ARES: Scalable and Practical Gradient Inversion Attack in Federated Learning through Activation Recovery

Zirui Gong[1], Leo Yu Zhang✉[1], Yanjun Zhang[1], Viet Vo[2], Tianqing Zhu[3], Shirui Pan[1], Cong Wang[4]
[1] *Griffith University*
[2] *Swinburne University of Technology*
[3] *City University of Macau*
[4] *City University of Hong Kong*

***Abstract*—Federated Learning (FL) enables collaborative model training by sharing model updates instead of raw data, aiming to protect user privacy. However, recent studies reveal that these shared updates can inadvertently leak sensitive training data through gradient inversion attacks (GIAs). Among them, active GIAs are particularly powerful, enabling high-fidelity reconstruction of individual samples even under large batch sizes. Nevertheless, existing approaches often require architectural modifications, which limit their practical applicability. In this work, we bridge this gap by introducing the Activation REcovery via Sparse inversion (ARES) attack, an active GIA designed to reconstruct training samples from large training batches without requiring architectural modifications. Specifically, we formulate the recovery problem as a noisy sparse recovery task and solve it using the generalized Least Absolute Shrinkage and Selection Operator (Lasso). To extend the attack to multi-sample recovery, ARES incorporates the imprint method to disentangle activations, enabling scalable per-sample reconstruction. We further establish the expected recovery rate and derive an upper bound on the reconstruction error, providing theoretical guarantees for the ARES attack. Extensive experiments on CNNs and MLPs demonstrate that ARES achieves high-fidelity reconstruction across diverse datasets, significantly outperforming prior GIAs under large batch sizes and realistic FL settings. Our results highlight that intermediate activations pose a serious and underestimated privacy risk in FL, underscoring the urgent need for stronger defenses.**

1. Introduction

Federated Learning (FL) [9] is a decentralized paradigm for training machine learning models, wherein multiple clients collaboratively optimize a shared global model without disclosing their private data. In this framework, each client preserves its local dataset and performs model training independently. The locally computed updates or gradients are then transmitted to a central server, which aggregates them to refine the global model and subsequently redistributes it to the clients for the next training round. FL has gained great popularity and has been widely adopted in healthcare and finance, as it enables the training of machine learning models on large-scale datasets without exposing clients' raw data to the central server [10]–[14].

However, recent studies demonstrate that FL can provide a false sense of privacy, as the server can extract sensitive information, including the clients' private training data, from the shared updates. Such threats, referred to as *gradient inversion attacks* (GIA) [1]–[8], [15], [16], can be broadly classified into two categories, as summarized in Table 1. In the passive setting, an honest-but-curious server attempts to reconstruct training samples by iteratively minimizing the discrepancy between the gradients of dummy samples and the observed gradients of the true data [1]–[4], [15]. While conceptually straightforward, these approaches often fail to generalize to complex datasets and suffer substantial performance degradation when the training batch size increases (usually limited to batch sizes smaller than 64).

In contrast, active attacks assume a malicious server that manipulates the model parameters or architecture to amplify privacy leakage [5]–[8]. A commonly used strategy is the *linear layer leakage*, where if only a single sample x activates neuron i in a fully connected (FC) layer, the sample x can be directly revealed by dividing the weight gradient by the bias gradient of neuron i. Building on this principle, the adversary designs malicious parameters to disentangle individual sample contributions across neurons, ensuring that each neuron's recovery corresponds to a distinct training sample. Despite their effectiveness, these attacks suffer from notable practical limitations, as they typically require modifications to the model architectures. First, linear layer leakage can only reconstruct the direct inputs to an FC layer. Consequently, prior attacks typically insert a specially designed FC layer before the target model to enable reconstruction [6], [8], resulting in a nonstandard architecture that may raise suspicion. Second, even for those networks that have an FC layer in the first layer (e.g., MLP), achieving high recovery rates demands the number of output neurons in the FL layer exceed the batch size, with about four times larger being optimal [8]. This requirement forces adjustments to the network dimensions relative to the batch size, increasing the extent of structural modification and

✉ *Correspondence to Leo Yu Zhang (leo.zhang@griffith.edu.au).*

TABLE 1: Comparison of different GIAs. ● supported; ❍ not supported; ◑ partially supported.

Methods	Attack Type	Large Batch	Architecture Integrity	One-shot	Complex Data	Theoretical Guarantee
iDLG [1]	Passive	❍	●	●	❍	❍
InvertingGrad (NeurIPS 2020) [2]	Passive	❍	●	●	❍	❍
GradInversion (CVPR 2021) [3]	Passive	❍	●	●	❍	❍
FedLeak (USENIX 2025) [4]	Passive	❍	●	●	●	❍
Fishing (ICML 2023) [5]	Active	◑*	●	❍	●	❍
RtF (ICLR 2022) [6]	Active	●	❍	●	●	●
Trap Weight (EuroS&P 2023) [7]	Active	●	◑†	●	●	❍
LOKI (S&P 2024) [8]	Active	●	❍	●	●	●
ARES (Ours)	Active	●	●	●	●	●

* Fishing [5] can only recover a single image from a batch. † Trap Weight [7] requires the input to be positive to preserve architecture integrity.

reducing the attacks' practicality.

Our work: We bridge this gap and propose the Activation REcovery via Sparse inversion (ARES) attack, a practical and effective active GIA that scales to realistic large batch sizes without necessitating architectural modifications, thereby enhancing its applicability to real-world scenarios. To achieve this, we first reveal that the practical limitations of prior active GIAs can be overcome by tackling the fundamental challenge of inverting hidden activations into training samples. Even without architectural modifications, the attacker can still leverage existing FC layers—typically located in deeper parts of the network—to obtain activations that are fed into them. These activations can then be further inverted to recover the corresponding training samples.

Motivated by this observation, this work focuses on the inversion problem from activations to training samples. To tackle this, we first formulate the operations before the FC layer as a linear transformation followed by a nonlinear transformation (caused by activation functions). Under this formulation, we identify two key challenges that hinder recovery. First, the nonlinear activations in earlier layers render the overall mapping non-invertible, thereby precluding exact recovery of the training sample through a direct inverse transformation. Second, the linear transformation is often underdetermined (i.e., the number of unknowns exceeds the number of known measurements), particularly in MLP-based architectures. For example, sample features typically lie in a high-dimensional space (e.g., 14,784 for ImageNet), while the activation in the FC layer resides in a much lower-dimensional space, equal to the number of output neurons. Even in CNN-based networks, ReLU activations before the FC layer may zero out many measurements, discarding information that could aid recovery. Consequently, the available measurements are insufficient compared to the dimensionality of the target sample, making the reconstruction of training data from activations an *underdetermined nonlinear inversion problem*.

To address this problem, we relax the inversion task and reformulate it as a *noisy sparse recovery problem*, and use the knowledge from compressed sensing theory to solve it [17]–[21]. Specifically, to mitigate the challenges posed by nonlinearity, we approximate the nonlinear transformation as a noisy, scaled linear mapping. To further address the underdetermined nature of the problem, we exploit the fact that many types of data (e.g., natural images, text embeddings, and audio signals) admit sparse representations in suitable domains. In other words, such data can be effectively compressed into a vector with most entries being zero, reducing the number of unknowns to recover. Based on these observations, we reformulate the problem as a noisy sparse recovery task and employ the generalized Least Absolute Shrinkage and Selection Operator (Lasso) method [20] to identify the sparsest solution consistent with the observed measurements. To extend recovery from single-sample to a batch of samples, we integrate the *imprint method* proposed by RtF [6]. In particular, we leverage the bias of the FC layer as cut-offs so that different samples activate different neurons. As a result, each neuron primarily captures the contribution of a single sample, enabling linear layer leakage to recover the corresponding activation. Here, we do not require the number of neurons in the FC layer to exceed the batch size, as we allow the second layer separation. Finally, by invoking the recovery guarantees provided by the Restricted Isometry Property (RIP) [19], we establish a theoretical upper bound on the reconstruction error.

We evaluate our attack across five image datasets, including MNIST [22], CIFAR-10 [23], ImageNet [24], HAM10000 [25], Lung-Colon Cancer [26], one text dataset (Wikitext [27]), and one audio dataset (AudioMNIST [28]) using representative CNN and MLP architectures. Our results demonstrate that our ARES consistently outperforms all state-of-the-art attacks, achieving up to 7 $\times$ improvement in PSNR across various datasets and batch sizes. Furthermore, we assess the robustness of our attack under five defense strategies, including differential privacy (DP) [29], gradient quantization [30], gradient sparsification [30], data augmentation [31] and secure aggregation [32], [33], showing that ARES remains effective in these protected settings. Our key contributions can be summarized as follows:

- We reveal the practical limitation of the existing active GIA lies in the unsolved challenge of inverting hidden activations into training samples. Based on the observation, we formulate the inversion task as a noisy sparse recovery problem and leverage principles from compressed sensing to solve it.
- We propose ARES, a practical and effective active GIA that scales to realistic large batch sizes without requiring architectural modifications. ARES

achieves this by exploiting linear layer leakage to extract intermediate activations and leveraging sparse recovery techniques to reconstruct the training samples from the extracted activations.
- We provide a theoretical upper bound on recovery error and conduct extensive experiments[1] on image, text, and audio datasets, demonstrating that ARES consistently outperforms state-of-the-art attacks by up to 7 $\times$ in PSNR across different settings.

2. Preliminary

2.1. Gradient Inversion Attacks

Passive GIAs [1]–[4], [15], [34] assume an honest-but-curious server or an external adversary with access to the model and individual gradients from each client. The attacker tries to minimize the difference between the observed ground-truth gradient and the gradient generated by the dummy sample, thereby optimizing the dummy sample to approximate the original input. Formally, the reconstruction can be formulated as

$$\tilde{x} = \arg\min_{x} \|\nabla\mathcal{L}(x) - g\|^2, \tag{1}$$

where x is the dummy sample, $\tilde{x}$ is the reconstructed sample, $\mathcal{L}$ is the loss function, and g is the observed gradient. Recent works enhance this optimization by incorporating various regularizers, such as total variation (TV) [2], or some image priors tailored to natural image distributions [3], to improve visual fidelity. These methods often yield good reconstruction results only when the batch size is small and the dataset is relatively simple. However, as the batch size increases, gradient contributions from different samples become entangled, making the optimization landscape more complex and the reconstruction less accurate.

By contrast, active GIAs assume a malicious server that can modify the model parameters or architecture to launch a stronger attack [5]–[8]. A commonly used strategy is the *linear layer leakage*, where if only a *single* sample x activates neuron i in an FC layer, the sample x can be directly revealed by solving

$$x = \frac{\partial\mathcal{L}}{\partial W_i} \Big/ \frac{\partial\mathcal{L}}{\partial b_i}, \tag{2}$$

where $\frac{\partial\mathcal{L}}{\partial W_i}$ and $\frac{\partial\mathcal{L}}{\partial b_i}$ are the gradients of the loss $\mathcal{L}$ with respect to the weight and bias for neuron i, respectively. To recover a batch of samples, RtF [6] introduces the imprint method, which encourages each sample in the batch to leave a distinct imprint on a specific neuron, thereby enabling each neuron to be reverted to reveal individual samples. However, this approach only enables the recovery of inputs fed directly into the FC layer. Consequently, reconstructing the original training samples requires placing the imprint module (the specially designed FC layer) at the beginning of the target model, leading to a non-standard architecture that could raise suspicion on the client side.

Trap Weight [7] attempts to overcome this limitation by leveraging the existing FC layer within the network for training sample reconstruction, thereby eliminating the need for architectural modifications. It initializes the weight matrices of all layers preceding the FC layer to act as direct-pass (identity) mappings, allowing inputs to propagate through unchanged to the FC layer. Then, the linear layer leakage can directly reveal the training samples. LOKI [8] extends this idea and proposes an attack targeting secure aggregation–based FL. In this setup, each client receives the model with distinct parameter configurations, where a subset of kernels (e.g., three) is set as direct-pass mappings and the remaining kernels are set to zero. This client-specific configuration prevents weight gradient from mixing across clients, thereby enabling large-scale recovery. However, both methods are effective only when the network processes nonnegative inputs. Under standard settings, where inputs are normalized to follow $\mathcal{N}(0, 1)$, ReLU activations in the preceding layers suppress negative values, thereby breaking the intended identity mapping and leading to information loss. Scale-MIA [35] also leverages the model's built-in FC layer to conduct an attack without modifying the architecture. However, it requires an auxiliary dataset (i.e., a subset of the training dataset) to train a decoder that maps latent representations back to the original samples, which limits its ability to generalize to unseen domains. Detailed descriptions of attacks mentioned in Table 1 are provided in Appendix B.2.

2.2. Defenses Against Gradient Inversion Attacks

Defenses against GIA can be classified into three main categories: gradient perturbation-based methods [29], [30], data augmentation-based defense [31], [36], and secure aggregation-based methods [32], [37]. Gradient perturbation-based methods modify gradients sent to the server to avoid directly leaking training sample-related information to the server. Differential privacy (DP) [29] perturbs ground-truth gradients by adding random noise. Gradient sparsification [30], [38] transmits only the most significant gradient elements, while gradient quantization [30] reduces precision by representing gradient values with fewer bits. Although effective, these strategies incur a trade-off between model utility and privacy, as more substantial modifications yield better protection but degrade the gradient utility.

Data augmentation–based defenses [31], [36] apply carefully chosen transformations to the training data to prevent adversaries from reconstructing both the augmented and original samples from shared gradients, while preserving model utility. The key idea is to disrupt the prior knowledge exploited by attackers, i.e., total variation or batch normalization statistics, that guides the reconstruction process.

Secure aggregation-based methods protect user training data by ensuring that the server can only access the plaintext of the aggregated gradients [32], [33]. This makes data reconstruction significantly more challenging, as the

1. Our code is available at **https://github.com/gongzir1/ARES**.

aggregated gradients correspond to a larger global batch size that must be recovered. Detailed descriptions of defenses are provided in Appendix B.3.

2.3. Threat Model and Attack Scope

Threat Model. We consider a *malicious server* that controls the FL training process and can modify the weights and biases of the global model before distributing them to clients. Unlike prior works [6], [8], the server cannot alter the network architecture or design a non-standard model to facilitate an attack. This assumption is realistic because, in FL, the model architecture and training protocol are typically agreed upon in advance, and unsolicited architectural changes are generally rejected or detected. Given that clients depend on the server for model distribution and without insight into its internal operations, it is reasonable to treat the server as potentially malicious [5], [35], [39], [40]. Such an attack can also be executed by any party that obtains the server's state, e.g., through a temporary breach [40]. The adversary's objective is to reconstruct as many distinct training examples as possible.

Attack Scope. We evaluate attacks on two widely used families of models: (i) CNN-based networks, consisting of convolutional layers followed by fully connected layers, and (ii) MLP-based networks, including fully connected layers. We also assume that the clients' private training data admits a sparse representation in a suitable domain. This is reasonable, as most real-world data (e.g., natural images, text embeddings, and audio signals) can be sparsely represented in suitable domains [41]–[43].

3. Method

3.1. Motivation

Existing active gradient inversion attacks (aGIAs) exploit linear layer leakage to analytically reconstruct training samples from the gradients of an FC layer. By configuring malicious parameters, the attacker aims to have each neuron activated by a single sample (or to imprint each neuron with a single sample), so that the recovery from each neuron directly reveals individual samples [5]–[8]. Compared to passive GIAs, aGIAs induce stronger privacy breaches and are more effective at recovering samples from large training batches. Despite their effectiveness, a major criticism is their reliance on modifying the network architecture, which mainly stems from two factors. First, linear layer leakage can only reveal the direct inputs to the FC layer. Thus, recovering the original training samples requires the FC layer to be the first layer of the model. This condition is not satisfied in most modern architectures, such as CNNs, which contain multiple convolutional layers before the FC layer. To address this, Trap Weight [7] proposes initializing the layer preceding the FC layer with a direct-pass (identity-like) weight matrix to avoid value distortion, thereby aligning the FC-layer inputs with the model's original inputs. However, this approach works only for nonnegative inputs, as the ReLU activation zeros out negative values and results in information loss. Consequently, existing works typically insert a specially designed FC layer before the target network to enable the training sample recovery [6], [8]. Second, even for models where an FC layer is already the first layer (e.g., multi-layer perceptron–based networks), achieving a high recovery rate requires the number of output neurons to exceed the batch size [6], [8], and in practice, four times larger than the batch size is optimal [8]. Therefore, the attacker must modify the network's dimensionality to satisfy this requirement.

However, we observe that both constraints can be overcome by addressing the fundamental challenge of inverting activations into training samples. For the first constraint, even without model modification, we can leverage the built-in FC layer in standard networks to extract the individual activations that are fed into it. For instance, consider a layer $l > 1$ that is an FC layer in the network. If only a single sample activates neuron i, the corresponding activation $h^{(l-1)}$ associated with neuron i can be reconstructed as

$$h^{(l-1)} = \frac{\partial \mathcal{L}}{\partial W_i^{(l)}} \Big/ \frac{\partial \mathcal{L}}{\partial b_i^{(l)}}, \tag{3}$$

where $\frac{\partial \mathcal{L}}{\partial W_i^{(l)}}$ and $\frac{\partial \mathcal{L}}{\partial b_i^{(l)}}$ are the weight gradient and bias gradient of neuron i in layer l (see Appendix A.1 for a detailed derivation). Although feasible, the unresolved challenge lies in reconstructing the training samples from recovered activations. For the second constraint, which mandates a large number of output neurons in the FC layer, this limitation can be mitigated by employing a subsequent FC layer to further disentangle samples that remain mixed in the first layer. In this case, it still need to address the challenge of reconstructing the training samples from activations (separated in the second layer).

Motivated by the above observations, in this work, we focus on the problem of reconstructing training samples from hidden activations. To start with, we formalize the computation prior to layer l as

$$h^{(l-1)} = f(Wx + b), \tag{4}$$

where f denotes a nonlinear transformation (caused by activation functions), and W and b represent the effective weight and bias of the linear transformation (i.e., weight and bias of the first layer), respectively. While this formulation is straightforward, inverting it to recover x is far from trivial for two main reasons.

Challenges: First, the nonlinear activations in preceding layers render the overall transformation non-invertible, preventing exact recovery of x via a direct inverse mapping. Second, the linear transformation in Eq. (4) is often underdetermined, particularly in MLP-based networks. Specifically, sample features typically lie in a high-dimensional space (e.g., 14,784 for ImageNet), whereas the activation of the FC layer resides in a much lower-dimensional space. Even in CNN-based networks, ReLU activations preceding the FC layer may zero out many measurements, which discards

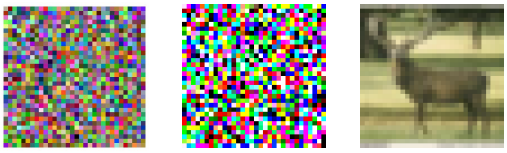


Figure 1: Recovery results obtained using activation matching (optimizing activation discrepancy) and pseudoinverse (approximating inverse).

the information that could aid recovery. As a result, the number of available measurements is insufficient relative to the dimensionality of the target sample. Consequently, recovering x from $h^{(l-1)}$ constitutes an *underdetermined nonlinear inversion problem.*

Solving this problem is inherently challenging. One possible approach is to optimize the discrepancy between ground-truth activations and the dummy activations to optimize the dummy samples that best match the original; we refer to this approach as *activation matching*. However, unlike traditional gradient matching, which leverages gradients from all layers to guide the optimization, this method relies solely on activations from a single layer. As a result, the recovery process is highly ill-posed and often yields a suboptimal reconstruction effect (as shown in the left panel of Fig. 1). Another approach involves using the Moore–Penrose pseudoinverse of the weight matrix to approximate the inverse of W, and only using the linear part of the activation function (i.e., the region where ReLU is active and the output equals the input) to calculate the value of x. Nevertheless, because the system remains underdetermined, the solution is not unique; therefore, this method also fails to reconstruct the original samples faithfully (as shown in the middle panel of Fig. 1).

3.2. Overview

Based on the above observations, this work focuses on addressing the underdetermined nonlinear inversion problem to recover training samples from activations, thereby enabling an effective and practical GIA. To achieve this, the malicious server (hereafter referred to as the attacker) operates in two main stages: preparation and inference, as illustrated in Fig. 2. During the preparation stage, the attacker configures the network parameters to maximize information leakage. Specifically, for the FC layer, it designs malicious weights and biases such that each neuron leaves a distinct imprint corresponding to a single sample, effectively isolating one sample per neuron. For layers preceding the FC layer (if any), the attacker sets the weight parameters to provide sufficient and non-redundant measurements, facilitating accurate reconstruction of training samples from recovered activations. Once configured, the attacker sends malicious parameters to clients for local training.

After clients complete local training and return their updates to the server, the attacker proceeds to the inference stage. The attacker first leverages linear layer leakage by using the gradients of the FC layer to recover either individual training samples or the activations fed into the FC layer. If the recovery directly yields the training samples, they are retained; otherwise, the attacker reconstructs the samples from the recovered activations. To handle the underdetermined and nonlinear nature of this reconstruction task, the attacker reformulates it as a noisy sparse recovery problem and solves it using the generalized Lasso method [20]. By leveraging the recovery guarantees provided by the Restricted Isometry Property (RIP) [19], we further derive a theoretical upper bound on the recovery error.

In the following sections, we first provide an explanation of how to recover a single training sample from its activation under underdetermined and nonlinear transformations (Section 3.3). We then extend this approach to enable the recovery of batches of samples (Section 3.4). Finally, we present attack implementations for CNN and MLP networks and analyze the expected recovery rate (Section 3.5).

3.3. Noisy Sparse Recovery

In this section, we focus on recovering a single training sample from activation under underdetermined and nonlinear measurement. To make the problem tractable, we first relax the nonlinear mapping as a noisy and scaled linear mapping, which is a commonly used technique in the literature [20], [44], [45]. Specifically, we assume $f_i(\xi) = \mu\xi + z_i$, where μ is a scaling factor that captures the linear component of the f and $z_i \sim \mathcal{N}(0, \sigma^2)$ is the noise term. To quantify the linear component and nonlinearity of the function, we provide the following definition.

Definition 1 (Linear Component and Nonlinearity of a Function [20])**.** *Let $f : \mathbb{R}^d \to \mathbb{R}^k$ be a nonlinear function, and let $\xi \sim \mathcal{N}(0, 1)$ be a standard Gaussian random variable. The effective linear component of f is defined as*

$$\mu := \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[f_i(\xi)\xi], \tag{5}$$

which represents the average linear component of f across k dimensions. The residual nonlinearity is quantified by

$$\sigma^2 := \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[(f_i(\xi) - \mu\xi)^2], \quad \eta^2 := \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[(f_i(\xi) - \mu\xi)^2 \xi^2], \tag{6}$$

which measures the variance of the nonlinear part and how it interacts with the input magnitude, respectively.

Under this approximation, Eq. (4) becomes[2]

$$h \approx \mu(Wx + b), \tag{7}$$

and the recovery problem reduces to a noisy, underdetermined linear inversion problem. To further handle the underdetermined issue, we leverage the fact that many types

2. We omit the layer index for clarity; unless explicitly stated otherwise, we assume layer l is the FC layer and h is the input to layer l.

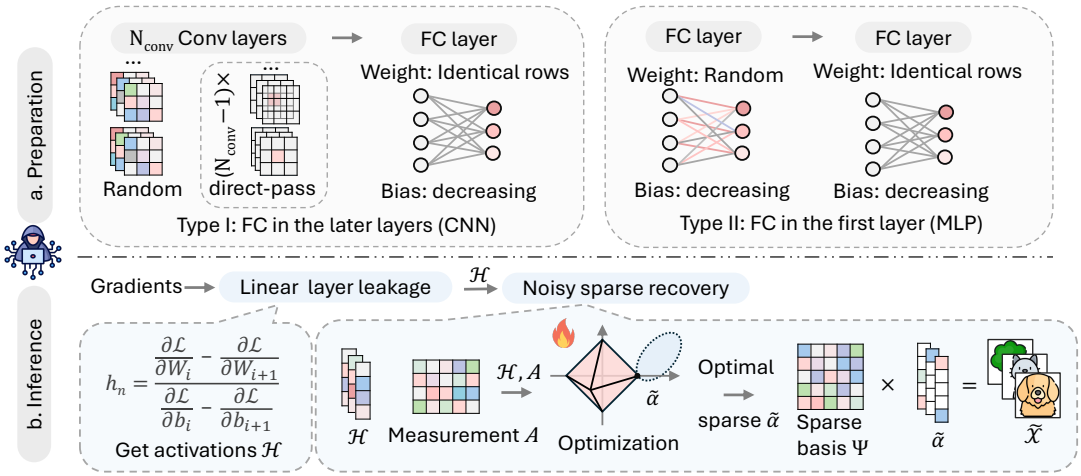


Figure 2: Overview of ARES attack. The method consists of two main stages: (a) the attacker initializes network with malicious parameters to facilitate information leakage; (b) using gradients returned by the client, the attacker first recovers activations through linear layer leakage and then reconstructs input samples via noisy sparse recovery.

of data (e.g., natural images, text embeddings, and audio signals) admit sparse representations in suitable domains [42], [43]. In other words, these data can be effectively compressed into vectors with most entries being zero, thereby reducing the number of unknowns to be recovered. Under this condition, we express $x = \Psi\alpha$, where α is a sparse coefficient vector that encodes the essential information of x, with most of its entries being zero and Ψ is a sparse basis matrix that maps the sparse representation α to the original signal space. To obtain the sparse basis Ψ for the image and audio data, we apply Discrete Cosine Transform (DCT) compression, providing a data-independent basis that efficiently represents samples in the frequency domain. For the text dataset, we learn a sparse basis from the public token embeddings matrix of the pretrained model, which more effectively captures the underlying sparse structure of the token representations. Then, Eq. (7) can be expressed as

$$h \approx \mu A\alpha + \mu b, \tag{8}$$

where $A = W\Psi$ denotes the sensing matrix that acts directly on the sparse vector α (see top of Fig. 3 for illustration). This transformation enforces sparsity on the variable to be recovered, effectively reducing its degrees of freedom.

Nonetheless, achieving an exact and unique recovery of α from h in Eq. (8) further requires the measurement A to possess sufficient information. In particular, it should provide enough independent measurements and preserve the relative geometry among all sparse vectors, such that different sparse vectors yield distinct outcomes under the mapping A. Formally, this requirement is characterized by the Restricted Isometry Property (RIP) [19].

Definition 2 (Restricted Isometry Property [19])**.** *A matrix A is said to satisfy the Restricted Isometry Property (RIP) of order s with constant $\delta_s \in (0,1)$ if, for all s-sparse vectors α (i.e., vectors with at most s non-zero entries),*

$$(1-\delta_s)\|\alpha\|_2^2 \le \|A\alpha\|_2^2 \le (1+\delta_s)\|\alpha\|_2^2. \tag{9}$$

Here, δ_s is the restricted isometry constant *that quantifies how A preserves the Euclidean norm of all s-sparse vectors.*

Eq. (9) means that multiplying a sparse vector α by A changes its Euclidean norm by at most a factor of $(1 \pm \delta_s)$. A smaller δ_s indicates better preservation of the Euclidean norm of the sparse vector.

Remark I. If a matrix A satisfies the *Restricted Isometry Property* (RIP) of order s, it approximately preserves the Euclidean norm of all s-sparse vectors. Moreover, if A satisfies the RIP of order $2s$, it also approximately preserves the pairwise Euclidean distances between all s-sparse vectors, since the difference between any two s-sparse vectors is at most $2s$-sparse. In other words, the measurement $A\alpha$ preserves the geometric relationships among sparse signals, ensuring that distinct sparse inputs remain distinguishable after transformation. Consequently, the information contained in $A\alpha$ is sufficient to enable exact recovery of the sparse vector α in the noiseless case, and stable recovery with small error when noise is present.

We now consider how to configure the malicious parameters such that the resulting measurement matrix A satisfies the RIP, thereby enabling the exact recovery of α from h. This can be achieved by initializing the weight W as a Gaussian random matrix, since the product of a Gaussian random matrix and a sparse orthonormal basis Ψ satisfies the RIP with high probability [46]. Building on the RIP condition, we focus on recovering the sparse vector α. Intuitively, our goal is to find the sparsest solution that satisfies Eq. (8). Formally, this can be expressed as solving

$$\tilde{\alpha} = \arg\min_{\alpha} \|\alpha\|_0 + \|h - \mu A\alpha - \mu b\|_2, \tag{10}$$

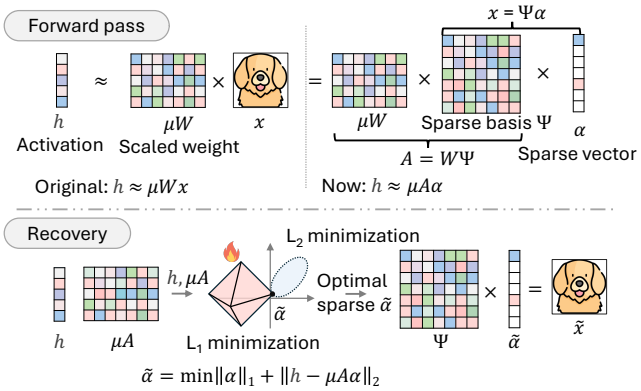


Figure 3: Top: forward pass through the network. Bottom: sparse vector recovery via ℓ_1 optimization. The bias term is omitted for clarity.

where the first term denotes the number of non-zero entries in α and the second term quantifies the discrepancy between the observed activation h and the activation reconstructed from the sparse vector α. Since solving Eq. (10) is NP-hard, a common relaxation is to replace the ℓ_0 norm with the convex ℓ_1 norm,

$$\tilde{\alpha} = \arg\min_{\alpha} \|\alpha\|_1 + \|h - \mu A\alpha - \mu b\|_2, \tag{11}$$

where $\|\alpha\|_1$ promotes sparsity while keeping the optimization tractable. We solve Eq. (11) as a convex Lasso problem using the CVXPY framework [47], which dispatches the underlying solver to compute the solution. This procedure recovers the sparsest vector α consistent with the observed activations. Once $\tilde{\alpha}$ is obtained , it can be mapped back to the input space via $\tilde{x} = \Psi\tilde{\alpha}$ (see bottom of Fig. 3 for illustration). We now provide an upper bound on the recovery error of the solution to Eq. (11).

Theorem 1 (Recovery Error [20])**.** *Let α be an s-sparse vector, and let A be a measurement matrix satisfying the RIP of order $2s$. Then, the solution $\tilde{\alpha}$ to Eq. (11) recovers α with the error bounded by*

$$\varepsilon \sim \frac{\sqrt{s\log(d/s)}\,\sigma + \eta}{\sqrt{m}} + |\mu - 1|\|\alpha\|_2, \tag{12}$$

where s is the sparsity of α, d is the ambient dimension of α, $s\log(d/s)$ quantifies the effective dimension of the sparse signal, and m is the number of effective measurements (i.e., the number of non-redundant rows of A).

Proof. Theorem 1 follows from Theorem 1.4 of [20], which provides an upper bound for $\|\tilde{\alpha} - \mu\alpha\|_2$. We then apply the triangle inequality to yield the upper bound for $\|\tilde{\alpha} - \alpha\|_2$ as stated in Eq. (12). □

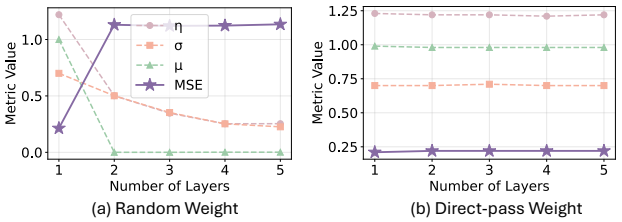


Figure 4: Upper bound of the squared recovery error with varying numbers of convolutional layers.

Remark II. The first term in Eq. (12) captures the combined effect of the signal's sparsity and dimension, measurement noise, and residual nonlinearity on recovery error. It indicates that the recovery error ε decreases as the number of measurements m increases and ε increases with the effective dimension of the sparse signal increase. Here, $s\log(d/s)$ reflects signal complexity, and sparser (smaller s) or lower-dimensional signals (smaller d) have smaller $s\log(d/s)$. Furthermore, σ and η capture the effect of the nonlinear function f on recovery. Larger values of σ or η correspond to higher variance or stronger nonlinearity of f, increasing recovery error. The second term in Eq. (12) quantifies the error due to potential mismatch in scaling between $\mu\alpha$ and the true signal α.

Theorem 1 indicates that, for a fixed number of measurements m, input dimension d, and sparsity s, the factors influencing recovery error are the nonlinear characteristics of the function f, i.e., μ, σ, and η, as defined in Definition 1. Building on this observation, we further examine how the network architecture, particularly the number of layers preceding the FC layer, influences these parameters. As illustrated in Fig. 4, which shows the empirical nonlinear values of f, alongside the squared recovery error (MSE) computed from Eq. (12), when the weights of each layer are randomly initialized (left in Fig. 4), a network with only a single layer before the FC layer exhibits $\mu \approx 1$, indicating that the linear component of f is reasonably captured and consequently the upper bound of the MSE remains acceptable (0.25). However, from two layers onward, μ approaches zero, reflecting the increasingly nonlinear nature of f (due to the activation functions). Meanwhile, the nonlinear residual decreases as the overall variance of f diminishes with depth (due to ReLU masking). As a result, linear approximations become less accurate, causing an increase in the upper bound of the recovery error.

To mitigate this issue and enable effective recovery in deeper networks, we manipulate the weights of convolutional layers to control the nonlinearity of the function f. Specifically, we observe that a single activation function preserves acceptable linearity, but adding more layers with activation functions significantly increases nonlinearity, making the function harder to invert. To prevent this

accumulation, we adjust the convolutional weights *from the second layer onward* using the direct-pass initialization method, so that these layers direct pass the inputs unchanged to the activation function. This ensures that all activation functions in the network operate on the same input, producing a consistent output rather than compounding nonlinear effects across layers. Practically, we implement direct-pass by setting the central element of each input channel to 1 and all other elements to 0, creating an identity-like mapping that preserves the input through the convolutional operation. As shown in Fig. 4 (right), the parameter μ remains stable across all layers under the direct-pass initialization, indicating that the network's linear component is consistently preserved. Consequently, the upper bound of MSE remains stable across deeper layers, indicating the effectiveness of our method on deeper networks.

3.4. Multiple Samples Recovery

In this section, we extend our recovery method to a batch of samples. The objective function in Eq. (11) addresses the problem of recovering a single input sample x from the observed activation h. However, when a batch of N samples is propagated through the network, the activation of neuron i in an FC layer reflects a weighted combination of contributions from all samples that activate neuron i. Consequently, Eq. (3) is modified as

$$\frac{g_i^{(W)}}{g_i^{(b)}} = \frac{\sum_{n=1}^{N_i} \gamma_i^n h_n}{\sum_{n=1}^{N_i} \gamma_i^n}, \tag{13}$$

where $g_i^{(W)}$ and $g_i^{(b)}$ denotes the weight and bias gradient of neuron i, respectively; γ_i^n is the backpropagated error for neuron i in sample n, h_n is the activation of sample n in layer $l-1$ (i.e., the input to layer l), and N_i is the number of samples that activate neuron i (see Appendix A.1 for detailed derivation). Given k output neurons, we obtain k such equations (i.e., Eq. (13)). However, the total number of unknowns is $N(k+1)$, as each of N samples contributes one activation variable h_n and k backpropagation error γ_i^n. Consequently, recovering all variables from these equations alone is impossible. Fortunately, our objective is not to solve for all unknowns but only to recover the activations. To achieve this, we adopt the *imprint method* [6], which enforces a structured activation pattern in the FC layer such that each neuron uniquely corresponds to the activation of a single sample. Specifically, we configure the weight matrix in the FC layer to have identical rows (i.e., $W_i^{(l)} = w^{(l)}$ for all i, where $w^{(l)}$ is a constant vector), so each output neuron receives the same weighted combination of inputs. Based on the distribution of $w^{(l)}h$, we adjust the biases to partition the input space into k bins, maximizing the likelihood that each activation falls into a distinct bin.

As shown in Fig. 5, to divide the input space into k equal mass bins, we first derive the probability density function (PDF) of $w^{(l)}h$. Here, no prior knowledge of the dataset is required; we simply assume that the network input is normalized to follow a standard normal distribution, i.e.,

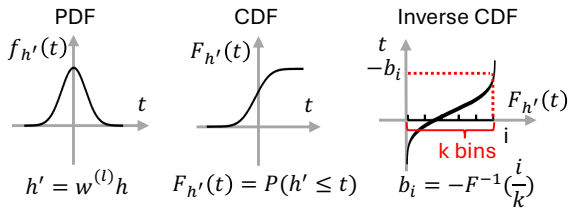


Figure 5: Bias values are set to divide the projected inputs into k equal-probability bins.

$x \sim \mathcal{N}(0,1)$, which is a reasonable assumption for most training setups. Given the value of malicious weight, we can then obtain the corresponding PDF of $w^{(l)}h$. Then we compute its cumulative distribution function (CDF) and partition it into k intervals of equal probability, so that each bin has approximately the same likelihood of containing a sample. Specifically, let $F(\cdot)$ denote the CDF of $w^{(l)}h$. To partition the space into k equal-probability bins, we set the bias of each neuron as

$$b_i = -F^{-1}\left(\frac{i}{k}\right), \quad i = 1, \ldots, k. \tag{14}$$

where F^{-1} is the inverse CDF, which maps uniform probability intervals back to the original value space.

Once clients train on the configured weights and biases, each activation h_n is likely to fall into a distinct bin. When an activation h_n enters a bin, e.g., the interval $[-b_i, -b_{i+1})$, it activates neuron i as well as all neurons associated with larger biases. This yields a triangular (or progressive) activation pattern, where among the activated neurons, the one with the smallest bias is activated by a single sample, followed by the next neuron, which is activated by two samples, and so on. This pattern naturally motivates a recursive elimination strategy in which we first invert the neuron activated by a single sample, subtract its contribution from the remaining neurons, and then iteratively isolate the activations of all other samples. Formally, individual activations can be recovered by iteratively solving

$$h_n = \frac{\gamma_i^n g_i^{(W)} - \gamma_{i+1}^n g_{i+1}^{(W)}}{\gamma_i^n g_{i+1}^{(b)} - \gamma_{i+1}^n g_{i+1}^{(b)}}. \tag{15}$$

However, Eq. (15) cannot be solved directly because the values γ_i^n are unknown. A straightforward approach is to assume that all γ_i^n are identical for each sample, i.e., $\gamma_i^n = \gamma^n$ for all i, which allows γ^n to be eliminated from the equation. This can be achieved if the next layer has identical columns in the weight matrix [6], i.e., $W_j^{(l+1)} = w^{(l+1)}$ for all j columns (see Appendix A.2 for a detailed derivation). However, this constraint can be further relaxed by leveraging reasonable auxiliary knowledge, assuming that the attacker can infer the ground-truth label once the sample gets recovered. In this case, the full gradient signal $\{\gamma_i^n\}_{i=1}^k$ can be computed by feeding the recovered samples and their corresponding labels into the network. This approach is useful when only a single FC layer is present in the model.

Once the attacker obtains individual activations, it solves Eq. (11) to recover all $\tilde{\alpha}$ and reconstruct $\tilde{x}$ for the batch.

3.5. Attack Implementation and Recovery Rate

In this section, we illustrate the implementation of our attack under CNN and MLP-based networks. Our attack for the CNN-based network is summarized in Algorithm 1. Specifically, in the preparation stage, the attacker first initializes the convolutional kernels: the first kernel is initialized using a Gaussian random distribution (line 2), while the remaining kernels are initialized using the direct-pass method (line 3). For the FC layer, the weight matrix $W^{(l)}$ is initialized with identical rows (line 4). Based on $W^{(l)}$, the attacker derives the CDF of the projection values (line 5) and assigns the biases as the negative quantiles of this distribution, effectively disentangling the contribution of samples (lines 6–8). In the inference stage, the attacker first reconstructs the set of activations $\mathcal{H}$ from the gradients in the FC layer using linear layer leakage (line 10). Then, for each activation $h_n \in \mathcal{H}$, the attacker computes a sparse coefficient vector $\tilde{\alpha}_n$ (line 12). Finally, the training sample $\tilde{x}_n$ is recovered by projecting $\tilde{\alpha}_n$ back into the input space using the sparse basis Ψ (line 13).

Algorithm 1 ARES for CNN Network.

Input: Weight gradient $g^{(W)}$ and bias gradient $g^{(b)}$, sparse basis Ψ, FC layer l with k output neurons
Output: A set of recovered training samples $\tilde{\mathcal{X}}$

1: **// Attacker Preparation**
2: $K^{(1)} \leftarrow$ Gaussian random weight ▷ first conv layer
3: $K^{(2)}, K^{(3)}, \ldots, K^{(l-1)} \leftarrow$ direct-pass weight ▷ remaining conv layers
4: $W^{(l)} \leftarrow$ identical rows ▷ FC layer weights
5: $F \leftarrow$ estimate CDF of $w^{(l)}h$
6: **for** $i = 1, 2, \ldots, k$ **do**
7: $\quad b_i \leftarrow -F^{-1}(i/k)$ ▷ FC layer biases
8: **end for**
9: **// Attacker Inference**
10: $\mathcal{H} \leftarrow$ linear layer leakage via Eq. (15) ▷ activations
11: **for** $n = 1$ **to** $|\mathcal{H}|$ **do**
12: $\quad \tilde{\alpha}_n \leftarrow$ get sparse vector from h_n using Eq. (11)
13: $\quad \tilde{x}_n \leftarrow \Psi\, \tilde{\alpha}_n$ ▷ recover sample from sparse vector
14: $\quad \tilde{\mathcal{X}} \leftarrow \tilde{\mathcal{X}} \cup \{\tilde{x}_n\}$
15: **end for**

Our attack for MLP is summarized in Algorithm 2. Specifically, in the preparation stage, the attacker first initializes the weights of the first FC layer using a Gaussian random distribution (line 2). Based on $W^{(1)}$, the attacker estimates the CDF of the projection values (line 3) and assigns the biases $b_i^{(1)}$ of the first FC layer as the negative quantiles of this distribution (lines 4–6). For the second FC layer, the weight matrix $W^{(2)}$ is initialized with identical rows (line 7), and the corresponding biases $b_i^{(2)}$ are computed similarly using the estimated CDF from $W^{(2)}$ and the activations $h^{(1)}$ (lines 8–10). In the inference stage, the attacker first reconstructs the set of samples $\tilde{\mathcal{X}}^{(1)}$ from the first FC layer using linear layer leakage (line 13). Then it conducts linear layer leakage again on the second FC layer and gets a set of activations $\mathcal{H}^{(1)}$ (line 14). For each activation $h_n^{(1)} \in \mathcal{H}^{(1)}$, a sparse coefficient vector $\tilde{\alpha}_n$ is computed (line 16), and the corresponding input sample $\tilde{x}_n^{(2)}$ is reconstructed by projecting $\tilde{\alpha}_n$ back into the input space using the sparse basis Ψ (line 17). Each recovered sample is appended to the set $\tilde{\mathcal{X}}^{(2)}$, and finally, $\tilde{\mathcal{X}}^{(1)}$ and $\tilde{\mathcal{X}}^{(2)}$ are combined to obtain the full recovered samples $\tilde{\mathcal{X}}$ (line 20).

Algorithm 2 ARES for MLP Network

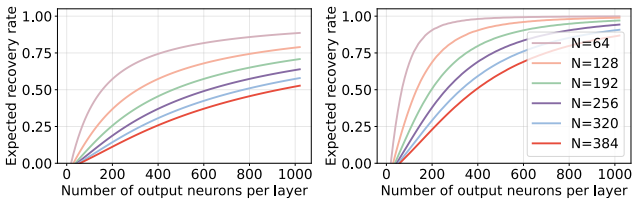
Input: Weight and bias gradient $g^{(W)}$, $g^{(b)}$, sparse basis Ψ
Output: A set of recovered training samples $\tilde{\mathcal{X}}$

1: **// Attacker Preparation**
2: $W^{(1)} \leftarrow$ Gaussian random weight ▷ first FC layer
3: $F^{(1)} \leftarrow$ estimate CDF of $w^{(1)}x$
4: **for** $i = 1, 2, \ldots, k^{(1)}$ **do**
5: $\quad b_i^{(1)} \leftarrow -\left(F^{(1)}\right)^{-1}\left(\frac{i}{k^{(1)}}\right)$ ▷ first FC layer biases
6: **end for**
7: $W^{(2)} \leftarrow$ identical rows ▷ second FC layer
8: $F^{(2)} \leftarrow$ estimate CDF of $w^{(2)}h^{(1)}$
9: **for** $i = 1, 2, \ldots, k^{(2)}$ **do**
10: $\quad b_i^{(2)} \leftarrow -\left(F^{(2)}\right)^{-1}\left(\frac{i}{k^{(2)}}\right)$ ▷ second FC layer biases
11: **end for**
12: **// Attacker Inference**
13: $\tilde{\mathcal{X}}^{(1)} \leftarrow$ first linear layer leakage ▷ samples
14: $\mathcal{H}^{(1)} \leftarrow$ second linear layer leakage ▷ activations
15: **for** $n = 1$ **to** $|\mathcal{H}^{(1)}|$ **do**
16: $\quad \tilde{\alpha}_n \leftarrow$ get sparse vector from $h_n^{(1)}$ using Eq. (11)
17: $\quad \tilde{x}_n^{(2)} \leftarrow \Psi\, \tilde{\alpha}_n$ ▷ recover input from sparse vector
18: $\quad \tilde{\mathcal{X}}^{(2)} \leftarrow \tilde{\mathcal{X}}^{(2)} \cup \{\tilde{x}_n^{(2)}\}$
19: **end for**
20: $\tilde{\mathcal{X}} \leftarrow \tilde{\mathcal{X}}^{(1)} \cup \tilde{\mathcal{X}}^{(2)}$

In the following, we provide the expected recovery rate for both the one-layer and two-layer recovery. Intuitively, the expected number of recovered samples can be derived by enumerating all possible assignments of N samples into k equal-mass bins. A sample is considered recovered if it occupies a bin alone, since in this case the linear-layer inversion can uniquely identify that sample. Formally, the expected recovery rate for a single FC layer is given by

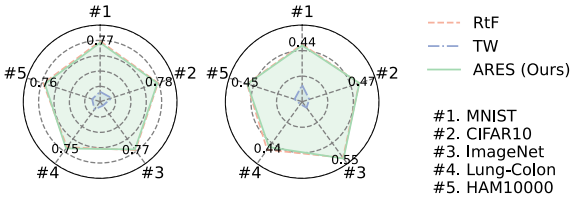
$$E(N,k) = \frac{1}{\binom{k+N-1}{k-1}} \sum_{i=1}^{N-2} i \binom{k}{i} \times \sum_{j=1}^{\lfloor \frac{N-i}{2} \rfloor} \binom{k-i}{j} \binom{N-i-j-1}{j-1} + r(N,k), \tag{16}$$

where k is the number of output neurons and N is the number of samples in the batch, and $r(n,k)$ captures special configurations, e.g., samples didn't fall into any bin (cf. RtF [6]). Let $p_1 = E(N,k^{(1)})/N$ denote the proportion



(a) One Layer (b) Two Layers

Figure 6: Expected recovery rate with different numbers of output neurons (k) and batch size (N).



(a) Batch size 64 (b) Batch size 256

Figure 7: Recovery rate comparison in CNN.

of samples successfully recovered in the first layer. The proportion of samples successfully recovered second layer is $p_2 = E(N(1-p_1), k^{(2)})/N(1-p_1)$. Consequently, the total number of samples recovered across the two layers is $N\,[p_1 + (1-p_1)p_2]$. As illustrated in Fig. 6, increasing the number of output neurons results in a higher expected recovery rate. Moreover, adding a second FC layer yields a substantial performance improvement; for instance, with 1024 output neurons per layer, a two-layer configuration can recover over 80% of the samples in a batch of 384.

4. Experiments

We conduct extensive experiments to evaluate the effectiveness of ARES. Section 4.2 compares ARES with SOTA attacks on CNN- and MLP-based networks. Section 4.3 evaluates ARES under gradient perturbation, data augmentation, and secure aggregation defenses. Section 4.4 explores ARES on text and audio data, non-IID scenarios, FedAvg, and asynchronous FL settings.

4.1. Experimental Setup

Dataset. We use five image datasets, including MNIST [22], CIFAR-10 [23], ImageNet [24], HAM10000 [25], Lung-Colon Cancer [26], one text dataset, i.e., Wikitext dataset [27], and one audio dataset, i.e., AudioMNIST [28] to evaluate the effect of our attack.

Evaluated Networks. We adopt two representative network architectures: a 4-layer CNN, which consists of convolutional layers followed by FC layers, and a 4-layer MLP, which comprises four FC layers. The detailed descriptions for networks are provided in Table 4 in the Appendix.

Evaluation Metric. We employ four metrics to assess the effectiveness of our attack, including PSNR (higher is better), MSE (lower is better), LPIPS (lower is better), and recovery rate (higher is better). For PSNR, MSE, and LPIPS, we report the averaged value across the batch. The detailed descriptions for each metric are provided in Appendix B.1.

Compared Attacks. We compare our methods with nine state-of-the-art GIAs, including iDLG [1], InvertingGrad (IG) [2], GradInversion (GI) [3], FedLeak [4], Fishing [5], Robbing (RtF) [6], Trap Weight (TW) [7], LOKI [8] and Scale-MIA [35]. To implement RtF and TW in a CNN-based network, we adopt the approach proposed in TW [7], which uses the direct-pass initialization method for convolutional layers to avoid value distortion. Our implementation for each method is based on the open-source code[3]. The detailed descriptions for each attack are provided in Appendix B.2.

Evaluated Defenses. We consider three categories of defenses. First, we evaluate three gradient perturbation–based approaches, including differential privacy (DP) [29], gradient quantization [30], and gradient sparsification [30]. Second, we consider a data augmentation–based defense, ATS [31], implemented using public code[4] . Finally, we examine secure aggregation–based defenses [32], [33]. The detailed explanations of each defense are provided in Appendix B.3.

4.2. Performance Comparison with Baselines

Comparison on CNN-based Networks. Table 2 demonstrates that our attack consistently outperforms SOTA GIAs across all datasets and practical batch sizes. We leave the visual illustration of the recovery effect in Appendix C (Fig. 19). For fairness, we focus on comparison with RtF and TW, as both are active attacks that leverage the linear layer leakage technique. As shown in Fig. 7, both RtF and ARES achieve a higher recovery rate than TW, with improvements of 6.58 $\times$ and 6.57 $\times$ for a batch size of 64, and 7.14 $\times$ and 7.05 $\times$ for a batch size of 256, respectively. This improvement is attributed to the use of the imprint method, which maximizes the likelihood of assigning each sample to distinct bins, thereby enhancing the probability of separating samples within a batch. However, even at a comparable recovery rate, ARES achieves a significantly higher PSNR than RtF, with an average improvement of 4.8 $\times$. This improvement stems from our method's ability to address the key challenge of reconstructing activations back to the original training samples. Although initializing convolutional kernels using the direct-pass method can partially mitigate value distortion, ReLU activations in preceding layers still mask out almost half of the signals.

We further compare the empirical PSNR for single image with the theoretical bound provided by Theorem 1. Theorem 1 establishes an upper bound on the ℓ_2 distance between the estimated and ground-truth signals, which translates into a lower bound on PSNR for image reconstruction. Under the ARES design, the theoretical MSE upper bound is

3. https://github.com/lhfowl/robbing_the_fed, https://github.com/JonasGeiping/breaching, https://github.com/unknown123489/Scale-MIA.
4. https://github.com/gaow0007/ATSPrivacy

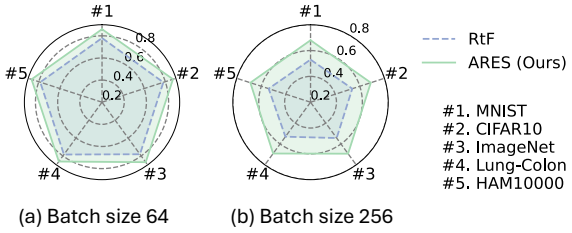


Figure 8: Recovery rate comparison in MLP.

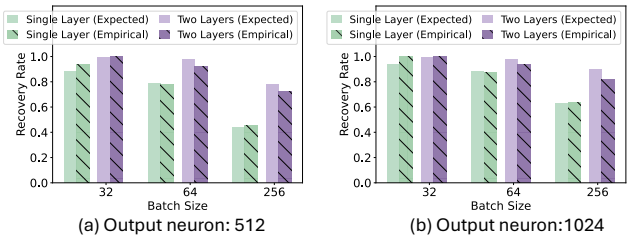


Figure 9: Expected vs. empirical recovery rates.

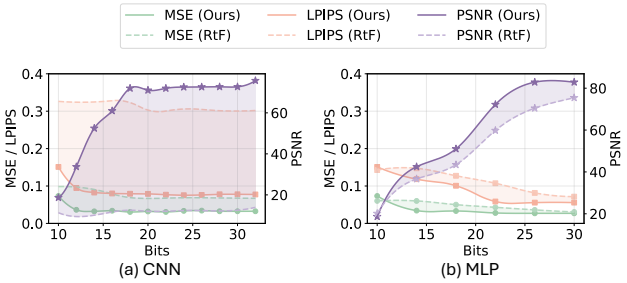


Figure 10: Our attack under gradient quantization defense.

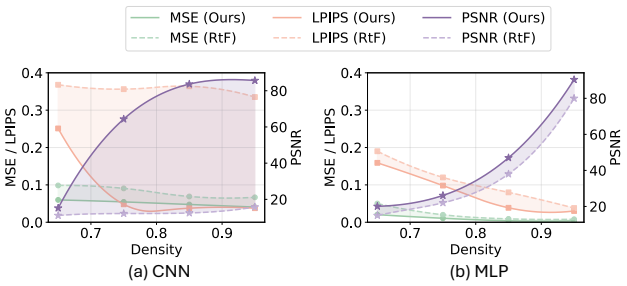


Figure 11: Our attack under gradient sparsification defense.

around 0.25, corresponding to a PSNR lower bound of 54. In practice, the PSNR for a single image can exceed 100, which remains well within the guarantee provided by the theorem. This gap reflects the conservative nature of the bound, as it holds for any s-sparse vector, including worst-case or adversarially constructed signals, whereas real images typically exhibit smooth regions and structured patterns that facilitate easier reconstruction.

Comparison on MLP-based Networks. To evaluate our attack on an MLP-based network, we primarily compare our method with RtF, which is currently the strongest active attack. As shown in Fig. 8, ARES improves the recovery rate by 10.75% and 28.34% for batch sizes of 64 and 256, respectively. This improvement arises from our method's ability to perform second-layer separation.

Theoretical and Empirical Recovery Rates. We compare the expected recovery rates derived from Eq. (16) with the empirical recovery rates obtained by averaging the results across the datasets reported in Table 2. As shown in Fig. 9, the empirical results closely follow the expected values across different settings, with an average deviation of 3.5%.

4.3. Performance Under Different Defenses

Attack Against Gradient Perturbation-based Defense. Fig. 10 shows the performance of ARES under gradient quantization defense with a batch size of 32 on the ImageNet dataset. For fairness, we compare only with RtF, which is the strongest baseline according to Table 2. As shown, ARES consistently outperforms RtF on both CNN and MLP networks, achieving PSNR improvements of 5× and 1.35×, respectively. As the quantization bit increases, the reconstruction quality improves. Fig. 11 shows the performance of ARES under gradient sparsification defense with a batch size of 32 on the ImageNet dataset. Here, the density denotes the fraction of gradient retained after sparsification. ARES consistently outperforms RtF on both CNN and MLP networks, achieving PSNR improvements of 3 × and 1.12 ×, respectively. As this density increases, the reconstruction quality improves. Fig. 12 shows the performance of ARES under different DP noise with a batch size of 32 on the ImageNet dataset. Following prior works [6], [8], we apply the Laplace mechanism with varying privacy budgets ε. ARES consistently outperforms RtF on both CNN and MLP networks, achieving PSNR improvements of 2.5 × and 1.14 ×, respectively. At $\varepsilon = 10$, the recovered images achieve PSNR $\approx$ 20. Further reducing the privacy budget leads to a significant drop in training accuracy. We leave the visual effect in Appendix C (Fig. 20, Fig. 21 and Fig. 22).

Attack Against Data Augmentation-based Defense. We evaluate our ARES against ATS [31], a data augmentation–based defense that aims to learn the optimal augmentation policy to prevent the reconstruction of both original and transformed training samples, while preserving model

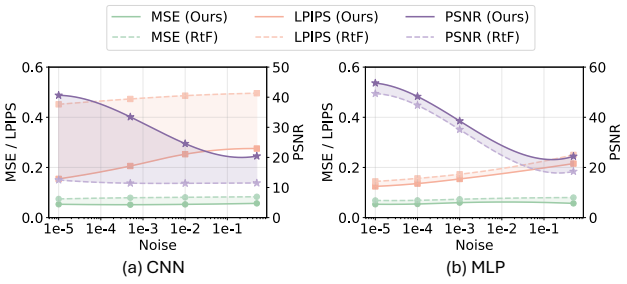


Figure 12: Our attack under differential privacy noise.

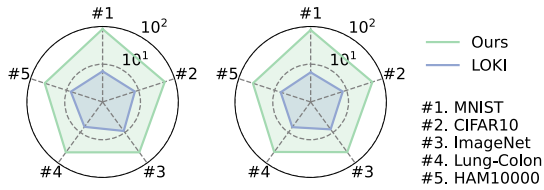
TABLE 2: PSNR comparison of state-of-the-art attacks versus our ARES across different batch sizes and datasets. For each row, the best-performing attack is highlighted in bold. ✗ indicates that the attack fails to get visually meaningful images.

Dataset	Batch	iDLG	IG	GI	FedLeak	Finshing*	RtF	TW	ARES (Ours)
MNIST	32	9.39	9.96	11.17	21.57	12.40	17.53	13.89	**105.29**
	64	9.27	9.88	10.94	21.12	12.58	16.68	13.02	**94.16**
	256	✗	✗	✗	✗	12.40	13.08	11.87	**42.42**
CIFAR-10	32	9.51	11.28	10.59	21.33	12.08	17.35	13.55	**92.27**
	64	8.52	✗	10.01	17.20	12.36	17.20	13.03	**90.47**
	256	✗	✗	✗	✗	12.36	15.08	13.51	**37.08**
ImageNet	32	✗	11.45	8.06	19.07	12.71	16.58	12.92	**104.89**
	64	✗	10.75	✗	19.01	12.68	15.20	12.59	**90.72**
	256	✗	✗	✗	✗	12.49	14.86	12.56	**41.61**
HAM10000	32	✗	✗	✗	15.32	12.67	15.32	15.29	**120.93**
	64	✗	✗	✗	22.16	12.52	15.01	14.39	**69.96**
	256	✗	✗	✗	✗	12.48	14.97	14.88	**37.72**
Lung-Colon	32	✗	✗	✗	17.73	12.88	17.10	14.36	**106.64**
	64	✗	✗	✗	16.06	12.41	16.64	14.20	**67.37**
	256	✗	✗	✗	✗	12.27	15.06	13.84	**33.12**

* Fishing column reports the PSNR for a single sample; others present the average PSNR across all samples in the batch.

utility. Our attack achieves an average PSNR of 86.8 (comparing reconstructed samples with the transformed samples) on the CIFAR-10 dataset with a batch size of 32. We leave the visual illustration of the augmented samples and the recovered samples in Appendix C (Fig. 23).

Attack Against Secure Aggregation Defense. A commonly used strategy to bypass secure aggregation is to exploit model inconsistency [8], [48], where the server sends different model parameters (within the same model architecture) to each client, preventing the mixing of gradients from different clients. One such approach is LOKI [8], which leverages model inconsistency to perform GIA under secure aggregation defense. To evaluate our method under the same defense, we adopt LOKI's setup and deliberately manipulate convolutional weights to introduce model inconsistency, preventing the clients' weight gradients mixing during secure aggregation. Because ARES is orthogonal to LOKI (i.e., model inconsistency method), integrating ARES into the LOKI setup yields complementary attack capabilities. In our experiment, each client receives a model in which only a client-unique subset of kernels (e.g., three per client) is initialized with the malicious weight (lines 2–3 in Algorithm 1), while all remaining kernels are set to zero. This design keeps each client's weight gradients in the FC layer unmixed thus improve the recovery rate. We evaluate this approach using 10 clients per training round. As shown in Fig. 13, our method achieves an average PSNR of 53.7 and 51.9 on CNN-based networks with global batch sizes (i.e., local batch size × number of clients per round) of 320 and 640, respectively, outperforming LOKI by 7.3 × and 7.1 ×. The higher PSNR is achieved by addressing the challenge of reconstructing training samples from activations with minimal information loss. Although LOKI can initialize the convolutional kernels using the direct-pass method to reduce



(a) Global batch size 320 (b) Global batch size 640

Figure 13: PSNR of LOKI and ours under secure aggregation-based defenses. The radial axis is log-scaled for better visualization.

value distortion, ReLU activations still suppress half of the inputs to the FC layer, resulting in information loss.

Global batch size	32	64	128	256
Scale-MIA	28.52	28.53	28.26	27.06
ARES	92.27	90.47	85.01	37.08

TABLE 3: PSNR comparison of Scale-MIA and ARES on CIFAR-10 dataset in CNN.

We also compare ARES with an attack that relies on stronger adversarial assumptions, namely Scale-MIA [35], which assumes the attacker has access to a subset of the training data to train a decoder that maps activations back to the original samples. As shown in Table 3, ARES achieves an average PSNR improvement of 2.71 × over Scale-MIA despite having less prior knowledge.

4.4. Attack in Diverse Settings

Attack on Text Data. We use an MLP network to test the effect of our attack. We report three evaluation met-

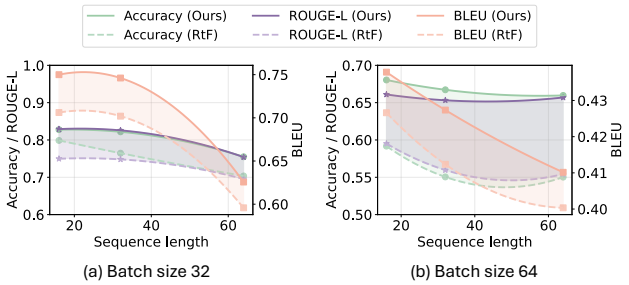


Figure 14: Our attack on the Wikitext dataset.

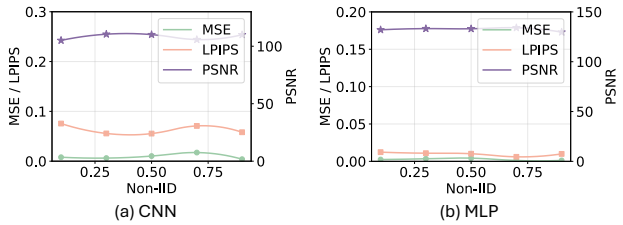


Figure 15: Our attack on non-iid data with label skew.

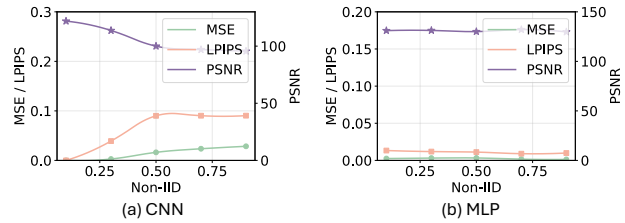


Figure 16: Our attack on non-iid data with feature skew.

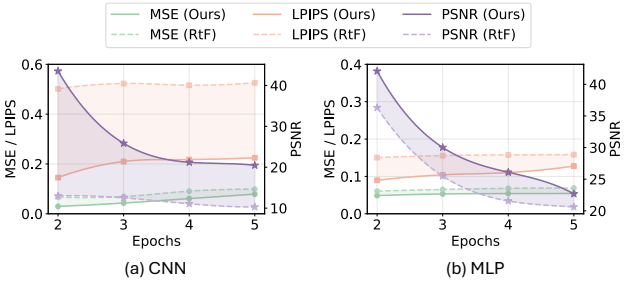


Figure 17: Our attack in the FedAvg setting.

rics commonly used in text datasets, including accuracy, BLEU score, and ROUGE-L (details in Appendix B.1). As shown in Fig. 14, our method consistently outperforms RtF across all combinations of sequence lengths and batch sizes, achieving improvements of 5.76%, 5.84%, and 8.92% in accuracy, BLEU, and ROUGE-L on batch size 32, and 15.71%, 2.91%, and 13.29% on batch size 64. We leave the recovery effect in Appendix C (Fig. 24).

Attack on Audio Data. We use a CNN to test the effect of our attack. Our method achieves an average PSNR of 58.55 and 45.83 on a batch size of 32 and 64, respectively. Ground truth and recovered audio files (WAV) are available[5].

Label Skew. We evaluate our attack under varying degrees of label skew, where each client holds only a subset of classes. The skew scalar controls the level of non-IIDness (0: IID; larger values indicate fewer classes per client). Results on the CIFAR-10 dataset with a batch size of 32 show that our attack consistently achieves strong performance across all levels of label skew (Fig. 15).

Feature Skew. We evaluate our attack under varying degrees of feature skew by partitioning the dataset in the feature space. Specifically, we first project samples onto the top principal components using Principal Component Analysis (PCA) and divide the resulting feature space into multiple regions. Each client is then assigned samples from only a subset of these regions, creating feature distribution differences across clients. The skew scalar controls the number of regions assigned to each client (0: IID; larger values correspond to fewer regions per client and therefore stronger feature skew). Experimental results on the CIFAR-10 dataset with a batch size of 32 show that, in the CNN network, the PSNR decreases as feature skew increases because more samples fall into the same bin compared to the standard IID case. Nevertheless, the attack still achieves a PSNR of around 100 dB. In contrast, in the MLP network, our attack performs well across all degrees of feature skew, as the second layer enables further separation of samples (Fig. 16).

Attack on FedAvg. In FedAvg, each client trains locally on its own data for T epochs and then sends model updates to the server. To attack under this setting, the attacker first estimates the gradient using

$$\hat{g} \approx -\frac{1}{lr\,T}\Delta W, \tag{17}$$

where $\hat{g}$ is the estimated gradient, lr is the local learning rate, T is the number of local training epochs, ΔW is the client's model update. Next, the server further estimates the intermediate weight for each local epoch as

$$\hat{W}_t = W_g - t\,lr\hat{g}, \quad t = 1, \ldots, T, \tag{18}$$

where $\hat{W}_t$ is the estimated model at local epoch t and W_g is the global model at the start of the round. After obtaining the estimated gradient and weight, the attacker uses Eq. (15) and Eq. (11) to get the training samples. We evaluate our method on FedAvg using the HAM dataset with a local batch size of 8. As shown in Fig. 17, our approach consistently outperforms RtF[6] on both CNN and MLP networks, achieving PSNR improvements o 2 $\times$ and 1.16 $\times$, respectively.

Attack on Asynchronous FL. Similarly, we use Eq. (17) and Eq. (18) to estimate the gradient and local model weight for each client. And use Eq. (15) and Eq. (11) to

5. https://github.com/gongzir1/ARES/tree/main/AudioExample

6. We compare with the main method (Eq. (4) in [6]). Although the original paper proposes a variant that performs well under the FedAvg setting, it requires a malicious modification of the activation function, which is beyond the scope of our threat model.

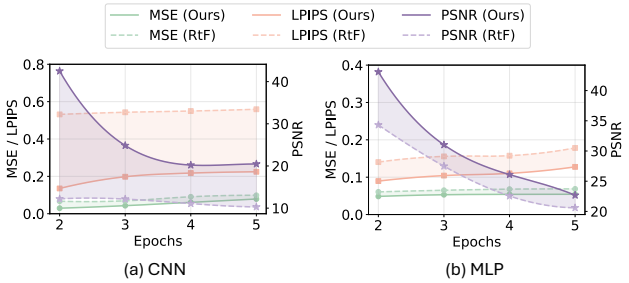


Figure 18: Our attack in the Asynchronous FL setting.

get the training samples. We evaluate our method on the Asynchronous FL framework [49] using the HAM dataset with a local batch size of 8. As shown in Fig. 18, our approach consistently outperforms RtF on both CNN and MLP networks, achieving PSNR improvements of 2 $\times$ and 1.15 $\times$ in PSNR, respectively.

Impact of Activation Function on Noisy Sparse Recovery. We evaluate the performance of our noisy sparse recovery under commonly used activation functions, including GELU, ELU, and SiLU. The problem is solved by optimizing Eq. (12) using a gradient-based method (SGD). Experiments on the Conv4 network show strong reconstruction performance across all activation functions, achieving single-sample PSNR values of 82.25 dB (GELU), 78.39 dB (ELU), and 85.71 dB (SiLU).

Impact of Activation Functions on Linear Layer Leakage. We evaluate the performance of linear-layer leakage under commonly used activation functions, including GELU, ELU, and SiLU. Although the choice of activation function can influence leakage, these activations can be adapted to exhibit ReLU-like behavior. Specifically, ELU reduces to ReLU when its scaling parameter is set to zero. For GELU and SiLU, scaling the pre-activation FC weights pushes activations away from zero, thereby making them behave similarly to ReLU. Experiments on Conv4 with a batch size of 32 demonstrate strong performance across all activation functions, achieving a PSNR of 100.95 dB with a 97% recovery rate for ELU, 94.70 dB with a 93% recovery rate for GELU, and 94.56 dB with a 93% recovery rate for SiLU.

5. Conclusion

In this work, we introduce ARES, a practical and effective active GIA capable of recovering training data from gradients with high fidelity under realistic large batch sizes and reasonable adversary assumptions. It achieves this by addressing the fundamental challenge of inverting hidden activations into training samples. By exploiting the sparse nature of real-world data and leveraging principles from compressed sensing, we formulate the inversion task as a noisy sparse recovery problem, overcoming the underdetermined and nonlinear challenges inherent in the task. We establish a theoretical upper bound on the recovery error and validate our approach through extensive experiments across diverse datasets, architectures, and defense mechanisms. The results demonstrate that our ARES consistently achieves superior recovery fidelity, highlighting the underestimated privacy risks of FL in real-world settings. Investigating the attack's scalability to architectures without the Linear+ReLU structure and its robustness against targeted defenses and homomorphic encryption remains a promising direction for future work.

6. Ethics Considerations

The experiments are conducted exclusively on public datasets and open-source models within controlled environments, without access to real-world deployments or personal data. No undisclosed vulnerabilities are introduced or exploited, and no human subjects are involved in this study.

7. LLM Usage Considerations

LLMs were used for editorial purposes in this manuscript, and all outputs were inspected by the authors to ensure accuracy and originality.

References

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

[2] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 16 937–16 947.

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

[4] M. Fan, F. Wang, C. Chen, and J. Zhou, "Boosting gradient leakage attacks: Data reconstruction in realistic fl settings," in *USENIX Security Symposium*, 2025.

[5] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," in *ICML*, 2022.

[6] L. H. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *International Conference on Learning Representations*.

[7] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 175–199.

[8] J. C. Zhao, A. Sharma, A. R. Elkordy, Y. H. Ezzeldin, S. Avestimehr, and S. Bagchi, "Loki: Large-scale data reconstruction attack against federated learning through model manipulation," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 1287–1305.

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

[10] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (Csur)*, vol. 55, no. 3, pp. 1–37, 2022.

[11] P. M. Mammen, "Federated learning: Opportunities and challenges," *arXiv preprint arXiv:2101.05428*, 2021.

[12] Z. Gong, Y. Zhang, L. Y. Zhang, Z. Zhang, Y. Xiang, and S. Pan, "Not all edges are equally robust: Evaluating the robustness of ranking-based federated learning," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 2527–2545.

[13] Z. Gong, L. Shen, Y. Zhang, L. Y. Zhang, J. Wang, G. Bai, and Y. Xiang, "Agramplifier: Defending federated learning against poisoning attacks through local update amplification," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1241–1250, 2023.

[14] Z. Chen, Z. Gong, J. Ning, Y. Zhang, and L. Y. Zhang, "Beyond denial-of-service: The puppeteer's attack for fine-grained control in ranking-based federated learning," in *Proceedings of the Web Conference 2026 (WWW '26)*. ACM, 2026.

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

[16] X. Feng, Z. Ma, Z. Wang, E. J. Chegne, M. Ma, A. Abuadbba, and G. Bai, "Uncovering gradient inversion risks in practical language model training," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 3525–3539.

[17] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[18] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[19] E. J. Candes, "The restricted isometry property and its implications for compressed sensing," *Comptes rendus. Mathematique*, vol. 346, no. 9-10, pp. 589–592, 2008.

[20] Y. Plan and R. Vershynin, "The generalized lasso with non-linear observations," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1528–1537, 2016.

[21] L. Y. Zhang, K.-W. Wong, Y. Zhang, and J. Zhou, "Bi-level protected compressive sampling," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1720–1732, 2016.

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

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

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[25] P. Tschandl, C. Rosendahl, and H. Kittler, "The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, no. 1, pp. 1–9, 2018.

[26] A. A. Borkowski, M. M. Bui, L. B. Thomas, C. P. Wilson, L. A. DeLand, and S. M. Mastorides, "Lung and colon cancer histopathological image dataset (lc25000)," *arXiv preprint arXiv:1912.12142*, 2019.

[27] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[28] S. Becker, J. Vielhaben, M. Ackermann, K.-R. Müller, S. Lapuschkin, and W. Samek, "Audiomnist: Exploring explainable artificial intelligence for audio analysis on a simple benchmark," *Journal of the Franklin Institute*, vol. 361, no. 1, pp. 418–428, 2024.

[29] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[30] K. Yue, R. Jin, C.-W. Wong, D. Baron, and H. Dai, "Gradient obfuscation gives a false sense of security in federated learning," in *32nd USENIX security symposium (USENIX Security 23)*, 2023, pp. 6381–6398.

[31] W. Gao, X. Zhang, S. Guo, T. Zhang, T. Xiang, H. Qiu, Y. Wen, and Y. Liu, "Automatic transformation search against deep leakage from gradients," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 650–10 668, 2023.

[32] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[33] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame *et al.*, "Safelearn: Secure aggregation for private federated learning," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 56–62.

[34] X. Feng, Z. Ma, Z. Wang, A. Abuadbba, and G. Bai, "Mitigating gradient inversion risks in language models via token obfuscation," in *Proceedings of the 2026 on ACM ASIA Conference on Computer and Communications Security (AsiaCCS)*, 2026.

[35] S. Shi, N. Wang, Y. Xiao, C. Zhang, Y. Shi, Y. T. Hou, and W. Lou, "Scale-MIA: A scalable model inversion attack against secure federated learning via latent space reconstruction," in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*.

[36] W. Gao, S. Guo, T. Zhang, H. Qiu, Y. Wen, and Y. Liu, "Privacy-preserving collaborative learning with automatic transformation search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 114–123.

[37] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[38] L. Xue, S. Hu, R. Zhao, L. Y. Zhang, S. Hu, L. Sun, and D. Yao, "Revisiting gradient pruning: A dual realization for defending against gradient attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, 2024, pp. 6404–6412.

[39] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data *et al.*, "A field guide to federated optimization," *arXiv preprint arXiv:2107.06917*, 2021.

[40] L. H. Fowl, J. Geiping, S. Reich, Y. Wen, W. Czaja, M. Goldblum, and T. Goldstein, "Decepticons: Corrupted transformers breach privacy in federated learning for language models," in *The Eleventh International Conference on Learning Representations*.

[41] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[42] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[43] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[44] M. Genzel and P. Jung, "Recovering structured data from superimposed non-linear measurements," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 453–477, 2019.

[45] M. Genzel, "High-dimensional estimation of structured signals from non-linear observations with general convex loss functions," *IEEE Transactions on Information Theory*, vol. 63, no. 3, pp. 1601–1619, 2016.

[46] E. J. Candes and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[47] S. Diamond and S. Boyd, "Cvxpy: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[48] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2429–2443.

[49] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 3581–3607.

Appendix A. Derivations

A.1. Activation Recovery from FC Layer

For a batch of N samples $\mathcal{N} = \{1, \dots, N\}$, the forward pass of neuron i in layer l for sample n is

$$z_{i,n}^{(l)} = W_i^{(l)} h_n^{(l-1)} + b_i^{(l)}, \tag{19}$$

where $h_n^{(l-1)}$ denotes the input vector of sample n to layer l. According to the chain rule, the gradient of the loss with respect to the weight for neuron i is

$$g_i^{(W)} = \frac{\partial \mathcal{L}}{\partial W_i^{(l)}} = \sum_{n \in \mathcal{N}} \frac{\partial \mathcal{L}}{\partial z_{i,n}^{(l)}} \frac{\partial z_{i,n}^{(l)}}{\partial W_i^{(l)}}. \tag{20}$$

The gradient with respect to the bias $b_i^{(l)}$ is

$$g_i^{(b)} = \frac{\partial \mathcal{L}}{\partial b_i^{(l)}} = \sum_{n \in \mathcal{N}} \frac{\partial \mathcal{L}}{\partial z_{i,n}^{(l)}} \frac{\partial z_{i,n}^{(l)}}{\partial b_i^{(l)}}. \tag{21}$$

Using the linear relationship in Eq. (19), we have

$$\frac{\partial z_{i,n}^{(l)}}{\partial W_i^{(l)}} = h_n^{(l-1)}, \quad \frac{\partial z_{i,n}^{(l)}}{\partial b_i^{(l)}} = 1. \tag{22}$$

Combine Eq. (20), Eq. (21) and Eq. (22), we have

$$g_i^{(W)} = \sum_{n \in \mathcal{N}} \frac{\partial \mathcal{L}}{\partial z_{i,n}^{(l)}} h_n^{(l-1)} = \sum_{n \in \mathcal{N}} \gamma_i^n h_n^{(l-1)}, \tag{23}$$

$$g_i^{(b)} = \sum_{n \in \mathcal{N}} \frac{\partial \mathcal{L}}{\partial z_{i,n}^{(l)}} = \sum_{n \in \mathcal{N}} \gamma_i^n, \tag{24}$$

where $\gamma_i^n := \frac{\partial \mathcal{L}}{\partial z_{i,n}^{(l)}}$ denotes the gradient of the loss with respect to the pre-activation of neuron i. Taking the ratio of the weight and bias gradients yields

$$\frac{g_i^{(W)}}{g_i^{(b)}} = \frac{\sum_{n \in \mathcal{N}} \gamma_i^n h_n^{(l-1)}}{\sum_{n \in \mathcal{N}} \gamma_i^n}. \tag{25}$$

This expression corresponds to a *weighted average* of the input vectors $\{h_n^{(l-1)}\}_{n=1}^N$, where the weights are given by the gradient coefficients $\{\gamma_i^n\}_{n=1}^N$. When the batch size reduces to $N = 1$, Eq. (25) simplifies to

$$\frac{g_i^{(W)}}{g_i^{(b)}} = h^{(l-1)}, \tag{26}$$

which recovers the single-sample result in Eq. (3).

A.2. Derivation of Equal Backpropagation Signals

Consider neuron i in layer l with backpropagation signal

$$\gamma_i := \frac{\partial \mathcal{L}}{\partial z_i^{(l)}} = \sum_j \frac{\partial \mathcal{L}}{\partial z_j^{(l+1)}} \frac{\partial z_j^{(l+1)}}{\partial z_i^{(l)}} = \sum_j \frac{\partial \mathcal{L}}{\partial z_j^{(l+1)}} W_{ji}^{(l+1)}, \tag{27}$$

where $W^{(l+1)}$ is the weight matrix of layer $l+1$. Similarly, for neuron k,

$$\gamma_k = \sum_j \frac{\partial \mathcal{L}}{\partial z_j^{(l+1)}} W_{jk}^{(l+1)}. \tag{28}$$

Once

$$W_{ji}^{(l+1)} = W_{j,k}^{(l+1)}, \tag{29}$$

then it can achieve $\gamma_i = \gamma_k$.

Appendix B. Detailed Explanation on Experiment Set Up

Architecture	Layers
CNN	Conv(out=12, k=3, s=1, p=1, act=relu) Conv(out=24, k=3, s=1, p=1, act=relu) FC(k=1024, act=relu) FC(k=#class, act=softmax)
MLP	FC(k=512, act=relu), FC(k=512, act=relu), FC(k=512, act=relu), FC(k=#class, act=softmax)

TABLE 4: Network architectures. Conv: *out* = number of filters, *k* = kernel size, *s* = stride, *p* = padding, *act* = activation. FC: *k* = number of neurons, *act* = activation.

B.1. Evaluation Metrics

MSE (Mean Squared Error) is a metric that computes the average of squared intensity differences between the reconstructed image and the ground truth. Lower MSE indicates better pixel-wise fidelity. **PSNR** (Peak Signal-to-Noise Ratio) is a metric that quantifies the fidelity of reconstructed images relative to ground truth. It is defined as a logarithmic function of the MSE between two images, with higher values indicating better reconstruction quality. **LPIPS** (Learned Perceptual Image Patch Similarity) is a metric that measures perceptual distance using deep neural network features pretrained on large-scale image data. Lower values indicate higher perceptual similarity. **Recovery Rate.** We count the number of samples that fall into distinct bins and divide this number by the batch size to compute the recovery rate. **Reconstruction accuracy** measures the proportion of tokens in the reconstructed text that match the original tokens. **BLEU score** evaluates the overlap of n-grams (sequences of one or more words) between the reconstructed and original text, rewarding partial matches and fluency even when not all tokens are identical. **ROUGE-L** captures similarity based

on the longest common subsequence, highlighting how well the global word order and sentence structure in the reconstruction align with the original text.

B.2. Compared Attacks

iDLG (Improved Deep Leakage from Gradients) [1] is a passive GIA attack that enables recovery of both training sample and label for a single sample. The key idea is that, under cross-entropy loss with one-hot labels, the ground-truth label can be directly inferred from the sign of the gradient of the last FC layer. With the true label identified, iDLG then reconstructs the input sample by iteratively optimizing dummy data to match the observed gradients.
InvertingGrad (IG) [2] is a passive GIA that reconstructs data by optimizing dummy inputs to match the shared gradients. To improve reconstruction quality, it incorporates additional priors (i.e., total variation) to regulate the optimization space. This regularization yields natural-looking images and enhances the fidelity of the recovered data.
GradInversion (GI) [3] is a passive GIA that assumes access to batch normalization (BN) statistics to constrain reconstructions. Following iDLG, it directly infers the ground-truth label from the final layer's gradients, avoiding unstable label optimization. To further improve reconstruction quality, GradInversion introduces a group fidelity term that iteratively aligns reconstructed gradients with the originals, producing high-resolution and semantically accurate images.
FedLeak [4] is a passive GIA designed for realistic FL settings. It tackles the core challenge of gradient matching through two techniques: partial gradient matching, which targets informative gradient components, and gradient regularization, which stabilizes optimization.
Fishing [5] is an active GIA that aims to recovery single sample in a batch of samples. By decreasing the network's confidence in the target class and target feature, it encourage the gradient come from only the target (single) sample.
Robbing (RtF) [6] is an active GIA that uses the linear layer leakage to recover training samples. By carefully designing the weights and biases of FC layers, RtF imprints each neuron with a single data point, ensuring that its activation predominantly corresponds to that sample.
Trap Weight (TW) [7] is an active GIA that reconstructs training samples by configuring FC weights so each neuron responds to a single input. It sets roughly half of the weights in FC layer to small negative and half to positive values, isolating individual samples in a batch. TW also uses direct-pass weights in convolutional layers to avoid architectural changes, but it only works for nonnegative inputs; standard normalization with ReLU zeroes negative values, breaking the identity mapping and causing information loss.
LOKI [8] is an active GIA targeting secure aggregation-based FL, where only aggregated gradients are visible to the server. It inserts an extra convolutional layer before the FC layer and assigns each client a unique subset of kernels with direct-pass method and others are set to zero. This ensures client-specific activations, enabling the server to disentangle and recover individual gradients after aggregation. Building on the imprint method proposed by RtF [6], it enables individual sample recovery at scale.
Scale-MIA [35] is an active GIA built upon RtF to separate sample contributions in FC layer. It avoids architectural modifications by leveraging the built-in FC layer for linear layer leakage. However, it requires a subset of the training data to train a decoder that maps latent representations back to samples, limiting its generalization to unseen domains.

B.3. Evaluated Defenses

Differential Privacy (DP) [29] protects client data by adding random noise to local gradients before they are shared with the server. Each client first clips its gradient to a maximum norm to limit the influence of any individual training sample, and then adds noise, so that the resulting gradient reveals only limited information.
Gradient Quantization [30] reduces the precision of client gradients before sending them to the server. This limits the amount of information that can be extracted from individual gradients, while also reducing communication overhead.
Gradient Sparsification [30] reduces the amount of information transmitted by sending only a subset of the gradient elements to the server, typically those with the largest magnitudes. This not only lowers communication costs in FL but also limits the information available to potential attackers attempting gradient inversion.
Data augmentation techniques apply carefully chosen transformations to the training data to prevent adversaries from reconstructing both the augmented and original samples from shared gradients [31], [36]. To achieve this, ATS [36] employs a privacy score and a training-free accuracy metric to automatically discover effective transformations, yielding a lightweight privacy defense.
Secure aggregation is a privacy-preserving techniques in FL that allow the server to aggregate local model updates from multiple clients without ever seeing individual updates. A commonly used example is Masking-Based Secure Aggregation (SA) [32], [33]. In this approach, each client adds a random mask to its local model update before sending it to the server. When the server sums all masked updates, the masks cancel out, enabling the server to recover only the plaintext of the aggregated model.

Appendix C. Visualization Results

Fig. 19 presents the ground truth and recovered samples across five image datasets using a CNN with a batch size of 32. In all datasets, AERS successfully reconstructs the samples without visually perceptible loss whenever the samples fall into distinct bins. Fig. 24 shows examples of the original ground-truth training text and the corresponding recovered text on the WikiText dataset using an MLP network. Fig. 20 to 23 shows the recovery effect of our attack under different defenses, including gradient quantization-based defense (Fig. 20), gradient sparsity-based defense

Figure 19: Visual illustration of the recovery effect on five image datasets.

Figure 20: Visualization of attack effect under gradient quantization-based defense.

(Fig. 21), differential privacy-based defense (Fig. 22), and data argumentation-based defense (Fig. 23).

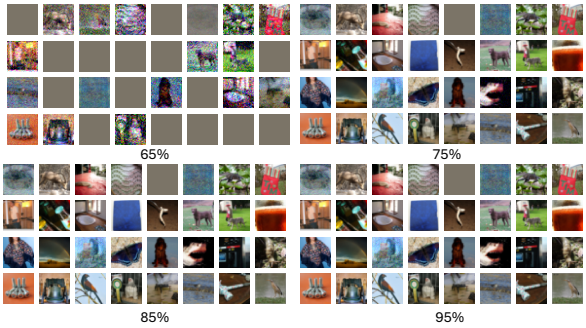


Figure 21: Visualization of attack effect under gradient sparsity-based defense.

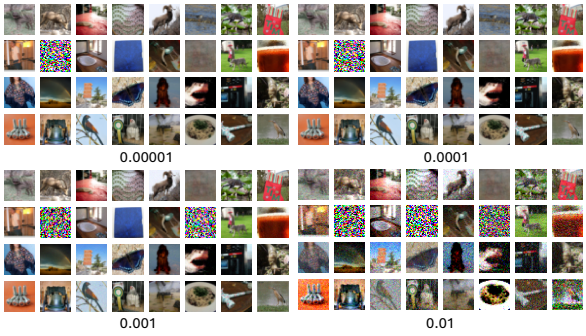


Figure 22: Visualization of attack effect under different differential privacy noise.

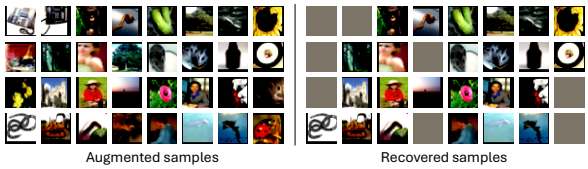


Figure 23: Visualization of the augmented samples and recovered samples.

Original: It was home to the Arkansas Museum of Natural History and Antiquities from 1942 to 1997 and the MacArthur Museum of Arkansas Military History since 2001.
Recovered: The original plans called for it to be built of stone, however, was home to the Arkansas Museum of Natural History and Antiquities from 1942 to 1997 and the MacArthur Museum of Arkansas Military History since 2001.

Original: Besides being the last remaining structure of the original Little Rock Arsenal and one of the oldest buildings in central Arkansas, it was also the birthplace of General Douglas MacArthur, who became the supreme commander of US forces in the South Pacific during World War II.
Recovered: Work began on the Tower War on the horizon, a company of the Second United States Artillery, consisting of sixty @-@ five men, was transferred to Little Rock under the command remaining structure of the original Little Rock Arsenal and one of the oldest buildings in central Arkansas, it was also the birthplace of General Douglas MacArthur, who became the supreme commander of US forces in the South Pacific during World War II.

Original: It was also the starting place of the Camden Expedition.
Recovered: It was also the starting place of the Camden Expedition.

Figure 24: Visual illustration of the recovery effect on the Wikitext dataset. Red text indicates the matching tokens.

Appendix D.
Meta-Review

D.1. Summary

This paper proposes ARES, an active gradient inversion attack against FL that reconstructs private training data by formulating activation inversion as a noisy sparse recovery problem. The attack leverages existing fully-connected layers and imprint-based activation disentanglement to successfully extract data at scale without requiring modifications to the model architecture.

D.2. Scientific Contributions

- Identifies an Impactful Vulnerability.
- Provides a Valuable Step Forward in an Established Field.

D.3. Reasons for Acceptance

1) This paper identifies an impactful vulnerability. It demonstrates that intermediate activations can be exploited at scale without architectural modifications, which poses a significant practical privacy threat in FL systems.
2) The paper provides a valuable step forward in an established field. Although gradient inversion attacks are known, this work relaxes previous assumptions and addresses scalability and practicality issues. Reformulating the problem and combining techniques like noisy sparse recovery is non-trivial, and strong empirical evaluations demonstrate superiority over prior work.

D.4. Noteworthy Concerns

1) There are questions regarding the attack's scalability to more modern, deeper architectures like ResNet or ViT, as well as its robustness when using alternative activation functions that do not induce sparsity like ELU or GELU.
2) The paper lacks a robustness discussion on attack detection and targeted defenses. It remains unclear if the attack's parameter initializations are easily detectable, or how it would handle lightweight defenses like homomorphic encryption applied to the fully connected layer.