Scale-MIA: A Scalable Model Inversion Attack against Secure Federated Learning via Latent Space Reconstruction

Shanghao Shi[*], Ning Wang[†], Yang Xiao[‡], Chaoyu Zhang[*], Yi Shi[*], Y. Thomas Hou[*], and Wenjing Lou[*]
[*]Virginia Tech, [†]University of South Florida, [‡]University of Kentucky
[*]{shanghaos, chaoyu, yshi, thou, wjlou}@vt.edu, [†]ningw@usf.edu, [‡]xiaoy@uky.edu

***Abstract*—Federated learning is known for its capability to safeguard the participants' data privacy. However, recently emerged model inversion attacks (MIAs) have shown that a malicious parameter server can reconstruct individual users' local data samples from model updates. The state-of-the-art attacks either rely on computation-intensive iterative optimization methods to reconstruct each input batch, making scaling difficult, or involve the malicious parameter server adding extra modules before the global model architecture, rendering the attacks too conspicuous and easily detectable.**

To overcome these limitations, we propose Scale-MIA, a novel MIA capable of efficiently and accurately reconstructing local training samples from the aggregated model updates, even when the system is protected by a robust secure aggregation (SA) protocol. Scale-MIA utilizes the inner architecture of models and identifies the latent space as the critical layer for breaching privacy. Scale-MIA decomposes the complex reconstruction task into an innovative two-step process. The first step is to reconstruct the latent space representations (LSRs) from the aggregated model updates using a closed-form inversion mechanism, leveraging specially crafted linear layers. Then in the second step, the LSRs are fed into a fine-tuned generative decoder to reconstruct the whole input batch.

We implemented Scale-MIA on commonly used machine learning models and conducted comprehensive experiments across various settings. The results demonstrate that Scale-MIA achieves excellent performance on different datasets, exhibiting high reconstruction rates, accuracy, and attack efficiency on a larger scale compared to state-of-the-art MIAs. Our code is available at https://github.com/unknown123489/Scale-MIA.

I. Introduction

Federated learning (FL) is a distributed learning framework that enables its participants to collaboratively train a machine learning model without sharing their individual datasets [1]. Within this framework, the training process occurs iteratively between a central parameter server and a group of clients. During each training round, the parameter server first broadcasts a global model with a pre-agreed model architecture to all or a fraction of clients. The server then collects and aggregates the model updates (either gradient or parameter updates) submitted by the clients, which are trained and derived from their respective local datasets. As no individual training data samples are exchanged between participants in this process, FL is widely recognized as a communication-efficient and privacy-preserving learning paradigm.

A. Privacy Leakage of Federated Learning

Unfortunately, recent research reveals that the privacy of FL is susceptible to breaches, enabling the attackers to infer information about the clients' proprietary datasets [2]. Of particular concern are the *model inversion attacks (MIAs)* [3], [4], [5], [6], [7], [8], [9], in which the adversary tries to reconstruct the original training samples from the model updates submitted by the clients. In these attacks, the parameter server is considered an *honest-but-curious attacker* whose goal is to closely approximate the input samples by minimizing the distance between real gradients uploaded by individual clients and those generated by approximated dummy samples. These attacks can successfully reconstruct high-fidelity training samples when the server gains access to *individual model updates* and undergoes sufficient optimization iterations.

To counter these attacks, Bonawitz et al. propose the *secure aggregation (SA)* protocol [10] to prevent the server from gaining knowledge about individual model updates. SA is a specialized secure multi-party computation (MPC) protocol that allows the server to compute the summation of model updates without knowing individual values. SA ensures that individual model updates are cryptographically masked and the server cannot distinguish them from *random numbers*. SA is considered one of the most robust defense mechanisms against various inference attacks targeting federated learning systems [11], and several follow-up works have been proposed to further reduce the communication and computation overhead of the original SA protocol [12], [13], [14], [15].

Despite SA's initial security guarantees, recent privacy attacks show that the SA protocol is breakable when the attacker can modify the model parameters or architectures, which *goes beyond the honest-but-curious threat model*. Two distinct attack strategies have been identified to break the SA protocol. The first strategy involves obtaining individual model updates from the aggregated results by carefully manipulating the global model parameters and having the target client's model update dominate the aggregated results. This can be

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240644
www.ndss-symposium.org

achieved by eliminating the model updates of all other clients except the target [16], or amplifying only the gradients of the target victim [17]. After obtaining individual model updates, the attacker can utilize the existing optimization-based MIAs to reconstruct input samples. However, these attacks still require costly optimization-based MIAs as part of their attack flows, limiting their applicability at scale.

The second strategy involves a more direct approach to reconstructing the input samples just from the aggregated results. To accomplish this, existing work requires the attacker to insert either a two-layer linear module [18] or a convolutional module [19] before the pre-agreed global model architecture, as well as possessing a representative auxiliary dataset. These modules are meticulously crafted or trained using the auxiliary dataset, enabling the attacker to reconstruct the inputs from the gradients of crafted layers within these modules, using customized analytical methods. However, modifying the pre-agreed model architecture is highly conspicuous and is unlikely to be accepted by the clients.

B. Our Attack

In this paper, we present a novel model inversion attack named Scale-MIA to break the secure aggregation (SA) protocol in FL. Scale-MIA is designed to be scalable, stealthy, efficient, and highly effective, overcoming the deficiencies of existing attacks. It is capable of accurately reconstructing the clients' local data samples from the aggregated results, requires no modifications to the pre-agreed model architecture, and is more difficult to detect. Scale-MIA also eliminates the costly per batch or sample search-based optimization process. This enables the reconstruction of hundreds of data samples in parallel and results in significant computational speed-up.

The proposed Scale-MIA involves two distinct phases—the *adversarial model generation* phase and the actual *input reconstruction* phase. The adversarial model generation can be done offline by the malicious parameter server once at the outset. The purpose is to generate an adversarial global model that follows the same architecture as the pre-agreed model architecture that will be distributed to the clients during the FL iterations. More specifically, the attacker first trains a surrogate autoencoder, with its encoder having *the same* architecture to the encoder of the real global model, and a customized generative decoder capable of reconstructing the inputs, utilizing a collected auxiliary dataset. Subsequently, the attacker feeds the auxiliary dataset to the already-trained encoder, enabling the estimate of essential statistical parameters for crafting linear layers in the adversarial global model. Finally, the adversarial global model is assembled by having its encoder identical to the surrogate autoencoder, and the following linear layers are crafted with the estimated parameters.

In the second phase, the attacker disseminates the crafted adversarial global model to clients and awaits local updates from them. Assuming the presence of the SA protocol, the attacker only receives the aggregated model updates from these clients. The proposed input reconstruction phase takes the aggregated model updates as input and aims to reconstruct as many local samples as possible. We design a novel model inversion method, in which we decompose the input reconstruction phase into two steps, both only involving efficient matrix computation and feed-forward neural network computations to reduce its complexity, making it super efficient to conduct. More specifically, we first disaggregate the received aggregated model update into batched latent space representations (LSRs) through a closed-form *linear leakage* module and then feed these representations into the pre-trained generative decoder to reconstruct the original input batch. This phase can be executed in a single federated learning round and is adaptable for launch at any desired time, such as during the FL training initialization. Our attack does not cause significant impacts on the training performance and can be launched repeatedly in multiple FL rounds to harvest as many local training samples as possible.

Three factors could significantly impact the performance of our attack—the reconstruction number is restricted by the neuron number of the first linear layer in the latent space; the reconstruction quality is sensitive to the quantity and quality of the auxiliary dataset; and the reconstruction rate relies on the availability of a good statistical estimation of the distribution of LSRs. Fortunately, in practical scenarios, all three factors do not pose a significant barrier for the attacker. Popular machine-learning models commonly feature large linear layers; auxiliary datasets can be easily collected from various online resources and public datasets; and data representations in latent space often follow a Gaussian distribution and are easy to estimate, contributing to better attack performance.

We conducted extensive experiments to evaluate the performance of the proposed attack on the Fashion MNIST (FMNIST) [20], Colorectal Histology MNIST (HMNIST) [21], CIFAR-10 [22], TinyImageNet [23], CelebA [24], and ImageNette [23] datasets. A thorough comparison was made between our attack and existing MIAs and the results show a significant improvement in terms of attack accuracy, reconstruction fidelity, and efficiency when using Scale-MIA. We also evaluated Scale-MIA's performance under different data settings, including variations in data amount and whether the data was iid or non-iid. The results consistently highlighted the success of Scale-MIA across all these settings. Notably, the results show that the attacker can successfully carry out a targeted attack using their collected dataset focused on a specific class.

C. Contributions

This paper makes the following contributions:

1) We identify the latent space of a machine learning model as the pivotal layer for launching an MIA to breach user data privacy in federated learning systems. This insight motivates us to focus on the privacy risks posed by individual components within the model architecture, enabling us to devise a more efficient and effective attack strategy from a white-hat attacker's perspective.
2) We propose Scale-MIA, a novel MIA launched by a malicious parameter server. Scale-MIA efficiently and accurately reconstructs a large batch of user data samples from the aggregated model updates, given that the

federated learning system is under the protection of a robust secure aggregation protocol.

3) Compared to existing attacks, Scale-MIA demonstrates significantly improved efficiency and scalability as it removes the requirement for expensive per-batch search-based optimization. Moreover, Scale-MIA is stealthier in its approach, as it does not require any modifications to the global model architecture and can be accomplished within one FL training round.
4) We provide a comprehensive analysis and evaluate the key factors that significantly impact the performance of Scale-MIA. Alongside our analysis, we present several practical attack scenarios of Scale-MIA to promote the need for novel defenses against such advanced attacks.
5) We conducted extensive experiments to evaluate the performance of Scale-MIA under diverse settings. We examined Scale-MIA's performance on popular model architectures including Alexnet, VGGnet, ResNet, and ViT, as well as various datasets. The results show the effectiveness, efficiency, and scalability of our attack.

In Table I, we summarize the definitions and notations used in our paper.

II. Background and Related Work

A. Federated Learning

Federated learning (FL) allows a set of clients $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$ to train a global model $G = f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ on a global dataset $\mathcal{D} = \cup_{i=1}^{n} D_i$ that is distributed along the users where each client c_i holds a local dataset D_i without the need to share these data samples. FL is conducted iteratively in rounds until the model parameter θ converges. In each round t, the parameter server S first publishes the global model parameter θ^t to a subset of selected clients $\mathcal{C}^t \subseteq \mathcal{C}$. Then these clients compute the gradients with their local batches D_i^t as $g_i^t = \frac{1}{|D_i^t|}\nabla L(\theta_t, D_i^t)$, where $L()$ refers to the loss function. The clients send their computed updates back to S and the latter will aggregate the updates with the FedSGD algorithm [1]:

$$\theta^{t+1} = \theta^t - \eta \sum_{i:c_i \in \mathcal{C}^t} \frac{\alpha_i}{|D_i^t|} \nabla L(\theta_t, D_i^t) \tag{1}$$

where η is the global learning rate and α_i is the weight assigned to client c_i. The summation of all weights $\{\alpha_i\}_{i:c_i \in \mathcal{C}^t}$ is 1 and can be adjusted according to the size of local datasets D_i^t to avoid training bias. The clients can also train the received global model G_t for L_i^t local rounds before providing the model updates δ_i^t to the server. In this case, the server employs the FedAVG algorithm to conduct the training process:

$$\theta^{t+1} = \sum_{i:c_i \in \mathcal{C}^t} \alpha_i \delta_i^t \tag{2}$$

In the following sections, we will omit the notation t because our attack is a single-round attack and can be launched in any FL training round.

TABLE I: Definition and notations.

Symbol	Definition
c_i	Federated learning clients
n	Number of federated learning clients
G	Global model
θ	Global model parameters
t	Training round
m	Input batch size
D_i	Local datasets
g_i	Individual gradients
δ_i	Individual model updates
u_i	Masked model updates
SA	Secure Aggregation
x_i	Input samples
$\hat{x_i}$	Reconstructed samples
L	Loss function
$W^{[2]}$	Two-layer linear leakage module
D_{Adv}	Auxiliary dataset
G_{Adv}	Adversarial model
MLP	Multi-layer perception
$\hat{Enc}$	Surrogate encoder
$\hat{Dec}$	Generative decoder
$\hat{G}$	Modified global model
LSR	Latent Space Representation
CDF	Cumulative Density Function

B. Gradient Inversion

The gradient inversion problem is to find a function that can reverse the individual gradient g_i uploaded by a client c_i back to the local dataset D_i under the FL setting, which can be defined as $D_i \overset{?}{=} Reverse(g_i, G)$.

Optimization-based Gradient Inversion: Recent research shows that a *honest-but-curious* attacker can solve the gradient inversion problem by solving the following optimization problem:

$$\arg\min_{\hat{D}_i} [d(\nabla \hat{D}_i - \nabla D_i) + r(\hat{D}_i)] \tag{3}$$

where $\hat{D}_i$ refers to randomly initialized dummy samples, $d()$ refers to the distance function, and $r()$ refers to the regulation function. Zhu et al. [3] first identify this problem, and propose the deep leakage from gradient (DLG) attack which chooses the second norm as the distance function, and uses the L-BFGS optimizer [26] to solve the optimization problem. Then Zhao et al. [4] improve this attack by proposing an analytical method to recover ground-truth labels from the gradients that help DLG achieve better performance. Geiping et al. [5] further improve the optimization tool and achieve better image reconstruction fidelity, but requires a strong assumption as the labels of the inputs must be known. Yin et al. [7] focus on reconstructing batched inputs on the ImageNet dataset and ResNet model architecture, making the attack more practical. Hatamizadeh et al. [9] customize the attack for the vision transformer and achieve better performance than previous attacks. Dimitrov et al. [25] devise the attack applicable to the FedAVG setting, making it deployable on real systems.

However, these optimization-based gradient inversion attacks are computationally costly and need hundreds of optimization

TABLE II: A comparison between different federated learning model inversion attacks.

Attack	Break Secure Aggregation?	Attacker's Capability	Attack Overhead	Attack Scale	Need Auxiliary Dataset?	Model Agnostic?
DLG [3], iDLG [4]	No	Weak (Curious)	Large	Single image	No	Yes
Inverting Grad [5]	No	Weak (Curious)	Large	8	No	Yes
GradInversion [7]	No	Weak (Curious)	Large	48	No	No (ResNet)
GradViT [9]	No	Weak (Curious)	Large	8	No	No (ViT)
APRIL-Optim [8]	No	Weak (Curious)	Large	Single-image	No	No (ViT)
APRIL-Analytic [8]	No	Weak (Curious)	Small	Single-image	No	No (ViT)
R-GAP [6]	No	Weak (Curious)	Small	Single-image	No	Yes
Leak in FA [25]	No	Weak (Curious)	Small	50	No	Yes
Fishing for data [17]	Yes	Medium (Modify params)	Large	256	Yes	Yes
Eluding SecureAgg [16]	Yes	Medium (Modify params)	Large	512	Yes	Yes
Robbing the fed [18]	Yes	Strong (Change architect)	Small	1024+	Yes	Yes
LOKI [19]	Yes	Strong (Change architect)	Small	1024+	Yes	Yes
Scale-MIA	Yes	Medium (Modify params)	Small	1024+	Yes	Yes

iterations to reconstruct one input batch [11]. Many of them (e.g., [3], [4], [5], [9], [25]) only perform well for a single or small batch of images (typically smaller than 16), representing a scalability challenge.

Closed-form Gradient Inversion: Instead of using the costly optimization approach, some other works seek to solve the gradient inversion problem with closed-form derivation. Aono et al. [27] analyze the privacy leakage of linear models and show that the inputs to linear layers can be perfectly reconstructed from its gradients, which is referred to as the "linear leakage". This primitive is later revised and used as a component of many advanced attacks [18], [17], [19] including our work. Zhu et al. [6] propose an analytical recursive attack on privacy (R-GAP) to reconstruct the input layer by layer back from the output layer, particularly targeting linear and convolutional neural networks (CNNs). Lu et al. [8] propose an analytical reconstruction attack specialized for the vision transformer (ViT) and can reconstruct high-fidelity images. Unfortunately, all of these analytical attacks are designed for specific model architectures and cannot be generalized for others. Furthermore, they are designed for single gradient inversion and cannot reconstruct large input batches.

C. Secure Aggregation

Previous gradient inversion attacks rely on the assumption that the attacker (i.e., the parameter server) can obtain the individual gradients $\{g_i\}$ from clients, especially for cross-device federated learning. As a countermeasure, Bonawitz et al. [10] propose the secure aggregation (SA) protocol that masks the original model updates from clients. Specifically, SA is a multi-party computation (MPC) protocol that masks the original model updates δ_i with random bits from secret sharing but keeps the summation of masked updates $\sum_{i=1}^{n} u_i$ equals to $\sum_{i=1}^{n} \delta_i$. Mathematically, this can be defined as:

$$f^{SA}(\delta_1, \delta_2, \cdots, \delta_n) = (u_1, u_2, \cdots, u_n) \\ s.t. \sum_{i=1}^{n} u_i = \sum_{i=1}^{n} \delta_i \tag{4}$$

where f^{SA} refers to the abstract function of the SA protocol and the attacker cannot distinguish u_i from a random number. This implies that nothing more than the final aggregated result is leaked to the attacker. Because the final result is aggregated from all training samples submitted by all participants, the previous gradient inversion attacks cannot reconstruct meaningful information from such large input batches. The SA protocol is also communication efficiency and drop-out resilience, making it one of the most robust defense mechanisms against federated learning privacy inference attacks. The follow-up works further improve the performance of the original SA protocol by reducing the communication and computation overheads [12], [28], [13], [14], enabling verifiable aggregation [15], and bolstering the robustness of the secure aggregation against malicious attacks [29], [30], [31], [32].

D. Breaking the SA

However, when assuming the parameter server is malicious and capable of modifying the global model G 's parameters, the SA is breakable. Two general types of attacks have been identified, including the *gradient disaggregation attacks*, aiming to overturn SA's main function by inferring individual model updates δ_i from the aggregated result $\sum_{i=1}^{n} \delta_i$ with crafted model $\hat{G}$, i.e., $\delta_i = Infer(\sum_{i=1}^{n} \delta_i, \hat{G})$; and the *large batch reconstruction attack* that aims to directly reconstruct the global batch $\cup_{i=1}^{n} D_i$ from the aggregated results $\sum_{i=1}^{n} \delta_i$ with the help of additional adversarial module M_{adv} placing in front of global model G and a representative auxiliary dataset D_{aux}, i.e., $\cup_{i=1}^{n} D_i = Reverse(\sum_{i=1}^{n} \delta_i, G \oplus M_{adv}, D_{aux})$.

Gradient Disaggregation Attacks: Wen et al. [17] propose a "fishing strategy" that magnifies the gradient of a targeted class to dominate the aggregated result with crafted model parameters. The attack generates a close enough approximation of the target gradient out of the final aggregated gradient, which is enough for the attacker to reconstruct input samples through existing optimization-based gradient inversion methods. Pasquini et al. [16] propose a gradient suppression attack that zeros out all the gradient updates except the target victim's, making the final aggregated result identical to the target's gradient. The attack achieves this by crafting the parameters of a single linear layer and keeping the outputs of that specific layer always smaller than zero, which further leads to zero gradients if the ReLU activation function is used. The key limitation of this type of attack is they still use the existing optimization-based gradient inversion methods as their attack component, resulting in poor scalability performance and large computation costs.

Large Batch Reconstruction Attacks: Directly reconstructing the whole input batch from the aggregated result is a challenging task and the existing attacks [18], [19] usually have strong assumptions to accomplish it, including allowing the attacker to modify the pre-agreed model architecture and possessing an auxiliary dataset that has a similar distribution as the training dataset. Fowl et al. [18] modify the global model architecture by attaching an adversarial two-linear-layer module in the front. The attacker can leverage the "linear leakage" primitive to perfectly reconstruct the original inputs with large batch sizes by customizing the parameters of the adversarial module, which are generated from the statistics of the auxiliary dataset. Zhao et al. [19] improve this attack by changing the linear module to a customized convolutional module. As a result, the attacker can recover large batches under a more practical FedAVG setting and can help to identify the belongings of the reconstructed samples. The key drawback of these attacks is they require the attacker to change the pre-agreed model architecture, which can be easily detected when the clients employ some integrity-checking mechanisms.

We summarize the pros and cons of existing attacks in Table II. We compare the existing MIAs with our proposed attack concerning the attack assumption, overhead, and performance. Notably, our proposed Scale-MIA scales significantly better than the existing optimization-based gradient inversion attacks [3], [4], [5], [7], [9], [8] along with a small attack overhead, being model-agnostic, and the ability to launch a targeted attack against a certain class. Compared to the gradient disaggregation attacks [17], [16], Scale-MIA does not involve any per-batch optimization process and thus reduces the attack overhead (Scale-MIA can be used to replace the costly optimization-based inversion process after the individual gradient is obtained). Compared to the existing large-batch reconstruction attacks [18], [19], Scale-MIA assumes weaker attacker capability and is more stealthy and harder to detect.

III. Threat Model

In this section, we formalize the attacker's capability and goal. We assume the parameter server is malicious and knows the global model G and its parameters θ of every round. We consider the state-of-the-art SA protocol (as used in [10], [12]) is in place and the attacker only sees the already masked model updates $\{u_i\}_{i=1}^n$ from the clients rather than the original ones $\{\delta_i\}_{i=1}^n$. The attacker cannot distinguish u_i from a random number but he can obtain the aggregated model update $\sum_{i=1}^n \delta_i$ through summing the masked inputs $\sum_{i=1}^n \delta_i = \sum_{i=1}^n u_i$. We assume the communications between the parameter server and clients are secure and no third party can alter the transmitted messages between them. We assume the attacker is able to modify the global model G's parameters but **not architecture** and knows global training configurations such as the learning rate η and weights α_i, as in [16] and [17]. We also assume the attacker possesses an auxiliary dataset D_{Adv} that has similar data distribution as training data D_{Train}, following the same assumptions in [18], [19]. In practice, the attacker can collect this D_{Adv} by using the existing public resources, manually collecting samples, or even colluding with a fraction of clients.

The attack aims to achieve the same goal as [18], [19], which is to recover the whole global input batch $\cup_{i=1}^n D_i$ efficiently and precisely, i.e. $\cup_{i=1}^n D_i = Reverse(\sum_{i=1}^n \delta_i, \hat{G}, D_{Adv})$. We illustrate our threat model in Figure 1.

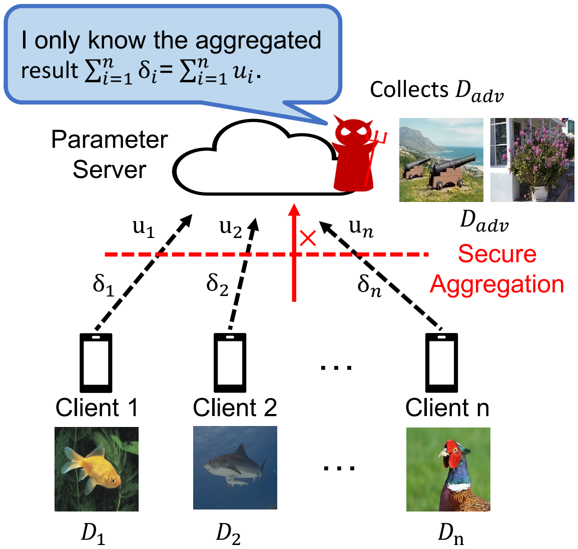


Fig. 1: Scale-MIA threat model.

IV. Attack Preliminaries

A. Autoencoder

Autoencoder is an unsupervised learning technique that helps to learn an informative and compressed data representation [33], [34], [35]. The autoencoder's model architecture contains an encoder Enc and a decoder Dec. It is trained to minimize the difference between the original inputs and the reconstructed outputs, i.e., $\arg\min_\theta d(x, Dec(Enc(x)))$, where θ is the autoencoder's parameter vector and $d()$ refers to the error function such as the mean square error. As a result, a fine-tuned autoencoder can almost perfectly reconstruct its model inputs at the outputs, i.e. $x \approx Dec(Enc(x))$, even for the batched inputs $\cup_{j=1}^m x_j$. More specifically, a well-trained autoencoder consists of an encoder that encode the input samples to their latent space representations (LSRs), i.e. $\cup_{j=1}^m LSR_j = Enc(\cup_{j=1}^m x_j)$, and a decoder that decodes the LSRs to samples, i.e. $\cup_{j=1}^m \hat{x_j} = Dec(\cup_{j=1}^m LSR_j)$ and $\hat{x_j}$ satisfies $\hat{x_j} \approx x_j$.

B. Linear Leakage

Linear leakage refers to a mathematical property that a model with two subsequent linear layers, denoted by $W^{[2]}$, with a non-linear activation function (e.g. ReLU) in between can

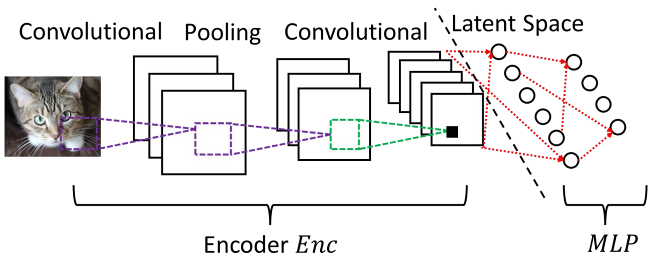


Fig. 2: The model architecture of machine-learning classifiers.

be crafted to *perfectly reconstruct* its batched inputs $\cup_{j=1}^{m} x_j$ from the aggregated gradients of the layers $\sum_{j=1}^{m} g_j^{[2]}$ with the help of a representative auxiliary dataset D_{aux}, i.e. $\cup_{j=1}^{m} x_j = LinearLeak(\sum_{j=1}^{m} g_j^{[2]}, W^{[2]}, D_{aux})$ [18].

To describe this property, we define the linear module as:

$$\begin{aligned} y &= \delta(w_1 x + b_1) \\ z &= w_2 y + b_2 \end{aligned} \tag{5}$$

where $x \in \mathbb{R}^d$, $y \in \mathbb{R}^k$, $z \in \mathbb{R}^o$, $w_1 \in \mathbb{R}^{k \times d}$, $b_1 \in \mathbb{R}^k$, $w_2 \in \mathbb{R}^{o \times k}$, $b_2 \in \mathbb{R}^o$, and δ is the ReLU function. Suppose the attacker can accurately estimate the CDF of feature $h(x) = v_h.x$ of the input dataset as $\psi(h(x))$ from the auxiliary dataset D_{aux}, where $v_h \in \mathbb{R}^{1 \times d}$. Suppose the loss function is $L(x;\theta)$ or simply L where θ refers to the module parameters. The input batch size is m and can be expressed as $\cup_{i=1}^{m} x_j = [x_1, x_2, \cdots, x_m]$.

The attacker can craft the linear leakage module in the following steps: (1) Having the row vectors $w_{1(r)}, r \in \{1, 2, \cdots, k\}$ of weight matrix w_1 all identical to v_h; (2) dividing the distribution of the feature $h(x)$ into equally k bins by calculating $h_i = \psi^{-1}(\frac{i}{k})$, which results in having a random variable h the same probability to falls in each bin $[h_s, h_{s+1}]$; (3) assigning the bias vector b_1 identical to the opposite values of h vector $[-h_1, -h_2, \cdots, -h_k]$; and (4) letting the row vectors of weight matrix w_2 be identical. After this, the attacker conducts the FL training process to obtain the aggregated gradients and then calculates the following equation to create k bins to reconstruct the input samples, for $r \in \{1, 2, \cdots, k\}$:

$$(\nabla_{w_1(r+1)} L - \nabla_{w_1(r)} L)/(\nabla_{b_1(r+1)} L - \nabla_{b_1(r)} L) \tag{6}$$

where specially we have $\nabla_{w_1(k+1)} L$ and $\nabla_{b_1(k+1)} L$ equal zero.

When batch size m is smaller than neuron number k, each input sample will only activate and be reconstructed by one bin. More specifically, sample x_p as the p^{th} smallest one in terms of feature $h(x)$ that falls in the l^{th} bin $[h_l, h_{l+1}]$ alone will be reconstructed by Eq. 6 when $r = l$:

$$\begin{aligned} \frac{\nabla_{w_1(l+1)} L - \nabla_{w_1(l)} L}{\nabla_{b_1(l+1)} L - \nabla_{b_1(l)} L} &= \frac{\frac{\partial L}{\partial y_{l+1}} \frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} - \frac{\partial L}{\partial y_l} \frac{\partial y_{(l)}}{\partial w_{1(l)}}}{\frac{\partial L}{\partial y_{l+1}} \frac{\partial y_{(l+1)}}{\partial b_{1(l+1)}} - \frac{\partial L}{\partial y_l} \frac{\partial y_{(l)}}{\partial b_{1(l)}}} \\ &= \frac{\sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_l} x_v}{\sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_l}} \\ &= \frac{\sum_{v=1}^{p} \frac{\partial L}{\partial y_l} x_v - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_l} x_v}{\sum_{v=1}^{p} \frac{\partial L}{\partial y_l} - \sum_{v=1}^{p-1} \frac{\partial L}{\partial y_l}} \\ &= \frac{\frac{\partial L}{\partial y_l} x_p}{\frac{\partial L}{\partial y_l}} = x_p \end{aligned} \tag{7}$$

Note that we have leveraged the property of $\nabla_{w_1(l+1)} L = \frac{\partial L}{\partial y_{l+1}} \frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} = \sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v$, and $\frac{\partial L}{\partial y_{l+1}} = \frac{\partial L}{\partial y_l}$. Their detailed mathematical proof can be found in the Appendix.

On the other hand, when batch size m is larger than neuron number k, linear leakage cannot ensure that one sample is only activated and reconstructed by one bin. In some bins, the samples collide with each other and are mixed together during the reconstruction process, leading to reconstruction failures. Therefore, we regard the neuron number k as the performance bottleneck of the linear leakage primitive.

Linear Leakage in Federated Learning: In the federated learning system with SA, the parameter server S needs to infer the aggregated gradients $\sum_{j=1}^{m} g_j$ from the aggregated model updates $\sum_{i=1}^{n} \delta_i$ in order to launch the linear leakage attack. For the FedSGD system, the two values are identical as $\sum_{i=1}^{n} \delta_i = \sum_{j=1}^{m} g_j$. But for the FedAVG system, each model updates δ_i are trained by clients for several local rounds. Assuming the clients employ the SGD algorithm for local training, the server can only get approximated aggregated gradients $\sum_{j=1}^{m} \hat{g}_j$ from the aggregated model updates $\sum_{i=1}^{n} \delta_i$ with analytical tools, resulting in a slightly decreased linear leakage performance, i.e. $\cup_{j=1}^{m} x_j \approx LinearLeak(\sum_{i=1}^{n} \delta_i, W^{[2]}, D_{aux})$.

V. Attack Method

A. Attack Intuition

Linear leakage provides us with a powerful primitive to reconstruct samples. A straightforward attack strategy is to place a crafted linear leakage module $W^{[2]}$ right in front of the global model G as $G \oplus W^{[2]}$ and publish it to the clients. As a result, when the server receives the aggregated gradients $\sum_{i=1}^{n} g_i^{[2]}$ from clients, it can reconstruct the input samples with the primitive [18]. However, as we have discussed in Section II, it is too suspicious and can be easily prevented by integrity checking as the attacker needs to change the global model architecture. We abandon this architectural modification approach and examine the *built-in components* of models for potential attack exploitation.

We observe that machine learning classifiers are commonly composed of a feature extraction encoder Enc followed by a multi-layer perceptron (MLP) in their model architectures, as exemplified in Figure 2. Motivated by this, we target the *latent space* as the key layer to launch the attack based on the following reasons.

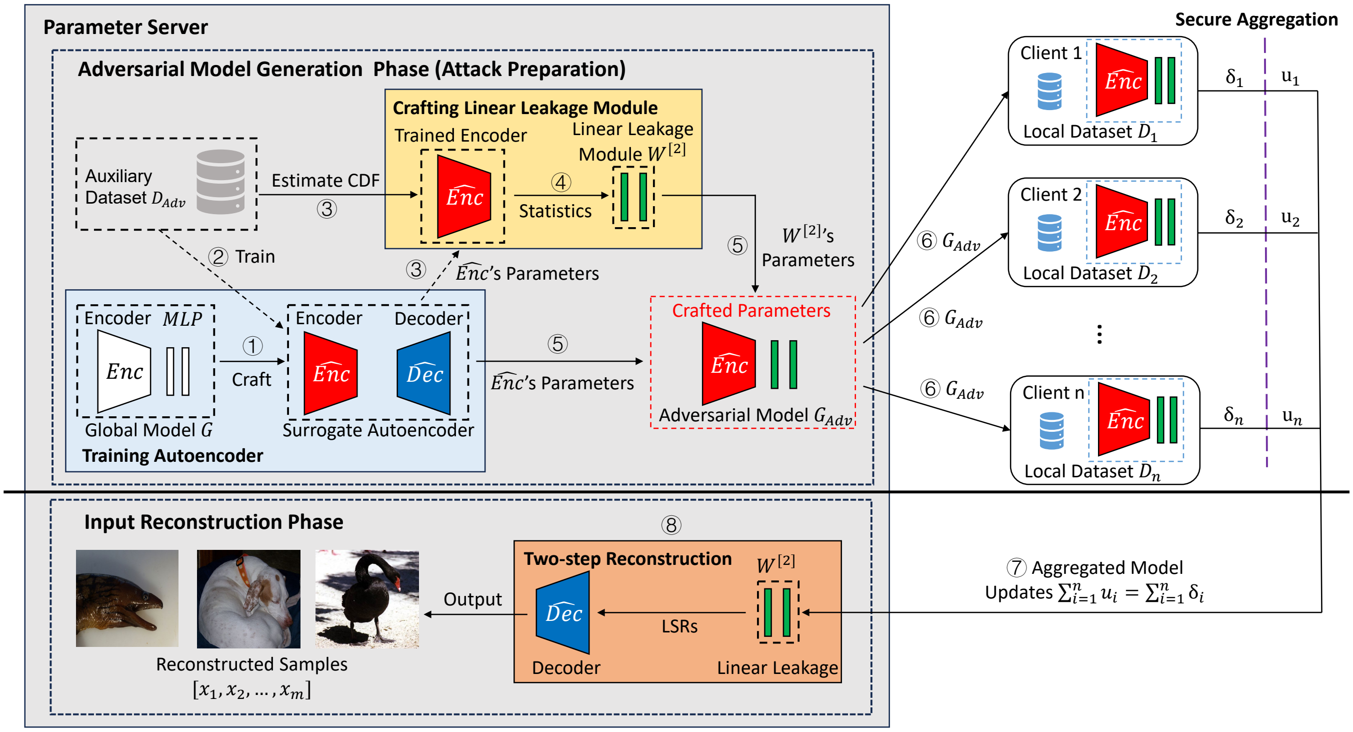


Fig. 3: Scale-MIA is a two-phase attack. The first phase is performed locally to produce essential information to conduct the second phase. The second is the actual attack phase, during which the attacker interacts with the clients and reconstructs their local training samples.

1) LSRs contain enough information to reconstruct the inputs and are widely regarded as the "information bottleneck" within the whole model architecture.
2) LSRs have relatively lower dimensions and can be processed more efficiently.
3) In most machine learning models, LSRs are followed up by MLPs, which can be exploited to launch the analytical linear leakage primitive.

a) Problem Decomposition: Based on our previous findings, we decompose the original complex reconstruction task $\cup_{i=1}^{n} D_i = Reverse(\sum_{i=1}^{n} \delta_i, \hat{G}, D_{Adv})$ into two sub-problems, including firstly reconstruct the LSRs with the linear leakage primitive from the crafted MLPs (in terms of parameters) in the latent space, i.e. $\cup_{j=1}^{m} LSR_j = Reverse(\sum_{i=1}^{n} \delta_i, \hat{W^{[2]}}, D_{Adv})$, then reconstruct the training samples by feeding these LSRs into a fine-tuned decoder, i.e. $\cup_{j=1}^{m} x_j = Dec(\cup_{j=1}^{m} LSR_j)$, where $\cup_{j=1}^{m} x_j$ is identical to $\cup_{i=1}^{n} D_i$. Both two steps only involve matrix computation and feed-forward neural network computations, making the attack super efficient.

B. Attack Overview

We illustrate the attack flow of Scale-MIA in Figure 3. Scale-MIA consists of two main phases: the adversarial model generation phase (attack preparation, Steps ①-⑤) and the input reconstruction phase (Steps ⑥-⑧). The first phase is conducted locally on the parameter server S and its purpose is to generate an adversarial global model G_{adv} with crafted parameters, as well as a highly accurate generative decoder $\hat{Dec}$. This phase contains two sub-functions including training a crafted surrogate autoencoder $\hat{A}$ and crafting a linear leakage module $W^{[2]}$. $\hat{A}$ is trained with the auxiliary dataset D_{Adv} and has the same encoder architecture $\hat{Enc}$ as the global model G's encoder Enc, as well as a customized decoder $\hat{Dec}$ that has the capability to reconstruct the inputs. $W^{[2]}$ is crafted on the MLP layers of the global model G, whose necessary parameters and statistics are produced through feeding the auxiliary dataset D_{Adv} to the already-trained encoder $\hat{Enc}$. Finally, the adversarial model G_{Adv} is assembled with the surrogate encoder $\hat{Enc}$ and the linear leakage module $W^{[2]}$. By doing so the adversarial model G_{Adv} has the same architecture as the original global model G but with the necessary parameters to launch our reconstruction attack. This phase is conducted completely offline and covers all the required training efforts.

For the input reconstruction phase, the attacker first distributes the adversarial model G_{Adv} to the clients and awaits their feedback. After receiving the feedback, the attacker examines the model updates of the MLP layers in the adversarial global model to first reconstruct the batched latent space representations (LSRs) through the linear leakage primitive, and then reconstruct the input samples by feeding the LSRs to the trained decoder $\hat{Dec}$. All the operations involved in this actual attack phase only involve linear-complexity calculations and the reconstruction is significantly accelerated.

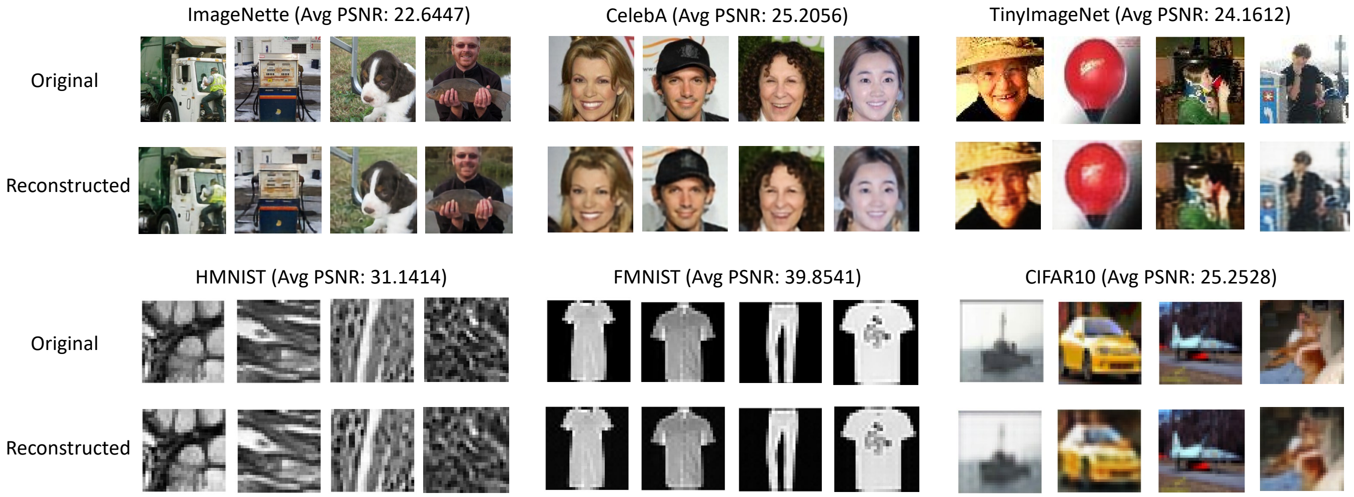

ImageNette (Avg PSNR: 22.6447)
CelebA (Avg PSNR: 25.2056)
TinyImageNet (Avg PSNR: 24.1612)
Original
Reconstructed
HMNIST (Avg PSNR: 31.1414)
FMNIST (Avg PSNR: 39.8541)
CIFAR10 (Avg PSNR: 25.2528)
Original
Reconstructed

Fig. 4: Reconstruction examples. These examples are taken from large reconstruction batches. Full reconstructed batches and more discussions can be found in the Appendix.

C. Detailed Workflow

Adversarial Model Generation: We assume that the global model G has an architecture consisting of an encoder Enc followed by a multi-layer perception (MLP). This assumption is practical and commonly observed in popular image classification models such as CNN-based AlexNet, VGGNet, ResNet, and Vision Transformers. In step ①, the attacker crafts a surrogate autoencoder $\hat{A}$ consisting of an encoder $\hat{Enc}$ and a decoder $\hat{Dec}$. The encoder $\hat{Enc}$ has the same model architecture as the global model's encoder Enc and the decoder $\hat{Dec}$ is constructed according to the model architecture of $\hat{Enc}$. For example, for an encoder with several convolutional layers, the decoder can have several de-convolutional layers to reconstruct the input. In step ②, the attacker trains the surrogate autoencoder $\hat{A}$ using the auxiliary dataset D_{Adv} with the objective of minimizing the distance between its inputs and outputs. D_{Adv} can be a publicly available dataset or a custom-collected dataset targeting a specific purpose or victim. In step ③, the attacker feeds the auxiliary dataset D_{Adv} to the already trained encoder $\hat{Enc}$ to get the LSRs (LSR_{Adv}) of the dataset. The attacker estimates the CDF of the ***brightness*** (the average value among all pixels) of the LSR_{Adv} and calculates the corresponding bias vector $H = [-h_1, -h_2, \cdots, -h_k]$ of the k bins according to the "linear leakage" primitive we described. Here k is identical to the neuron number of the first MLP layer of the global model G. Then, in step ④, the attacker crafts a linear leakage module $W^{[2]}$ on the first two MLP layers with respect to H. More specifically, the attacker has all the elements of the weight matrix of the first layer w_1 identical to $\frac{1}{d}$, where d is the dimension of the LSRs; as well as having the bias vector of the first layer b_1 equals H and the row vectors of the second weight matrix w_2 identical. Finally, the attacker generates the adversarial global model G_{Adv} by having the parameters of G_{Adv}'s encoder identical to $\hat{Enc}$ and the parameters of the MLP layers identical to $W^{[2]}$.

Input Reconstruction: After generating the adversarial model G_{Adv}, the attacker publishes it in step ⑥ to all clients. Then according to the FL framework, in step ⑦ the clients send back their model updates δ_i. Because we assume the SA is in place, the server receives the aggregated model update $\sum_{i=1}^{n} \delta_i$ instead of individual ones. In step ⑧, the reconstruction module takes the aggregated model update as the input and first uses the linear leakage primitive to recover a batch of LSRs, i.e. $[LSR_1, LSR_2, \cdots, LSR_m] \approx LinearLeak(\sum_{i=1}^{n} \delta_i, W^{[2]}, D_{Adv})$, where m is the global batch size identical to the cardinality of $\cup_{i=1}^{n} D_i$. Then these LSRs are taken as the inputs to the trained decoder $\hat{Dec}$ and the attacker finally gets $\cup_{j=1}^{m} \hat{x_j} = \hat{Dec}(\cup_{j=1}^{m} LSR_j) = \hat{Dec}(\hat{Enc}(\cup_{j=1}^{m} x_j))$ as the reconstruction outputs. According to the mathematical property of the linear leakage primitive and autoencoder, $\hat{x_j}$ and x_j are identical or highly similar.

D. Efficacy and Efficiency Analysis

Performance Bottleneck: Three main factors affect the performance of Scale-MIA. The first one is the neuron number of the first linear layer k, subject to the linear leakage's constraints. However, the neuron numbers of the first linear layers of popular machine learning models are usually very large, typically in the scale of thousands (e.g. 4096 for ImageNet classifiers), which is enough for the attacker to reconstruct hundreds or even a few thousands of samples simultaneously. In practice, we observe that the attacker can achieve high reconstruction rates (≥ 0.7) when the batch size m is lower than $\frac{k}{2}$, i.e. $m < \frac{k}{2}$, instead of the theoretical threshold k because of imperfect estimations and noise. We observe more collided samples in different bins and gradually dropped reconstruction rates when m exceeds $\frac{k}{2}$. But we argue that both the theoretical k and practical $\frac{k}{2}$ thresholds are not hard bounds, meaning that the attacker can still reconstruct a portion of local samples even when these thresholds are exceeded, although may only with relatively low reconstruction rates (≤ 0.3).

TABLE III: The comparison between Scale-MIA and the existing MIAs. * and + means that we use different thresholds other than 18. For [5] we select the threshold to be 14 and for [7] select the threshold to be 10.

Attack	Metric	1	2	4	8	16	64	256
DLG [3] /iDLG [4]	PSNR (dB)	33.0844	–	–	–	–	–	–
	Time (s)	127.1043	–	–	–	–	–	–
	Rate (ratio)	0.634	–	–	–	–	–	–
InvertingGrad [5]	PSNR (dB)	15.8407	16.2223	15.4679	14.8693	–	–	–
	Time (s)	280.126	305.6257	477.1157	866.8414	–	–	–
	$Rate*$ (ratio)	0.1999	0.12	0.0833	0.0417	–	–	–
GradInversion [7]	PSNR (dB)	13.3059	12.7896	12.0550	11.1410	10.1029	–	–
	Time (s)	324.8003	341.6113	348.2077	377.3184	462.2559	–	–
	$Rate^{+}$ (ratio)	0.99	0.95	0.85	0.65	0.245	–	–
Robbing the Fed [18]	PSNR (dB)	150.568	147.9793	142.0058	134.6393	129.1871	112.3125	103.9919
	Time (s)	0.1042	0.1124	0.1137	0.1141	0.1169	0.1658	0.3192
	Rate (ratio)	0.89	0.925	0.91	0.8988	0.8981	0.8743	0.8335
Loki [19]	PSNR (dB)	-	85.8121	54.2885	43.1088	41.1882	40.4387	38.7883
	Time (s)	-	3.4335	4.4684	5.7417	8.6886	24.8163	99.3890
	Rate (ratio)	-	1.0	0.875	0.7666	0.7916	0.7344	0.7109
Scale-MIA	PSNR (dB)	29.4192	29.4162	29.3292	29.2977	29.1853	28.9188	27.2986
	Time (s)	0.01951	0.02154	0.02463	0.03060	0.04134	0.09541	0.2214
	Rate (ratio)	1.0	0.999	0.993	0.9927	0.992	0.9438	0.8464

The second factor is the quality of the auxiliary dataset D_{Adv}. Scale-MIA can achieve near-perfect reconstruction performance when D_{Adv} can represent the target training dataset D_{Train} well. In practice, the attacker can leverage various online resources including public-available datasets, image-searching tools, and even generative models such as GAN [36], [37], [38], and diffusion models [39], [40], [41] to help collect the auxiliary dataset. Note that the auxiliary samples do not necessarily need to be highly similar to the targets, all samples in the same format and class can help to improve the reconstruction performance. On the other hand, Scale-MIA enables the attacker to launch a targeted reconstruction even if the auxiliary dataset is biased, amount-deficient, and even skewed to some extent.

The third factor is whether the attacker can have a good estimation of the required statistics (i.e. the LSR distribution) to craft the linear leakage module. Fortunately, the LSRs in the latent space usually exhibit Gaussian or Laplace distribution and can be accurately estimated by the attacker. Furthermore, because the surrogate autoencoder's training process is fully controlled by the attacker, it can also use the variational autoencoder (VAE), a variant of the autoencoders to regulate the LSRs to follow Gaussian distribution [34], [42], [43]. In this way, the LSR distribution can be perfectly estimated.

Attack Overhead: The major overhead imposed by Scale-MIA is in the adversarial model generation phase, or more specifically, the training process of the autoencoder. Except this, the other steps in phase one and the second input reconstruction phase only involve analytical operations and are efficient to perform. Scale-MIA allows this autoencoder training process to be conducted fully offline and the attacker can leverage any computation resource to fulfill this. The attacker can also resort to finding publicly available pre-trained autoencoders. Scale-MIA is a single-round attack and once the attack preparation phase is finished, the attacker can iteratively launch attacks for multiple rounds, i.e. "train once, and attack multiple rounds".

Binary Reconstruction: Scale-MIA exhibits a binary reconstruction property, meaning that for one specific sample, the reconstruction performance is either good enough to obtain a highly similar reconstructed sample or completely fails to obtain any meaningful content. This is because the major reason for reconstruction failure is the collisions within the reconstruction bins of the linear leakage primitive. The successful samples fall in one bin alone and can be properly reconstructed. But the failed ones fall in the same bin together and their reconstructed images are also mixed and blurred with each other.

Targeted Attack: Scale-MIA allows the attacker to launch targeted reconstruction with biased auxiliary dataset D_{Adv} containing limited classes of samples. For example, an auxiliary dataset full of "dog" samples can help the attacker reconstruct samples with the "dog" label among all input samples. In our experiment, we find that an auxiliary dataset with a few hundred samples in one class can reconstruct new images in the same class with high accuracy. This is particularly useful, as in many cases, the attacker may have a biased auxiliary dataset and is only curious about certain classes of samples.

VI. Evaluation

A. Experiment Settings

We implemented Scale-MIA on the PyTorch platform. We run all the experiments on a server equipped with an Intel Core i7-8700K CPU 3.70GHz×12, two GeForce RTX 2080 Ti GPUs, and Ubuntu 18.04.3 LTS.

We considered three important evaluation metrics including the reconstruction batch size, reconstruction rate, and the peak signal-to-noise ratio (PSNR) score. The batch size refers to

TABLE IV: The reconstruction performance of Scale-MIA over different batch sizes on FedSGD and FedAVG systems.

Systems	Datasets	64		128		256		512		1024	
		Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
FedSGD (iter=1)	CIFAR-10	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407	0.4841	25.2302
	FMNIST	0.9402	39.4908	0.9057	38.6366	0.7882	34.2213	0.6613	33.9857	0.4173	28.2084
	HMNIST	0.9619	28.3522	0.9150	26.9998	0.8134	24.6165	0.6719	23.2551	0.4271	22.4616
	TinyImageNet	0.8906	25.0072	0.8804	23.2042	0.8136	22.9310	0.6826	22.6568	0.5252	22.3976
	ImageNette	0.8875	22.8267	0.8132	22.7786	0.7136	22.5842	0.5852	22.6251	0.4155	22.4451
	CelebA	0.9234	23.3090	0.8656	23.1878	0.7625	23.1235	0.5871	22.6476	0.3721	22.3869
FedAVG (iter=3)	CIFAR-10	0.9231	28.3490	0.8770	28.0728	0.8006	27.4762	0.6129	25.2569	0.4709	24.7438
	FMNIST	0.9687	39.2831	0.9063	38.6342	0.8068	36.7750	0.6641	32.1269	0.4146	28.3235
	HMNIST	0.9434	28.5524	0.9111	26.9872	0.7246	24.7298	0.6426	22.9255	0.5073	22.0577
	TinyImageNet	0.9070	23.2623	0.8203	23.0105	0.7488	22.8969	0.6804	22.6052	0.5210	22.2692
	ImageNette	0.8678	22.8635	0.8016	22.7005	0.7141	22.6861	0.5640	22.6489	0.3541	22.1800
	CelebA	0.9188	23.1996	0.8453	23.1373	0.7796	23.2074	0.5492	22.6717	0.3529	22.3813
FedAVG (iter=5)	CIFAR-10	0.9121	28.0774	0.8412	27.7808	0.8010	27.9652	0.6401	26.1579	0.4507	24.7552
	FMNIST	0.9046	38.7166	0.8984	37.8936	0.7421	34.0984	0.6308	34.2613	0.4355	28.8201
	HMNIST	0.9551	28.2060	0.9089	27.0392	0.7651	24.5628	0.5439	22.1618	0.4628	22.1073
	TinyImageNet	0.8917	23.2300	0.8125	23.1355	0.6804	22.6053	0.5527	22.4353	0.5261	22.4493
	ImageNette	0.8615	22.7716	0.7953	22.6870	0.7121	22.6863	0.5883	22.6171	0.3506	22.3832
	CelebA	0.9125	23.2661	0.8559	23.1304	0.7547	23.1276	0.5931	23.0004	0.3524	22.4913

TABLE V: The reconstruction performance of Scale-MIA over different models and batch sizes.

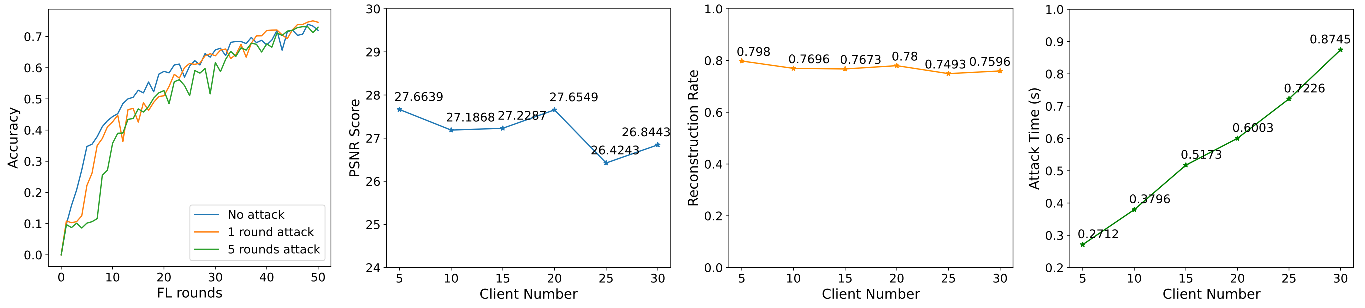
Models	64		128		256		512		1024	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
Alexnet (512)	0.9146	29.0523	0.8383	28.8489	0.7308	28.5446	0.4974	27.2409	–	–
Resnet (512)	0.9162	27.9498	0.8408	27.6572	0.7136	26.7230	0.4869	24.1208	–	–
ViT (512)	0.7656	21.3008	0.6641	20.9629	0.5391	21.0911	0.3809	20.8525	–	–
CNN (1024)	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407	0.4841	25.2302
VGG (1024)	0.9572	28.2689	0.9191	28.2333	0.8463	27.7959	0.7301	26.9002	0.5032	25.0889

the global batch size, i.e. the cardinality of the union of all the local datasets $|\cup_{i=1}^{n} D_i|$. This can be interpreted as the multiplication of local batch sizes and the number of clients n. For example, a global batch with 512 samples can be uploaded by 512 mobile clients with each client having one sample, by 16 clients with each having 32 samples, or just by one client with 512 samples. By default, we consider the system to contain 8 clients in one FL training round. The reconstruction rate is the ratio between the successfully reconstructed samples and the total samples. The definition of the successfulness of reconstructing a sample is by calculating the PSNR score between the original input sample and the reconstructed one, and checking whether the score exceeds a certain threshold th. In our work, we take $th = 18$ because this threshold is enough for the attacker to distinguish the meaningful contents from the reconstructed figures clearly. The PSNR score is a widely adopted metric to quantify reconstruction quality for images and video subject to lossy compression. It can be expressed as $PSNR = 20\log_{10}(\frac{\max_I}{\sqrt{MSE}})$, where $\max_I$ refers to the maximum image pixel value and MSE refers to the mean square error. In this work, we use it to measure the performance of Scale-MIA, following the convention of the existing papers [5], [18], [17], [19].

We implemented Scale-MIA on the Fashion MNIST (FMNIST) [20], Colorectal Histology MNIST (HMNIST) [21], CIFAR-10 [22], TinyImageNet [23], CelebA [24], and ImageNette [23] datasets. Their detailed introduction can be found in the Appendix. For each dataset, we randomly selected a subset of the training set (containing $\{1\%, 3\%, 10\%$, and $100\%\}$ of total samples) as the auxiliary dataset and aimed to reconstruct the whole evaluation set, making sure there is **no intersection** between them. We evaluated Scale-MIA's performance on common machine learning model architectures including the convolutional neural network (CNN), AlexNet [44], VGGNet [45], ResNet [46], and ViT [47]. For each result, we repeated our experiment 5 times to eliminate uncertainty and noise. In Figure 4, we demonstrate several reconstructed examples over the six datasets. We plot the original images in the first row and the reconstructed ones in the second row along with the average PSNR scores. More reconstruction figures with larger batch sizes (from Figure 7 to 12) can be found in the Appendix.

B. Benchmark Comparison

We implemented and compared Scale-MIA with state-of-the-art model inversion attacks that have publicly available artifacts, including the DLG [3], iDLG [4], GradInversion [7], Inverting Gradient [5], robbing the fed [18], and Loki [19]. Among them, DLG/iDLG, GradInversion, and Inverting Gradient attacks are well-received optimization-based gradient inversion attacks, which are used as the fundamental building blocks of more



(a) FL training performance with and without Scale-MIA attack.

(b) Reconstruction PSNR scores over different FL client numbers.

(c) Reconstruction rates over different FL client numbers.

(d) Attack time (s) over different FL client numbers.

Fig. 5: Scale-MIA's attack performance over different federated learning settings.

advanced gradient disaggregation attacks including fishing for user [17], and eluding secure aggregation [16] attacks. Robbing the Fed [18] and Loki [19] represent the most recent large-batch reconstruction attacks that aim to reconstruct large input batches but at the cost of modifying model architectures. To make a fair comparison, we re-implemented all these attacks on the CIFAR-10 dataset and with the CNN model architecture. We focused on the reconstruction rate, average PSNR score, and attack time (the time to reconstruct one batch of inputs) metrics to evaluate the effectiveness and efficiency of the attacks on reconstructing sample batches whose sizes ranged from {1, 2, 4, 8, 16, 64, 256}. We summarize the experiment results in Table III.

From the experiment results, we observe that Scale-MIA outperforms all other attacks in terms of reconstruction rate and attack time while keeping decent PSNR scores. More specifically, the optimization-based attacks [3], [4], [5], [7] suffer from super long reconstruction time (hundreds of seconds), poor reconstruction rate for even small batch sizes, and low PSNR scores. We also experienced large uncertainty during our implementations of these methods because they rely on search-based optimization methods and we got completely different results with different initialization settings. Note that although the batch sizes we used in this experiment have reached or exceeded the performance upper bound of these optimization-based attacks, they have not yet reached the performance bottleneck of Scale-MIA. In comparison, Robbing the fed attack [18] achieves good reconstruction rates with large batch sizes, comparable attack efficiency (reconstruct samples in milliseconds), and even better PSNR scores compared to Scale-MIA. This is because Robbing the Fed is a closed-form attack and no optimization process is required. The other closed-form attack Loki also achieves good reconstruction rates and even better PSNR scores. However, its attack efficiency is significantly worse and requires tens of seconds when the batch size becomes large (as shown in Table III). This is because it adopts a complex model crafting and reversion algorithm. Moreover, as we have discussed in Section II, both Robbing the Fed and Loki attacks adopt a much stronger assumption that requires the attacker to modify the pre-defined model architecture, i.e., adding extra modules before the original model architecture, making them easy to detect. In contrast, Scale-MIA abandons this assumption and still obtains comparable or even better attack performance.

C. Large Batch Recovery Performance

We evaluated Scale-MIA's performance with global batch sizes ranging from {64, 128, 256, 512, 1024} to validate Scale-MIA's performance on reconstructing large global batches under both the FedSGD and FedAVG settings over different datasets. We changed the number of local training iterations that the clients conducted ranging from {1,3,5}. We used a convolutional neural network (CNN) as the target model architecture. The experiment results are shown in Table IV.

We find that Scale-MIA achieves high reconstruction rates and PSNR scores and there is no obvious performance pitfall when the global batch size is smaller than 512. We observe that both the reconstruction rates and PSNR scores are monotonically decreasing with respect to larger batch sizes, which is consistent with our theoretical results, as larger reconstruction batches increase the probability of reconstruction collisions and failures. We also find that when clients conduct more local training iterations, the attack performance slightly decreases. This is because, with more local iterations, the accuracy of the approximated aggregated gradients from the aggregated model updates decreases. But in general, under all assumptions, Scale-MIA achieves decent attack performance and is not affected by whether the system employs the FedSGD or FedAVG algorithms.

We also evaluated Scale-MIA's performance concerning different model architectures under the FedSGD setting on the CIFAR-10 dataset. The results are shown in Table V. We include the neuron number of the first linear layer k beside the model architectures and find that the reconstruction rate is largely affected by it. The models with larger k have better reconstruction rates for a fixed batch size except for a few outliers. We observe that Scale-MIA achieves good reconstruction rates when the batch sizes are smaller than half of the neuron number k, in line with our analysis. In general, Scale-MIA achieves decent attack performance on most model architectures, with only achieving a slightly worse attack performance on the ViT, demonstrating that our attack can be applied to different architectures and is model agnostic.

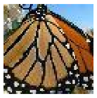
TABLE VI: The reconstruction performance of Scale-MIA over different amounts of data.

Batch	1% (500 samples)		3% (1500 samples)		10% (5000 samples)		100% (50000 samples)	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
16	0.9808	25.1174	0.9824	25.4396	0.9809	27.9021	0.9827	29.4134
32	0.9537	24.9192	0.9569	25.2514	0.9550	27.7217	0.9586	29.1968
64	0.9090	24.8604	0.9136	25.1489	0.9143	27.6032	0.9146	29.0523
128	0.8243	24.6688	0.8316	24.9584	0.8317	27.3481	0.8313	28.8489
256	0.6873	24.3534	0.6923	24.6209	0.7018	26.8835	0.7308	27.2409

TABLE VII: The reconstruction performance of Scale-MIA over different numbers of classes of data.

Batch	1 class		2 classes		3 classes		10 classes	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
16	0.9818	25.1320	0.9856	24.8238	0.9753	25.2728	0.9827	29.4134
32	0.9557	25.0041	0.9701	24.8306	0.9505	25.2147	0.9586	29.1968
64	0.8984	24.8337	0.9128	24.7875	0.8971	25.1607	0.9146	29.0523
128	0.8307	24.5928	0.8568	24.5584	0.8229	25.0155	0.8313	28.8489
256	0.6914	24.1795	0.7214	24.0819	0.6615	24.5275	0.7308	27.2409

TABLE VIII: The reconstruction performance of Scale-MIA over data skew.

Training Data	Testing Data 1	Testing Data 2
		

Batch	Intra-class Skew		Inter-class Skew	
	Rate	PSNR	Rate	PSNR
16	0.9851	23.2206	0.7025	20.2722
32	0.9750	23.1560	0.6625	20.1722
64	0.9194	22.4435	0.6468	19.7398
128	0.8463	22.2957	0.6057	19.6457
256	0.7929	22.1004	0.4678	19.4764

D. Performance over Different FL Settings

In Figure. 5, we demonstrate Scale-MIA's performance under different federated learning settings. We first investigated the overall training accuracy of the FL system with and without our attack. We trained the FL system with 8 clients using the CNN classifier on the CIFAR-10 dataset for 50 rounds. We consider two different attack scenarios, including the attacker only launching a single-round attack in the first training round, and the attacker continuously launching attacks in the first 5 training rounds. From the results, we find that the convergence speed of the FL training process becomes slower in the initial training rounds when the attacks are launched, but the final training accuracy is not affected. In our experiment, the trained model obtained 0.7298 accuracy without any attack, 0.7354 accuracy under a single-round attack, and 0.7301 accuracy under a 5-round attack.

We also investigated Scale-MIA's performance over different numbers of FL clients. We implemented our attack on the CIFAR-10 dataset and fixed the reconstruction batch size to 300. We increased the FL client number from 5 to 30, which was {5, 10, 15, 20, 25, 30}, and evaluated the PSNR scores, reconstruction rates, and elapsed time (in seconds). From the results, we find that the PSNR score and reconstruction rate are slightly affected by the FL client number and remain at a decent high level. This indicates that the reconstruction performance is not affected and the reconstructed images are of good quality. However, we observe that the attack time is almost linear increasing with respect to the FL client number. This is reasonable because more clients involve more communication and computation overhead within the FL system and the complexity of our inversion attack increases accordingly.

E. Data Deficiency and Bias

The quality of the auxiliary dataset D_{Adv} is an important factor that impacts the performance of Scale-MIA. The ideal situation is that D_{Adv} has the same distribution and can represent the training dataset well. However, in practice, there is usually data deficiency and bias and we evaluate the impacts of them in this section.

Data Deficiency: We varied the amount of data available to the attacker for launching Scale-MIA and evaluated its performance on the CIFAR-10 dataset using the AlexNet model. We considered scenarios where the attacker possesses different proportions of the total training data in the auxiliary dataset, ranging from 1%, 3%, 10%, to 100% (equivalent to 500, 1500, 5000, to 50,000 images). The attack performance was tested on the entire validation set, which consists of 10,000 images. Given that the first linear layer of the AlexNet model has 512 neurons, we varied the reconstruction batch size from 16 to 256 to ensure a reasonable reconstruction performance.

We demonstrate the attack performance in table VI. We observe that the number of available samples has little impact on the reconstruction rate, as it remains relatively stable at high values across the full range of data availability. The PSNR score does decrease slightly when the number of available samples decreases, but it remains in a decent range. Specifically, even if the attacker only has 1% samples (500 images in total and 50 images for each class) of the total training dataset, Scale-MIA still achieves very high reconstruction rates and PSNR scores in our experiment. This indicates that Scale-MIA is a practical

TABLE IX: The reconstruction performance of Scale-MIA when the system is protected by DP.

Budget/Batch	64		128		256		512	
	Rate	PSNR	Rate	PSNR	Rate	PSNR	Rate	PSNR
No DP	0.9328	28.6453	0.8988	28.3626	0.8464	27.2986	0.7126	26.7407
$(1, 10^{-5})$	0.9253	25.2141	0.9079	25.1279	0.8445	24.9721	0.6855	24.6703
$(1, 10^{-4})$	0.9140	25.1581	0.9019	25.1392	0.8542	25.0477	0.6934	24.7081
$(5, 10^{-5})$	0.9194	25.1359	0.9034	25.1417	0.8359	25.1688	0.6933	24.5526

attack and the attacker only needs to collect or generate a few hundred samples to obtain a decent attack performance.

Data Bias: We assumed the auxiliary dataset D_{Adv} to have different numbers of data classes from the CIFAR-10 dataset to evaluate Scale-MIA's performance over biased data. We considered the attacker to have $\{1,2,3,10\}$ classes of data samples and evaluated Scale-MIA's performance on these particular classes. For example, if we had the attacker possess the "dog" samples, we would only evaluate Scale-MIA's performance on reconstructing the "dog" samples in the validation set, ignoring the samples from other classes. We focused on the PSNR score and reconstruction rate for batch sizes ranging from 16 to 256, following the same setting as our previous experiments.

Table VII presents the targeted attack's performance. The results show that data bias has a very limited impact on the reconstruction rate, as it remains stable even when the attacker has only a few classes of samples. The decrease in PSNR score is also not significant, and even the worst value remains at a relatively high level. These results demonstrate that Scale-MIA's attack performance is robust against data bias. The attacker can successfully launch a targeted attack on specific classes with minimal performance degradation.

Data Skew: We considered the auxiliary dataset D_{Adv} to contain skewed data from the target images. We considered two data skew cases including intra-class skew and inter-class skew. We conducted our experiment on the TinyImageNet dataset and evaluated the attack reconstruction rates and PSNR scores. We considered the auxiliary dataset to contain 500 "monarch butterfly" images. For the intra-class skew, we assumed the target images were 500 "sulfur butterfly" images. For the inter-class skew, we assumed the targets were 500 "frog" images.

In Table VIII, we demonstrate the attack performance over different data skew settings. We find that the attack performance slightly decreases (but remains decent) under intra-class skew settings, while the attack performance significantly drops under inter-class skew settings. This is because autoencoders can only reconstruct images similar to training samples by design. The results indicate that Scale-MIA can overcome intra-class skew well, but still faces gaps in dealing with inter-class skew.

F. Differential Privacy

Differential privacy (DP) [48], [49], [50] has been widely used to protect the training data's privacy in machine learning systems [51], [52], [53], [54]. It has shown its effectiveness in protecting client-level and data record-level membership privacy for FL systems by preventing the attacker from knowing whether one item (either a client or a record) exists in the system. The fundamental idea of DP is to add artificial noise to the model updates before they are sent to the parameter server. Though DP can protect the FL systems against membership inference attacks by its definition, it is demonstrated to be less effective against the model inversion attacks [55].

In this section, we consider the FL system is protected by both the DP and SA protocols, and we examine Scale-MIA's performance on it to check whether our attack can still break them. We assume the clients adopt the DP-SGD algorithm [51] during the local training process with different (ϵ, δ) privacy budgets. In our experiment, we implement DP with the open-source Python-based DP library named Opacus [56]. We demonstrate the results in Table IX. From the results, we find that the reconstruction rate is slightly affected by the DP mechanism and remains stable at high levels under different privacy budgets. The PSNR scores decrease slightly when DP is employed but not significantly. This shows that Scale-MIA can still reconstruct samples with high accuracy and good scalability performance even when the DP is in place. However, Scale-MIA cannot link the reconstructed samples back to its clients (membership inference), showing that DP can still preserve a certain level of privacy.

VII. Discussion

Privacy by Shuffling: Scale-MIA allows a malicious parameter server to reconstruct the whole input batch accurately from the aggregated model updates, demonstrating a serious privacy vulnerability of the SA protocol and federated learning system. However, under the current design, the attacker cannot infer the belongings of these reconstructed samples and attribute them to certain clients. This property is known as "privacy by shuffling", which prevents the attacker from conducting membership inference and shows that the SA protocol can still preserve a certain level of privacy. To further break this privacy guarantee, Scale-MIA can be used in conjunction with the gradient disaggregation attacks [17], [16] by launching these attacks in the first step to obtain the individual model updates from the victims and then using Scale-MIA as a replacement of the existing gradient inversion mechanisms to boost the reconstruction performance.

Two-Linear-Layer Limitation: Scale-MIA requires the model to have two consecutive linear layers to craft the linear leakage module. However, we acknowledge that not all machine learning models necessarily have this component in their architectures although most machine learning classifiers contain it. For example, some Resnet-based and ViT-based models only have one linear layer in the latent space and the attacker must add extra linear layers to launch Scale-MIA. Meanwhile, we observe that some non-linear modules in the feature extraction layers such as the convolutional layers and vision transformers may also leak private information analytically, which can help the attacker bypass the two-linear-layer limitation. We leave this as our future work to investigate.

Attack Stealthiness: Scale-MIA is a single-round attack that can be executed at any stage of FL training. To enhance attack stealthiness while maintaining effective performance, the attacker may choose to launch the attack during the initial or the first training rounds. In these rounds, the model parameters can be initialized with arbitrary patterns, making it challenging for defenders to distinguish between adversarial and benign parameters. From a performance perspective, the parameter server receives relatively large aggregated gradients during these initial rounds, as the global model has not yet converged, which facilitates more accurate linear leakage calculations and improves the model inversion performance.

Potential Countermeasure: The data synthesis method could be a potential countermeasure against our novel attack. The fundamental idea is to have each client generate a *mask set* that hides the original sensitive data samples. These mask sets ensure that the mask samples within them rather than the original local samples owned by individual clients are reconstructed during the reconstruction attack. At the same time, the mask sets shall not affect the federated learning training performance. We consider the guided diffusion model a promising tool for generating such mask sets. But we leave the detailed technical design as the future work.

VIII. Conclusion

In this paper, we propose Scale-MIA, a powerful MIA that breaks the strong SA protocol by reconstructing the whole global batch possessed by the clients efficiently from the already masked and aggregated model updates. Scale-MIA launches the inversion attack from a new perspective by delving into the detailed architecture of the global model and decomposing the complex model inversion problem into two steps: an LSR reconstruction step and an input generation step, based on the observation that the latent space is the critical layer to breach the privacy. Scale-MIA uses a closed-form "linear leakage" primitive to conduct the first step and a fine-tuned generative decoder for the second, making it highly efficient and suitable for large-scale reconstruction. Scale-MIA is also a very stealthy attack as it does not modify the model architecture and can be conducted in any FL training round. With these distinct features, Scale-MIA represents a potent and inconspicuous approach to breach privacy in FL systems, prompting the need for more robust defense mechanisms against such advanced attacks.

Acknowledgments

This work was supported in part by the Office of Naval Research under grants N00014-24-1-2730 and N00014-19-1-2621, by the US National Science Foundation under grants 2247560, 2247561, 2154929, 1916902, 2312447, 2332675, 2331936, and 2235232, and by the Virginia Commonwealth Cyber Initiative (CCI).

References

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

[2] D. Enthoven and Z. Al-Ars, "An overview of federated deep learning privacy attacks and defensive strategies," *Federated Learning Systems: Towards Next-Generation AI*, pp. 173–196, 2021.

[3] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

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

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

[6] J. Zhu and M. Blaschko, "R-gap: Recursive gradient attack on privacy," *arXiv preprint arXiv:2010.07733*, 2020.

[7] 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.

[8] J. Lu, X. S. Zhang, T. Zhao, X. He, and J. Cheng, "April: Finding the achilles' heel on privacy for vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 051–10 060.

[9] A. Hatamizadeh, H. Yin, H. R. Roth, W. Li, J. Kautz, D. Xu, and P. Molchanov, "Gradvit: Gradient inversion of vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 021–10 030.

[10] 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.

[11] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7232–7241, 2021.

[12] J. H. Bell, K. A. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, "Secure single-server aggregation with (poly) logarithmic overhead," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1253–1269.

[13] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, "V eri fl: Communication-efficient and fast verifiable aggregation for federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1736–1751, 2020.

[14] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, "Fastsecagg: Scalable secure aggregation for privacy-preserving federated learning," *arXiv preprint arXiv:2009.11248*, 2020.

[15] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[16] 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.

[17] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," *arXiv preprint arXiv:2202.00580*, 2022.

[18] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," *arXiv preprint arXiv:2110.13057*, 2021.

[19] 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.

[20] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[21] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti *et al.*, "Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)," *arXiv preprint arXiv:1902.03368*, 2019.

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

[23] 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.

[24] Z. Liu, P. Luo, X. Wang, and X. Tang, "Large-scale celebfaces attributes (celeba) dataset," *Retrieved August*, vol. 15, no. 2018, p. 11, 2018.

[25] D. I. Dimitrov, M. Balunovic, N. Konstantinov, and M. Vechev, "Data leakage in federated averaging," *Transactions on Machine Learning Research*, 2022.

[26] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-newton method for large-scale optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[27] 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.

[28] B. Choi, J.-y. Sohn, D.-J. Han, and J. Moon, "Communication-computation efficient secure aggregation for federated learning," *arXiv preprint arXiv:2012.05433*, 2020.

[29] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Attestable robustness for secure federated learning," *arXiv e-prints*, pp. arXiv–2107, 2021.

[30] M. Rathee, C. Shen, S. Wagh, and R. A. Popa, "Elsa: Secure aggregation for federated learning with malicious actors," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1961–1979.

[31] J. Bell, A. Gascón, T. Lepoint, B. Li, S. Meiklejohn, M. Raykova, and C. Yun, "{ACORN}: input validation for secure aggregation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4805–4822.

[32] Y. Ma, J. Woods, S. Angel, A. Polychroniadou, and T. Rabin, "Flamingo: Multi-round single-server secure aggregation with applications to private federated learning," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 477–496.

[33] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[34] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in neural information processing systems*, vol. 29, 2016.

[35] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[36] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[37] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.

[38] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *arXiv preprint arXiv:1806.03384*, 2018.

[39] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[40] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[41] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[42] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli, "The autoencoding variational autoencoder," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 077–15 087, 2020.

[43] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

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

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

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[48] C. Dwork, "Differential privacy," in *International colloquium on automata, languages, and programming*. Springer, 2006, pp. 1–12.

[49] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[50] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[51] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[52] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar, "A critical review on the use (and misuse) of differential privacy in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–16, 2022.

[53] B. Jayaraman and D. Evans, "Evaluating differentially private machine learning in practice," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1895–1912.

[54] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2615–2632.

[55] S. H. Na, H. G. Hong, J. Kim, and S. Shin, "Closing the loophole: Rethinking reconstruction attacks in federated learning from a privacy standpoint," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 332–345.

[56] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao *et al.*, "Opacus: User-friendly differential privacy library in pytorch," *arXiv preprint arXiv:2109.12298*, 2021.

Appendix A
Proof of Linear Leakage Properties

We provide mathematical proofs for the following two properties of the linear leakage primitive, which help to prove the Eq. 7 in Section IV.

Property 1: For l in $\{1, 2, \cdots, k\}$, considering x_p is the p^{th} smallest sample in terms of feature $h(x)$ and falls in the l^{th} bin $[h_l, h_{l+1}]$ alone, $\nabla_{w_{1(l+1)}} L$ satisfies $\nabla_{w_{1(l+1)}} L = \frac{\partial L}{\partial y_{l+1}} \frac{\partial y_{(l+1)}}{\partial w_{1(l+1)}} = \sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v$.

Proof: According to the chain rule, we can calculate the gradient as $\nabla_{w_{1(l+1)}} L = \sum_{v=1}^{k} \frac{\partial L}{\partial y_{l+1}} x_v$. We decompose this equation into two parts as $\sum_{v=1}^{p} \frac{\partial L}{\partial y_{l+1}} x_v + \sum_{v=p+1}^{k} \frac{\partial L}{\partial y_{l+1}} x_v$. For the second part, we have $y_{(l+1)} < 0$ because all these x_v are smaller than x_p and can not activate bin l. Then according to the property of the ReLU function (zero gradients for negative values), $\frac{\partial L}{\partial y_{l+1}} = 0$ for these values. This indicates that the second part is always zero and the property holds.

Property 2: By letting the row vectors of w_2 identical, we have $\frac{\partial L}{\partial y_{l+1}} = \frac{\partial L}{\partial y_l}$.

TABLE X: Attack performance on the FashionMNIST dataset with and without VAE regulation.

Batch	Without VAE		VAE Regulated	
	Rate	PSNR	Rate	PSNR
64	0.9402	39.4908	0.9625	28.0435
128	0.9057	38.6366	0.9107	27.8447
256	0.7882	34.2213	0.8775	27.3211
512	0.6613	33.9857	0.8307	26.4764

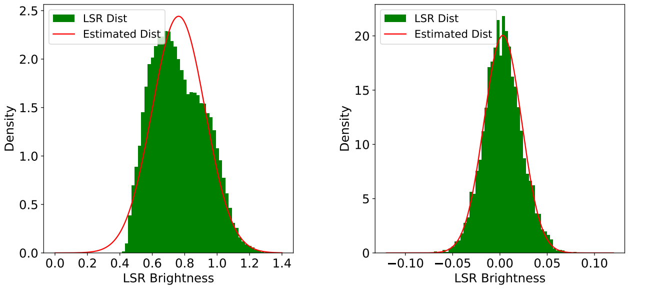
Proof: According to the chain rule, we have $\frac{\partial L}{\partial y_{l+1}} = \sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_{l+1}}$. Because all the row vectors of w_2 are identical, $\frac{\partial z_i}{\partial y_{l+1}} = \frac{\partial z_i}{\partial y_l}$ for all $i \in \{1, 2, \cdots, o\}$. Then we take it back and have $\sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_{l+1}} = \sum_{i=1}^{o} \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial y_l} = \frac{\partial L}{\partial y_l}$.

Appendix B
Experiment Datasets

The FMNIST dataset consists of a training set of 60,000 samples and a test set of 10,000 samples. Each sample is a 28 × 28 grayscale image, associated with a label from 10 classes, including T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. HMNIST is a medical dataset that contains 5000 images for 8 types of skin cancers. Each sample is a 28 × 28 grayscale image, associated with a label from 8 classes of cancers. The CIFAR-10 dataset consists of 60,000 32 × 32 color images in 10 classes, with 50,000 training images and 10,000 test images. Each image is from one of the ten classes, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The TinyImageNet dataset contains 120000 images of 200 classes (600 for each class) sized to 64×64 colored images. Each class has 500 training images, 50 validation images, and 50 test images. The CelebFaces Attributes (CelebA) dataset contains 202,599 face images from 10,177 celebrities of size 178×218, each annotated with 40 binary labels indicating facial attributes like hair color, gender, and age. The training set consists of 162770 images, the test set consists of 19867 images and the validation set consists of 19962 images. The ImageNette dataset is a subset of the super large ImageNet dataset that contains 10 out of 1000 classes in the original ImageNet dataset, including Tench (a type of fish), English springer (dog breed), Cassette player, Chain saw, Church, French horn, Garbage truck, Gas pump, Golf ball, and Parachute. The dataset contains 13000 images of pixel size 320×320.

Appendix C
Regulating LSR Distribution

In this section, we evaluate the Scale-MIA's attack performance when the LSR distribution is regulated to certain distributions. We clarify that the attacker controls this regulation process during the attack preparation phase. More specifically, the attacker can regulate the surrogate autoencoder's local training process to ensure that the LSR distribution of the auxiliary dataset follows pre-defined distributions when the surrogate model is converged. We assume the attacker uses the widely used variational autoencoder (VAE) as the regulation method. We evaluated the regulation performance on the FashionMNIST dataset as without regulation the LSR distribution of the FashionMNIST dataset is biased and not similar to the standard Gaussian distribution. In Figure. 6, we demonstrate the LSR distribution of the dataset before and after the VAE regulation. From the figure, we can clearly observe that the LSR distribution of the FashionMNIST dataset becomes an almost perfect Gaussian distribution after the VAE is used. This makes the statistical estimation much easier. We further demonstrate attack performance in Table. X. We find that using VAE will make the PSNR score lower but remain in a decent range. The main benefit of using VAE is that it makes the LSR distribution shapes better and this contributed to better reconstruction rates, which is validated by the reconstruction rate performance in Table. X.



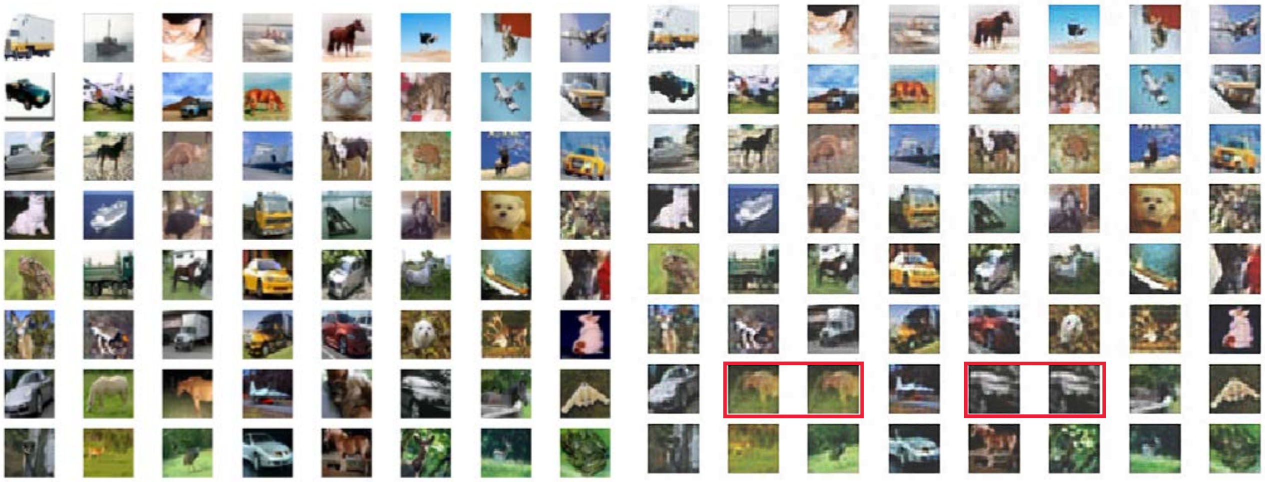
(a) LSR distribution of FMNIST dataset without regulation.

(b) LSR distribution of FMNIST dataset with VAE regulation method.

Fig. 6: The LSR distribution of the FMNIST dataset before and after using the VAE regulation method.

Appendix D
Batched Reconstruction Examples

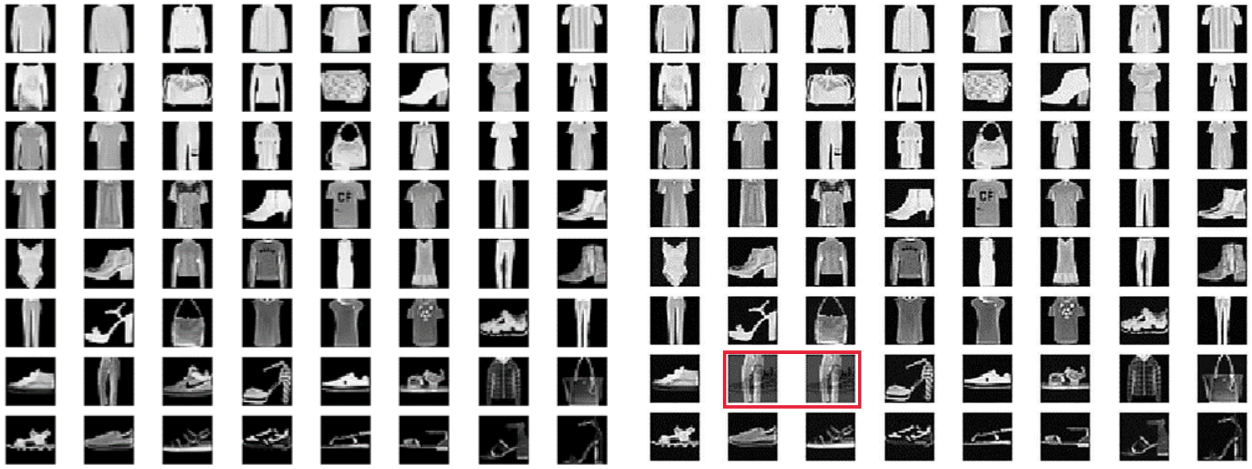
We plot six randomly selected reconstructed batches with batch size 64 from the CIFAR-10, FMNIST, HMNIST, TinyImageNet, ImageNette, and CelebA datasets in Figure 7, Figure 8, Figure 9, Figure 10, Figure 11, and Figure 12 respectively. We can observe excellent reconstruction rates and good PSNR scores on all 6 batches. For each dataset, 60, 62, 55, 57, 62, and 61 out of 64 images are successfully reconstructed respectively. The successfully reconstructed images are very clear and visually identical to the original ones.We also mark the reconstruction failure images in red rectangles. We observe that the failure images usually collide with their neighborhoods and are mixed with each other during the reconstruction process. For these collided pairs, usually one sample instead of two is repeatedly reconstructed. There are also a few reconstruction failure samples that completely collapse and no useful information can be extracted from them.



(a) A batch of 64 images from the CIFAR-10 dataset.

(b) The reconstructed images for the CIFAR-10 input batch.

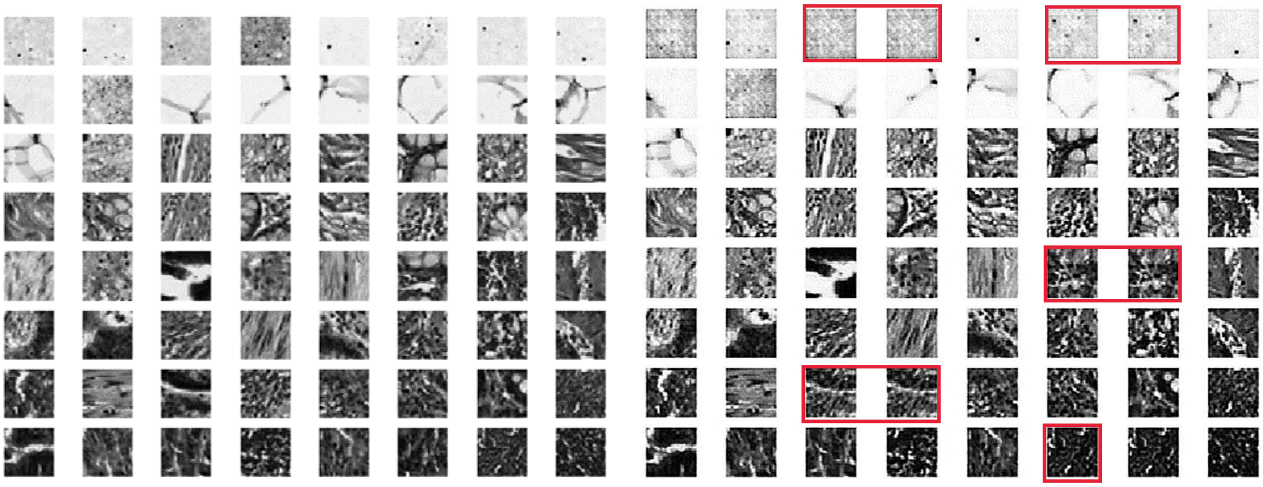
Fig. 7: The comparison between the original images and the reconstructed images with batch size 64 on CIFAR-10.



(a) A batch of 64 images from the FMNIST dataset.

(b) The reconstructed images for the FMNIST input batch.

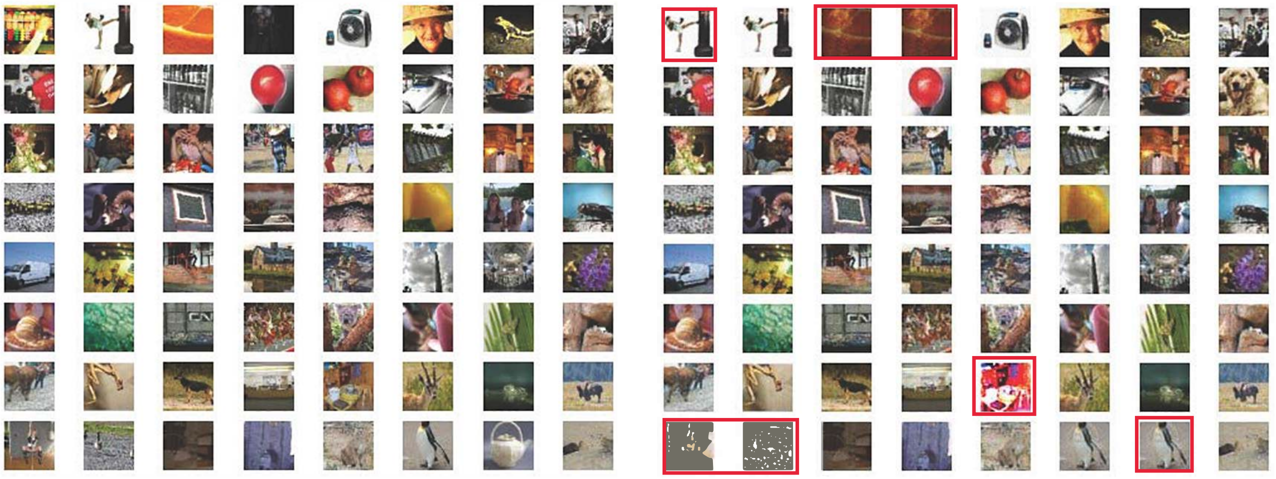
Fig. 8: The comparison between the original images and the reconstructed images with batch size 64 on FMNIST.



(a) A batch of 64 images from the HMNIST dataset.

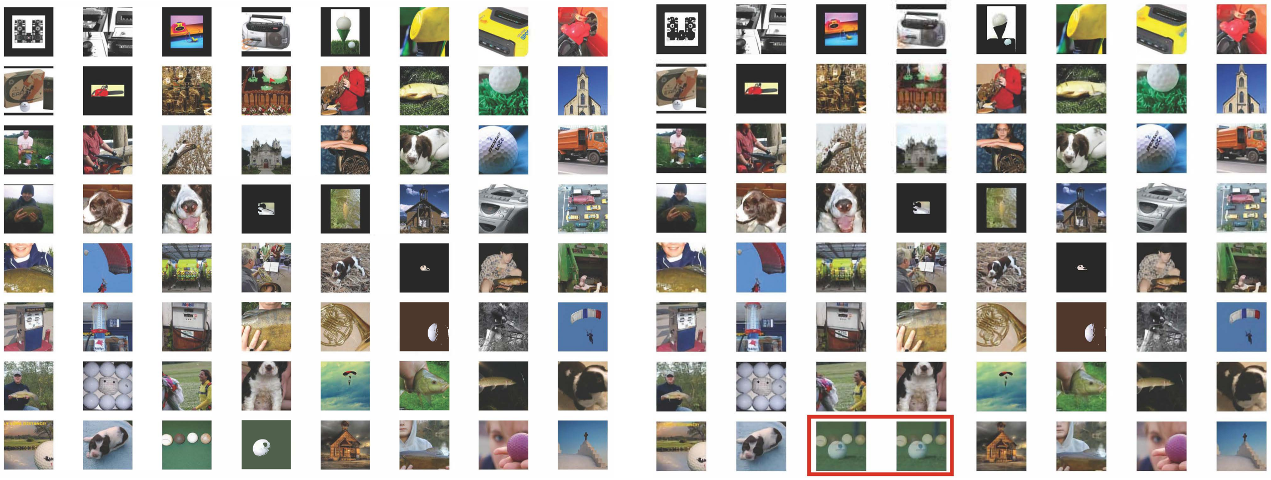
(b) The reconstructed images for the HMNIST input batch.

Fig. 9: The comparison between the original images and the reconstructed images with batch size 64 on HMNIST.



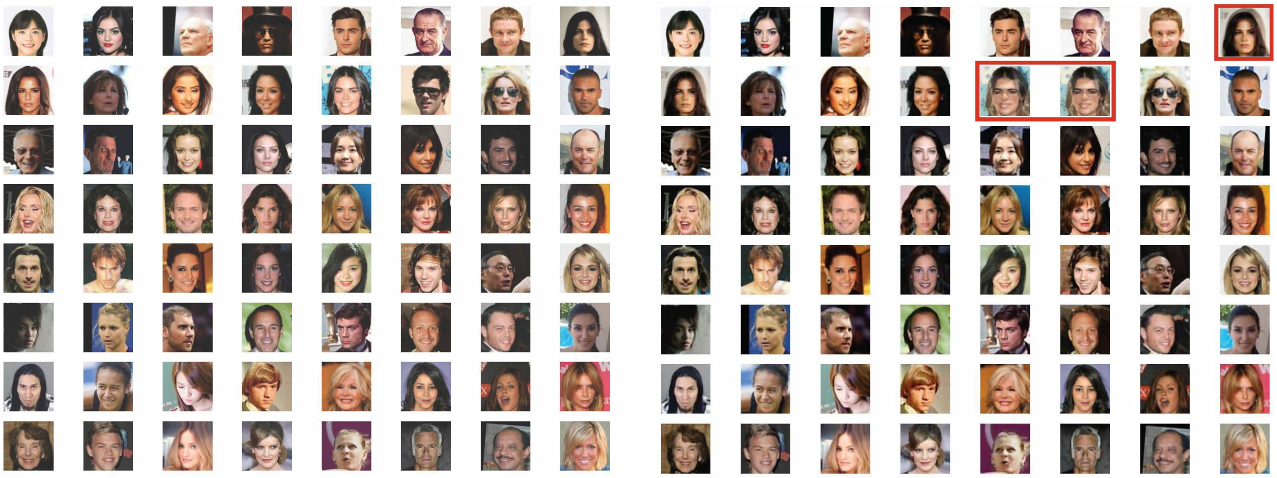
(a) A batch of 64 images from the TinyImageNet dataset. (b) The reconstructed images for the TinyImageNet input batch.

Fig. 10: The comparison between the original images and the reconstructed images with batch size 64 on TinyImageNet.



(a) A batch of 64 images from the ImageNette dataset. (b) The reconstructed images for the ImageNette input batch.

Fig. 11: The comparison between the original images and the reconstructed images with batch size 64 on ImageNette.



(a) A batch of 64 images from the CelebA dataset. (b) The reconstructed images for the CelebA input batch.

Fig. 12: The comparison between the original images and the reconstructed images with batch size 64 on CelebA.

Appendix E
Artifact Appendix

In this work[1], we implement a novel model inversion attack against the federated learning system protected by secure aggregation protocols. Our attack, named Scale-MIA, can reverse the aggregated model updates back to local training samples owned by the clients. This breaks the fundamental privacy-preserving property of the federated learning systems. In this artifact, we provide a complete description of the implementation details of our attack, followed by detailed instructions on how to run it.

A. Description & Requirements

1) How to access: The artifact is publicly available at https://github.com/unknown123489/Scale-MIA and on Zenodo at https://doi.org/10.5281/zenodo.14249064. We provide all implementation code and experiment results for our attack. However, due to space limitations, we omit the large experiment datasets (on the GB scale) in the artifact. Instead, we incorporate downloading code in the files and they will be automatically downloaded when the attack is conducted. Users can also resort to manually downloading these public-available datasets from their official websites.

2) Hardware dependencies: Our artifact can be run on a desktop equipped with GPUs and capable of conducting common machine-learning tasks such as image classification. To ensure all artifact experiments run smoothly, we recommend using a desktop equipped with a GTX 3080 Ti GPU or better.

3) Software dependencies: Our artifact requires a Linux operating system, preferred to be Ubuntu 18.03.3 LTS or later versions. Our artifact is programmed by Python (3.8.10) and requires commonly used machine learning and data processing packages including torch 2.0.1, torchvision 0.15.2, matplotlib 3.7.4, numpy 1.24.4, datasets 3.0.1, scipy 1.14.1, and opacus 1.3.0.

B. Artifact Installation & Configuration

We require our repository to be downloaded to a local folder via *git clone*. For the software dependencies, we recommend users download them with the *pip* command, i.e. *pip install package name*. Alternatively, they can be manually downloaded from their official channels.

C. Experiment Workflow

The artifact contains 5 independent folders organized by the experiment datasets and target systems. For example, the *fedavg-tinyimagenet* folder contains all experiment code and results related to the TinyImageNet dataset on the FedAVG system. Within each folder, the attack consists of three Python files including the *fedavg-adv-train.py*, *fedavg-para-gen.py*, and *fedavg-recover-attack.py*. The first two files correspond to the attack preparation phase (steps ① to ⑤), including the training of the surrogate autoencoder and the parameter crafting of the adversarial global model. The third file implements the actual attack phase (steps ⑥ to ⑧), which reverses the aggregated model updates back to local training samples. Users can execute the three files in sequence to achieve the complete attack process. However, due to the large computation and memory overhead introduced in the attack preparation phase, we provide well-trained intermediate models and results for the first two files in each folder and users can bypass this overhead by directly executing the third file (i.e. the attack phase). This provides the same results as executing three files in order, while also significantly reducing the execution overheads.

Regarding the experiment results, our code automatically prints the reconstruction rate, MSE score, PSNR score, and attack elapsed time when they are executed. Our artifact also provides figures for the training loss trend, original images, reconstructed images, and reconstruction statistics in the *figs* folder.

Particularly, within the *fedavg-cifar* folder, we provide the implementation and results of our attack under data deficiency and bias settings, using different machine learning model architectures, as well as under differential privacy protections. We provide flexible arguments for users to specify the detailed attack parameters for these experiments.

D. Major Claims

- (C1): Our attack can accurately reconstruct large batches of local training samples owned by the client from the aggregated model updates under the FedAVG system. This is proven by reconstructed images yielded in E1. The results show similar values (±0.05 for reconstruction rate and ±3 for PSNR score) as Tab 4.
- (C2): Our attack can be applied to many popular machine learning models. This is proven by experiment (E2) as it yields experiment results similar to those in Tab. 5.
- (C3): Our attack can handle data deficiency and bias settings and maintain decent attack performance. This is proven by experiment (E3), which yields experiment results similar to those in Tab. 6 and Tab. 7.
- (C4): The differential privacy mechanism cannot prevent our attack. This is proven by experiment (E4), which yields experiment results similar to those in Tab. 8.

E. Evaluation

1) Experiment (E1): [Image Reconstruction Results] [1 compute-hour]: This experiment demonstrates Scale-MIA's capability to reconstruct large-batch and high-quality local training samples.

[Preparation] Users need to go into a specific folder and execute the following attack preparation files in sequence:

```
cd foldername
python fedavg-adv-train.py
python fedavg-para-gen.py
```

These steps are in charge of training the surrogate autoencoder and crafting the adversarial global model. However, this step is optional because we have already stored all necessary

[1] This paper was undergoing major revision at the time the artifact evaluation committee conducted their evaluation. The content of the paper has since been updated.

intermediate results within each folder. Users can directly go to the *execution* step to evaluate the attack performance.

[Execution] To evaluate the attack performance under different attack settings, users can execute the following command and specify detailed attack arguments (e.g. batch size, test rounds):

```
python fedavg-recover-attack.py --batch_size
=64 --test_rounds=10
```

We provide the following arguments for users:

`--batch_size`: Reconstruction batch size.

`--test_rounds`: Reconstruction batch numbers.

`--client_num`: Number of FL clients

`--local_epoch`: Each client's local training epochs.

[Results] After executing the attack main file, the reconstruction rate, MSE score, PSNR score, and attack time will be printed in the command line interface. Users can further execute the *visual.py* to produce two figures for original images and reconstructed images. They are shown in the *figs* folder.

```
python visual.py
```

2) Experiment (E2): [Regarding Model Architectures] [1 compute-hour]: This experiment demonstrates that Scale-MIA can be applied to many popular machine learning model architectures.

[Preparation] Users need to go into the *fedavg-cifar* folder and execute the following attack preparation files in sequence while specifying the target model name (e.g. Resnet):

```
cd fedavg-cifar
python multimodel-adv-train.py --model_name=
Resnet
python multimodel-para-gen.py --model_name=
Resnet
```

Again, this step is optional and users can directly go to the following *execution* step. We define the following argument so that users can specify which ML model is evaluated. Candidate models include Vggnet, Alexnet, Resnet, CNN, and Vit.

`--model_name`: Name of the ML model.

[Execution] Users can execute the following command to evaluate Scale-MIA's performance on different model architectures:

```
python multimodel-recover-attack.py --model
_name=Resnet
```

[Results] The experiment results will be automatically printed in the command line interface.

3) Experiment (E3): [Data Deficiency and Bias Results] [1 compute-hour]: This experiment demonstrates Scale-MIA's performance when its auxiliary dataset is under data deficiency and bias settings.

[Preparation] Users need to go into the *fedavg-cifar* folder and execute the following attack preparation files in sequence:

```
cd fedavg-cifar
python targeted-adv-train.py
python targeted-para-gen.py
```

This step is still optional and users can directly go to the following *execution* step.

[Execution] Users can execute the following command to evaluate Scale-MIA's performance under data deficiency settings:

```
python fedavg-recover-attack.py --aux=1
```

The `--aux` argument specifies the percentage of training samples the auxiliary dataset contains. Candidate values are 1,3,10,100. Note that the auxiliary dataset has no intersection with the reconstructed images.

Additionally, users can execute the following command to evaluate Scale-MIA's performance under data bias settings:

```
python targeted-recover-attack.py
```

[Results] The experiment results will be automatically printed in the command line interface.

4) Experiment (E4): [Differential Privacy Results] [1 compute-hour]: This experiment demonstrates that the differential privacy mechanisms cannot defend our attack.

[Preparation] Users need to go into the *dp-cifar* folder and execute the following attack preparation files in sequence:

```
cd fedavg-cifar
python fedavg-adv-train.py
python fedavg-para-gen.py
```

Users can also resort to directly going to the *execution* step as these intermediate execution results have already been stored.

[Execution] We specify the `--epsilon` and `--delta` arguments for users to change the experiment parameters of the DP mechanism. The two parameters represent the privacy budget of the DP mechanism.

```
python dp-recover-attack.py --epsilon=1
--delta=1e-4
```

[Results] As before, the experiment results will be automatically printed in the command line interface.