USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Refiner: Data Refining against Gradient Leakage Attacks in Federated Learning

Mingyuan Fan, *East China Normal University;* Cen Chen, *East China Normal University and The State Key Laboratory of Blockchain and Data Security, Zhejiang University;* Chengyu Wang, *Alibaba Group;* Xiaodan Li, *East China Normal University and Alibaba Group;* Wenmeng Zhou, *Alibaba Group*

https://www.usenix.org/conference/usenixsecurity25/presentation/fan-refiner

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

Refiner: Data Refining against Gradient Leakage Attacks in Federated Learning

Mingyuan Fan[1] Cen Chen[1,2*] Chengyu Wang[3] Xiaodan Li[1,3] Wenmeng Zhou[3]
[1]*East China Normal University*
[2]*The State Key Laboratory of Blockchain and Data Security, Zhejiang University*
[3]*Alibaba Group*
mingyuan_fmy@stu.ecnu.edu.cn *cenchen@dase.ecnu.edu.cn*
{ chengyu.wcy, fiona.lxd, wenmeng.zwm}@alibaba-inc.com

Abstract

Recent works highlight the vulnerability of Federated Learning (FL) systems to *gradient leakage attacks*, where attackers reconstruct clients' data from shared gradients, undermining FL's privacy guarantees. However, existing defenses show limited resilience against sophisticated attacks. This paper introduces a novel defensive paradigm that departs from conventional gradient perturbation approaches and instead focuses on the construction of robust data. Our theoretical analysis indicates such data, which exhibits low semantic similarity to the clients' raw data while maintaining good gradient alignment to clients' raw data, is able to effectively obfuscate attackers and yet maintain model performance. We refer to such data as robust data, and to generate it, we design *Refiner* that jointly optimizes two metrics for privacy protection and performance maintenance. The utility metric promotes the gradient consistency of key parameters between robust data and clients' data, while the privacy metric guides the generation of robust data towards enlarging the semantic gap with clients' data. Extensive empirical evaluations on multiple benchmark datasets demonstrate the superior performance of *Refiner* at defending against state-of-the-art attacks.

1 Introduction

Federated Learning (FL) enables privacy-preserving DNN training by having clients compute gradients locally and share them with a central server for updating the global model [12], as depicted in Figure 1. However, recent *Gradient Leakage Attacks* (GLAs) [56, 64] have shown that clients' data can be reconstructed by aligning uploaded gradients with those generated from dummy data. While some works [11, 49] argued that such attacks are limited to simple scenarios, recent advances in GLAs [22, 56] have shown these threats are increasingly effective in practical FL settings, especially when the server is allowed to manipulate the global model [37, 63]. Moreoverr, the uploaded gradients can also be utilized to infer dataset properties and membership information [7, 30]. This highlights the need for effective defenses in FL to safeguard the confidentiality of sensitive client information.

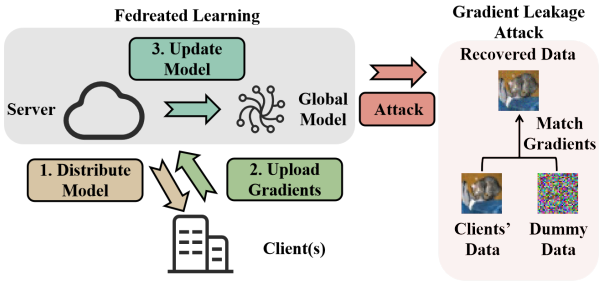


Figure 1: A sketch map of FL and GLAs.

Two primary types of methods have been proposed to safeguard privacy in FL: encryption-based methods [4, 14] and perturbation-based methods [1, 43, 50, 61, 64]. In encryption-based methods [14], clients transmit encrypted gradients, but these encrypted gradients are ultimately decrypted into plaintext for aggregation purposes [4, 14], leaving them vulnerable to GLAs [64]. Moreover, encryption-based methods incur significant computational/communication overheads that dominate the training process, significantly affecting their practicability [50, 58].

Perturbation-based methods impose subtle perturbations into ground-truth gradients, which helps to obscure the adversary's (the server's) ability to recover accurate data from the gradients. Notable defenses include differential privacy [1] and gradient pruning [2], which perturb gradients by injecting random noise or discarding part of gradients. State-of-the-art defenses [43, 50] estimate the privacy information contained in each gradient element for gradient pruning. Despite demonstrating promising privacy protection ability in certain FL scenarios, these methods inevitably compromise the model's utility due to gradient perturbations. Consequently, developing methods that can achieve a better trade-off between utility and privacy remains a significant research topic.

*Corresponding author.

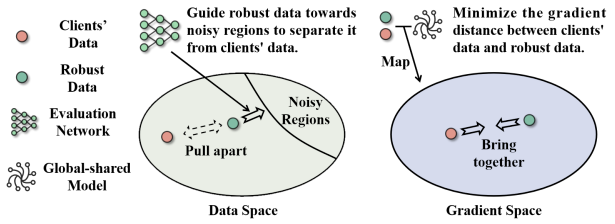


Figure 2: *Refiner* enhances privacy protection by employing an evaluation network that directs robust data toward noisy regions, increasing the semantic distance from clients' data. To maintain utility, *Refiner* utilizes the global-shared model to map clients' data and robust data onto gradient space and minimizes the gradient distance between them.

Our contributions. This paper explores a novel defensive paradigm that diverges from conventional gradient perturbation approaches, centering instead on the construction of robust data. Our inspiration stems from an intuitive yet compelling idea: if we can generate robust data that exhibit substantial semantic dissimilarity compared to clients' data while maintaining utility, the gradients associated with these data are likely to have the potential to effectively obfuscate attackers while imposing only a small degradation in the model's performance. To bring this idea to fruition, we present *Refiner* which crafts robust data by jointly optimizing two key metrics for privacy protection and performance maintenance. As illustrated in Figure 2, the privacy metric, *i.e.*, the evaluation network, encourages robust data to deviate semantically from clients' data, while the utility metric aligns the gradients of model parameters associated with clients' data and robust data. In practice, to protect privacy, clients refrain from uploading gradients directly derived from their raw data. Instead, they upload gradients associated with robust data, thereby maintaining the confidentiality of clients' data.

- *Utility metric.* The utility metric is defined as the weighted distance between the gradients of model parameters with respect to robust data and clients' data. By minimizing the gradient distance, the utility of the robust data approaches that of the clients' data. The weighting scheme prioritizes the preservation of gradients associated with important parameters, thus better maintaining the utility of the robust data. This weighting mechanism consists of element-wise weight and layer-wise weight. The element-wise weight is determined based on empirical and theoretical observations, considering the values of both gradients and parameters to identify critical parameters. Additionally, the layer-wise weight exploits the significance of earlier layers by allocating more attention to them during the learning process.
- *Privacy metric.* While norm-based distances between robust data and original data are commonly used to measure privacy [43], their effectiveness is limited due to the significant divergence with human-vision-system difference metrics [46]. To address this limitation, *Refiner* instead employs an evaluation network to accurately estimate the human perception distance between robust data and original data. The *perception distance* serves as the privacy metric, reflecting the overall amount of disclosed privacy information associated with clients' data. By leveraging the human perception distance, *Refiner* provides a more accurate and effective evaluation of the privacy information contained in robust data.

The main body of this paper focuses on the threat of data reconstruction due to its severity, while discussions regarding *Refiner*'s performance on text data and other types of privacy leakage are included in Appendix G.

2 Background & Related Work

2.1 Federated Learning

This paper is anchored in a typical FL scenario, where a server collaborates with m clients to train a model $F_\theta(\cdot)$ parameterized by θ, utilizing loss function $\mathcal{L}(\cdot,\cdot)$. The i-th client's local dataset is denoted by D_i. The server and clients interact with each other over N communication rounds by executing the two steps:

- **Model Distribution and Local Training.** The server samples a subset of K clients from the entire client pool and broadcasts the global model $F_\theta(\cdot)$ to selected clients. The k-th selected client ($k = 1, \cdots, K$) computes gradients $g_k = \nabla_\theta \mathcal{L}(F_\theta(x), y)$ using raw data x, y sampled from its local dataset.
- **Global Aggregation.** The server averages the gradients received from the selected clients to update the global model: $\theta = \theta - \frac{\eta}{K}\sum_{i=1}^{K} g_i$ where η is the learning rate.

In the literature [40], clients may update the local model over multiple steps and upload the model parameters. We employ a default local step of 1, where clients directly upload the gradients. This decision is motivated by the fact that the server typically derives gradients by comparing the global model parameters with the uploaded local model parameters. Multiple local steps could potentially cause a discrepancy between the derived gradients and the ground-truth gradients, thereby reducing attack effectiveness [61]. Thus, we adopt the setting with the most powerful attack scenario to better examine defense performance.

2.2 Threat Model

We consider a prevalent attack scenario in which the adversary is an *honest-but-curious* server [50].

- **Adversary's Objective.** The primary goal of the server is to recover private information from the gradients shared by clients, endeavoring to capture a wealth of semantic details involved in clients' raw data.
- **Adversary's Capabilities.** The server will faithfully execute FL protocols. The server can access the model's

weights and the uploaded gradients but is unable to manipulate the model's architecture or weights to better suit their attacks.

- **Adversary's Knowledge.** The server can leverage public datasets to augment their attacks and possesses sufficient computational resources. These public datasets could be similar to clients' data to serve as strong priors for enhancing their attacks, but are not duplicates of the clients' datasets.

In addition, we also study a malicious active server, where the adversary can manipulate model parameters during training. Appendix H evaluates *Refiner*'s performance under the stronger malicious threat model. Notice that, the honest-but-curious server remains the primary focus of this paper due to its practical prevalence in real-world FL deployments. We suppose that clients are aware of the privacy risks, and, as a remedy, clients proactively adopt defenses.

- **Defender's Objective.** Clients' objective is to prevent the server from reconstructing their data by using the gradients they upload. Moreover, clients wish to maintain model performance.
- **Defender's Capabilities.** Clients possess the autonomy to adjust their ground-truth gradients before upload, utilizing moderate computational resources to enable their defense strategies.
- **Defender's Knowledge.** Defender's knowledge includes their awareness of the server's reliability and possible countermeasures. We assume that clients know potential defense methods and can access to necessary resources from trusted third parties for implementing these defenses, such as evaluation network (See Section 3.3).

To simplify the notations, subscripts, and superscripts representing specific clients are omitted throughout the paper.

2.3 Gradient Leakage Attack

Given randomly initialized dummy data x' with dummy label y' and uploaded gradients g, the seminal GLA Zhu et al. [64] frames reconstruction as a gradient matching problem:

$$x', y' = \arg\min_{x', y'} ||\nabla_\theta \mathcal{L}(F_\theta(x'), y') - g||_2. \quad (1)$$

The follow-up works [48, 62] showed how to infer the ground-truth label y from the gradients of the fully-connected layer, reducing the complexity of Equation 1. However, several studies [16, 52, 56] empirically demonstrated that bad initialization causes convergence towards local optima, making recovered data noisy. In response, Geiping et al. [16] and Jeon et al. [22] explored various regularization techniques and heuristic tricks, including total variation, restart, and improved initialization, aiming to eliminate invalid reconstructions. Recently, since the output of generative models is not noisy, Li et al. [24] and Yue et al. [57] trained a generative network to recover clients' data. The potential assumption is the availability of an abundance of data that is similar to the clients' data. They optimized latent vectors of the generative network to produce images whose gradients are similar to uploaded ones.

2.4 Gradient Leakage Defense

While cryptographic methods have emerged as potential solutions in defending GLAs, the substantial overhead restricts their applicability [24, 50, 64] and motivates more lightweight perturbation-based defenses. One such perturbation-based method [3, 53, 64] is differential privacy (DP), which adds random noises to ground-truth gradients to generate perturbed gradients. Moreover, some works [57, 64] explored gradient pruning [2] or gradient quantization (GQ) [40] to obfuscate the server. Soteria [43] employs a theoretically-derived metric to evaluate the privacy information contained in each gradient element and selectively prunes gradients in fully-connected layers. However, Balunović et al. [3], Wang et al. [50] identified that Soteria could be compromised by muting gradients in fully-connected layers during the reconstruction process (Equation 1). In response, they introduced a gradient mixing mechanism to solve the problem.

2.5 Image Anonymization

Robust data is defined to be differently semantic from clients' data while maintaining comparable utility. Image anonymization strips sensitive content from images, thereby offering a potential approach for generating robust data. Traditional anonymization techniques include blurring, masking [34], and k-means clustering [27, 33]. Recent advancements [44, 55] introduce the use of generative models for the synthetic replacement of sensitive content. However, image anonymization has been demonstrated to be vulnerable to malicious attacks [15, 32, 35, 59], causing potential privacy leakage. Moreover, these methods are primarily developed with privacy protection, instead of finding a satisfactory trade-off between utility and privacy [21]. As a result, directly employing image anonymization in *Refiner* is unsuitable.

3 Our Approach: *Refiner*

3.1 Problem Formulation

Refiner aims to construct robust data x^* that preserves high utility while minimizing privacy information leakage from raw client data x. Then, the gradients generated by x^* serve as uploaded gradients. *Refiner* formulates the following optimization problem to craft robust data:

$$x^* = \arg\min_{x^*} UM(x^*, x) - \beta \cdot PM(x^*, x), \quad (2)$$

where UM measures the utility gap between x^* and x (lower values indicate better utility), and PM quantifies the reduction

in privacy information leakage relative to x (higher values indicate stronger privacy). β is a balance factor to regulate the trade-off between utility and privacy. A higher value of β intensifies the prioritization of privacy. Section 3.2 and Section 3.3 will elaborate on UM and PM, respectively.

3.2 Utility Metric

3.2.1 Motivation

The utility of data is defined as the extent to which it contributes to reducing the model loss. DNNs utilize gradients to update model parameters to minimize loss, making it straightforward to exploit the mean square error (MSE) between gradients generated by x and x^* as the utility metric, *i.e.*, $UM(x^*,x) = ||\nabla_\theta \mathcal{L}(F_\theta(x^*),y) - \nabla_\theta \mathcal{L}(F_\theta(x),y)||_2^2$. However, employing the MSE in its vanilla form treats every gradient element without distinction, neglecting that some parameters may exert a more pronounced influence.

To address this, we customize a weighted MSE to more effectively evaluate the utility contained in x^*. The overall idea of weighted MSE is to endow the gradients of important parameters with higher weights, so as to pay more attention to aligning the gradients of important parameters associated with x and x^*. While existing parameter importance estimation techniques exist, they often incur high computational overhead (See Appendix A). We instead design a lightweight weighting mechanism that trades off a slight performance compromise for considerable savings in computational costs.

To determine the weights, we consider two factors: element-wise weight along with layer-wise weight. The product of element-wise weight and layer-wise weight is used as the ultimate weight for the corresponding gradient element.

3.2.2 Element-wise Weight

Insight 1 *The significance of parameters is closely tied to both their values and gradients. Intuitively, parameters with high-magnitude values are crucial since they can considerably amplify their inputs, thereby yielding a high output that is more likely to exert substantial influence in the model's predictions. Besides, the essence of gradient is to quantify the sensitivity of the loss function to small changes in the parameter value [5]. Hence, parameters associated with larger gradients are considered more influential as they have a greater impact on shaping the model performance.*

Insight 1 motivates defining element-wise weights as the absolute value resulting from multiplying the parameter's value with corresponding gradients. The values of parameters reflect their significance to upstream layers, and the gradients of parameters suggest their importance to downstream layers. Consequently, the element-wise weight intuitively serves as an effective metric to evaluate the importance of parameters.

Table 1: The model performance with three different gradient pruning strategies.

Pruning Rate	0.2	0.4	0.6	0.8
Weight	51.35	48.61	46.80	42.72
Grad	51.67	50.90	49.14	47.35
Ours	**52.88**	**51.45**	**50.96**	**49.54**

Table 2: The model performance when perturbing different layers. The settings are the same as those in Table 1.

Variance	Layer 1	Layer 2	Layer 3	Linear Layer
0.001	48.94	49.92	50.84	51.73
0.01	48.40	49.72	50.25	51.60
0.1	47.56	49.55	49.54	51.38

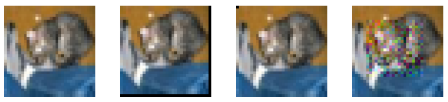
Theoretical justification. We theoretically validate the effectiveness of element-wise weight using the Q-function $Q(u) = \mathcal{L}(F_{u\theta}(x),y)$, where u scales parameter magnitudes [47]. Taking the derivative of $Q(u)$ with respect to u and setting $u = 1$ yields $Q'(1) = \nabla_\theta \mathcal{L}^T \theta$. It is well-known that the model achieves its optimality when $\nabla_\theta \mathcal{L} = 0$ [5]. Therefore, the value of $Q'(1)$ can be used to check the model's optimality. Enforcing $Q'(1) \approx 0$ requires minimizing $|Q'(1)| = |\sum_i \nabla_{\theta_i} \mathcal{L} \cdot \theta_i|$, where i indicates the i-th element of the vector. If only a subset of parameters is allowed to be optimized, focusing on optimizing the parameters with the largest product of absolute weight and gradient, *i.e.*, the parameters that our element-wise weight identifies, can minimize $Q(1)$ to the fullest extent.

Empirical validation. We examine element-wise weight against gradient- and weight-based pruning on LeNet trained on CIFAR-10 (20 epochs, learning rate 0.01, batch size 128), following Zhu et al. [64]. We investigate the performance implications of three distinct gradient pruning strategies: (1) pruning based on the smallest absolute gradient (Grad), (2) the smallest absolute weight (Weight), and (3) the smallest absolute product of weight and gradient (Ours). Table 1 compares test accuracy under varying pruning ratios. As can be seen, element-wise weight achieves the best performance, as it can better identify important parameters compared to the other two pruning strategies.

3.2.3 Layer-wise Weight

Insight 2 *DNNs exhibit hierarchical parameter importance, with shallow layers playing a more critical role in model performance. This stems from two key properties: (1) perturbations in shallow layers propagate exponentially through forward propagation, amplifying their impact on outputs; and (2) shallow layers specialize in learning low-level, shared features across data samples [42].*

Drawing upon Insight 2, the parameters in shallow layers commonly are more important than those in late layers. As



(a) Original (b) Shift (c) Scale (d) Noise

Figure 3: We perturb the original image by shifting it towards the top-left direction by a unit, scaling a pixel, and adding random noises. These make MSEs of 0.025, 0.106, and, 0.004, respectively. Adding random noises produces better privacy protection visually than shifting and scaling, but the former two yield higher MSEs. See Appendix B for SSIM and LPIPS.

reported in Table 2, empirical validation using Gaussian noise injection in gradient confirms that disrupting shallow-layer gradients causes significantly greater performance degradation than deeper layers, motivating our layer-wise weight.

The specific form of layer-wise weight draws inspiration from the multiplicative structure of DNNs. DNNs typically consist of stacked layers, each of which is a linear function followed by a nonlinear activation function $\sigma(\cdot)$ [38]. In mathematical terms, we represent a DNN as $\sigma\cdots\sigma(w_2\sigma(w_1x))$.This structure implies that a perturbation in the parameters of a single layer can exert an exponential effect on the final model output, specifically when considering piece-wise linear activation functions like ReLU. Hence, our layer-wise weight employs an exponential decay mechanism to assign weights.

Suppose that $F_\theta(\cdot)$ comprise a total of K layers, with the i-th layer parameterized with $\theta[i]$ ($\theta=\{\theta[1],\theta[2],\cdots,\theta[K]\}$). The layer-wise weight of gradient elements in i-th layer $\theta[i]$ is defined as $power(\tau,i)$, where τ is a decay factor ($0\leq\tau\leq 1$) and $power(\cdot,\cdot)$ is the power function.

3.2.4 Ultimate Weight

We define the ultimate weight for $\theta[i]$ as the product between element-wise weight and layer-wise weight:

$$weight(\theta[i]) = |grad(\theta[i])\cdot value(\theta[i])\cdot power(\tau,i)|, \quad (3)$$

where $grad(\theta[i])$ and $value(\theta[i])$ denote taking out the gradients associated with $\theta[i]$ and values of $\theta[i]$. UM is then represented as the weighted sum of the squared differences between the gradients associated with x and x^*:

$$UM(x^*,x) = ||\sum_{i=1}^{K} weight(\theta[i]) (\nabla_{\theta[i]}\mathcal{L}(F_\theta(x^*),y) - grad(\theta[i]))||_2^2. \quad (4)$$

3.3 Privacy Metric

3.3.1 Motivation

PM aims to quantify the level of privacy leakage regarding x caused by x^*, or in other words, it measures the dissimilarity between x and x^* in terms of human perception. However, common distance metrics such as MSE fall short as an effective PM. In detail, for common metrics, high similarity scores ensure perceptual equivalence, but low scores do not guarantee perceptual differences. Consider Figure 3, where seemingly indistinguishable images to human eyes are deemed significantly different by MSE. Consequently, relying on MSE risks pseudo-privacy protection. This necessitates a PM that better reflects human perceptual thresholds for privacy leakage.

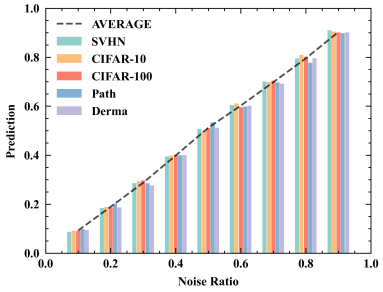


Figure 4: We extract 1000 samples from SVHN, CIFAR10, CIFAR100, Path, and Derma. We report the average predictions of the evaluation network for the mixture of these samples and random noises with proportions ranging from 0.1 to 0.9.

3.3.2 Metric Design

An ideal PM should satisfy the monotonicity principle: increasing $PM(x^*,x)$ corresponds to decreasing privacy information in x^*. One intuitive idea for designing such a PM is to generate a reference sample devoid of any private information associated with x. This reference sample acts as a benchmark to evaluate the amount of privacy information retained in x^*. The closer x^* is to the reference sample, the less private information from x will be embedded in x^*. However, a key challenge arises: how to obtain a suitable reference sample?

At first glance, one might consider utilizing random noises[1] as the reference sample. Though technically feasible, it produces a bottleneck for the performance of *Refiner* by restricting x^* solely to a limited trajectory from x to that specific reference, thus bypassing other possibly more informative regions of noises. To address the problem, we define the distance of x^* to the uniform distribution as our PM. Unlike specific random noises, the uniform distribution offers a more flexible and adaptive benchmark.

Let x^* be distributed according to Dirac delta distribution $p(x)$ that concentrates mass at x^*. To measure the distance between x^* and uniform noise distribution $q(x)$, we employ the JS divergence: $\frac{1}{2}\int p(x)log\frac{2p(x)}{p(x)+q(x)} + q(x)log\frac{2q(x)}{p(x)+q(x)}dx$. Since directly evaluating JS divergence is inhibited by the high dimensionality of x, we reformulate JS divergence as

[1] Random noises are considered as non-information samples [41].

follows (see Appendix C for details):

$$\max_{D,D(x)\in[0,1]} \mathbb{E}_{x\sim p(x)}[log(1-D(x))]+\mathbb{E}_{x\sim q(x)}[logD(x)]. \quad (5)$$

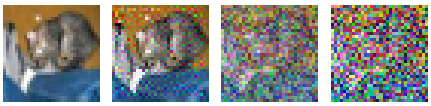
Practical Implementation. Realizing Equation 5 entails constructing a new D for each x to estimate the JS divergence. Seeking efficiency, we search for a single D for all images. We employ a neural network, dubbed the evaluation network, to serve this role, as DNNs are capable of approximating any function. The goal of Equation 5 is to encourage D to output 1 for random noises and 0 for natural samples. Intuitively, the evaluation network in fact quantifies the proportion of noise within x^*. This inspires us to create training data-label pairs for the network in an interpolate manner, *i.e.*, $((1-r)\cdot t_1 + r\cdot t_2, r), r\sim U(0,1), t_1\sim p, t_2\sim q$, and these pairs are supplied into the network as supervised signals for training. Doing so enjoys that the monotonicity principle can be explicitly instilled into the evaluation network.

We train the evaluation network using TinyImageNet dataset [29] due to its wide range of common images (more training details can be found in Appendix D). Figure 4 shows the network's ability to predict noise ratios across mixed samples of random noise and natural images, showcasing strong cross-dataset generalization capabilities. Notably, it even delivers outstanding performance on significantly different datasets like SVHN (a dataset composed of digit images) [31] and medical datasets including Path and Derma, further solidifying its wide-ranging applicability. We highlight that this network can be trained using publicly available, non-sensitive datasets like TinyImageNet, preventing privacy leaks. The abundance of public, non-private datasets makes it feasible for clients to have an evaluation network. In practical scenarios, clients can obtain the trained evaluation network from a trusted third party at no cost or train their own evaluation network using publicly available data. Moreover, the model has a compact size of approximately 1MB and requires less than 10 minutes of training time on a single GTX A10 GPU, making training cost manageable. In this paper, all clients share this trained evaluation network by default[2].

3.4 Solving Optimization Task (Equation 2)

Upon establishing the specific formulations for both UM and PM, the remaining challenge lies in resolving Equation 2. A straightforward solution is to harness gradient descent algorithm [5], widely recognized for its efficacy in solving optimization tasks that feature primarily convex components, such as UM. However, the algorithm encounters obstacles when confronted with highly non-linear and non-convex neural networks like PM.

To show this, we initialize x^* as x and then maximize $PM(x,x^*)$, *i.e.*, with the hope of resultant x^* being noisy. As shown in Figure 5(b), the resultant x^* deviates only slightly from x (Figure 5(a)), yet the model confidently identifies it as a noisy sample. This phenomenon and x^* are known as adversarial vulnerability and adversarial examples [6, 18].



(a) (b) (c) (d)

Figure 5: We optimize the above image (a) and (c) for 10 iterations with learning rate of 1. The above image (c) represents a mixture of the original image and random noises, with $\alpha = 0.5$. The above image (b) and (d) denote the optimized images resulting from the above image (a) and (c), respectively. Without noise-blended initialization, the adversarial vulnerability of DNNs results in generated images that maintain similar semantic information to the original image.

To address the above problem, we propose noise-blended initialization which mixes x^* with random noises, *i.e.*, $x^* = (1-\alpha)x + \alpha v,\ v\sim q(x), \alpha\in[0,1]$. α is a blend factor. A high value of α positions x^* closer to areas dense with noises and ensures that the evaluation network will maintain x^* as a noisy sample during the optimization process. Figure 5(c) exemplifies the noise-blended initialization with $\alpha = 0.5$. The resultant x^* is devoid of semantic information regarding x, showing the effectiveness of the noise-blended initialization.

4 Theoretical Analysis

4.1 Performance Maintenance

Assumption 1 *$\mathcal{L}(F_\theta(x),y)$ is L-smooth: $\forall\theta_1,\theta_2$, it holds that $\mathcal{L}(F_{\theta_1}(x),y) \le \mathcal{L}(F_{\theta_2}(x),y) + \nabla_{\theta_2}\mathcal{L}(F_{\theta_2}(x),y)^T(\theta_1-\theta_2) + \frac{L}{2}||\theta_1-\theta_2||_2^2$.*

Assumption 2 *$\mathcal{L}(F_\theta(x),y)$ is u-strongly convex: $\forall\theta_1,\theta_2$, there is $\mathcal{L}(F_{\theta_1}(x),y) \ge \mathcal{L}(F_{\theta_2}(x),y) + \nabla_{\theta_2}\mathcal{L}(F_{\theta_2}(x),y)^T(\theta_1-\theta_2) + \frac{u}{2}||\theta_1-\theta_2||_2^2$.*

Assumption 3 *Let x_i, y_i be uniformly random samples drawn from the local dataset D_i of the i-th client. The variance of stochastic gradients is bounded: $\mathbb{E}||\nabla_\theta\mathcal{L}(F_\theta(x_i),y_i) - g_i^{full}||_2^2 \le \sigma_i^2, i=1,2,\cdots,K$. Wherein, g_i^{full} represents the gradients of model parameters $\nabla_\theta\mathcal{L}(F_\theta(\cdot),\cdot)$ computed across the complete local dataset D_i.*

Assumption 4 *There exists a real number G and the expected squared norm of stochastic gradients is uniformly bounded by G: $\mathbb{E}||\nabla_\theta\mathcal{L}(F_\theta(x_i),y_i)||_2^2 \le G^2, i=1,2,\cdots,K$.*

Theorem 1 *Let Assumption 1, 2, 3, and 4 hold. Let x^* be the optimal solution of Equation 2. Suppose the gradient distance associated with x and x^* is bounded: $||\nabla_\theta\mathcal{L}(F_\theta(x),y) -$*

[2]We observe comparable defense effectiveness when clients train individual networks using public data.

$\nabla_\theta \mathcal{L}(F_\theta(x^*), y)||_2^2 \leq \varepsilon^2$. *Let* θ^* *represent the global optimal solution and* θ_i^* $(i = 1, 2, \cdots, K)$ *denote the local optimal solution for i-th client. Choose* $\kappa = \frac{L}{u}, \gamma = \max\{8\kappa, 1\}, \eta = \frac{2}{u(\gamma+t)}$. *Let* $\Gamma = \mathbb{E}[\mathcal{L}(F_{\theta^*}(\cdot), \cdot)] - \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}[\mathcal{L}(F_{\theta_i^*}(\cdot), \cdot)]$. *There is:*

$$\begin{aligned} &\mathbb{E}[\mathcal{L}(F_\theta(\cdot), \cdot)] - \mathcal{L}(F_{\theta^*}(\cdot), \cdot) \\ &\leq \frac{2\kappa}{\gamma + N}\left(\frac{Q + C}{u} + \frac{u\gamma}{2}\mathbb{E}||\theta_{initial} - \theta^*||_2^2\right), \end{aligned}$$

where $Q = \frac{1}{K^2}\sum_{i=1}^{K}(\varepsilon^2 + \sigma_k^2) + 6L\Gamma$, $C = \frac{4}{K}(\varepsilon^2 + G^2)$, *and* $\theta_{initial}$ *is initial parameters of the globally-shared model.*

This subsection examines how *Refiner* contributes to the convergence of the global model. We here demonstrate Theorem 1 by making Assumption 1-4. See I for the proof of Theorem 1. Assumption 1 posits that DNNs' loss landscape does not exhibit abrupt curvature changes, which is satisfied by common DNNs and empirically validated [13]. Assumption 2 follows the technical conventions. Notably, existing theoretical work [28] demonstrated that DNNs exhibit strong convexity near local optima. From initialization perspective, DNNs' parameters reside within a basin of attraction and converge to the corresponding optimal point (though some methods [9] explore transitions between basins). Assumptions 3 and 4 are intuitively valid. In practice, it is almost impossible for the gradient from certain data points to be infinitely far from the full dataset's gradient or infinitely large (Assumptions 3 and 4). As a result, Theorem 1 enjoys good applicability over common DNNs.

Theorem 1 delineates an upper bound on the gap between the optimal solution and the model trained with *Refiner* after sufficient rounds. As the gradient distance between robust and client data decreases, this upper bound also decreases, implying better performance of the global model. This highlights the effectiveness of minimizing the gradient distance between robust data and clients' data in maintaining performance. The inclusion of the gradient distance term (UM) in Equation 2 guarantees the boundedness of $||\nabla_\theta \mathcal{L}(F_\theta(x), y) - \nabla_\theta \mathcal{L}(F_\theta(x^*), y)||$. In practice, we modify the gradients of robust data slightly.

If the difference between the gradients of robust data and clients' data exceeds ε, we search for the closest gradients within the ε-constraint and use them as the final uploaded gradients (see Appendix J for more details). Doing so enables us to evaluate the utility-privacy trade-off of *Refiner* by varying values of ε[3].

Moreover, FL commonly involves two scenarios: when the client data distributions are identical (IID) and non-identical (Non-IID). Our theoretical analysis does not build upon homogeneity across client datasets, rendering Theorem 1 applicable to both scenarios.

[3]Although adjusting the value of β is also an alternative way, we find that tuning ε is more convenient.

4.2 Privacy Protection

GLAs vary widely in formulation but share a common goal: reconstructing training data from gradients via inverse mappings $\hat{f}^{-1}(\cdot)$. To enable broad privacy analysis, we abstract GLAs into a unified framework where $f(x) = g$ represents the forward gradient mapping, and $\hat{f}^{-1}(g)$ approximates the inverse mapping with bounded error σ (i.e., $\text{dist}(\hat{f}^{-1}(\cdot), f^{-1}(\cdot)) \leq \sigma$). $\text{dist}(\cdot,\cdot)$ is unspecified distance metric and σ is a small positive number. This abstraction accommodates diverse attack variants (Examples 1-2) by focusing on their shared objective of minimizing $\text{dist}(\hat{f}^{-1}(g), x)$.

Example 1 *(Inverting Gradients [16]). Similar to DLG, Geiping et al. [16] also solves a gradient matching problem for data reconstruction. But* L_2*-norm distance is replaced by cosine similarity and a total variation term used to regularize reconstructed data is introduced:* $\hat{x} = \hat{f}^{-1}(g) = \arg\min_{\hat{x}} 1 - \frac{<\nabla_\theta F_\theta(\hat{x}), g>}{||\nabla_\theta F_\theta(\hat{x})||_2 ||g||_2} + TV(x)$.

Example 2 *(Generative Gradient Leakage (GGL) [24]). GGL optimizes a latent input vector of a generative model to search data whose gradients align well with uploaded gradients. Moreover, GGA adds a KL divergence term to avoid too much deviation between the latent vector and the generator's latent distribution:* $\hat{x} = \hat{f}^{-1}(g) = G(z)$, *where* $z = \arg\min_x ||\nabla_\theta F_\theta(G(z)) - g||_2^2 + KL(z||q)$. q *is Gaussian distribution with a mean of 0 and a standard deviation of 1.*

When employing *Refiner*, clients upload gradients associated with x^*, denoted by $g^* = f(x^*)$. The server exploits g^* to recover data, *i.e.*, $\hat{f}^{-1}(g^*)$. We now consider the distance of x and $\hat{f}^{-1}(g^*)$:

$$\begin{aligned} &\text{dist}(\hat{f}^{-1}(g^*), x) = \text{dist}(\hat{f}^{-1}(g^*), f^{-1}(g)) \\ &\geq |\ \text{dist}(f^{-1}(g^*), f^{-1}(g)) - \text{dist}(\hat{f}^{-1}(g^*), f^{-1}(g^*))\ | \\ &= |\ \text{dist}(x^*, x) - \text{dist}(\hat{f}^{-1}(g^*), f^{-1}(g^*))\ |. \end{aligned} \tag{6}$$

Equation 6 establishes a lower bound of $\text{dist}(\hat{f}^{-1}(g^*), x)$, which hinges upon x^*, x, and σ. In practice, x^* obtained by Equation 2 are often noisy and significantly differ from x, *i.e.*, $\text{dist}(x^*, x) > \sigma$ (Appendix K). When the value of σ is small, the lower bound is simplified as follows:

$$\text{dist}(\hat{f}^{-1}(g^*), x) \geq \text{dist}(x^*, x) - \sigma. \tag{7}$$

Equation 7 suggests that a smaller σ results in higher privacy lower bound, *i.e.*, stronger privacy protection capabilities. On the other hand, considering a large σ may be a little trivial, because a large σ leads to a significant discrepancy between the recovered data and clients' data.

To delve deeper, we adopt two specific metrics, including Euclidean distance and our evaluation network. The use of Euclidean distance enables Equation 7 to be restated:

$$||\hat{f}^{-1}(g^*) - x||_2^2 \geq ||x^* - x||_2^2 - \sigma. \tag{8}$$

Table 3: Time complexity comparison. $\mathcal{N}, \mathcal{N}_{last}, S_1, S_2, I$ denote the total number of model parameters, the total number of model parameters in the last fully connected layer, the time required for one forward-backward propagation of the globally-shared model and the evaluation network, the iteration number for solving Equation 2, respectively.

DP	GQ	Pruning	Soetria	Ours
$O(\mathcal{N})$	$O(\mathcal{N})$	$O(\mathcal{N}\log(\mathcal{N}))$	$O(\mathcal{N}_{last} \cdot S_1)$	$O(\imath \cdot (2 \cdot S_1 + S_2))$

The distance for the evaluation network is defined as the absolute difference in noise ratios between the two data[4]. Given that x is natural data, $\text{dist}(\hat{f}^{-1}(g^*), x)$ and $\text{dist}(x^*, x)$ reflect the amount of random noises present within $\hat{f}^{-1}(g^*)$ and x^*. Thus, Equation 7 with the evaluation network is translated as follows:

$$D(\hat{f}^{-1}(g^*)) \geq D(x^*) - \sigma. \tag{9}$$

The above equation indicates that, if x^* carries a substantial noise fraction, the data recovered by attackers, too, will be significantly distorted by noises.

4.3 Time Complexity

Before analyzing time complexity, it is essential to recognize that achieving excellence in all aspects of utility, privacy, and time is almost impossible. Generally, improving the utility-privacy balance demands additional time investment. Consider DP, which equally perturbs all gradient elements without considering their privacy information. In contrast, Soteria estimates the privacy information of each gradient element and selectively prunes those with the highest privacy information, thereby improving the trade-off but at the cost of extra time incurred by the estimation process.

DP, GQ, and gradient pruning, none of which necessitate extra forward-backward propagation, are discussed together. DP generates noises for each gradient element, GQ quantizes each gradient element, and gradient pruning sorts and then removes part of the gradient elements. Assuming a model with $\mathcal{N}$ total parameters, the time complexity of both DP and GQ scales linearly with the number of parameters, $O(\mathcal{N})$. For gradient pruning, the dominant factor is the sorting time complexity, *i.e.*, $O(\mathcal{N}\log(\mathcal{N}))$. Soteria estimates privacy information for each gradient element in the fully connected layer[5]. Given that the total count of parameters contained within the layer is to be denoted as $\mathcal{N}_{last}$, and once forward-backward propagation of the global model holds a time complexity of $O(S_1)$. Soteria iteratively performs forward-backward propagations a specific number of times, corresponding to the number of neurons in the layer. Consequently, the overall time complexity of Soteria can be expressed as $O(\mathcal{N}_{last} \cdot S_1)$.

[4] See Appendix L for proof of the definition qualified as distance metric.

[5] Typically, a fully connected layer comprises around 1024 neurons.

The time complexity of *Refiner* centers around solving Equation 2. Each iteration of Equation 2 requires twice forward-backward propagations of the global model, along with one forward-backward propagation of the evaluation network. Hence, the overall time complexity is the sum of these operations. Assuming the time complexity of a single forward-backward propagation of the evaluation network is $O(S_2)$ and the quantity of iterations required to solve Equation 2 is $\imath$, the time complexity of *Refiner* is $O(\imath \cdot (2 \cdot S_1 + S_2))$. Table 3 summarizes the time complexities of these defenses.

4.4 Security Discussion

We here conduct a security analysis of *Refiner*, *i.e.*, the resilience of *Refiner* against adaptive attacks. Let us consider a white-box scenario where attackers have complete access to any knowledge of the deployed defense mechanisms, excluding x and g. According to our theoretical analysis, attackers can reconstruct x^* via clients' uploaded gradients g^*. Our method centers on Equation 2, with attackers attempting to reverse-engineer the ground-truth gradients associated with x using Equation 2. Consequently, Equation 2 boils down to $||g^* - g||_2^2 = const$, where all constant terms are absorbed into *const*, and we omit gradient weighting factors for notation simplicity. Owing to the high dimensionality of the gradients, this equation stands as inherently under-determined, unless the gradient dimension is 1, which is nearly impossible in real-world scenarios. Further discussions and validations can be found in Appendix M.

5 Evaluation Setup

5.1 Attack

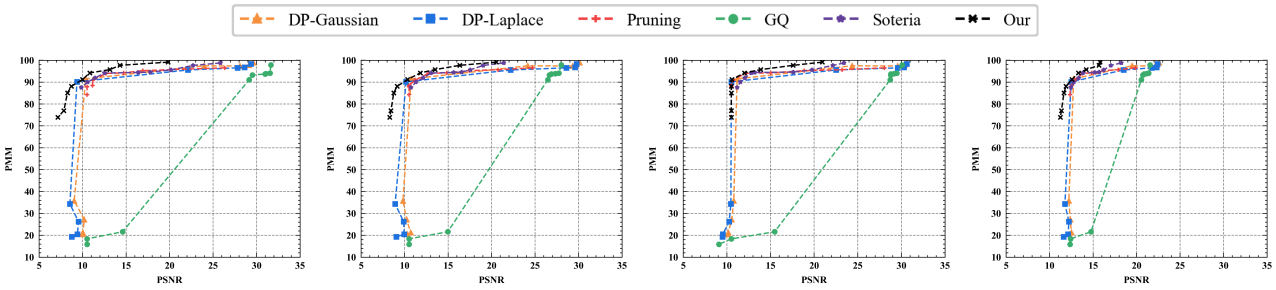
Four state-of-the-art attacks are used to test the performance of *Refiner*, including iDLG [62], GradInversion [56], InvertingGrad [16] and GGL [24, 57]. These attacks cover the primary types of GLAs currently explored. The former three attacks address variants of Equation 1 by employing varying optimizers, loss functions, etc. GGL diverges from this path by leveraging latent vector alignments within a GAN model. We implement these attacks at the start of training, as this stage is most vulnerable to GLAs [3]. For GGL, we use public codes [6] to train a GAN. See Appendix D for an overview of the attack distinctions and the hyperparameters employed.

5.2 Competitors

We compare *Refiner* to the following defenses: DP [1], GQ [40, 57], gradient pruning [2, 64], and Soteria [50][7]. Each

[6] https://raw.githubusercontent.com/pytorch/examples/master/dcgan/main.py

[7] We use Soteria proposed in [50] to compare, which is an improved version compared with the original Soteria [43].



(a) iDLG (b) GradInversion (c) InvertingGrad (d) GGL

Figure 6: The performance (PMM-PSNR curves) of defenses against four state-of-the-art attacks in LeNet trained with CIFAR10.

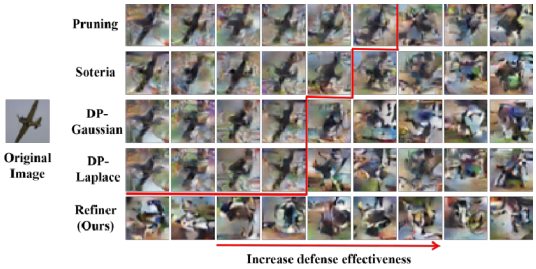
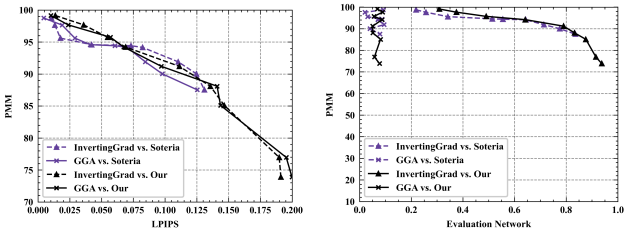


Figure 7: A visualization of five defenses over varying hyperparameters against GGL. The images above the red broken lines contain privacy information of the original images.



(a) LPIPS (b) Evaluation Network

Figure 8: The defense performance of Soteria and *Refiner* against InvertingGrad and GGL using LPIPS and the evaluation network to assess. We use LeNet and CIFAR10.

of these defenses involves a utility-privacy trade-off that is regulated by hyperparameters such as noise magnitude for DP, discretization level for GQ, and pruning rate for pruning and Soteria. We vary these hyperparameters to obtain the utility-privacy trade-off curves for each defense. Specifically, for DP, we employ two commonly used kinds of noises, namely Gaussian (DP-Gaussian) and Laplace (DP-Laplace), with magnitude ranging from 10^{-6} to 10^{2} and C of 1. As for GQ, we consider discretization to 1, 2, 4, 8, 12, 16, 20, 24, and 28 bits. The pruning ratio for gradient pruning and Soteria is chosen from the range $0.1, 0.2, \cdots, 0.9$. Besides, we alter ε of *Refiner* over $\{0.01, 0.03, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 0.9\}$ (see Section 4.1). Unless otherwise specified, for *Refiner*, the default values of α (blend factor in noise-blended initialization), β (balance factor in Equation 2), ι (iterations for solving Equation 2), and τ (decay factor in layer-wise weight) are set to 0.5, 1, 10, and 0.95. These values are identified through preliminary experiments as effective in balancing privacy and utility, with our results demonstrating that *Refiner*'s performance is not highly sensitive to hyperparameter choices.

5.3 Evaluation Metrics

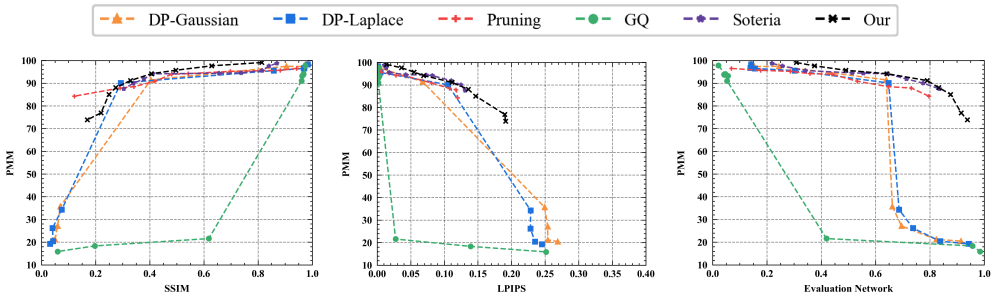
We assess defenses from three fundamental dimensions: performance maintenance, privacy protection, and time cost.

Performance maintenance. We define the performance maintenance metric (PMM), which calculates the accuracy ratio achieved with defenses compared to the original accuracy without any defenses: $PMM = \frac{defense\_acc}{original\_acc} \times 100\%$, where $original\_acc$ and $defense\_acc$ denote the accuracy of the trained global-shared model with and without defense over test sets of corresponding datasets[8].

Privacy protection. Privacy protection measures the amount of privacy information recovered from the reconstructed images. Given the lack of a universally perfect privacy assessment metric, we employ a diverse range of metrics, including PSNR [19] (↓), SSIM [51] (↓), LPIPS [60] (↑), and the prediction noise ratio of the evaluation network (↑), to achieve a more comprehensive assessment. Thresholds of 15 for PSNR, 0.5 for SSIM, 0.05 for LPIPS, and a noise ratio of 0.4 from the evaluation network represent the junctures at which human eyes struggle to discern any private information of the original data from recovered data. Please see Appendix N for a visual guide that illustrates how changes in these values affect the level of privacy exposure.

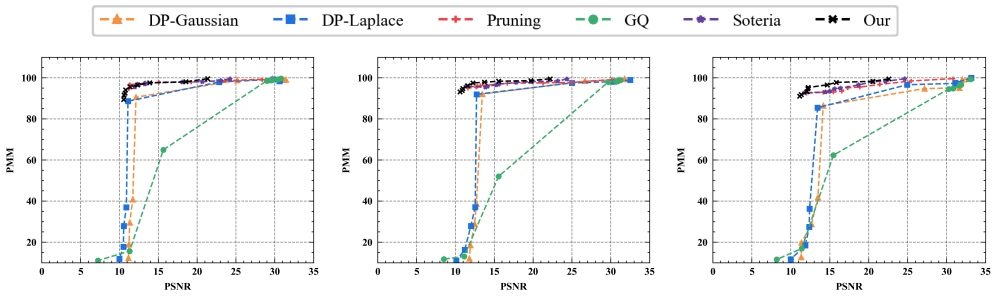
Time cost. Time cost is also an important dimension for evaluating a defense. We record the time required for a single iteration of the model when employing defenses.

[8]The original accuracies of LeNet are 84.12%, 54.01%, and 21.45% on SVHN, CIFAR10, and CIFAR100. The original accuracies of ResNet10, ResNet18 and ViT are 83.25%, 84.95%, 87.52% on CIFAR10.



(a) SSIM (b) LPIPS (c) Evaluation Network

Figure 9: The performance of defenses against InvertingGrad in LeNet trained with CIFAR10 using different metrics to assess.



(a) ResNet10 (b) ResNet18 (c) ViT

Figure 10: The performance of various defenses against InvertingGrad in ResNet10, ResNet18 and ViT trained with CIFAR10.

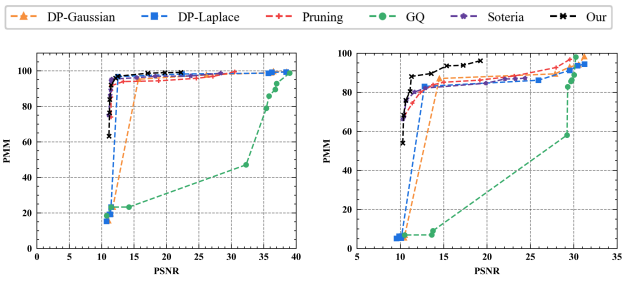
5.4 Model, Dataset, and Attack Settings

We select three widely-used models, LeNet, ResNet, and ViT [10], on three benchmark datasets SVHN, CIFAR10, and CIFAR100. LeNet and ResNet are CNN-based models while ViT is a transformer-based model, covering mainstream model architectures. We include ResNet10 and ResNet18 to study the effects of model scale. Our focus is on a typical FL scenario involving 10 clients collaborating to train a global model, with a server.

Our evaluation also includes non-IID settings [20]. Following [26], the IID setting equitably and randomly segments training datasets among clients, while the non-IID setting introduces label imbalance, distributing data unevenly among the 10 clients [20, 25].

Training setting. The FL training process takes place over 8000 rounds, during which each client computes gradients locally using a batch size of 128. The server averages the uploaded gradients for global model updates with a learning rate of 0.01. Input data is normalized within the 0 to 1 range.

Attack setting. According to [61], a batch size of 1 in the first few training rounds leads to the most effective attack scenario, as smaller batch sizes can reduce the search space and increase attack effectiveness. Motivated by this finding, we set the batch size to 1 to assess defense performance under worst-case scenarios. Here, clients craft uploaded gradients for each sample with a certain defense. The server then exploits the uploaded gradients to reconstruct clients' raw data and we report the average quality of 1000 reconstructed data. See Figure 18 (Appendix E) for the defense performance of *Refiner* over different rounds.



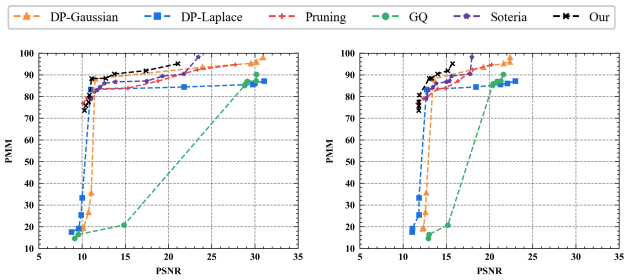
(a) SVHN (b) CIFAR100

Figure 11: The performance of various defenses against InvertingGrad in LetNet trained with SVHN and CIFAR100.

6 Empirical Evaluation

6.1 Privacy-Utility Trade-off

Numerical results. Figure 6 depicts the trade-off curves between PMM (utility) and PSNR (privacy) for various defenses



(a) InvertingGrad (b) GGL

Figure 12: The performance (PMM-PSNR curves) of defenses against InvertingGrad in LetNet with Non-IID setting.

against four state-of-the-art attacks on CIFAR10. Overall, *Refiner* consistently outperforms baselines by achieving lower PSNR while maintaining comparable PMM. Moreover, we observe that the impact on model performance by *Refiner*, Soteria, and Pruning is relatively mild compared to other defenses. Particularly, excessively aggressive defense strengths, such as a noise magnitude of 10^{-2} or above for DP and a quantization precision of 8 bits or fewer for GQ, can significantly deteriorate the model's performance.

Semantic results. We observe that the reconstruction quality of GGL, as measured by PSNR, is significantly inferior to the other three attacks. One possible explanation is that GGL focuses more on semantics rather than exact pixel-level detail. To substantiate this conjecture, Figure 7 showcases the reconstruction images of GGL. Wherein, GGL indeed recovers the privacy information embedded in the original image (i.e., airplane), highlighting its attack effectiveness. Furthermore, compared to the PSNR metric, LPIPS places greater emphasis on the similarity of DNN-extracted features, thereby underscoring the importance of semantic-level similarity over pixel-level similarity. Therefore, we deploy LPIPS metric and the evaluation network for further assessment of GGL's effectiveness. From Figure 8(a), we observe that the attack performance of GGL is competitive with InvertingGrad against Soteria and *Refiner*. Intriguingly, in Figure 8(b), the evaluation network consistently rates GGL's attack effectiveness highly across varying defense strengths. This aligns with our expectations, given that GGL's output images are noise-free, as displayed in Figure 7. The above analysis stresses the necessity of employing diverse privacy metrics.

Multi-Metric evaluation. Figure 9 shows the performance of defenses under various privacy metrics. As can be seen, *Refiner* achieves a superior trade-off between privacy and utility across a multiplicity of evaluation metrics. For instance, *Refiner* enables the inclusion of approximately 30% noise information in constructed data with negligible impact on the model's performance, whereas Soteria achieves only around 20% noise inclusion capability.

Safeguarding different model architectures. Figure 10 presents the performance of various defenses when applied to ResNet and ViT. Remarkably, *Refiner* exhibits a superior trade-off between utility and privacy, particularly manifest when implemented on larger-scale models. We hypothesize that this can likely be attributed to the increased non-linearity capacity of large models, rendering the underlying assumptions of other defenses less valid and consequently leading to degraded effectiveness.

Table 4: The elapsed time in seconds (s) for a client per round with varying defenses using an A10 GPU. We use a batch size of 128. "Ours + Warm-Start" refers to *Refiner* with the warm-start strategy, where robust data from previous rounds is reused to reduce the iteration count (halving ι).

Dfense	LeNet	ResNet10	ResNet18	ResNet34	ViT
Pruning	0.0187	0.0495	0.0873	0.1370	1.1232
Gaussian	0.0467	0.7317	1.8408	3.3020	15.5051
Laplace	0.0469	0.7322	1.8414	3.3025	15.5236
GQ	0.0179	0.0277	0.0514	0.1050	0.4489
Soteria	0.4103	11.3148	21.7539	39.4410	347.5668
Ours	0.3340	1.7341	2.8539	4.7012	30.2268
Ours + Warm-Start	0.1418	0.8523	1.4262	2.3548	15.0234

Table 5: The PSNR and PMM achieved by *Refiner* over varying β against InvertingGrad in LeNet trained with CIFAR10.

β	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}	10^{3}
PSNR	13.89	13.62	13.44	13.20	13.08	12.70	12.04
PMM	91.93	91.12	91.06	90.69	90.00	89.30	88.90

Safeguarding different datasets. We extend our examination to alternative datasets, namely SVHN and CIFAR100. The results, as shown in Figure 11, not only confirm the consistent superiority of *Refiner*' performance, but also reveal two noteworthy observations. Firstly, we observe a better reconstruction quality for attacks carried out on SVHN, with the highest achieved PSNR approaching 40. This is probably because SVHN is a less sophisticated dataset, rendering the construction easier. Secondly, we observe the remarkable effectiveness of *Refiner* compared to baselines when applied to the CIFAR100 dataset, suggesting that *Refiner* exhibits greater resilience in handling more complex datasets.

Defense with Non-IID setting. We use client-label-imbalance configuration and employ Dirichlet distribution with a concentration parameter of 1 to allocate fractions of each class label to different clients. We then perform without-replacement sampling from the corresponding original training datasets for each client, guaranteeing alignment with this allocation. The empirical results, shown in Figure 12, shows that *Refiner* maintains its edge over the baselines even under the Non-IID setting. Moreover, while we observe little change in the quality of attacker-reconstructed images under non-IID conditions, the difficulty of maintaining performance increases significantly compared to the IID setting.

Table 6: The PSNR and PMM achieved by *Refiner* over varying ι against InvertingGrad in LeNet trained with CIFAR10.

ι	5	10	15	20
PSNR	13.05	13.20	13.37	13.45
PMM	85.61	90.69	91.19	92.17

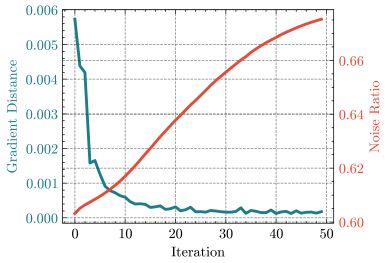
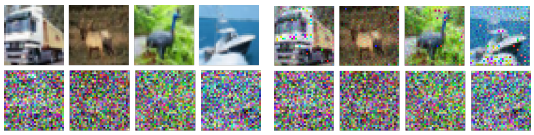


Figure 13: Gradient distance and noise ratio trends across iterations.



(a) Raw Data and Robust Data (b) Recovered Images

Figure 14: (a) original data (top row) and robust data (bottom row). (b) gradient-reconstructed counterparts.

6.2 Visual Analysis

To intuitively understand *Refiner*'s effectiveness, we provide qualitative analyses here. Figure 13 shows the gradient distance between robust and original data alongside the evaluation network's predicted noise ratio across iterations. As shown, *Refiner* achieves convergence within 10 iterations, with gradient distances stabilizing after 20 iterations. Figure 14 visualizes four randomly selected CIFAR-10 samples: (a) original vs. robust data (10 iterations), and (b) images reconstructed from gradients of raw data and robust data using InvertingGrad. The robust data appears as pure noise to the human eye. According to our theoretical analysis, the corresponding reconstructed data should also exhibit noise characteristics, which Figure 14(b) verifies. Furthermore, we observe that reconstructed images exhibit strong correlations with their robust counterparts. In detail, the third sample shows green tones, and the last sample preserves blue hues.

6.3 Empirical Time Complexity

The additional time cost imposed by defenses is a crucial dimension in evaluating their practicality. Table 4 compares the runtime required for a single client per round when employing different defenses. While *Refiner* and Soteria incur similar

Table 7: The PSNR and PMM achieved by *Refiner* over varying α against InvertingGrad in LeNet trained with CIFAR10.

α	0	0.1	0.3	0.5	0.7	0.9
PSNR	21.24	16.52	14.99	13.20	12.53	10.52
PMM	98.52	95.45	92.61	90.69	90.08	89.34

Table 8: The PSNR and PMM achieved by *Refiner* over varying τ against InvertingGrad in LeNet trained with CIFAR10.

τ	0.91	0.93	0.95	0.97	1.00
PSNR	12.93	13.05	13.20	13.25	15.44
PMM	90.98	90.82	90.69	90.46	89.28

time costs for compact models like LeNet, both are an order of magnitude higher than lightweight methods like Pruning, Gaussian, Laplace, and Gradient GQ. This is due to the fact that *Refiner* and Soteria require multiple forward-backward passes to search for better utility-privacy trade-offs.

When scaling up to larger models including ResNet and ViT, the execution time for Pruning and GQ scales almost linearly. *Refiner*, Gaussian, and Laplace present time expenses of the same magnitude. Notably, Soteria makes a significantly higher time cost of 40s on ResNet34, which is 8.4× higher than the time required for *Refiner*. The heavy overhead of Soteria stems from the intensive execution of forward-backward passes per neuron within the fully connected layers. As a result, as the number of fully connected neurons increases in large models, the time cost of Soteria escalates drastically. In contrast, *Refiner* 's time cost depends on the predefined number of iterations. Empirical results indicate that *Refiner* achieves satisfactory performance within merely 10 iterations, far lower than the overhead associated with Soteria. Moreover, we introduce a warm-start strategy to further reduce *Refiner* 's overhead, which reuses robust data generated in earlier rounds as initialization for subsequent iterations (excluding the first round for generating robust data for x). Figure 15 evaluates the impact of this strategy on *Refiner* 's performance across different ι values. We find that reducing ι to 5 just results in tiny performance degradation: PMM decreases slightly from 90.69 to 90.45, while PSNR increases marginally from 13.20 to 13.27 on LeNet trained with CIFAR-10 against InvertingGrad. Therefore, we set ι to 5 after employing the warm-start strategy to reassess the time cost, as shown in Table 4. This makes *Refiner* significantly more efficient than Soteria, while maintaining a better utility-privacy trade-off than DP, GQ, and gradient pruning. Additionally, we estimate *Refiner* 's overhead on edge devices to be within acceptable limits (Appendix O provides details).

6.4 Comparison to Cryptographic Methods

Table 9 compares the performance of secret-sharing-based cryptographic methods [4] against InvertingGrad in CIFAR-

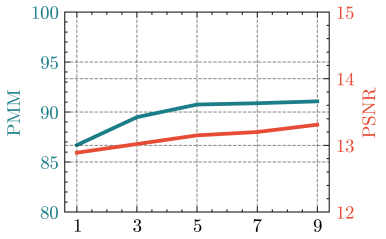


Figure 15: Impact of warm-start strategy on *Refiner*'s performance under varying ι values. We employ InvertingGrad on LeNet trained with CIFAR-10.

Table 9: Performance of cryptographic methods [4] under InvertingGrad attacks. PSNR values correspond to batch sizes of 1 and 128.

Model	PMM	PSNR	Time (Client/Server)
LeNet	99.91	18.67/10.24	2.0784s/2.3185s
ResNet10	100.24	15.58/10.39	642.6589s/719.8361s
ResNet18	100.05	15.52/10.08	1460.1683s/1635.4287s

10, reporting PSNR values for batch sizes of 1 and 128. Cryptographic methods impose computational overhead not only on clients but also on the server, so we report the total time per round for both client and server. Compared to perturbation-based methods (e.g., *Refiner* in Table 4), cryptographic methods incur much higher costs. For LeNet, cryptographic methods are 6× slower than *Refiner* without warm-start and 14.6× slower with warm-start. On larger models like ResNet, the overhead becomes orders of magnitude greater, reaching hundreds of times the cost of *Refiner*. While cryptographic methods maintain near-perfect PMM (e.g., 99.91–100.24), their prohibitive time costs severely limit practicality. Notice that homomorphic encryption-based methods are not included due to two key factors [54]. First, they incur significantly higher computational costs than secret-sharing-based methods. Second, homomorphic encryption can enable the server to perform algebraic operations directly on encrypted parameters without decryption [36], which inherently eliminates the threat model assumed by GLAs.

6.5 Sub-components Analysis

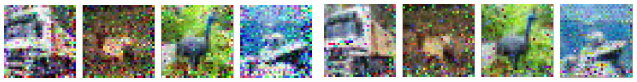
In this subsection, we investigate the impact of four sub-components of *Refiner* on its performance. We fix $\varepsilon = 0.1$.

Impact of balance factor β. Table 5 reports the performance of *Refiner* as β varies against InvertingGrad. Overall, the higher β is, the lower PMM is, and the higher MSE is. Moreover, the performance of *Refiner* is more robust to adjustments in β as compared to α, with negligible performance fluctuations across different values of β. Thus, maintaining β at 1 may be a practical default.

Impact of iterations ι. Table 7 reports the performance of *Refiner* when configured with varying α against InvertingGrad. Increasing iterations from 5 to 10 significantly bolsters *Refiner*'s performance, enabling the acquisition of higher PMM with similar PSNR. Intuitively, early iterations tend to yield higher marginal benefits as the algorithm rapidly approaches convergence. Beyond a certain number of iterations (here identified as 10 iterations) is surpassed, *Refiner* reaches a point of saturation, beyond which significant returns are no longer generated. Hence, from both a performance and computational perspective, an approximation of 10 iterations serves as a practical choice.

Table 10: The performance of *Refiner* with and without element-wise weight against InvertingGrad in LeNet trained with CIFAR10.

Refiner	PSNR	PMM
w.o. Element-wise Weight	13.15	85.32
w. Element-wise Weight	13.20	90.69



(a) w.o. Evaluation Network (b) Reconstructions

Figure 16: Visualization of robust data generation and InvertingGrad reconstructions when replacing the evaluation network with MSE distance metric.

Impact of α. A higher value of α suggests that the initialized images drift further from the clients' data, enhancing privacy at the potential cost of reduced utility. The results in Table 7 align with this thought where a rise in α corresponds to lower PSNR and PMM values. As analyzed in Section 3.4, when α is set to 0 (without noise-blended initialization), the PSNR surges, indicating high semantic similarity between robust and clients' data. To mitigate adversarial vulnerability, we employ noise-blended initialization, enabling a significant decrease in PSNR with a moderate sacrifice in PMM. For example, by adjusting α from 0 to 0.1, we achieve a substantial reduction in PSNR by approximately 5, while the decline in PMM remains modest, estimated at only around 3.

Impact of τ. τ shapes the alignment between the gradients of client data and robust data. Table 8 reports the results of tuning τ. By including τ into *Refiner* (transitioning from 1 to 0.97), a marginal PMM reduction of 1% is observed while simultaneously securing a considerable 2-point decrease in PSNR. Further increases in τ yield negligible benefits, suggesting that the observed improvements reach a plateau.

Impact of element-wise weight. Table 10 compares the performance of *Refiner* with and without element-wise weight. While the element-wise weight introduces only a small increase in PSNR (from 13.15 to 13.20), it achieves a significant improvement in PMM (from 85.32 to 90.69), demonstrating its critical role in *Refiner*.

Impact of the evaluation network. Figure 16 visualizes robust data generated by *Refiner* and InvertingGrad reconstructions when replacing the evaluation network with MSE distance metric. Compared to Figure 14, the MSE-based *Refiner* focuses on altering specific pixels rather than disrupting global semantic structures. This results in reconstructed images retaining most semantic information from the original data. The gradient distance between the generated robust data and real data remains approximately 0.001, achieving a PMM score of 91.05.

7 Discussion, Conclusion, and Future Work

Why does *Refiner* work? We are interested in why *Refiner* is more effective than perturbation-based methods. In fact, we discover that perturbation-based methods are theoretically optimal under certain assumptions, which, unfortunately, do not often hold for DNNs. To explicate, perturbing each gradient element equally (DP and GQ) assumes that every gradient element carries equal privacy information and contributes equally to the model performance, making uniform perturbation theoretically optimal under this assumption. By considering all gradient elements as possessing identical privacy information, the optimal solution derived from Taylor expansion is the preservation of gradients with large magnitudes, *i.e.*gradient pruning. Soteria improves on gradient pruning by estimating the privacy information of each gradient element based on the *linearity assumption* of DNNs, selectively removing those with the highest privacy information.

Refiner uploads gradients generated by robust data. By reducing semantic similarity between the robust data and clients' data, the uploaded gradients contain less privacy information about clients' data. Importantly, such privacy estimation fashion avoids potential estimation errors that may arise from questionable assumptions made by perturbation-based methods. Regarding utility, *Refiner* prioritizes important parameters and narrows the gap between the gradients of these parameters with respect to robust and clients' data. Section 3.3 validates that preserving the gradient of important parameters is a better solution than retaining gradients with the largest magnitude. These insights explain *Refiner*'s superior performance over perturbation-based defenses.

Conclusion and future work. Overall, we believe that *Refiner* brings a novel idea for privacy protection by shifting the focus from gradient perturbation or cryptographic approaches to data-centric defenses. Our empirical validation also demonstrated that *Refiner* achieves superior privacy-utility tradeoffs compared to existing methods. The main drawback of *Refiner* is its higher computational demand. However, this additional cost is justified by the enhanced performance and is manageable in typical resource-constrained settings like IoT (Appendix O). Looking ahead, we aim to reduce the computational burden.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant number 62202170, Fundamental Research Funds for the Central Universities under grant number YBNLTS2025-017, Alibaba Group through the Alibaba Innovation Research Program, and the Open Research Fund of The State Key Laboratory of Blockchain and Data Security, Zhejiang University.

Ethics Considerations

This paper focuses on defenses against gradient leakage. Our evaluations use only publicly available datasets, ensuring no threats to personal privacy or ethical concerns.

Open Science

See https://zenodo.org/records/15544639 for our implementation.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proc. of ACM CCS*, pages 308–318, 2016.

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *ArXiv*, 2017.

[3] Mislav Balunović, Dimitar I Dimitrov, Robin Staab, and Martin Vechev. Bayesian framework for gradient leakage. In *Proc. of ICLR*, 2022.

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. B. McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proc. of ACM CCS*, 2017.

[5] Stephen P. Boyd and Lieven Vandenberghe. Convex optimization. *Journal of the American Statistical Association*, 100:1097 – 1097, 2005.

[6] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*, 2017.

[7] Yue Cui, Syed Irfan Ali Meerza, Zhuohang Li, Luyang Liu, Jiaxin Zhang, and Jian Liu. Recup-fl: Reconciling utility and privacy in federated learning via user-configurable privacy defense. In *Proc. of AsiaCCS*, 2023.

[8] Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. Tag: Gradient attack on transformer-based language models. In *Findings of EMNLP*, 2021.

[9] Wei Deng, Guang Lin, and Faming Liang. A contour stochastic gradient langevin dynamics algorithm for simulations of multi-modal distributions. In *Proc. of NeurIPS*, 2020.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of ICLR*, 2021.

[11] Jiacheng Du, Jiahui Hu, Zhibo Wang, Peng Sun, Neil Zhenqiang Gong, Kui Ren, and Chun Chen. Sok: On gradient leakage in federated learning. *arXiv preprint arXiv:2404.05403*, 2024.

[12] 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.

[13] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. *Efficient and accurate estimation of lipschitz constants for deep neural networks*. 2019.

[14] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, T. Schneider, Hossein Yalame, and Shaza Zeitouni. Safelearn: Secure aggregation for private federated learning. *2021 IEEE Security and Privacy Workshops (SPW)*, 2021.

[15] Andrew C Gallagher and Tsuhan Chen. Clothing cosegmentation for recognizing people. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[16] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? In *Proc. of NeurIPS*, 2020.

[17] Amirata Ghorbani and James Zou. Neuron shapley: Discovering the responsible neurons. *arXiv preprint arXiv:2002.09815*, 2020.

[18] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of ICLR*, 2014.

[19] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. *Proc. of PR*, 2010.

[20] Tzu-Ming Harry Hsu, Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *ArXiv*, 2019.

[21] Håkon Hukkelås and Frank Lindseth. Does image anonymization impact computer vision training? In *Proc. of CVPR*, pages 140–150, 2023.

[22] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. In *Proc. of NeurIPS*, 2021.

[23] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. 2020.

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

[25] David John Cameron MacKay. Information theory, inference, and learning algorithms. *IEEE Transactions on Information Theory*, 50:2544–2545, 2004.

[26] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*, 2016.

[27] Blaz Meden, Ziga Emersic, Vitomir Struc, and Peter Peer. κ-same-net: neural-network-based face deidentification. In *2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI)*, pages 1–7. IEEE, 2017.

[28] Tristan Milne. Piecewise strong convexity of neural networks. In *Proc. of NeurIPS*, 2019.

[29] Mohammed Ali mnmoustafa. Tiny imagenet, 2017.

[30] Milad Nasr, R. Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. *Proc. of IEEE S&P*, 2018.

[31] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[32] Carman Neustaedter, Saul Greenberg, and Michael Boyle. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(1):1–36, 2006.

[33] Elaine M Newton, Latanya Sweeney, and Bradley Malin. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, 17 (2):232–243, 2005.

[34] Elaine M Newton, Latanya Sweeney, and Bradley Malin. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, 17 (2):232–243, 2005.

[35] Seong Joon Oh, Rodrigo Benenson, Mario Fritz, and Bernt Schiele. Faceless person recognition: Privacy implications in social media. In *Proc. of ECCV*, 2016.

[36] Jaehyoung Park and Hyuk Lim. Privacy-preserving federated learning using homomorphic encryption. *Applied Sciences*, 12(2):734, 2022.

[37] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proc. of ACM CCS*, 2022.

[38] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[39] Tobiasz Rafal, Wilczynski Grzegorz, Graszka Piotr, Czechowski Nikodem, and Luczak Sebastian. Edge devices inference performance comparison. *Journal of Computing Science and Engineering*, 2023.

[40] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *Proc. of AISTATS*, 2019.

[41] Fazlollah M Reza. *An introduction to information theory*. Courier Corporation, 1994.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[43] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proc. of CVPR*, 2021.

[44] Qianru Sun, Ayush Tewari, Weipeng Xu, Mario Fritz, Christian Theobalt, and Bernt Schiele. A hybrid model for identity obfuscation by face replacement. In *Proc. of ECCV*, 2018.

[45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proc. of ICML*, 2017.

[46] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322, 2021. ISSN 0925-2312.

[47] Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[48] Aidmar Wainakh, Till Müßig, Tim Grube, and Max Mühlhäuser. Label leakage from gradients in distributed machine learning. *Proc. of CCNC*, 2021.

[49] 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*, 2024.

[50] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Protect privacy from gradient leakage attack in federated learning. In *Proc. of IEEE INFOCOM*, 2022.

[51] Zhou Wang, Alan Conrad Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[52] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating client privacy leakages in federated learning. In *European Symposium on Research in Computer Security*, 2020.

[53] Wenqi Wei, Ling Liu, Yanzhao Wut, Gong Su, and Arun Iyengar. Gradient-leakage resilient federated learning. In *Proc. of ICDCS*, pages 797–807. IEEE, 2021.

[54] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *Int. J. Mach. Learn. Cybern.*, 14(2):513–535, 2023.

[55] Yunqian Wen, Bo Liu, Ming Ding, Rong Xie, and Li Song. Identitydp: Differential private identification protection for face images. *Neurocomputing*, 2022.

[56] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proc. of CVPR*, pages 16337–16346, 2021.

[57] K. Yue, Richeng Jin, Chau-Wai Wong, Dror Baron, and Huaiyu Dai. Gradient obfuscation gives a false sense of security in federated learning. *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[58] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX Annual Technical Conference*, 2020.

Table 11: The performance and overhead of different metrics.

Metric	Accuracy (%)	Time (s)
Integrated Gradient	49.54	2155.85
Shapley Value	49.50	193672.67
Our Metric	49.55	209.20

[59] Ning Zhang, Manohar Paluri, Yaniv Taigman, Rob Fergus, and Lubomir Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In *Proc. of CVPR*, 2015.

[60] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. of CVPR*, 2018.

[61] Rui Zhang, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. A survey on gradient inversion: Attacks, defenses and future directions. *ArXiv*, 2022.

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

[63] Joshua C. Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *Proc. of IEEE S&P*, 2024.

[64] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Proc. of NeurIPS*, 2019.

A Weight Comparison

Common metrics for parameter importance include Neuron Shapley [17] and Integrated Gradient [45]. While Neuron Shapley theoretically captures neuron contributions to model performance, its prohibitive computational cost limits practicality [17]. Here, parameters of neurons considered insignificant by Neuron Shapley are deemed non-important. We train a LeNet model on the CIFAR10 dataset for 20 epochs (20 epochs, learning rate 0.01, batch size 128). We utilize these two metrics and our metric to differentiate important parameters and subsequently eliminate gradients (80% pruning rate) from neurons deemed non-important. Results in Table 11 show that our metric achieves competitive performance while requiring significantly less computational overhead compared to Neuron Shapley and Integrated Gradient.

B Validation of SSIM and LPIPS

We here demonstrate that SSIM and LPIPS fail to align with semantic similarity, rendering them unsuitable for privacy evaluation. Figures 17(a) and 17(b) apply semantic-preserving transformations (e.g., rotation, flipping, color jitter) to original images, resulting in drastic metric changes: SSIM drops to 0.1 with rotation, and LPIPS increases to 0.3 with color jitter. While these values suggest extreme dissimilarity, the transformed images retain semantic equivalence to their originals. This highlights that SSIM and LPIPS are inappropriate for privacy metrics.

(a) SSIM (b) LPIPS

Figure 17: (a) Rotation and flipping. (b) Color jitter.

C Proof of Privacy Metric

A convex function $g(x)$ can be represented using its conjugate form as follows:

$$\begin{aligned} g(x) &= \max_t \; tx - g(t), \\ t &= g'(x), g(t) = -f(x) + f'(x)x, \end{aligned} \tag{10}$$

where t belongs to the range of $g'(x)$. Equation 10 indicates that the value of $g(x)$ at a certain point can be evaluated by maximizing t. When given an x, there exists a fixed corresponding value for t, which we denote as $t = H(x)$. Consequently, Equation 10 can be rewritten as follows:

$$g(x) = \max_{H(x)} \; H(x)x - g(H(x)), \tag{11}$$

where the range of $H(x)$ belongs to the range of $g'(x)$. The JS divergence of $p(x)$ and $q(x)$ can be written as

$$\begin{aligned} &\frac{1}{2}\int p(x) log \frac{2p(x)}{p(x)+q(x)} + q(x) log \frac{2q(x)}{p(x)+q(x)} dx \\ &= \frac{1}{2}\int p(x)\{log \frac{2}{1+q(x)/p(x)} + \frac{q(x)}{p(x)} log \frac{2q(x)/p(x)}{1+q(x)/p(x)}\} dx. \end{aligned}$$

Since $h(z)$ is a convex function, we can represent JS divergence using its conjugate form as follows:

$$\begin{aligned} &\frac{1}{2}\int p(x) log \frac{2p(x)}{p(x)+q(x)} + q(x) log \frac{2q(x)}{p(x)+q(x)} dx \\ &= \frac{1}{2}\int p(x) h(\frac{q(x)}{p(x)}) dx \\ &= \frac{1}{2}\int p(x)\{\max_{H(\frac{q(x)}{p(x)})} H(\frac{q(x)}{p(x)})\frac{q(x)}{p(x)} - g(H(\frac{q(x)}{p(x)}))\} dx \\ &= \frac{1}{2}\max_{H(\frac{q(x)}{p(x)})} \int q(x) H(\frac{q(x)}{p(x)}) - p(x) g(H(\frac{q(x)}{p(x)})) dx. \end{aligned} \tag{12}$$

Table 12: Total variance (TV) quantifies the smoothness of inputs. L_2-norm controls the pixel value within a legal range $[0 \sim 1]$. BN statistics force the statistics of recovered images to close the statistics of datasets. KL distance restricts the latent variables to follow Gaussian distribution.

Attack	iDLG	InvertingGrad	GradInvertion	GGL
Loss Function	Euclidean	Cosine Similarity	Euclidean	Euclidean
Regularization	None	TV	TV+L_2-norm +BN statistics	GAN+KL Distance
Optimizer	L-BFGS	Adam	Adam	Adam
Learning Rate	1	0.01	0.01	0.01
Label Inference	✓	✓	✓	✓
Attack Iteration	300	4000	4000	4000

According to Equation 11, the corresponding form of $g(\cdot)$ to $h(\cdot)$ is $g(t) = -log(2-e^t)$. Substituting the specific form of $g(t) = -log(2-e^t)$ into Equation 12, we obtain:

$$\frac{1}{2}\max_{H(\frac{q(x)}{p(x)})} \int q(x)H(\frac{q(x)}{p(x)}) - p(x)g(H(\frac{q(x)}{p(x)}))dx$$
$$= \frac{1}{2}\max_{H(\frac{q(x)}{p(x)})} \int q(x)H(\frac{q(x)}{p(x)}) + p(x)log(2-e^{H(\frac{q(x)}{p(x)})})dx.$$

Setting $H(\frac{q(x)}{p(x)}) = log(2D(x))$ and substituting it into the ebove equation, we obtain:

$$\frac{1}{2}\max_{D(x)} \mathbb{E}_{x\sim p(x)}[log(1-D(x))] + \mathbb{E}_{x\sim q(x)}[logD(x)] + log2.$$

Moreover, the definition of conjugate function requires $H(\cdot)$ to be limited to the range of $g'(t)$. Since the range of $g'(t)$ is smaller than $log2$, we can derive that the range of $D(\cdot)$ is constrained to the interval of 0 to 1.

D Detailed Settings

Table 12 summarizes the differences between employed attacks and the hyperparameters used.

The architecture and training settings of the evaluation network. The evaluation network comprises four layers: three convolutional layers (kernel size 3×3, stride 2, padding 1, with ReLU activation) and a fully connected layer (sigmoid activation). The number of filters in the convolutional layers is 128, 256, and 512, respectively. The network predicts noise proportions in inputs and is trained for 20 epochs using Adam optimizer (10^{-4} learning rate, batch size 128). During training, each batch is mixed with random noise 11 times, with $\alpha \in \{0, 0.1, \ldots, 1\}$ before proceeding to the next batch.

Non-IID setting. We generate non-IID datasets by assigning local data distributions via a symmetric Dirichlet distribution with concentration parameter 1. For each client, we sample a distribution over data categories, determine the number of samples per label, and randomly select corresponding samples from the training dataset. This process ensures heterogeneous data distributions across clients, enabling evaluation of defense robustness under non-IID conditions in Section 6.

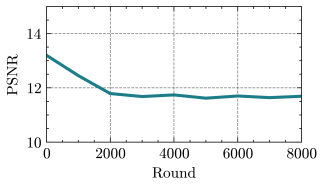


Figure 18: The performance of *Refiner* over different rounds.

E *Refiner* 's Performance over Rounds

Figure 18 presents the performance of *Refiner* against InvertingGrad across varying rounds. As the number of rounds increases, *Refiner* demonstrates better defensive performance. This is aligned with prior findings that prolonged training reduces model sensitivity to input data, thereby making data reconstruction more challenging.

F *Refiner* on Textual Data

Overall idea. For textual data, meaningless data can be considered as sentences composed of random words. We train an evaluation network to predict the likelihood that a given piece of text resembles a sentence made up of random words. Notice NLP models commonly convert discrete tokens into continuous word embeddings. Since discrete data is challenging to optimize directly, we construct robust data for their word embeddings. In this scenario, clients no longer upload gradients for the parameters of the first layer, which is less impactful for NLP models since NLP models rarely fine-tune the initial layer. For data from other modalities, we can define their meaningless data similarly to train evaluation network for *Refiner* execution.

Implementation. In practice, we replace a portion of words in a normal sentence with random words, then feed the resulting embeddings into evaluation network. The network is trained to predict the proportion of random words. We use DistilBERT fine-tuned on IMDb, fine-tuned on IMDb dataset using Huggingface's default configuration[9]. The process of generating robust data for word embedding vectors follows the same way used for generating robust data for images.

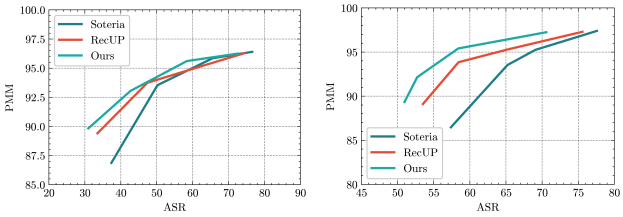
Evaluation. We employ BERT and SST-2 dataset using an IID setup with 10 clients and Huggingface's default configuration[10]. Since Soteria does not apply to textual data, we evaluate *Refiner*'s defensive performance against InvertingGrad and TAG [8], comparing it with DP and Pruning. TAG [8]

[9] https://huggingface.co/lvwerra/distilbert-imdb

[10] https://huggingface.co/gchhablani/bert-base-cased-finetuned-sst2

Table 13: The performance of three defenses against TAG and InvertingGrad.

Attack	Metric	Pruning	DP-Gaussian	*Refiner*
TAG	Model Accuracy (%)	90.7	90.6	90.9
	F1 Score (%)	32.28	30.44	24.55
InvertingGrad	Model Accuracy (%)	90.8	90.5	91.1
	F1 Score (%)	50.09	49.36	35.64



(a) Attribute Inference (b) Membership Inference

Figure 19: The performance of *Refiner* against attribute and membership inference attacks.

is a GLA specific to textual data, and we follow its original paper's hyperparameters. We set pruning, DP, and *Refiner* defenses to strengths of 0.2, 0.001, and $\varepsilon = 0.05$, respectively, as these hyperparameters are found to offer the strongest defense while maintaining model performance. BERT without any defenses achieves an accuracy between 91% to 92% on SST-2. We measure attack performance via precision, recall, and F1 metrics over 1000 reconstructed samples. For utility, we report the model's accuracy on the test set. As shown in Table 13, *Refiner* still achieves better defense performance (lower F1 score) while maintaining model utility.

G The Effectiveness of *Refiner* against Attribute and Membership Inference

Attribute inference attacks try to infer specific attributes from the client's data, while membership inference attacks seek to determine if a specific data point belongs to a client's dataset. The effectiveness of these attacks primarily arises from models' tendency to overfit their training data rather than from generalizable features. *Refiner* promotes semantic divergence between original and robust data during gradient alignment. This can be intuitively understood as the process of removing task-irrelevant information from the data while preserving features essential for the original task. From this standpoint, *Refiner* can effectively mitigate both attribute inference and membership inference attacks. Following the experimental setup in [7], we evaluate *Refiner*'s effectiveness against attribute inference attacks [7] and membership inference attacks [30] on LFW, comparing with Soteria and RecUP [7]. As shown in Figure 19, *Refiner* achieves comparable model utility while significantly reducing the attack success rate (ASR)

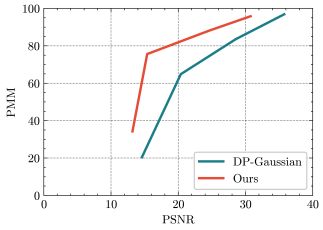


Figure 20: Defense effectiveness of *Refiner* against LOKI [63] on CIFAR-10.

Table 14: The distance between x and corresponding recovered data using four different attack methods.

Attack	iGLA	GradInvertion	InvertingGrad
σ (Euclidean distance)	0.00018	0.00053	0.00046
σ (Our evaluation network)	0.003	0.007	0.005

Table 15: The similarity between x and reconstructed data.

	PSNR	Our evaluation network
iDLG	10.6742	0.9847
InvertingGrad	10.4378	0.9785

across both attack scenarios.

H Evaluation in Malicious Threat Model

We evaluate *Refiner*'s performance under a malicious threat model, where the server is allowed to modify model parameters, as considered in [63]. Following the configuration from [63], we conduct experiments on CIFAR-10 with a batch size of 16 and 4 local training steps per round. As shown in Figure 20, *Refiner* demonstrates superior performance compared to DP against LOKI [63]. Specifically, while maintaining approximately 80% of PMM, *Refiner* achieves a PSNR of around 14, whereas DP achieves a PSNR of 25, underscoring the effectiveness of *Refiner*.

I Proof of Theorem 1

We assume $||\nabla_\theta \mathcal{L}(F_\theta(x), y) - \nabla_\theta \mathcal{L}(F_\theta(x^*), y)|| \leq \varepsilon$. By leveraging Assumption 3, there is:

$$\begin{aligned}
&\mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i), y_i) - g_i^{full}||_2^2 \\
&= \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i), y_i) - \nabla_\theta \mathcal{L}(F_\theta(x^*), y) \\
&+ \nabla_\theta \mathcal{L}(F_\theta(x^*), y) - g_i^{full}||_2^2 \\
&\leq \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i), y_i) - \nabla_\theta \mathcal{L}(F_\theta(x^*), y)||_2^2 \\
&+ \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x^*), y) - g_i^{full}||_2^2 = \sigma_i^2 + \varepsilon^2.
\end{aligned} \tag{13}$$

Based on Assumption 4, we obtain:

$$\begin{aligned}&\mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i^*), y_i)||_2^2 \\ &= \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i^*), y_i) - \nabla_\theta \mathcal{L}(F_\theta(x_i), y_i) + \nabla_\theta \mathcal{L}(F_\theta(x_i), y_i)||_2^2 \\ &\leq \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i^*), y_i) - \nabla_\theta \mathcal{L}(F_\theta(x_i), y_i)||_2^2 \\ &+ \mathbb{E}||\nabla_\theta \mathcal{L}(F_\theta(x_i), y_i)||_2^2 = G^2 + \varepsilon^2.\end{aligned}$$

Applying the above two equations in Theorem 2 in [23] makes: $\mathbb{E}[\mathcal{L}(F_\theta(\cdot),\cdot)] - \mathcal{L}(F_{\theta^*}(\cdot),\cdot) \leq \frac{2\kappa}{\gamma+N}(\frac{Q+C}{u} + \frac{u\gamma}{2}\mathbb{E}||\theta_{initial} - \theta^*||_2^2)$.

J Projection Algorithm

Given the gradient that clients upload, denoted as g_{upload}, our objective is to ensure that its distance from ground-truth gradients $g = \nabla_\theta \mathcal{L}(F_\theta(x), y)$ does not exceed ε:

$$\begin{aligned}&\min ||g_{upload} - g^*||_2^2, \\ &||g_{upload} - g||_2^2 \leq \varepsilon^2, g^* = \nabla_\theta \mathcal{L}(F_\theta(x^*), y).\end{aligned} \tag{14}$$

For this optimization problem, there are two cases. If $||g_{upload} - g||_2^2 \leq \varepsilon^2$, the optimal solution obviously is $g_{upload} = g^*$. Otherwise, it is $g_{upload} = g + \varepsilon \cdot \frac{g^* - g}{||g^* - g||_2}$. In our experiment, clients check for $||g_{upload} - g||_2^2 \leq \varepsilon^2$. If it's satisfied, them upload g^*. Otherwise, it makes slight modifications to g^*, *i.e.*, uploading $g_{upload} = g + \varepsilon \cdot \frac{g^* - g}{||g^* - g||_2}$.

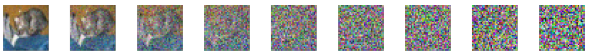
K Evidence Supporting Section 4.2

We here empirically demonstrate that $\sigma \ll \text{dist}(x^*, x)$. When employing Euclidean distance, σ represents the MSE distance between the image x and the data reconstructed from the gradients associated with x. In the context of our evaluation network, σ denotes the noise discrepancy between the original image x and its gradient-based reconstruction. We randomly extract 1000 from the training set of CIFAR10 as x. We utilize LeNet and follow the same configuration as in Section 5. Table 14 reports the results averaged over 1000 samples using four different attack methods. We further construct robust data for these 1000 samples.

The MSE distance and noise difference, averaged over these 1000 samples, between x and their robust data are 0.0839 and 0.6258, significantly larger than σ, thus indicating the effectiveness of privacy protection analysis.

L Evaluation Network as Distance Metric

Our evaluation network quantifies the amount of noise present in given data, and we define a distance metric as $\text{dist}(x, y) = |D(x) - D(y)|$, where $D(\cdot)$ denotes the predicted noise ratio.



PSNR	29.265	23.081	19.558	17.125	15.062	13.516	12.162	10.926	10.224
SSIM	0.9471	0.8178	0.6519	0.5153	0.3569	0.2672	0.1713	0.0884	0.0441
LPIPS	0.0031	0.0099	0.0320	0.0467	0.1042	0.1287	0.1449	0.1902	0.2238
Our	0.0791	0.1825	0.2837	0.3911	0.4967	0.5997	0.7171	0.8189	0.9193

Figure 21: The relationship between different metric values and the degree of privacy leakage in the images.

This definition inherently ensures the non-negativity. Furthermore, there is:

$$\begin{aligned}&dist(x,z) = |D(x) - D(z)| = |D(x) - D(y) + D(y) - D(z)| \\ &\leq |D(x) - D(y)| + |D(y) - D(z)| = dist(x,y) + dist(y,z).\end{aligned}$$

Since our derivation only requires non-negativity and the triangle inequality (identity is not enforced), it qualifies as a valid metric for distance in this paper.

M Further Security Analysis

In Section 4.4, we show that the equation $||g^* - g||_2^2 = const$ is underdetermined. Here, we engage in an empirical analysis. Specifically, we employ gradient optimization algorithm to solve $||g^* - g||_2^2 = const$ and denote the results as g'. We then apply GLAs to reverse g'. Notice that, in practice, we do not omit the weight factors. Table 15 reports the average PSNR between the reconstructed data from g' and x across 1000 samples. As observed, the values of PSNR are typically low, indicating a significant difference between the reconstructed data and x. Thus, *Refiner* is capable of effectively withstanding adaptive attacks.

N Concrete Formula of Evaluation Metrics

Figure 21 illustrates how metric values correlate with privacy leakage. For our evaluation network, a threshold of 0.4 is used to distinguish private vs. non-private outputs. For PSNR, SSIM, and LPIPS, threshold values for safeguarding privacy are 15, 0.5, and 0.05, respectively.

O Overhead of *Refiner* in Edge Device Chips

We evaluate *Refiner*'s computational overhead on four edge devices (Jetson Nano, Coral USB, Neural Stick, Coral PCIe) using ResNet50 with 224×224 input. Based on [39], a single forward pass takes 28.47 ms, 37.09 ms, 38.42 ms, and 44.04 ms, respectively. Assuming backward passes match forward durations and the evaluation network is ResNet50-sized, solving Equation 2 with a batch size of 32 and 10 iterations ($640\times$ forward passes) yields total runtimes of 18.22s, 23.74s, 24.59s, and 28.19s. While non-trivial, these costs remain practical for real-world deployment.