Boosting Gradient Leakage Attacks: Data Reconstruction in Realistic FL Settings

Mingyuan Fan[1] Fuyi Wang[2] Cen Chen[1]* Jianying Zhou[3]
[1]*East China Normal University* [2]*RMIT University* [3]*Singapore University of Technology and Design*
mingyuan_fmy@stu.ecnu.edu.cn *fuyi.wang@rmit.edu.au*
cenchen@dase.ecnu.edu.cn *jianying_zhou@sutd.edu.sg*

Abstract

Federated learning (FL) enables collaborative model training among multiple clients without the need to expose raw data. Its ability to safeguard privacy, at the heart of FL, has recently been a hot-button debate topic. To elaborate, several studies have introduced a type of attacks known as gradient leakage attacks (GLAs), which exploit the gradients shared during training to reconstruct clients' raw data. On the flip side, some literature, however, contends no substantial privacy risk in practical FL environments due to the effectiveness of such GLAs being limited to overly relaxed conditions, such as small batch sizes and knowledge of clients' data distributions.

This paper bridges this critical gap by empirically demonstrating that clients' data can still be effectively reconstructed, even within realistic FL environments. Upon revisiting GLAs, we recognize that their performance failures stem from their inability to handle the gradient matching problem. To alleviate the performance bottlenecks identified above, we develop FEDLEAK, which introduces two novel techniques, partial gradient matching and gradient regularization. Moreover, to evaluate the performance of FEDLEAK in real-world FL environments, we formulate a practical evaluation protocol grounded in a thorough review of extensive FL literature and industry practices. Under this protocol, FEDLEAK can still achieve high-fidelity data reconstruction, thereby underscoring the significant vulnerability in FL systems and the urgent need for more effective defense methods.

1 Introduction

Federated learning (FL) [20, 42] has become the de facto privacy-preserving approach for training deep neural networks (DNNs). In FL, two steps are iteratively performed: 1) Each client downloads the current global model and computes gradients locally on its own data; 2) The server collects and aggregates these locally computed gradients from all participating clients to update the global model. By transmitting only the gradients rather than raw data, FL inherently offers a plausible mechanism for privacy protection, promoting its adoption in various privacy-sensitive scenarios [19, 39].

Although FL's privacy protection mechanism appears intuitive, recent studies have shown that clients' raw data can still be reconstructed by exploiting the uploaded gradients, known as gradient leakage attacks (GLAs) [4, 22, 58]. We here focus on a passive attack scenario where the server does not manipulate the FL training process. The core of GLAs lies in the gradient matching problem, which aligns the gradients generated from dummy data with the gradients uploaded from clients. The vanilla GLA [58], for example, starts by randomly initializing dummy data and labels, subsequently adjusting these initializations iteratively to approximate clients' raw data by minimizing the L_2 distance between the uploaded gradients and the gradients generated by these initializations.

However, the effectiveness of existing GLAs remains constrained to overly simplified scenarios, casting doubts about their actual threat in real-world FL environments. ***First***, current GLAs [4, 26, 50] work well only when facing small models, simplistic datasets, and limited batch sizes, which are rarely reflective of real-world scenarios. While recent efforts have sought to extend these attacks to more sophisticated models and datasets, e.g., ResNet and ImageNet, the construction results remain either insufficient or contingent upon access to clients' batch normalization statistics [52] or their data distributions [27, 32, 53]. This reliance poses significant challenges, since clients are not obligated to share such statistics [31], and gathering large amounts of data similar to that of clients is rather difficult [16], particularly in data-scarce domains like healthcare. ***Second***, the performance of GLAs is positively correlated with the sensitivity of the input data to gradient changes. High sensitivity allows attackers to more easily manipulate input data to generate gradients that align with the uploaded gradients. Conversely, low sensitivity complicates the gradient matching problem, making it harder for attackers to identify how changes in input will affect the gradients. Generally, this sensitivity peaks at the outset of training when the model weights are randomly ini-

*Corresponding author.

tialized [4, 16]. Yet, this sensitivity can be greatly reduced through the use of pre-trained weights, as pre-trained weights have already seen other data, making the resulting model less responsive to clients' data. ***Third***, in practice, clients can adopt various defense strategies to mitigate the impact of GLAs [19, 23, 46].

To investigate whether FL can truly safeguard client privacy in real-world environments, we design FEDLEAK. To the best of our knowledge, FEDLEAK represents the first attack method capable of reconstructing high-fidelity data from gradients shared by clients in practical environments. Importantly, FEDLEAK operates without the need for any extra resources, relying exclusively on the information allowed by the basic FL training protocol [39], such as the model and gradients. This indicates that the attack implication of FEDLEAK can be extended across almost all FL systems that adhere to the basic FL training protocol. Moreover, we clarify that FEDLEAK should not be misconstrued as a malicious threat to existing FL systems; rather, it is intended as a constructive resource aimed at deepening our understanding of FL's privacy protection mechanism and assessing the effectiveness of various defense methods. Below we describe the challenges encountered in designing FEDLEAK and present our key ideas in addressing these challenges.

Challenge I: What limits the effectiveness of existing GLAs? Instead of incrementally refining existing attack methods, we take a step back to re-examine the gradient matching problem itself to uncover the root causes behind the failure of current GLAs. We start by analyzing whether the gradient matching problem is well-posed, a crucial factor in determining the feasibility of GLAs. The analysis largely yields a positive response, challenging the common belief that the inherent multiplicity of solutions in the gradient matching problem leads to the suboptimal performance of current GLAs [19, 32]. We argue that the underperformance of these attacks stems from their inability to properly solve the underlying gradient matching problem. To bridge this gap, we derive a sufficient condition to address the gradient matching problem, providing a guiding principle for designing effective attack methods.

Challenge II: How to implement the proposed guiding principle into attack methods? Based on the proposed principle, we introduce two novel techniques, namely partial gradient matching and gradient regularization. Partial gradient matching extends the original gradient matching problem by only matching a selected subset of the gradient elements, while gradient regularization penalizes the gradients of dummy data. We demonstrate that both techniques help satisfy the derived sufficient conditions more effectively. Moreover, we develop approximate solutions for these two techniques to address the prohibitively high computational cost associated with their original formulations, thereby enabling more efficient implementation in practical attack scenarios.

Challenge III: How can we assess the attack performance of* FEDLEAK *in real-world environments? To evaluate the effectiveness of FEDLEAK in real-world environments, it is essential to design a practical evaluation protocol. To this end, we carefully review existing studies in GLAs, identifying eight key factors that significantly influence attack performance. Next, we examine a wide range of FL studies and available industrial-grade FL libraries, particularly those in privacy-sensitive domains such as healthcare, so as to derive commonly used values for these factors. Leveraging this knowledge, we align the identified influential factors with real-world environments to develop an evaluation protocol grounded in hands-on examples. Through this protocol, we can gain a more practical implication of FEDLEAK against FL's privacy protection ability in real-world environments.

Contributions. We highlight our contributions below:

- Through our reexamination of the gradient matching problem, we pinpoint the root cause behind the failure of existing GLAs. We also derive a sufficient condition to resolve the gradient matching problem.
- We introduce FEDLEAK, which involves two novel techniques: partial gradient matching and gradient regularization. We further present two approximate solutions to address their prohibitively high computational costs.
- We formulate a practical evaluation protocol and conduct extensive experiments to validate the attack performance of FEDLEAK in accordance with this protocol. The attack results indicate that, even in real-world environments, FL still poses significant privacy leakage risks.

2 Background & Related Work

2.1 Federated Learning

We denote the global model as $F(\cdot, w)$ parameterized by w. Let U be the pool of available clients and $\mathcal{D}_i$ be the local dataset for the i-th client. The training process unfolds through iterative rounds $t \in \{1, 2, \ldots, T\}$:

- **Model distribution and local training.** The server broadcasts the current model parameters w to a subset of clients $U_t \subseteq U$ as participants for the t-th training round. Each client $i \in U_t$ subsequently samples a mini-batch $\mathcal{B}_i = (x_i, y_i) \subseteq \mathcal{D}_i$ to compute the local gradients $\nabla_w \mathcal{L}(F(x_i, w), y_i)$, where $\mathcal{L}$ denotes the loss function.
- **Gradient aggregation and model update.** The computed gradients $\nabla_w \mathcal{L}(F(x_i, w), y_i)$ are sent back to the server. The server aggregates these gradients, commonly by averaging them, and updates the global model.

Several challenges complicate the efficient implementation of FL. Many clients possess limited computational power, memory, and bandwidth [40]. To alleviate this, some studies [39, 42] suggested that selected clients update their local models multiple times, followed by sending the resulting model parameters to the server. The server then aver-

ages these uploaded parameters to form a new global model. However, this strategy risks causing local models to become overly specialized to their respective datasets, hindering the global model's peformance [30]. Notably, in this strategy, the server can deduce approximate gradients by comparing the parameters of the old global model and the newly uploaded model. Another critical challenge arises from data heterogeneity across clients, known as the non-IID problem [31, 39], which can lead to slower convergence rates and performance degradation. In this paper, we focus on GLAs.

2.2 Gradient Leakage Attack

Recent studies [56, 58] have highlighted the potential of shared gradients to be exploited in data reconstruction. Let $\hat{g}$ denote the gradients estimated by the server. Deep leakage from gradient (DLG) [58] iteratively adjusts dummy data x', y' to bring their generated gradients closer to $\hat{g}$, proving effective for small batches and simple networks:

$$x', y' = \arg\min_{x', y'} ||\nabla_w \mathcal{L}(F(x', w), y') - \hat{g}||_2^2. \quad (1)$$

Zhao et al. [56] discovered that with small batches, exact ground-truth labels can be inferred by analyzing the signs of the shared gradients, enabling more efficient data reconstruction. Subsequent works [13, 37, 49] developed approximate label inference techniques for larger batches. Geiping et al. [22] found that cosine distance is more effective than Euclidean distance for data reconstruction and introduced total variation to regularize the reconstructed images.

Recent advancements [27, 32, 52, 53] have improved GLAs by allowing the server to access more information from clients. Yin et al. [52] assumed that batch normalization statistics are accessible, which provide additional cues to regularize the reconstructed images. Balunovic et al. [4] re-approached existing attacks through a Bayesian lens, proposing the Bayes attack to enhance reconstruction quality by leveraging prior knowledge. Other studies [27, 32, 53] took a step further by positing that servers possess knowledge of clients' data distributions, allowing the server to train a generative model for data reconstruction. They optimized the latent vectors of the generative model to output images that can generate gradients similar to the shared gradients. Nonetheless, some literature [16, 26, 47, 50] has criticized these assumptions as not conforming to actual FL environments.

2.3 Gradient Leakage Defense

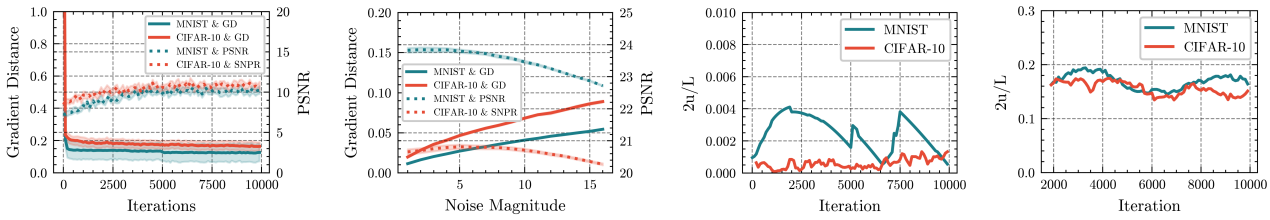
Defenses against gradient leakage can be classified into two main categories: cryptography-based methods [7, 41, 54] and perturbation-based methods [19, 46, 47]. Cryptography-based methods secure gradients by ensuring that the server only sees the plaintext of the aggregated gradients [3, 7, 41, 54]. This makes data reconstruction more challenging because the aggregated gradients imply larger batches, expanding the search space and complicating reverse engineering. However, cryptography-based methods rely on complex operations like modular arithmetic and exponentiation. These result in significant computational and communication overheads, making them unsuitable for secure aggregation against gradient leakage in resource-constrained environments. Moreover, their efficacy diminishes if attackers can reconstruct high-fidelity data from larger batch gradients [48], and when the server is allowed to send maliciously-crafted parameters that can exclude non-interested clients' gradients from the decrypted gradients [43].

Perturbation-based methods offer a more lightweight alternative by slightly modifying gradients to obscure the server. This strategy, however, incurs a delicate trade-off between utility and privacy where more substantial modifications yield better protection but also degrade the gradient utility. Differential privacy (DP) [23] introduces random noises to perturb ground-truth gradients. Gradient sparsification [53] retains only the most significant gradient elements, and gradient quantization [53] represents gradient values with fewer bits. Recent advancements include Soteria [46, 48], OUTPOST [47], and Guardian [19], which selectively perturb the most cost-effective gradient elements.

2.4 Threat Model

We consider the server to be honest-but-curious [58], indicating that it honestly obeys the FL training protocol, yet attempts to recover clients' data from shared gradients. This assumption is more reflective of real-world situations [16, 39] than a malicious model that permits the server to take malicious actions to facilitate reconstruction. Such malicious actions, such as modifying the training procedure or model parameters [6, 12, 51, 57], could be easily spotted and undermine attack stealthiness [16]. Moreover, given the effectiveness of label inference techniques [13, 37, 49], the server's primary objective is to recover clients' data.

Server's knowledge and capability. According to the basic FL training protocol [39], the server has full access to parameters of clients' uploaded models and the old global model, along with knowledge of training configurations such as learning rate and batch size. The server is not allowed to access auxiliary information such as clients' batch normalization statistics and data distributions. This restriction is justified, as practical clients are not required to furnish this information to the server [16, 26, 47]. Notice that, while this limitation indeed increases our attack difficulty, it also broadens the applicability of FEDLEAK across various FL scenarios. Moreover, for thoroughness, we still include a comparative analysis of attack methods that assume access to this auxiliary information in our evaluations. Finally, the server possesses moderate computational resources necessary for executing attacks.



(a) GD (b) GD (c) iDLG (d) FEDLEAK

Figure 1: Figure 1a shows the L_2 GD between the images recovered by iDLG and the corresponding ground-truth images. Figure 1b demonstrates the L_2 GD between the ground-truth images and the same images with varying levels of random sign noise added. Figure 1c and Figure 1d illustrate the values of $\frac{2\mu}{L}$ associated with x' achieved by the gradient descent algorithm and FEDLEAK, both with a step size of 1×10^{-4}. All results are averaged over 100 samples.

We defer our evaluation protocol, i.e., the configuration of practical FL environments, to Section 5. For now, we just highlight that clients can perturb their ground-truth gradients to upload.

3 Gradient Leakage Attack Revisited

For the sake of discussion, let us assume that the gradients $\hat{g}$ estimated by the server are identical to the gradients generated by local data (x, y), i.e., $\hat{g} = \nabla_w \mathcal{L}(F(x, w), y)$. We also assume that the labels inferred by the server are the same as y. Note that $\hat{g}$ can either be treated as the gradients from an individual client or as the aggregated gradients. Let us first re-examine the gradient matching problem, formulated as follows:

$$x' = \arg\min_{x'} \text{dist}(\nabla_w \mathcal{L}(F(x', w), y), \hat{g}), \tag{2}$$

where $\text{dist}(\cdot, \cdot)$ denotes a certain distance metric.

The hypothesis of multiple solutions is likely a misunderstanding. Most studies [26, 50, 56] have empirically demonstrated that images recovered through gradient optimization algorithms for Equation (2) are quite noisy, giving rise to the hypothesis of multiple noisy solutions beyond just x. This means different input samples can yield identical gradients[1]. To illustrate this rmore concretely, consider the case of training a three-hidden-layer network with ReLU activation function on two benchmark datasets MNIST and CIFAR-10. We employ iDLG [56] and observe the reconstruction performance of this attack with a batch size of 1 (see Appendix A for more attacks). Figure 1a illustrates that the L_2 gradient distance (GD) and PSNR between x' and x appear to converge to approximately 0.15∼0.2 and 10∼11, respectively. A PSNR value of 10∼11 indicates that the recovered images are notably noisy [19]. Furthermore, the L_2 GD of 0.15∼0.2 seems quite small at first glance[2], potentially leading to an interpretation that GLAs find noisy data that produce gradients almost identical to those of x, thereby reinforcing the hypothesis of multiple solutions. We first clarify that this interpretation is a misunderstanding. Specifically, the recovered noisy images do not constitute a solution to Equation (2), rather, they are attributable to an improper handling of Equation (2).

To substantiate the above claim, we generate Gaussian random noises for 100 input images, then multiply its sign by a strength of $\frac{1}{255} \sim \frac{16}{255}$, and add them to the input images. This process is repeated 100 times, and Figure 1b shows the expected L_2 GD for varying noise strengths. Empirical studies [5] say that, for low-resolution images (32×32), the L_∞ distance between two similar images should be less than $\frac{8}{255}$. Generally, similar gradients arise from similar images, suggesting that the threshold for assessing gradient similarity can be roughly approximated by the GD between an image and its perturbed counterpart at a noise strength of $\frac{8}{255}$. However, as shown in Figure 1b, even when the noise strength reaches as high as $\frac{16}{255}$, the expected L_2 GD stays below 0.1. Consequently, a loss value (GD) around 0.2, as achieved by iDLG, signifies fundamental non-convergence in Equation (2) and falls far short of indicating that the dummy data have similar gradients to x. Besides, this non-convergence is unlikely due to being trapped in local optima or saddle points, as the use of multiple restarts [38] and temporarily large learning rates [35] did not ameliorate the situation (Appendix A). To further clarify this point, we now take a closer look at Equation (2).

The examination of Equation (2). At its core, Equation (2) can boil down to solving the following system of nonlinear equations:

$$\nabla_w \mathcal{L}(F(x', w), y)[i] = \hat{g}[i], \; i = 1, \cdots, |\hat{g}|. \tag{3}$$

Due to $\hat{g} = \nabla_w \mathcal{L}(F(x, w), y)$, the system of nonlinear equations has at least one solution $x' = x$, suggesting that the system is not ill-conditioned. Furthermore, it is highly likely

[1] $f(x) = \sin(x)$ has the same gradients at $x = 0$ and $x = 2\pi$. In batches, gradient equivalence occurs through (1) direct gradient matching between distinct inputs, or (2) coincidental averaging of gradients from multiple samples (e.g., $f'(3) + f'(-1) = f'(0) + f'(2)$ for $f(x) = x^2$ where the gradients at any two points are not the same). We here examine the former one.

[2] The absolute difference between two gradient elements is on the order of 10^{-4}.

that the solution is unique, i.e., $\nabla_w \mathcal{L}(F(x',w),y) \neq \nabla_w \mathcal{L}(F(x,w),y)$ for $x' \neq x$, for several reasons. Firstly, in Equation (2), the number of constraints (the number of model parameters) significantly exceeds the number of variables (the dimensionality of the input data)[3]. The over-constrained nature of the system reduces the chances of multiple solutions existing. Actually, for fully connected neural networks, the solution is guaranteed to be unique, as substantiated by Theorem 1. Moreover, we highlight that Theorem 1 can, to some extent, be applied to convolutional neural networks (CNNs) and transformers, as convolutional layers and attention layers can be viewed as specialized variants of fully connected layers [44].

Theorem 1. *Consider F whose first hidden layer is a fully connected layer, followed by at least one more fully connected layer. If gradients of $\mathcal{L}$ with respect to the outputs of these fully connected layers are not the zero vectors, then for any $x' \neq x$, it holds that $\nabla_w \mathcal{L}(F(x',w),y) \neq \nabla_w \mathcal{L}(F(x,w),y)$ [22].*

Intuitively, different inputs generally yield distinct intermediate outputs within F. These intermediate outputs affect the computation of the gradient of F's parameters, making it challenging for different inputs to produce identical gradients. In other words, if there exist two different inputs that generate identical gradients, it would imply that F's error feedback in the parameter space for these inputs is the same, which is highly improbable unless the network degenerates significantly, for example, if all parameters are zero. In summary, the noisy solutions are primarily due to the ineffectiveness of existing optimization algorithms in solving Equation (2), motivating us to identify the specific conditions that allow for effective solutions to Equation (2).

Convergence conditions. Let $g' = \nabla_w \mathcal{L}(F(x',w),y)$. We use x'_j and g'_j to denote the starting point for x' and the gradients generated by x'_j at the j-th iteration. The update rule of the gradient descent algorithm for x' at j-th iteration is given by $x'_{j+1} = x'_j - \eta \nabla_{x'_j} \text{dist}(g'_j, \hat{g})$ where η is the step size. To ensure convergence of x' toward x, we require $||x - x'_{j+1}||_2^2 < ||x - x'_j||_2^2$. Then, expanding $||x - x'_{j+1}||_2^2$ produces:

$$\begin{aligned} ||x - x'_{j+1}||_2^2 &= ||x - x'_j + \eta \nabla_{x'_j} \text{dist}(g'_j, \hat{g})||_2^2 \\ &= ||x - x'_j||_2^2 - 2\eta \langle x'_j - x, \nabla_{x'_j} \text{dist}(g'_j, \hat{g}) \rangle \\ &+ \eta^2 ||\nabla_{x'_j} \text{dist}(g'_j, \hat{g})||_2^2. \end{aligned} \tag{4}$$

Let us impose the following conditions to hold:

$$\begin{aligned} \langle x'_j - x, \nabla_{x'_j} \text{dist}(g'_j, \hat{g}) \rangle &> \mu \, ||x - x'_j||_2^2, \\ ||\nabla_{x'_j} \text{dist}(g'_j, \hat{g})||_2^2 &< L \, ||x - x'_j||_2^2. \end{aligned} \tag{5}$$

[3]For example, a ResNet10 model for MNIST has 4902090 parameters. For batch sizes of 1, 8, 32, 64, and 128, the corresponding input dimensions are 784, 6272, 25088, 50176, and 100352 respectively. The number of model parameters is 6252, 781, 195, 97, and 48 times the input dimensions for these batch sizes respectively.

Substituting the above conditions into Equation (4) yields:

$$||x - x'_{j+1}||_2^2 < (1 + \eta^2 L - 2\eta\mu) ||x - x'_j||_2^2. \tag{6}$$

To guarantee convergence, it is essential that $1 + \eta^2 L - 2\eta\mu \leq 1$, i.e., $0 < \eta < \frac{2\mu}{L}$. Here, μ measures the curvature of the loss landscape near minima, while L bounds the gradient smoothness (smaller L implies smoother, more consistent gradients). Larger μ and smaller L together broaden the range of safe learning rates η that ensure convergence. We further evaluate the value of $\frac{2\mu}{L}$ associated with x' over different optimization iterations, as shown in Figure 1c. We see that the value of $\frac{2\mu}{L}$ is very small and susceptible to fluctuations, which poses a significant challenge in determining an appropriate step size η for addressing the gradient matching problem with gradient optimization algorithms. This also demonstrates that the failure of existing GLAs is due to a lack of an effective way to solve the gradient matching problem, rather than the existence of multiple feasible solutions. In light of these observations, we develop FEDLEAK, designed to more effectively resolve Equation (2). The details of FEDLEAK will be elaborated in the following section.

Remark. The analysis in this section relies on the idealized assumption of $\hat{g} = \nabla_w \mathcal{L}(F(x,w),y)$, which may not hold. This is primarily because clients might perform multiple local steps or upload perturbed gradients, which obfuscates the server from obtaining the exact gradients with respect to (x,y). Despite this difference from reality, we highlight that, in most cases, $\hat{g}$ are still close enough to $\nabla_w \mathcal{L}(F(x,w),y)$ to yield meaningful reconstructions (as shown in Section 6.3), since excessive perturbation or an overly large number of local steps can lead to a significant degradation in model performance. Additionally, we provide an analysis of the impact of such estimation errors in Appendix B for interested readers seeking further insights. We will not elaborate on this here, as it is not the primary focus of this paper.

4 Our Attack: FEDLEAK

4.1 Overview

To efficiently address the gradient matching problem, FEDLEAK harnesses two novel techniques, partial gradient matching and gradient regularization. The partial gradient matching selectively matches only specific portions of gradients and can be regarded as a generalized form of gradient matching. This technique not only significantly reduces μ, but also ensures that the optimal solutions remain consistent with those derived from complete gradient matching. Moreover, to increase the value of L, FEDLEAK incorporates a gradient regularization term into partial gradient matching. The inclusion of the gradient regularization term, however, requires the computation of Hessian matrix, a computationally intensive task. To alleviate this burden, we propose an approximation method to enhance the practicality and scalability of

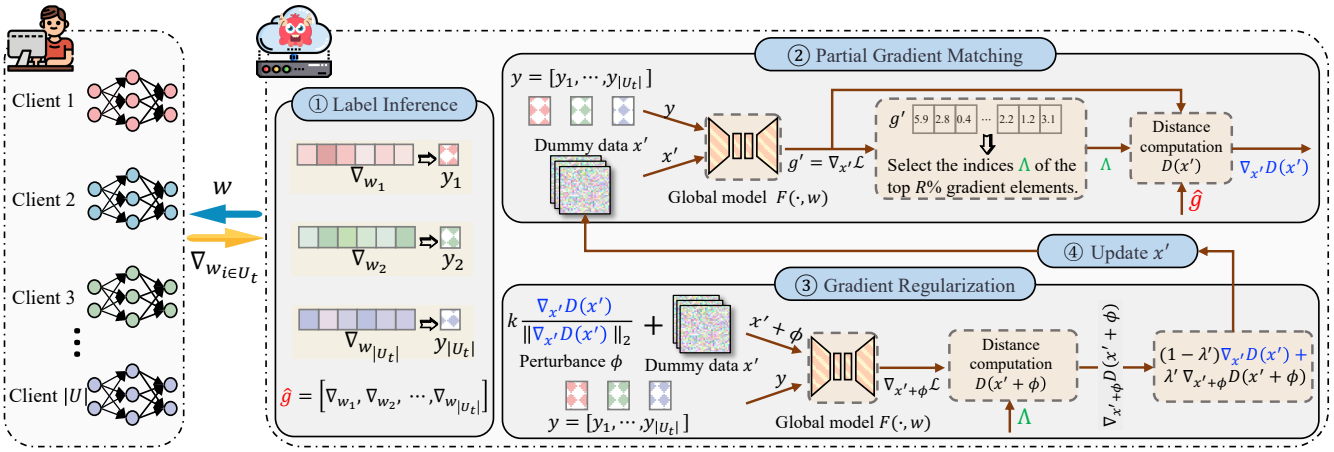


Figure 2: The overview of the proposed attack FEDLEAK.

Algorithm 1 Our attack: FEDLEAK

Require: The global model F and parameters w; the gradients estimated by the server $\hat{g}$; the loss function (cross-entropy loss) $\mathcal{L}$; the step size η; the blend factor λ'; the matching ratio R; maximum attack iterations I;

1: Infer the ground-truth labels y from $\hat{g}$ using existing label inference techniques.
2: Initialize dummy data x' with a uniform distribution ranging from 0 to 1 and $x'_0 = x'$.
3: **for** $j \leftarrow 0$ to $I-1$ **do**
4: Compute the gradients of the loss with respect to dummy data $g'_j = \nabla_{x'_j}\mathcal{L}(F(x'_j, w), y)$.
5: Select the indices of the top R% gradient elements with the largest magnitude in g'_j to form Λ.
6: Compute the distance $D(x'_j)[\Lambda]$ based on Equation (15).
7: Compute $\phi = k\frac{\nabla_{x'_j}D(x'_j)[\Lambda]}{||\nabla_{x'_j}D(x'_j)[\Lambda]||_2}$ and the distance $D(x'_j+\phi)[\Lambda]$ based on Equation (15).
8: Mix $\nabla_{x'_j}D(x'_j)[\Lambda]$ and $\nabla_{x'_j+\phi}D(x'_j+\phi)[\Lambda]$ using the blend factor λ' to update x'_j with step size η.
9: Project x'_j onto feasible domain $[0,1]$ to obtain x'_{j+1}.
10: **end for**
11: **Return** the recovered data $x' = x'_I$.

FEDLEAK. Figure 1d demonstrates that the combination of the two techniques yields higher and more stable values of $\frac{2\mu}{L}$. Algorithm 1 summarizes FEDLEAK and Figure 2 presents the entire attack process of FEDLEAK.

4.2 Less is Better: Partial Gradient Matching

The core idea of partial gradient matching is to align only a subset of gradient elements rather than all. Consider an index set Λ containing the indices of the gradient elements to be matched. The partial gradient matching problem can be formulated as follows:

$$x' = \arg\min_{x'} \text{dist}(g'[\Lambda], \hat{g}[\Lambda]). \tag{7}$$

When Λ contains all gradient elements' indices, the partial gradient matching degenerates into the original gradient matching problem (Equation (2)). As illustrated in Theorem 2, the partial gradient matching problem (Equation (7)) possesses several favorable properties compared to Equation (2).

Theorem 2. *Should Equation (3) be overdetermined and free of redundant constraints*[4]*, there exists a certain index set Λ such that Equation (7) not only preserves the same solution as Equation (2), but also enjoys a higher value of μ than that of Equation (2), with dist$(\cdot,\cdot)$ being specified as the L_1 distance*[5]*.*

The first part of Theorem 2 asserts that Equations (2) and (7) yield the same solution, which is obvious from the definition of overdetermined systems. We mainly demonstrate the second part: Equation (7) achieves a higher value of μ than Equation (2). By setting dist$(\cdot,\cdot)$ as L_1 distance, $\langle x'-x, \nabla_{x'}\text{dist}(g', \hat{g})\rangle$ can be written as follows:

$$\begin{aligned}\langle x'-x, \nabla_{x'}\text{dist}(g', \hat{g})\rangle &= \langle x'-x, \frac{1}{|g'|}\frac{\partial}{\partial x'}\sum_i |g'[i]-\hat{g}[i]|\ \rangle \\ &= \langle x'-x, \frac{1}{|g'|}\sum_i \text{sign}(g'[i]-\hat{g}[i])\frac{\partial g'[i]}{\partial x'}\ \rangle. \end{aligned} \tag{8}$$

[4]Redundant constraints refer to equivalent constraints. This assumption primarily ensures that removing some constraints does not render Equation (3) underdetermined. In practice, it is uncommon for the removal of some constraints to significantly alter the solution, allowing us to relax our concerns about this assumption.

[5]L_1 distance evaluates each gradient element uniformly and can facilitate our discussion. Other distance functions are essentially equivalent to considering the gradients with different weights. For example, the L_2 distance prioritizes gradient elements that have yet to be well-aligned.

Based on Theorem 2 in Appendix G, if Λ is chosen to include indices corresponding to the terms with the largest inner product values in Equation (8), we arrive at Equation (9):

$$\begin{aligned} &\frac{1}{|\Lambda|}\sum_{i\in\Lambda}\langle x'-x, \text{sign}(g'[i]-\hat{g}[i])\frac{\partial g'[i]}{\partial x'}\rangle \\ &\geq \frac{1}{|\hat{g}|}\sum_{i}\langle x'-x, \text{sign}(g'[i]-\hat{g}[i])\frac{\partial g'[i]}{\partial x'}\rangle. \end{aligned} \tag{9}$$

Equation (9) suggests $\langle x'-x, \nabla_{x'}\text{dist}(g'[\Lambda], \hat{g}[\Lambda])\rangle \geq \langle x'-x, \nabla_{x'}\text{dist}(g', \hat{g})\rangle$. Thus, there exists a certain index set Λ that yields a higher value μ.

The remaining challenge is how to determine the appropriate Λ. While the preceding analysis suggests that Λ ought to include indices associated with the largest inner product values, identifying these requires evaluating the gradient for each individual gradient element $g'[i]$ with respect to x', i.e., $\frac{\partial g'[i]}{\partial x'}$. Given the vast number of parameters in DNNs, this is computationally prohibitive. To address this issue, we present an approximate solution here.

Generally, within a sufficiently small neighborhood, aligning each gradient element should produce a gradient direction positively correlated with $x'-x$ [8], meaning that each term is usually positive. As a result, our task reduces to finding the indices i for which $\frac{\partial g'[i]}{\partial x'}$ has bigger magnitude. Moreover, the term $\frac{\partial g'[i]}{\partial x'}$ with greater magnitude often corresponds to $g'[i]$ with large magnitude. To clarify, according to the chain rule, there is $\frac{\partial g'[i]}{\partial x'} = \frac{\partial g'[i]}{\partial \mathcal{L}}\frac{\partial \mathcal{L}}{\partial x'}$. As can be seen, $\frac{\partial \mathcal{L}}{\partial x'}$ remains identical across gradient elements and we focus on the first term, which measures how the gradient changes in response to variations in the loss function. We can quantify this sensitivity using Fisher information of $g'[i]$, approximated by the square of $g'[i]$ [36]. Empirical validation of this can be found in Appendix C, where the correlation coefficient between the magnitude of gradient elements and their sensitivity to the input data is around 0.7372, a statistically significant value. To sum up, we match the top $R\%$ gradient elements with the largest magnitude in g'.

The philosophy behind partial gradient matching is intuitive. Gradient elements with larger magnitudes correspond to key parameters vital for capturing significant feature information, while gradient elements with smaller magnitudes are related to less influential parameters that primarily contribute noise to the optimization landscape [33].

4.3 Gradient Regularization

For simplicity, let us define $D(x')$ as $\text{dist}(g'[\Lambda], \hat{g}[\Lambda])$ where $g' = \nabla_w \mathcal{L}(F(x', w), y)$. From Equation (5), it is clear that a smaller $\nabla_{x'}D(x')$ along the entire optimization path can enable a smaller value of L. To leverage this insight, we introduce a gradient regularization term in Equation (7) as follows:

$$x' = \arg\min_{x'} D(x') + \lambda||\nabla_{x'}D(x')||_2, \tag{10}$$

where λ is a weighting factor to balance the two terms in Equation 10. Gradient-based optimization methods are commonly employed to solve Equation (10). However, as indicated in Equation (11), the gradients of Equation (10) involve the Hessian matrix evaluated at x', which is computationally prohibitive in high-dimension spaces.

$$\nabla_{x'}\left(D(x')+\lambda||\nabla_{x'}D(x')||_2\right) = \nabla_{x'}D(x') + \lambda H\frac{\nabla_{x'}D(x')}{||\nabla_{x'}D(x')||_2}.$$

To circumvent the need for direct Hessian evaluation, we present an approximate estimation for $H\frac{\nabla_{x'}D(x')}{||\nabla_{x'}D(x')||_2}$. Consider a random small perturbance ϕ applied to x'. A Taylor expansion of $D(x'+\phi)$ yields:

$$D(x'+\phi) = D(x') + \nabla_{x'}D(x')^\top\phi. \tag{11}$$

Differentiating both sides of Equation (11) results in:

$$\nabla_{x'+\phi}D(x'+\phi) = \nabla_{x'}D(x') + H\phi. \tag{12}$$

Setting ϕ in the direction of $\nabla_{x'}D(x')$ leads to:

$$H\frac{\nabla_{x'}D(x')}{||\nabla_{x'}D(x')||_2} = \frac{\nabla_{x'+\phi}D(x'+\phi) - \nabla_{x'}D(x')}{k}. \tag{13}$$

where $\phi = k\frac{\nabla_{x'}D(x')}{||\nabla_{x'}D(x')||_2}$ and k is a small constant ensuring ϕ remains small. We then substitute Equation (13) into Equation (11):

$$\begin{aligned} &\nabla_{x'}D(x') + \frac{\lambda}{k}(\nabla_{x'+\phi}D(x'+\phi) - \nabla_{x'}D(x')) \\ &= (1-\frac{\lambda}{k})\nabla_{x'}D(x') + \frac{\lambda}{k}\nabla_{x'+\phi}D(x'+\phi). \end{aligned} \tag{14}$$

Equation (14) only involves first-order gradients (linear computation time) and is more efficient than direct Hessian evaluation, which demands cubic computation time. Moreover, if we treat $\frac{\lambda}{k}$ as a constant between 0 and 1, Equation (14) essentially blends the gradients generated by x' and $x'+\phi$. Therefore, we instead set $\lambda' = \frac{\lambda}{k}$ and tune λ' within the interval $[0,1]$ to modulate the strength of the gradient regularization.

A closer examination of Equation (14) reveals its actual effect in the optimization process. Notice that, without gradient regularization, each update can be viewed as moving x' a certain distance in the direction of ϕ. When the gradients at x' and $x'+\phi$ oppose each other, it means that the step taken may be excessively large, potentially overshooting the optimal point. By blending the gradients from both locations, we can effectively dampen the gradient at x', which acts as a form of adaptive step size adjustment. Thus, it is intuitive that Equation (14) indeed improves convergence by dynamically adjusting step sizes based on gradient alignment, allowing for more effective navigation through the complex landscape of the loss function. We provide an illustrative example in Appendix D.

4.4 Concrete Formulation

Loss function. Our distance metric consists of four terms, including L_1 distance, cosine distance, total variation (TV), and activation value penalty. Drawing inspiration from [18], we employ both L_1 distance and cosine distance to quantify the gradient distance. The choice of L_1 distance is due to its resilience against gradient perturbations, which can occur from various defense methods, making it preferable over L_2 distance. Moreover, TV helps capture differences between neighboring pixels, thereby improving the fidelity of the reconstructed images, given that natural images exhibit smooth transitions in adjacent pixels. The activation value penalty applies an L_1 penalty to the outputs of the model's intermediate layers, promoting sparsity in these activations. This is based on the observation that pre-trained and subsequently fine-tuned models often produce sparse activations, with many neurons having near-zero values for natural images [44]. The ultimate $\text{dist}(\cdot,\cdot)$ is formulated as follows:

$$\text{dist}(g'[\Lambda], \hat{g}[\Lambda]) = \underbrace{||g'[\Lambda] - \hat{g}[\Lambda]||_1}_{\text{L}_1\text{distance}} + \underbrace{1 - \frac{g'[\Lambda]^\top \hat{g}[\Lambda]}{||g'[\Lambda]||_2 ||\hat{g}[\Lambda]||_2}}_{\text{cosine distance}} + \alpha \cdot \underbrace{\text{TV}(x')}_{\text{total variation}} + \beta \cdot \sum_l \underbrace{||F^{(l)}(x', w)||_1}_{\text{activation value penalty}}, \quad (15)$$

where $F^{(l)}(x', w)$ denotes the activation values of the l-th layer of the model given the input x'. α and β are used to balance the influence of the two terms.

Optimization. Any gradient-based optimization method can be employed to address Equation (15). However, there is an additional constraint: pixel values are required to be limited within the range $[0, 1]$. To adhere to this constraint, we include a projection step, as shown in the 9th step of Algorithm 1, clamping pixel values to this range.

5 Evaluation Protocol

Current evaluation protocols do not adequately capture the nuances of real-world FL environments. To fill this gap, we develop an evaluation protocol that faithfully reflects production environments, drawing from our investigation (Appendix E). We identify eight critical factors that significantly influence attack performance and align their values with those found in production environments, establishing a more practical evaluation criterion. Table 1 provides a comparison of their values in previous studies versus those in this paper:

- **Batch size.** The batch size determines the number of optimization variables included in the gradient matching problem. When the batch size increases, there are more variables to optimize, making it harder to reconstruct clients' data. Appendix E shows that practical batch sizes typically range from 16 to 64.

- **Model.** Beyond CNNs, transformer models are emerging as competitive alternatives. Appendix E illustrates the usage frequency of both types in practical applications, revealing that CNNs are predominant while transformer models are less common. Thus, although transformer models will be evaluated, the primary focus remains on CNNs. We evaluate four distinct models: ResNet, DenseNet, MobileNet, and Vision Transformer (ViT). The first three are based on convolutional structures, and ViT is based on transformers. Moreover, to provide a comprehensive evaluation, we employ variants of ResNet with 10, 18, and 34 layers, as well as expanded channel widths for ResNet10, labeled as ResNet10_x3 and ResNet10_x5, to study how model complexity impacts attack performance.

- **Dataset selection.** Real-world datasets are more sophisticated than commonly used small benchmark datasets, such as MNIST, SVHN, CIFAR-10, and CIFAR-100. Alongside benchmark datasets, we include three more sophisticated datasets, ImageNet, HAM10000, and Lung-Colon Cancer. The latter two are noteworthy as they pertain to medical imaging. HAM10000 contains 10000 dermatoscopic images of pigmented skin lesions across seven classes, while Lung-Colon Cancer features 25000 histopathological images categorized into five classes. Given that FL is primarily applied in privacy-sensitive fields like healthcare, and considering the substantial differences between medical images and common images, evaluating GLAs in HAM10000 and Lung-Colon Cancer can provide a clearer picture of their practical implications.

- **Attack stage.** Attacks can occur at any communication round in FL. However, the quality of reconstructed data is significantly higher in the initial communication rounds compared to later ones [4, 16]. Note that clients of interest may not participate in the initial communication rounds. Therefore, rather than focusing solely on the attack performance of GLAs in the initial rounds, we evaluate their performance both at the onset and during later phases of FL to better understand the overall privacy leakage risk associated with FL.

- **Initialization method.** Some studies [4] found that GLAs perform better in untrained models with parameters initialized through wide uniform distributions. However, Appendix E illustrates that, in practice, about 70% of cases use default random initialization, while a smaller percentage employs pre-trained weights. We focus on default random initialization and pre-trained weights.

- **Auxiliary information.** We argue against the use of auxiliary information for two reasons. First, such information may not always be available in real-world scenarios [26, 47], limiting the generalizability of attack methods depending on it. Second, methods with auxiliary information often incur

Table 1: The configurations considered by existing attack methods. When multiple settings are employed, we report the most sophisticated configuration. The ConvNet is a small CNN developed by Zhu et al. [58]. Yin et al. [52] utilized a batch size of 48, but the resulting reconstructions are of low fidelity. CNNs indicate the use of various CNN architectures. Some methods employ ImageNet but resize images from 224 to either 64 or 128. "Early" and "Late" indicate when the attack occurs: during the first communication round and late training phase, respectively. "Default" refers to initialization methods provided by PyTorch while "Wide Uniform" indicates the use of a wide uniform distribution for initializing model parameters. Instances where authors do not provide specific details are marked as "N/A," and a "−" indicates no auxiliary information used.

Evaluation Setting	Batch Size	Model	Dataset	Attack Stage	Initialization Method	Auxiliary Information	Local Step
Zhu et al. [58]	≤ 8	ConvNet	Small Datasets	Early	Wide Uniform	–	1
Zhao et al. [56]	≤ 8	ConvNet	Small Datasets	Early	Wide Uniform	–	1
Geiping et al. [22]	1	ResNet	ImageNet	Early	Pretrained	–	1
Yin et al. [52]	≤ 48	ResNet	ImageNet	Early	Pretrained	Batch Normalization Statistics	1
Balunovic et al. [4]	1	ConvNet	Small Datasets	Early & Late	Wide Uniform	–	1
Jeon et al. [27]	≤ 48	ResNet	ImageNet (64×64)	Early	Default	Clients' Data Distribution	1
Li et al. [32]	1	ResNet	ImageNet	Early	Default	Clients' Data Distribution	1
Yue et al. [53]	16	CNNs	ImageNet (128×128)	Early	N/A	Clients' Data Distribution	5
Hu et al. [25]	16	CNNs	Small Datasets	Early	N/A	–	5
Ours	≤ 64	CNNs & Transformers	ImageNet & Medical Datasets	Early & Late	Default & Pretrained	–	12

high attack costs and complexities, making them less practical to implement. At the attack design level, FEDLEAK does not utilize such information.

- **Local step.** To reduce communication costs, clients can perform several local training steps. More local steps mean that model updates sent to the server are based on a larger amount of data. Also, the gradients estimated by the server are cumulative from each local step, rather than reflecting the gradients from all relevant data points, leading to estimation errors. Both of these two aspects make it harder to recover clients' data. Appendix E indicates that in real-world environments, using a local step of 1 and training for one epoch locally are the two most common configurations.

- **Defenses.** In real-world scenarios, as shown in Appendix E, most studies avoid cryptography-based methods due to their substantial computational costs. Instead, the majority of studies favor Gaussian-DP [23]. In addition to DP, we also include four state-of-the-art defense methods: quantization [53], Soteria [48], OUTPOST [47], and Guardian [19].

Evaluation approach. Our evaluation begins with a seed configuration using ResNet10, small benchmark datasets, ImageNet-pretrained weights, and a local step of 1. We then progressively explore more intricate configurations by changing one factor at a time and keeping track of the most challenging values encountered.

Generic setup for FL. The remaining FL configurations largely follow existing works [19, 46, 48]. We utilize an FL scenario with 100 clients collaborating to train a global model. We consider both IID and Non-IID settings. In the IID setting, the training dataset is randomly partitioned into 100 equal subsets, one per client. See Section 6.3 for Non-IID settings. The server performs average aggregation, and each client updates its local model with a learning rate of 1×10^{-4}.

Attacks. We select three state-of-the-art attacks, iDLG [56], InvertingGrad (IG) [22], and GradInversion (GI) [52] as baselines. Additionally, although we assume that clients' data distributions are inaccessible, we also include comparisons with generative model-based methods, i.e., GGL [32] and ROGS [53], to demonstrate the effectiveness of FEDLEAK.

Evaluation metrics. To assess the similarity between reconstructed images and their original counterparts, four metrics are used, including PSNR (↑), SSIM (↑), LPIPS (↓), and attack success rate (ASR) (↑). Here, (↑) indicates that larger values represent better results and (↓) signifies the opposite. PSNR measures logarithmic L_2 distance between reconstructed and original images. SSIM quantifies the structural similarity between two images. LPIPS measures perceptual similarity by comparing DNN-extracted features. For ASR, we enlist three human volunteers to determine whether the entities in the two images have a high semantic overlap and report the proportion of instances with strong overlap.

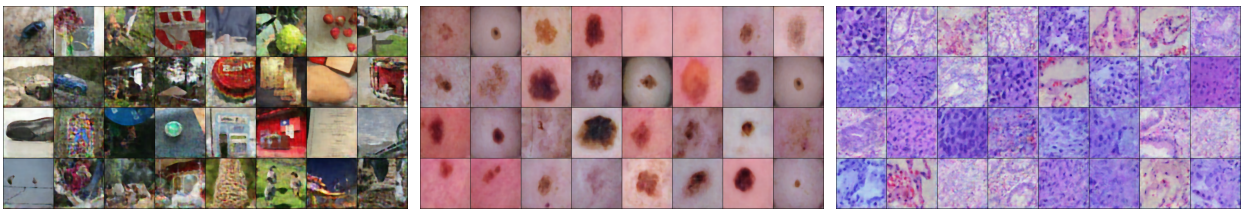
Label inference. We infer ground-truth labels by aggregating gradients from the final fully connected layer along the feature dimension and identifying the indices corresponding to minimum values [52]. Formally, let $\Delta W \in \mathbb{R}^{K \times N}$ denote the gradient matrix where K is the feature dimension and N is the total number of classes. The inferred labels are computed as: $y' = \arg\min(\sum_{i=1}^{K} \Delta W[i,:])[:\text{batch size}]$. If the batch size exceeds the number of classes, we infer labels based on gradient magnitudes. The count for each class j is calculated as: $c_j = \lfloor \text{batch size} \times \frac{\sum_{i=1}^{K}(\Delta W[i,j] - \max W)}{\sum_{i=1}^{K}\sum_{j=1}^{N}(\Delta W[i,j] - \max W)} \rfloor$. If $\sum c_j <$ batch size, we supplement the batch by iteratively adding samples from the class with the lowest gradient strength until the batch size requirement is satisfied. We also jointly optimize the inferred labels alongside the input images during the inversion process. Appendix F examines the impact of label inference.

Table 2: A summary of the configurations across different sections along with corresponding privacy leakage risks. In particular, Section 6.3 includes four configuration changes, labeled ① through ④, with red indicating weaker configurations to be discarded. ② and ④ involve multiple variables. Moreover, in ④, clients have the option to perform a local step of 1 with batch sizes of 16, 32, or 64, or alternatively, they can train for one local epoch using a batch size of 16 (which corresponds to 12 local steps in the Lung-Colon Cancer dataset). The latter is more challenging. The risk levels are categorized into three levels: Very High (PSNR of ≥ 20), High (PSNR of $15 \sim 20$), and Medium (PSNR of $10 \sim 15$), which indicate that nearly all, most, and only a small portion of clients' data can be semantically reconstructed, respectively.

Config.	Batch	Model	Dataset	Attack Stage	Initialization	Heterogeneity	Local Step	Defense	Risk
Sec. 6.1	16,32,64	ResNet10	Benchmark Datasets	Early & Late	Pretrained	IID	1	✗	Very High
Sec. 6.2	16,32,64	ResNet10	Sophisticated Datasets	Early & Late	Pretrained	IID	1	✗	High to Very High
Sec. 6.3	(④ 16,32,64) vs. 16	(③ Others) vs. ResNet10	Lung-Colon Cancer	(② Early) vs. Early & Late	(② Random) vs. Pretrained	(① Non-IID) vs. IID	(④ 1) vs. 12	✗	Medium to High
Sec. 6.4	16	ResNet10	Lung-Colon Cancer	Early & Late	Pretrained	IID	12	✔	Medium

Table 3: The PSNRs of reconstructed images using four attack methods against FL configurations with benchmark datasets and varying batch sizes. The best results are given in bold.

Dataset	MNIST				SVHN				CIFAR-10				CIFAR-100			
Batch Size	iDLG	IG	GI	Ours	iDLG	IG	GI	Ours	iDLG	IG	GI	Ours	iDLG	IG	GI	Ours
16	10.70	10.96	11.57	**22.52**	10.04	10.28	11.43	**21.62**	9.90	10.26	11.26	**23.06**	9.86	10.00	11.42	**22.94**
32	9.39	9.96	11.17	**21.57**	9.53	9.43	10.41	**20.11**	9.51	9.58	10.59	**21.33**	9.39	9.88	10.48	**21.50**
64	9.27	9.88	10.94	**21.12**	8.86	9.37	10.17	**18.09**	8.52	9.42	10.01	**19.83**	8.58	9.40	10.26	**19.38**



(a) ImageNet (b) HAM10000 (c) Lung-Colon Cancer

Figure 3: The images recovered by FEDLEAK in three datasets. The batch size here is 32. See Figure 11 for ground-truth ones and Figure 12 for reconstructions of ROGS.

Hyperparameters and others. Adam optimizer is utilized for Equation (15) with step size $\eta = 1 \times 10^{-4}$ and attack iterations $I = 10000$. The hyperparameters α, β, and λ' for FEDLEAK are configured to 1×10^{-5}, 1×10^{-4}, and 0.7, respectively. We match the most significant 50% of gradient elements, i.e., $R = 50$. Default hyperparameters are used for alternative attack methods, with attack iterations increased to 10000 for convergence. Attacks are carried out at the $2000 \times \{0,1,2,3,4,5\}$-th rounds. For each image, we find the most similar reconstructed image to calculate metrics.

6 Evaluation Results

6.1 Attack in Benchmark Datasets

Table 3 presents the PSNR of reconstructed images using four different attack methods, evaluated across various datasets and batch sizes. Throughout all datasets and batch sizes, FEDLEAK consistently outperforms existing attack methods. For instance, in CIFAR-10 with a batch size of 16, FEDLEAK achieves a PSNR of 23.06, while the other methods lag far behind with PSNRs of 9.90, 10.26, and 11.26, respectively.

6.2 Attack in Sophisticated Datasets

We now evaluate the performance of the attack methods on sophisticated datasets, including ImageNet, HAM10000, and Lung-Colon Cancer. Existing attack methods, including iDLG, IG, and GI, generally exhibit poor reconstruction quality on high-resolution images. To facilitate a quantifiable comparison, we contrast FEDLEAK with GGL and ROGS. For HAM10000 and Lung-Colon Cancer, we resize input images to 224×224. Table 4 reports the performance of the three methods across these sophisticated datasets.

Overall, FEDLEAK significantly outperforms GGL and ROGS by a considerable margin. For example, in HAM10000,

Table 4: The PSNRs of reconstructed images using three attack methods against FL configurations with sophisticated datasets.

Dataset	ImageNet			HAM10000			Lung-Colon Cancer		
Batch Size	GGL	ROGS	Ours	GGL	ROGS	Ours	GGL	ROGS	Ours
16	12.50	16.26	**20.07**	11.76	14.54	**23.35**	10.81	11.33	**18.40**
32	11.31	14.18	**19.07**	10.61	13.28	**22.47**	9.86	10.44	**17.73**
64	10.17	12.96	**19.01**	9.81	11.47	**22.16**	9.48	10.27	**16.06**

Table 5: The performance of FEDLEAK under different non-IID scenarios in Lung-Colon Cancer. We use a batch size of 32. For feature skew, $0 \sim 20\%$ is the most simple group.

Label Skew	1 class	2 class	3 class	4 class	5 class
PSNR	18.82	18.38	17.97	17.85	17.73
Quantity Skew	50	100	150	200	250
PSNR	17.80	17.71	17.76	17.72	17.73
Feature Skew	0-20%	20-40%	40-60%	60-80%	80-100%
PSNR	19.64	18.56	17.84	16.94	16.07

Table 6: The performance of FEDLEAK under varying data heterogeneity levels.

α	0.1	0.5	1	5
ROGS	10.93	10.78	10.40	10.38
Ours	18.06	17.86	17.65	17.60

Table 7: The PSNRs of reconstructed images with different weight initialization methods in Lung-Colon Cancer.

Batch Size	32		64	
Weight	Random	Pre-trained	Random	Pre-trained
ROGS	13.95	10.44	12.04	10.27
Ours	**19.62**	**17.73**	**18.32**	**16.06**

FEDLEAK achieves a PSNR of at least 22.16, while GGL and ROGS reach a maximum PSNR of only 14.54. Crucially, the impressive attack performance of FEDLEAK is attained without needing access to clients' data distribution, whereas both GGL and ROGS do require such access. Figure 3 shows reconstructed batches for all three datasets. We also note that the attack methods tend to be less effective on the Lung-Colon Cancer dataset compared to the others. This is likely due to the intricate structures, e.g., various tissue patterns and cellular details, presented in Lung-Colon Cancer, as shown in Figure 3. In contrast, the images in HAM10000 are relatively simpler and lack such complex structural information, making them easier to reconstruct. To conduct a more rigorous evaluation of FEDLEAK's attack performance, our subsequent experiments will employ Lung-Colon Cancer.

6.3 Attack in Diverse Settings

In this subsection, we examine how the data heterogeneity among clients, the use of pre-trained weights, and the choice of models affect attack performance. We will subsequently increase the number of local steps to create more challenging attack configurations. Finally, to mitigate the risk of FEDLEAK overfitting to a single evaluation metric, we incorporate three supplementary metrics for a more comprehensive assessment.

Evaluation in Non-IID settings. We use Lung-Colon Cancer and stimulate three non-IID scenarios, including label skew (1-5 classes per client), quantity skew (allocating local datasets of $50 \sim 250$ samples), and feature skew (partitioning images into five percentile groups based on their TV values, with lower TV indicating simpler images). As shown in Table 5, FEDLEAK maintains strong attack performance across all non-IID scenarios. Notably, higher label skew (e.g., single-class clients) and simpler features (low TV) are more vulnerable to FEDLEAK, while sample quantity variations show negligible impact. In Table 6, we further test FEDLEAK with different levels of data heterogeneity by varying Dirichlet distribution α values. A smaller α indicates stronger data heterogeneity [6]. We see that data heterogeneity does have some impact on the performance of GLAs, but this effect is not particularly significant. Actually, data heterogeneity mainly affects model performance, with a lesser effect on the gradient matching problem. Therefore, we will retain the current IID data distribution configuration.

Evaluation in randomly initialized models. Table 7 reports the performance of GLAs in the first training round in models with both randomly initialized and pre-trained weights. As anticipated, GLAs indeed present higher attack performance against randomly initialized weights. The reason has been intuitively discussed in Section 1. The attack results in later training phases are not reported since we observe negligible differences in attack effectiveness after several training rounds. Given that pre-trained weights offer greater resistance against GLAs, we will continue to use them moving forward.

Evaluation over different model architectures, depths, and widths. We evaluate the performance of FEDLEAK from three perspectives: model architecture, depth, and width. The results are detailed in Table 8. Across diverse model architectures, depths, and widths, FEDLEAK consistently surpasses the performance of ROGS. We note that larger models exac-

[6] https://github.com/TsingZ0/PFLlib.

Table 8: The PSNRs of reconstructed images with different models in Lung-Colon Cancer.

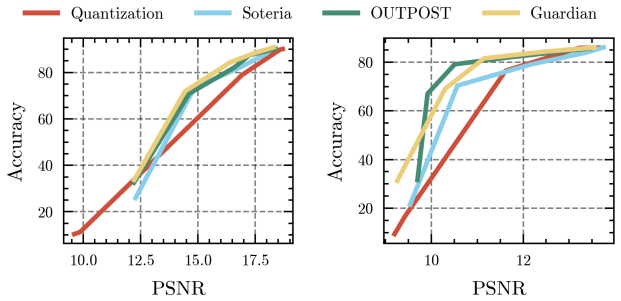
Batch Size	Attack	ResNet10	Depth		Width		Architecture		
			ResNet18	ResNet34	ResNet10_x3	ResNet10_x5	DenseNet121	MobileNetV2	ViT
32	ROGS	10.44	10.66	11.15	10.96	11.12	10.22	9.48	9.01
	Ours	**17.73**	**18.15**	**18.92**	**18.46**	**19.41**	**17.42**	**16.85**	**15.64**
64	ROGS	10.27	10.73	11.22	11.43	11.68	9.90	9.19	8.57
	Ours	**16.06**	**16.94**	**17.98**	**16.64**	**18.18**	**15.50**	**15.20**	**13.98**

Table 9: The PSNRs of reconstructed images with varying local steps in Lung-Colon Cancer.

Local Step	IG	GI	ROGS	Ours
1	8.82	10.12	11.33	**18.40**
2	8.44	9.86	10.98	**16.74**
4	8.53	9.53	10.88	**15.57**
8	8.28	9.41	10.74	**14.34**
12	8.59	9.10	10.43	**13.55**

Table 10: The quality of reconstructed images assessed by three distinct evaluation metrics in Lung-Colon Cancer.

Metric	SSIM	LPIPS	ASR (%)
ROGS	0.55	0.4736	0.35 %
Ours	**0.68**	**0.3447**	**20.40%**



(b) Local Step of 1 (c) Local Step of 12

Figure 4: The utility-privacy trade-offs of various defenses in Lung-Colon Cancer against FEDLEAK.

erbate privacy leakage, aligning with conclusions drawn by [9–11, 22]. Specifically, for FEDLEAK, increasing the model depth from 10 to 34 layers leads to PSNR improvements of 1.19 and 1.92 for batch sizes of 32 and 64, respectively. This trend is even more pronounced with model width, yielding enhancements of 1.68 and 2.12 in PSNR when transitioning from ResNet10 to ResNet10_x5. In terms of architecture, FEDLEAK performs less effectively with MobileNetV2 and ViT compared to ResNet and DenseNet. The reduced performance with MobileNetV2 is due to its lightweight design, while ViT's unique architecture and information processing pose challenges for FEDLEAK. Nonetheless, we highlight that FEDLEAK achieves significant attack performance, particularly in comparison to ROGS which typically obtains PSNRs around 10. Given that CNNs are more prevalent in practical scenarios and less data-hungry, we decide to stick with ResNet10. We will also see that ResNet10 can attain high accuracy on Lung-Colon Cancer in Section 6.4, eliminating the need for larger and potentially more complex models.

Evaluation over varying local step sizes. Here, we examine the case where the local epoch is set to 1. Lung-Colon Cancer comprises 25000 samples, with 5000 samples reserved for the test set, resulting in each client having a local dataset of 200 samples. Using a batch size of 16[7], each participating client performs $\frac{200}{16} = 12.5$ local steps in a training round. For convenience, we round the local step size at 12. Table 9 presents the performance of four attack methods when clients execute 1, 2, 4, 8, and, 12 local steps using a batch size of 16. A key observation is that FEDLEAK achieves better performance over ROGS. Moreover, as expected, the performance of all attack methods declines as the local step increases. However, commonly used local steps, either one step or a single local epoch, are insufficient to safeguard against FEDLEAK. We will use a local step size of 12 for further analysis.

Evaluation over diverse metrics. Table 10 reports the performance of ROGS and FEDLEAK, scrutinized through three distinct metrics. Due to the high cost of manual ASR evaluation, we only calculate ASRs for data reconstructed at the first client during $2000 \times \{0,1,2,3,4,5\}$-th training rounds. The results in Table 10 demonstrate the superiority of FEDLEAK over ROGS across all three metrics. Specifically, the SSIM for reconstructed data using FEDLEAK reaches an impressive score of 0.68 (68% similarity). Furthermore, human evaluation reveals an ASR of 20.40% for FEDLEAK. These numbers underscore the significant privacy vulnerability in FL.

6.4 Attack in State-of-the-art Defenses

We now employ perturbation-based defenses to construct the most challenging configuration. Although large defense strengths can render GLAs ineffective, they also cause severe

[7]This represents a worst-case scenario, as a small batch size for a fixed number of training samples indicates a higher number of local steps, potentially increasing the server's gradient estimation error. But excessively small batch sizes, e.g., a batch size of 1, may lead to model overfitting on the local dataset.

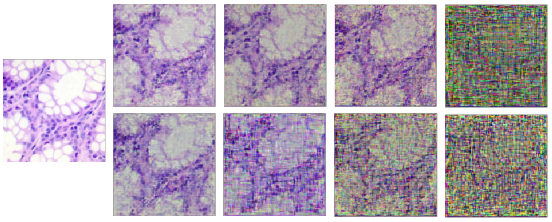


Figure 5: Recovered images of FEDLEAK against combined defenses of Soteria and DP in Lung-Colon-Cancer. The leftmost image is the original image. The top row represents adaptive DP, while the bottom row shows standard DP-SGD. The values of ε from left to right are $\{10^5, 10^4, 10^3, 10^2\}$.

performance degradation. In light of this, we adjust defense strengths to plot the utility-privacy trade-off curve to evaluate the performance of GLAs against these defenses.

Utility-privacy trade-offs. We examine two configurations: local steps of 1 and 12. Figure 4 illustrates the utility-privacy trade-offs (accuracy-PSNR curves) of different defenses in Lung-Colon Cancer. For Soteria and OUTPOST, the pruning ratios used are $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Quantization is assessed with bit levels $\{32, 16, 8, 4, 1\}$, and Guardian's effectiveness is evaluated with β of $\{0.1, 1, 10, 10^2, 10^3\}$. Since other attacks already exhibit poor performance without defense, their results are no longer reported. In Figure 4, all defenses require a trade-off in model accuracy to reduce PSNR. Generally, a PSNR below 10 indicates that the reconstructed data no longer retains meaningful information about the original data [19]. With a local step of 1, few methods achieve full privacy protection, except for Quantization. While Quantization lowers PSNR below 10, it severely compromises model performance, resulting in only 10% accuracy. Increasing the local step from 1 to 12 can significantly enhance the defense effectiveness, allowing us to maintain a model accuracy of around 50% while achieving a PSNR of 10. However, this accuracy remains unsatisfactory. These findings underscore achieving satisfactory model performance while preventing privacy leakage still remains a significant challenge in FL. Notice that increasing the local step can lead to a decline in model performance due to overfitting to the local dataset. For instance, with a local step of 1, the model accuracy is approximately 90%, whereas with a local step of 12, it drops to around 85%. Through this, increasing the local step seems to still be an effective mitigation strategy, as it only reduces model accuracy by 5% while lowering PSNR from about 18 to approximately 13.5.

Combining multiple defense strategies. Beyond individual defenses, an alternative approach is to combine multiple defense strategies. We here evaluate FEDLEAK's performance with the combination of DP and Soteria. In detail, we employ DP-SGD [23] in clients' local training, with $C = 1$, $\delta = 10^{-5}$, and noise variance calculated via $\sigma = \frac{\sqrt{2\log(1/\delta)}}{\varepsilon}$

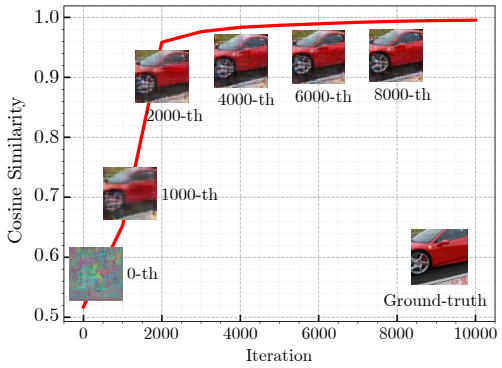


Figure 6: The gradient similarity between reconstructed images and ground-truth ones. We use ImageNet, ResNet10, and a batch size of 64.

[17]. We also implement adaptive DP [21]. ε takes values of $\{10^5, 10^4, 10^3, 10^2\}$. We apply Soteria with a pruning strength parameter of 0.5 to process model updates. Figure 5 visualizes reconstructed examples of Lung-Colon Cancer. Overall, when $\varepsilon \geq 10^3$, FEDLEAK is still able to extract some semantic information. Complete protection is achieved only when $\varepsilon \leq 10^2$. This observation contrasts with common literature [23] where $\varepsilon = 10^2$ often provides negligible privacy guarantees, primarily due to simpler datasets like MNIST or CIFAR-10. In contrast, sophisticated datasets are more sensitive to ε. For example, ResNet18 with DP-SGD achieves only about 10% accuracy at $\varepsilon = 71$ in ImageNet [28], indicating that gradients become heavily noisy. Under such conditions, it is reasonable that FEDLEAK struggles to extract useful private information. Moreover, we find that the defensive capability of adaptive DP is, in fact, weaker than that of standard DP, because of adaptive DP's σ decay mechanism, which reduces gradient noise injection over time.

6.5 Attack Cost

We here evaluate the computational cost of FEDLEAK using an NVIDIA RTX 4090 GPU. Figure 6 shows the convergence curve of FEDLEAK, where we observe that approximately 1000 iterations suffice to recover the most semantic information from the ground-truth images. Beyond 4000 iterations, the reconstructed images show only negligible perceptual changes. For computational efficiency, FEDLEAK requires approximately 20 minutes to complete 10000 iterations. For 10000 iterations, GI and IG require about 5 minutes, while GGL and ROGS take about 11 minutes. GI and IG converge within approximately 1000 iterations, whereas GGL and ROGS stabilize around 2000 iterations. This eliminates the possibility of non-convergence in GLAs. While FEDLEAK indeed incurs higher computational costs compared to existing GLAs, its cost remains within an acceptable range.

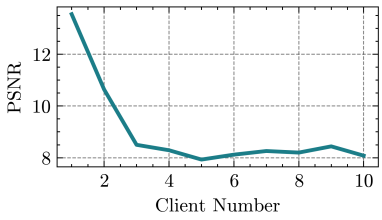


Figure 7: Performance of cryptography-based methods [7] against FEDLEAK with varying participating clients.

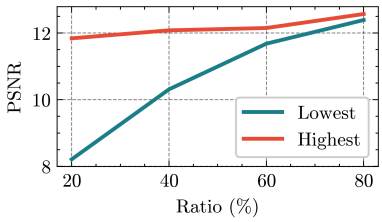


Figure 8: The performance of FEDLEAK over two different gradient pruning strategies.

7 Mitigation and Future Work

Section 6.4 shows that perturbation-based methods are insufficient to fully resist FEDLEAK. We also observe that increasing local steps can significantly degrade the performance of FEDLEAK, primarily due to the amplification of the batch size. Cryptography-based methods [7] exploit this by ensuring that clients only observe the plaintext of aggregated gradients, essentially scaling the batch size by the number of participating clients. Figure 7 evaluates the performance of cryptography-based methods [7] against FEDLEAK with varying participating clients per round. Notably, FEDLEAK's performance diminishes remarkably with just two clients per round and becomes ineffective with three clients (resulting in a batch size of 576). However, cryptography-based methods are often limited in practical deployment due to their complex protocol requirements and computationally intensive operations. For a configuration of 12 local steps, a batch size of 16, and ResNet-10 on Lung-Colon-Cancer, cryptography-based methods incur substantial extra computation time: 702.03 seconds/client and 793.79 seconds/server per communication round [7]. To put this computational cost into perspective, the time required for clients' local training is only 2.6 seconds.

What can we learn from FEDLEAK? One insight in FEDLEAK is the importance of restricting server access to gradient elements with the largest magnitudes[8]. Figure 8 shows FEDLEAK's effectiveness when the server can access varying proportions of the highest and lowest magnitude gradient elements. When the server can only observe low-magnitude components (e.g., the bottom 20%), FEDLEAK achieves very low PSNR, whereas access to high-magnitude components (top 20%) enables reconstruction performance comparable to full gradients. This insight could guide improvements in existing methods.

For instance, cryptography-based methods can encrypt just the top 80% of gradient elements by magnitude, as the remaining ones are less likely to contain private information. This selective encryption could reduce cryptographic overhead by approximately 20%. Another potential solution is for clients to inject noise proportionally to the magnitude of gradient elements so as to obfuscate high-magnitude elements. Consider noise vector $\mathcal{M}$. To counterbalance the impact of this noise on model performance, the client could send $-\mathcal{M}$ to another client, which adds it to its own model updates. In this way, the noise vector would be canceled out in aggregation process. However, this method requires that clients do not disclose $\mathcal{M}$ to the server. Furthermore, we highlight that all defense methods inherently entail trade-offs among model utility, privacy preservation, computational efficiency, and threat model assumptions. Future research should investigate these trade-offs to develop more effective defense strategies.

[8] Intuitively, if FEDLEAK converges, the elements with the greatest magnitude in g and g' will be the same. The gradient elements in g' play an important role for FEDLEAK.

References

[1] Mohammed Adnan, Shivam Kalra, Jesse C Cresswell, et al. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1):1953, 2022.

[2] Abeer Aljohani and Rua Y Aburasain. A hybrid framework for glaucoma detection through federated machine learning and deep learning models. *BMC Medical Informatics and Decision Making*, 24(1):115, 2024.

[3] Yoshinori Aono, Takuya Hayashi, Lihua Wang, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE TIFS*, 13(5):1333–1345, 2017.

[4] Mislav Balunovic, Dimitar Iliev Dimitrov, Robin Staab, and Martin T. Vechev. Bayesian framework for gradient leakage. In *ICLR*, 2022.

[5] Brian R. Bartoldson, James Diffenderfer, Konstantinos Parasyris, and Bhavya Kailkhura. Adversarial robustness limits via scaling-law and human-alignment studies. In *ICML*, 2024.

[6] Franziska Boenisch, Adam Dziedzic, Roei Schuster, et al. When the curious abandon honesty: Federated learning is not private. In *IEEE EuroS&P*, pages 175–199. IEEE, 2023.

[7] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, et al. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191, 2017.

[8] Stephen Boyd. Convex optimization. *Cambridge UP*, 2004.

[9] Nicholas Carlini, Florian Tramer, Eric Wallace, et al. Extracting training data from large language models. In *USENIX Security*, pages 2633–2650, 2021.

[10] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, et al. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

[11] Nicolas Carlini, Jamie Hayes, Milad Nasr, et al. Extracting training data from diffusion models. In *USENIX Security*, pages 5253–5270, 2023.

[12] Hong-Min Chu, Jonas Geiping, Liam H. Fowl, et al. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *ICLR*, 2023.

[13] Trung Dang, Om Thakkar, Swaroop Ramaswamy, et al. Revealing and protecting labels in distributed training. In *NeurIPS*, volume 34, pages 1727–1738, 2021.

[14] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[16] Jiacheng Du, Jiahui Hu, Zhibo Wang, et al. Sok: Gradient leakage in federated learning. *CoRR*, abs/2404.05403, 2024.

[17] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[18] Mingyuan Fan, Cen Chen, Chengyu Wang, and Jun Huang. Exploiting pre-trained models and low-frequency preference for cost-effective transfer-based attack. *ACM Trans. Knowl. Discov. Data*, jul 2024. ISSN 1556-4681.

[19] Mingyuan Fan, Yang Liu, Cen Chen, et al. Guardian: Guarding against gradient leakage with provable defense for federated learning. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 190–198, 2024.

[20] Mingyuan Fan, Chengyu Wang, Cen Chen, Yang Liu, and Jun Huang. On the trustworthiness landscape of state-of-the-art generative models: A survey and outlook. *International Journal of Computer Vision*, 2025.

[21] Jie Fu, Zhili Chen, and Xiao Han. Adap dp-fl: Differentially private federated learning with adaptive noise. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 656–663. IEEE, 2022.

[22] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? In *NeurIPS*, volume 33, pages 16937–16947, 2020.

[23] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[24] Hao Guan, Pew-Thian Yap, Andrea Bozoki, and Mingxia Liu. Federated learning for medical image analysis: A survey. *Pattern Recognit.*, 151:110424, 2024.

[25] Jiahui Hu, Jiacheng Du, Zhibo Wang, et al. Does differential privacy really protect federated learning from gradient leakage attacks? *IEEE Transactions on Mobile Computing*, 2024.

[26] Yangsibo Huang, Samyak Gupta, Zhao Song, et al. Evaluating gradient inversion attacks and defenses in federated learning. In *NeurIPS*, pages 7232–7241, 2021.

[27] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, et al. Gradient inversion with generative image prior. In *NeurIPS*, volume 34, pages 29898–29908, 2021.

[28] Alexey Kurakin, Steve Chien, Shuang Song, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. Toward training at imagenet scale with differential privacy. *CoRR*, abs/2201.12328, 2022.

[29] Edward H Lee, Michelle Han, Jason Wright, et al. An international study presenting a federated learning ai platform for pediatric brain tumors. *Nature communications*, 15(1):7615, 2024.

[30] Xiang Li, Kaixuan Huang, Wenhao Yang, et al. On the convergence of fedavg on non-iid data. In *ICLR*, 2020.

[31] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, et al. Fedbn: Federated learning on non-iid features via local batch normalization. In *ICLR*, 2021.

[32] Zhuohang Li, Jiaxin Zhang, Luyang Liu, and Jian Liu. Auditing privacy defenses in federated learning via generative gradient leakage. In *CVPR*, pages 10132–10142, 2022.

[33] Yang Liu, Mingyuan Fan, Cen Chen, et al. Backdoor defense with machine unlearning. In *INFOCOM*, page 280–289. IEEE, 2022.

[34] Yuan Liu, Jinzao Huang, Jyh-Cheng Chen, et al. Predicting treatment response in multicenter non-small cell lung cancer patients based on federated learning. *BMC cancer*, 24(1):688, 2024.

[35] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[36] Alexander Ly, Maarten Marsman, Josine Verhagen, et al. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

[37] Kailang Ma, Yu Sun, Jian Cui, et al. Instance-wise batch label restoration via gradients in federated learning. In *The 11th International Conference on Learning Representations*, 2023.

[38] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[39] Brendan McMahan, Eider Moore, Daniel Ramage, et al. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[40] Rahul Mishra and Hari Gupta. Transforming large-size to lightweight deep neural networks for iot applications. *ACM Computing Surveys*, 55(11):1–35, 2023.

[41] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S&P*, pages 19–38. IEEE, 2017.

[42] Dinh C. Nguyen, Ming Ding, Pubudu N. Pathirana, et al. Federated learning for internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutorials*, 23(3): 1622–1658, 2021.

[43] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2429–2443, 2022.

[44] Samira Pouyanfar, Saad Sadiq, Yilin Yan, et al. A survey on deep learning: Algorithms, techniques, and applications. *ACM computing surveys*, 51(5):1–36, 2018.

[45] Micah J Sheller, Brandon Edwards, G Anthony Reina, Jason Martin, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 10(1):12598, 2020.

[46] Jingwei Sun, Ang Li, Binghui Wang, et al. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *CVPR*, pages 9311–9319, 2021.

[47] Fei Wang, Ethan Hugh, and Baochun Li. More than enough is too much: Adaptive defenses against gradient leakage in production federated learning. *IEEE/ACM Transactions on Networking*, 32(4):3061–3075, 2024. doi: 10.1109/TNET.2024.3377655.

[48] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Protect privacy from gradient leakage attack in federated learning. In *INFOCOM*, pages 580–589. IEEE, 2022.

[49] Zhibo Wang, Zhiwei Chang, Jiahui Hu, et al. Breaking secure aggregation: Label leakage from aggregated gradients in federated learning. *arXiv preprint arXiv:2406.15731*, 2024.

[50] Wenqi Wei, Ling Liu, Margaret Loper, et al. A framework for evaluating gradient leakage attacks in federated learning. *CoRR*, abs/2004.10397, 2020.

[51] Yuxin Wen, Jonas Geiping, Liam Fowl, et al. Fishing for user data in large-batch federated learning via gradient magnification. In *ICML*, volume 162, pages 23668–23684. PMLR, 2022.

[52] Hongxu Yin, Arun Mallya, Arash Vahdat, et al. See through gradients: Image batch recovery via gradinversion. In *CVPR*, pages 16337–16346, 2021.

[53] Kai Yue, Richeng Jin, Chau-Wai Wong, et al. Gradient obfuscation gives a false sense of security in federated learning. In *USENIX Security*, pages 6381–6398, 2023.

[54] Chengliang Zhang, Suyi Li, Junzhe Xia, et al. BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In *2020 USENIX annual technical conference*, pages 493–506, 2020.

[55] Fan Zhang, Daniel Kreuter, Yichen Chen, et al. Recent methodological advances in federated learning for healthcare. *Patterns*, 5(6):101006, 2024.

[56] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[57] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, et al. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *IEEE S&P*, pages 1287–1305. IEEE, 2024.

[58] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *NeurIPS*, volume 32, 2019.

Table 11: The gradient distance (GD) and PSNR using different attack strategies in MNIST and CIFAR-10.

Attack Strategy	MNIST		CIFAR-10	
	GD	PSNR	GD	PSNR
BayesAttack	0.19	10.99	0.21	11.08
IG	0.25	8.70	0.16	10.32
GI	0.17	12.01	0.19	12.03
BayesAttack + Multiple Start	0.16	11.11	0.20	11.10
IG + Multiple Start	0.24	9.01	0.16	10.33
GI + Multiple Start	0.16	12.23	0.18	12.05
BayesAttack + Cosine Annealing	0.15	11.04	0.19	11.32
IG + Cosine Annealing	0.23	9.22	0.14	10.65
GI + Cosine Annealing	0.14	12.05	0.16	12.57

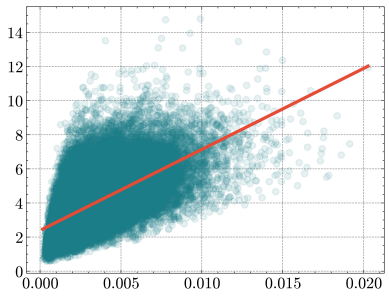


Figure 9: The x-axis represents the absolute value of the gradient elements ($|g'[i]|$), while the y-axis represents the sensitivity of gradient elements to input changes ($\sum_j |\frac{\partial g'[i]}{\partial x'[j]}|$).

A Supplementary Experiment for Section 3

We employ three attack methods (BayesAttack [4], IG [22], GI [52]) on a three-layer fully connected network (Section 3), with results in Table 11. Note that IG uses cosine distance while others use L_2 distance. The PSNR values of the reconstructed data are notably low in both MNIST and CIFAR-10 datasets, as evidenced by PSNRs around 10.

We also incorporate multiple restart [38] and cosine annealing learning rate [35] for these attacks. Multiple restarts mitigate saddle points and local minima through repeated attacks with different seeds to avoid poor initializations.

Cosine annealing involves a sharp increase in the step size after a predetermined number of iterations (1000 in our experiments) to escape saddle points and local minima. As reported in Table 11, although these strategies can somewhat improve attack performance, the quality of the reconstructed data remains significantly low. Thus, the recovery of noisy data is unlikely to be mainly attributed to saddle points and local minima.

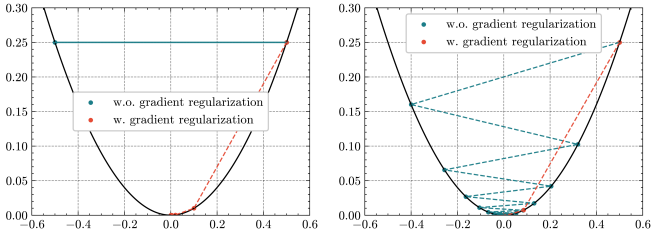


Figure 10: The search paths of the gradient descent algorithm with and without gradient regularization for minimizing $y = x^2$. The left image is with a step size of 1. The right image is with a step size of 0.9.

B Gradient Estimation Error

Here, we analyze the scenario where $\hat{g} \neq \nabla_w \mathcal{L}(F(x,w),y)$. We denote $\nabla_w \mathcal{L}(F(x,w),y)$ as g. Let us suppose $\varepsilon = \hat{g} - g$, with $\hat{g}$ corresponding to the gradients generated by x^*, i.e., $\hat{g} = \nabla_w \mathcal{L}(F(x^*,w),y)$. According to the mean value theorem, we have $\hat{g} - g = \nabla^2 \mathcal{L}(F(z,w),y)\ (x^* - x)$, where z is between x and x^*. Consequently, according to basic theorem in algebra [8], if $\nabla^2 \mathcal{L}(F(z,w),y)$ is positive definite, we obtain the following inequality:

$$||\hat{g} - g|| = ||\nabla^2 \mathcal{L}(F(z,w),y)\ (x^* - x)|| \geq \sigma_{\min}\ ||x^* - x||,$$

where $\sigma_{\min}$ denotes the smallest eigenvalue of $\nabla^2 \mathcal{L}(F(z,w),y)$. This indicates that when the estimation error is ε, the distance between ground-truth data associated with $\hat{g}$ and g is bounded by $\frac{||\varepsilon||}{||\nabla^2 f(z)||}$. Notice that in practice, if the model fits the dataset well, the Hessian matrix ($\nabla^2 \mathcal{L}(F(z,w),y)$) should indeed be positive definite. Moreover, a similar analytical framework can be applied to the estimation errors associated with y and we will not elaborate on that further here.

C Empirical Validation about Section 4.2

Here, we empirically demonstrate that gradient elements with larger magnitudes are more sensitive to changes in input data. We employ ResNet10 and CIFAR-10. Specifically, we input 100 samples into the model and calculate the gradients of the final linear layer ($g'[i]$). We then calculate the absolute sum of each gradient element with respect to the input data ($\sum_j |\frac{\partial g'[i]}{\partial x'[j]}|$) and visualize the results in Figure 9. As can be seen, there is a clear trend: as the magnitude of the gradient elements increases, so does their sensitivity to input changes. The Pearson correlation coefficient is 0.7372 ± 0.0566. In statistics, correlation coefficient above 0.7 is generally considered significant. Thus, gradient elements with larger magnitudes indeed respond more sensitively to variations in the input data.

D Example for Gradient Regularization

To clarify the role of gradient regularization (Equation 14), we illustrate its effect on minimizing $y = x^2$ using gradient descent with and without regularization. Starting at $x = 0.5$, as illustrated on the left side of Figure 10, a fixed step size of 1 without gradient regularization causes oscillation between 0.5 and -0.5. Even with a reduced learning rate 0.9, oscillations persist, slowly converging to $x = 0$.

Equation 14 combines gradients at the current point x and a subsequent point along the descent path. Intuitively, if the gradient at the forward point opposes that at the current point, it indicates that the step size is too large. Then, the gradient at the forward point will shrink the gradient at the current point x, which indirectly reduces the step size. This behavior is illustrated in Figure 10, where we set λ' to 0.3. The gradient at $x = 0.5$ is 1, while the gradient at the next point $x = -0.5$ is -1. Consequently, the resulting mixed gradient is $(1 - \lambda') \times 1 + \lambda' \times (-1) = 0.4$. This demonstrates how gradient regularization facilitates the optimization process.

E Supplementary Investigation Result

Upon review of existing research related to GLAs, we identify eight key factors that significantly impact the performance of these attacks and summarize these factors in Table 12. These factors are either explicitly mentioned or shown to have a substantial impact on the effectiveness of GLAs. To collect common values for eight factors in production environments, we reviewed extensive FL literature on FL and open-source FL libraries. The attack stage is server-determined, while auxiliary information is largely scenario-driven, such as whether the specified scenario can collect substantial data similar to that of the clients. We focus on the remaining seven factors, including batch size, model, dataset, initialization method, auxiliary information, local step, and defense.

- **Batch size.** Some survey papers [39] indicate that the common batch size ranges from 16 to 64. Our examination of application cases published in journals such as Nature, Science, and BMC over the past several years reveals that many studies use vague expressions like "specific/common parameters" without providing precise values. A few papers specify batch sizes of 16 [34], 32 [1, 2], 60[9], and 64 [29]. We conjecture that these studies predominantly adopt the default recommended values from existing frameworks. Introductory examples in common industrial libraries also typically use batch sizes between 20 and 64, as summarized in Table 13. Therefore, we consider the range of 16 to 64 as standard practice.

- **Model.** Guan et al. [24] compiled the studies utilizing FL in the medical domain. Among the 47 papers they reviewed, 35 employed CNN architectures, primarily ResNet, DenseNet, and MobileNet, while 4 utilized transformer models. This highlights that CNNs remain the dominant architecture in practical applications. This is likely due to their higher computational efficiency and lower data requirements compared to transformer models [15], making them better suited for data-scarce medical environments.

- **Dataset.** As noted in most literature [24, 29, 34, 55], real-world datasets typically have higher resolutions compared to commonly used benchmark datasets, which are often limited to a resolution of 32×32.

- **Initialization method.** Zhang et al. [55] summarized the weight initialization approaches of existing FL application papers in the medical domain. Among the 89 papers they reviewed, 68 explicitly discussed approaches for weight initialization. Among 89 studies, 45/89 used random initialization, 16/89 pre-determined parameters, and 7/89 pre-trained weights. Pre-determined parameters can vary significantly across tasks. As such, we focus on random initialization and pre-trained weights.

- **Local step.** Most papers vaguely describe local steps. Common values include 1 step or 1 local epoch [1, 14, 45].

- **Defense.** Zhang et al. [55] reviewed 89 FL application papers and found that most did not actively implement defense methods. Specifically, six papers applied DP, five of which were based on Gaussian mechanism. It seems that practitioners in real-world scenarios generally have strong confidence in FL's privacy protection capabilities. Although the use of privacy-enhancing methods is not yet widespread in practice, we still evaluate the effectiveness of FEDLEAK against defense methods.

F Label Inference, Sensitivity Analysis, and Ablation Study

We first discuss label inference. We then study the impact of different hyperparameters on the performance of FEDLEAK. By default, we use a batch size of 32 and CIFAR-10.

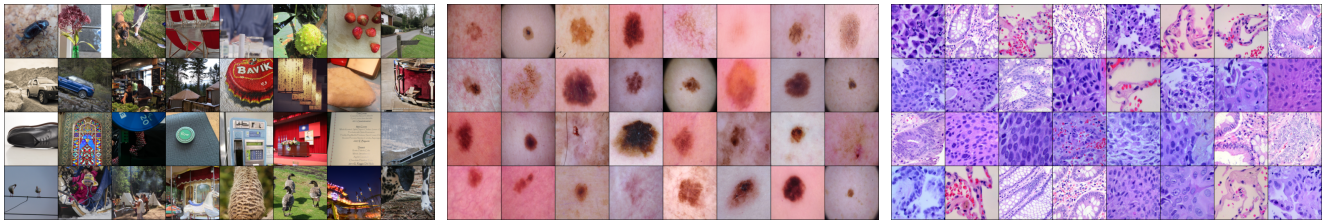
The impact of label inference. Table 14 reports the accuracy of label inference and the FEDLEAK's performance with and without label inference across different batch sizes, over 1280 ImageNet samples. We see label inference generally achieves high accuracy and enhances reconstruction quality.

The impact of matching ratio *R*. We here incrementally increase the value of R from 15 to 100 to observe the performance change of FEDLEAK, as illustrated in Table 15. Looking at the results we see a performance trend that initially rises and peaks at $R = 50$, followed by a subsequent decline. This trend can be intuitively understood as a result of balancing the number of matched gradient elements. Matching too few gradient elements might lead to the omission of

[9] https://github.com/gkaissis/PriMIA/blob/master/configs/torch/pneumonia-conv.ini

Table 12: A summary of critical factors influencing the performance of GLAs.

Factor	Literature	Explanation
Batch Size	[16, 25, 26, 50, 52, 53, 58]	A larger batch size introduces more optimization variables, which complicates the data reconstruction process.
Model	[19, 22, 25]	Intuitively, a larger model should impose more constraints on the gradient matching problem, potentially enhancing the reconstruction quality. However, this increased complexity also leads to heightened nonlinearity of the model. Compared to linear problems, non-linear problems are notoriously more difficult to solve. Early GLAs present poor performance on large models, likely due to their inability to effectively address the gradient matching problem associated with large models.
Dataset	[16, 22, 25, 27, 32, 50, 53]	Higher-resolution images suggest more optimization variables, thereby exacerbating reconstruction difficulty.
Attack Stage	[4, 16]	GLAs tend to yield better attack results during the early stages of training, which can be attributed to the model's higher sensitivity at that time. This is elaborated in Section 1.
Initialization Method	[22, 47]	Different weight initialization methods can affect the model's sensitivity to data, thereby influencing the reconstruction difficulty. For instance, wide uniform initialization can amplify the weight intensity, causing minor data variations to lead to significant changes in the model's output. This increased sensitivity can facilitate data reconstruction.
Auxiliary Information	[16, 26, 27, 47]	Apparently, auxiliary information gives the server more advantages in solving the gradient matching problem.
Local Step	[16, 47, 50]	When the batch size is fixed, an increase in local steps leads to uploaded model updates being derived from more data, which complicates the data reconstruction. Besides, an increase in local steps also can introduce greater estimation errors on the server side, adding another layer of difficulty to the data reconstruction.



(a) ImageNet (b) HAM10000 (c) Lung-Colon Cancer

Figure 11: The ground-truth batch of Figure 3.

Table 13: The default batch sizes of introductory examples in various open-source libraries.

Library	Batch Size
NVIDIA FLARE	4-32
FLOWER	32-64
Substra	32
PySyft	64
OpenFL	32
TensorFlow Federated	20

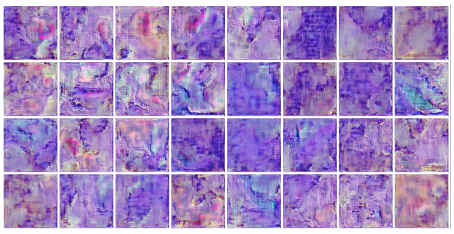


Figure 12: ROGS's recovered batch in Lung-Colon Cancer.

Table 14: Label inference accuracy and performance metrics across different batch sizes in ImageNet.

Batch Size	Label Inference Accuracy	PSNR (with Label Inference)	PSNR (w.o. Label Inference)
32	100%	19.07	17.94
64	96.88%	19.01	17.50
128	92.19%	18.40	16.45

Table 15: The impact of R.

R	15%	30%	50%	65%	80%	100%
PSNR	18.85	20.34	21.33	20.76	19.80	15.06

Table 16: The impact of λ'.

λ'	0.1	0.3	0.5	0.7	0.9
PSNR	18.85	20.34	21.33	20.76	19.80

many useful elements, whereas matching too many can lead to an overly small μ value.

The impact of blend factor λ'. A higher λ' heightens emphasis on gradient regularization, thereby reducing the value of L. Nonetheless, an excessively high λ' might disproportionately prioritize optimizing the gradient regularization term, potentially neglecting the gradient matching. As anticipated, Table 16 illustrates this rising and then falling trend.

The impact of α and β. Table 17 and Table 18 present

Table 17: The impact of α.

α	0	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
PSNR	21.05	21.33	21.99	21.01	17.79	14.31

Table 18: The impact of β.

β	0	0.0001	0.001	0.01	0.1
PSNR	20.84	21.33	21.80	21.17	20.20

the impact of varying α and β on FEDLEAK's performance. Both α and β exhibit a rising and subsequently falling pattern regarding attack performance. Specifically, an overly large α leads to overly smooth constructions, where all pixel values converge to the same number. Likewise, a high β causes the recovered images to have activation values close to zero, thus reconstructing most pixel values to zero.

Table 19: The impact of loss function.

Loss Function	w.o. cos	w.o. L_1	cos & L_1
PSNR	20.04	19.86	21.33

Ablation study on loss function. We further assess the performance of FEDLEAK when removing either the L_1 distance or cosine distance from Equation (15). As reported in Table 19, omitting either distance measure results in performance degradation, showing the necessity of both components in FEDLEAK's effectiveness.

G Supplementary Theory

Theorem 3. *Removing the minimum values from a set $S = \{a_1, \cdots, a_n\}$ and then calculating the average of the remaining values will result in an increase in the average.*

Proof. Without loss of generality, let a_n be the smallest element in the set S. The average of the original S is given by $\text{Average}(S) = \frac{1}{n}\sum_{i=1}^{n} a_i$. After removing the smallest element a_n, the new set is defined as $S' = S \setminus \{a_n\}$. The average of the new set S' can be expressed as $\text{Average}(S') = \frac{1}{n-1}(\sum_{i=1}^{n} a_i - a_n)$. We demonstrate that $\text{Average}(S') \geq \text{Average}(S)$. This inequality can be rewritten as $\frac{1}{n-1}(\sum_{i=1}^{n} a_i - a_n) \geq \frac{1}{n}\sum_{i=1}^{n} a_i$. Cross-multiplying yields:

$$n(\sum_{i=1}^{n} a_i - a_n) \geq (n-1)\sum_{i=1}^{n} a_i.$$

Expanding and rearranging gives:

$$n\sum_{i=1}^{n} a_i - na_n \geq n\sum_{i=1}^{n} a_i - \sum_{i=1}^{n} a_i.$$

Simplifying leads to $na_n \leq \sum_{i=1}^{n} a_i$. Since a_n is the minimum value in S, it follows that $na_n \leq \sum_{i=1}^{n} a_i$ holds true. We can iteratively remove subsequent minimum elements and apply the same reasoning. Each removal of the smallest element results in an increased average of the resulting set, thereby confirming the validity of Theorem 2. □