x)] + E_{\tilde{x}\sim p_g}[D(x)] + \lambda E_{\hat{x}\sim p_{\hat{x}}}\left[\left\|\nabla_x D(x)\right\|_p - 1\right]^2\Big]. \tag{2}$$

In WGAN-GP, the loss of generator is formula (1) and the loss of discriminator is formula (2). The third item of formula (2) is to impose a gradient penalty on each sample independently. The gradient penalty solves the gradient explosion problem by connecting parameters with constraints to achieve real Lipschitz constraints. In formula (1) and formula (2), $x \sim P_{\mathrm{data}}$ is real data and $x \sim P_g$ is generated data. For the $\hat{x}$, we perform random interpolation sampling on x and $\tilde{x}$, $\hat{x} = \varepsilon x + (1-\varepsilon)\tilde{x}, \varepsilon \in [0,1]$.

The generator is updated through stochastic gradient descent:

$$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} -D_w(G_\theta(\tilde{x})). \tag{3}$$

The discriminator is updated through stochastic gradient ascent:

$$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m}\left(-D_w(x) + D_w(\tilde{x}) + \lambda\left(\left\|\nabla_{x\wedge} D_w(x\wedge)\right\|_2 - 1\right)^2\right). \tag{4}$$

Formula (3) indicates that the generator intends to increase the score of the false sample as much as possible. Formula (4) indicates that the discriminator intends to increase the score of the true sample as much as possible and lower the score of the false sample.

The deep neural network is widely used in feature representation to learn hidden feature representation. We use neural networks to learn feature representations from raw input data of parts A and B. Feature representation can protect the raw data in a certain extent. And based on feature representation, we can further use privacy protection technology to protect data.

$u_r^p = h_c^p(x_c^p)$ is defined as learning the feature representation of x_c^p through neural network h_c^p, whereas $p \in \{A, B\}$, $c \in \{o, n\}$, $r \in \{o, n, m\}$, $u_r^p \in \mathbb{R}^{N^p * d^p}$, d^p is the dimension of the hidden representation layer at the top of the neural network. Let u_o^p be the feature representations learned from the overlap sample x_o^p. u_n^A and u_n^B are the feature representations learned from the nonoverlap sample x_n^A and x_n^B, respectively. Besides, $\tilde{u}_m^A$ and $\tilde{u}_m^B$ are the generated missing part. For the feature expression, u_o^A,u_n^A,u_o^B,u_n^B were produced by the samples$x_o^A, x_n^A, x_o^B, x_n^B$ through the neural network$h_o^A, h_n^A, h_o^B, h_n^B$, respectively.

The data augmentation process in VFL based on nonoverlap data of two participants is shown in Figure 2. The dataset of every participant consists of two parts (overlap data and nonoverlap data). And each participant has its own generator and discriminator. We use G and D to represent generator and discriminator, respectively.

First, the encryption-based masking technology is used to match the same identity samples between A and B. Furthermore, the deep neural network is used to obtain the feature representation of the overlap samples (u_o^A and u_o^B) and transmit them to the other party, so each participant obtains the feature representations of the shared samples. Furthermore, we train the generator (G^A, G^B) and discriminator (D^A, D^B) in each participant. For the generator, the local nonoverlap samples (x_n^A, x_n^B) are input to the deep neural network to obtain nonoverlap feature representation (u_n^A, u_n^B), and then, the nonoverlap feature representation is input to the generator. The generator through the nonoverlap feature representation to obtain missing parts $(\tilde{u}_m^B, \tilde{u}_m^A)$. For the discriminator, the input data consists of two parts. One part $(\tilde{u}^A, \tilde{u}^B)$ is composed of the feature representation

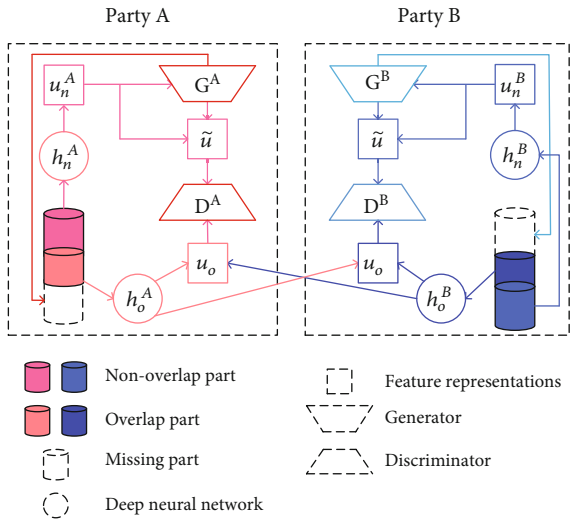


Figure 2: View of the virtual dataset in vertical federated generative adversarial networks.

of the input generator data (u_n^A, u_n^B) and its corresponding generated data $(\tilde{u}_m^B, \tilde{u}_m^A)$. Another part (u_o) is the feature representation of the overlap samples by both parties. After that, the model only needs to train the discriminator and generator until the model converges continuously. Finally, we use the trained generator based on the nonoverlap samples to generate the missing part and merge the missing part and nonoverlap part to get a new overlap set.

We take party A as an example and use Figure 3 to describe the process of data augmentation process in detail.

Step 1: x_o^A, x_n^A, x_o^B get their feature representations u_o^A, u_n^A, u_o^B through the deep neural network h_o^A, h_n^A, h_o^B, respectively.

Step 2: the overlap part data is concatenated to get u_o, as per the following formula.

$$u_o = \text{concat}\left[u_o^A ; u_o^B\right]. \tag{5}$$

Step 3: for the nonoverlap part $\tilde{u}$, missing data $\tilde{u}_m^B$ need to be generated before concatenation. Input u_n^A into the generator and generate $\tilde{u}_m^B$ corresponding to u_n^A, as per formula (6). Then, concatenate u_n^A and $\tilde{u}_m^B$ to get the nonoverlap part $\tilde{u}$, as per formula (7).

$$\tilde{u}_m^B = G_\theta\left(u_n^A\right), \tag{6}$$

$$\tilde{u} = \text{concat}\left[u_n^A ; \tilde{u}_m^B\right]. \tag{7}$$

Step 4: input the u_o as the real part and $\tilde{u}$ as the false part into the discriminator. Use the real part features to guide the false part features, so that the $\tilde{u}_m^B$ generated by the generator can be more realistic and closer to the actual u_m^B. Repeat the train of the generator and discriminator until convergence. Formulas (8) and (9) are updated by formulas (3) and (4).

$$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} - D_w(\tilde{u}), \tag{8}$$

$$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m}\left(-D_w(u_o) + D_w(\tilde{u}) + \lambda\left(\left\|\nabla_{x\wedge} D_w(x\wedge)\right\|_2 - 1\right)^2\right). \tag{9}$$

Step 5: when the generator converges, the feature representations of all nonoverlap samples are input into their respective generators to generate their corresponding missing parts. Then, the nonoverlap feature representation u_n^A and generated feature representations $\tilde{u}_m^B$ are concatenated to get an enlarged dataset $\chi^{A\tilde{B}} = \{u_n^A ; \tilde{u}_m^B\}$.

Similarly, according to the above steps, we can generate the missing data $\tilde{u}_m^A$ corresponding to the nonoverlap part u_n^B of B. By concatenating missing data part and the nonoverlap part, we get the $\chi^{\tilde{A}B}=\{\tilde{u}_m^A ; u_n^B\}$. So, we get a large training set $\chi = \{\chi^{AB}, \chi^{A\tilde{B}}, \chi^{\tilde{A}B}\}$. Hence, A and B participating in the VFL have completed data enhancement based on overlap data and their respective nonoverlap data.

3.3. Algorithm. The entire process of A-party data generation can be summarized as Algorithm 1. A and B have the same training process; the whole algorithm is a loop program. In the first subloop (lines 2-16), we train the discriminator. In WGAN-GP, in order to balance the training of the generator and the discriminator, we need to strengthen the training of the discriminator. In each iteration, we train the discriminator n_{critic} times, and each time, we train m batchs of samples. First, FedDA learn the feature representation of the sample data through the neural network, as executed in lines 4-6. After that, FedDA concatenate the feature representation of the overlap part as the line 7. In addition, the generator generates the missing part of B as line 8. Then, FedDA concatenate the feature representation of the nonoverlap part of A with the feature representation of the missing part of B as the line 9. FedDA calculate the discriminator loss through line 13 according to the data prepared in lines 10-12. Finally, in line 15, FedDA update the discriminator D^A by increasing the stochastic gradient. Next, FedDA train the generator. In the second subloop (lines 17-22), FedDA prepare the training data for the generator. And in the line 23, FedDA update the discriminator by decreasing the stochastic gradient.

The process of generating B-party data is similar to Algorithm 1. When Party A and Party B have completed training, we use the trained generator G^A and G^B to generate the missing part $\chi^{A\tilde{B}}$, $\chi^{\tilde{A}B}$. Finally, FedDA concatenate the overlap part and the supplementary parts into a larger overlap part χ.

In the training process of FedDA, there is no need to share raw data and no restrictions on local models. Each party does not need to know the other party's model structure in each iteration and only exchanges encrypted data hidden feature representations (feature representations can be further encrypted using encryption technology). The deep neural network transforms the feature representation through multiple layers of transformation (our method allows both parties to use different models. And better models can improve the training effect).

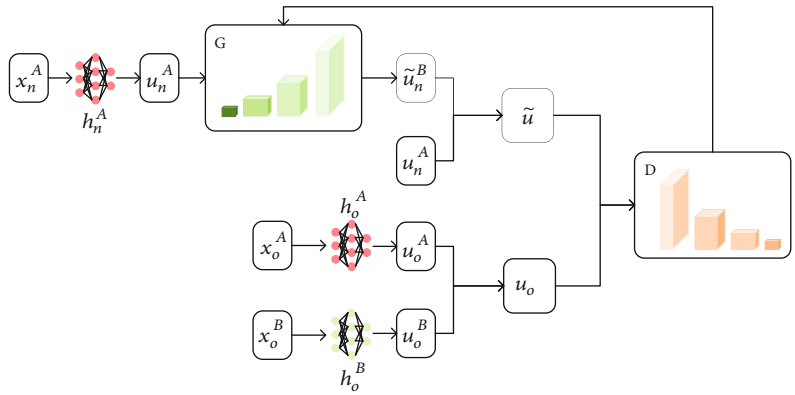


Figure 3: Vertical federated generative adversarial networks of structure diagram.

Input datasets D^A and D^B, neural networks h_o^A, h_o^B, h_n^A, a random number $\varepsilon \sim [0,1]$, epoch number K, the batch size m, Adam hyper-parameters α, intitial cirtic parameters w_0, initial generator parameters θ_0.

for $e = 1, 2, \cdots K$ **do.**
  for $t = 1, 2, \cdots n_{\text{critic}}$ **do.**
    for $i = 1, 2, \cdots m$ **do.**
      Sample $\{x_i^A, x_i^B\}_{i=0}^m$,a batch from the D_o;
      Sample $\{x_j^A\}_{j=0}^m$, a batch from the D_n^A;
      Learn u_i^A, u_i^B, u_j^A through h_o^A, h_o^B, h_n^A;
      $u_i = \text{concat}(u_i^A ; u_i^B)$;
      $\tilde{u}_j^B = G_\theta(u_j^A)$;
      $\tilde{u}_j = \text{concat}(u_j^A ; \tilde{u}_j^B)$;
      Sample real data $x \longleftarrow u_i$;
      Latent variable $\tilde{x} \longleftarrow u_j$;
      $\hat{x} = \varepsilon x + (1 - \varepsilon)\tilde{x}$;
      $L^{(i)} \longleftarrow D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{x\wedge} D_w(x\wedge)\|_2 - 1)^2$
    end for
    $w \longleftarrow \text{Adam}(\nabla_w (1/m) \sum_{i=1}^m L^{(i)}, w, \alpha)$
  end for
  for $i = 1, 2, \cdots m$ **do**
    Sample $\{x_j^A\}_{j=0}^m$, a batch from the D_n^A;
    Learn u_j^A through h_n^A;
    $\tilde{u}_j^B = G_\theta(u_j^A)$;
    $\tilde{u}_j = \text{concat}(u_j^A ; \tilde{u}_j^B)$;
end for
$\theta \longleftarrow \text{Adam}(\nabla_\theta (1/m) \sum_{i=1}^m - D_w(\tilde{u}_j), \theta, \alpha)$
end for

Algorithm 1: FedDA algorithm of part A. We use default values of λ=10, n_{critic}=5, and α=0.0001.

4. Experiment Evaluation

In this section, to verify the performance of the method we proposed, we designed a series of experiments based on the dataset of MNIST and CIFAR-10. And we use the Fréchet Inception Distance (FID) as the metrics to quantitatively evaluate the FedDA data augmentation method [20].

In our simulations, we still suppose that there are two participants, A and B. The two parties are entirely independent. Moreover, each of them has its own features. As we described above, the datasets of A and B have overlap parts and nonoverlap parts. To imitate the dataset composition, we divide the examples in MNIST or CIFAR-10 into two parts. One part is the overlap part which is shared by both A and B. The other part is the nonoverlap part which belongs to participant A. To establish the relationships in these two parts of samples, we have to split each sample (a picture in MNIST or CIFAR-10) into two pieces. For the overlap part, we distribute the two pieces of one picture to A and B, respectively. For the nonoverlap part, we distribute

one piece to A and drop the remaining piece. Therefore, we can use the settings to simulate the circumstance of vertical federated learning with two participants.

Our FedDA model tries to generate a dropping piece of picture in B basing on a piece of picture which belongs to the nonoverlap parts of A. Hence, the combinations of generation piece and nonoverlap piece can be used as a new overlap sample. It denotes that the data of overlap parts can be augmented by our method. Theoretically, the generated piece should be the same as the piece we dropped when we form the nonoverlap part. In practice, if the generation piece is more similar to the dropped piece, the effect of the augmentation is better. To verify the effect of augmentation, we calculate the FID scores of each pair of generation piece and dropped piece.

The FID can capture the similarity of generated images to real ones. It is a method of discrimination based on spatial distribution. FID first calculates the mean value and covariance of the image. It calculates the statistical information through the multivariate GAUSSIAN function to obtain the real image's spatial distribution and the generated image's spatial distribution. The Fréchet distance (also known as the Wasserstein-2 distance) is then used to calculate the distance between these two distributions. The lower the FID score, the better the quality of the image; on the contrary, the higher the score, the worse the quality of the image. When the score is 0, the two pictures can be considered the same. When a generated image is output, we will use FID to calculate the similarity between the generated image and the target image to measure our generated image's quality.

Figure 4 shows calculation process of the FID score. First, the nonoverlap data of A has to be prepared according to the previous settings. Then, after each round of training, we input a batch of data into the trained generator. Further, we get the missing data in part B corresponding to A by generator. Finally, we combine the output data and nonoverlap data into complete data. And we use the complete data and the original data to calculate the FID score.

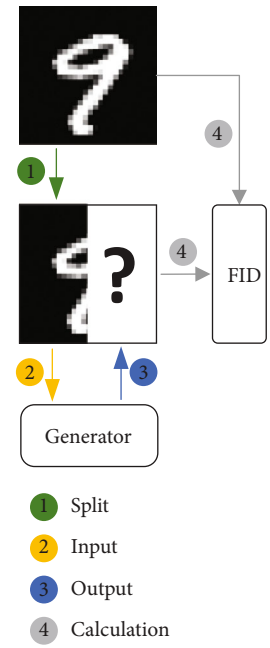


Figure 4: The FID calculates process.

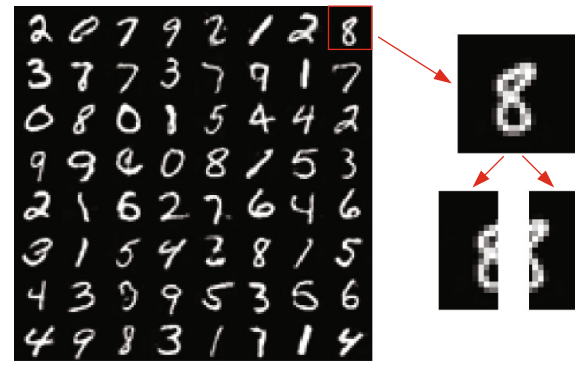


Figure 5: Samples on the MNIST dataset.

4.1. Simulations on Dataset of MNIST. MNIST is the classic dataset that is heavily used in the field of machine learning. It contains 70000 handwritten digital images. And the entire dataset was separated into 60000 training images and 10000 test images. As our design, the images will be used as samples of participants. To construct the overlap and nonoverlap data, we horizontally divide every picture ($28 * 28$ pixels) in MNIST into two pieces which are rectangle pictures ($28 * 14$ pixels), as shown in Figure 5. Considering the diversity of the samples, to ensure the reliability of the FID score, for each iteration of the experiments, we randomly select a batch (the batch size is 64) to calculate its FID score according to the process shown in Figure 4; we repeat it 200 times and finally get the average FID score of this iteration.

The first experimental group is aimed at studying the effect of using FedDA for data augments under different numbers of overlap samples and with a certain number of nonoverlap samples. Figure 6 shows that the FID score continues to decline, while the number of overlap samples rises. The result means that the increase in the number of overlap samples can help us generate higher quality images. As we all know, increasing the number of samples can help us train a better model. When the number of samples is insufficient, the model's performance is often terrible. However, when the number of samples is lacking, FedDA also have a good effect. For instance, the blue line has only 250 overlap samples, but its average FID score has reached a lower level, which means that our method can enhance data and solve the problem of insufficient training data.

Moreover, we study the effect with or without nonoverlap data. For training without nonoverlap samples, we set the source of the generator to only come from the overlap sample dataset as a baseline. Hence, we set the baseline of different sizes (represented by the blue lines) and use 10000 nonoverlap data (represented by the orange lines) as a comparison. The result is shown in Figure 7. We noticed that in Figures 7(a)–7(c), the blue and orange lines converge in similar iteration times, but the average FID score of the

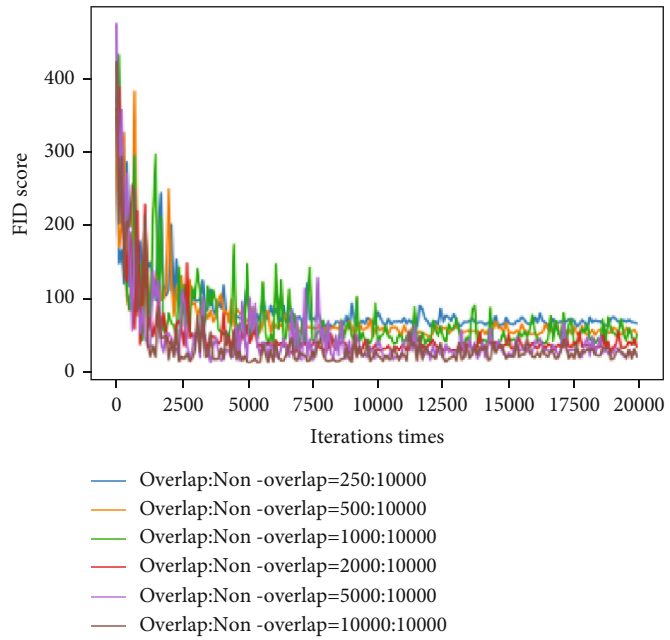


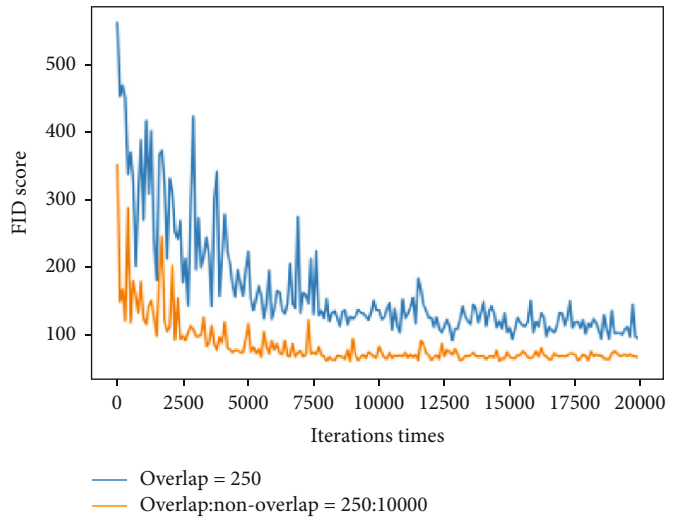
Figure 6: FID score on MNIST: nonoverlap = 10000 and overlap = 250, 500, 1000, 2000, 5000, and 10000.

orange lines is significantly lower than that of the blue lines. These findings prove that when the number of overlap samples is small, although the convergence speed of the blue line is almost the same as that of the orange line, the effect of the orange lines is significantly better than that of the blue lines. It means that our data augmentation can provide better samples. Furthermore, in Figures 7(d)–7(f), as the number of overlap samples increases, the iteration times required for the blue lines have significantly increased over the orange lines. In Figure 7(f), the iteration time difference has reached about 5000 times. We can find that when the number of overlap samples is huge, the quality of pictures without nonoverlap data is only slightly lower than that using nonoverlap data. However, the convergence speed of orange lines is significantly faster than that of blue lines.

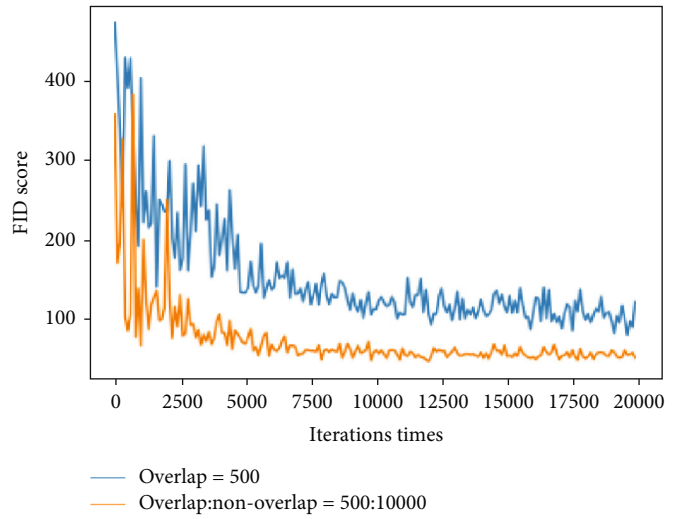
More intuitively, we listed the average FID score of each curve after convergence in Figure 7 in Table 1. Table 1 highlights the difference between using the nonoverlap samples and without the nonoverlap samples. Obviously, comparing the results with nonoverlap and without nonoverlap, we found that no matter how many samples in the overlap datasets, the results with nonoverlap is always smaller than the results without nonoverlap. Besides, as the number of overlap samples increases, the average FID score difference between with nonoverlap and without nonoverlap gradually becomes smaller. For example, when used with 250 overlap samples, the average FID score without nonoverlap is 119.45. In contrast to the results with nonoverlap, the average FID score is 68.67, a drop of 50.78. This score difference is still very obvious, but when the number of overlap samples reaches 10000, the average FID score with nonoverlap drops by 9.97 compared with the result without nonoverlap. After using nonoverlap samples, the available samples for learning are augmented, the model can learn more data, and the quality of the generated pictures is significantly improved. What is more, notice that when the number of overlap samples is 2000, and with nonoverlap samples, the result is lower than when the number of overlap samples is 10000 and without nonoverlap samples. The result of using the nonoverlap parts when the overlap is small can be similar to when the overlap parts are large. It confirms our conclusion that we can use a large amount of local nonoverlap data to augment data in the case of insufficient overlap data.

Based on the above phenomenon, we have summarized the conclusion that when the number of overlap samples is small, the iteration times with nonoverlap are slightly faster than those without overlap, but the result with nonoverlap is significantly better than that without nonoverlap. Further, when the number of overlap samples is huge, the result with nonoverlap is slightly better than that without overlap, but the iteration times with nonoverlap is significantly faster than those without nonoverlap.

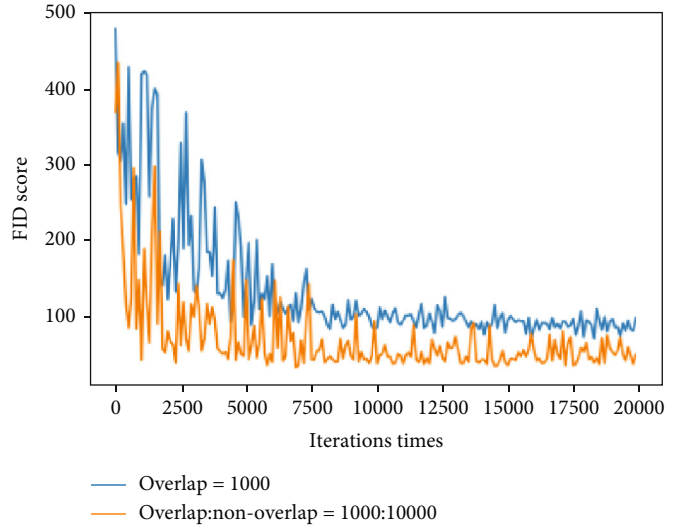
Further, we fixed the number of overlap samples and changed the number of nonoverlap samples to analyze the impact of changes in the number of nonoverlap samples on the model. Figure 8 shows the difference in the average FID scores when the number of overlap samples is 500 (Figure 8(a)) and 1000 (Figure 8(b)) under a different number of nonoverlap samples. Obviously, as the number of nonoverlap samples increases, the average FID score will decrease. This means that the larger the number of nonoverlap data, the better the effect of our data enhancement method. In addition, we found that when the number of nonoverlap samples is less than 2000, the average FID score has increased rapidly. But, when the number of nonoverlap samples is more than 2000, the improvement is gradually slow. Finally, the result with 10000 nonoverlap samples is only slightly higher than that with 2000 nonoverlap samples. It is apparent from this picture that the results will be significantly improved after using a small number of nonoverlap



(a)

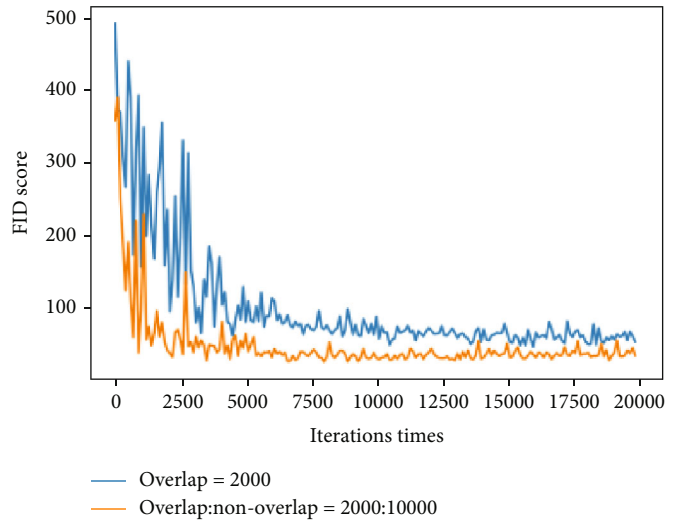


(b)

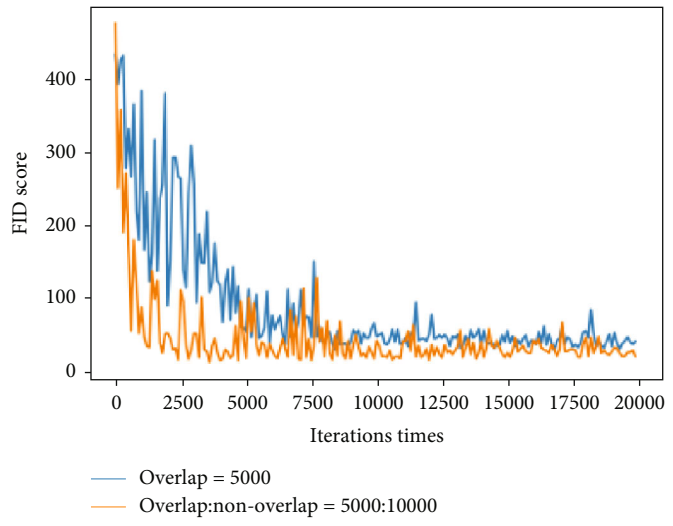


(c)

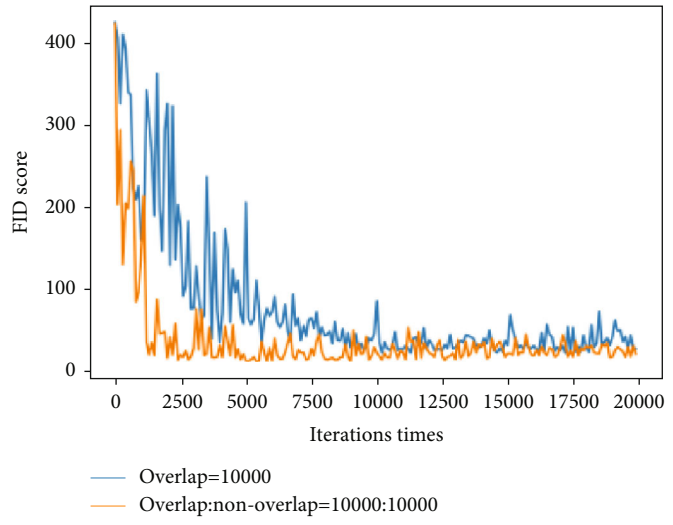
Figure 7: Continued.



(d)



(e)



(f)

Figure 7: FID score on MNIST: overlap = 250 (a), 500 (b), 1000 (c), 2000 (d), 5000 (e), and 10000 (f); nonoverlap = 10000 or without nonoverlap.

Table 1: The average FID score corresponding to Figure 7.

Overlap	250	500	1000	2000	5000	10000
With nonoverlap	68.67	57.02	48.54	34.93	30.33	25.47
Without nonoverlap	119.45	112.44	92.49	61.55	44.73	35.44

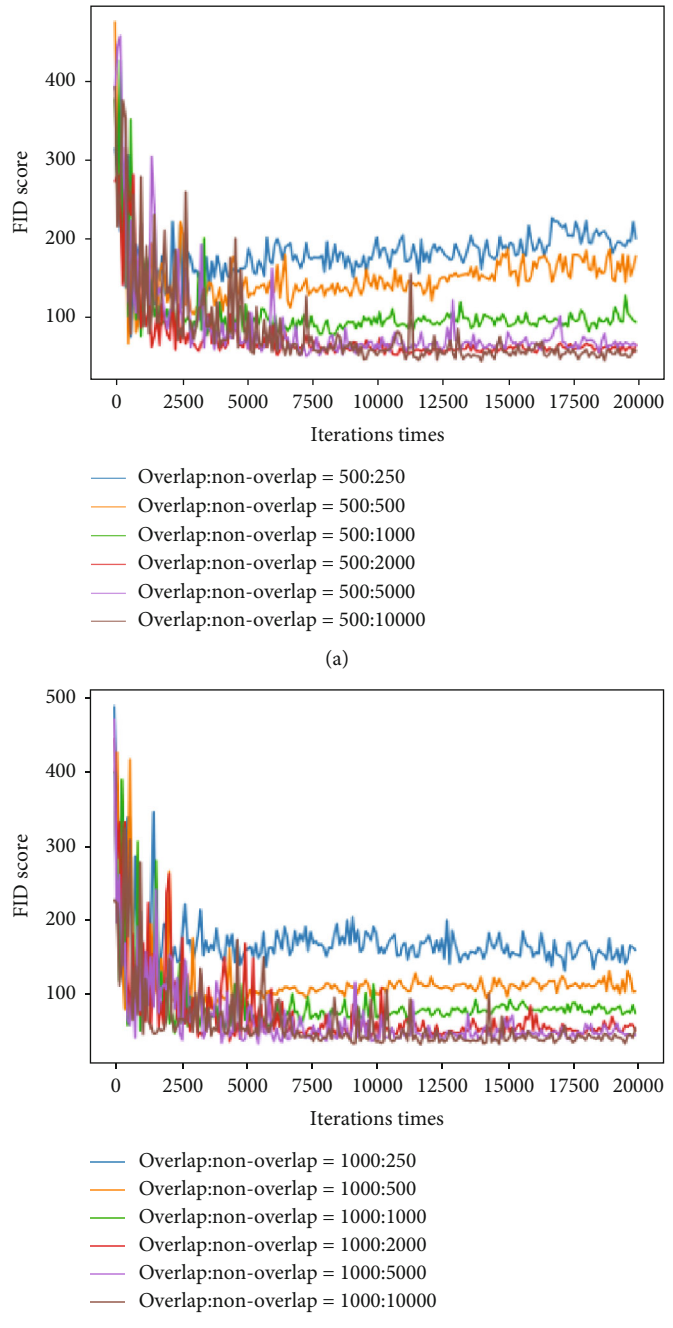


Figure 8: (a) FID score on MNIST: overlap = 500 and nonoverlap = 250, 500, 1000, 2000, 5000, and 10000. (b) FID score on MNIST: overlap = 1000 and nonoverlap = 250, 500, 1000, 2000, 5000, and 10000.

samples. But this improvement will gradually decrease as the number of nonoverlap samples increases. Besides, the improvement of training results can increase the number of overlap samples and also can improve the results of training by increasing the number of nonoverlap samples. Comparing the lines in Figures 8(a) and 8(b), we can see that even only using 500 overlap samples, the training results have been significantly improved. However, it is usually

Table 2: The average FID score corresponding to Figure 8.

Nonoverlap	250	500	1000	2000	5000	10000
Overlap (500)	191.35	156.70	96.24	67.84	59.86	57.02
Overlap (1000)	160.68	112.00	78.68	61.55	54.67	48.54

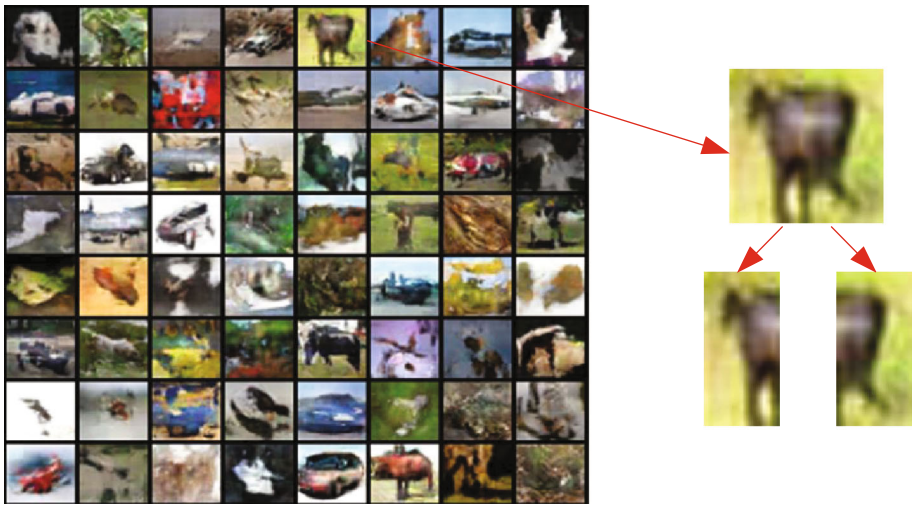


Figure 9: Samples on the CIFAR-10 dataset.

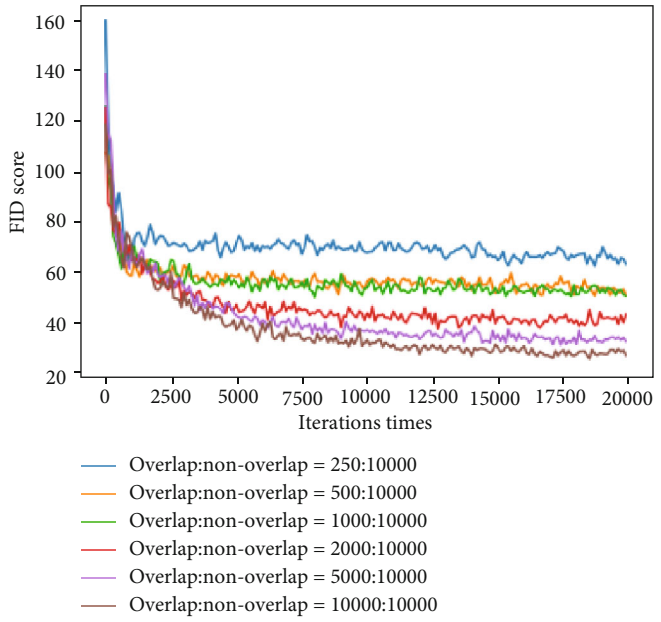
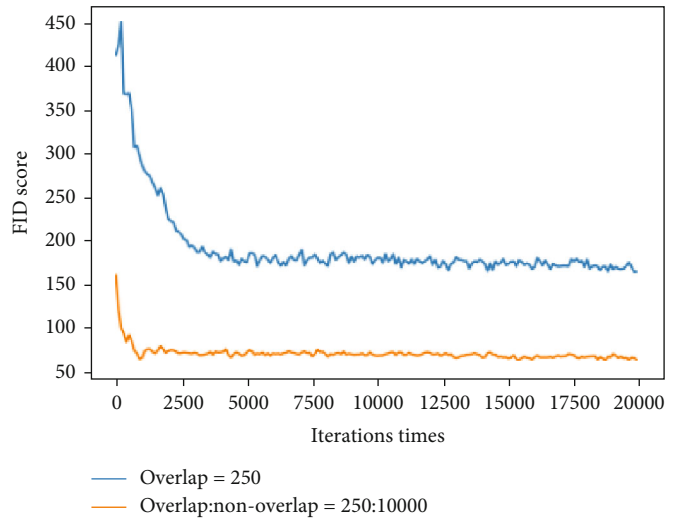


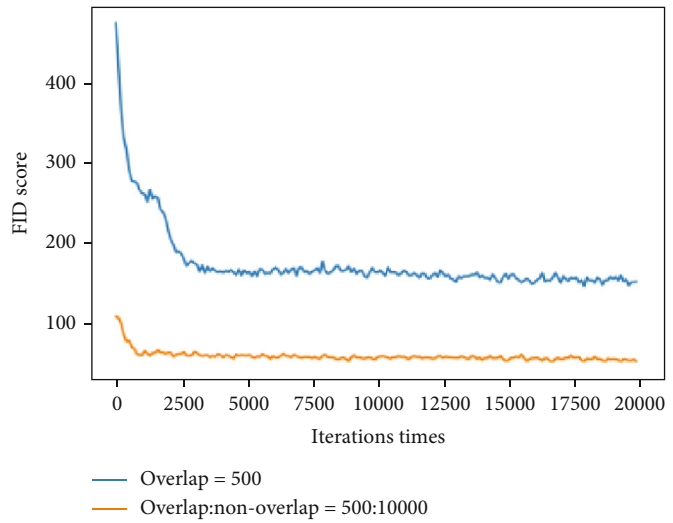
Figure 10: FID score on CIFAR-10: nonoverlap = 10000 and overlap = 250, 500, 1000, 2000, 5000, and 10000.

difficult to increase the number of overlap samples, but we can significantly improve the performance of the model by using a large number of local nonoverlap samples.

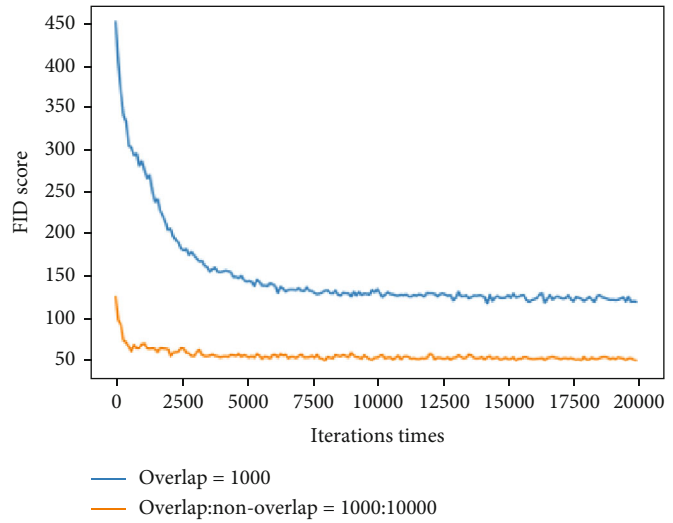
Table 2 shows the average of the FID scores in Figure 8. As shown in the table, regardless of the number of the overlap samples, when we use more nonoverlap samples, the average FID score decreases. When using 500 overlap samples and 250 nonoverlap samples, the average FID score is 191.35. Then, as the number of overlap samples increases, the average FID score gradually decreases. When the number of nonoverlap samples added reaches 10000, the average FID score will fall to 57.02. Similarly, when the overlap samples are 1000, we also can see that with the increase of nonoverlap samples, the average FID score is gradually decreasing. It denotes that using more nonoverlap data can help further improve the quality of generated data. Besides, we noticed that when the number of overlap samples is small, the result of adding the same number of nonoverlap samples is better than overlap samples. For example, when the number of nonoverlap samples is 500, adding 500



(a)

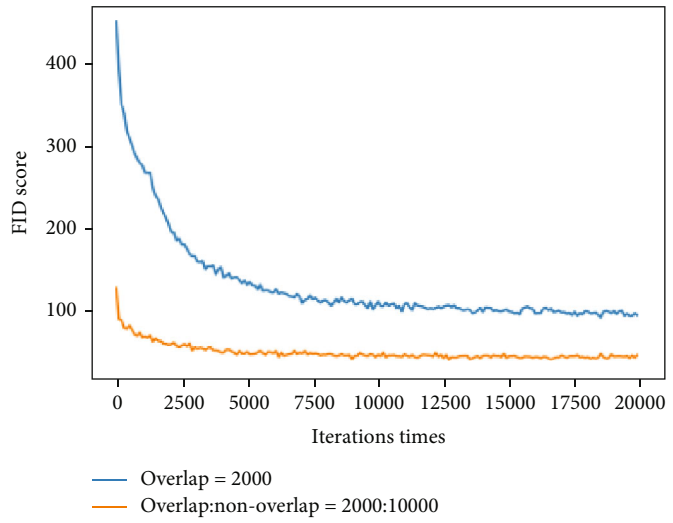


(b)

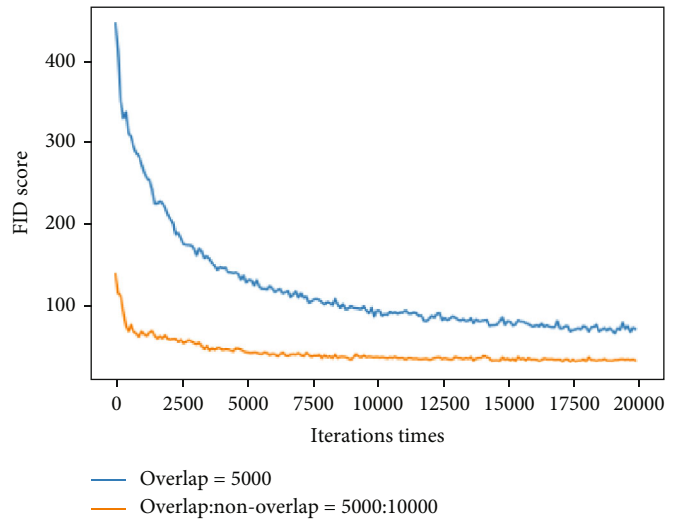


(c)

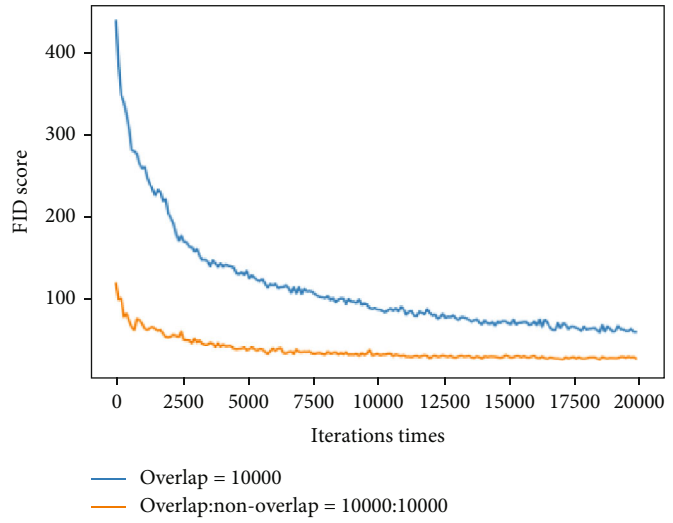
Figure 11: Continued.



(d)



(e)



(f)

Figure 11: FID score on MNIST: overlap = 250 (a), 500 (b), 1000 (c), 2000 (d), 5000 (e), and 10000 (f); nonoverlap = 10000 or without nonoverlap.

Table 3: The average FID score corresponding to Figure 11.

Overlap	250	500	1000	2000	5000	10000
With nonoverlap	66.66	54.07	51.78	40.62	33.58	28.28
Without nonoverlap	172.41	155.22	124.44	97.27	78.16	71.53

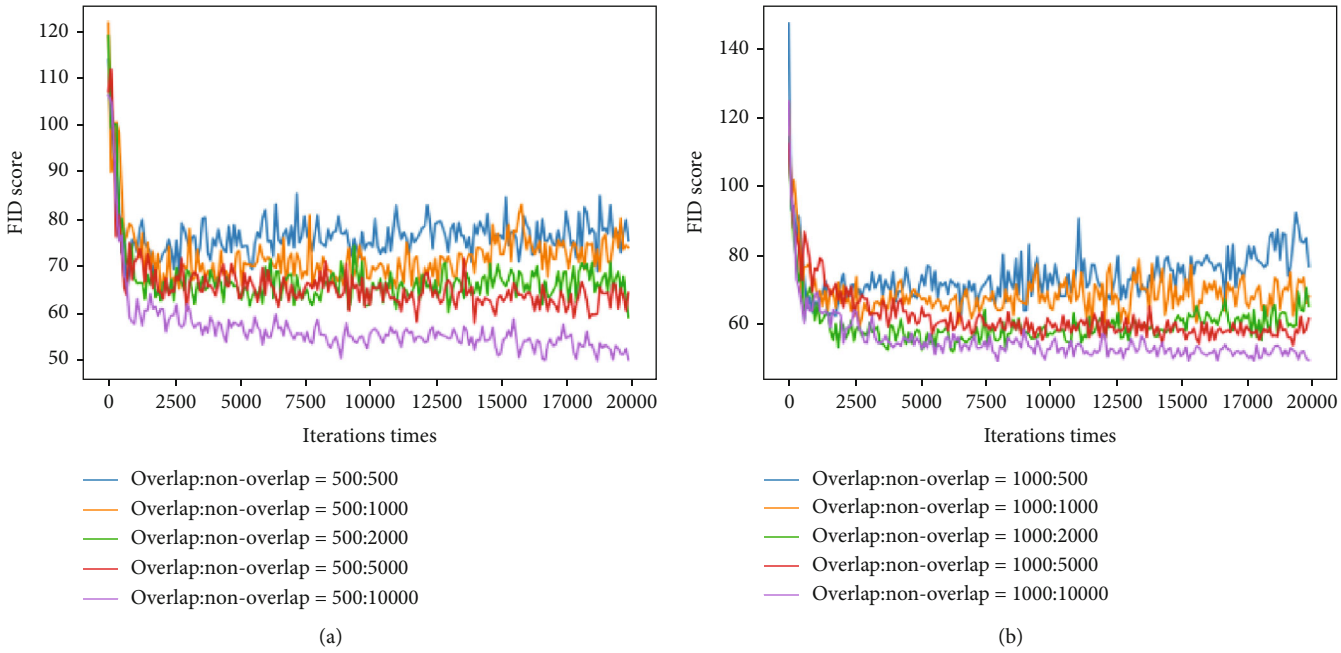


Figure 12: (a) FID score on CIFAR-10: overlap = 500 and nonoverlap = 250, 500, 1000, 2000, 5000, and 10000. (b) FID score on CIFAR-10: overlap = 1000 and nonoverlap = 250, 500, 1000, 2000, 5000, and 10000.

overlap samples will reduce the average FID score by 44.70. However, adding 500 nonoverlap samples will reduce the average FID score by 60.4. Hence, when overlap samples are scarce, it is necessary to utilize the local many nonoverlap samples.

Table 4: The average FID score corresponding to Figure 12.

Nonoverlap	500	1000	2000	5000	10000
Overlap (500)	76.74	72.14	66.10	63.31	54.07
Overlap (1000)	78.97	68.86	61.90	57.93	51.78

4.2. Simulations on Dataset of CIFAR-10. In this section, we execute a series of similar experiments on CIFAR-10, shown in Figure 9. CIFAR-10 is a dataset of color images which are closer to universal matters. It contains ten categories of RGB color pictures with $32 * 32$ pixels. And each category has 6000 images. Different from MNIST, the pictures in CIFAR-10 have a 3-channel color. And the size of pictures in CIFAR-10 is slightly larger. Besides, the proportions and characteristics of the objects in the sample are different, which bring great difficulties to recognition. Therefore, it can better compare and represent the effects of our method. Hence, in this part of the experiment, the experimental setting is the same as the setting on the MNIST.

In the same way, we execute the experiments on CIFAR-10. First, using the same number of nonoverlap samples, we studied the effect of data augments when the number of overlap samples increases. Figure 10 shows that the FID score continues to decline, while the number of overlap samples rises. In addition, when the number of samples in the overlap part is insufficient, the FID score will converge to a bigger value which is actually a lower level. And the tendency of curves in Figure 10 is consistent with the result on MNIST.

Apparently, compared with the same experiments on MNINT (shown in Figure 6), it has better performance on the CIFAR-10 (shown in Figure 10). The curves fluctuated more stably, the model converged faster, and the FID score is smaller. The reason is that a larger size of the image can provide more information.

Moreover, we study the effect with or without nonoverlap data. The result is shown in Figure 11. Similar to Figure 7, when the number of overlap samples is insufficient, the result of the orange line is significantly better than that of the blue. When the number of overlap samples is sufficient, the orange line iteration times are significantly faster than those of the blue line. Furthermore, comparing Figure 11 to Figure 7, the fluctuation of the FID score curves becomes smaller, and the number of iteration times required for convergence is relatively reduced.

In addition, in Figures 11(a)–11(e), the number of iteration times for the blue line to converge is approximately four times faster than that for the orange line. Even in Figure 11(f),

the blue line has not converged in 20000 iteration times, while the orange line stabilized in 6000 iteration times. Unlike the previous experiment on MNIST, after adding nonoverlap samples, even if the number of overlap samples is small, the number of iteration times required for convergence is also significantly reduced. On the other hand, it can be clearly found that as the overlapped samples increase, the distance between the blue line and the orange line gradually decreases. When the number of overlap samples is 10000, the distance between the blue line and the orange line is very close on MNIST, but on CIFAR-10, the distance between them is still very obvious.

In Table 3, we list the average FID score of each curve after convergence in Figure 11. It highlights the difference between data augmentation with the nonoverlap samples and without the nonoverlap samples. Comparing Table 3 with Table 1, our method has more excellent performance on more complex images. In Table 3, as the number of overlap samples increases, the score of using nonoverlap samples is always lower than the score without nonoverlap samples. However, the difference of a pair of scores in each column is gradually decreasing. When the number of overlap samples reaches 10000, the score of using nonoverlap samples is still 43.25 ahead of that without nonoverlap samples. This means that the result of using nonoverlap is still significantly better than that without nonoverlap, even if the number of overlap samples is large.

Further, we verify the influence of changing the number of nonoverlap samples. We fixed the number of overlap samples and changed the number of nonoverlap samples to analyze the effects of augmentation with the different number of nonoverlap samples. Figure 12 shows the trend of FID scores with the different number of nonoverlap samples when the number of overlap samples is fixed at 500 and 1000. Obviously, as the number of nonoverlap samples increases, the FID score continuously decreased. The result is different from experiments on MNIST. In Figure 8, when the number of nonoverlap samples exceeds 2000, the FID score will decrease slowly. However, in Figure 12, the FID score can still steadily fall when it exceeds 2000. This means that nonoverlap samples with rich features can better train the generator to improve the quality of augmenting data.

Table 4 shows the average FID scores of each line in Figure 12. The results are similar to Table 2. For the same number of overlap samples, as nonoverlap samples increase, the FID score steadily decreases. Besides, as nonoverlap samples increase, the difference of FID score between 500 overlap samples and 1000 overlap samples gradually becomes smaller. In addition, the FID score in Table 4 is significantly better than that in Table 2 when the number of nonoverlap samples is less than 2000. This means that when the number of nonoverlap samples is small, the quality of the generated picture based on the complex image was improved. Further, the score difference between the average FID score of 500 overlaps and 1000 overlaps in Table 4 is smaller than that in Table 2. The average FID score with 500 overlap samples is slightly lower than the average FID score with 1000 overlap samples. This result proved that our method could alleviate the lack of overlap samples in large-sized images. In one special case on column 1, when the nonoverlap sample is 500, the FID score with 1000 overlap samples is greater than with 500 overlap samples. The reason is the overfitting of the model. As shown in Figure 12(b), the blue curve with 500 overlap samples starts rising after 15000 iteration times.

5. Conclusion

In vertical federated learning, the overlap data is usually only a small subset of the entire dataset of participants. Each participant has a large amount of nonoverlap data, but it is not utilized for vertical federated learning. Considering this situation, we propose a data enhancement method called FedDA. FedDA generates corresponding missing data based on local nonoverlap data. It can provide a large amount of overlap data for subsequent vertical federation training. We designed a series of experiments on MNIST and CIFAR-10. The results confirm that our method can generate overlap data based on nonoverlap parts of the data. Further, whether it is to utilize the overlap data or to not utilize the nonoverlap data, the quality of the generated data will be improved. Normally, with the utilized overlap data, the number of iterations obviously decreases. Meanwhile, it is worth noting that in the absence of overlap data, using a large number of nonoverlap samples can significantly improve the quality of the generated data. Furthermore, in the dataset with richer sample features, the quality of the missing data generated is generally better. The fusion of training federated learning model methods is an exciting problem, and we will explore it in further research.

Data Availability

All data included in this study are available upon request by contact with the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by the project "User Behavior Features Oriented Research on Analysis of Multi-Source Data in CDN" (20200401082GX), which is financially supported by the Science & Technology Development Program of Jilin Province, China.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[2] D. Byrd and A. Polychroniadou, "Differentially private secure multiparty computation for federated learning in financial applications," https://arxiv.org/abs/2010.05867, 2020.

[3] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[4] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: challenges and opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 2020.

[5] Y. Zhao, J. Zhao, L. Jiang, R. Tan, and D. Niyato, "Mobile edge computing, blockchain and reputation-based crowdsourcing IoT federated learning: a secure, decentralized and privacy-preserving system," https://arxiv.org/abs/1906.10893, 2019.

[6] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279–1283, 2019.

[7] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid, "Privacy preserving schema and data matching," *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 653–664, 2007.

[8] M. Ogburn, C. Turner, and P. Dahal, "Homomorphic encryption," *Procedia Computer Science*, vol. 20, pp. 502–509, 2013.

[9] S. Hardy, W. Henecka, H. Ivey-Law et al., "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," https://arxiv.org/abs/1711.10677, 2017.

[10] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning via over-the-air computation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2022–2035, 2020.

[11] S. Yang, B. Ren, X. Zhou, and L. Liu, "Parallel distributed logistic regression for vertical federated learning without third-party coordinator," https://arxiv.org/abs/1911.09824, 2019.

[12] Y. Liu, Y. Liu, Z. Liu et al., "Federated forest," *IEEE Transactions on Big Data*, 2020.

[13] K. Cheng, T. Fan, Y. Jin et al., "Secureboost: a lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[14] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 4, pp. 4642–4649, 2020.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial networks," https://arxiv.org/abs/1406.2661, 2014.

[16] A. Rajagopal and V. Nirmala, "Federated AI lets a team imagine together: federated learning of GANs," *Corr*, vol. 7, no. 5, pp. 704–709, 2019.

[17] M. Rasouli, T. Sun, and R. Rajagopal, "FedGAN: federated generative adversarial networks for distributed data," https://arxiv.org/abs/2006.07228, 2020.

[18] C. Fan and P. Liu, "Federated generative adversarial learning," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 3–15, Springer, 2020.

[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of Wasserstein GANs," https://arxiv.org/abs/1704.00028, 2017.

[20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," https://arxiv.org/abs/1706.08500, 2017.