Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 5510329, 12 pages
https://doi.org/10.1155/2023/5510329

Research Article

Fast and Accurate Deep Leakage from Gradients Based on Wasserstein Distance

Xing He,[1,2] **Changgen Peng**,[1,3] **and Weijie Tan**[1,3,4]

[1]*State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China*
[2]*Guizhou Minzu University, Guiyang 550025, China*
[3]*Guizhou Big Data Academy, Guizhou University, Guiyang 550025, China*
[4]*Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, Guizhou University, Guiyang 550025, China*

Correspondence should be addressed to Changgen Peng; cgpeng@gzu.edu.cn

Received 5 December 2022; Revised 15 February 2023; Accepted 28 February 2023; Published 21 March 2023

Academic Editor: Yu-an Tan

Shared gradients are widely used to protect the private information of training data in distributed machine learning systems. However, Deep Leakage from Gradients (DLG) research has found that private training data can be recovered from shared gradients. The DLG method still has some issues such as the "Exploding Gradient," low attack success rate, and low fidelity of recovered data. In this study, a Wasserstein DLG method, named WDLG, is proposed; the theoretical analysis shows that under the premise that the output layer of the model has a "bias" term, predicting the "label" of the data by whether the "bias" is "negative" or not is independent of the approximation of the shared gradient, and thus, the label of the data can be recovered with 100% accuracy. In the proposed method, the Wasserstein distance is used to calculate the error loss between the shared gradient and the virtual gradient, which improves model training stability, solves the "Exploding Gradient" phenomenon, and improves the fidelity of the recovered data. Moreover, a large learning rate strategy is designed to improve model training convergence speed in-depth. Finally, the WDLG method is validated on datasets from MNIST, Fashion MNIST, SVHN, CIFAR-100, and LFW. Experiments results show that the proposed WDLG method provides more stable updates for virtual data, a higher attack success rate, faster model convergence, higher image fidelity during recovery, and support for designing large learning rate strategies.

1. Introduction

Shared gradients are common in computationally and time-intensive task systems. Machine learning (ML) model training is frequently used in distributed training and collaborative learning due to its similar task characteristics. The gradients are shared and aggregated across multiple microprocessors (nodes) for machine learning model training; each worker node operates in parallel to accelerate ML model training. Therefore, distributed training methods are critical for speeding up model training on large-scale datasets.

In recent years, research using shared gradient methods has yielded excellent results. McMahan et al. [1] were the first to propose the Federated Learning (FL) architecture. FL builds iterable aggregated models by training distributed models across multiple data sources with local data, only exchanging model parameters or intermediate results models, and thus learning a shared target model. Improved approaches based on FL methods [2–9] have carried out a lot of research work in achieving a balance between data privacy protection and data sharing computation. Currently, more researchers are using cryptographic privacy-preserving methods and differential privacy-preserving methods to achieve privacy-preserving security for local gradients in the federal learning security problem [10, 11]. Meanwhile, for large-scale ML tasks, other distributed training [12–14] and collaborative learning [15, 16] methods are widely used.

In distributed ML systems, it is widely assumed that "shared gradients" do not leak the training dataset. The output of machine learning model decisions frequently contains a large amount of inference information from training samples; recent studies have shown that these outputs are vulnerable to membership inference attacks [17–21] and gradient reversal attacks [16, 22–25], which result in leaking training datasets. To the best of our knowledge, the most dominant Deep Leakage from Gradients (DLG) [24] and Improved DLG (iDLG) [25] are currently among the most effective gradient-based inverse attacks. Nevertheless, the DLG method still generates random noisy images for training data recovery, and "Exploding Gradients" is one of the main reasons for the low success rate of DLG method attacks. In addition, the DLG method has problems such as slow convergence speed of model training, low fidelity of recovered data (the original training data can be recovered; however, the recovered training data still has random noise in some positions of the image), and an undesirable design strategy of a large learning rate.

For starters, data label prediction is more accurate and faster. We theoretically prove that the prediction of data labels is independent of the recovery of training data to predict data labels more accurately and improve attack accuracy, assuming that the activation function is ReLU and using cross-entropy to calculate the loss value. Our method's time complexity is much lower than that of iDLG's method, especially when the last hidden layer contains more neurons. Second, model training is more stable, and convergence occurs more quickly. The WDLG method can stabilize model training and play the role of fast convergence, and the main reason why the analysis can solve the phenomenon of "Exploding Gradient" is that the original gradient calculated by random initialization of model parameters has certain randomness, and the gradient calculated during training still has certain randomness. The superiority of calculating the distance between two random variables according to the Wasserstein distance can better stabilize the model training to better calculate the distance between two random variables, which is the first attempt to improve the original DLG method as far as we know. Third, large learning rate model training is more stable; we attempt to provide a large learning rate design strategy to improve model training speed and achieve better results; however, the balance between model training stability and learning rate design must be fully considered.

Our proposed WDLG method has been validated on MNIST [26], Fashion MNIST [27], SVHN [28], CIFAR-100 [29], and LFW [30] datasets. The experiments demonstrate the benefits of the WDLG method for more stable virtual data updates for model training, higher attack success rates, faster model convergence, and support for large learning rate design strategies. This paper's main contributions are as follows:

(i) We assume that the output layer employs a biased ReLU activation function and that the model employs cross-entropy loss. Based on a differentiable model, the WDLG method differentiates the bias term and theoretically demonstrates that the prediction of label data is only related to the bias of the output layer, independent of the recovery of the training data. As a result, the true labels are extracted with 100% accuracy from the shared gradients, allowing for more accurate data extraction.

(ii) Wasserstein distance is used to calculate the gradient loss, and a new loss function is selected to solve the stability of the model training. The model training converges faster and solves the "Exploding Gradient" phenomenon.

(iii) Through experiments, we analyze the relationship between the stability of model training and the fidelity of recovered data with different learning rates and conclude that the WDLG method can support large learning rate design strategies.

The rest of this paper is organized as follows: Section 2 describes the related work in detail. Section 3 presents our method, which includes the extremely important prediction proofs for training data labels, the choice of the loss function, and the approximation of the Wasserstein distance calculation. Section 4 gives three valid comparison experiments, which are the comparison of prediction accuracy of training data labels, the comparison of training model stability and convergence speed, and the comparison of large learning rate training. Finally, Section 6 concludes the paper.

2. Related Works

Many studies have been conducted to improve the scalability of distributed training of ML models during distributed training of ML models. Sharing data gradients is a common practice in existing distributed ML systems, and multiple parties' data are jointly trained using the shared gradient information. Because of stochastic gradient descent's stable scalability, the algorithm [31–34] and framework [35–37] have essentially adopted synchronized stochastic gradient descent as the dominant method for parameter updating. However, both centralized [38–40] and decentralized [34, 41] training methods have the potential technical security risk of data and model parameters being compromised during training and application. One of the most common attacks for model leakage is the model reversal attack, in which the attacker extracts information about the training data from the model prediction results. Model reversal attacks are classified into two types based on the different model output information possessed by attackers: model output-based data leakage and gradient update-based data leakage.

2.1. Model Output-Based Data Leakage. The output of machine learning model decisions frequently contains many inferences about the training data, and recent studies have shown that these outputs can be used to steal information about the original training data. Fredrikson et al. [17] proposed a confidence-based model reversal attack method and achieved better attack results on the model [42–44]. It demonstrated that confidence could be used as a measure to recover training data and learn sensitive private information about individuals for adversarial access to ML models. The membership inference attack was first proposed by Shokri et al. [18]; the basic principle of the attack is to statistically analyze the discrepancy between the prediction results of ML models on training and nontraining data; the attacker can implement the membership inference attack if he can capture this discrepancy. On this basis, the equivalent attack effect of Shokri's proposed method is achieved by fitting the membership inference attack [45] and membership inference attack in the black box scenario [46]. The literature [6, 47] investigated the membership inference attack in white-box scenarios and successfully built more robust attacks by exploiting the difference in model gradients between training and nontraining data, resulting in better attack results. Song et al. [48] discovered that adding adversarial training during the training of ML models increases the risk of the model leaking members' private information through an in-depth study of membership inference attacks. By using generative adversarial networks [49] to learn the statistics of real data distributions, Hayes et al. [19] completed the construction of membership inference attacks. Furthermore, Salem et al. [20] improved the membership inference attack by extending it to the domain of online learning.

2.2. Gradient Update-Based Data Leakage. The gradients continuously generated by machine learning models during training still imply rich privacy information about the training dataset. Melis et al. [16] used user-updated model parameters as features for training attack model inputs for inferring relevant attributes of other user datasets. The literature [7, 50, 51] employs generative adversarial networks to generate methods for recovering training data from other users, and Mahendran et al. [22] investigate gradient inversion information maximization to synthesize real data from training networks, but both rely on a priori information from auxiliary datasets. Mordvintsev et al. [23] use only the gradients in the input to enable the separation of noise and image, making it difficult to obtain higher-fidelity information on large datasets. Geiping et al. [8] were the first to push decision boundaries to ImageNet-level gradient inversion, but there is an issue with an unpredictable gradient averaging emergence. To the best of our knowledge, the most popular Deep Leakage from Gradients (DLG) [24] and Improved Deep Leakage from Gradients (iDLG) [25] methods are currently among the most effective methods for attacking gradient-based data leakage methods. The method generates virtual data with the same size as the training samples and labels at random and then uses the virtual data as the model input and error direction propagation to calculate the virtual gradient. The virtual data are learned using an optimization algorithm in such a way that the gradient obtained by backpropagation on the common model is similar to the real gradient, and the training data and labels are obtained after several rounds of iterative optimization. At the moment, this is one of the hottest topics in the study of variants of DLG-based methods [9, 52–54].

Although the DLG method has produced good attack results, its attack capability is limited, and the attack succeeds only when the predicted label is correct, and the model converges slowly; if the predicted label fails, the attack never converges, resulting in the DLG method's low accuracy. The improved iDLG, on the other hand, theoretically proves that data label prediction is independent of training data recovery under the condition that the activation function is a ReLU activation function or a sigmoid function and the loss value is calculated by cross-entropy, allowing the iDLG method to always attack successfully and thus recover the training data with high fidelity. Although the iDLG method produces good experimental results, the "Exploding Gradient" phenomenon occurs when training the model, resulting in slow model convergence and an undesirable large learning rate design strategy.

We analyze the causes of the more unstable training process of the DLG method and the improved iDLG method; the least squares method to calculate loss values typically suffers from the uncertainty of the gradient calculated by randomly initializing the weights. There may be multiple identical possible outcomes for the same pseudo-data input during iterative training of the model, and in these cases, the least-squares loss function aims for fuzzy prediction. When viewed through the lens of stochasticity, the original gradients computed by randomly initializing the model parameters suffer from a certain degree of randomness, as do the gradients of the parameters generated during the training process. This results in inefficient DLG method attacks, a lack of stability in model training, the "Exploding Gradient" in training data recovery, and undesirable design strategies for high learning rates.

3. WDLG Method

We give the structure of the WDLG method, as shown in Figure 1. Among the key problems, we need to solve is the prediction of the data labels and the construction based on the Wasserstein distance loss value calculation, which leads to the recovery of the training data. Therefore, we first describe the proof process of data label prediction, followed by the selection of the Wasserstein distance-based loss function, and finally the approximate solution of the Wasserstein distance.

3.1. Data Label Prediction. There is some randomness in the DLG method for the label recovery. In the iDLG method [25], it is found that shared gradients can leak the true labels, and for the labels encoded by unique heat, it is proposed that any differentiable model trained based on cross-entropy loss can extract the true labels effectively. Based on the iDLG approach, we give a more concise method for true label recovery. For the classification scenario, where the neural network model is usually trained using the cross-entropy loss of the unique heat-encoded true labels, it is defined as follows:

$$\text{Loss}(x, c) = -\log\left(\frac{e^{y_c}}{\sum_k^C e^{y_k}}\right) = -y_c + \log\left(\sum_k^C e^{y_k}\right), \tag{1}$$

where x denotes the training sample, C denotes the total number of classification categories, and y_c denotes the confidence that the training sample x is predicted to be the c-th class.

$$y_c = f\left(\sum_{j=1}^{t} y_j + b_c\right), \tag{2}$$

where t denotes the number of neurons in the penultimate hidden layer and $\sum_{j=1}^{t} y_j$ denotes the input of the c-th neuron in the last layer, whose corresponding bias is b_c. For the ReLU activation function, $f(x) = \max(0, x)$ and $f'(x) = 1$. The loss gradient for bias b_c is

$$\begin{aligned} g_i &= \frac{\partial \mathrm{Loss}(x, c)}{\partial b_i} = -\frac{\partial y_c}{\partial b_i} + \frac{\partial \log\left(\sum_k^C e^{y_k}\right)}{\partial b_i} = -\frac{\partial f\left(\sum_{j=1}^{t} y_j + b_c\right)}{\partial b_i} + \frac{\partial \log\left(\sum_k^C e^{f\left(\Sigma_{j=1}^{t} y_j + b_k\right)}\right)}{\partial b_i} \\ &= -\frac{\partial f\left(\sum_{j=1}^{t} y_j + b_c\right)}{\partial b_i} + \frac{e^{f\left(\sum_{j=1}^{t} y_j + b_c\right)}}{\sum_k^C e^{f\left(\sum_{j=1}^{t} y_j + b_k\right)}} \\ &= \begin{cases} -1 + \dfrac{e^{f\left(\sum_{j=1}^{t} y_j + b_c\right)}}{\sum_k^C e^{f\left(\sum_{j=1}^{t} y_j + b_k\right)}}, & \text{if } i = c, \\ \dfrac{e^{f\left(\sum_{j=1}^{t} y_j + b_c\right)}}{\sum_k^C e^{f\left(\sum_{j=1}^{t} y_j + b_k\right)}}, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

Since $e^{f(\sum_{j=1}^{t} y_j + b_c)} / \sum_k^C e^{f(\sum_{j=1}^{t} y_j + b_k)} \in (0, 1), g_i \in (-1, 0)$ when $i = c$, otherwise $g_i \in (0, 1)$.

Therefore, we can get the same conclusion of true label prediction as in the iDLG method, but we are more efficient in terms of computational complexity. We only need to obtain the negative bias term of the last layer of neurons to obtain the corresponding true labels of the input training sample x.

3.2. WDLG Loss Function.

The DLG method makes the virtual gradient approximate the original gradient by optimization, and in theory, the virtual data will also be closer to the training data. Given the gradient of the original training data, the training data are obtained by minimizing the following objective:

$$x'^{*}, y'^{*} = \underset{x', y'}{\operatorname{argmin}} \left\| \nabla W' - \nabla W \right\|^2 = \underset{x', y'}{\operatorname{argmin}} \left\| \frac{\partial \mathrm{Loss}\left(F\left(x', W\right), y'\right)}{\partial W} - \nabla W \right\|^2, \tag{4}$$

where $F(x', W)$ is a neural network parameterized by W with x' as input. The $\|\nabla W' - \nabla W\|^2$ distance is differentiable concerning the training data x' and the virtual label y' and thus can be optimized using a standard gradient-based approach.

Based on the superior performance of the Wasserstein distance, we propose the WDLG method and recompute the distance between $\|\partial \mathrm{Loss}(F(x', W), y')/\partial W - \nabla W\|^2$. Our WDLG method obtains the training data by minimizing the following objectives:

$$x'^{*} = \underset{x'}{\operatorname{argmin}} \mathscr{W}_d\left(\nabla W', \nabla W\right) = \underset{x'}{\operatorname{argmin}} \mathscr{W}_d\left(\frac{\partial \mathrm{Loss}\left(F\left(x', W\right), y'\right)}{\partial W}, \nabla W\right), \tag{5}$$

where $\mathscr{W}_d$ denotes the Wasserstein distance.

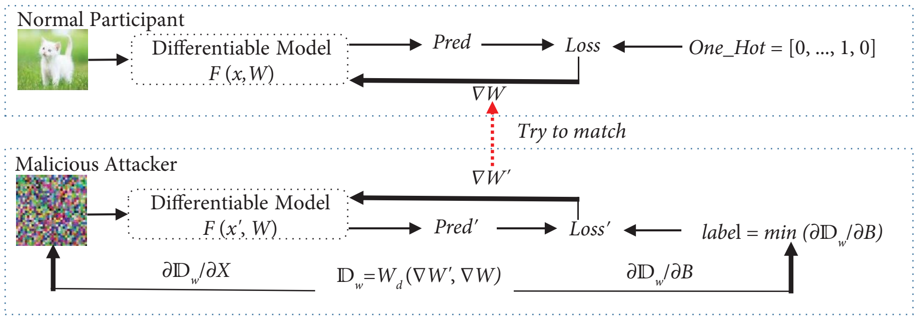

Figure 1: Overview of our WDLG algorithm. Normal participant inputs training sample $\mathbf{x}$ to obtain the original gradient $\nabla\mathbf{W}$, and malicious attacker randomly initializes the virtual data $\mathbf{x}'$ to obtain the virtual gradient $\nabla\mathbf{W}'$ and minimizes the Wasserstein distance between the original gradient $\nabla\mathbf{W}$ and the virtual gradient $\nabla\mathbf{W}'$. When the iterative optimization is completed, the virtual data $\mathbf{x}'$ randomly initialized by a malicious attacker converges to $\mathbf{x}$ with a preset threshold, thereby obtaining the training sample $\mathbf{x}$ from the normal participant.

We refer to the DLG [24] method and proposed the WDLG method in Figure 1, where the color "red" is not the same. The specific process is given as follows:

Step 1: normal participant inputs the training sample x into the differentiable deep network model F to obtain the predicted value Pred and at the same time calculates the label corresponding to the input sample according to equation (3), noting that the prediction of the label is independent of the training process of the whole network, and our method can recover 100% of the correct label. We record the gradient ∇W (the original gradient) of the real sample corresponding to each parameter.

Step 2: malicious attacker randomly initializes the virtual data x', which is sampled from the standard positive-terrestrial distribution. x' is used as the input of the deep network model F to obtain the virtual gradient $\nabla W'$ corresponding to the virtual data.

Step 3: we calculate the Wasserstein distance between $\nabla W'$ and ∇W to calculate the loss value according to equation (5) and iteratively update the input virtual data x' cyclically until the fidelity index MSE between x' and x reaches the preset threshold.

In Algorithm 1, $F(x, W)$ denotes the differentiable deep learning model, and the model is parametrically represented by W. $b_{c'}$ $(c' = 1, 2, \cdots, C)$ denotes the bias of the c' neuron in the output layer, and under the assumption of cross-entropy output, the subscript corresponding to the negative bias term, i.e., is the class of the sample prediction.

3.3. Wasserstein Distance Calculation.

The Wasserstein distance measure is widely used in econometrics, machine learning, and other fields. The proposal of the WGAN [55] model, in particular, has reignited a surge in its research and applications.

3.3.1. Wasserstein Distance Definition.

Let $\Pi(P, Q)$ be the set of all possible joint probability distributions of the combination of two probability distributions $P(\mathbf{x})$ and $Q(\mathbf{y})$, and for any joint probability distribution $\gamma(\mathbf{x}, \mathbf{y})$, the distance $d(\mathbf{x}, \mathbf{y})$ of the sample $(\mathbf{x}, \mathbf{y}) \sim \gamma$ distribution is defined as follows:

$$\begin{aligned} W(P, Q) &= \inf_{\gamma\sim\Pi(P,Q)} \iint \gamma(\mathbf{x}, \mathbf{y}) d(\mathbf{x}, \mathbf{y}), \\ &= \inf_{\gamma\sim\Pi(P,Q)} E_{(\mathbf{x},\mathbf{y})\sim\gamma}[d(\mathbf{x}, \mathbf{y})], \end{aligned} \tag{6}$$

where $d(\mathbf{x}, \mathbf{y})$ is a cost function. For each possible joint distribution γ, samples $\mathbf{x}$ and $\mathbf{y}$ can be obtained by sampling $(\mathbf{x}, \mathbf{y}) \sim \gamma$ from it and calculating the cost $d(\mathbf{x}, \mathbf{y})$ between the pair, so the expectation $E_{(\mathbf{x},\mathbf{y})\sim\gamma}[d(\mathbf{x}, \mathbf{y})]$ of the sample pair cost under this joint distribution γ can be calculated. The lower bound that can be taken on this expectation in all possible joint distributions is the Wasserstein distance. However, it is intuitive to interpret $E_{(x,y)\sim\gamma}[d(\mathbf{x}, \mathbf{y})]$ as the consumption required to move the P distribution to the Q distribution under this path planning of γ. Therefore, the Wasserstein distance is the minimum consumption under optimal path planning. Therefore, the optimal transmission Sinkhorn algorithm [56] is used to approximate the Wasserstein distance. Regularization encourages transmission using mostly small traffic paths while penalizing sparse paths. Therefore, introducing entropy regularization can limit the complexity of the solution of the optimal transmission problem, which leads to an approximate solution of the optimal transmission problem. To obtain the Sinkhorn algorithm, the solution of the regularized Kantorovich problem can be written in the following form:

$$P_{(i,j)} = u_i K_{i,j} v_j, \forall (i, j) \in [n] \times [m], \tag{7}$$

where $C_{i,j}$ denotes the cost of transferring a unit mass from a_i to b_j and (i, j) denotes the matrix subscript in the cost matrix C. $P = \mathrm{diag}(\mathbf{u}) K \mathrm{diag}(\mathbf{v})$, $(\mathbf{u}, \mathbf{v}) \in \mathbb{R}_+^n \times \mathbb{R}_+^m$, and $K_{i,j} = e^{-C_{i,j}/\varepsilon}$. ε is the regularization factor whose magnitude determines the strength of the regularization effect. The vectors $\mathbf{u}$ and $\mathbf{v}$ are the variables to be required by Sinkhorn's algorithm to join the mass conservation condition for optimal transmission.

$$\mathrm{a} = \mathbf{u} \odot \mathbf{K}\mathbf{v} \text{ and } \mathrm{b} = \mathbf{v} \odot \left(\mathbf{K}^T \mathbf{v}\right), \tag{8}$$

```
    Input: $\nabla W$: real gradient generated by real training sample $x$. Epochs: maximum number of iterations. $\eta$: learning rate.
(1)   $c' \leftarrow \min\{b_1, b_2, \cdots, b_C\} \leq 0$//Get the subscript $c'$ of the last layer of negative bias
(2) $c_{\text{pred}} \leftarrow$ One_Hot $(c')$//Convert $c'$ to One_Hot code
(3)   $x' \leftarrow \mathcal{N}(0,1)$//Initialize virtual data with the same dimensions as $x$
(4)   for $i \leftarrow 1$ to $N$ do
(5)     $\nabla \mathbf{W}' \leftarrow \partial \text{Loss}(F(x', \mathbf{W}), c_{\text{pred}})/\partial \mathbf{W}$//Calculating virtual gradients
(6)     $\text{Loss}_{\mathscr{W}_d} \leftarrow \text{WDCA}(\nabla \mathbf{W}', \nabla \mathbf{W})$//WDCA is Algorithm 2
(7)     $\mathbf{x}' \leftarrow \mathbf{x}' - \eta \nabla_{\mathbf{x}'} \text{Loss}_{\mathscr{W}_d}$
(8)   end for
    Output: $x'$
```

Algorithm 1: WDLG Algorithm.

```
     Input: input original gradient original_grad, virtual gradient du mmy_grad.
(1)    Initialization $u^0, v^0$
(2)    $C \leftarrow$ cost_matrix (original_grad, du mmy_grad)//Calculate the cost matrix $C$
(3)    $K \leftarrow e^{-C/\varepsilon}$
(4)    $a \leftarrow u^0 \odot \mathbf{K} v^0$
(5)    $b \leftarrow v^0 \odot (\mathbf{K}^T u^0)$
(6)    for $i \leftarrow 0$ to max _iter do
(7)      $u^{t+1} \leftarrow (a/\mathbf{K} v^t) + \mathbf{u}^t$
(8) $v^{t+1} \leftarrow (b/\mathbf{K}^T u^{t+1}) + \mathbf{v}^t$
(9)      If sum(abs$(u^{t+1} - u^t)) \leq$ err_thresh, then
(10)       $u^* \leftarrow u^{t+1}$
(11)       $v^* \leftarrow v^{t+1}$
(12)       break
(13)     end if
(14)   end for
(15)   $\mathscr{W}_d \leftarrow \mathbb{D}(C, u^*, v^*)$//$\mathbb{D}$ obtains an approximation of the Wasserstein distance $\mathscr{W}_d$
     Output: $\mathscr{W}_d$
```

Algorithm 2: Wasserstein distance calculation algorithm (WDCA).

where $\odot$ is the Hadamard product of the vector. Therefore, we can solve for $\mathbf{u}$ and $\mathbf{v}$ iteratively. $\mathbf{u}$ is updated at each step, then $\mathbf{v}$ is updated, and finally, the iterations converge and both sides of the equation are satisfied simultaneously, and we get the following iterative equation:

$$\mathbf{u}^{t+1} = \frac{a}{\mathbf{K}\mathbf{v}^t}, \mathbf{v}^{t+1} = \frac{b}{\mathbf{K}^T \mathbf{u}^{t+1}}. \tag{9}$$

In algorithm 2, we give an approximate solution algorithm WDCA for $\mathscr{W}_d$.

4. Experiments

In purpose to give comparative experiments with fairness, we choose PyTorch [57] as the experimental platform and give the same network model and hyperparameter settings uniformly except for different algorithms. In Algorithm 1, batch_size = 1, regularization factor $\varepsilon = 0.01$, maximum number of iterations max _iter = 100, and error threshold $\text{err}_\text{t}\text{hresh} = 0.1$ are given. We perform comparative validation and analysis of DLG, iDLG, and WDLG methods on MNIST, Fashion MNIST, CIFAR-100, SVHN, and LFW public datasets for accuracy of data label prediction, stability and convergence speed of virtual data training to recover data fidelity (MSE indicator), and large learning rate design strategies.

4.1. Comparison of the Accuracy of Predicted Labels. The DLG method that uses the least squares method to calculate the gradient loss values for data label recovery is initialized with a randomization method for iterative learning, and there is certain randomness in its ability to recover data labels, which makes the attack efficiency (accuracy of label prediction) not high. In contrast, the iDLG method theoretically demonstrates that under the condition of using the ReLU activation function as well as cross-entropy to calculate the loss value, the label prediction is independent of the network iterative training and the training labels can be fully recovered.

In our WDLG method, it is assumed that the last hidden layer of the deep network model has N neurons and the training dataset has C categories. According to the basic premise assumption of the iDLG method, it is further assumed that the i-th neuron of the output layer has bias b_i. We can still prove theoretically that the recovery of data labels is independent of the iterative training process of the network and refer to the proof of (equation (3)) for details. This allows the WDLG method to always attack successfully and thus recover the training data with high fidelity. In Table 1, we give a comparison of the accuracy of DLG, iDLG, and WDLG in predicting the true labels. Both WDLG and iDLG methods have a 100% success rate in predicting the data labels, while DLG often makes errors in predicting the true labels. However, for the computational time complexity of prediction of data labels, the computation step iDLG_Num of the iDLG method is much higher than that of our WDLG method, especially when the number N of neurons in the last hidden layer is very large; the advantage of our method is more significant.

In general, $C \ll N$. The number of computational steps for the iDLG method to recover the labels is $N \times C$, while our method requires only C computational steps. Therefore, it can be theoretically demonstrated that the computational time complexity of the WDLG method on the task of recovering labels is only $1/N$ that of the iDLG method, which is significant from the computational analysis, as shown in Table 1 for detailed comparison. Our proposed data label recovery method assumes a stronger conditional generalization capability.

4.2. Comparison of the Stability and Convergence Speed of Model Training. The training process of the DLG method is less stable, and the loss value is often very large in the initial iteration step, which makes the whole iterative computation process difficult to converge. The improved iDLG method uses the cross-computed loss function and also uses the least squares method to calculate the gradient loss value, which still suffers from the difficulty of convergence of the loss value. Analyzing the main reason, the least squares method to calculate the loss values usually suffers from the uncertainty of the gradient calculated by random initialization weights. When iteratively training the model, there may be various equally possible results for the same random data input. In all these cases, the least squares loss function aims to accommodate the uncertainty in the prediction by fuzzy prediction, or simply, by taking the average of the possible outputs. This is because the average of all possible outcomes will lead to overall minimization of the parameter space of the inputs during training.

For the comparison of model training stability and model convergence speed experiments, we experimentally validate the DLG, iDLG, and WDLG methods on simple datasets with a single feature, MNIST, Fashion MNIST, and datasets with complex features, SVHN, CIFAR-100, and LFW. In Figure 2, the visualization of depth leakage on MNIST and Fashion MNIST (from top to bottom) images is shown separately; the first image is the original training image, and from left to right are the recovered views of training data on the same number of iterations for DLG, iDLG, and WDLG methods, respectively. On the images with clear training images and single features, there is not much difference in the number of iterations between DLG and iDLG methods to achieve the same image fidelity quality, while our method WDLG only requires fewer iterations, and the model converges faster. With the same number of iterations, our method recovers more realistic training data with higher fidelity.

To compare the comparison on feature-complex datasets, in Figure 3, the visualization of depth leakage on SVHN, CIFAR-100, and LFW (top-to-bottom) images is shown separately, with the first image being the original training image, and from left to right, the recovered data views of the DLG, iDLG, and WDLG methods on the same number of iterations, respectively, are given. On the training image with insufficient sharpness and feature-rich images, the number of iterations required by the DLG, iDLG, and WDLG methods increases substantially to achieve the same image fidelity quality. However, our method requires fewer iterations, and the model training still maintains a significant advantage in terms of convergence speed.

In particular, we note that the loss values and fidelity during training are carried out as follows to compare the experimental results better; the output loss values and fidelity are taken logarithmically and then shifted up, and such operation does not change the nature of the function. The purpose is to make a better comparison in detail on the same graph. Finally, schematic diagrams of the relationship between loss value and the number of iterations, fidelity and number of iterations, and fidelity and loss value on complex feature datasets are given, respectively. In Figure 4, the relationship between loss value and fidelity versus the number of iterations for DLG, iDLG, and WDLG methods on three datasets, SVHN, CIFAR-100, and LFW, is given. In the (a), (b), and (c) plots, our WDLG method has the smallest loss value and better convergence than the other two methods on the same iteration steps. In the (d), (e), and (f) plots, our WDLG method has the smallest MSE and higher fidelity of recovered data on the same iteration steps. In the (g), (h), and (i) plots, to achieve the same image fidelity and loss values, the DLG method requires the most iteration steps, the iDLG method is the second, and our WDLG method requires the least iteration steps. The proposed WDLG method consistently outperforms the DLG and iDLG methods in recovering data and has significant advantages in all three tasks. In the plot of (f), the advantage of WDLG is significant on the difficult task of LFW, with MSE close to 0 and higher fidelity of recovered data.

Through the abovementioned analysis, to solve the problems existing in the least squares method to calculate the loss value, the proposed Wasserstein distance to calculate the loss value plays a relatively good and stable training effect in the training process, and the main reason for its success, analyzed from the perspective of randomness, the original gradient ∇W calculated by randomly initializing the model parameters also has a certain degree of randomness, and the parameter gradient generated in the training process $\nabla W'$ also has some randomness. The aim is to better calculate the distance between two random variables. This is because the KL-divergence does not satisfy the symmetry of the distance definition, and the JS-divergence always keeps the distance as $\log 2$ when the two distributions do not overlap at all, leading to difficulties in providing an effective gradient calculation during the model training process. The key to solving the problem lies in using the Wasserstein distance to measure the distance between two distributions, and the WGAN [55] theoretically demonstrates that the Wasserstein

Table 1: Comparison of DLG, iDLG, and WDLG prediction labels and prediction complexity.

Datasets	DLG	iDLG	iDLG_Num	WDLG (%)	WDLG_Num
MNIST	89.9%	100.0%	$N \times C$	100.0	C
Fashion MNIST	×	×	$N \times C$	100.0	C
SVHN	×	×	$N \times C$	100.0	C
CIFAR-100	83.3%	100.0%	$N \times C$	100.0	C
LFW	79.1%	100.0%	$N \times C$	100.0	C

The accuracy of DLG [24], iDLG [25], and WDLG in predicting data labels. Note that iDLG and WDLG always predict the correct labels, while DLG often makes errors in predicting the true labels.

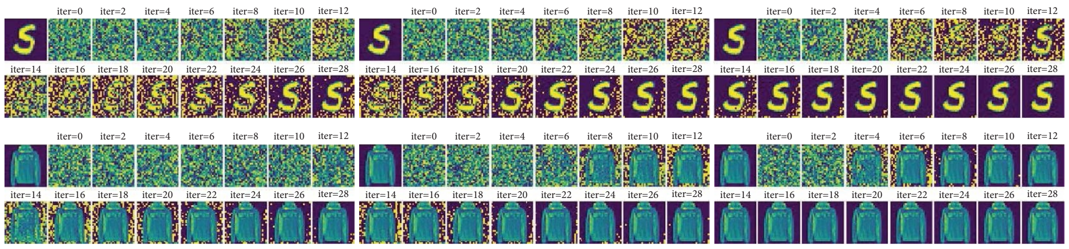

Figure 2: Comparison of the stability of model training on a dataset with a single feature.

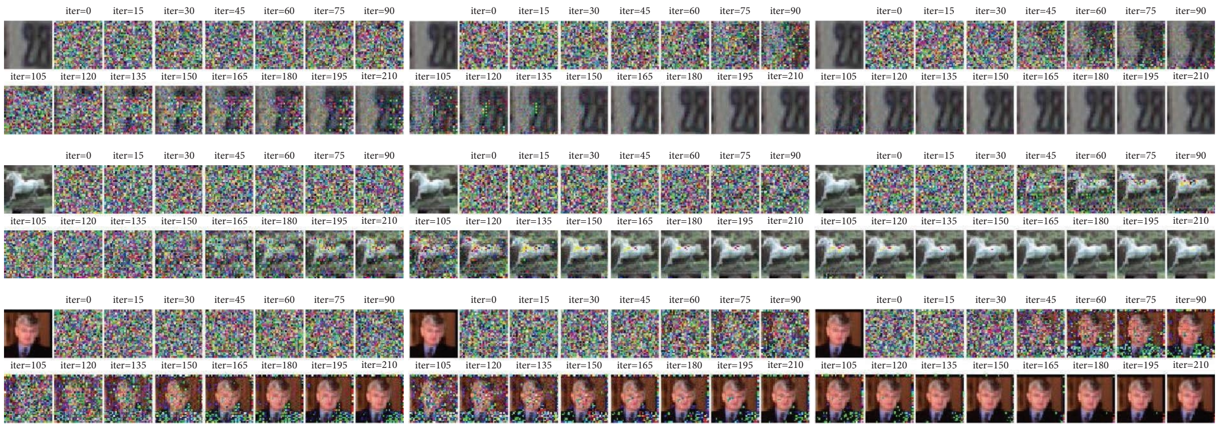

Figure 3: Comparison of stability of model training on datasets with complex features.

distance can reflect the distance between two distributions and provide an effective gradient calculation even if they do not have any overlap. Therefore, our proposed WDLG method can better assign singular values to other random components when singular values appear in the iterative gradient calculation $\nabla W'$ and thus will be more stable for model training.

4.3. Comparison of Large Learning Rate Training. To analyze our WDLG method supporting large learning rate strategy design in detail, we test and validate the loss value vs. the number of iterations and fidelity vs. the number of iteration curves at different learning rates only on the complex dataset LFW.

In (a) of Figure 5, the loss value decreases with increasing the learning rate, achieving a surprising effect. However, we should pay more attention to the existence of an inflection point in the loss function curve for lr = 0.01 and lr = 0.1 at iterations < 20, which causes an abrupt change in the slope of the curve and is one of the reasons for the difficulty in the convergence of the model training, corresponding to the difficulty in the convergence of the recovered data fidelity (MSE) at a finite number of iterations in (b). However, for the cases of lr = 0.5 and lr = 1, especially at lr = 1, the rate of decline of the loss function curve is quite stable throughout the training process, and there is no case of singularity, and the corresponding MSE curve decreases steadily, and the fidelity continues to increase. Therefore, we have reasons to believe that our WDLG method can support a large learning strategy design.

In the course of our experiments, we also tried to provide larger learning rates to the DLG and iDLG methods, but they failed frequently to recover the fidelity of the data. To analyze the reason, the optimization path direction is dominated by the gradient in the direction of the singular value when the gradient $\nabla W'$ of the iterative computation appears singular, resulting in the optimization path deviating from the feasible solution path. Therefore, the learning rate of DLG and iDLG

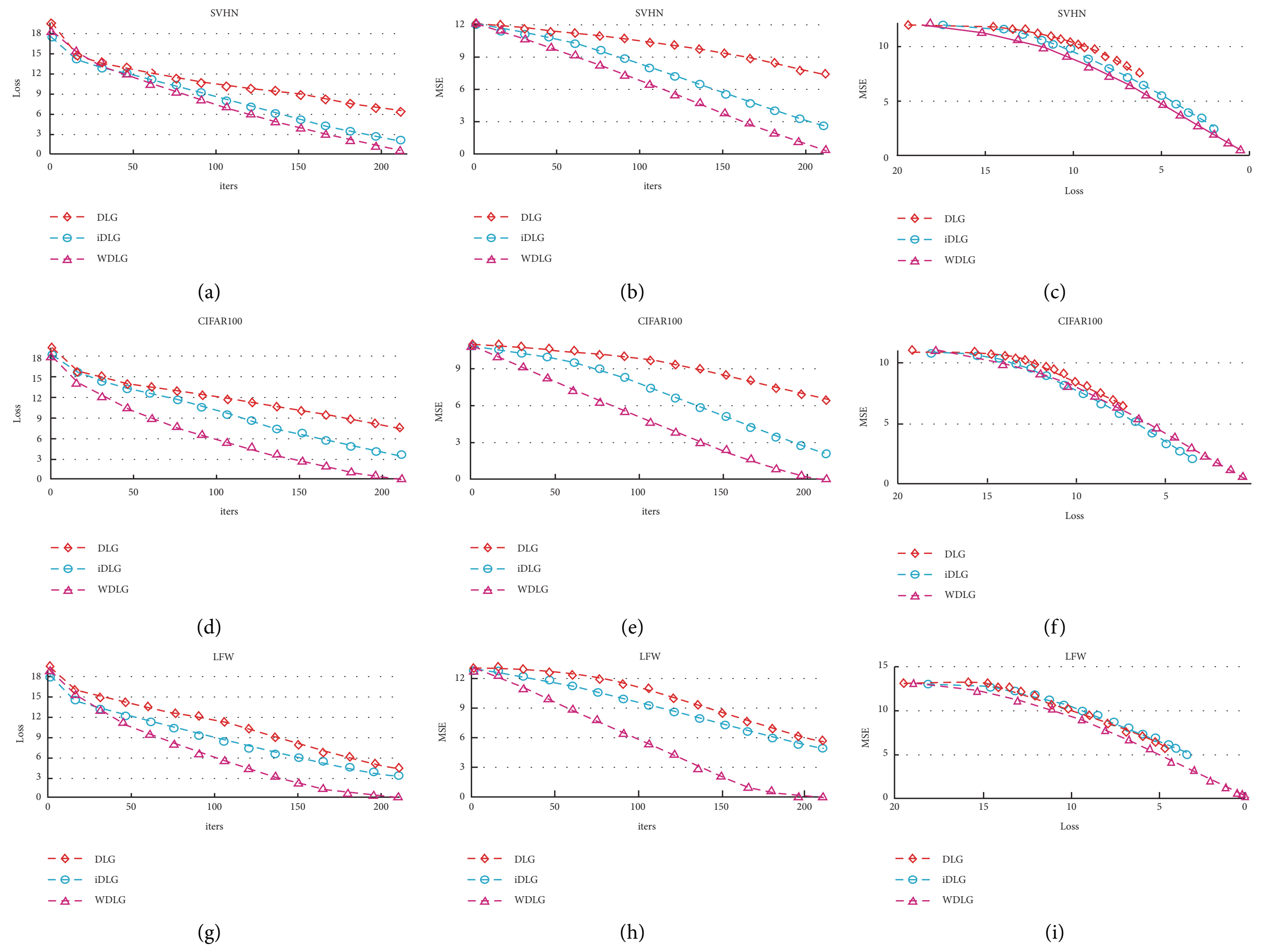

Figure 4: The relationship between loss values and fidelity versus the number of iterations for DLG, iDLG, and WDLG methods on three datasets, SVHN, CIFAR-100, and LFW.

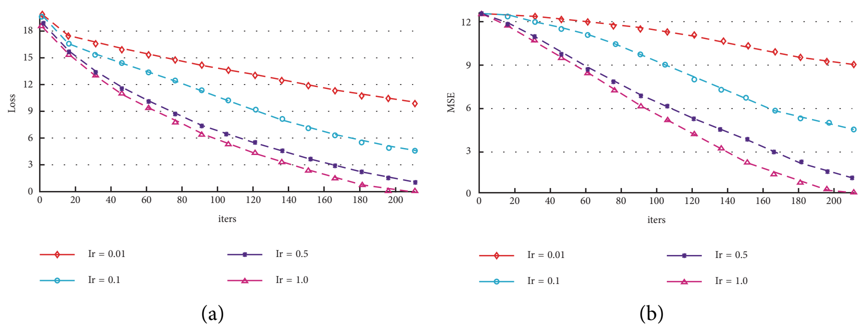

Figure 5: Variation of loss value and fidelity curves by changing the learning rate lr strategy on the LFW dataset.

methods is set too large to cause the network to fail to converge or to hover around the feasible solution, thus neglecting the location of the feasible solution.

In our WDLG method, the iterative generation of the singular gradient $\nabla W'$ assigned to other random components of the distance calculation is still the main reason why we can set a larger learning rate successfully. We still try to provide a larger learning rate during the training of the network, and Figure 5 shows our experimental results. In (a), it is clear that the loss value decreases faster with increasing the learning rate on the same iteration step, while MSE, which portrays the fidelity metric, also shows a significant decrease in (b), so we have reasons to believe that our WDLG method for learning rate generalization ability is stronger. Although the larger the learning rate, the faster the model training reaches the optimal value, the learning rate increases to a certain threshold, and the whole network no longer converges. Therefore, in our approach, the design of a large learning rate strategy is encouraged and the stability of model training should be considered more.

5. Conclusion

In this paper, a WDLG method is proposed, and the theoretical analysis shows that the prediction of data labels is independent of the learning process of the virtual data used, and the labels of the data can be recovered accurately. In the proposed method, the Wasserstein distance is used to calculate the error loss between the shared gradient and the virtual gradient, which solves the stable model training and "Exploding Gradient" phenomena and improves the fidelity of the recovered data. Meanwhile, it is experimentally demonstrated that our method fully supports large learning rate strategies and therefore improves the convergence speed of model training.

Based on this research, we will consider extending the WDLG method to deeper network models and more complex training datasets in the future. Meanwhile, a more secure machine learning algorithm based on differential privacy is proposed based on the attack capability of the WDLG method. The purpose is to allow researchers to reacquaint themselves with the hidden potential security issues in distributed ML, to provide thoughts on security research for the next generation of AI model training, and to provide a research idea to solve the conflict between AI applications and security.

Data Availability

The data used to support the finding of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the National Key Research and Development Program of China (grant no. 2022YFB2701401), the National Natural Science Foundation of China (grant no. 62272124), the Guizhou Science Contract Plat Talent (grant no. [2020]5017), the projects of the Education Department of Guizhou province (grant no. [2018]140), the projects of the Education Department of Guizhou Province (grant no. [2018]141), the Research Project of Guizhou University for Talent Introduction (grant no. [2020]61), the Cultivation Project of Guizhou University (no. [2019]56), and the Open Fund of Key Laboratory of Advanced Manufacturing Technology, Ministry of Education (grant no. GZUAMT2021KF[01]).

References

[1] B. McMahan, E. Moore, and D. Ramage, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics PMLR*, pp. 1273–1282, 2017.

[2] J. Konečný, H. B. McMahan, and F. X. Yu, "Federated learning: strategies for improving communication efficiency," 2016, https://arxiv.org/abs/1610.05492.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[4] T. Li, A. K. Sahu, and M. Zaheer, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[5] P. Kairouz, H. B. McMahan, B. Avent et al., "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[6] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning," in *Proceedings of the 2019 IEEE symposium on security and privacy (SP)*, pp. 739–753, San Francisco, CA, USA, May 2019.

[7] T. Ha and T. K. Dang, "Inference attacks based on GAN in federated learning," *International Journal of Web Information Systems*, vol. 18, no. 2/3, pp. 117–136, 2022.

[8] J. Geiping, H. Bauermeister, and H. Dröge, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16937–16947, 2020.

[9] J. Geng, Y. Mou, and F. Li, "Towards general deep leakage in federated learning," 2021, https://arxiv.org/abs/2110.09074.

[10] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2018.

[11] M. Abadi, A. Chu, and I. Goodfellow, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, Vienna, Austria, October 2016.

[12] K. Bonawitz, H. Eichner, and W. Grieskamp, "Towards federated learning at scale: system design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[13] D. Li and W. J. Fedmd, "Heterogenous federated learning via model distillation," in *Proceedings of the NeurIPS Workshop Feder. Learn. Data Privacy Confidentiality*, vol. 1–8, Vancouver, Canada, December 2019.

[14] Y. Zhang, Y. Liang, B. Jia, P. Wang, and X. Zhang, "A blockchain-enabled learning model based on distributed deep learning architecture," *International Journal of Intelligent Systems*, vol. 37, no. 9, pp. 6577–6604, 2022.

[15] L. Melis, C. Song, and E. De Cristofaro, "Inference attacks against collaborative learning," p. 13, 2018, https://arxiv.org/abs/1805.04049.

[16] L. Melis, C. Song, and E. De Cristofaro, "Exploiting unintended feature leakage in collaborative learning," in *Proceedings of the 2019 IEEE symposium on security and privacy (SP)*, pp. 691–706, IEEE, California, CA, USA, May 2019.

[17] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic

countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, Colorado, CO, USA, October 2015.

[18] R. Shokri, M. Stronati, and C. Song, "Membership inference attacks against machine learning models," in *Proceedings of the 2017 IEEE symposium on security and privacy (SP)*, pp. 3–18, IEEE, California, CA, USA, May 2017.

[19] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "LOGAN: membership inference attacks against generative models," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 133–152, 2019.

[20] A. Salem, A. Bhattacharya, and M. Backes, "Updates-leak: data set inference and reconstruction attacks in online learning," in *Proceedings of the 29th USENIX security symposium (USENIX Security 20)*, pp. 1291–1308, Berkeley, CA, USA, August 2020.

[21] C. G. Peng, "Artificial intelligence security governance challenges and countermeasures," *Journal of Information Security Research*, vol. 8, no. 04, pp. 318–325, 2022.

[22] A. Mahendran and A. Vedaldi, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, 2016.

[23] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: going deeper into neural networks," *Google Research Blog*, 2015.

[24] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] B. Zhao, K. R. Mopuri, and H. Bilen, "Idlg: improved deep leakage from gradients," 2020, https://arxiv.org/abs/2001.02610.

[26] Y. LeCun, C. Cortes, and C. J. C. Burges, "The mnist database of handwritten digits," 1998, https://idr.openmicroscopy.org/.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017, https://arxiv.org/abs/1708.07747.

[28] Y. Netzer, T. Wang, and A. Coates, "Reading digits in natural images with unsupervised feature learning," *In NIPS Workshop*, vol. 5, 2011.

[29] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's Thesis, University of Toronto, Canada, 2009.

[30] G. B. Huang, M. Mattar, and T. Berg, "Labeled faces in the wild: a database forstudying face recognition in unconstrained environments," *Technical Report 07–49*, University of Massachusetts, MA, USA, 2007.

[31] B. Recht, C. Re, and S. Wright, "Hogwild: a lock-free approach to parallelizing stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[32] Q. Ho, J. Cipar, and H. Cui, "More effective distributed ml via a stale synchronous parallel parameter server," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[33] X. Lian, C. Zhang, and H. Zhang, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[34] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," 2018, https://arxiv.org/abs/1802.05799.

[35] M. Abadi, P. Barham, and J. Chen, "TensorFlow a system for Large-Scale machine learning," in *Proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, Savannah, GA, USA, November 2016.

[36] X. Li, Q. Ding, and J. Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[37] J. Yuan, X. Li, and C. Cheng, "Oneflow: redesign the distributed deep learning framework from scratch," 2021, https://arxiv.org/.

[38] F. N. Iandola, S. Han, and M. W. Moskewicz, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," 2016, https://arxiv.org/abs/1602.07360.

[39] W. Wen, C. Xu, and F. Yan, "Terngrad: ternary gradients to reduce communication in distributed deep learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[40] S. Wang, A. Pi, X. Zhou, J. Wang, and C. Z. Xu, "Overlapping communication with computation in parameter server for scalable DL training," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2144–2159, 2021.

[41] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 117–124, 2009.

[42] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016, https://arxiv.org/abs/1611.01144.

[43] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[45] S. Yeom, I. Giacomelli, and M. Fredrikson, "Privacy risk in machine learning: analyzing the connection to overfitting," in *Proceedings of the 2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282, Oxford, UK, July 2018.

[46] A. Salem, Y. Zhang, and M. Humbert, "Ml-leaks: model and data independent membership inference attacks and defenses on machine learning models," 2018, https://arxiv.org/abs/1806.01246.

[47] K. Leino and M. Fredrikson, "Stolen memories: leveraging model memorization for calibrated white-box membership inference," in *Proceedings of the 29th USENIX security symposium (USENIX Security 20)*, pp. 1605–1622, Boston, MA, USA, August 2020.

[48] L. Song, R. Shokri, and P. Mittal, "Privacy risks of securing machine learning models against adversarial examples," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 241–257, London, UK, November 2019.

[49] I. J. Goodfellow, J. Pouget-Abadie, and M. Mirza, "Generative adversarial nets," *Stat*, vol. 1050, p. 10, 2014.

[50] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 603–618, Dallas, TX, USA, November 2017.

[51] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, "A method of information protection for collaborative deep learning under gan model attack," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 871–881, 2021.

[52] X. Pan, M. Zhang, and Y. Yan, "Theory-oriented deep leakage from gradients via linear equation solver," 2020, https://arxiv.org/abs/2010.13356.

[53] Z. Li, M. Hubchak, and Y. Zhu, "Deep leakage from gradients in multiple-label medical image classification," *IEEE*, in *Proceedings of the 2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pp. 447-448, Victoria, BC, Canada, August 2021.

[54] Z. Luo, C. Zhu, L. Fang, G. Kou, R. Hou, and X. Wang, "An effective and practical gradient inversion attack," *International Journal of Intelligent Systems*, vol. 37, no. 11, pp. 9373–9389, 2022.

[55] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the International conference on machine learning*, pp. 214–223, Sydney, Australia, August 2017.

[56] M. Cuturi, "Sinkhorn distances: lightspeed computation of optimal transport," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[57] A. Paszke, S. Gross, and S. Chintala, "Automatic differentiation in PyTorch," *In NIPS-W*, vol. 3, p. 6, 2017.