The Gradient Puppeteer: Adversarial Domination in Gradient Leakage Attacks Through Model Poisoning

Kunlan Xiang, Haomiao Yang, *Senior Member, IEEE*, Meng Hao, *Member, IEEE*, Shaofeng Li, *Member, IEEE*, Haoxin Wang, Zikang Ding, Wenbo Jiang, *Member, IEEE*, and Tianwei Zhang, *Member, IEEE*

***Abstract*—In Federated Learning (FL), clients share gradients with a central server while keeping their data local. However, malicious servers could deliberately manipulate the models to reconstruct clients' data from shared gradients, posing significant privacy risks. Although such Active Gradient Leakage Attacks (AGLAs) have been widely studied, they suffer from two severe limitations: 1) coverage: no existing AGLAs can reconstruct all samples in a batch from the shared gradients; 2) stealthiness: no existing AGLAs can evade principled checks of clients. In this paper, we address these limitations with two core contributions. First, we introduce a new theoretical analysis approach, which uniformly models AGLAs as backdoor poisoning. This analysis approach reveals that the core principle of AGLAs is to bias the gradient space to prioritize the reconstruction of a small subset of samples while sacrificing the majority, which theoretically explains the above limitations of existing AGLAs. Second, we propose Enhanced Gradient Global Vulnerability (`EGGV`), the first AGLA that achieves complete attack coverage while evading client-side detection. In particular, `EGGV` employs a gradient projector and a jointly optimized discriminator to assess gradient vulnerability, steering the gradient space toward the point most prone to data leakage. Extensive experiments show that `EGGV` achieves complete attack coverage and surpasses state-of-the-art (SOTA) with at least a 43% increase in reconstruction quality (PSNR) and a 45% improvement in stealthiness (D-SNR).**

***Index Terms*—Federated learning, gradient leakage attack, model poisoning, malicious attack.**

Received 10 April 2025; revised 15 August 2025; accepted 24 August 2025. Date of publication 8 September 2025; date of current version 30 October 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4501200; in part by the National Natural Science Foundation of China under Grant U24A20259; in part by the Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Qilu University of Technology (Shandong Academy of Sciences) under Grant 2023ZD003; and in part by the Key Laboratory of Data Protection and Intelligent Management, Ministry of Education, Sichuan University under Grant SCUSAKFKT202403Y. The work of Shaofeng Li was supported in part by the National Natural Science Foundation of China under Grant 62502086 and in part by the Start-Up Research Fund of Southeast University under Grant RF1028624178. The associate editor coordinating the review of this article and approving it for publication was Dr. Jason (Minhui) Xue. *(Corresponding author: Haomiao Yang.)*

Kunlan Xiang, Haomiao Yang, Zikang Ding, and Wenbo Jiang are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: klxiang@std.uestc.edu.cn; haomyang@uestc.edu.cn; dzkang0312@163.com; wenbo_jiang@uestc.edu.cn).

Meng Hao is with Singapore Management University, Singapore 188065 (e-mail: menghao303@gmail.com).

Shaofeng Li is with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: shaofengli2013@gmail.com).

Haoxin Wang is with Sichuan University, Chengdu 610000, China (e-mail: whx1122@stu.scu.edu.cn).

Tianwei Zhang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: tianwei.zhang@ntu.edu.sg).

Digital Object Identifier 10.1109/TIFS.2025.3607271

I. Introduction

FEDERATED Learning (FL) [1], [2], [3] has emerged as a promising framework for privacy-preserving distributed learning, allowing multiple clients to jointly train a global model without sharing raw data. In each communication round, the server distributes the model to clients, who compute gradients on their private data and return them to the server for aggregation. However, a growing body of research has revealed that these shared gradients can be exploited by adversaries to reconstruct private clients' data, an attack known as Gradient Leakage Attacks (GLAs) [4]. GLAs severely compromise the core privacy promise of FL and are broadly categorized into two types [5]: Passive Gradient Leakage Attacks (PGLAs) [6], [7], [8], and Active Gradient Leakage Attacks (AGLAs) [5], [9], [10], [11].

In PGLAs, attackers reconstruct client data from the gradients shared in the FL system, without manipulating the model structure, model parameters, and the FL protocol. Zhu et al. [4] first demonstrate the possibility of reconstructing data by optimizing randomly initialized pixel data to produce gradients that match the observed ones. Subsequent methods such as iDLG [12], IG [13], and STG [14] improve the reconstruction results by adding image priors as an optimization objective. However, the effectiveness of these attacks heavily depends on the initialization of model parameters. The model parameters in an unfavorable position produce the gradients lacking data features, rendering these attacks ineffective [15]. For instance, when model parameters are initialized to zero, the gradients will also be zero and thus contain no data feature, thereby preventing any successful reconstruction by existing GLAs. Our experimental results in Table III and Figure 4 demonstrate that, for the first time, even previously considered effective PGLAs fail to reconstruct the data when improper initialization methods are used by the server.

In contrast, in AGLAs, attackers achieve data reconstruction by modifying the model structure and parameters. Some AGLAs [5], [9], [10], [11] achieve direct and accurate reconstruction by inserting a fully connected layer at the beginning

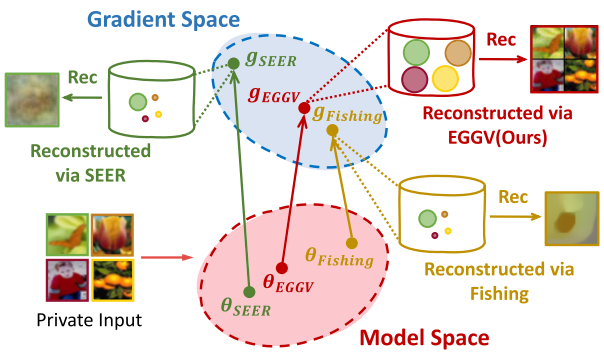


Fig. 1. Illustration of the core principles and reconstruction outcomes of Fishing [16], SEER [17], and EGGV (Ours). Each cylinder denotes the gradient space of a batch, where each inner circle represents the features of each sample contained in the average gradient, with a larger radius indicating more features contained in the averaged gradient for that sample. Existing AGLAs amplify a few samples while suppressing others, leading to partial recovery. In contrast, EGGV (Ours) uniformly enhances the feature of all samples in the gradient, enabling full-batch reconstruction.

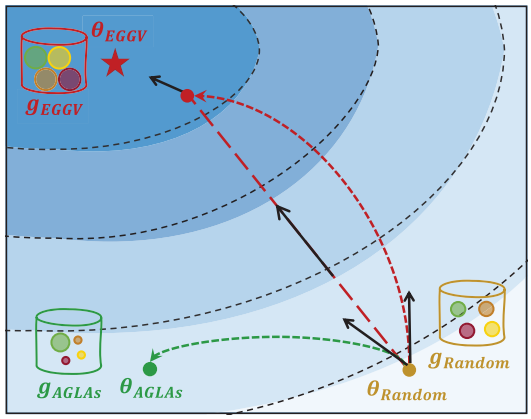


Fig. 2. Fundamental principle comparison between EGGV and gradient-biased AGLAs.

of the model and modifying the parameters of this layer. However, such structural modifications are easily detected. Other AGLAs [16], [17] improve PGLAs by reducing the effective samples used for gradient computation. However, this reduction also limits the improved reconstruction of PGLAs to only a few samples or even just a single sample, as shown in Figure 1. Moreover, recent work [17] reveals that all prior AGLAs are detectable by detection metric. In summary, existing AGLAs exhibit several limitations in attack coverage and stealthiness.

This paper addresses the above critical challenges with two major contributions. First, we propose a new theoretical approach to rethinking and analyzing AGLAs. The core of our approach is the introduction of a parameter λ, which can quantify the relative contribution of each sample within a batch to the activation of neurons in each class. It discloses that the fundamental principle of existing AGLAs is to prioritize the reconstruction of a small subset of samples with specific properties, while sacrificing the majority of other samples. Such properties are analogous to triggers in backdoor attacks. This principle explains the critical limitations of existing AGLAs, underscoring the pressing need for an advanced attack with a fundamentally different principle.

Second, based on the above theoretical analysis, we propose Enhanced Gradient Global Vulnerability (EGGV), a novel AGLA that ensures complete attack coverage and evades client-side detection. Different from existing attacks, EGGV equally enhances the gradient vulnerability of all samples in a batch, as illustrated in Figure 1. Its key insights include: (i) treating gradients as a latent space of data, with the forward and backward propagation of the model as an encoding process; (ii) introducing a discriminator jointly trained with the model to assess the gradient vulnerability of the model. Importantly, EGGV opens up a new research path thoroughly different from the existing AGLAs.

Our contributions are summarized as:

- We introduce a backdoor-theoretic perspective to frame the fundamental principles of AGLAs and identify two critical limitations in their principles: incomplete attack coverage and poor stealthiness.
- We propose a novel GLA EGGV that extends AGLAs to achieve both complete coverage and enhanced stealthiness. We further provide theoretical guarantees for the existence of optimal poisoned parameters for gradient leakage and the stealthiness of the proposed attack.
- Extensive experiments show that the proposed EGGV significantly surpasses SOTA methods, achieving at least a 43% improvement in reconstruction quality (measured by PSNR) and a 45% enhancement in stealthiness (measured by D-SNR) with a complete attack coverage.

Paper Organization: The remainder of this paper is organized as follows. Section II reviews related works on GLAs. Section III provides the necessary background, including the FL framework, the threat model, and motivation for this work. In Section IV, we present a backdoor-theoretic analysis of AGLAs to reveal the fundamental limitations in their principles. Section V introduces the proposed attack, detailing its formulation, optimization approach, and theoretical guarantees. Experimental results are presented in Section VI. Finally, Section VII draw a conclusion.

II. Related Work

In this section, we provide an overview of prior GLAs.

A. Passive Gradient Leakage Attacks (PGLAs)

Most PGLAs are carried out through gradient matching. The DLG attack [4] is the first to optimize dummy inputs and corresponding dummy labels by matching their gradients to the observed gradients. Its optimization objective is:

$$x'^*, y'^* = \underset{x',y'}{\operatorname{argmin}} \left\| \frac{\partial \ell(F(x', \theta), y')}{\partial \theta} - \nabla\theta \right\|^2, \tag{1}$$

DLG works well on small models, such as LeNet-5 [18]. Later, iDLG [12] further improves DLG by inferring data labels from the gradients, but this method is limited to batches with unique labels. Subsequent works, such as LLG [19], extend iDLG to handle larger batch sizes, while other methods [20], [21], such as instance-wise reconstruction [20], successfully recover ground-truth labels in large batches even with duplicate labels.

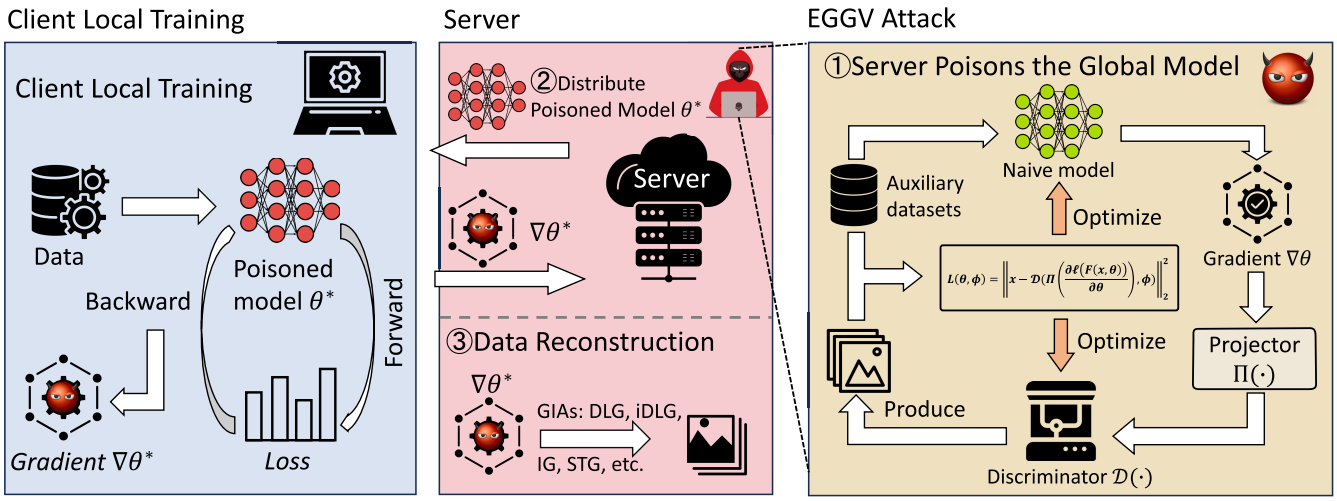


Fig. 3. Overview of the proposed EGGV, consisting of three steps: ① poison the model parameters to make its gradient space vulnerable; ② distribute poisoned model and gather vulnerable gradients; ③ implement existing GLAs on these gradients.

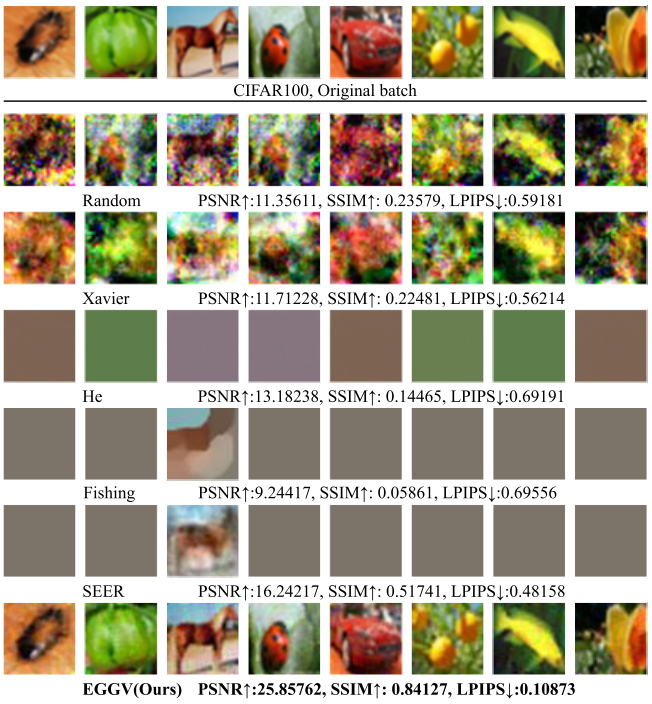


Fig. 4. Visual reconstruction of IG on the model with Random initialization, Xavier initialization, He initialization, Fishing poisoning [16], SEER poisoning [17], and EGGV poisoning.

IG [13] further introduces Total Variation and a Regularization term to the optimization objective. The study [14] leverages the mean and variance from batch normalization layers as priors to enhance GLAs. Instead of optimizing dummy inputs directly, GI [22] optimizes the generator and its input latent to generate dummy images whose gradients match the observed gradients. GGL [23] simplifies this by focusing solely on the latent space of pre-trained BigGAN for gradient matching. More recently, GGDM [24] uses captured gradients to guide a diffusion model for reconstruction. However, despite these advancements, PGLAs fail under improper model parameter initializations that yield gradients containing minimal data features. Our experiments in Table III and Figure 4, for the first time, question the practical effectiveness of these attacks under popular model initializations. This exposes a critical limitation in real-world scenarios.

B. Active Gradient Leakage Attacks (AGLAs)

Based on the manipulation strategies, most AGLAs can be classified into two categories: structure-modified AGLAs and gradient-biased AGLAs.

1) Structure-Modified AGLAs: This type of attack mainly enhances PGLAs by assuming a dishonest server that manipulates the model structure [5], [9], [10], [11]. Some studies [5], [11] insert an FC layer at the beginning of the model, while another work [10] inserts a convolutional layer and two FC layers. These inserted layers are referred to as "trap weights", which are maliciously modified so that the neurons inside are activated only by samples with specific properties, enabling the reconstruction of the sample with the strongest property. However, their explicit modifications to the model structure are inherently detectable, rendering them impractical in real-world scenarios. Therefore, this work focuses on another type of AGLAs [16], [17], [25] which poisons model parameters instead of modifying the model structure.

2) Gradient-biased AGLAs: Gradient-biased AGLAs [16], [17], [25], [26] poison model parameters to skew the gradient space, ensuring that selected samples dominate the batch-averaged gradients while suppressing others. For instance, the attack [25] zeros out most of the convolutional layers, ensuring that only one sample's features reach the classification layer, activating the relevant neurons. The method [16] assigns many 0s and 1s to the last FC layer to make the averaged gradient close to the gradient of a single sample, thereby enhancing the reconstruction of PGLAs on a single sample. The method [26] distributes inconsistent models to clients, forcing non-target users' ReLU layers [27] to output zero gradients, thereby retaining only the target user's gradients, which can then be

TABLE I
MAJOR NOTATIONS USED IN THIS PAPER

Notation	Description
x	Data batch input into the model
x_i	i-th sample in a batch x
y, y_i	Ground-truth label for x or x_i
$\hat{y}$	Model prediction, i.e., $\hat{y} = F(x, \theta)$
$F(\cdot)$	Neural network model
θ	Model parameters
W, W^k	Weight matrix and its k-th column in an FC layer
η	Learning rate
$\mathcal{D}(\cdot)$	Gradient-to-inputs decoder network
$\ell(\cdot, \cdot)$	Loss function (e.g., cross-entropy)
$\nabla_\theta \ell$	Gradient of the loss with respect to θ
$R(\cdot)$	Gradient leakage reconstruction function
x'	Reconstructed inputs from gradients
B	Batch size
C	The number of classification classes
k	Class index
θ^*	Optimized model parameters maximizing gradient leakage
ϕ	Parameters of the decoder $\mathcal{D}$
λ_i^k	Contribution weight of x_i to class-k neuron gradient
$\mathbf{\Lambda}$	Weight matrix formed by λ_i^k over i and k
$\bar{x}^{(k)}$	Weighted average input reconstructed via class-k gradient
b, b^k	Bias vector and its k-th element
$\Pi(\cdot)$	Gradient projection operator
$\tilde{g}$	Projected gradient vector
ρ	Projection ratio of gradient dimensionality
$L(\theta, \phi)$	EGGV training loss for model and decoder
D / D_a	Client dataset / Auxiliary dataset available to attacker

exploited to leak the targeted private data. Recent work [17] observes that all these AGLAs bias the averaged gradient toward the gradients of a small subset of data within a batch while suppressing the gradients of other samples. Exploiting this bias in the biased gradients, research [17] introduces a D-SNR detection metric to check poisoned model parameters, which is calculated as below:

$$D-SNR(\boldsymbol{\theta}) = \max_{W \in \boldsymbol{\theta}_{lw}} \frac{\max_{i \in \{1,\ldots,B\}} \left\| \frac{\partial \ell(F(x_i), y_i)}{\partial W} \right\|}{\sum_{i=1}^{B} \left\| \frac{\partial \ell(F(x_i), y_i)}{\partial W} \right\| - \max_{i \in \{1,\ldots,B\}} \left\| \frac{\partial \ell(F(x_i), y_i)}{\partial W} \right\|}, \tag{2}$$

where θ_{lw} denotes the set of weights of all dense and convolutional layers. D-SNR claims that all prior AGLAs are detectable by principled checks.

III. BACKGROUND

This section provides the necessary background for understanding our work. We first present an overview of the FL paradigm. We then describe the adversarial assumptions and capabilities underlying our threat model. We also discuss the motivation behind our proposed attack. For clarity, the main notations used in this paper are summarized in Table I.

A. Federated Learning

Federated Learning (FL) is a decentralized learning framework that enables multiple clients to collaboratively train a global model without exchanging their raw data, thereby preserving data privacy [1]. In each communication round t, a central server selects a subset of clients $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ and distributes the current global model $F(\theta_t)$ to them. Each client c_i performs local training on its private data D_i by minimizing the empirical loss $\ell(F(\theta_t), D_i)$ and computes the corresponding gradient:

$$g_i^t = \nabla_{\theta_t} \ell(F(\theta_t), D_i). \tag{3}$$

Clients then upload their gradients $\{g_i^t\}_{i=1}^n$ to the server. The server aggregates the gradients (e.g., via weighted averaging) and updates the global model as follows:

$$\theta_{t+1} = \theta_t - \eta \sum_{i=1}^{n} w_i \cdot g_i^t, \quad \text{where } w_i = \frac{|D_i|}{\sum_j |D_j|}, \tag{4}$$

where η denotes the learning rate. This process is repeated iteratively until model convergence. Although FL avoids direct access to raw data, recent studies have shown that the shared gradients can still reveal client data. This vulnerability has led to a line of research on GLAs, which aim to reconstruct private data from gradients. Our work builds upon this threat by exploring more general and stealthy attack mechanisms.

TABLE II
COMPARISON OF RECONSTRUCTION PERFORMANCE AMONG FISHING [16], SEER [17], AND EGGV (OURS) ON CIFAR100 WITH A BATCH SIZE OF 8

	Min PSNR ↑	Pruned Average PSNR ↑	Max PSNR ↑
Fishing [16]	0.00000	0.00000	12.92526
SEER [17]	0.00000	0.00000	15.97548
EGGV (Ours)	**20.37788**	**21.59001**	**22.86605**

B. Threat Model

Our threat model operates within an FL framework where the server is dishonest and curious. It attempts to infer private client data by poisoning the global model parameters before distribution. However, the server is constrained from modifying the model structure, as structural changes (e.g., inserting layers) are easily detected by clients via structure verification, integrity checks, or test queries. Following the setting in many prior works [7], [8], [17], we assume the malicious client can take some publicly available datasets as the auxiliary dataset, which can be easily obtained from open repositories such as Hugging Face [28], Kaggle [29], or OpenML [30]. This is a realistic assumption, as adversaries can readily download such datasets in practical scenarios. Moreover, we experimentally demonstrate that the effectiveness of our attack is not sensitive to the distribution gap between the auxiliary and target datasets, further supporting the rationality of this setting.

C. Motivation

While PGLAs have been extensively studied and shown to be effective in controlled experimental settings, their success is highly sensitive to the model's parameter initialization. Our empirical observations reveal that models initialized using commonly adopted schemes such as Random, Xavier [31],

TABLE III
PERFORMANCE COMPARISON OF THE PROPOSED EGGV AGAINST BASELINE MODEL INITIALIZATIONS RANDOM, XAVIER, AND HE ON CIFAR10, CIFAR100, AND TINYIMAGENET DATASETS

Method	Interation	CIFAR10			CIFAR100			TinyImageNet		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Random+iDLG	200	15.86852	0.595493	0.289589	16.996	0.537246	0.344143	14.04722	0.222929	0.575494
Xavier [31]+iDLG		20.77170	0.78643	0.24686	19.85292	0.73102	0.26831	12.18539	0.23047	0.58592
He [32]+iDLG		-1.15507	-0.00052	0.72385	-1.94603	-0.00166	0.75271	-1.05347	-0.00043	0.80013
EGGV (Ours)+iDLG		**29.70104**	**0.86494**	**0.10815**	**28.44256**	**0.89706**	**0.09068**	**19.94374**	**0.62674**	**0.22103**
Random+IG	1000	19.16213	0.62193	0.30759	19.34636	0.63830	0.31395	15.47002	0.25633	0.52080
Xavier [31]+IG		24.47016	0.86330	0.14445	23.16149	0.80442	0.17232	13.06239	0.21848	0.57367
He [32]+IG		13.30730	0.10187	0.62424	10.97889	0.09065	0.66285	12.70339	0.20885	0.72560
EGGV (Ours)+IG		**31.96512**	**0.91660**	**0.07358**	**31.55152**	**0.92673**	**0.06176**	**28.62325**	**0.91401**	**0.07571**

and He [32] may fail to embed enough data features into the gradients, rendering even SOTA PGLAs incapable of reconstructing the original samples. This strong dependence on the model parameters fundamentally limits PGLAs to a passive role—attackers lack control over the feature distribution in the gradient space, leading to unpredictable and unreliable performance in real-world deployments.

AGLAs have attempted to overcome these limitations by modifying the model structure or injecting targeted biases into the parameter space. However, such approaches often amplify the gradient of only a subset of samples in the batch while suppressing others, resulting in low attack coverage. Moreover, these manipulations typically introduce detectable anomalies in the gradients or model behavior, thereby increasing the risk of being discovered by defensive mechanisms on the client side. This indicates that existing AGLAs are inherently incapable of simultaneously ensuring attack stealth and integrity, due to their fundamental principles.

Inspired by these observations, we propose a novel attack to enhance the overall data leakage capacity of the gradient. We note that if we can actively guide the model to learn a "balanced encoding" gradient space during training, ensuring that all samples have similar feature expression strengths in the gradient (i.e., enhancing the leakage potential of each sample), this would not only guarantee the reconstruction of all samples but also maintain the naturalness of the gradient distribution, thus reduce the risk of detection.

IV. BACKDOOR-THEORETICAL ANALYSIS

In this section, we introduce a new approach to analyze GLAs. It offers a deep insight into the relationship between model parameters and gradient bias, and explains why existing AGLAs cannot recover all samples in a batch and are detectable.

For a simple neural network that is only comprised of fully connected layers $F(x) = xW + b$, where $x \in \mathbb{R}^{B\times m}$ is a batch of data, $W \in \mathbb{R}^{m\times n}$ is the weight parameters. $b \in \mathbb{R}^{1\times n}$ is the bias, with B being the batch size and n being the number of classification categories. When data x is fed into the model, the outputs are represented by $\hat{y} = xW + b$. As seen in prior work [11], the gradients of the weights and biases of the FC layer can be directly used to reconstruct a weighted average of the input data:

$$\bar{x}^{(k)} = \frac{\nabla_{W^k}\ell(F(x,\theta),y)}{\nabla_{b^k}\ell(F(x,\theta),y)} = \sum_{i=1}^{B}\lambda_i^k \cdot x_i, \tag{5}$$

where $k \in [1,n]$ is the class index, $\nabla_{W^k}\ell(F(x,\theta),y)$ (abbreviated as ∇W^k) denotes the gradient of the k^{th} column of the weight matrix W, and $\nabla_{b^k}\ell(F(x,\theta),y)$ (abbreviated as ∇b^k) represents the gradient of the k^{th} element of the bias b. λ is defined as below.

Theorem 1: Let $F(x) = xW + b$ be the classification model with one FC layer, where x is the input data, W is the weight matrix, and b is the bias vector, and the corresponding model outputs are $\hat{y} = F(x)$. Suppose $\ell(\hat{y}, y)$ is the loss function between the model outputs $\hat{y}$ and the ground-truth labels y. For any class index $k \in \{1,2,\ldots,C\}$ and sample index $i \in \{1,2,\ldots,B\}$, the coefficient λ holds that:

$$\lambda_i^k = \frac{\frac{\partial\ell(\hat{y}_i^k,y_i)}{\partial\hat{y}_i^k}}{\sum_{j=1}^{B}\frac{\partial\ell(\hat{y}_j^k,y_j)}{\partial\hat{y}_j^k}}, \text{and} \sum_{i=1}^{B}\lambda_i^k = 1, \tag{6}$$

where $\partial\ell(\hat{y}_i^k, y_i)/\partial\hat{y}_i^k$ denotes the partial derivative of the loss function with respect to the outputs $\hat{y}_i^k$.

Proof: Consider a batch of B samples $\{(x_i, y_i)\}_{i=1}^{B}$, where $x \in \mathbb{R}^{B\times\text{Channel}\times\text{Height}\times\text{Width}}$ represents the input data and y_i are the corresponding ground-truth labels. The model outputs for each sample within a batch are given by: $\hat{y}_i = x_iW + b \in \mathbb{R}^C$, where C denotes the number of classification classes. The total loss over the batch is $\frac{1}{B}\sum_{i=1}^{B}\ell(\hat{y}_i, y_i)$. Next, we derive the gradients of weights and biases for the k^{th} class and express the ratio $\frac{\nabla W^k}{\nabla b^k}$ in terms of λ_i^k and x_i. The gradient of the weight matrix W with respect to the loss for class index k is given by the average of the gradients over all samples:

$$\nabla W^k = \frac{1}{B}\sum_{i=1}^{B}\nabla W_i^k. \tag{7}$$

According to the chain rule, we can further obtain:

$$\begin{aligned}\nabla W^k &= \frac{1}{B}\sum_{i=1}^{B}\nabla W_i^k = \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k,y_i)}{\partial\hat{y}_i^k}\cdot\frac{\partial\hat{y}_i^k}{\partial W_i^k}\\ &= \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k,y_i)}{\partial\hat{y}_i^k}\cdot x_i.\end{aligned} \tag{8}$$

Similarly, the gradient of the bias corresponding to the k^{th} category index can be derived as:

$$\nabla b^k = \frac{1}{B}\sum_{i=1}^{B}\nabla b_i^k = \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}\cdot\frac{\partial \hat{y}_i^k}{\partial b^k} = \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}\cdot 1 = \frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}. \tag{9}$$

Therefore, $\nabla W^k/\nabla b^k$ can be derived as:

$$\begin{aligned}\frac{\nabla W^k}{\nabla b^k} &= \frac{\frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}\cdot x_i}{\frac{1}{B}\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}} = \frac{\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}\cdot x_i}{\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}} \\ &= \frac{\frac{\partial l(\hat{y}_1^k, y_1)}{\partial \hat{y}_1^k}\cdot x_1}{\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}} + \frac{\frac{\partial l(\hat{y}_2^k, y_2)}{\partial \hat{y}_2^k}\cdot x_2}{\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}} + \cdots \\ &\quad + \frac{\frac{\partial l(\hat{y}_B^k, y_B)}{\partial \hat{y}_B^k}\cdot x_B}{\sum_{i=1}^{B}\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}} \\ &= \sum_{i=1}^{B}\frac{\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}}{\sum_{j=1}^{B}\frac{\partial l(\hat{y}_j^k, y_j)}{\partial \hat{y}_j^k}}\cdot x_i.\end{aligned} \tag{10}$$

This expression can be rewritten as:

$$\frac{\nabla W^k}{\nabla b^k} = \sum_{i=1}^{B}\frac{\frac{\partial l(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}}{\sum_{j=1}^{B}\frac{\partial l(\hat{y}_j^k, y_j)}{\partial \hat{y}_j^k}}\cdot x_i = \sum_{i=1}^{B}\lambda_i^k\cdot x_i. \tag{11}$$

Therefore, $\lambda_i^k = \frac{\partial \ell(\hat{y}_i^k, y_i)}{\partial \hat{y}_i^k}/\sum_{j=1}^{B}\frac{\partial \ell(\hat{y}_j^k, y_j)}{\partial \hat{y}_j^k}$, *and* $\sum_{i=1}^{B}\lambda_i^k = 1$.

Taking a binary classification network with an inputs of 4 samples as an example, the weighted average sample obtained by the gradient of the weights and biases of the two categories can be expressed as:

$$\begin{bmatrix}\frac{\nabla W^1}{\nabla b^1}\\ \frac{\nabla W^2}{\nabla b^2}\end{bmatrix} = \begin{bmatrix}\bar{x}^{(1)}\\ \bar{x}^{(2)}\end{bmatrix} = \begin{bmatrix}\lambda_1^1 & \lambda_2^1 & \lambda_3^1 & \lambda_4^1\\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \lambda_4^2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\end{bmatrix} = \mathbf{\Lambda X}. \tag{12}$$

Equation (12) shows weighted average data resolved by gradient of the weights and biases *w.r.t* a given class, which is actually a weighted summation of the features of the input layer. λ is exactly the weight factor to quantify the bias in the neurons' gradient space toward specific samples within a batch.

λ is crucial for the gradient-biased AGLAs, as it controls the weighted feature proportions computed from gradients. It is the first technical measure to quantify the relative contribution of each sample in the batch to the activation of each class of neurons in the model. Interestingly, we find the AGLAs are very analogous to poisoning-based backdoor attacks. In the later, the attacker poisons the model training process to an expected state to manipulate the model outputs and eventually control the distribution of λ. Therefore, we call this approach the backdoor-theoretical perspective.

The core mechanism of manipulating λ inherently results in several challenges: (1) samples without gradient space bias cannot be reconstructed; (2) reconstruction becomes impossible when two samples with the required properties coexist; and (3) the presence of anomalous gradients in the gradient space makes the attack detectable. This explains the fundamental limitations of existing AGLAs.

This is the first theoretical analysis for AGLAs, which systematically reveals the underlying mechanism behind gradient bias via the backdoor-theoretical lens. It not only bridges a critical gap in the current literature but also provides principled insights into why existing AGLAs are inherently incomplete in reconstruction and detectable due to gradient anomalies.

V. Enhanced Gradient Global Vulnerability

The above-mentioned challenges suggest that instead of controlling λ to achieve reconstruction, we should focus on increasing the concentration of input features and enhancing feature representation at the source rather than compressing the features of some samples to amplify others. Following this inspiration, we introduce `EGGV`, a new attack that poisons the model parameters θ to equally enhance the leakage potential of all samples in a batch, thus ensuring a comprehensive attack and evading detection. Figure 2 provides an intuitive comparison between `EGGV` and existing gradient-biased AGLAs. While existing attacks achieve reconstruction by suppressing the features of non-target samples to amplify those of specific ones in the gradient, `EGGV` instead uniformly enhances the encoded features of each sample in the gradient, following an entirely different principle.

A. Attack Overview

The proposed `EGGV` is comprised of three main steps. Algorithm 1 shows the detailed process. Figure 3 provides a visual overview of `EGGV`.

Step I: The malicious server poisons the global model before its distribution (Step ① in Figure 3, PoisonModel in Algorithm 1). In this stage, the server iteratively optimizes the global model and discriminator locally with the objective function $L(\theta, \phi)$ using an auxiliary dataset, enriching the gradients with encoded features.

Step II: The server distributes the poisoned model to clients (Step ② in Figure 3, CollectClientGradients in Algorithm 1). Each client, following the FL protocol, feeds its own training data into the poisoned model to generate gradients that are then uploaded back to the server.

Step III: The server uses these uploaded gradients to perform data reconstruction using any existing PGLAs (Step ③ in Figure 3, ReconstructData in Algorithm 1).

B. Problem Formulation

The objective of the malicious server is to minimize the difference between the reconstructed data and the original data. Formally, we have the following objective:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\|x - x'\|_p, \tag{13}$$

where x represents one data batch from the auxiliary dataset D, x' denotes the reconstructed data obtained by the attacker

Algorithm 1 Poisoning and Reconstruction of EGGV

Input: Auxiliary dataset D_a, global model $F(\theta)$, acceptable error τ

Output: Reconstructed data x'

Main Process:
$F(\theta^*) \leftarrow$ PoisonModel$(D_a, F(\theta), N)$
$\nabla\theta^* \leftarrow$ CollectClientGradients$(F(\theta^*))$
$x' \leftarrow$ ReconstructData$(\nabla\theta^*)$
return x'
function POISONMODEL$(D_a, F(\theta), N)$
  Initialize θ_0 randomly
  $t \leftarrow 0$
  while $L(\theta_t, \phi_t) > \tau$ **do**
    for each $(x_j, y_j) \in D_a$ **do**
      $g_t = \Pi\left(\frac{\partial \ell(F(x_j,\theta_t),y_j)}{\partial \theta_t}\right)$
      $L(\theta_t, \phi_t) \leftarrow \left\| x_j - \mathcal{D}(g_t, \phi_t) \right\|_2^2$
      Update θ_t, ϕ_t using gradient descent:
      $\theta_{t+1} \leftarrow \theta_t - \alpha_1 \nabla_{\theta_t} L(\theta_t, \phi_t)$
      $\phi_{t+1} \leftarrow \phi_t - \alpha_2 \nabla_{\phi_t} L(\theta_t, \phi_t)$
      $t \leftarrow t + 1$
    end for
  end while
  return $F(\theta^*)$ with updated θ
end function
function COLLECTCLIENTGRADIENTS$(F(\theta^*))$
  Client i receives the global model $F(\theta^*)$
  Client i calculates the gradient $\nabla_{\theta^*}\ell(F(x, \theta^*), y)$ using its data (x, y)
  return $\nabla_{\theta^*}\ell(F(x, \theta^*), y)$
end function
function RECONSTRUCTDATA$(\nabla\theta^*)$
  Select any prior PGLA methods $R(\cdot)$
  Reconstruct client data x' through $R(\nabla\theta^*)$
  return reconstructed data x'
end function

using the gradient leakage method $R(\cdot)$, and p denotes the norm order used to measure the reconstruction error. The gradients are computed on the client side during local training. Specifically, the client inputs local training data x into the model, yielding the outputs $\hat{y} = F(x, \theta)$, and then calculates the gradient $\frac{\partial \ell(F(x,\theta),y)}{\partial \theta}$. The optimization objective can therefore be expanded as follows:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x\sim D}\left[\left\| x - R\left(\frac{\partial \ell(F(x,\theta),y)}{\partial \theta}\right)\right\|_p\right] \tag{14}$$

The server aims to optimize the model parameters θ such that, for any given reconstruction target x, the gradient leakage method $R(\cdot)$ can produce a reconstruction that closely approximates the original inputs x.

C. Detailed Solution

As shown in Equation (14), the results of a gradient leakage attack are largely determined by the model parameters. Consequently, there exists an optimal set of model parameters, denoted by θ^*, whose gradients are particularly prone to leaking model inputs. In other words, gradient leakage attacks can reconstruct data more effectively from gradients produced by θ^* than from those of other parameter sets. Our goal is to poison the model parameters θ toward this optimal set, thereby making the resulting model more vulnerable to gradient leakage. Achieving this requires repeatedly evaluating the gradient vulnerability of the target model during poisoning training and utilizing it as the poisoning objective.

However, directly using the full gradient to evaluate its vulnerability to guide model poisoning is computationally expensive due to its high dimensionality and the heterogeneity of dimensionality across layers. To enable efficient and stable evaluation during the poisoning process, we first introduce a projector $\Pi(\cdot)$ that reduces gradient dimensionality while preserving the essential information of the original gradient. The projector samples gradient entries at fixed positions within each layer, flattens them into vectors, and concatenates these vectors across all layers. Formally,

$$\Pi(g) = (g_1[p_1], \ldots, g_L[p_L], \rho)^T, \tag{15}$$

where p_1, …, p_L represent pre-specified sets of gradient positions of the 1^{st} to the L^{th} layer, and ρ represents the ratio of the number of parameters of the projected gradient to that of the original gradient. In subsequent iterative poisoning of the model, gradients are sampled at these fixed positions, which ensures the stability and consistency of model poisoning. A natural question arises: which effective positions should be chosen? The fixed positions are determined prior to poisoning by assigning a unique index to each model parameter and randomly selecting a subset of indices corresponding to a fraction ρ of the total. This random selection enhances diversity and representativeness and allows the projected gradient to preserve the statistical characteristics of the whole gradient.

By applying the projector $\Pi(\cdot)$, we map the high-dimensional gradient space into a lower-dimensional vector space as follows:

$$\tilde{g} = \Pi\left(\frac{\partial \ell(F(x,\theta),y)}{\partial \theta}\right). \tag{16}$$

Next, we design a reliable mechanism to evaluate gradient vulnerability for the model and take it as a poisoning objective to steer model optimization toward producing more leakage-prone gradients. Traditional approaches, such as Equation (14), assess this vulnerability by performing a full end-to-end reconstruction attack and taking the reconstruction error between the reconstructed data and original data as a metric. While accurate, this method has a high computational cost, as end-to-end reconstruction requires iterative optimization of dummy data for each update of θ. Additionally, iterative reconstruction introduces a nested optimization structure, making it challenging to compute the second-order derivative of the loss.

To overcome these limitations, we design the discriminator as a lightweight decoder that directly evaluates the potential for gradient leakage without end-to-end reconstruction. This design is motivated by prior findings that the risk of gradient leakage increases with the amount of input features embedded in the gradient. From an information-encoding perspective, the forward and backward propagation of the model serve as

an encoding process, mapping the input data into gradient representations. The decoder serves as the inverse of the gradient representations into the original data. The closer the outputs decoded from the gradient are to the original inputs, the more features from the input data are embedded in the gradient and the greater its leakage potential, thereby providing a direct and efficient means of assessing gradient leakage risk.

With the discriminator established, its reconstruction error is directly used to guide model poisoning. The attacker poisons the model θ and optimize decoder ϕ by minimizing:

$$L(\theta, \phi) = \left| x - \mathcal{D}\left(\Pi\left(\frac{\partial \ell(F(x,\theta), y)}{\partial \theta}\right), \phi\right)\right|_2^2, \tag{17}$$

where $\mathcal{D}(\cdot)$ denotes the discriminator and ϕ denotes its parameters. As the poisoning objective does not bias toward any specific sample but uniformly enhances the features of each sample embedded in the batch-averaged gradient, our method can reconstruct the entire batch, thereby ensuring complete attack coverage and strong stealthiness.

D. Optimal Model for the Gradient Leakage

We prove a global minimum exists for the proposed loss function, where the corresponding model parameters maximize the vulnerability of gradient space to data leakage.

Assumption 1: The global model $F(\theta)$ is continuous, and the parameter space Θ of θ is a non-empty compact set.

Assumption 2: The loss function $\ell(\cdot, \cdot)$ for the client training is continuously differentiable with respect to the model parameters θ, allowing for gradient computation with respect to θ.

Assumption 3: The decoder $\mathcal{D}$ is a linear function of the form $\mathcal{D}(\tilde{g}) = W \cdot \tilde{g} + b$ and is continuous. The parameter space Φ of ϕ is a non-empty compact set.

Theorem 2: Under the above assumptions, there exists parameters $\theta^* \in \Theta$, $\phi^* \in \Phi$ such that the loss function $L(\theta, \phi)$ defined in Equation (17) attains its global minimum:

$$\theta^*, \phi^* = \arg \min_{\theta \in \Theta, \phi \in \Phi} L(\theta, \phi). \tag{18}$$

At $\theta = \theta^*, \phi = \phi^*$, the gradient $\nabla_\theta \ell(F(x, \theta), y)$ encodes the maximum amount of feature from the input data x, making the gradient space most susceptible to leakage.

Proof:

1) *Continuity of $L(\theta, \phi)$*: From Assumption (2), since $\ell(\cdot, \cdot)$ is continuously differentiable with respect to θ, the gradient $\nabla_\theta \ell(F(x, \theta), y)$ is continuous with respect to θ. The projector Π is a fixed-position sparse sampling linear operator, so the composite function $\Pi(\nabla_\theta l(F(x, \theta), y))$ is also continuous with respect to θ. By Assumption 3, the decoder $\mathcal{D}$ is continuous. Therefore, the composite function $\mathcal{D}(\Pi(\nabla_\theta l(F(x, \theta), y)))$ is continuous with respect to θ and ϕ. The squared Euclidean norm $\|\cdot\|_2^2$ is a continuous. Hence, the loss function $L(\theta, \phi)$ is continuous with respect to θ and ϕ.
2) *Existence of Global Minimum:* From Assumption (1) and Assumption (3), the parameter space Θ and Φ is a non-empty compact set. By the Weierstrass Extreme Value Theorem [33], [34], any continuous function on a compact set attains its maximum and minimum values. Therefore, there exists $\theta^* \in \Theta$ and $\phi^* \in \Phi$ such that $\theta^*, \phi^* = \arg \min_{\theta \in \Theta, \phi \in \Phi} L(\theta, \phi)$.
3) *Maximum Vulnerability of the Gradient Space:* At $\theta = \theta^*$ and $\phi = \phi^*$, the loss function $L(\theta, \phi)$ attains its global minimum, which indicates that the reconstruction error is minimized. This indicates that the encoding and decoding processes of the gradient have reached an optimal state. If a better decoder or gradient construction existed, it would further reduce $L(\theta, \phi)$, contradicting the minimality of $L(\theta^*, \phi^*)$. Therefore, at $\theta = \theta^*$ and $\phi = \phi^*$, the risk of data leakage from the gradient space to the input data x is maximized. This means the gradient $\nabla_\theta \ell(F(x, \theta^*), y)$ contains the most features of x, rendering the gradient space most vulnerable. □

E. Theoretical Guarantee of Stealthiness

In the previous subsection, we established the existence of optimal poisoned parameters (θ^*, ϕ^*) minimizing the poisoning loss in Equation (17). Here we theoretically demonstrate that gradients derived from θ^* exhibit strong stealthiness, undermining detection methods such as D-SNR [17].

Stealthiness can be quantified by the variance of gradient norms within a batch. Let g_i be the gradient norm of the i-th sample using optimal parameters θ^*:

$$g_i = \left\| \frac{\partial \ell(F(x_i; \theta^*), y_i)}{\partial \theta^*} \right\|_2. \tag{19}$$

We define stealthiness as minimizing the variance of g_i:

$$\mathrm{Var}(g_i) = \frac{1}{B} \sum_{i=1}^{B} (g_i - \mu_g)^2, \quad \text{with} \quad \mu_g = \frac{1}{B} \sum_{i=1}^{B} g_i. \tag{20}$$

Theorem 3 (Gradient uniformity and stealthiness): Under Assumptions 1–3, the optimal poisoned parameters (θ^*, ϕ^*) ensure that the gradient variance is bounded by a small constant ϵ, leading to bounded D-SNR indistinguishable from naturally trained gradients:

$$\mathrm{Var}(g_i) \leq \epsilon, \quad D - SNR(\theta^*) \leq \gamma \cdot \epsilon, \tag{21}$$

where $\gamma > 0$ is a constant dependent on the batch size and model structure.

Proof: At the optimal (θ^*, ϕ^*), the reconstruction loss (Equation (17)) attains its minimum, ensuring uniform reconstruction errors across the batch:

$$\|x_i - x_i'\|_2 \leq \delta, \quad \forall i, \tag{22}$$

for a small constant δ. Given the Lipschitz continuity [33] of decoder $\mathcal{D}$ with constant L_D, we have:

$$\|\Pi(\nabla_{\theta^*} \ell(F(x_i, \theta^*), y_i)) - \Pi(\nabla_{\theta^*} \ell(F(x_j, \theta^*), y_j))\|_2 \leq 2L_D^{-1}\delta. \tag{23}$$

Since the projector Π preserves norm differences up to a constant factor M_Π, we derive:

$$|g_i - g_j| \leq 2M_\Pi L_D^{-1}\delta, \quad \forall i, j. \tag{24}$$

Thus, gradient variance satisfies:

$$\mathrm{Var}(g_i) \leq 4(M_\Pi L_D^{-1})^2 \delta^2 := \epsilon. \tag{25}$$

As D-SNR monotonically decreases with gradient variance [17], we conclude:

$$D - SNR(\theta^*) \leq \gamma \cdot \epsilon, \tag{26}$$

establishing that gradients from θ^* remain indistinguishable from naturally trained models, ensuring stealthiness.

VI. Experimental Evaluation

In this section, we present a series of experiments to evaluate the effectiveness of the proposed method. The experiment results show that EGGV significantly outperforms the SOTA AGLAs in reconstruction quality and stealthiness.

A. Setup

We use ResNet18 [35] as the default global model for FL. The CIFAR10, CIFAR100 [36], and TinyImageNet [37] datasets are employed as training data for clients. The three classic evaluation metrics for reconstruction quality, namely PSNR [38], SSIM [39], and LPIPS [40], are used to assess attack quality. We use the detection metric D-SNR [17] to evaluate the stealthiness of model modifications. The default projection ratio is set to 0.4%, with a linear layer as the default discriminator. We compare our method with the closely related SOTA methods, Fishing [16] and SEER [17], with the maximal brightness as the selected property, which is reported to provide the best attack performance for SEER. As our method is the first to poison model parameters for enhanced data leakage across entire batches, we also evaluate its performance against popular naive model initialization methods, including Random, Xavier [31], and He [32]. In our implementation, Xavier initialization uses a uniform distribution to balance variance across layers, while He initialization adapts weights for leaky ReLU activations to ensure smoother gradient flow during training.

B. Main Results

1) Comparison Between EGGV and SOTA AGLAs: Firstly, we compare the performance of the proposed EGGV with two SOTA AGLAs, Fishing [16], and SEER [17], on CIFAR100 with a batch size of 8. Table II reports the minimum PSNR, pruned average PSNR, and maximum PSNR over 100 batches, where a PSNR of 0 indicates no reconstruction. EGGV achieves significantly higher PSNR, consistently reconstructing all samples per batch with minimal variation, while SOTA methods reconstruct only one sample per batch. The visual comparison in Figure 4 demonstrates the superiority of the proposed EGGV method. Specifically, EGGV successfully reconstructs every sample in the batch with a high similarity to the originals. In contrast, SOTA methods (Fishing and SEER) reconstruct only a single image, and the similarity between these reconstructions and the originals is significantly lower than that of EGGV.

Remark 1: Our method applies not only to model initialization but to any round in FL. Unlike prior work [16], [25] producing suspicious parameters (e.g., zeros or ones), our poisoned parameters exhibit natural distributions. Moreover, since each training round aggregates updates from multiple clients, individual clients remain unaware of the aggregated parameter state, enabling our attack to stealthily poison model parameters at any training round.

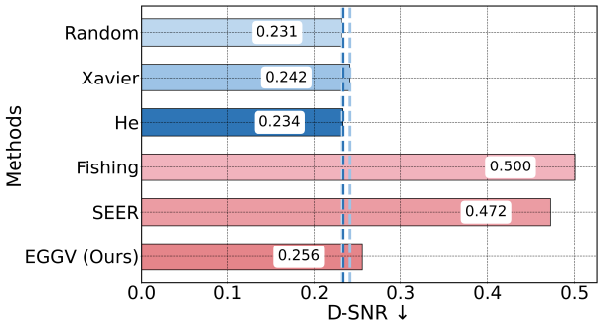


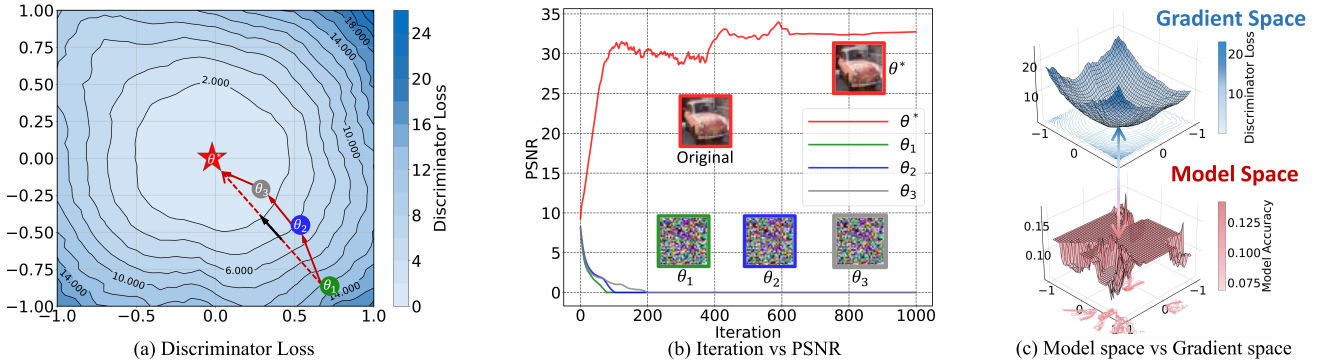
Fig. 5. Bar chart of D-SNR value of gradients generated by models with three naive initialization methods (Random, Xavier, He) and three poisoning methods (Fishing, SEER, EGGV) on 100 same batches. Lower values indicate greater stealthiness. EGGV achieves a highly stealthy, closely approaching D-SNR of standard initialization methods.

2) Comparison Between EGGV and Popular Model Initialization Methods in Enhancing PGLAs: Considering that EGGV is the first AGLA to reconstruct all samples in the batch, we compare its performance with three naive model initialization methods. We implement the iDLG [12] and IG [13] on models that proposed EGGV poisons, naively initialized by Random, Xavier, and He. As shown in Table III, the EGGV significantly outperforms Random, Xavier, and He initialization methods across all three datasets, regardless of whether iDLG or IG is used. Figure 4 provides a visual comparison of reconstruction results, clearly illustrating the superiority of EGGV.

He initialization is widely recognized for its advantages in global model training when used by honest servers, but it often results in attack failures for adversaries, including the server itself. This indicates that relying solely on original model parameters results in poor attack performance. Our research further shows that to improve the effectiveness of attacks, adversaries cannot rely only on standard model parameters. Instead, they should adopt poisoning techniques like EGGV to actively manipulate the gradient space.

The experimental results also highlight the critical importance of the gradient position within the gradient space for the success of GLAs. Unfortunately, previous PGLAs have overlooked this factor. Traditional PGLAs are typically limited by the current state of the model parameters, thus making it difficult to achieve optimal results. Although AGLAs attempt to address this by poisoning model parameters, their effectiveness is limited to a small subset of samples within the batch, as illustrated in Figure 4. Notably, EGGV is the first method to tackle this key challenge for both PGLAs and AGLAs by poisoning model parameters to enhance the gradient vulnerability across the entire batch.

3) Stealthiness Comparisons of EGGV with SOTA AGLAs: We calculate 100 gradients on CIFAR100 with each ResNet18 poisoned by Fishing, SEER, and EGGV and initialized by naive initialization methods Random, Xavier, and He. We then report the D-SNR values for these gradients. As illustrated



(a) Discriminator Loss (b) Iteration vs PSNR (c) Model space vs Gradient space

Fig. 6. Figure (a): Contour map of the discriminator loss landscape across 441 model parameters generated by shifting θ^* along two random directions. The map marks the selected points θ_1, θ_2, θ_3, and θ^*. Figure (b): PSNR curves from IG attacks on a ResNet18 model at θ_1, θ_2, θ_3, and θ^*, demonstrating that θ^* achieves the best reconstruction quality. Figure (c): 3D surface plots of the discriminator loss (top) and model accuracy (bottom) for the same 441 model parameters shifted θ^* along two random directions, illustrating the relationship between gradient vulnerability and model accuracy across the parameter space.

in Figure 5, EGGV demonstrates high stealthiness, achieving D-SNR values similar to those of the naive initialization methods Random, Xavier, and He. In contrast, Fishing and SEER show significantly higher D-SNR values, suggesting that clients can detect these methods more easily. This is because EGGV evenly enhances the leakage potential of all samples without introducing any gradient bias. In contrast, SOTA methods exhibit biased gradients across all layers, making them more prone to detection.

TABLE IV

RECONSTRUCTION RESULTS OF IG ATTACK RESNET18 POISONED BY EGGV WITH PROJECTION RATIOS OF 1.6%, 0.8%, AND 0.4%. EGGV AT 0.4% PROJECTION RATIO ACHIEVES THE BEST OVERALL PERFORMANCE

Initiation Method	PSNR ↑	SSIM ↑	LPIPS ↓
Random + IG	15.47002	0.25633	0.52080
Xavier [31] + IG	13.06239	0.21848	0.57367
He [32] + IG	12.70339	0.20885	0.72560
EGGV (Ours) (ρ : 1.60%) + IG	25.24615	0.80658	0.12803
EGGV (Ours) (ρ : 0.80%) + IG	27.63074	0.85759	0.09526
EGGV (Ours) (ρ : 0.40%) + IG	**28.62325**	**0.91402**	**0.07575**

C. Ablation Study

1) Evaluating Gradient Space Vulnerability Using the Discriminator Instead of End-To-End Iterative Attacks: We now turn to explore the effectiveness of using a discriminator to assess gradient space vulnerability, as opposed to traditional end-to-end iterative attacks. We randomly select two model directions, x and y, in the model parameter space and systematically shift the poisoned model parameter θ^* along these axes, generating 441 model parameters. The discriminator evaluates the gradient vulnerability for each of these parameters, and the resulting contour map of gradient vulnerability is shown in Figure 6(a). In this map, we select four points: θ_1, θ_2, θ_3, θ^*, with corresponding vulnerability scores of 14.99276, 5.42985, 1.28999, and 0.00655 assigned by the discriminator. Subsequently, we perform the IG attack on these four models with the CIFAR10 dataset. Figure 6(b) depicts the PSNR convergence during the attacks on these four models. As expected, θ^* yields the best reconstruction. The reconstructed images are highly similar to the original inputs, and the PSNR value remains the highest throughout the convergence process, significantly outperforming the other three parameters. These findings demonstrate the discriminator's effectiveness in evaluating gradient space vulnerability and predicting the likelihood of a successful reconstruction attack. In contrast to traditional end-to-end reconstruction methods, this method enables attackers to quickly identify and poison model parameters that could lead to attack failure, thereby improving the overall success rate for attacks.

2) Exploring the Relationship Between Gradient Space Vulnerability and Model Accuracy: To explore the relationship between gradient vulnerability and model accuracy, we conduct experiments by shifting the poisoned model parameter θ^* evenly 21 times along two randomly selected directions, x and y, generating 441 model parameters. Each parameter receives a gradient vulnerability score from the discriminator, visualized in the 3D surface plot at the top of Figure 6(c), representing the gradient vulnerability landscape across the parameter space. We then evaluate the classification accuracy of these same 441 model parameters using the CIFAR10 dataset, producing the lower plot of Figure 5. This plot shows the model accuracy at the same parameter positions as in the gradient vulnerability plot. A comparison between the two plots reveals that the model parameters with the highest gradient vulnerability do not coincide with those that yield the highest accuracy. In fact, model parameters with the greatest gradient vulnerability often show low accuracy, indicating no direct correlation between gradient vulnerability and model accuracy.

3) Exploring the Effect of Gradient Projection Ratios: A crucial component of the EGGV is the projector, which compresses high-dimensional gradients into a one-dimensional vector. Next, we examine how different projection ratios influence EGGV to enhance the vulnerability of the gradient space. We select commonly used Random, Xavier, and He initialization methods as comparison benchmarks, and set three different gradient projection ratios of 1.60%, 0.80%, and 0.40%. IG is used to conduct gradient leakage on the TinyImageNet dataset with the models initialized by the Random and EGGV. Comparison of experimental results in Table IV shows that EGGV consistently enhances the gradient

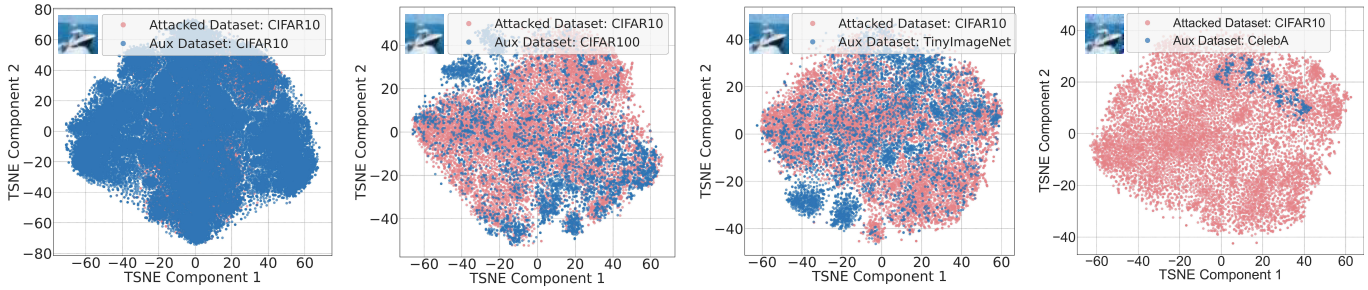


Fig. 7. Visualization of distributional differences between auxiliary and target datasets and IG reconstruction results on EGGV-poisoned models using each auxiliary dataset.

TABLE V

PERFORMANCE OF iDLG AND IG ATTACKS ON EGGV-POISONED MODELS WITH CIFAR10 AS TARGET AND DIFFERENT AUXILIARY DATASETS, SHOWING EGGV'S ROBUSTNESS TO DISTRIBUTIONAL SHIFTS

Attacked Dataset: CIFAR10	Auxiliary Datasets (PSNR ↑)			
	CIFAR10	CIFAR100	TinyImgaeNet	CelebA
EGGV+iDLG	29.70104	29.93789	27.83867	20.87152
EGGV+IG	31.96511	36.28306	36.64095	23.15686

vulnerability across all three projection ratios, outperforming Random, Xavier, and He initialization methods. We observe an interesting phenomenon: the smallest projection ratio of 0.4% achieves the best effect. This phenomenon can be attributed to the fact that smaller projection ratios force the model to embed more data features into the entire gradient, ensuring that the projected gradients retain enough data features to be inverted by the discriminator back to the original inputs. Under a smaller projection ratio, the model will more actively adjust its own parameters, thus containing more data features in the entire gradient, which is more conducive to the attack of subsequent attack methods.

4) Exploring the Effect of Distribution Differences Between Auxiliary and Target Datasets on EGGV: Next, we explore the effect of distributional differences between auxiliary and target datasets on the performance of the EGGV. We select CIFAR10 as the target dataset and CIFAR10, CIFAR100, and TinyImageNet as the auxiliary datasets. We additionally include CelebA [41] as an auxiliary dataset to assess EGGV with larger distribution gaps. To align the class counts with CIFAR10 to ensure compatibility for model poisoning, 10 classes are randomly sampled from CIFAR100, TinyImageNet, and CelebA to serve as auxiliary datasets. Table V reports the performance of iDLG and IG attacks on EGGV-poisoned models under the above setting. The results show that EGGV achieves comparable PSNR values across CIFAR10, CIFAR100, TinyImageNet, and CelebA auxiliary datasets, highlighting its robustness to distributional differences between auxiliary and target datasets, although the PSNR values for CelebA are relatively lower due to two factors: (1) CelebA exhibits a substantially different distribution from CIFAR10, making feature alignment more challenging; and (2) each CelebA class has few samples, so selecting 10 classes produces a small auxiliary dataset, insufficient to guide the model toward its most gradient-vulnerable point. In addition, we use the t-SNE algorithm [42] to reduce the dimensionality of these datasets to 2D and visualize their distributions, along with the corresponding reconstruction results shown in Figure 7. The visualizations confirm that the effectiveness of the EGGV attack is independent of distributional differences between the auxiliary and target datasets. In contrast, some SOTA methods, such as SEER, require the auxiliary dataset to be the training dataset of the target dataset, which limits their application to practical FL systems.

VII. CONCLUSION

In this work, we introduce a new backdoor-theoretic perspective to rethink and frame existing AGLAs. Through this lens, we theoretically identify that all prior AGLAs suffer from incomplete attack coverage and detectability issues. We further propose EGGV, a new solution that extends existing AGLAs to be more comprehensive and stealthier to address the above challenges. EGGV is the first AGLA capable of fully inverting all samples within a target batch while evading existing detection metrics. Extensive experiments demonstrate that EGGV significantly outperforms SOTA AGLAs in both stealthiness and attack coverage. These results encourage the FL community to explore further privacy protection mechanisms to counter these emerging security risks.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2016, pp. 1273–1282.

[2] K. Bonawitz et al., "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[3] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *Proc. 11th USENIX Symp. Operating Syst. Design Implement. (OSDI 14)*, 2014, pp. 571–582.

[4] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 14774–14784.

[5] M. V. Nowak, T. P. Bott, D. Khachaturov, F. Puppe, A. Krenzer, and A. Hekalo, "QBI: Quantile-based bias initialization for efficient private data reconstruction in federated learning," 2024, *arXiv:2406.18745*.

[6] J. Zhu and M. B. Blaschko, "R-GAP: Recursive gradient attack on privacy," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–17.

[7] H. Yang, M. Ge, K. Xiang, and J. Li, "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 818–830, 2023.

[8] K. Yue, R. Jin, C.-W. Wong, D. Baron, and H. Dai, "Gradient obfuscation gives a false sense of security in federated learning," in *Proc. 32nd USENIX Secur. Symp. (USENIX Secur)*, Aug. 2022, pp. 6381–6398. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/yue

[9] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," in *Proc. IEEE 8th Eur. Symp. Secur. Privacy (EuroSP)*, Jul. 2023, pp. 175–199.

[10] 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 *Proc. IEEE Symp. Secur. Privacy (SP)*, Jul. 2024, pp. 1287–1305.

[11] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *Proc. Int. Conf. Learn. Represent.*, Jun. 2021, pp. 1–25.

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

[13] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16937–16947.

[14] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via GradInversion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 16337–16346.

[15] J. Du et al., "SoK: On gradient leakage in federated learning," 2024, *arXiv:2404.05403*.

[16] Y. Wen, J. A. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 23668–23684.

[17] K. Garov, D. I. Dimitrov, N. Jovanović, and M. Vechev, "Hiding in plain sight: Disguising data stealing attacks in federated learning," in *Proc. Int. Conf. Learn. Represent.*, 2023, pp. 1–27.

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

[19] A. Wainakh et al., "User-level label leakage from gradients in federated learning," *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 2, pp. 227–244, Apr. 2022.

[20] K. Ma, Y. Sun, J. Cui, D. Li, Z. Guan, and J. Liu, "Instance-wise batch label restoration via gradients in federated learning," in *Proc. 11th Int. Conf. Learn. Represent.*, Aug. 2023, pp. 1–15.

[21] Y. Wang, J. Liang, and R. He, "Towards eliminating hard label constraints in gradient inversion attacks," in *Proc. 12th Int. Conf. Learn. Represent.*, Aug. 2024, pp. 1–19.

[22] J. Jeon, J. Kim, K. Lee, S. Oh, and J. Ok, "Gradient inversion with generative image prior," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29898–29908.

[23] Z. Li, J. Zhang, L. Liu, and J. Liu, "Auditing privacy defenses in federated learning via generative gradient leakage," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10132–10142.

[24] H. Gu, X. Zhang, J. Li, H. Wei, B. Li, and X. Huang, "Federated learning vulnerabilities: Privacy attacks with denoising diffusion probabilistic models," in *Proc. ACM Web Conf.*, May 2024, pp. 1149–1157.

[25] S. Zhang, J. Huang, Z. Zhang, and C. Qi, "Compromise privacy in large-batch federated learning via malicious model parameters," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2023, pp. 63–80.

[26] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Sep. 2022, pp. 2429–2443.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[28] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6/

[29] Kaggle.(2023). *Kaggle: Your Machine Learning and Data Science Community*. [Online]. Available: https://www.kaggle.com

[30] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *ACM SIGKDD Explor. Newslett.*, vol. 15, no. 2, pp. 49–60, Jun. 2014.

[31] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[33] W. Rudin, *Principles of Mathematical Analysis*. New York, NY, USA: McGraw-Hill, 1976. [Online]. Available: https://books.google.com.sg/books?id=kwqzPAAACAAJ

[34] R. Bartle and D. Sherbert, *Introduction to Real Analysis*. Hoboken, NJ, USA: Wiley, 2011. [Online]. Available: https://books.google.com.sg/books?id=YawbAAAAQBAJ

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009. [Online]. Available: https://www.cs.utoronto.ca/~kriz/learningfeatures-2009-TR.pdf

[37] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[38] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 2366–2369.

[39] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 586–595.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[41] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[42] L. Van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.