An Advanced Gradient Leakage Attack Against Duplicate Labels via Model Outputs Reconstruction

Kunlan Xiang, Haomiao Yang, *Senior Member, IEEE*, Meng Hao, *Student Member, IEEE*, Zikang Ding, Hongwei Li, *Fellow, IEEE*, Qingchuan Zhao, *Member, IEEE*, and Tianwei Zhang, *Member, IEEE*

***Abstract*—Federated learning (FL) is a prevalent distributed machine learning framework that allows multiple clients to train one model by uploading gradients without sharing data, enabling cooperative learning while preserving the training data privacy. Nevertheless, recent research has revealed that shared gradients can still expose clients' private training data. These attacks, however, often become ineffective in two practical scenarios: (1) gradients are computed on high-resolution data; (2) labels are duplicated within the attacked batch. In this work, we introduce an advanced Gradient Leakage Attack against Duplicate labels (GLAD), which can effectively recover high-resolution training data from gradients while considering duplicate labels, making it applicable in more realistic FL scenarios. The key technique of GLAD is to formalize the relationships between model outputs, gradients, model parameters, and training data labels. Based on these relationships, GLAD further reconstructs the model outputs and inverts the reconstructed model outputs back to the corresponding model inputs. Our method can achieve state-of-the-art recovery accuracy while ensuring efficiency. Extensive experimental results demonstrate that GLAD can reconstruct images of 224×224 pixels with a batch size of 256 with duplicate labels.**

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

I. Introduction

As a promising distributed machine learning paradigm, Federated Learning (FL) [1] typically consists of a server and multiple clients. The clients keep their private training data locally and only send model parameters or gradients to the server. The server then updates the global model with the aggregated contributions. Since the client data is always retained locally, FL is considered to well preserve the privacy of client data [2]. This has led to its growing popularity in various domains, including healthcare [3], finance [4], Internet of Things [5], and natural language processing [6].

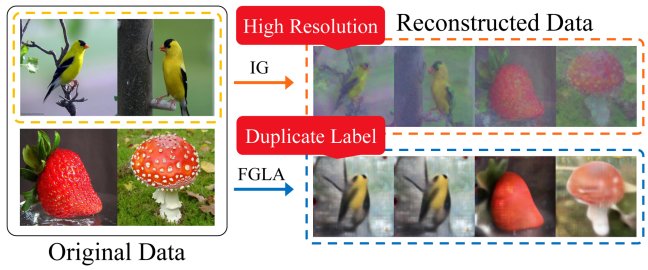

Fig. 1. Visualization of existing GLAs (IG [9] and FGLA [16]) on high-resolution data with duplicate labels in a batch. Data in the yellow boxes share the same label.

However, recent studies have demonstrated that FL is vulnerable to gradient leakage attacks (GLAs) [7], [8], [9], [10], [11]. By exploiting shared gradients, GLAs can reconstruct sensitive training samples, thus undermining the fundamental privacy assurances of FL. Broadly, GLAs can be classified into two main categories [12]: (1) optimization-based attacks [7], [8], [9], [13], [14]. This strategy performs optimization to find the data and label whose gradient has the minimal distance from the observed one. (2) Analytics-based attacks [10], [11], [15], [16]. This strategy directly reconstructs the data from the gradient through a series of analytical equations.

Nevertheless, these two types of GLAs may struggle to work in realistic FL settings. First, most existing optimization-based attacks are limited to small-batch or low-resolution data. For example, the DLG [7] is only effective for images with a batch size smaller than 8 and a low resolution of 32×32 pixels. However, high-resolution data, more commonly used in FL systems, poses greater optimization challenges as highlighted by DLG [7], particularly due to the increased number of variables involved in optimizing either high-resolution or large batch size scenarios. Fig. 1 shows the ineffectiveness of the "Inverting Gradient" (IG) attack [9] in a simple FL setting containing high-resolution training data.

Second, analytics-based attacks are often only effective when targeting a single sample [10], [13] or models consisting of only fully connected layers (FCL) [15], [17]. These methods become

Received 10 May 2024; revised 2 December 2025; accepted 23 January 2026. Date of publication 29 January 2026; date of current version 12 May 2026. This work was supported 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. *(Corresponding author: Haomiao Yang.)*

Kunlan Xiang, Haomiao Yang, Meng Hao, Zikang Ding, and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China (e-mail: kunlxiang@gmail.com; haomyang@uestc.edu.cn; menghao303@gmail.com; dzkang0312@163.com; hongweili@uestc.edu.cn).

Qingchuan Zhao is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qizhao@cityu.edu.hk).

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

Our source code is available at https://github.com/SuperX612/GLAD.

Digital Object Identifier 10.1109/TDSC.2026.3659196

1545-5971 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

infeasible to directly derive the analytical equations when more complex layers are in the target model, which is common for FL tasks.

Third, most existing GLAs rely on an unrealistic assumption that each label in the attacked batch must be unique. Recent studies [10], [16] highlight that label duplication poses severe challenges to data reconstruction [10], [11], [18], [19]. Specifically, for optimization-based methods, Qian et al. [18] demonstrate that duplicate labels significantly complicate the optimization process, resulting in a decline in the quality of the reconstructed results. Similarly, for analytics-based attacks, Fowl et al. [11] and Yang et al. [16] emphasize that analytics-based attacks become ineffective once duplicate labels occur. Fig. 1 illustrates the ineffectiveness of "Fast Gradient Leakage Attack" (FGLA) [16] on samples with duplicate labels.

To address the above challenges, we introduce GLAD, an advanced gradient leakage attack that effectively reconstructs high-resolution data of large batch sizes without assuming a specific label distribution. The key idea of GLAD is to decompose the GLA into two tasks: model outputs reconstruction and model inversion. This is different from all prior works which perform input reconstruction directly. Specifically, GLAD consists of two steps. (1) We perform an in-depth analysis of the gradients and propose two key propositions. The first proposition formalizes an exact relationship between the model outputs, gradients, and model weights. The second proposition offers an approximate relationship between the individual model output of the batch data, batch size, number of duplicate labels, and gradient of biases in FCL. Based on these theoretical evidences, we establish equations for every value in the model outputs and propose a customized adversarial optimization objective to reconstruct the model output of each sample. (2) We introduce a novel model inversion attack (MIA), which inverts the extracted model outputs to the model inputs. Our method is the first to bridge two distinct classes of attacks: GLA and MIA.

GLAD can effectively reconstruct the model outputs from the gradient in a general FL setting, which cannot be achieved in prior works. By accurately reconstructing every value in the model output for each sample, GLAD addresses the challenge of duplicate labels. In addition, GLAD optimizes the model outputs with a smaller optimization space proportional to the batch size and number of classes, enhancing efficiency and addressing the challenge of high-resolution data reconstruction.

Our main contributions are:

- We propose GLAD, an advanced and practical gradient leakage attack against FL. GLAD addresses the challenge of reconstructing high-resolution data, and is the first to handle the label duplication setting.
- We propose two propositions for disclosing the relationships between model outputs, model parameters, gradient, batch size, and number of duplicate labels, etc, thus facilitating the application of MIAs for GLAs.
- Extensive experiments show that GLAD outperforms SOTA methods, exhibiting strong robustness to label distributions, batch sizes, resolutions, popular defenses, and datasets. GLAD can reconstruct data of 224×224 pixels in a batch size of 256 in just $\frac{1}{200}$ of the time required by optimization-based methods, thereby enhancing the efficiency and accuracy of GLAs. GLAD proves that gradients can still leak data in more complex FL settings.

TABLE I
KEY NOTATIONS LIST

Notation	Description
$\mathcal{S}$	The honest-but-curious sever
$\mathcal{D}_i$	Private / Training data sets of client i
$\boldsymbol{x}$	Target data / Input data to be reconstructed
y	The label corresponding to $\boldsymbol{x}$
$\mathcal{M}(\boldsymbol{W}, \boldsymbol{b})$	The global model / The model to be attacked
$(\boldsymbol{W}, \boldsymbol{b})$	Parameters of the global model
$(\nabla \boldsymbol{W}, \nabla \boldsymbol{b})$	The gradients of training data
$(\nabla \boldsymbol{W}_{(i)}, \nabla \boldsymbol{b}_{(i)})$	The gradients of training data of client i
$\mathcal{D}_{aux}$	The auxiliary data set
$(\nabla \boldsymbol{W}_i, \nabla \boldsymbol{b}_i)$	The gradients of i^{th} layer in global model
$(\nabla \boldsymbol{W}_i^j, \nabla \boldsymbol{b}_i^j)$	The j^{th} row of the gradient matrix for the i^{th} layer in the global model
$\boldsymbol{x}'$	The dummy inputs / Data
$\boldsymbol{y}'$	The dummy label
$l(\cdot, \cdot)$	The loss function used to train global model
$\hat{\boldsymbol{y}}$	The model outputs for target data
$\hat{\boldsymbol{y}}'$	The dummy model outputs for target data
B	Batch size
H	The number of neurons in the input layer
C	The number of classification classes
$\mathcal{L}_w$	The optimization objective for the dummy model outputs related to the model's weights and its gradients
$\mathcal{L}_b$	The optimization objective for the dummy model outputs related to the model's bias and its gradients
$\mathcal{L}_{loss}$	The optimization objective for the dummy model outputs related to the model loss
$\mathcal{L}_{total}$	The total optimization objectives for the dummy model outputs consisting of $\mathcal{L}_w$, $\mathcal{L}_b$ and $\mathcal{L}_{loss}$
$\mathcal{G}$	The data generator by inputting feature maps
$\boldsymbol{\lambda}_j$	The occurrence number of label j in the $\boldsymbol{y}$

Roadmap: In Section II, we introduce some preliminary knowledge essential to this work, including federated learning, existing attacks and the motivation of this work. Section III delves into the details of the proposed method. In Section IV, we provide experimental evidence and analysis. Section V discusses the limitations, possible extensions and a defense solution. Finally, we conclude in Section VI.

II. BACKGROUND

In this section, we present the background of federated learning, a distributed learning framework aimed at collaborative model training with an emphasis on data privacy. We outline the core components and iterative process of FL and present related challenges. We then review existing GLAs that exploit shared gradients to reconstruct private data. Finally, we highlight the limitations of these approaches, which motivate our proposed method. Key notations used in our paper are presented in Table I for reference.

A. Federated Learning

FL is first proposed by Google in 2016 [1] to enable collaborative model training across decentralized mobile devices. The core goal of FL is to secure information during data exchange while protecting endpoint data privacy [20], [21], [22]. As a distributed learning framework, FL typically involves N clients

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

and a central server $\mathcal{S}$. Each client has a dataset $\mathcal{D}_i$, which contains n data pairs $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$. Each learning round generally has four steps: ① The server distributes the latest global model $\mathcal{M}$ to the clients; ② Each client i inputs its data batch $(\boldsymbol{x}, \boldsymbol{y})$ into the global model $\mathcal{M}(\boldsymbol{W}, \boldsymbol{b})$ and computes the corresponding gradient $(\nabla \boldsymbol{W}_{(i)}, \nabla \boldsymbol{b}_{(i)})$. ③ All the clients send their gradients back to the server; ④ The parameter server performs gradient aggregation and updates the global model by $(\boldsymbol{W}^{(t+1)}, \boldsymbol{b}^{(t+1)}) = (\boldsymbol{W}^{(t)}, \boldsymbol{b}^{(t)}) - \eta \sum_{i=1}^{N} (\nabla \boldsymbol{W}_{(i)}, \nabla \boldsymbol{b}_{(i)})$. This iterative process continues until the global model converges.

B. Gradient Leakage Attacks

Despite the natural advantages of FL in protecting data privacy, a series of challenges are emerging as its application domains continue to expand, including communication efficiency [23], model personalization [24], and adversarial attacks [25]. A major concern in FL is the privacy risk from data leakage [7], where attackers can potentially infer sensitive data by analyzing the shared gradient. Such gradient leakage attacks (GLAs) have been widely studied, and existing attack techniques can be classified into two categories [26].

Optimization-based methods: Optimization-based methods leverage some optimization solutions to identify the data whose gradient is as close as possible to the observed gradient. Zhu et al. [7] first proposed the "Deep Leakage from Gradients" (DLG) method, which reconstructs the data from gradients using the L-BFGS loss function [27]. DLG can work well on low-resolution images of 32×32 pixels and batch sizes of up to 8. Subsequently, Zhao et al. [13] improve DLG by revealing the ground-truth label through FCL gradient signs. However, this method is limited to reconstructing a single sample. Geiping et al. [9] propose the "Inverting Gradient" (IG) by using cosine similarity as a distance metric and total variation regularization for image reconstruction. Although IG successfully recovers sample images from the ImageNet dataset resized to 32×32 pixels, it requires up to 2000 iterations, taking approximately two hours for a single image reconstruction. Yin et al. [28] further improve DLG using multiple optimization terms, including the standard image priors [29] that penalize the total variance and l_2 norm as well as strong priors batch norm. However, this reliance on BN statistics limits its applicability. Addressing this, Fan et al. [30] propose FEDLEAK, which employs partial gradient matching and regularization to enable reconstruction without relying on unrealistic BN statistics. Shifting focus to generative priors, Jeon et al. [8] optimize low-dimensional features and a generator to utilize a pre-trained generative model to invert the gradient. Based on this, GGL [14] only optimizes low-dimensional features to invert the gradient. Recent studies introduce diffusion models to gradient inversion tasks. Gu et al. [31] propose GGDM, which uses shared gradients to guide the diffusion sampling process. Similarly, Wu et al. [32] develop DGGI, which combines diffusion priors with consistency regularization for data reconstruction. Although diffusion models enable their successful reconstruction on high-resolution data (e.g., 224×224), they are constrained by relatively small batch sizes, with Gu et al. [31] restricted to a batch size of 8 and Wu et al. [32] to a batch size of 48.

Hatamizadeh et al. [33] design a gradient inversion attack for vision transformers. Yang et al. [34] investigate a new initialization for better quality under compressed gradient, but it is only limited to the batch size of 1 and resolution of 32× 32 pixels. Yue et al. [35] explore a GLA at a semantic level and measure the corresponding image privacy leakage. Wu et al. [36] train a model to learn the mapping between the gradient and corresponding inputs. Liang et al. [37] introduce EGIA, highlighting threats from external adversaries in the communication channel.

Analytics-based methods: Analytics-based methods aim to compute private data directly using analytical equations associated with the gradient. Phong et al. [15] introduce an equation that utilizes the ratio of the gradient of the weight to the gradient of the bias in the FCL for data reconstruction. While innovative, this method is limited to the scenario of a single sample and a single FCL. Fan et al. [17] extend this approach to handle multi-layer perceptrons. Hitaj et al. [38] propose DMU-GAN, which reconstructs data points representing the generalized characteristics of each class. Zhu et al. [10] proposed R-GAP, which provides a closed-form recursive procedure to recover data from gradients. However, this method is limited to specific scenarios: a batch size of 1, binary classification, and simple model architectures. Fowl et al. [11] retrieve portions of data of batch size 64 by modifying the model structure, but it can be easily detected. Boenisch et al. [39] propose the "trap weights" to augment the number of neurons in the model, enhancing the possibility of directly computing training data from the gradient. "Cocktail Party Attack" (CPA) [40] treats the gradient inversion as a Blind Source Separation (BSS) problem, and uses independent component analysis to address this BSS problem. A more recent work FGLA [16] uses an analytical method to separate the feature maps of original data from gradients and feed it into a pre-trained generator, enabling rapid reconstruction of a data batch. Furthermore, Zhao et al. [41] presented an "Attacker's Cookbook" for training downstream models using leaked data by the above attacks. This work underscores that privacy breaches extend beyond visual inspection to practical data utility.

C. Motivation

Table II briefly summarizes existing GLAs from different perspectives. We observe that these attacks are restricted to unrealistic settings. Specifically, a practical FL system typically involves large-scale data (refer to high-resolution images of large batch size in this paper) and has data of the same labels in each batch. However, most of these attacks can only be applied to reconstruct small-scale training data and batch sizes. They also assume that data in each batch has no duplicated labels, which normally does not hold. More importantly, they can be easily mitigated with simple defense mechanisms, such as masking the gradient with noise, or compressing the gradient.

Our goal is to design an advanced gradient leakage attack, which is effective against high-resolution data of large batch size

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

TABLE II
A COMPARISON OF GLAD WITH EXISTING METHODS BASED ON CRITERIA: BATCH SIZE, IMAGE RESOLUTION, THE NUMBER OF CLASSES C, RECONSTRUCTION SPEED, STRICT DEFENSE, AND DUPLICATE LABELS

Method	Max Batch Size × Image Resolution	$C > 2$	Fast Speed	Strict Defense[1]	Duplicate Label
[7]	8×(32×32)	●	○	○	◐
[8]	48×(64×64)	●	○	○	◐
[9]	100×(32×32)	●	○	○	◐
[13]	1×(32×32)	●	○	○	◐
[14]	1×(224×224)	●	○	○	◐
[31]	8×(256×256)	●	○	○	◐
[32]	48×(224×224)	●	○	○	◐
[34]	1×(32×32)	●	○	○	◐
[35]	128×(128×128)	●	○	○	○
[10]	1×(32×32)	○	●	○	○
[11]	64×(224×224)	●	●	○	○
[15]	1×(32×32)	●	●	○	○
[16]	256×(224×224)	●	●	○	○
[17]	8×(32×32)	●	●	○	○
Ours	(C+1)×(336×336)	●	●	●	●

[1] Strict defense refers to a highly compressed gradient with a compression ratio of 99.9%, and a noisy gradient with $\sigma = 0.01$.
[2] ●: able; ○: unable; ◐: partly unable (duplicate labels complicate the optimization process, resulting in a decline in the quality of reconstruction result and even partial failure of reconstruction).

and accounts for the repetition of labels. Furthermore, our attack aims to remain robust in the face of popular defense strategies, such as noise addition and gradient compression, ensuring it is not easily neutralized.

III. METHODOLOGY

Motivated by the above observations, we introduce GLAD, a new gradient leakage attack against practical FL systems. We first present our threat model. Then we give the overview of GLAD, followed by the details of its two stages. Along with the descriptions, we provide a theoretical analysis elucidating the conditions and principles under which GLAD works.

A. Threat Model

We consider an FL system consisting of a parameter server $\mathcal{S}$ and multiple clients. We assume that the parameter server $\mathcal{S}$ is an honest but curious adversary, and it aims to reconstruct the client's data from the uploaded gradient while following the FL system protocol. The target model in our attack is the convolutional neural network with the FCL as the final layer. This setting is widely adopted for image classification and widely assumed as the target model in most existing methods [7], [9], [11], [16]. $\mathcal{S}$ has access to the gradients uploaded by the clients, the structure and parameters of the global model $\mathcal{M}(\boldsymbol{W}, \boldsymbol{b})$. Following prior works [14], [16], we assume $\mathcal{S}$ has access to an auxiliary dataset $\mathcal{D}_{aux}$, which can be easily obtained from open-source websites.

B. Overview

Fig. 2 illustrates the workflow of GLAD, and Algorithm 1 outlines the corresponding pseudocode. Our method comprises two main steps: 1) Model outputs reconstruction: we derive three innovative equations to formalize the relationship between the model outputs, gradients, model parameters, and labels explicitly decoupling them from the model inputs. We then employ these equations in an optimization technique to reconstruct the model outputs. 2) Inversion from model outputs to inputs: we propose a model inversion technique that inverts the obtained model outputs back to the model inputs. This step first uses the decomposed model outputs to invert to feature maps and then directly generates the original data through a trained generator.

Algorithm 1: Private Data Leakage From the Captured Gradients.

Input: $(\boldsymbol{W}, \boldsymbol{b})$: the global model parameters, $(\nabla \boldsymbol{W}, \nabla \boldsymbol{b})$: gradients calculated by training data, $\mathcal{D}_{aux}$: the auxiliary dataset to train the generator, and $\boldsymbol{y}$: training data label.
Output: dummy training data $\boldsymbol{x}'$

procedure GLAD($\boldsymbol{W}, \boldsymbol{b}$), $(\nabla \boldsymbol{W}, \nabla \boldsymbol{b})$, $\boldsymbol{y}$, $\mathcal{D}_{aux}$
  $\hat{\boldsymbol{y}}'_1 \leftarrow \mathcal{N}(0, 1)$
  for $i \leftarrow 1$ to $Epochs$ **do**
    $\mathcal{L}oss^i_{total} \leftarrow \alpha_1 \mathcal{L}_w(\hat{\boldsymbol{y}}'_i) + \alpha_2 \mathcal{L}_b(\hat{\boldsymbol{y}}'_i) + \alpha_3 \mathcal{L}_{loss}(\hat{\boldsymbol{y}}'_i)$
    $\boldsymbol{y}'_{i+1} \leftarrow \boldsymbol{y}'_i - \eta \nabla_{\boldsymbol{y}'_i} \mathcal{L}oss^i_{total}$
  end for
  $\boldsymbol{f}' = (\partial l(\hat{\boldsymbol{y}}', \boldsymbol{y}) / \partial \hat{\boldsymbol{y}}')^{-1} \cdot \nabla \boldsymbol{W}_L$
  $\mathcal{G} \leftarrow$ Train a generator by inputting feature maps and outputting corresponding images using model parameters $(\boldsymbol{W}, \boldsymbol{b})$ and auxiliary dataset $\mathcal{D}_{aux}$.
  $\boldsymbol{x}' \leftarrow \mathcal{G}(\boldsymbol{f}')$
  return $\boldsymbol{x}'$
end procedure

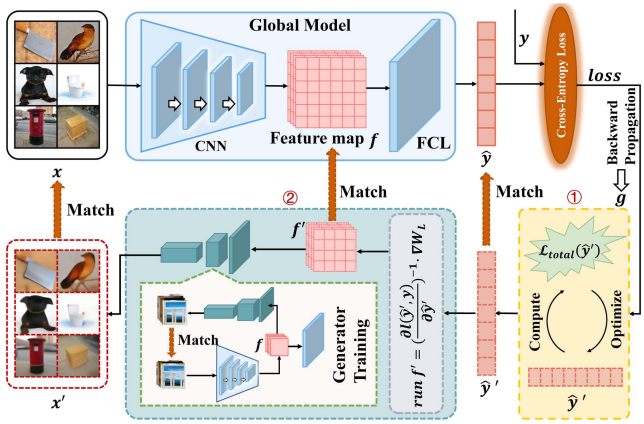

Fig. 2. General overview of the proposed method GLAD. GLAD comprises two steps. ① Decomposition of model outputs through the optimization; ② Model inversion, which includes the decomposition of the feature maps corresponding to each sample in the batch based on the model outputs obtained in ① through the analytic equation, and direct generation of the original batch based on the decomposed feature maps.

C. Model Outputs Reconstruction

To understand the computational aspects of the proposed method, we first formalize the standard data processing flow

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

within a Convolutional Neural Network (CNN). Upon receiving the global model, a client inputs a private data batch $\boldsymbol{x} \in \mathbb{R}^{B\times Channel\times Height\times Width}$ into a CNN. The data $\boldsymbol{x}$ is first processed by a convolutional layer, producing feature maps $\boldsymbol{f} \in \mathbb{R}^{B\times H}$. $\boldsymbol{f}$ is then processed by a final FCL (also L^{th} layer) via $\hat{\boldsymbol{y}} = \boldsymbol{f} \cdot \boldsymbol{W}_L + \boldsymbol{b}_L$, thus yielding model outputs $\hat{\boldsymbol{y}} \in \mathbb{R}^{B\times C}$. Then, the model outputs $\hat{\boldsymbol{y}}$ and the ground-truth labels $\boldsymbol{y} \in \mathbb{R}^{B\times 1}$ are fed into the loss function to compute the $loss$. Finally, the client computes the gradients $(\nabla \boldsymbol{W}, \nabla \boldsymbol{b})$ via backpropagation based on $loss$. Here, $\cdot$, B, H, and C denote the matrix multiplication, batch size, the number of neurons in the input layer, and classification classes, respectively.

In optimization-based attacks [7], [9], [28], data reconstruction can be formalized as an optimization problem:

$$\boldsymbol{x}^*, \boldsymbol{y}^* = \arg\min_{\boldsymbol{x}', \boldsymbol{y}'} \text{dis}(\nabla_{\boldsymbol{\theta}} l(\boldsymbol{x}', \boldsymbol{y}'), \nabla_{\boldsymbol{\theta}} l(\boldsymbol{x}, \boldsymbol{y})) + \text{Reg}(\boldsymbol{x}'), \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{y}$ are the private training data and its labels respectively, $\boldsymbol{x}', \boldsymbol{y}'$ are the dummy inputs and dummy label respectively, $\text{dis}(\cdot,\cdot)$ is the function to measure the distance of the dummy and the actual gradient, $\boldsymbol{\theta}$ is the parameters of the global model including weights $\boldsymbol{W}$ and bias $\boldsymbol{b}$, $l(\cdot,\cdot)$ is the loss function used to calculate the gradient, and $\text{Reg}(\cdot)$ is a regular term for $\boldsymbol{x}'$. However, the performance of these attacks degrades sharply with increasing batch size or image resolution [7], due to a multiplicative increase in the number of variables to be optimized. To address this challenge, we propose an innovative method that optimizes model outputs by analytically decoupling the relationship between gradients and model inputs, focusing solely on the relationship between gradients and model outputs. Consequently, our method searches the model outputs with a time complexity of $O(B \cdot C)$, which is independent of the resolution of the input data and requires searching for far fewer variables compared to optimizing model inputs with a time complexity of $O(B \cdot Channel \cdot Height \cdot Width)$. For example, for an ImageNet subset (10 classes) with a batch size of 8, a channel number of 3, and an image resolution of 224×224, the optimization-based methods need to optimize $8 \times 3 \times 224 \times 224 = \mathbf{1,204,224}$ variables, while our method needs to optimize only $8 \times 10 = \mathbf{80}$ variables. In this case, the traditional optimization-based methods need to solve **15,052** times more variables than our method. This extensive computational demand can render the optimization-based attack impractical in many scenarios. Extracting the model outputs directly from the gradient is not straightforward, however; it requires establishing a direct relationship between the gradient and the model outputs while eliminating the relationship between the gradient and the unknown inputs. The following describes our proposed technique to achieve the goal of reconstructing the model outputs.

Firstly, we propose Proposition 1 to establish a generalized mathematical framework that connects the parameters of FCL, the gradient of the FCL, the model outputs, and the partial derivative of the loss between the model outputs and the ground-truth labels with respect to the model outputs.

Proposition 1:

$$\nabla \boldsymbol{W}_L \cdot \boldsymbol{W}_L = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot (\hat{\boldsymbol{y}} - \boldsymbol{b}_L), \tag{2}$$

where $\boldsymbol{W}_L$ indicates the weights of FCL (also L^{th} layer in global model), $\boldsymbol{b}_L$ indicates the bias of FCL, $\hat{\boldsymbol{y}}$ denotes the model outputs for private data, $\boldsymbol{y}$ denotes the training data label, and $l(\cdot,\cdot)$ denotes the loss function used to train the global model.

Proof: The forward propagation of FCL can be written as $\hat{\boldsymbol{y}} = \boldsymbol{f} \cdot \boldsymbol{W}_L + \boldsymbol{b}_L$, thus we can have:

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}_L} = \boldsymbol{f}, \tag{3}$$

based on the chain rule of derivation, we can reach:

$$\nabla \boldsymbol{W}_L \cdot \boldsymbol{W}_L = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}_L} \cdot \boldsymbol{W}_L, \tag{4}$$

we substitute $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}_L}$ in (4) with its expression from (3) as follows:

$$\begin{aligned}\nabla \boldsymbol{W}_L \cdot \boldsymbol{W}_L &= \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}_L} \cdot \boldsymbol{W}_L = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \boldsymbol{f} \cdot \boldsymbol{W}_L \\ &= \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot (\hat{\boldsymbol{y}} - \boldsymbol{b}_L)\end{aligned} \tag{5}$$

□

Given that $\nabla \boldsymbol{W}_L$, $\boldsymbol{W}_L$, and $\boldsymbol{b}_L$ are accessible to the attacker and $\boldsymbol{y}$ is proven to be perfectly reconstructed even with label duplication [42], Proposition 1 facilitates the computation of the model outputs $\hat{\boldsymbol{y}}$. It can be observed that Proposition 1 establishes $C \times C$ equations through the matrix $\nabla \boldsymbol{W}_L \cdot \boldsymbol{W}_L$ of size $C \times C$ in the left side of the Proposition 1. Then, to fully utilize the data information embedded in the gradient of the bias and to augment our method with additional equations, we formulate the second system of equations as follows:

$$\nabla \boldsymbol{b}_L = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \frac{\partial \hat{\boldsymbol{y}}_{(:,j)}}{\partial \boldsymbol{b}_L^j} = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \mathbb{I}, \tag{6}$$

where $\mathbb{I} \in \mathbb{R}^{B\times 1}$, $\mathbb{I}_i = 1$ for $i \in [1, B]$, and $j \in [1, C]$ is the index of the classification categories. Since $\nabla \boldsymbol{b}_L$ is available to the attacker and $\boldsymbol{y}$ can be almost reconstructed perfectly by the attacker, we can obtain the model outputs through (6), which provides C equations.

Given there are no appropriate mathematical techniques to analytically deduce model outputs $\hat{\boldsymbol{y}}$ directly through the equations provided by Proposition 1 and (6), we have to employ an optimization technique to search for model outputs $\hat{\boldsymbol{y}}$. In the optimization to search model outputs, we initialize the $\hat{\boldsymbol{y}}$ randomly, which we call dummy model outputs $\hat{\boldsymbol{y}}'$ and then continuously adjust and optimize it until it satisfies the established equations. Accordingly, we define the optimization objective $\mathcal{L}_1$ of model outputs as:

$$\mathcal{L}_1\left(\hat{\boldsymbol{y}}'\right) = \alpha_1 \overbrace{\left\| \nabla \boldsymbol{W}_L \cdot \boldsymbol{W}_L - \frac{\partial l\left(\hat{\boldsymbol{y}}', \boldsymbol{y}\right)}{\partial \hat{\boldsymbol{y}}'} \cdot \left(\hat{\boldsymbol{y}}' - \boldsymbol{b}_L\right) \right\|^2}^{Proposition1:\mathcal{L}_w}$$

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

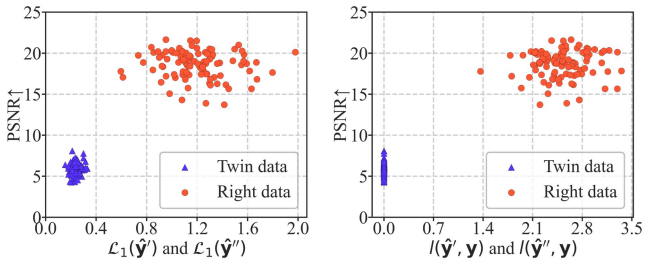

Fig. 3. **Left**: PSNR values vs. $\mathcal{L}_1(\hat{\boldsymbol{y}}')$ for right data and PSNR values vs. $\mathcal{L}_1(\hat{\boldsymbol{y}}'')$ for twin data. **Right**: PSNR vs. $l(\hat{\boldsymbol{y}}', \boldsymbol{y})$ for right data and PSNR vs. $l(\hat{\boldsymbol{y}}'', \boldsymbol{y})$ for twin data.

$$+ \alpha_2 \overbrace{\left\| \nabla \boldsymbol{b}_L - \frac{\partial l(\hat{\boldsymbol{y}}', \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}'} \cdot \mathbb{I} \right\|^2}^{Eq\text{·}(6):\mathcal{L}_b} \tag{7}$$

However, the model outputs is not always reconstructed perfectly through the optimization objective in (7). Proposition 1 and (6) provide a total of $C \times C + C$ equations, and the model outputs $\hat{\boldsymbol{y}}$ we need to search is $B \times C$ variables. Thus the client's batch size B directly determines the feasibility of recovering $\hat{\boldsymbol{y}}$ through the above equations as the number of classes C for a selected target model is given. When $B > C + 1$, the established system of equations is underdetermined with the number of variables greater than the number of equations. Consequently, recovering a unique $\hat{y}$ is impossible. Conversely, for $B \leq C$, the system of equations is determined, allowing for the recovery of $\hat{\boldsymbol{y}}$. So we assume $B <= C + 1$ in the attacked system. Although our attack requires the number of classes to be higher than the batch size, it can tackle any batch size and can reconstruct every sample in the batch. Our attack is also the first to be able to handle label duplication. The setting with the batch size B is less than the number of classes is entirely practical and common in FL systems with a large variety of classification categories. But, there may be multiple solutions in the situation of $B \leq C$ due to the non-monotonicity of $\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})/\partial \hat{\boldsymbol{y}}$ [10], of which only one is the right solution, and the remaining solutions are referred to in prior work R-GAP [10] as the twin solutions. Therefore, for cases where $B \leq C$, we need to observe the difference between the right solution and the twin solutions and use this difference to guide the optimization toward the right solution.

To investigate the difference between the right data with higher PSNR values (a metric assessing the similarity between the reconstructed data and the actual data) and the twin data with lower PSNR values, we conduct experiments and collect 100 batches of right data and 100 batches of twin data. The left of Fig. 3 shows PSNR values vs. $\mathcal{L}_1$ (computed through (7)) of twin data and right data. Unexpectedly, the twin data instead exhibit lower $\mathcal{L}_1$ values. Considering that the reconstructed results depend solely on the reconstructed dummy model outputs, our intuition suggests that the reconstructed dummy model outputs $\hat{\boldsymbol{y}}''$ of twin data are not close to the actual model outputs but are close to the ground-truth labels $\boldsymbol{y}$, which results in a much smaller $l(\hat{\boldsymbol{y}}'', \boldsymbol{y})$ of twin data than that of right data. So the gradient of the twin data is much smaller than that of the right data. Thus the gradient as well as $\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})/\partial \hat{\boldsymbol{y}}$ involved in $\mathcal{L}_1$ computation for the twin data is smaller, allowing it to achieve a lower $\mathcal{L}_1$ despite not matching the actual data. To verify our intuition, we compute $l(\hat{\boldsymbol{y}}', \boldsymbol{y})$ for 100 batches of right data and $l(\hat{\boldsymbol{y}}'', \boldsymbol{y})$ for 100 batches of twin data and show results in the right of Fig. 3. These results prove our intuition. Therefore, we can guide the optimization direction towards the right data by constraining the loss between the dummy model outputs $\hat{\boldsymbol{y}}'$ and the training data label $\boldsymbol{y}$. Following these observations, we propose the Proposition 2 for approximating the loss of the attacked batch to guide the optimization of the dummy model outputs.

Proposition 2:

$$l(\hat{\boldsymbol{y}}, \boldsymbol{y}) \approx -\frac{1}{B} \sum_{i=1}^{B} \ln \left(\nabla \boldsymbol{b}_L^{\boldsymbol{y}_{(i)}} + \boldsymbol{\lambda}_{\boldsymbol{y}_{(i)}} / B \right), \tag{8}$$

where $\boldsymbol{y}$ indicates the ground-truth label, $\boldsymbol{y}(i)$ indicates the ground-truth label for the i^{th} sample in a batch, $\nabla \boldsymbol{b}_L$ represents the gradient of the bias term in the FCL, and $\boldsymbol{\lambda}_j$ indicates the occurrence number of label j in the $\boldsymbol{y}$.

Proof: Due to the common use of the cross-entropy loss function in image classification tasks, we use the cross-entropy loss function as the loss function to train the global model, so the loss function can be defined as (9):

$$l(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\frac{1}{B} \sum_{i=1}^{B} \ln \frac{e^{\hat{\boldsymbol{y}}_{(i, \boldsymbol{y}(i))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,j)}}}, \tag{9}$$

where $\hat{\boldsymbol{y}}$ indicates the model outputs, which is also the outputs of the FCL and $\hat{\boldsymbol{y}}_{(i,j)}$ indicates the logit value for j^{th} class when i^{th} sample in a batch is given as input to the model. We can write the gradient value $\nabla \boldsymbol{b}_L^j$ w.r.t. the bias connected to the j^{th} outputs representing the j^{th} class confidence in the outputs layer as following (10):

$$\begin{aligned} \nabla \boldsymbol{b}_L^j &= \nabla \hat{\boldsymbol{y}}_{(:,j)} \cdot \frac{\partial \hat{\boldsymbol{y}}_{(:,j)}}{\partial \boldsymbol{b}_L^j} = \nabla \hat{\boldsymbol{y}}_{(:,j)} \cdot \mathbb{I} \\ &= -\frac{\boldsymbol{\lambda}_j}{B} + \frac{1}{B} \sum_{i=1}^{B} \frac{e^{\hat{\boldsymbol{y}}_{(i, \boldsymbol{y}(i))}}}{\sum_{t=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,t)}}}, \end{aligned} \tag{10}$$

when the j in (10) equal to $\boldsymbol{y}(k^*)$, then we have (11):

$$\nabla \boldsymbol{b}_L^{\boldsymbol{y}(k^*)} = -\frac{\boldsymbol{\lambda}_{\boldsymbol{y}(k^*)}}{B} + \frac{1}{B} \sum_{i=1}^{B} \frac{e^{\hat{\boldsymbol{y}}_{(i, \boldsymbol{y}(i^*))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,j)}}}, \tag{11}$$

where $k^* \in [1, B]$ denotes the index of a specific sample within a batch. In the initialization or pre-training phase of model training, all logit values in the model outputs $\hat{\boldsymbol{y}}$ are similar, due to the model's lack of discriminative ability over the input samples. This phenomenon can be expressed as:

$$\hat{\boldsymbol{y}}(i,j) \approx \hat{\boldsymbol{y}}(k,t), \tag{12}$$

where $i, k \in [1, B]$ indicate the index of the sample within the input batch, and $j, t \in [1, C]$ indicate the index of the classes. Based on this observation and the (11), we can further formulate

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

TABLE III
AVERAGE $L2$ ERRORS OF APPROXIMATION AT THE MODELS WITH DIFFERENT BATCH SIZE B

Batch size	8	16	32	64	128	256
$\mathcal{L}_{loss}$	0.5764	0.71750	0.84480	0.93210	0.95530	0.96800

the following equation:

$$\frac{e^{\hat{\boldsymbol{y}}_{(k^*,\boldsymbol{y}(k^*))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(k^*,j)}}} \approx \frac{1}{B}\sum_{i=1}^{B}\frac{e^{\hat{\boldsymbol{y}}_{(i,\boldsymbol{y}(k^*))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,j)}}} = \nabla \boldsymbol{b}_L^{\boldsymbol{y}(k^*)} + \frac{\boldsymbol{\lambda}_{\boldsymbol{y}(k^*)}}{B}, \tag{13}$$

Then, we replace $\frac{e^{\hat{\boldsymbol{y}}_{(i,\boldsymbol{y}(i))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,j)}}}$ in (9) with the expression from (13), as follows:

$$\begin{aligned} l(\hat{\boldsymbol{y}}, \boldsymbol{y}) = & -\frac{1}{B}\sum_{i=1}^{B}\ln\frac{e^{\hat{\boldsymbol{y}}_{(i,\boldsymbol{y}(i))}}}{\sum_{j=1}^{C} e^{\hat{\boldsymbol{y}}_{(i,j)}}} \\ \approx & -\frac{1}{B}\sum_{i=1}^{B}\ln\left(\nabla \boldsymbol{b}_L^{\boldsymbol{y}(i)} + \frac{\boldsymbol{\lambda}_{\boldsymbol{y}(i)}}{B}\right) \end{aligned} \tag{14}$$

□

Beyond the theoretical proof of Proposition 2 described above, we also conduct experiments to further validate the approximation error. To achieve this, we calculate the average $L2$ approximation error across batch sizes of 8, 16, 32, 64, 128, and 256 over 100 batches, as defined $\mathcal{L}_{loss}$ in (15). The experimental results presented in Table III show that the approximation error in Proposition 2 is small, confirming the reliability of our approximation. Additionally, we observed that the approximation error increases with the batch size, as larger batch sizes make it more challenging to estimate the $L2$ loss of an attacked batch. However, even at a batch size of 256, the approximation error remains minimal, indicating that the approximation remains valid even for larger batch sizes.

Proposition 2 regularizes the system of equations established in Proposition 1 and (6). Despite introducing only a single scalar constraint, it effectively disambiguates the solution space, guiding the optimization toward the ground-truth solution. Based on Proposition 2, we propose the following optimization objective:

$$\mathcal{L}_{loss}(\hat{\boldsymbol{y}}') = \left\| l(\hat{\boldsymbol{y}}', \boldsymbol{y}) - \left[-\frac{1}{B}\sum_{i=1}^{B}\ln\left(\nabla \boldsymbol{b}_L^{\boldsymbol{y}(i)} + \frac{\boldsymbol{\lambda}_{\boldsymbol{y}(i)}}{B}\right)\right]\right\|^2, \tag{15}$$

After integrating $\mathcal{L}_{loss}$ into (7), we define the total loss of reconstructing model outputs of GLAD as (16).

$$\mathcal{L}_{total}(\hat{\boldsymbol{y}}') = \alpha_1\mathcal{L}_w(\hat{\boldsymbol{y}}') + \alpha_2\mathcal{L}_b(\hat{\boldsymbol{y}}') + \alpha_3\mathcal{L}_{loss}(\hat{\boldsymbol{y}}'), \tag{16}$$

Rationale Analysis: The formulations of $\mathcal{L}_w$, $\mathcal{L}_b$, and $\mathcal{L}_{loss}$ provide the theoretical rationale for how **GLAD** resolves the challenges of reconstructing high-resolution and duplicate-label data. (i) **Decoupling resolution complexity:** GLAD shifts the optimization target from the model inputs pixel space to the model outputs logits space. This reduces the number of optimization variables from $O(B \times Channel \times Height \times Width)$ to $O(B \times C)$, thereby decoupling the computational complexity from the image resolution. (ii) **Disentangling duplicate-label data:** Unlike analytics-based methods that rely on aggregated gradients (averaged: linear combinations), $\mathcal{L}_w$ imposes constraints on *every individual element* of the model outputs logits matrix. This mechanism forces the optimization to recover a unique, sample-specific logits vector for each sample, even for those sharing the same label, thus effectively disentangling duplicate-label samples.

D. Inversion From Model Outputs to Inputs

Following our method for reconstructing model outputs, we propose an MIA method to revert the obtained model outputs to their corresponding model inputs. MIAs [43] aim to reconstruct original input data or extract sensitive information from the model outputs. These attacks leverage the tendency of deep learning models to retain some information about the input data from their outputs. They typically employ optimization techniques to search for the data that can minimize the loss between its model outputs and the target outputs [38]. Additionally, generative models, such as generative adversarial networks [44], are also used to map the model outputs back to the input space [45]. MIAs have proven effective in data reconstruction and sensitive attribute inference [46].

Our MIA primarily consists of two steps: 1) the analytical derivation of feature maps; and 2) the direct generation via a generator. Specifically, recent analytics-based methods [10], [11], [16] explore the extraction of the feature maps and demonstrate that the feature maps input to the last FCL can be approximated by (17).

$$\boldsymbol{f}_i \approx \nabla \boldsymbol{W}_L^{\boldsymbol{y}(i)} / \nabla \boldsymbol{b}_L^{\boldsymbol{y}(i)}, \tag{17}$$

where $i \in [1, B]$ is the index of the sample within the input batch, $\nabla \boldsymbol{W}_L^{\boldsymbol{y}(i)}$ is the row of the gradient matrix for the FCL's weights corresponding to the index $\boldsymbol{y}(i)$, and $\nabla \boldsymbol{b}_L^{\boldsymbol{y}(i)}$ is the row of the gradient matrix for the FCL's bias corresponding to the index $\boldsymbol{y}(i)$. However, such a decomposition may not be applicable when dealing with data with the same labels, since the derived feature maps are not for a single sample, but are linear combination of multiple samples with the same labels [10], [11], [19]. For example, if the sample $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in a batch have the same label $\boldsymbol{y}(i) = \boldsymbol{y}(j)$, they would incorrectly receive the same feature map $\boldsymbol{f}_i = \boldsymbol{f}_j$, a result that is logically implausible as each sample should yield a distinct feature map. Fig. 4 visualizes the causes and results of the failure of the analysis-based methods in reconstructing data with identical labels. Based on this observation, we propose a novel analytical equation in the second line of (18) to compute feature maps using decomposed model outputs.

$$\begin{aligned} \nabla \boldsymbol{W}_L &= \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{W}_L} = \frac{\partial l(\hat{\boldsymbol{y}}, \boldsymbol{y})}{\partial \hat{\boldsymbol{y}}} \cdot \boldsymbol{f} \\ \Longrightarrow \boldsymbol{f}' &= \left(\frac{\partial l\left(\hat{\boldsymbol{y}}', \boldsymbol{y}\right)}{\partial \hat{\boldsymbol{y}}'}\right)^{-1} \cdot \nabla \boldsymbol{W}_L \end{aligned} \tag{18}$$

Since $\hat{\boldsymbol{y}}'$ can be obtained by optimizing with the objective in (7), and $\boldsymbol{y}$ and $\nabla \boldsymbol{W}_L$ is available to the attacker, we can compute the dummy feature maps $\boldsymbol{f}'$ directly through (18). Considering

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

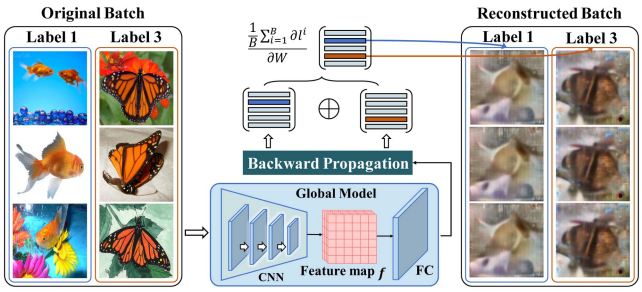

Fig. 4. The success of the GLAs lies in the features of the data hidden in the gradient. Features of data with the same label often entangle in the same row of the gradient matrix of model weights, which then represent a linear combination of these features. This causes analysis-based methods to be unable to distinguish between each sample in a batch with the same labels; they can only reconstruct linear combinations of these samples.

that matrix $\partial l(\hat{\boldsymbol{y}}', \boldsymbol{y})/\partial \hat{\boldsymbol{y}}'$ may not be invertible, we employ the "Moore-Penrose inverse" method [47], a widely recognized method for computing generalized inverses [48], to obtain its pseudo-inverse. After obtaining the feature maps, we feed the feature maps into the generator to invert the feature maps to the input space. The structure of the generator is described in prior work [16]. This generator is readily accessible to a general attacker. The generator is trained by inputting feature maps obtained by inputting data from an auxiliary dataset into the target model, and it generates these. The distribution of the auxiliary dataset need not mirror that of the actual dataset. For further discussion on how the auxiliary dataset impacts reconstruction results, see Section IV.

Notably, once model outputs are leaked via the gradient, attackers can use any established MIAs [43], [49], [50], in addition to our proposed technique, to invert these outputs back to their inputs. Consequently, our method bridges the gap between the distinct domains of GLAs and MIAs, thus enabling attackers to employ any MIAs to deduce accurate model inputs. This enhancement broadens attackers' capabilities and lowers the barrier to executing gradient leakage attacks.

IV. Experiments

In this section, we first explain the selection of SOTA baselines and describe the experimental settings. Subsequently, we compare our method against these baselines across various experimental conditions, including different proportions of duplicate labels, batch sizes, image resolutions, datasets, and defense mechanisms. Finally, we conduct a series of ablation studies to further explore the proposed method.

A. Configurations

Baselines: We choose four established prior methods as our baselines: the seminal work on gradient inversion of DLG [7], the robust attack of IG [9] with a strong performance on large batch sizes, the generative attack of GGL [14] with a strong performance for high-resolution data and the more recent work FGLA [16] with current SOTA gradient inversion attack. These methods are chosen because they are compatible with our threat model and each has its official open-source code repository, thus avoiding potential comparison errors due to inaccurate replication.

Experimental Setup: We choose three classic datasets as target datasets: ImageNet [51], CIFAR-100 [52], and CelebA [53] with the ImageNet dataset serving as auxiliary dataset. We use the pre-trained ResNet50 network provided by PyTorch as the default global model for image classification in FL. We employ the Adam optimizer [54] using a learning rate of 0.001 to optimize the dummy outputs with a total of 20000 iterations and set $\alpha_1 = 10000$, $\alpha_2 = 1$, and $\alpha_3 = 100$, which are adjusted based on experimental experience. We measure the similarity between the original and reconstructed data by three commonly-used metrics: peak signal-to-noise ratio (PSNR↑), structural similarity (SSIM↑), and perceptual image similarity score (LPIPS↓) [55]. Here, the ↑ denotes that higher values indicate better reconstruction performance, and the ↓ denotes that lower values indicate better performance. Unless stated otherwise, we take a batch size of 8, a resolution of 224×224 pixels, an attacked dataset ImageNet, and a random assignment of label distributions within the batch as the default experimental settings. All experiments are conducted on a server equipped with two NVIDIA RTX 4090 GPUs.

B. Comparison With the SOTA Methods

1) Comparison Under Different Duplicate Label Proportions: We compare GLAD with the baselines introduced above with increasing proportions of duplicate labels. Fig. 5(a) illustrates that the performance of baselines deteriorates as the duplicate label proportion increases and these methods fail to reconstruct data when the duplicate label proportion is high. The decline for FGLA can be attributed to the inability to reconstruct data with duplicate labels, as discussed in Section III. Moreover, the entanglement in the gradient rows of the data with the same labels complicates the optimization process of DLG, IG, and GGL, leading to a degradation in the quality of the reconstructed results. This phenomenon is also discussed in the prior work [18]. In contrast, GLAD demonstrates robust performance even with high duplicate label proportions. We provide a reconstruction example of GLAD and the SOTA baselines in Fig. 6. From Fig. 6 we can see that GLAD effectively reconstructs data that closely resembles the original data in terms of evaluation metrics and visual appearance. It is also clear that DLG, IG, and GGL take at least 270 times longer than GLAD, but do not produce as good reconstruction results as GLAD. Such methods optimize dummy inputs through gradient matching, which involves inputting dummy data into the model to obtain the dummy gradient and align it with the true gradient. That process is time-consuming. Interestingly, we find that feature entanglement of data with the same labels within the same rows of the gradient matrix leads to "feature misalignment" of data with the same labels in the reconstruction results. For instance, as depicted in Fig. 6, the IG incorrectly places the parrot's neck from the first image into the second one, reconstructs the tail of the parrot of the second image in the first reconstructed image, and swaps the bottom right corners of the fourth and fifth

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

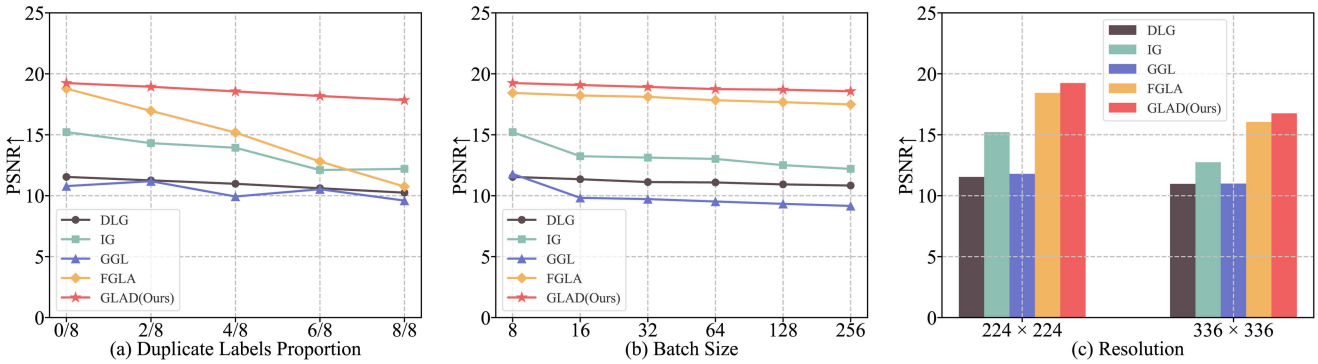

Fig. 5. Figure (a): average reconstructed image quality measured by PSNR vs. duplicate labels proportion. Figure (b): average reconstructed image quality measured by PSNR vs. batch size. Figure (c): average reconstructed image quality measured by PSNR vs. image resolution. These results are averaged results on the same 100 batches. GLAD outperforms the baseline methods in settings with different duplicate label proportions, batch sizes, and image resolutions.

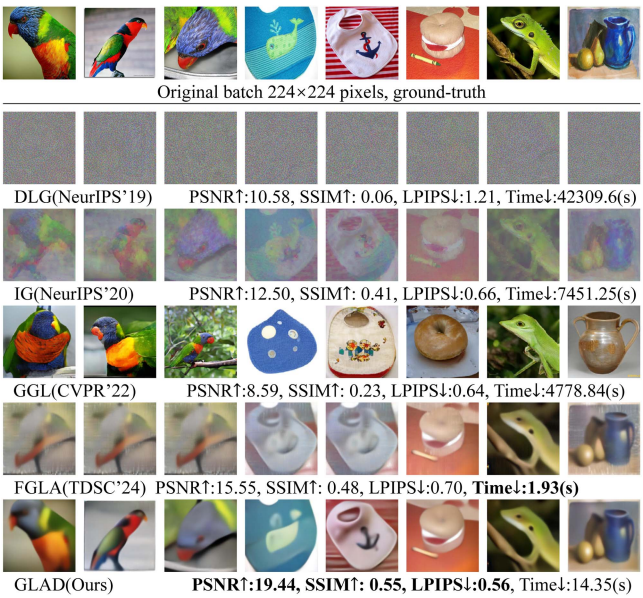

Fig. 6. A visualization of the comparison between our method and the SOTA GLA methods.

images. By contrast, the reconstructions of the sixth, seventh, and eighth images, each with distinct labels, did not exhibit this issue. Despite the employment of the pre-trained generator BigGAN [56], GGL reconstructs high-definition images but does not resemble the target batch. FGLA takes the shortest time to reconstruct the data, but cannot distinguish between data with the same label.

2) Comparison Under Different Batch Sizes: To assess the effect of batch size on GLAD and baselines, we conduct experiments with increasing batch size from 8 to 256 and measure the image reconstruction quality using PSNR. As is evident from Fig. 5(b), GLAD maintains the highest values of PNSR above 18 across all batch sizes. While the reconstruction pipeline of the FGLA seems independent of the batch size, an increase in duplicate labels within larger batches results in a decline in the PSNR of the reconstruction. The PSNR value for the IG

TABLE IV
THE RECONSTRUCTION RESULTS MEASURED BY PSNR AND EXECUTION TIME ON IMAGENET, CIFAR-100, AND CELEBA DATASET

Method	Iteration	PSNR↑			Time(s)
		ImageNet	CIFAR-100	CelebA	
DLG [7]	10000	11.53908	10.62043	11.43062	49455.25
IG [9]	20000	15.21782	14.92683	15.51359	7495.02
GGL [14]	1000	10.78704	10.47981	10.70272	2265.90
FGLA [16]	None	18.43037	17.24805	18.59564	**1.84**
GLAD (Ours)	20000	**19.24153**	**21.74278**	**19.41555**	14.23

gradually decreases from about 15 at a batch size of 8 to about 10 at a batch size of 256, and the GGL method exhibits a similar trend but with overall lower PSNR values. The DLG has the lowest PSNR value at all batch sizes, at about 10 or less. It is worth noting that GLAD performs better at a batch size of 256 than the baselines even at smaller batch sizes (e.g., 8). A visual example in Fig. 7 demonstrates that GLAD is still effective even at a batch size of 256.

3) Comparison Under Different Resolutions: Then, we evaluate the proposed method against the baselines on images with larger resolutions and quantify the reconstruction results using PSNR. Fig. 5(c) indicates that GLAD consistently achieves the highest PSNR values at both resolutions of 224×224 and 336×336, demonstrating its SOTA reconstruction capabilities. The other SOTA baselines (IG, GGL, and DLG) exhibit relatively low PSNR values, with DLG being the least effective. The decrease in the quality of reconstruction results for the original 336×336 pixels data can be attributed to the increased complexity and greater challenge posed by images of this resolution on the generator's reconstruction capability. Fig. 8 illustrates the visual reconstruction results at different resolutions, demonstrating that our method effectively maintains its ability to reconstruct visual appearance even at higher resolutions, thus showcasing its robustness against resolution.

4) Comparison Under Different Datasets: To compare the generalizability of our method and the baselines, we validate our method and the baselines on ImageNet, CIFAR-100, and CelebA (face classification task with 10177 categories). Table IV proves that GLAD consistently achieves the highest PSNR values on

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

Fig. 7. **Left:** Ground-truth batch of images with $256 \times 3 \times 224 \times 224$ varibles. **Right:** Visual reconstructed results.

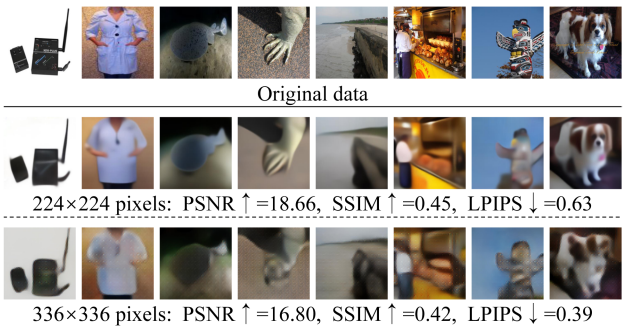

Fig. 8. Visual reconstruction results at different resolutions.

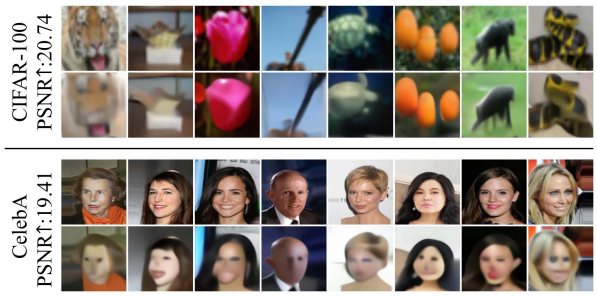

Fig. 9. The statistical metric and visualization of reconstruction results on the CIFAR-100 and CelebA datasets.

all datasets in a relatively short time. In contrast, IG, GGL, and DLG have run times hundreds of times longer than GLAD, but do not have matching reconstruction results.

We also provide a visual example of the reconstruction results on CIFAR-100 and CelebA in Fig. 9. These results clearly demonstrate the superior performance of our method across various datasets, emphasizing the robustness and generalizability of GLAD. Despite the blurry reconstructed results on the CelebA facial dataset, some attributes of the attacked images are still exposed, such as background, hair color, skin tone, and facial contours. It's essential to note that in the real-world FL system, the adversary might not always have precise knowledge about the client's dataset. Therefore, ensuring that GLA attacks remain robust when the auxiliary dataset differs from the target dataset is crucial.

Following our presentation on reconstruction results, we now discuss how the auxiliary dataset impacts the attack performance. We intuit that both the similarity between the auxiliary dataset and the target dataset, as well as the complexity of the auxiliary dataset, can influence the results of the attack. In our experiments, we use ImageNet as the auxiliary dataset and validate the proposed method on ImageNet, CelebA and CIFAR-100. The similarity between ImageNet and CIFAR-100 is higher compared to that with CelebA, and the reconstruction performance of the proposed method on CIFAR-100 outperforms that on CelebA. Therefore, it can be inferred that a higher similarity between the auxiliary dataset and the target dataset leads to improved attack performance. However, the lack of effective methodologies to quantify dataset similarity poses challenges in accurately assessing its impact on attack results. Nonetheless, attackers can fine-tune the auxiliary dataset based on the outcomes of the reconstruction to optimize the attack's efficacy. For instance, if attacking CelebA with ImageNet yields reconstructions that capture facial contours but lack detail, incorporating more facial images into the auxiliary dataset would enhance the reconstruction quality. Second, the complexity of the auxiliary dataset is higher than that of the attacked dataset is beneficial to the attack. For example, using ImageNet as an auxiliary dataset to attack CIFAR-100 allows the generator to learn more complex details, hence improving the attack effectiveness.

5) Comparison Under Different Possible Defense Mechanisms: We assess the effectiveness of our GLAD and

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

TABLE V
THE COMPARISON OF OUR GLAD AND THE SOTA BASELINES UNDER THREE RELATIVELY STRICT DEFENSE MECHANISMS

Method	Noisy gradient [19]			Gradient Compression [7]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DLG [7]	11.90780	0.09130	1.11990	11.19410	0.07190	1.23530
IG [9]	14.25990	0.44350	0.57790	12.45969	0.44249	0.80062
GGL [14]	11.26461	0.32251	0.56769	10.55477	0.27966	0.60754
FGLA [16]	18.84118	0.48547	0.55527	10.45432	0.26430	0.88357
GLAD (Ours)	**18.90560**	**0.52940**	**0.51770**	**16.28875**	**0.48409**	**0.60309**

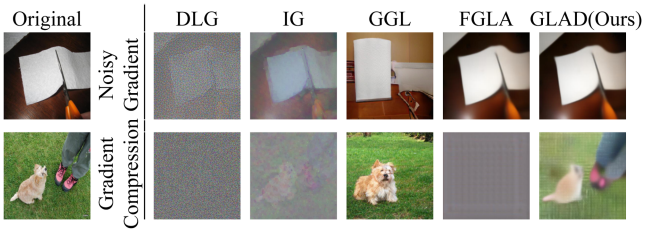

Fig. 10. Visual comparison of GLAD and SOTA baselines under the popular defense of additive noise with $\sigma = 0.01$ and gradient compression ratio with 99.9%. Our GLAD was the only method whose results resemble the original data even under gradient compression of compression ratio 99.9%.

baselines against two relatively strict defense mechanisms, similar to the ones used in the prior study [14], which include (1) Noisy gradient, where Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added to the gradients [19]; and (2) Gradient Compression, where gradients with smaller magnitudes are set to zero [7]. Specifically, we add noise with $\sigma = 0.01$ to the gradient [19] and apply gradient compression [7] at a ratio of 99.9%. The comparison in Table V shows that our method performs well even under these three strict defense mechanisms, particularly in defending against gradient compression, while gradient compression is frequently employed in FL systems to reduce bandwidth [57]. This is because both defense methods only produce changes to some parts of the gradient. For example, additive Gaussian noise adds Gaussian noise 0 to a portion of the gradient, and gradient compression retains gradients with large absolute values. GLAD leverages these unchanged gradients to accurately reconstruct the model outputs, thereby revealing the original private data.

The visual comparisons in Fig. 10 demonstrate that adding noise to gradients marginally impacts the effectiveness of both FGLA and GLAD. However, gradient compression, a technique often employed to reduce bandwidth in FL, significantly impedes FGLA's ability to recover the original data. In contrast, GLAD maintains its ability to reconstruct data closely resembling the original, even under severe gradient compression with a ratio of 99.9%. While the GGL method, utilizing the larger capacity generator BigGAN [56], produces high-quality reconstructions, these do not accurately represent the original data. Although DLG and IG reveal some information about the original data, they face difficulties in achieving accurate visual reconstructions.

TABLE VI
THE L_2 ERROR BETWEEN THE OPTIMIZED DUMMY MODEL OUTPUTS AND THE ACTUAL MODEL OUTPUTS AT DIFFERENT BATCH SIZES

Batch size	8	16	32	64	128	256
$\|\hat{\boldsymbol{y}}' - \hat{\boldsymbol{y}}\|^2$	0.00008	0.00020	0.00070	0.00130	0.00320	0.00360

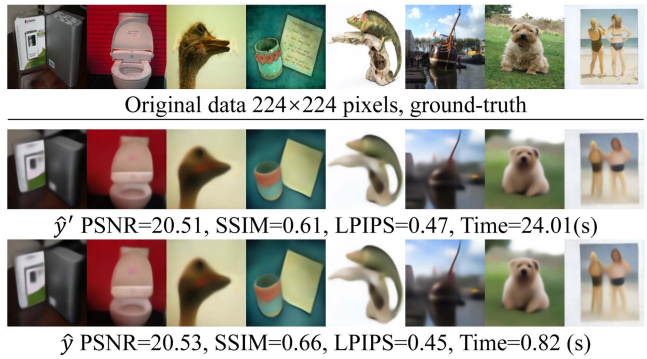

Fig. 11. The comparison between the reconstruction by the optimized model outputs and those by the actual model outputs.

TABLE VII
COMPARISON OF AVERAGE OUTCOMES ACROSS 100 BATCHES FOR DIFFERENT OPTIMIZATION OBJECTIVES COMBINATIONS

Optimization objectives	PSNR↑	SSIM↑	LPIPS↓
$\mathcal{L}_w$	14.55934	0.38931	0.71797
$\mathcal{L}_w$+$\mathcal{L}_{loss}$	19.16308	0.50066	0.57047
$\mathcal{L}_w$+$\mathcal{L}_b$	14.55934	0.38931	0.71797
$\mathcal{L}_w$+$\mathcal{L}_b$+$\mathcal{L}_{loss}$	**19.24153**	**0.50948**	**0.55870**

C. Ablation Study

1) Evaluation of the Ability to Leak the Model Outputs: As discussed and analyzed in Section III, inverting the training loss function for model outputs disaggregation is the most important part of our method. To assess the difference between the optimized model outputs and the actual model outputs, we conduct experiments and compute the L_2 distance between the dummy model outputs and the actual model outputs at batch sizes of 8, 16, 32, 64, 128, and 256. The experimental results in Table VI show that our method can accurately disaggregate the model outputs. Visual comparisons in Fig. 11 illustrate that images reconstructed from dummy model outputs closely resemble those from actual model outputs. This further demonstrates that our method can accurately capture the model outputs and accurately reconstruct the private data.

2) Dissecting the Contributions of Optimization Objectives in Model Outputs Recovery: Next, we conduct an ablation study to dissect the individual contributions of the three optimization objectives. Given $\mathcal{L}_w$ provides the maximum number of equations, we regard $\mathcal{L}_w$ as the primary component of our optimization framework. To assess the feasibility of relying solely on $\mathcal{L}_w$, and discern the auxiliary roles of $\mathcal{L}_b$ and $\mathcal{L}_{loss}$ for the optimization, we execute experiments with different optimization objective combinations as shown in Table VII, while keeping other settings consistent. A comparison of the first

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

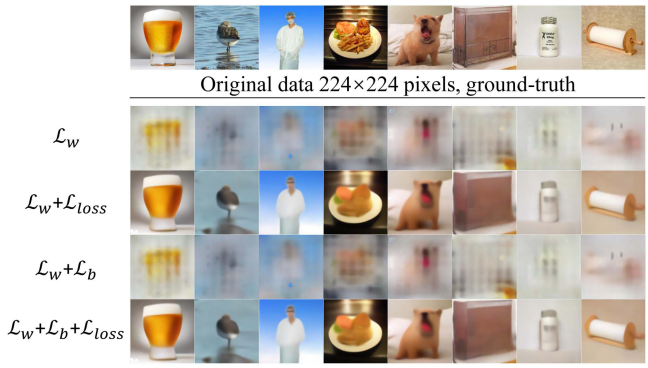

Fig. 12. A visual reconstruction example of different optimization item combinations.

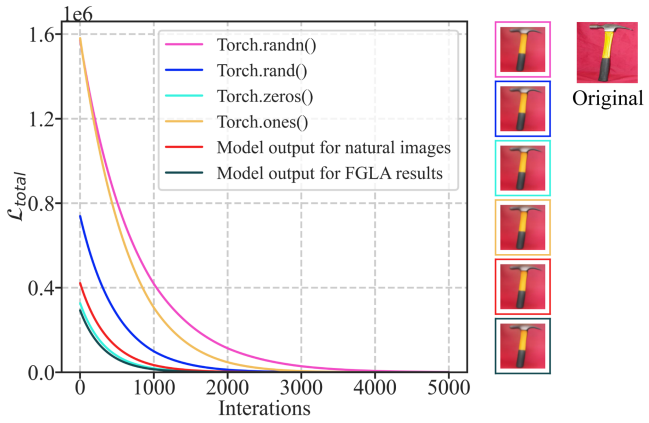

Fig. 13. The effect of different initialization methods on the optimization convergence speed and one reconstruction example with these initialization methods.

and fourth rows in Table VII shows that $\mathcal{L}_w$ alone is insufficient for model output recovery. Furthermore, a comparison between the first and third rows, and the second and fourth rows, suggests that $\mathcal{L}_b$ seemingly plays a relatively insignificant role. This is because, in the ImageNet dataset with $C = 1000$, the number of equations provided by $\mathcal{L}_w$ $(C \times C)$ vastly overshadows those from $\mathcal{L}_b$ (C). However, $\mathcal{L}_b$ manifests its utility when the magnitude of C decreases. Specifically, when the batch size increases to $C \times C + 1 < B \times C < C \times (C + 1) + 1$, $\mathcal{L}_b$ plays a decisive role in ensuring reconstruction success. Finally, Table VII demonstrates that $\mathcal{L}_{loss}$ regularizes $\mathcal{L}_w$ and $\mathcal{L}_b$ and distinguishes between right data and twin data, thereby steering the optimization towards the right solution. Empirically, we observed that $\mathcal{L}_{loss}$ achieved a nearly perfect success rate in distinguishing the right solution from twin solutions, with no failure cases observed. Visual comparisons of reconstruction results under different objective combinations are provided in Fig. 12.

3) Evaluation With the Different Initialization Methods for the Dummy Model Outputs: Existing optimization-based methods often fail due to inappropriate dummy data initialization [10], [34]. So, a natural question arises: is our method also sensitive to the initialization of the dummy model outputs? To explore this, we employed six different initialization methods for the dummy model outputs: uniform distribution within the range [0, 1] implemented using the torch.rand(), standard normal distribution implemented using torch.randn(), zero matrix initialization implemented using torch.zeros(), one matrix initialization implemented using torch.ones(), as well as initialization with model outputs for natural images and initialization with model outputs for the reconstruction results of FGLA [16]. Fig. 13 shows how the $\mathcal{L}_{total}$ varies with the number of optimization iterations and one visual example for each initialization method. It is clear from the right of Fig. 13 that despite using different initialization methods, all of them eventually converge to the same minimum value, therefore our method is not sensitive to initialization. Moreover, using the zero matrix initialization or using the result of FGLA reconstruction as the initialization allows the optimization process to start closer to the final minimum point. This means that with these initialization methods, fewer iterations and time for optimization are required. Regardless of the initialization method used, Our method consistently produces good reconstructions, showcasing its robustness to different initialization methods.

V. Discussion

A. Limitations and Future Work

The proposed method does not work when the batch size B is larger than $C + 1$ since there are more variables in model outputs $(B \times C)$ than the number of equations $(C \times (C + 1) + 1)$ provided by optimization objectives. This limitation on batch size is also a common limitation for most existing GLAs and necessitates further investigation. Moreover, the proposed method depends on the size of the feature maps. It is hard to execute the GLAD if the feature maps are small. Despite the potential for the adversary to modify the model to increase the size of feature maps, we continue to encourage further research in this direction.

B. Extensions

Our method disaggregates the model outputs through optimization. Once these outputs are obtained, our method can employ any MIA in a white-box scenario to further improve performance. Moreover, the obtained model outputs can be further used to decompose the feature maps and the decomposed feature maps can also be used to infer the attribute of private data. Finally, our work can also be extended to language models, which often end up with the FCL.

C. Possible Defense

Many modern defense techniques employ gradient clipping and compression to defend against gradient leakage [58], [59]. Following this direction, we evaluate a defense mechanism that combines gradient clipping [60] and noise addition [7]. This approach implements the principles of differential privacy (DP) to effectively resist GLAD. We conduct experiments

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

TABLE VIII
THE DP PARAMETER ϵ VALUE, THE RECONSTRUCTION PERFORMANCE OF GLAD MEASURED BY PSNR, AND THE ACCURACY (ACC) OF THE GLOBAL MODEL UNDER VARYING MAGNITUDES OF GRADIENT CLIPPING AND NOISE

	$\mathcal{C}=16$			$\mathcal{C}=8$			$\mathcal{C}=4$			$\mathcal{C}=2$		
σ^2	ϵ	PSNR ↑	Acc ↑	ϵ	PSNR ↑	Acc ↑	ϵ	PSNR ↑	Acc ↑	ϵ	PSNR ↑	Acc ↑
0.0001	3596.07155	19.10380	0.99888	1798.03577	18.84338	0.99476	899.01788	18.59498	0.98068	449.50894	17.91110	0.95884
0.001	1137.17767	19.30163	0.99886	568.58883	18.73510	0.99494	284.29441	18.24891	0.98118	142.14720	15.43059	0.96150
0.01	359.60715	15.67200	0.99894	179.80357	12.09762	0.99412	89.90178	9.31854	0.97976	44.95089	7.31292	0.95874
0.1	113.71776	7.05577	0.93550	56.85888	6.30624	0.89860	28.42944	5.95383	0.83278	14.21472	5.83537	0.74146

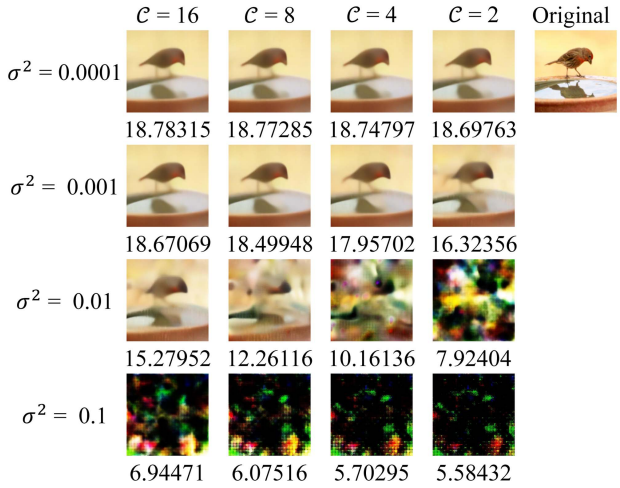

Fig. 14. A visual reconstruction example under DP defense with varying magnitudes of gradient clipping and noise. The reconstruction results are measured by PSNR.

with a pre-trained ResNet50 from PyTorch as global model, CIFAR-10 as the client training. Clients first clip the gradient with a shield $\mathcal{C}$, and then add Gaussian noise with mean 0 and variance σ^2 prior to transmission. The DP parameter epsilon ϵ can be calculated as follows: $\epsilon = \frac{\mathcal{C}}{\sigma}\sqrt{2\ln(1.25/\delta)}$, where δ is the minimum probability of privacy breach acceptable in DP and a smaller value of ϵ means stronger privacy protection. In our experiments, we set δ to 0.1 and vary the value of $\mathcal{C}$ from 16 to 2 and the variance σ^2 of Gaussian noise from 0.0001 to 0.1. The corresponding epsilon values can be referenced in Table VIII. Table VIII details the corresponding ϵ values and reconstruction PSNR. The reconstruction results demonstrate that this DP defense effectively counters our attack with minimal degradation of global model accuracy. The optimal balance, preventing GLAD and accuracy degradation, is achieved with $\mathcal{C}=4$ and $\sigma^2=0.01$. Fig. 14 provides a visual reconstruction under varying magnitudes of gradient clipping and noise.

VI. CONCLUSION

In this work, we propose GLAD, an advanced gradient leakage attack against duplicate labels. We analyze GLAD theoretically and empirically verify its enhanced capability of gradient inversion on high-resolution and duplicate-label data. Compared to existing works, our method achieves the SOTA reconstruction performance at varying duplicate label proportions, batch sizes, resolutions, datasets, and popular defense mechanisms, while running at a fast speed.

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 *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[2] S. Truex et al., "A hybrid approach to privacy-preserving federated learning," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 1–11.
[3] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Informat. Res.*, vol. 5, pp. 1–19, 2021.
[4] D. Byrd and A. Polychroniadou, "Differentially private secure multi-party computation for federated learning in financial applications," in *Proc. First ACM Int. Conf. AI Finance*, 2020, pp. 1–9.
[5] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for Internet of Things: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 23, no. 3, pp. 1622–1658, Third Quarter, 2021.
[6] M. Liu, S. Ho, M. Wang, L. Gao, Y. Jin, and H. Zhang, "Federated learning meets natural language processing: A survey," 2021, *arXiv:2107.12603*.
[7] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 17–31, doi: 10.1007/978-3-030-63076-8_2.
[8] J. Jeon, J. Kim, K. Lee, S. Oh, and J. Ok, "Gradient inversion with generative image prior," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 29898–29908.
[9] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?," 2020, *arXiv:2003.14053*.
[10] J. Zhu and M. Blaschko, "R-GAP: Recursive gradient attack on privacy," 2020, *arXiv:2010.07733*.
[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," 2021, *arXiv:2110.13057*.
[12] H. Yang, M. Ge, D. Xue, K. Xiang, H. Li, and R. Lu, "Gradient leakage attacks in federated learning: Research frontiers, taxonomy and future directions," *IEEE Netw.*, vol. 38, no. 2, pp. 247–254, Mar. 2024.
[13] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," 2020, *arXiv:2001.0261*.
[14] 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.*, 2022, pp. 10132–10142.
[15] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning: Revisited and enhanced," in *Proc. 8th Int. Conf. Appl. Techn. Inf. Secur.*, Auckland, New Zealand, 2017, pp. 100–110.
[16] H. Yang et al., "Fast generation-based gradient leakage attacks: An approach to generate training data directly from the gradient," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 1, pp. 132–145, Jan./Feb. 2025.
[17] L. Fan et al., "Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks," 2020, *arXiv:2006.11601*.
[18] J. Qian and L. K. Hansen, "What can we learn from gradients?," 2020. [Online]. Available: https://openreview.net/forum?id=gQn5xeVtz0I
[19] J. Sun, A. Li, B. Wang, H. Yang, H. Li, and Y. Chen, "Soteria: Provable defense against privacy leakage in federated learning from representation perspective," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9307–9315.
[20] Z. Deng et al., "Leakage-resilient and carbon-neutral aggregation featuring the federated AI-enabled critical infrastructure," *IEEE Trans. Dependable Secure Comput.*, vol. 22, no. 4, pp. 3661–3675, Jul./Aug. 2025.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.

[21] N. Wu, X. Yuan, S. Wang, H. Hu, and M. Xue, "Cardinality counting in" alcatraz": A privacy-aware federated learning approach," in *Proc. ACM Web Conf.*, 2024, pp. 3076–3084.
[22] Y. Zhang et al., "Privacy-preserving and fairness-aware federated learning for critical infrastructure protection and resilience," in *Proc. ACM Web Conf.*, 2024, pp. 2986–2997.
[23] J. Konečn, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.
[24] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023.
[25] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.
[26] V. Carletti, P. Foggia, C. Mazzocca, G. Parrella, and M. Vento, "SoK: Gradient inversion attacks in federated learning," in *Proc. 34th USENIX Conf. Secur. Symp.*, 2025, pp. 6439–6459.
[27] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Math. Programm.*, vol. 45, no. 1, pp. 503–528, 1989.
[28] 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.*, 2021, pp. 16337–16346.
[29] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2015, pp. 5188–5196.
[30] M. Fan, F. Wang, C. Chen, and J. Zhou, "Boosting gradient leakage attacks: Data reconstruction in realistic FL settings," in *Proc. 34th USENIX Conf. Secur. Symp.*, 2025, pp. 2985–3004.
[31] 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.*, 2024, pp. 1149–1157.
[32] L. Wu, Z. Liu, B. Pu, K. Wei, H. Cao, and S. Yao, "DGGI: Deep generative gradient inversion with diffusion model," *Inf. Fusion*, vol. 113, 2025, Art. no. 102620.
[33] A. Hatamizadeh et al., "Gradvit: Gradient inversion of vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10021–10030.
[34] H. Yang, M. Ge, K. Xiang, and J. Li, "Using highly compressed gradients in federated learning for data reconstruction attacks," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 818–830, 2022.
[35] 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.*, 2023, pp. 6381–6398.
[36] R. Wu, X. Chen, C. Guo, and K. Q. Weinberger, "Learning to invert: Simple adaptive attacks for gradient inversion in federated learning," in *Proc. Int. conf. Uncertainty Artif. Intell.*, 2023, pp. 2293–2303.
[37] H. Liang, Y. Li, C. Zhang, X. Liu, and L. Zhu, "EGIA: An external gradient inversion attack in federated learning," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 4984–4995, 2023.
[38] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 603–618.
[39] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," 2021, *arXiv:2112.02918*.
[40] S. Kariyappa et al., "Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 15884–15899.
[41] J. C. Zhao, A. Dabholkar, A. Sharma, and S. Bagchi, "Leak and learn: An attacker's cookbook to train using leaked data from federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 12247–12256.
[42] 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. Representations*, 2022, pp. 1–15.
[43] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1322–1333.
[44] I. J. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, pp. 139–144, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:1033682
[45] Z. Yang, J. Zhang, E.-C. Chang, and Z. Liang, "Neural network inversion in adversarial setting via background knowledge alignment," in *Proc. 2019 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 225–240. [Online]. Available: https://api.semanticscholar.org/CorpusID
[46] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 601–618.
[47] R. Penrose, "A generalized inverse for matrices," in *Mathematical Proceedings of the Cambridge Philosophical Society*. Cambridge, U.K.: Cambridge Univ. Press, 1955, pp. 406–413.
[48] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*. Philadelphia, PA, USA: SIAM, 2009.
[49] K.-C. J. Wang, Y. Fu, K. Li, A. Khisti, R. S. Zemel, and A. Makhzani, "Variational model inversion attacks," 2022, *arXiv:2201.10787*.
[50] N.-B. Nguyen, K. Chandrasegaran, M. Abdollahzadeh, and N.-M. Cheung, "Re-thinking model inversion attacks against deep neural networks," 2023, *arXiv:2304.01669*.
[51] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Miami, FL, USA, 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.
[52] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," 2009.
[53] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2014, pp. 3730–3738.
[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[55] 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.*, 2018, pp. 586–595.
[56] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," 2018, *arXiv:1809.11096*.
[57] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv:1712.01887*.
[58] J. Wang, S. Guo, X. Xie, and H. Qi, "Protect privacy from gradient leakage attack in federated learning," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 580–589.
[59] W. Wei, L. Liu, Y. Wu, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst.*, 2021, pp. 797–807.
[60] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," 2017, *arXiv:1712.07557*.

Authorized licensed use limited to: Nanyang Technological University Library. Downloaded on July 13,2026 at 03:26:55 UTC from IEEE Xplore. Restrictions apply.