Gradients Stand-in for Defending Deep Leakage in Federated Learning

1st Yi Hu
Department of Computer Science
Swansea University
Swansea, United Kingdom
845700@swansea.ac.uk

2nd Hanchi Ren*
Department of Computer Science
Swansea University
Swansea, United Kingdom
hanchi.ren@swansea.ac.uk
*Corresponding author

3th Chen Hu
Department of Computer Science
Swansea University
Swansea, United Kingdom
2100552@swansea.ac.uk

4th Yiming Li
Department of Computer Science
Swansea University
Swansea, United Kingdom
946802@swansea.ac.uk

5rd Jingjing Deng
Department of Computer Science
Durham University
Durham, United Kingdom
jingjing.deng@durham.ac.uk

6th Xianghua Xie
Department of Computer Science
Swansea University
Swansea, United Kingdom
x.xie@swansea.ac.uk

***Abstract*—Federated Learning (FL) has become a cornerstone of privacy protection, shifting the paradigm towards localizing sensitive data while only sending model gradients to a central server. This strategy is designed to reinforce privacy protections and minimize the vulnerabilities inherent in centralized data storage systems. Despite its innovative approach, recent empirical studies have highlighted potential weaknesses in FL, notably regarding the exchange of gradients. In response, this study introduces a novel, efficacious method aimed at safeguarding against gradient leakage, namely, "AdaDefense". Following the idea that model convergence can be achieved by using different types of optimization methods, we suggest using a local stand-in rather than the actual local gradient for global gradient aggregation on the central server. This proposed approach not only effectively prevents gradient leakage, but also ensures that the overall performance of the model remains largely unaffected. Delving into the theoretical dimensions, we explore how gradients may inadvertently leak private information and present a theoretical framework supporting the efficacy of our proposed method. Extensive empirical tests, supported by popular benchmark experiments, validate that our approach maintains model integrity and is robust against gradient leakage, marking an important step in our pursuit of safe and efficient FL.**

***Index Terms*—Federated Learning, Data Privacy, Gradient Leakage, Distributed Learning, Deep Neural Network**

I. Introduction

The remarkable advancements in Deep Neural Network (DNN) across diverse Deep Learning (DL) tasks have significantly impacted everyday life. This progress largely depends on the availability of extensive training datasets. In recent years, there has been a surge in data creation, including much sensitive private data that cannot be shared due to privacy concerns. The implementation of the General Data Protection Regulation (GDPR) in Europe underscores this by aiming to protect personal data integrity and regulate data exchange.

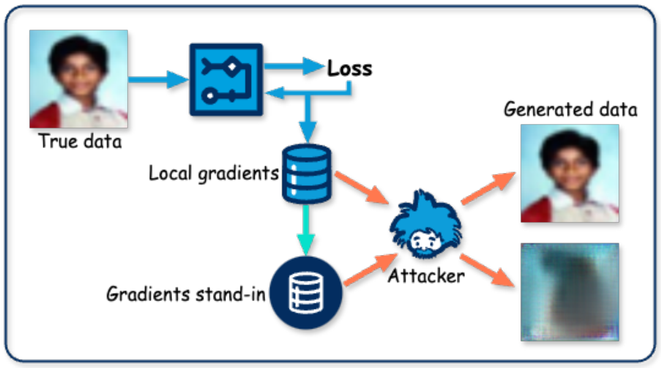


Fig. 1. Illustration of AdaDefense against gradient leakage

Consequently, DL models leveraging such datasets have enhanced the capabilities of numerous applications. Given these developments, exploiting all available data for training models in a legally compliant manner has gained substantial traction. This ensures the utility of expansive datasets while adhering strictly to privacy regulations.

For this purpose, Federated Learning (FL) [1]–[4] is employed as a decentralized model training framework where the central server accumulates local gradients from clients without requiring the exchange of local data. This approach ensures that data remains at its source, ostensibly preserving privacy because only gradient information is shared. Traditionally, these gradients have been assumed secure for sharing, making FL a prime application in privacy-sensitive scenarios. Nonetheless, recent studies have exposed vulnerabilities within this gradient-sharing framework. L. Zhu, Z. Liu and S. Han [5] introduced Deep Leakage from Gradients (DLG), a technique for reconstructing the original training data by iteratively optimizing randomly initialized input images and labels to match the shared gradients, rather than updating the model parameters directly. However, the performance of DLG is limited by factors such as the training batch size and the resolution of images, which can lead to instability in data recovery. To address these instabilities, J. Geiping, H. Bauermeister, H. Dröge and M. Moeller [6] proposed Inverting Gradient (IG), which utilizes a magnitude-invariant cosine similarity as the loss function to enhance the stability of training data reconstruction. This method has proven effective in recovering high-resolution images ($224 * 224$ pixels) from gradients of large training batches (up to 100 samples). Further refining the approach, Improved DLG (iDLG) by B. Zhao, K.R. Mopuri and H. Bilen [7] simplifies the label recovery process within DLG by analytically deriving the ground-truth labels from the gradients of the loss function relative to the *Softmax* layer outputs, thus improving the precision of the reconstructed data. Expanding on generative methods, H. Ren, J. Deng and X. Xie [8] developed Generative Regression Neural Network (GRNN), which integrates two generative model branches: one, a Generative Adversarial Network (GAN)-based branch for creating synthetic training images, and another, a fully-connected layer designed to generate corresponding labels. This model facilitates the alignment of real and synthetic gradients to reveal the training data effectively. Recently, Z. Li, J. Zhang, L. Liu and J. Liu [9] introduced Generative Gradient Leakage (GGL), a novel method that also utilizes a pre-trained GAN to generate fake data. This approach leverages the iDLG concept to deduce true labels and adjusts the GAN input sequence based on the gradient matching, thereby producing fake images that closely resemble the original training images. The paper [10] introduces a novel attack method named Gradient Leakage Attack Using Sampling (GLAUS) which targets Unbiased Gradient Sampling-based (UGS) secure aggregation methods in FL. These UGS, e.g. MinMax Sampling (MMS) [11], methods are designed to enhance security by using unbiased random transformations and gradient sampling to prevent direct access to the real gradients during model training, thereby obscuring private client data. These developments underscore the ongoing need to enhance the security measures in FL systems against sophisticated attacks that seek to exploit gradient information, thus compromising the privacy of sensitive data. The continuous evolution of defensive techniques is crucial to safeguarding data integrity in federated environments.

These issues prompted us to evaluate the reliability of the FL system. Concerns about gradient leakage have triggered various defense strategies, such as gradient perturbation [5], [8], [12]–[14], data obfuscation or sanitization [15]–[19], along with other techniques [20]–[24]. Nonetheless, these methods generally involve compromises between privacy protection and computational efficiency, often requiring substantial computational resources due to the complexities of encryption technologies. The study in [25] delves into the factors in an FL system that could influence gradient leakage, such as batch size, image resolution, the type of activation function used, and the number of local iterations prior to the exchange of gradients. Various studies corroborate these findings; for instance, the DLG method indicates that activation functions need to be bi-differentiable. IG techniques can reconstruct images with resolutions up to $224 * 224$, whereas GRNN are compatible with batch sizes of 256 and image resolutions of $256 * 256$, also assessing how the frequency of local iterations might impact privacy leakage. Moreover, recent advancements by D. Scheliga, P. Mäder and M. Seeland [26] introduced an innovative model extension, named Privacy Enhancing Module (PRECODE), designed specifically to enhance privacy safeguards against potential leakages in FL systems. This adaptability highlights the evolving trend of FL security measures that seek to enhance the balance between performance and privacy without imposing excessive computational requirements.

In traditional model training strategies, various optimization algorithms can alter the low-level representations of gradients while preserving their high-level representations. For instance, unlike Stochastic Gradient Descent (SGD) which utilizes raw gradients for model updates [27], Adam [28] adjusts gradient values using adaptive estimates based on first-order and second-order moments. This work introduces an alternative approach by employing gradients stand-in for global gradient aggregation, aimed at preventing the transmission of private information to the parameter server (refer to Fig. 1). After an epoch of local training, the local gradients are modified through the Adam optimization algorithm, which utilizes adaptive estimates of lower-order moments. Critical information, such as first-order and second-order moments, is retained locally, thereby rendering it impossible for an adversary to execute the forward-backward procedure without access to this data. Consequently, reconstructing local training data from the shared gradients becomes infeasible within the FL framework. The gradients stand-in calculated in this manner not only mitigate the risk of gradient leakage but also preserve model performance. Additionally, the simplicity and computational efficiency of the method, make it particularly attractive. The versatility of the approach is evident as it does not impose constraints on the model architecture, the local training optimization strategy, the dataset, or the FL aggregation technique. Comprehensive evaluations on various benchmark networks and datasets have demonstrated that our method, termed AdaDefense, effectively addresses the identified privacy concerns. The implementation of AdaDefense is available on Github[1].

The structure of this paper is organized as follows: Section II reviews related work on gradient leakage attacks along with existing defense mechanisms. Section III details the proposed method for defending against gradient leakage. Experimental results are presented in Section IV, where we compare the efficacy of our proposed AdaDefense with state-of-the-art attack methods. Finally, Section V concludes the paper and

[1]https://github.com/Rand2AI/AdaDefense

outlines future research directions.

II. Related Work

A. Attack Methods

Gradient leakage attacks pose a significant threat to privacy in both centralized and collaborative DL systems by potentially exposing private training data through leaked gradients. Such attacks are particularly prevalent in centralized systems, often through membership inference attacks as discussed by [29]–[34].

The work [5] was pioneer in examining data reconstruction from leaked gradients within collaborative systems, an effort furthered by [6] which introduced an optimization-based method to enhance the stability of such attacks. They utilized magnitude-invariant cosine similarity measurements for the loss function and demonstrated that incorporating prior knowledge could significantly augment the efficiency of gradient leakage attacks. Expanding on these findings, the work [35] argued that gradient information alone might not suffice for revealing private data, thus they proposed the use of a pre-trained model, GIAS, to facilitate data exposure. In [36], the authors found that in image classification tasks, ground-truth labels could be easily discerned from the gradients of the last fully-connected layer, and that Batch Normalization (BN) statistics markedly enhance the success of gradient leakage attacks by disclosing high-resolution private images.

Alternative approaches involve generative models, as explored in [37], which employed a GAN-based method for data recovery that replicates the training data distribution. Similarly, the work [38] developed mGAN-AI, a multitask discriminator-enhanced GAN, to reconstruct private information from gradients. GRNN, introduced [8], is capable of reconstructing high-resolution images and their labels, effectively handling large batch sizes. Likewise, in the work of GGL [9], a GAN with pre-trained and fixed weights was used. GGL differs in its use of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) as the optimizer, reducing variability in the generated data, which, while not exact replicas, closely resemble the true data. This characteristic significantly enhances GGL's robustness, allowing it to counteract various defensive strategies including gradient noising, clipping, and compression. This series of developments highlights the dynamic and evolving nature of combating gradient leakage attacks in DL.

GLAUS [10] demonstrates a significant vulnerability in UGS methods by showing that they can still be susceptible to gradient leakage attacks. This is done by reconstructing private data points with considerable accuracy despite the supposed robust security measures of UGS. GLAUS circumvents the safeguards by approximating the gradient using leaked indices and signs from the UGS's crafted random transformations. It effectively reduces the security of UGS frameworks to that of basic FL models without additional protections. It is capable of adaptively inferring the gradient without needing the exact gradient, utilizing an approximate gradient reconstructed through several steps. These include narrowing the gradient search range, estimating the magnitude of each gradient value, and revising the gradient signs. This method showcases that even when real gradients are not directly accessible, sensitive data can still be reconstructed, posing a serious security risk.

B. Defense Methods

Numerous strategies [34] have been developed to safeguard private data against potential leakage via gradient sharing in FL. Techniques such as gradient perturbation, data obfuscation or sanitization, Differential Privacy (DP), Homomorphic Encryption (HE), and Secure Multi-Party Computation (MPC) have been employed to protect both the private training data and the publicly shared gradients [21].

The work in [5] assessed the efficacy of Gaussian and Laplacian noise types in protecting data, discovering that the magnitude of the noise's distribution variance is crucial, with a variance threshold above 10^{-2} effectively mitigating leakage attacks at the cost of significant performance degradation. The work [19] introduced a data perturbation method that maintains model performance while securing the privacy of training data. This approach transforms the input data matrix into a new feature space through a multidimensional transformation, applying variable scales of transformation to ensure adequate perturbation. However, this technique depends on a centralized server for generating global perturbation parameters and may distort the structural information in image-based datasets. The method proposed in [23] implemented DP to introduce noise into each client's training dataset using a per-example-based DP method, termed Fed-CDP. They proposed a dynamic decay method for noise injection to enhance defense against gradient leakage and improve inference performance. Despite its effectiveness in preventing data reconstruction from gradients, this method significantly reduces inference accuracy and incurs high computational costs due to the per-sample application of DP.

Both PRECODE [26] and FedKL [39] aim to prevent input information from propagating through the model during gradient computation. PRECODE achieves this by incorporating a probabilistic encoder-decoder module ahead of the output layer, which normalizes the feature representations and significantly hinders input data leakage through gradients. This process involves encoding the input features into a sequence, normalizing based on calculated mean and standard deviation values, and then decoding into a latent representation that feeds into the output layer. While effective, the additional computational overhead from the two fully-connected layers required by PRECODE limits its applicability to shallow DNNs due to high computational costs. In contrast, FedKL method introduces a key-lock module that manages the weight parameters with a hyper-parameter controlling the input dimension and an output dimension optimized for specific architectures; 16 for *ResNet-20* and *ResNet-32* on CIFAR-10 and CIFAR-100, and 64 for *ResNet-18* and *ResNet-34* on the ILSVRC2012 dataset. This configuration not only secures the gradients against leakage but also maintains manageable computational demands, making it feasible for more complex DNNs.

Diverging from the above methods, our newly proposed AdaDefense strategically integrates the Adam algorithm as a plugin component. This integration processes the original local gradients to produce a gradients stand-in, which is then used for global gradient aggregation. The gradients stand-in preserves the high-level latent representations of the original gradients, thereby ensuring minimal impact on model performance following each aggregation round. Furthermore, since the complex computation of gradients stand-in and both the first-order and second-order moments are retained locally, it becomes infeasible for a malicious attacker to reconstruct private training data from the gradients stand-in. This enhancement not only fortifies the privacy safeguards within FL environments but also maintains the integrity and effectiveness of the learning process.

III. Methodology

In the initial subsection, we provide an overview of how gradient leakage can expose private training data. Subsequently, we performed a mathematical analysis to demonstrate the efficacy of our proposed gradients stand-in in defending leakage attacks. This analysis explains how the gradients stand-in protects sensitive information while maintaining model performance. Finally, we present a comprehensive introduction to the proposed AdaDefense method.

A. Gradient Leakage

In the context of Machine Learning (ML), particularly in training neural networks, the gradient points in the direction in which the function (often a loss or cost function) most quickly increases. Finding the function's minimum involves moving in the direction opposite to the gradient, a process known as gradient descent.

Definition 1. ***Gradient:*** *Consider a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, which maps a vector $\mathbf{x}$ in n-dimensional space to a vector $\mathbf{y}$ in m-dimensional space, defined by $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. The gradient of $\mathbf{f}$ with respect to $\mathbf{x}$ is represented by the Jacobian matrix as follows:*

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

This matrix consists of partial derivatives where the element in the i-th row and j-th column, $\frac{\partial f_i}{\partial x_j}$, represents the partial derivative of the i-th component of $\mathbf{y}$ with respect to the j-th component of $\mathbf{x}$.

The *Chain Rule* is a rule in calculus that describes the derivative of a composite function. In simple terms, if a variable z depends on y, and y depends on x, then z, indirectly through y, depends on x, and the chain rule helps in finding the derivative of z with respect to x. If $z = f(y)$ and $y = g(x)$, then:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Many recent studies have highlighted the vulnerability of gradient data to leakage attacks. In their research, the work [6] demonstrated that for fully-connected layers, the gradients of the loss with respect to the outputs can reveal information about the input data. By employing the *Chain Rule*, it is possible to reconstruct the inputs of a fully-connected layer using only its gradients, independent of the gradients from other layers. Expanding on this, in FedKL [39], the authors explored this vulnerability in convolutional and BN layers across typical supervised learning tasks. They provided a detailed analysis of how gradients carry training data, which can be exploited by an attacker to reconstruct this data. Their results confirmed a strong correlation between the gradients and the input training data across linear and non-linear neural networks, including Convolutional Neural Network (CNN) and BN layers. Ultimately, they concluded that in image classification tasks, the gradients from the fully-connected, convolutional, and BN layers contain ample information about the input data and the true labels. This richness of information enables an attacker to reconstruct both the inputs and labels by regressing the gradients. We can conclude that:

Proposition 1. *In image classification tasks, the gradients from the fully connected, convolutional and BN layers contain a great deal of information about the input data and the actual labels. These details allow an attacker to approximate these gradients and effectively reconstruct the corresponding inputs and labels.*

B. Theoretical Analysis on Gradients Stand-in

Model Performance: In this part, we explore the influence of using gradients stand-in on model performance, and provide justification for employing gradients stand-in during global gradient aggregation in a FL context. We begin by reviewing the Adam [28] optimization algorithm, a method of SGD that incorporates momentum concepts. Each iteration in Adam involves computing the first-order and second-order moments of the gradients, followed by calculating their exponential moving averages to update the model parameters. This approach merges the strengths of the Adagrad [40] algorithm, which excels in handling sparse data, with those of the RMSProp [41] algorithm, designed to manage non-smooth data effectively. Collectively, these features enable Adam to deliver robust performance across a wide range of optimization problems, from classical convex formulations to complex DP tasks. Further details on the Adam algorithm are presented in Algorithm 1.

In the local training phase of FL, we concentrate on the aggregated gradients from each training round rather than the gradients from individual iterations. Specifically, the local gradients that are transmitted to the global server represent the cumulative sum of all gradients produced during the local training iterations. Common FL aggregation methods,

Algorithm 1 Adam Optimization Algorithm

```
Require: Initial learning rate $\alpha$
Require: Exponential decay rates for moment estimates $\beta_1$, $\beta_2 \in [0,1]$
Require: Small constant for numerical stability $\epsilon$
Require: Maximum number of iterations $I$
 1: Initialize weights $\omega_0$
 2: Initialize the first-order moment vector $m_0 \leftarrow 0$
 3: Initialize the second-order moment vector $v_0 \leftarrow 0$
 4: Initialize the time-step $t \leftarrow 0$
 5: for $t = 1$ to $I$ do
 6:     $g_t \leftarrow \nabla_\omega \mathcal{L}(\omega_{t-1})$    ▶Get gradients w.r.t. stochastic objective at time-step $t$
 7:     $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$    ▶Update biased first-order moment estimate
 8:     $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$    ▶Update biased second-order moment estimate
 9:     $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$    ▶Compute bias-corrected first-order moment estimate
10:     $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$    ▶Compute bias-corrected second-order moment estimate
11:     $\omega_t \leftarrow \omega_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$    ▶Update parameters
12: end for
13: return $\omega_t$    ▶Output the optimized parameters
```

such as Federated Averaging (FedAvg) [1], FedAdp [42], and FedOpt [43], employ these cumulative local gradients for global model updates. This approach effectively outlines the model's convergence path and direction within an FL framework. Moreover, studies such as Adam [28] optimization algorithm has demonstrated superior convergence properties compared to traditional SGD, underscoring its widespread adoption in the DL domain. Based on these insights, we propose a novel method of representing local gradients using the Adam algorithm to enhance privacy and maintain model efficacy in FL. Experimental outcomes discussed in Section IV confirm the viability of using these modified gradients for global aggregation, thereby supporting the integrity and performance of the FL model.

Leakage Defense: In the FedKL [39], it was demonstrated that private training data leakage through gradients sent to a global server is a significant concern. The authors of FedKL introduced a novel approach involving a key-lock pair to generate shift and scale parameters in the BN layer, which are typically trainable within the model. Crucially, in FedKL, these generated parameters are retained on the local clients. The primary aim of FedKL is to sever the transmission of private training data via gradients, thereby safeguarding against potential leakage by malicious servers. This method involves modifications to the network architecture, including additional layers that are responsible for generating the shift and scale parameters. While these changes have minimal impact on model performance, they significantly reduce the system's efficiency due to the increased computational overhead. In our work, we adhere to the principle of preventing the propagation of private training data through gradients. However, unlike FedKL, we achieve this without altering the network architecture or adding extra training layers, thus avoiding additional computational burdens. The proposed approach maintains system efficiency while still protecting against data leakage.

Claim 1. *The gradients stand-in transmitted to the global server no longer contains information from which details of the private training data can be inferred.*

Proof. Assuming g_r is the local gradients in the r-th training round and $\hat{g}_r$ is the gradients stand-in for global gradients aggregation. By applying the Adam in Algorithm 1, we have:

$$m_r = \beta_1 m_{r-1} + (1-\beta_1) g_r \tag{1}$$

$$v_r = \beta_2 v_{r-1} + (1-\beta_2) g_r^2 \tag{2}$$

$$\hat{m}_r = \frac{m_r}{1-\beta_1^r} \tag{3}$$

$$\hat{v}_r = \frac{v_r}{1-\beta_2^r} \tag{4}$$

$$\hat{g}_r = \frac{\hat{m}_r}{\sqrt{\hat{v}_r}+\epsilon} \tag{5}$$

where β_1 and β_2 are set to 0.9 and 0.999, separately; the values in g_0 are initialized to all zeros; $g_r \in \mathbb{R}^D$, $m_r \in \mathbb{R}^D$, $v_r \in \mathbb{R}^D$, $\hat{m}_r \in \mathbb{R}^D$, $\hat{v}_r \in \mathbb{R}^D$ and $\hat{g}_r \in \mathbb{R}^D$, D is the dimension of the parameter space, m_r is the biased first-order moment estimate in the r-th training round, v_r is the biased second-order moments estimate, $\hat{m}_r$ is the bias-corrected first-order moment estimate and $\hat{v}_r$ is the bias-corrected second-order moment estimate. The derivative of the gradients stand-in, $\hat{g}_r$, w.r.t. the original local gradients, g_r, can be expressed as:

$$\frac{\partial \hat{g}_r}{\partial g_r} = \frac{\partial \hat{g}_r}{\partial \hat{m}_r}\frac{\partial \hat{m}_r}{\partial g_r} + \frac{\partial \hat{g}_r}{\partial \hat{v}_r}\frac{\partial \hat{v}_r}{\partial g_r} \tag{6}$$

Given (1) and (2), differentiating m_r and v_r w.r.t. g_r:

$$\frac{\partial m_r}{\partial g_r} = 1-\beta_1 \tag{7}$$

$$\frac{\partial v_r}{\partial g_r} = 2(1-\beta_2) g_r \tag{8}$$

Given (3) and (4), differentiating $\hat{m}_r$ and $\hat{v}_r$ w.r.t. g_r:

$$\frac{\partial \hat{m}_r}{\partial g_r} = \frac{\frac{\partial m_r}{\partial g_r}}{1-\beta_1^r} = \frac{1-\beta_1}{1-\beta_1^r} \tag{9}$$

$$\frac{\partial \hat{v}_r}{\partial g_r} = \frac{\frac{\partial v_r}{\partial g_r}}{1-\beta_2^r} = \frac{2(1-\beta_2) g_r}{1-\beta_2^r} \tag{10}$$

Given (5), using the quotient rule, where $u = \hat{m}_r$ and $z = \sqrt{\hat{v}_r}+\epsilon$:

$$\frac{\partial \hat{g}_r}{\partial g_r} = \frac{\frac{\partial \hat{m}_r}{\partial g_r} z - u \frac{\partial}{\partial g_r}(\sqrt{\hat{v}_r}+\epsilon)}{z^2} \tag{11}$$

Calculating $\frac{\partial}{\partial g_r}(\sqrt{\hat{v}_r}+\epsilon)$:

$$\begin{aligned}\frac{\partial\sqrt{\hat{v}_r}}{\partial g_r} &= \frac{1}{2\sqrt{\hat{v}_r}}\cdot\frac{\partial\hat{v}_r}{\partial g_r}\\ &= \frac{1}{2\sqrt{\hat{v}_r}}\cdot\frac{2(1-\beta_2)g_r}{1-\beta_2^r}\\ &= \frac{(1-\beta_2)g_r}{\sqrt{\hat{v}_r}(1-\beta_2^r)}\end{aligned} \tag{12}$$

Then, differentiating $\hat{g}_r$ w.r.t. g_r:

$$\frac{\partial\hat{g}_r}{\partial g_r} = \frac{(\frac{1-\beta_1}{1-\beta_1^r})(\sqrt{\hat{v}_r}+\epsilon) - \hat{m}_r\cdot\frac{(1-\beta_2)g_r}{\sqrt{\hat{v}_r}(1-\beta_2^r)}}{(\sqrt{\hat{v}_r}+\epsilon)^2} \tag{13}$$

Substitute $\hat{m}_r$ and $\hat{v}_r$ in (13) referring to (1), (2), (3) and (4). To simplify the equation. We define:

$$\begin{aligned}V &= \sqrt{\hat{v}_r}\\ &= \sqrt{\frac{\beta_2 v_{r-1} + (1-\beta_2)g_r^2}{1-\beta_2^r}}\end{aligned} \tag{14}$$

Then, the equation would be:

$$\frac{\partial\hat{g}_r}{\partial g_r} = \frac{(1-\beta_1)(V+\epsilon) - (\frac{(1-\beta_2)\beta_1 m_{r-1} + (1-\beta_2)(1-\beta_1)g_r}{V\sqrt{1-\beta_2^r}})g_r}{(1-\beta_1^r)(V+\epsilon)^2} \tag{15}$$

Given $\beta_1 = 0.9$, $\beta_2 = 0.999$ and ϵ is a very small value that can be ignored, considering both large or small r, we approximate:

$$(V+\epsilon) \approx \alpha g_r \tag{16}$$

where α is a non-zero scaling variant. In the end, we have:

$$\frac{\partial\hat{g}_r}{\partial g_r} = \frac{-\beta_1 m_{r-1}}{\alpha(1-\beta_1^r)}\cdot\frac{1}{g_r^2} \tag{17}$$

See Appendix for more specific derivation process.

In (17), the parameters α and $(1-\beta_1)$ are non-zero values, while m_{r-1} is exclusively preserved within the confines of local clients. This specific configuration ensures that the original local gradients, g_r, are safeguarded against deep leakage from gradients by potential malicious attackers. Consequently, the proposed gradients stand-in, $\hat{g}_r$, effectively obstructs any attempts to deduce or reconstruct private training data from the local gradients. This safeguarding mechanism enhances the security of the data during the FL training process, providing a robust defense against potential data privacy breaches. ■

C. AdaDefense in FL

Federated Learning: The foundational method underpinning recent FL techniques assumes the presence of multiple clients, represented as $\mathcal{C}$, each possessing their own local datasets $\mathcal{D}$. The learning task is defined by a model $\mathcal{F}$ with parameters ω. The local gradient g_i for each client i is computed as follows:

$$g_i = \frac{1}{||d_i||}\nabla_\omega \sum_j \mathcal{L}(\mathcal{F}(x^{(j)};\omega), y^{(j)}) \quad \text{for all } i\in\mathcal{C}, \tag{18}$$

where $||\cdot||$ denotes the L2 norm, used here to normalize the gradients by the size of the dataset d_i. In this formula, $x^{(j)}$ and $y^{(j)}$ are the input features and corresponding labels of the j-th example in the i-th local dataset.

After computing the local gradients, the server aggregates these gradients and performs an averaging process to update the global model parameters ω_r. This process is mathematically represented as:

$$\omega_r = \omega_{r-1} - \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|} g_i, \tag{19}$$

where $|\mathcal{C}|$ denotes the total number of clients. It is assumed that all local datasets are of equal size, i.e., $||d_i|| = ||d_k||$ for any $d_i, d_k \in \mathcal{D}$. This assumption simplifies the model updating process by maintaining uniform influence for each client's gradient.

AdaDefense: Emphasizing the innovative nature of AdaDefense is essential, as it integrates seamlessly into existing frameworks without altering the fundamental architecture of the model or the FL strategy it supports. AdaDefense is designed to act as an unobtrusive extension, focusing on the manipulation of local gradients. This process involves the creation of gradients stand-in that is used in the global aggregation phase. In a standard FL setup, the global model is initially crafted on a central server and then distributed to various clients. These clients perform secure local training and then send their local gradients stand-ins back to the server. At the server, these gradients stand-ins are aggregated to update and refine the global model. This cycle of local training and global updating repeats, with the server coordinating the aggregation of gradients to produce each new iteration of the global model. During this process, the server remains oblivious to all clients' first-order and second-order moments information. This omission is intentional and essential for security. This structure ensures that gradients stand-in cannot be exploited to reconstruct private, local training data. By maintaining this separation, AdaDefense strengthens the FL system against potential data reconstruction attacks. Consequently, in every round of local training that follows, each client adjusts its model based on the updated global model distributed by the server, thus advancing the collective learning process without compromising on privacy or security. See Fig. 2 for an illustration of AdaDefense in FL.

IV. Experiments

In this section, we begin by describing the datasets and metrics used for benchmark evaluation. We then detail a series of experiments designed to evaluate the performance of DNNs across two aspects. The first set of experiments investigates the impact of using the proposed gradients stand-in on the prediction accuracy. The second set assesses the effectiveness of this approach in defending against state-of-the-art gradient leakage attacks, specifically those involving GRNN, IG and GLAUS.

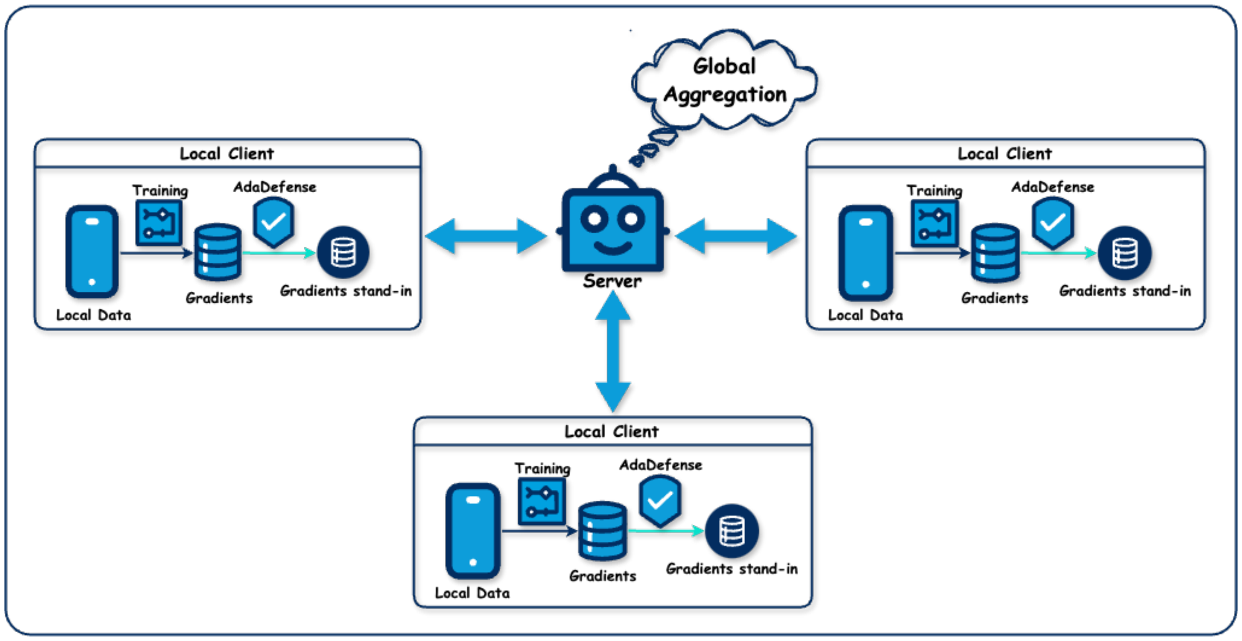


Fig. 2. Illustration of AdaDefense in FL

A. Benchmarks and Metrics

In our study, we conducted experiments using four well-known public datasets: MNIST [44], CIFAR-10 [45], CIFAR-100 [45], and ILSVRC2012 [46]. To assess the quality of images generated by our method, we employed four evaluation metrics: Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS) [47], and Structure Similarity Index Measure (SSIM) [48]. To optimize the conditions for gradient leakage attacks and effectively evaluate the defense capabilities, we set the batch size to one. This configuration allows for a direct comparison between the reconstructed image and the original image. The MSE metric quantifies the pixel-wise L2 norm difference between the reconstructed image $\hat{X}$ and the original image X, defined as $MSE(X, \hat{X}) = ||X - \hat{X}||_2$. PSNR, another criterion for image quality assessment, is calculated using the formula $PSNR(X, \hat{X}) = 10 \cdot \lg(\frac{255^2}{MSE(X,\hat{X})})$, where a higher PSNR value indicates a higher similarity between the two images. For perceptual similarity, we utilized the LPIPS metric, which measures the similarity between two image patches using pre-trained networks such as *VGGNet* [49] and *AlexNet* [50]. This metric aligns with human visual perception, where a lower LPIPS score indicates greater perceptual similarity between the compared images. The SSIM index evaluates the change in image structure, where a higher SSIM value suggests less distortion and thus a better quality of the reconstructed image. All neural network models were implemented using the PyTorch framework [51], ensuring robust and efficient computation.

B. Model Performance

To evaluate the impact on performance, we developed several distinct training methodologies. The first approach involved centralized training. The second approach was collaborative training within the FedAvg framework. The last the approach was also the FedAvg, but involved gradients stand-in. For the training of image classification models, we selected six foundational network architectures: *LeNet* [52], *ResNet-20*, *ResNet-32*, *ResNet-18*, *ResNet-34*, [53] and *VGGNet-16* [49] on different benchmarks including MNIST, CIFAR-10 and CIFAR-100 with image resolutions from $32 * 32$ to $224 * 224$. We systematically trained a total of thirty-three models across eleven experimental setups using both centralized and federated training strategies. This comprehensive approach allowed us to compare the performance of standard networks against those with gradients stand-in. The results of these comparisons are systematically detailed in Table I, providing a clear overview of the impact of the gradients stand-in on model performance in different training environments.

Centralized training consistently produces the most favorable results. The models appear to have a high performance on the MNIST and CIFAR-10 datasets, with accuracy slightly decreasing for the CIFAR-100 dataset. This pattern suggests that CIFAR-100 has increased complexity, due to a larger number of classes, poses a greater challenge for the models. For example, the implementation of *LeNet* on the MNIST dataset results in a remarkable accuracy of 99.05%, as reported in [52]. Similarly, *ResNet-20*, *ResNet-32*, *ResNet-18*, *ResNet-34* and *VGG-16* achieves their highest accuracies on CIFAR-10 and CIFAR-100 datasets, recording 91.63% and 67.59%, 92.49% and 70.35%, 91.62% and 72.15%, 92.20% and 73.21%, 89.13% and 63.23%, respectively. When employ-

TABLE I

TESTING ACCURACY (%) OF MODELS TRAINED UNDER VARIOUS CONFIGURATIONS. THE HIGHEST ACCURACY ACHIEVED IS HIGHLIGHTED IN RED, WHILE THE SECOND HIGHEST IS DENOTED IN BLUE. HERE, "C-10" AND "C-100" REFER TO CIFAR-10 AND CIFAR-100 DATASETS, RESPECTIVELY. "AD" STANDS FOR THE ADADEFENSE MODULE. THE ACCURACY FIGURES LISTED IN THE 'REFERENCE' ROW ARE SOURCED FROM CORRESPONDING PAPERS.

Model	*LeNet* (32*32)	*ResNet-20* (32*32)		*ResNet-32* (32*32)		*ResNet-18* (224*224)		*ResNet-34* (224*224)		*VGG-16* (224*224)	
Dataset	MNIST	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100	C-10	C-100
Centralized	98.09	91.63	67.59	92.34	70.35	91.62	72.15	92.20	73.21	89.13	63.23
FedAvg	98.14	91.20	58.58	91.37	61.91	89.50	68.59	89.27	68.30	88.13	61.91
FedAvg w/ AD	97.80	89.15	61.81	90.20	63.64	89.29	68.76	89.38	68.38	87.13	58.32
Reference	99.05 [52]	91.25 [53]	-	92.49 [53]	-	-	-	-	-	-	-

ing FedAvg, there is a noticeable trend where most models maintain a performance close to the centralized training, which showcases the effectiveness of FL in preserving model accuracy even when the data is decentralized. FedAvg gains four out of eleven second highest accuracies for *LeNet* on MNIST, *ResNet-18* on CIFAR-10, *VGG-16* on CIFAR-10 and CIFAR-100. The incorporation of the gradients stand-in into the training framework generally does not adversely affect the model's inference performance. In some cases, models equipped with the gradients stand-in even surpass the performance of FedAvg, achieving five out of eleven second highest accuracies. Within the AdaDefense framework, *ResNet-20* on CIFAR-100 achieves a testing accuracy of 61.81%, which is significantly higher by 5.51% than the conventional FedAvg model, which only reaches 58.58%.

C. Defense Performance

In evaluating the robustness of defense mechanisms against gradient leakage attacks, our study incorporated four advanced attack methodologies. These included GRNN [8], IG [6] and GLAUS [10]. For the datasets MNIST, CIFAR-10, and CIFAR-100, experiments were conducted using a batch size of one and an image resolution of $32*32$. However, for the ILSVRC2012 dataset, a higher resolution of $256*256$ was selected rather than the usual $224*224$. This adjustment was necessary because GRNN can only process image resolutions that are powers of two. To accommodate this requirement, images of lower resolutions were upscaled through linear interpolation to match the required dimensions. This approach ensures that the integrity and comparability of the results across different datasets and resolution settings are maintained.

In Table II, we present qualitative results that demonstrate the comparative effectiveness of various defense mechanisms against targeted leakage attacks. These defenses include GRNN and IG, which have been implemented across different backbone networks and benchmark datasets. The table provides a clear view of how each defense strategy performs, allowing for an assessment of their efficacy in a controlled environment.

As discussed in the paper [8], DLG is inadequate in several experimental scenarios, leading us to exclude its results from our analysis. Instead, we focused on GRNN, which has proven more effective. Our proposed AdaDefense that efficiently safeguards against the leakage of true images during gradient-based reconstruction processes. In our experiments using GRNN with a *ResNet-18* model at an image resolution of $256*256$, the reconstructed images retained only minimal and non-identifiable details from the original images. This indicates a significant protection against the exposure of private data. For other architectures, such as *LeNet* and *ResNet-20*, the results were even more promising, with the reconstructed images being completely unrecognizable and distorted. On the other hand, the GRNN is capable of revealing true labels from gradients. Consequently, we have included these labels in Table II for comprehensive analysis. It is evident from our observations that when models are trained using our AdaDefense module, no correct labels are inferred from gradients. This further indicates the effectiveness of AdaDefense in enhancing model security against gradient-based label inference attacks.

Furthermore, we compared the reconstruction capabilities of IG with those of GRNN. The findings confirmed that GRNN outperforms IG in safeguarding private training data across all tested configurations. Overall, our AdaDefense consistently demonstrated robust protection of private training information from gradient-based reconstructions in our experiments.

In the recent publication on the method GLAUS, it is noted that the official code supports only a specially designed architecture, *MNIST-CNN*, for use with the MNIST dataset. Attempts to adapt this method for other neural networks and datasets were unsuccessful. Consequently, the results presented in Table III are based solely on experiments conducted using the original, unmodified official code. This table highlights certain inherent shortcomings of GLAUS, even in the absence of our defense method. These limitations are generally considered acceptable, given that GLAUS is specifically tailored to attack UGS secure aggregation methods within the FL framework. Notably, when our AdaDefense module is integrated into the system, GLAUS is rendered ineffective in all trials concerning image reconstruction and label inference.

D. Quantitative Analysis

To conduct a quantitative assessment of our proposed AdaDefense module in comparison to existing state-of-the-art methods for mitigating gradient leakage, we utilized four distinct evaluation metrics: MSE, PSNR, LPIPS with both

TABLE II
COMPARISON OF IMAGE RECONSTRUCTION USING GRNN AND IG WITHOUT AND WITH ADADEFENSE.

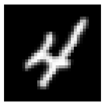
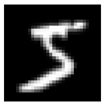
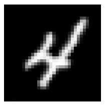
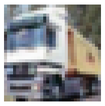
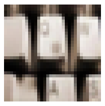
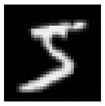
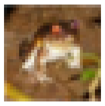
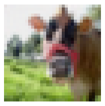
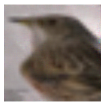
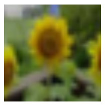
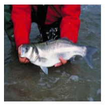
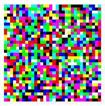
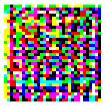
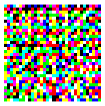
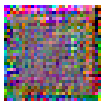
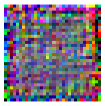
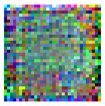
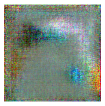
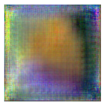
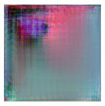
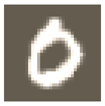
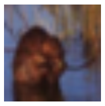
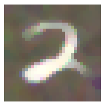
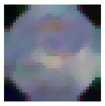
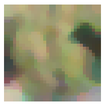
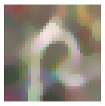
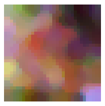
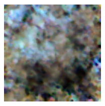
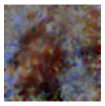
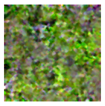
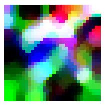
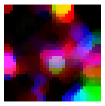
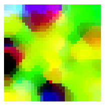
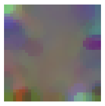
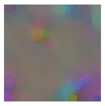
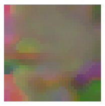
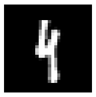
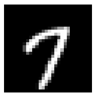
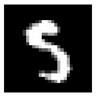
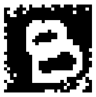
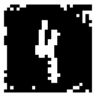
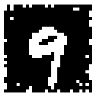
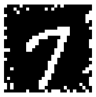
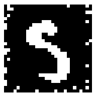
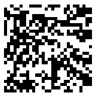
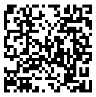
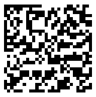
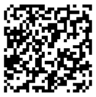
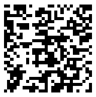
Model		*LeNet* (32*32)			*ResNet-20* (32*32)			*ResNet-18* (256*256)		
Dataset		MNIST	C-10	C-100	MNIST	C-10	C-100	C-10	C-100	ILSVRC
GRNN	True	 four	truck	keyboard	 five	frog	cattle	bird	sunflower	tench
	w/o	 four	 truck	 keyboard	 five	 frog	 cattle	 bird	 sunflower	 tench
	w/	 zero	 ship	 bowl	 four	 horse	 mushroom	 dog	 oak tree	 soccer ball
IG	True	two	airplane	willow	 zero	horse	rose	truck	 beaver	jay
	w/o	 -	 -	 -	 -	-	 -	 -	 -	 -
	w/	 -	 -	 -	 -	 -	 -	-	-	-

TABLE III
IMAGE RECONSTRUCTION FOR GLAUS WITHOUT AND WITH ADADEFENSE.

Model		*MNIST-CNN* (28*28)				
Dataset		MNIST				
GLAUS	True	eight	 four	nine	 seven	 five
	w/o	 eight	 four	 nine	 seven	 five
	w/	 five	 five	 five	 five	 five

VGGNet and *AlexNet*, and SSIM. The evaluation results, as detailed in Table IV, were derived from comparisons between generated images and their respective original images. Our analysis consistently revealed that images produced by models incorporating the AdaDefense module exhibited significantly lower degrees of similarity when compared to those from models lacking this feature. This outcome demonstrates the efficacy of the AdaDefense module in providing robust defense against gradient leakage attacks.

To enhance the clarity of our analysis on the defensive capabilities of AdaDefense, we have concentrated our examination primarily on GRNN. This focus is driven by our observations that GRNN exhibits a significantly stronger attack capability compared to IG. Through this targeted analysis, we aim to demonstrate more effectively the robustness of AdaDefense against more potent threats. Starting with MSE, we see that in every case, the introduction of AdaDefense results in a dramatic increase in MSE, which implies a higher error rate between generated and true images when the defense is active. For instance, the MSE for the *LeNet* model on the MNIST dataset jumps from 0.65 without AdaDefense to 173.44 with AdaDefense, representing an astonishing increase of approximately $26,600\%$. This trend of increased error rates with AdaDefense is consistent across all models and datasets,

TABLE IV

QUANTITATIVE COMPARISON OF GRNN AND IG WITHOUT AND WITH ADADEFENSE MODULE. THE RESULTS ARE COMPUTED FROM THE GENERATED IMAGES AND THEIR ASSOCIATED TRUE IMAGES.

Method	Model	Dataset	MSE↓		PSNR↑		LPIPS-V↓		LPIPS-A↓		SSIM↑	
			w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
GRNN	*LeNet* (32*32)	MNIST	0.65	173.44	50.11	25.74	0.05	0.75	0.01	0.56	1.000	-0.003
		C-10	1.97	143.19	45.28	26.57	0.00	0.56	0.00	0.35	0.997	0.000
		C-100	1.79	145.36	45.85	26.51	0.00	0.55	0.00	0.38	0.998	0.003
	ResNet-20 (32*32)	MNIST	3.55	125.81	44.04	27.14	0.12	0.70	0.01	0.47	0.954	0.003
		C-10	12.16	85.43	40.68	28.84	0.00	0.44	0.00	0.25	0.973	0.030
		C-100	22.30	86.43	37.37	28.84	0.00	0.43	0.00	0.24	0.958	0.053
	ResNet-18 (256*256)	C-10	17.36	70.76	37.55	29.69	0.01	0.43	0.01	0.57	0.955	0.151
		C-100	32.56	64.37	34.67	30.19	0.02	0.31	0.02	0.28	0.917	0.434
		ILSVRC	18.82	59.81	37.31	30.41	0.05	0.41	0.04	0.39	0.891	0.212
IG	*LeNet* (32*32)	MNIST	19.23	90.62	35.29	28.56	0.06	0.50	0.03	0.32	0.825	0.300
		C-10	22.16	101.09	34.68	28.08	0.18	0.63	0.06	0.30	0.637	0.060
		C-100	20.45	92.32	35.02	28.48	0.12	0.53	0.06	0.31	0.623	0.100
	ResNet-20 (32*32)	MNIST	49.25	60.81	31.23	30.30	0.22	0.27	0.15	0.24	0.361	0.125
		C-10	39.35	52.31	32.27	30.99	0.21	0.27	0.16	0.16	0.299	0.098
		C-100	38.19	47.25	32.37	31.41	0.22	0.31	0.10	0.20	0.309	0.110
	ResNet-18 (256*256)	C-10	64.38	45.54	30.11	32.08	0.35	0.32	0.22	0.21	0.325	0.476
		C-100	78.44	63.13	29.29	30.19	0.37	0.28	0.30	0.27	0.355	0.510
		ILSVRC	82.46	79.16	28.99	29.20	0.64	0.49	0.56	0.48	0.103	0.266

highlighting a significant protection of using AdaDefense in terms of gradients leakage. Conversely, the impact on PSNR, which measures image quality (higher is better), shows a general decline when AdaDefense is applied. For example, PSNR for the *ResNet-18* model on the ILSVRC dataset decreases from 37.31 to 30.41, a decrease of about 18.5%. LPIPS, evaluated using both *VGGNet* and *AlexNet*, measures perceptual similarity (lower is better). All the results from GRNN perform significantly worse when the AdaDefense module is adapted into the model. Lastly, the SSIM metric, which evaluates the structural similarity between the generated and true images, mostly shows a decline with AdaDefense. For the *ResNet-18* model on the CIFAR-10 dataset, SSIM decreases significantly from 0.955 to 0.151, about a 84.18% reduction. This supports the view that AdaDefense can protect against specific vulnerabilities.

V. CONCLUSION

In this study, we introduce, AdaDefense, a novel defense mechanism against gradient leakage tailored for the FL framework. The gist of AdaDefense is to use Adam to generate local gradients stand-ins for global aggregation, which effectively prevents the leakage of input data information through the local gradients. We provide a theoretical demonstration of how the gradients stand-in secures input data within the gradients and further validate its effectiveness through extensive experimentation with various advanced gradient leakage techniques. Our results, both quantitative and qualitative, indicate that integrating the proposed method into the model precludes the possibility of reconstructing input images from the publicly shared gradients. Furthermore, we evaluate the impact of the defense plugin on the model's classification accuracy, ensuring that the defense mechanism does not detract from the model's performance. We are excited to make the implementation of AdaDefense available for further research and application. Moving forward, our focus will shift to exploring the intricate relationships between the gradients of different layers and their connection to the input data, which promises to enhance the robustness of FL systems against various attack vectors.

CONTRIBUTION STATEMENT

Yi Hu: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing. Hanchi Ren: Conceptualization, Formal analysis, Investigation, Methodology, Supervision, Writing – review & editing. Chen Hu: Writing – review. Yiming Li: Writing – review. Jingjing Deng: Conceptualization, Supervision, Writing – review. Xianghua Xie: Supervision

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*. PMLR, 2017, pp. 1273–1282.

[2] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv:1610.02527*, 2016.

[3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv:1610.05492*, 2016.

[4] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, pp. 14 774–14 784, 2019.

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in neural information processing systems*, vol. 33, pp. 16 937–16 947, 2020.

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

[8] H. Ren, J. Deng, and X. Xie, "Grnn: Generative regression neural network–a data leakage attack for federated learning," *ACM Transactions on Intelligent Systems and Technology*, 2021.

[9] Z. Li, J. Zhang, L. Liu, and J. Liu, "Auditing privacy defenses in federated learning via generative gradient leakage," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 132–10 142.

[10] Y. Yang, Z. Ma, B. Xiao, Y. Liu, T. Li, and J. Zhang, "Reveal your images: Gradient leakage attack against unbiased sampling-based secure aggregation," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[11] Y. Zhao, Y. Zhang, Y. Li, Y. Zhou, C. Chen, T. Yang, and B. Cui, "Minmax sampling: A near-optimal global summary for aggregation in the wide area," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 744–758.

[12] X. Yang, Y. Feng, W. Fang, J. Shao, X. Tang, S.-T. Xia, and R. Lu, "An accuracy-lossless perturbation method for defending privacy attacks in federated learning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 732–742.

[13] L. Sun, J. Qian, and X. Chen, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[14] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9311–9319.

[15] A. T. Hasan, Q. Jiang, J. Luo, C. Li, and L. Chen, "An effective value swapping method for privacy preserving data publishing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3219–3228, 2016.

[16] M. A. P. Chamikara, P. Bertók, D. Liu, S. Camtepe, and I. Khalil, "Efficient data perturbation for privacy preserving and accurate data stream mining," *Pervasive and Mobile Computing*, vol. 48, pp. 1–19, 2018.

[17] M. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "Efficient privacy preservation of big data for accurate data mining," *Information Sciences*, vol. 527, pp. 420–443, 2020.

[18] H. Lee, J. Kim, S. Ahn, R. Hussain, S. Cho, and J. Son, "Digestive neural networks: A novel defense strategy against inference attacks in federated learning," *computers & security*, vol. 109, p. 102378, 2021.

[19] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, "Privacy preserving distributed machine learning with federated learning," *Computer Communications*, vol. 171, pp. 112–125, 2021.

[20] Z. Bu, J. Dong, Q. Long, and W. J. Su, "Deep learning with gaussian differential privacy," *Harvard data science review*, vol. 2020, no. 23, 2020.

[21] Y. Li, Y. Zhou, A. Jolfaei, D. Yu, G. Xu, and X. Zheng, "Privacy-preserving federated learning framework based on chained secure multiparty computing," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6178–6186, 2020.

[22] K. Yadav, B. B. Gupta, K. T. Chui, and K. Psannis, "Differential privacy approach to solve gradient leakage attack in a federated machine learning environment," in *International Conference on Computational Data and Social Networks*. Springer, 2020, pp. 378–385.

[23] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 797–807.

[24] H. Ren, J. Deng, X. Xie, X. Ma, and Y. Wang, "Fedboosting: Federated learning with gradient protected boosting for text recognition," *Neurocomputing*, vol. 569, p. 127126, 2024.

[25] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating client privacy leakages in federated learning," in *Computer Security*. Springer, 2020, pp. 545–566.

[26] D. Scheliga, P. Mäder, and M. Seeland, "Precode-a generic model extension to prevent deep gradient leakage," in *Winter Conference on Applications of Computer Vision*, 2022, pp. 1849–1858.

[27] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[30] S. Truex, L. Liu, M. E. Gursoy, L. Yu, and W. Wei, "Towards demystifying membership inference attacks," *arXiv:1807.09173*, 2018.

[31] S. Truex, L. Liu, M. Gursoy, L. Yu, and W. Wei, "Demystifying membership inference attacks in machine learning as a service," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2073–2089, 2021.

[32] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1964–1974.

[33] G. Zhang, B. Liu, T. Zhu, M. Ding, and W. Zhou, "Label-only membership inference attacks and defenses in semantic segmentation models," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[34] X. Xie, C. Hu, H. Ren, and J. Deng, "A survey on vulnerability of federated learning: A learning algorithm perspective," *Neurocomputing*, p. 127225, 2024.

[35] J. Jeon, K. Lee, S. Oh, J. Ok *et al.*, "Gradient inversion with generative image prior," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[36] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 337–16 346.

[37] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *ACM CCS*, 2017, pp. 603–618.

[38] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[39] H. Ren, J. Deng, X. Xie, X. Ma, and J. Ma, "Gradient leakage defense with key-lock module for federated learning," *arXiv preprint arXiv:2305.04095*, 2023.

[40] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[41] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012.

[42] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1078–1088, 2021.

[43] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *International Conference on Learning Representations*, 2020.

[44] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[45] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Citeseer*, 2009.

[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[47] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[48] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

Appendix

Proof of the Approximation:

In the context of simplifying the derivative $\frac{\partial \hat{g}_r}{\partial g_r}$ and considering whether V can be approximated by αg_r, we need to analyze the terms carefully.

Claim 2. *For large values of r, the model has undergone sufficient training to approach convergence. At this stage, V can be closely approximated by αg_r, which simplifies the expression for the derivative $\frac{\partial \hat{g}_r}{\partial g_r}$.*

Proof. Given the value $\beta_2 = 0.999$, evaluate the assumption that $V \approx \alpha g_r$, where:

$$V = \sqrt{\frac{\beta_2 v_{r-1} + (1-\beta_2)g_r^2}{1-\beta_2^r}}$$

Given the value of β_2, examine this approximation more critically. We first evaluate β_2^r. For larger r, β_2^r becomes quite small because $\beta_2 = 0.999$ is very close to 1. The decay term $1-\beta_2^r$ is crucial to understanding V's behavior. To compute $1-\beta_2^r$, say $r = 1000$ for example:

$$\beta_2^{1000} = (0.999)^{1000}$$

This can be computed as:

$$\beta_2^{1000} \approx e^{1000 ln(0.999)}$$

Referring to Taylor expansion $ln(1-x) \approx -x$ for small x, we have:

$$ln(0.999) \approx -0.001$$
$$1000 ln(0.999) \approx -1$$

Hence,

$$\beta_2^{1000} \approx e^{-1} \approx 0.3679$$

Given the computation above, we have:

$$V \approx \sqrt{\frac{\beta_2 v_{r-1} + (1-\beta_2)g_r^2}{0.6321}}$$

Assuming $v_{r-1} \approx g_r^2$ for simplicity (which can be the case as gradients do not change dramatically over large iterations), then:

$$V \approx \sqrt{\frac{0.999 g_r^2 + 0.001 g_r^2}{0.6321}} = \sqrt{\frac{g_r^2}{0.6321}} \approx \frac{g_r}{0.795} \approx 1.258 g_r$$

With $\beta_2 = 0.999$, the term V is approximate $1.258 g_r$ when assuming $v_{r-1} \approx g_r^2$ at larger r. So in this case, $\alpha \approx 1.258$ ■

Claim 3. *Given the condition of small r, the model does not converge effectively. Under this circumstance, the variable V can be accurately approximated by αg_r.*

Proof. With the values $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and considering small r, such as $r = 1$ or $r = 2$, we explore how V behaves. This scenario is typical at the beginning of the training, where initial values for moving averages m_0 and v_0 are set to zero, and the initial gradient g_0 is randomly initialized.

Calculation for $r = 1$:

$$\begin{aligned} m_1 &= \beta_1 m_0 + (1-\beta_1) g_0 = 0.1 g_0 \\ v_1 &= \beta_2 v_0 + (1-\beta_2) g_0^2 = 0.001 g_0^2 \\ \hat{m}_1 &= \frac{m_1}{1-\beta_1^1} = g_0 \\ \hat{v}_1 &= \frac{v_1}{1-\beta_2^1} = g_0^2 \\ V &= \sqrt{\hat{v}_1} = \sqrt{g_0^2} = g_0 \end{aligned}$$

Calculation for $r = 2$, assume $g_1 \approx g_0$ (for simplicity, assuming the gradient does not change too much):

$$\begin{aligned} m_2 &= \beta_1 m_1 + (1-\beta_1) g_1 = 0.19 g_0 \\ v_2 &= \beta_2 v_1 + (1-\beta_2) g_1^2 = 0.002 g_0^2 \\ \hat{m}_2 &= \frac{m_2}{1-\beta_1^2} \approx g_0 \\ \hat{v}_2 &= \frac{v_2}{1-\beta_2^2} \approx g_0^2 \\ V &= \sqrt{\hat{v}_2} \approx \sqrt{g_0^2} = g_0 \end{aligned}$$

With the values $\beta_1 = 0.9$ and $\beta_2 = 0.999$, the term V is approximate g_r when assuming $g_1 \approx g_0$ at the beginning of the training (small r). So in this case, $\alpha \approx 1$. ■

Derivation Process:

Now, we are going to provide specific derivation process of (17). Given that $(V + \epsilon) \approx \alpha g_r$ and considering ϵ to be negligible for simplification, we will further simplify the derivative of $\hat{g}_r$ w.r.t. g_r:

$$\hat{g}_r = \frac{\hat{m}_r}{\sqrt{\hat{v}_r} + \epsilon} \approx \frac{\hat{m}_r}{\sqrt{\hat{v}_r}} = \frac{\hat{m}_r}{V} = \frac{\hat{m}_r}{\alpha g_r}$$

According to the quotient rule $(\frac{u}{v})' = \frac{u'v - uv'}{v^2}$, we have:

$$\begin{aligned} \hat{m}_r &= \frac{\beta_1 m_{r-1} + (1-\beta_1) g_r}{1-\beta_1^r} \\ \frac{\partial \hat{m}_r}{\partial g_r} &= \frac{1-\beta_1}{1-\beta_1^r} \end{aligned}$$

Hence,

$$\begin{aligned} \frac{\partial \hat{g}_r}{\partial g_r} &= \frac{(\frac{1-\beta_1}{1-\beta_1^r})(\alpha g_r) - (\frac{\beta_1 m_{r-1} + (1-\beta_1) g_r}{1-\beta_1^r})(\alpha)}{(\alpha g_r)^2} \\ &= \frac{\frac{\alpha(1-\beta_1) g_r - \alpha(\beta_1 m_{r-1} + (1-\beta_1) g_r)}{1-\beta_1^r}}{\alpha^2 g_r^2} \\ &= \frac{\alpha(1-\beta_1) g_r - \alpha \beta_1 m_{r-1} - \alpha(1-\beta_1) g_r}{\alpha^2 g_r^2 (1-\beta_1^r)} \\ &= \frac{-\beta_1 m_{r-1}}{\alpha g_r^2 (1-\beta_1^r)} \end{aligned}$$