UNVEILING THE SHADOWS: INVESTIGATING PRIVACY RISKS OF GRADIENT LEAKAGE IN FEDERATED LEARNING

ZHANG RUI

PhD

The Hong Kong Polytechnic University

2025

The Hong Kong Polytechnic University
Department of Computing

Unveiling the Shadows: Investigating Privacy Risks of Gradient Leakage in Federated Learning

ZHANG Rui

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
August 2024

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: ZHANG Rui

Abstract

Driven by growing concerns about data privacy and the need for powerful machine learning models trained on diverse datasets, Federated Learning (FL) has emerged as a promising solution. This decentralized approach enables the training of global models without compromising individual user privacy. In FL, participating clients collaboratively train a shared model by uploading gradients calculated on their local private data. Hence, the raw data remains confined to the clients' devices. However, this seemingly secure systems are not impervious to attack. Adversaries can exploit the shared gradients through Gradient Inversion Attacks (GIAs) to disclose sensitive information about the training data, thereby threatening the privacy of FL clients.

Traditional GIAs employ an optimization-based approach to recover private training data used in FL. The process begins by creating synthetic data points generated by sampling from a Gaussian distribution for both input features and the corresponding labels. The attackers feed this synthetic data into the FL shared model and calculates the gradients of the model's output with respect to these artificial inputs. Through iterative optimization, the synthetic data is continuously refined. The distance between the gradients produced by the synthetic data and the actual gradients obtained from FL clients is a measure of accuracy. This distance guides adjustments to the synthetic data, aiming to minimize the difference. When the synthetic and the actual gradients become nearly indistinguishable, the reconstructed synthetic data approximates the private training data used in FL collaboration.

Building upon existing research on GIAs, this thesis delves into the limitations of current methods from an adversarial perspective. We identify three critical challenges hindering the development of effective and efficient data reconstruction mechanisms. (1) **Label Information Leakage**: Access to accurate labels is essential for reconstructing training samples. Existing methods either assume adversaries possess this knowledge or rely on some restrictions (e.g., non-negative activations) to circumvent this issue. (2) **Layer-wise Gradient Exploitation**: While some studies utilize gradients from fully connected layers for direct input recovery, the broader potential of exploiting layer-wise gradient relationships remains largely unexplored. Deeper exploration in this area is crucial for advancing attack effectiveness. (3) **Targeted Data Reconstruction**: Most GIAs focus on reconstructing comprehensive data from deep networks and large batches. However, the threat escalates when an adversary can selectively reconstruct specific data samples from gradients. This precision allows for more accurate privacy leakage and increases potential misuse.

In this thesis, we aim to advance the field of FL security and privacy by developing novel gradient-induced attacks that address the aforementioned challenges. Our main contributions are threefold, detailed in the following three parts.

In the first part, we establish the fundamental connection between training labels and gradients. Building upon this discovery, we introduce a versatile framework for label recovery applicable to diverse classification tasks. Our framework leverages the posterior probabilities – generated by functions like Sigmoid or Softmax – to infer labels in binary, multi-class, and even imbalanced scenarios. In particular, exploiting the globally shared model in FL, an adversary can estimate these posterior probabilities for training samples by using some auxiliary data. By incorporating these estimates into the relationship derived between labels and gradients, we can effectively recover the batch training labels from shard gradients.

In the second part, we propose the Gradient Bridge (GDBR) attack to expose privacy leakage through correlated layer-wise gradients in specific gradient-sharing scenarios.

GDBR begins by deriving theoretical relationships between gradients across different layer types: input-output and weight-output for fully connected and convolutional layers, respectively, and output with respect to activation functions. By meticulously tracking gradient flow across these layers, we formulate a recursive procedure that reconstructs the gradient of the model's output logits. By associating the reconstructed logit gradients with observable variables (e.g., hidden features, Softmax probabilities), GDBR can recover label information from training samples even when only a single layer's gradients at the bottom of the model are shared.

In the third part, we focus on reconstructing training data from sensitive or specified classes by devising a targeted attack called Gradient Filtering (GradFilt). GradFilt operates under the assumption of a malicious adversary capable of manipulating both the parameters and structure of the white-box FL model. This attack commences by strategically modifying the weights and biases of the final fully connected layer, effectively zeroing out gradients corresponding to non-target data while preserving those associated with the target class. This grants GradFilt control over the model's output probabilities, enabling it to recover labels for the entire batch and determine the number of instances in the target category. Finally, GradFilt reconstructs the target data by applying an optimization-based or analytical approach, depending on the number of target samples included in the training batch.

In summary, this thesis illuminates the inherent vulnerabilities within the mechanisms of gradient sharing, underscoring the critical need for ongoing vigilance in protecting client privacy within FL systems. We firmly believe that our findings will significantly contribute to the development of more robust and trustworthy FL solutions, ultimately unlocking the full potential of collaborative machine learning across a diverse range of applications.

Publications Arising from the Thesis

- **Rui Zhang**, Ka-Ho Chow, and Ping Li. Building Gradient Bridges: Label Leakage from Restricted Gradient Sharing in Federated Learning. Prepared to *the 30th European Symposium on Research in Computer Security (ESORICS)*, 2025.

- **Rui Zhang**, Song Guo, and Ping Li. GradFilt: Class-wise Targeted Data Reconstruction from Gradients in Federated Learning. In *Companion Proceedings of the 33rd ACM on Web Conference (WWW)*, 2024.

- **Rui Zhang**, Song Guo, and Ping Li. Posterior Probability-based Label Recovery Attack in Federated Learning. In *Workshop on Privacy Regulation and Protection in Machine Learning in Conjunction with ICLR 2024 (PML-ICLR-24)*, 2024.

- **Rui Zhang**, and Song Guo. Privacy Inference for Data Auditing. In *Proceedings of the International Symposium on AI, Data and Digitalization (SAIDD)*, 2023.

- **Rui Zhang**, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. A Survey on Gradient Inversion: Attacks, Defenses and Future Directions. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*,

2022.

- Xuan Liu, Siqi Cai, Lin Li, **Rui Zhang**, and Song Guo. MGIA: Mutual Gradient Inversion Attack in MultiModal Federated Learning (Student Abstract). In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

- Zicong Hong, Song Guo, **Rui Zhang**, Peng Li, Yunfeng Zhan, and Wuhui Chen. Cycle: Sustainable Off-Chain Payment Channel Network with Asynchronous Rebalancing. In *Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022.

Acknowledgments

Completing the thesis has been a challenging yet rewarding journey, made possible by the unwavering support and encouragement of many. I extend my heartfelt gratitude to all who have played a role in my Ph.D. pursuit.

First and foremost, I am deeply grateful to my chief supervisor, Prof. Ping Li, for his invaluable guidance, encouragement, and support throughout my Ph.D. study at The Hong Kong Polytechnic University. He has fostered my intellectual independence by granting me significant freedom in exploring research directions. When facing challenges, his generous support and insightful advice have inspired me to move forward. I am grateful for his mentorship and the opportunities he has provided for my growth as an independent researcher.

Then, I extend my sincere gratitude to my co-supervisor, Prof. Song Guo, for introducing me to the fascinating field of edge computing. His expertise in these areas has been instrumental in shaping my research interests. I am particularly grateful for his perceptive suggestions and constructive feedback, which consistently challenged me to think critically and creatively, pushing the boundaries of my research. Moreover, Prof. Guo has imparted to me the importance of collaboration and interdisciplinary thinking in advancing scientific inquiry.

Next, I would like to express my deepest gratitude to all research collaborators and colleagues, who enriched my Ph.D. study with their invaluable insights, constructive feedback, and powerful support. I cherish the opportunities we shared collaborating

on diverse projects. In particular, thanks to Prof. Wenchao Xu, Dr. Junxiao Wang, Dr. Xin Xie, Dr. Haozhao Wang, Dr. Jie Zhang, Mr. Leijie Wu, and Mr. Zicong Hong from the Hong Kong Polytechnic University; Dr. Ka-Ho Chow from The University of Hong Kong; and Prof. Dacheng Tao from The University of Sydney.

My doctoral journey would not have been as rewarding without my fantastic group mates and friends. I would like to thank Wenchao Xu, Jingcai Guo, Xin Xie, Junxiao Wang, Haozhao Wang, Jie Zhang, Kang Wei, Jinwen Liang, Qihua Zhou, Leijie Wu, Zicong Hong, Yang Deng, Yi Liu, Ziming Liu, Tao Guo, Xueyang Tang, Jiewei Zhang, Feijie Wu, Enyuan Zhou, Peiran Dong, Junbao Pan, Bingjie Wang, Ruibin Li, Fushuo Huo, Yunfeng Fan, Jinyu Chen, Chuan he, Yingchun Wang, Yue Zeng, Cheng Wang, Quan Chen, Yan Hong, and Jing Li at The Hong Kong Polytechnic University; and Dimitar I. Dimitrov at ETH Zürich. Their support and encouragement are precious, and I treasure their friendship and the memories we have created together.

Last but not least, I extend my heartfelt thanks to my parents and my wife, Ms. Jiwei Ma, for their unconditional love, unwavering support, and constant encouragement. Their steadfast belief in me has been an endless source of strength and motivation throughout this journey. I am profoundly grateful for their sacrifices and unshakeable faith in my abilities. This accomplishment would not have been possible without their love and support, which have been the bedrock of my academic pursuit.

To my beloved family,

for their love, support, and encouragement.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Motivation

In today's data-driven world, the demand for powerful machine learning models has surged across diverse fields, from healthcare and finance to scientific research. These sophisticated models require massive datasets for optimal accuracy and robustness, often necessitating the aggregation of information from multiple sources. However, this extensive data collection raises critical privacy concerns, particularly when handling sensitive information residing on edge devices or dispersed across numerous data silos. Growing regulatory constraints and the risk of data breaches have made users and organizations increasingly hesitant to share raw data [73].

Fueled by mounting concerns over data privacy and the persistent demand for high-performing machine learning models trained on diverse datasets, *Federated Learning* (FL) [44, 6] has emerged as a promising solution. FL empowers collaborative model training without requiring data centralization. Participants can retain their raw data locally while contributing to a shared global model by exchanging only model updates rather than the data itself. This innovative approach strikes a balance between harnessing collective knowledge and safeguarding individual data confidentiality.

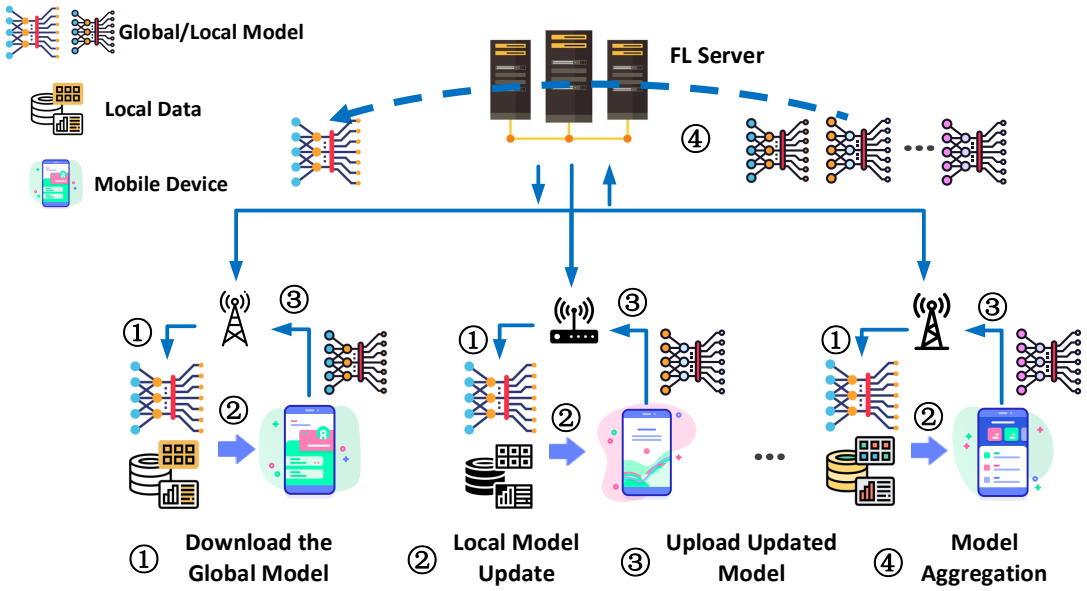


Figure 1.1: Illustration of representative Federated Learning (FL) workflows. ① Each participating client downloads the global model from the server. ② The clients train the model with their local data and compute gradients. ③ Gradients are uploaded to the server for aggregation. ④ The server updates the global model and redistributes it to clients. The collaboration process iterates until the global model converges.

In each communication round of FL, clients download the current global model from the server, train the model on their local datasets, and calculate the gradients of the model parameters. These gradients are then transmitted to the server, which aggregates them from the clients. Based on this aggregated information, the server updates the global model and redistributes it back to the clients for the next round. This iterative process continues until the model achieves a satisfactory level of accuracy. The representative workflow of an FL system is illustrated in Fig. 1.1.

However, this ostensibly secure system is not impervious to malicious exploitation. Sophisticated adversaries can leverage the shared gradients through techniques known as *Gradient Inversion Attacks* (GIAs) to recover or infer sensitive information about

the client's private training data [88, 18, 78] or target labels [84, 64, 42].

Traditional GIAs [88, 18] employ an optimization-based approach to recover training data used in FL. The process begins by creating synthetic data points, generated by sampling from a Gaussian distribution for both input features and the corresponding labels. The adversary feeds this synthetic data into the FL model and calculates the gradients of the model's output with respect to these artificial inputs. Through iterative optimization, the synthetic data is continuously refined. The distance between the gradients produced by the synthetic data and the actual gradients obtained from the FL clients serves as a measure of accuracy. This distance guides adjustments to the synthetic data, aiming to minimize the difference. When the synthetic and actual gradients become nearly indistinguishable, the reconstructed data provides a close approximation of the private training data used in FL collaboration.

By comprehensively surveying and summarizing the existing literature on GIAs and their corresponding defenses, we aim to pinpoint the limitations of current methods and explore advanced attack strategies in FL from an adversarial perspective. This deep understanding of attack vectors is crucial for developing robust and trustworthy FL systems. Based on our analysis, we identify three critical challenges that hinder the development of effective and efficient data reconstruction mechanisms.

1. **Label Information Leakage**: Access to accurate labels is essential for reconstructing training samples in GIAs. This crucial information allows adversaries to more precisely infer the characteristics of the original data. Existing methods addressing this challenge generally fall into two categories. The first assumes that adversaries already possess the knowledge of labels [18, 78, 21], which may be unrealistic in many real-world scenarios. The second approach attempts to tackle this issue by relying on model structure restrictions [78, 64], such as utilizing non-negative activation functions (ReLU or Sigmoid) or assuming unique labels in each class [84, 78, 9]. However, these constraints either sacrifice model

flexibility or prove infeasible in practical FL scenarios.

2. **Layer-wise Gradient Exploitation**: While several studies have successfully leveraged gradients from fully connected layers for direct input recovery [55], or recursively reconstructed training samples by solving linear equations from a convolutional neural network (CNN) [87, 7], the broader potential of exploiting layer-wise gradient relationships remains largely unexplored. This undeveloped area presents a significant opportunity for enhancing attack effectiveness and understanding the vulnerabilities of FL. Deeper exploration in this domain could reveal intricate connections between gradients across different layers, potentially leading to more sophisticated and powerful attack strategies. Moreover, investigating the correlations between convolutional, fully connected, and activation layers could provide valuable insights into data reconstruction.

3. **Targeted Data Reconstruction**: Traditional optimization-based GIAs such as [18, 78, 21] focus on reconstructing complex training data from deeper neural networks and larger batch sizes. However, few studies have explored the potential of selectively reconstructing specific data samples from gradients. If the above methods are adopted to extract target data, the entire batch must be reconstructed and then filtered to obtain the desired samples. This process is inefficient and computationally expensive, and only 28% of training data can be recovered from a ResNet50 model trained on a batch size of 48 [17]. Therefore, developing targeted data reconstruction mechanisms can not only enhance the efficiency of GIAs but also enable adversaries to extract high-value, sensitive information with greater accuracy. This selective reconstruction capability poses a more severe threat to individual privacy and data security in FL.

After our summary and analysis, we find that the above challenges or problems are necessary for understanding the vulnerabilities in FL and designing more secure and trustworthy FL systems. Therefore, in Section 1.2, we will introduce the main contri-

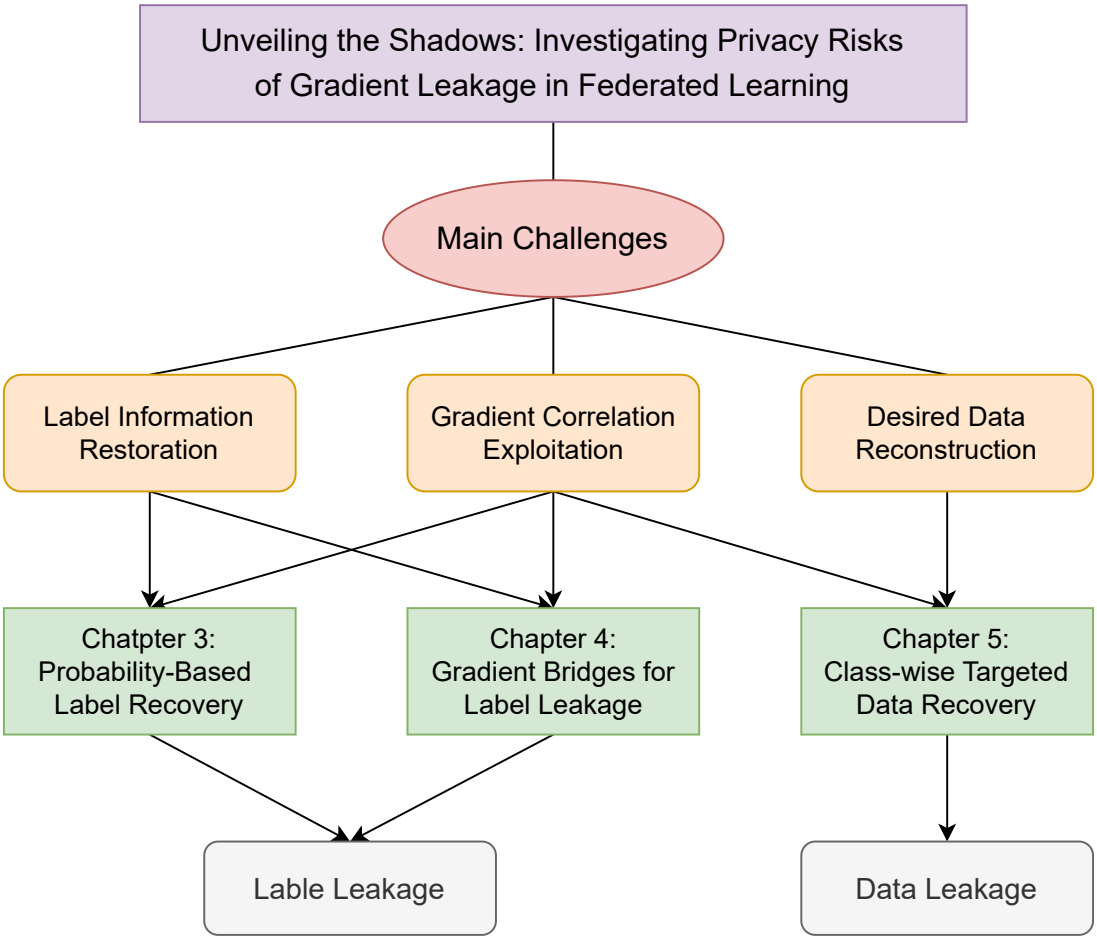


Figure 1.2: The roadmap of this thesis.

butions of this thesis, which address these key challenges. The roadmap of this thesis is shown in Fig. 1.2, outlining the structure and contributions of the thesis.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

1. We establish a fundamental connection between training labels and gradients. Building upon this discovery, we introduce a versatile framework for label recovery applicable to diverse classification tasks. Our framework leverages the

posterior probabilities that generated by functions like Sigmoid or Softmax, to assist label inference in binary, multi-class, and even imbalanced scenarios. In particular, exploiting the globally shared model in FL, an adversary can estimate these posterior probabilities for training samples by using some auxiliary dataset. By incorporating these estimates into the relationship derived between labels and gradients, we can effectively recover training labels in a mini-batch from shard gradients in FL collaboration.

2. We propose Gradient Bridge (GDBR), a novel attack that exposes privacy leakage by exploiting correlations between layer-wise gradients in specific gradient-sharing scenarios. GDBR leverages theoretical relationships derived between gradients across different layer types – input-output and weight-output for fully connected and convolutional layers, respectively, and output with respect to activation functions. By meticulously tracking gradient flow through these layers, we develop a recursive procedure that reconstructs the gradient of the model's output logits. Associating these reconstructed logit gradients with observable variables like hidden features or Softmax probabilities allows GDBR to recover label information from training samples even when only a single layer's gradients at the bottom of the model are shared.

3. We introduce Gradient Filtering (GradFilt), a targeted attack designed to reconstruct training data from sensitive or specified classes. GradFilt presumes a malicious adversary with the ability to manipulate both the parameters and structure of a white-box FL model. The attack proceeds by strategically modifying the weights and biases of the final fully connected layer, effectively suppressing gradients associated with non-target data while preserving those linked to the target class. This manipulation grants GradFilt control over the model's output probabilities, enabling it to recover labels for the entire batch and determine the number of instances belonging to the target class. Finally, GradFilt restores the target samples using either an optimization-based or analytical ap-

proach depending on the quantity of target data present.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

- **Chapter 2 (Background and Literature Review)**: In this chapter, we provide a comprehensive review of the background and related works on GIAs and their countermeasures in FL. We also discuss the limitations of existing methods and identify the key challenges that motivate our research.
- **Chapter 3 (Posterior Probability-Based Label Recovery Attack from Gradients in Federated Learning)**: This chapter presents a novel method that utilizes the posterior probabilities to infer the labels of the training data. We demonstrate the effectiveness of this approach in various classification tasks.
- **Chapter 4 (Building Gradient Bridges: Label Leakage from Restricted Gradient Sharing in Federated Learning)**: In this chapter, we propose Gradient Bridge (GDBR) to build bridges between the layer-wise gradients and infer the gradients with respect to the output logits. Taking the recovery of label distribution as an example, we demonstrate the effectiveness of GDBR by leveraging the correlations between gradients across different layers.
- **Chapter 5 (GradFilt: Class-wise Targeted Data Reconstruction from Gradients in Federated Learning)**: In this chapter, we propose GradFilt to recover targeted training samples by manipulating the parameter of the global model, and then reconstruct the target data from the filtered gradients.
- **Chapter 6 (Conclusion and Future Work)**: This chapter concludes the main contributions and limitations of the thesis and outline potential directions that can be explored in future research.

Chapter 2

Background and Literature Review

2.1 Background of Gradient Leakage

Federated Learning (FL) [44, 6] has become a popular paradigm for achieving collaborative training and data privacy at the same time. In a centralized training process, the parameter server initially sends a global model to each participant. After training with local data, the participants are only required to share gradients for model updates. Then the server aggregates the gradients and transmits the updated model back to each user. However, recent studies have shown that gradient sharing is not as secure as it is supposed to be. We consider an *honest-but-curious* attacker, who can be the centralized server or a neighbor in decentralized training. The attacker can observe the gradients of a victim at any communication round, and it attempts to recover the training data and labels from gradients. In general, such kinds of attacks are named as Gradient Inversion Attacks (GIAs).

A majority of GIAs [88, 18, 78] purpose to minimize the distance between the generated gradients and ground-truth gradients. To generate dummy gradients, a pair of random data and labels are fed to the global model. Taking the distance between the gradients as an error and the dummy inputs as parameters, the recovery process

can be formulated as an iterative optimization problem. When the optimization procedure converges, the private data is supposed to be fully reconstructed. Moreover, some newly presented studies [13, 87, 7] can also reconstruct the original data in a closed-form algorithm. The key insight is to exploit the implicit relationships among the input data, model parameters and gradients of each layer, and find the optimal solution with minimum error.

To prevent attackers from disclosing private information through gradient sharing, some cryptography-based methods [5, 55], differential privacy-based approaches [61, 70, 48] and pixel-based perturbations [15, 25] have been introduced to enhance the security and privacy levels of FL. In addition, such privacy leakage can be mitigated by increasing the local iterations or batch sizes [69, 26] during model training.

In this chapter, we present a comprehensive review of GIAs and defenses against GIAs in FL. We first introduce our proposed taxonomy of GIAs, which categorizes the existing studies into two paradigms: *Iteration-based* and *Recursion-based* attacks. Then, we delve into the iteration-based attacks and divide the main components into *Data Initialization*, *Model Training*, and *Gradient Matching*. We also summarize the emerging defense strategies against GIAs, which focus on three perspectives: *Data Obscuration*, *Model Improvement*, and *Gradient Protection*.

2.2 Gradient Inversion Attacks

In this section, we provide a taxonomy of GIAs by characterizing the existing works into two paradigms: iteration-based and recursion-based attacks. In particular, we delve into iteration-based attacks and divide the main components into data initialization, model training, and gradient matching. We first describe the workflow of each category and formulate the optimization objectives. Then, we introduce the representative studies. Additionally, we compare the differences with the former category

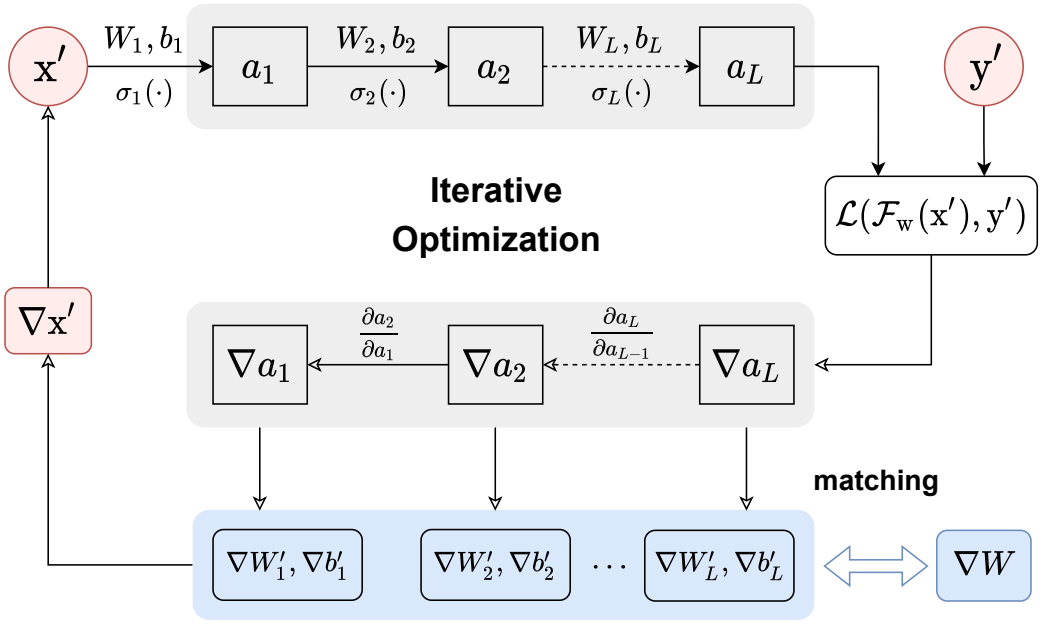


Figure 2.1: Reconstruction workflow of iteration-based GIAs. The process includes data, models, and gradients, which are represented in different colors. (**Red**: Reconstructed data, labels or gradients of the dummy inputs. **Gray**: Global model shared in distributed learning, whose parameters and structure are known. **Blue**: Ground-truth gradients and generated gradients for data recovery.)

from non-initialization, model structure, and linear solving. A detailed comparison of differences and similarities among recent GIAs is listed in Table 2.1.

2.2.1 Iteration-based Data Recovery

In the iteration-based workflow, the attacker first generates a pair of random data $\mathbf{x}'$ and labels $\mathbf{y}'$, which are regarded as optimization parameters for data recovery. After forward and backward propagation, generated gradients of model weights $\nabla\mathrm{W}'$ can be obtained. Here, the bias terms are ignored since they can be integrated into the networks by adding a neuron with the input of 1. According to the distance between generated gradients and ground-truth gradients of a victim, another backward prop-

agation is performed to calculate the gradients of dummy inputs, i.e., $\nabla \mathbf{x}'$ and $\nabla \mathbf{y}'$. From the perspective of reconstructed data, the reconstruction process includes once forward propagation, twice backward propagation and the update of dummy inputs, which is illustrated in Fig. 2.1.

The recovery of private data can be viewed as an iterative optimization process utilizing gradient descent. When the optimization converges, that is, the distance (e.g., ℓ_2 norm) between gradients is close, and the original data is supposed to be fully recovered. The optimization problem is formulated in Eq. (2.1), where $\mathbf{x}'^*, \mathbf{y}'^*$ are the optimized results.

$$\begin{aligned} \mathbf{x}'^*, \mathbf{y}'^* &= \arg\min_{\mathbf{x}',\ \mathbf{y}'} \left\| \nabla \mathrm{W}' - \nabla \mathrm{W} \right\|^2 \\ &= \arg\min_{\mathbf{x}',\ \mathbf{y}'} \left\| \frac{\partial \mathcal{L}\left(\mathcal{F}\left(\mathbf{x}', \mathrm{W}\right), \mathbf{y}'\right)}{\partial \mathrm{W}} - \nabla \mathrm{W} \right\|^2 \end{aligned} \tag{2.1}$$

Initialization of Input Data and Labels

To produce dummy gradients and perform gradient matching, the attacker first needs to generate random data and labels. Initialization involves the selection of distribution, the size of original data, and the asynchronous recovery of labels.

Distribution. Random Gaussian noise is most frequently used for data initialization in the majority of GIAs [88, 18, 78]. In addition, constant values [66] or random noise sampled from Uniform distribution [29] are also presented for data initialization. However, some experiment results demonstrate that GIAs often fail to converge due to bad initialization [18]. Hence, Wei et al. [69] prove that the convergence of a reconstruction process can be guaranteed by Lipschitz continuity:

$$f(\mathbf{x}'^{\mathrm{T}}) - f(\mathbf{x}'^*) \leq \frac{2L \left\| \mathbf{x}'^0 - \mathbf{x}'^* \right\|^2}{\mathrm{T}} \tag{2.2}$$

Table 2.1: Summary and classification of existing GIAs.

Publication	**Data Initialization**		**Model Training**		**Grad Matching**	
	Distribution	*Resolution*	*Network*	*Batch size*	*Loss-fn*	*Optimizer*
GIAs of Iteration-based Workflow						
DLG [88]	Gaussian	64×64	LeNet	8	ℓ_2 dist	L-BFGS
iDLG [84]	Uniform$^{\mathbb{L}}$	32×32	LeNet	1	ℓ_2 dist	L-BFGS
CPL [69]	Geometric	128×128	LeNet	8	ℓ_2 dist	L-BFGS
InvGrad [18]	Gaussian$^{\mathbb{L}}$	224×224	ResNet$^{\mathbb{T}}$	8 (100)	Cosine	Adam
SAPAG [66]	Constant	224×224	ResNet$^{\mathbb{T}}$	8	Gauss	AdamW
GradInversion [78]	Gaussian$^{\mathbb{L}}$	224×224	ResNet$^{\mathbb{T}}$	48	ℓ_2 dist	Adam
GradDisagg [36]	Gaussian	32×32	MLP	32 (128)	ℓ_2 dist	L-BFGS
GradAttack [26]	Gaussian$^{\mathbb{L}}$	224×224	ResNet$^{\mathbb{T}}$	128	Cosine	Adam
Bayesian [3]	Gaussian	32×32	ConvNet$^{\mathbb{T}}$	1 (32)	Cosine	Adam
CAFE [29]	Uniform	32×32	Loop-Net	100	ℓ_2 dist	SGD
GIAS [28]	Latent	64×64	ResNet$^{\mathbb{T}}$	4	Cosine	Adam
GIAs of Recursion-based Workflow						
PPDL-AHE [55]	N/A	20×20	MLP	1	Gradient division	
PPDL-SPN [13]	N/A	32×32	ConvNet	8	Linear solving	
R-GAP [87]	N/A	32×32	ConvNet	1	Inverse matrix	
COPA [7]	N/A	32×32	ConvNet	1	Least-squares	

$\mathbb{L}$: The labels can be directly identified or extracted from shared gradients.

$\mathbb{T}$: The results of data recovery are compared in different model training states.

where L is the Lipschitz constant, T denotes the condition of termination, and $\mathbf{x}'^0$, $\mathbf{x}'^T$, $\mathbf{x}'^*$ represent the initializing data, terminated results and optimal recovery, respectively. Based on Eq. (2.2), Wei et al. [69] propose patterned randomization and theoretical optimal initialization methods, which are more efficient and stable than Gaussian or Uniform noise.

Image Resolution. Without loss of generality, we consider the scenarios of image classification tasks. The resolution of raw images is an important factor that affects both initialization and the difficulty of recovery. The more pixels an image has, the more variables need to be optimized. Thus, it is more challenging to deploy attacks on complex datasets, such as ImageNet [12]. In contrast, relatively good results can be achieved on the low-resolution black-and-white images (e.g., MNIST [37], Fashion-MNIST [74]), even under mini-batch training. Till now, 224×224 pixels is the largest resolution for recovery.

Label Restoration. In a general procedure for iteration-based GIAs, dummy data and labels are simultaneously updated. However, if the ground-truth labels can be extracted in advance, data recovery will be accelerated and the computational complexity will also be reduced. Zhao et al. [84] first find out that the ground-truth label in a classification task can be directly revealed. For a mini-batch label restoration, [61, 78] propose that labels of different classes can be identified from each other, which assumes that there are non-repeating labels in the mini-batch training data. All of the above approaches extract the labels from the fully connected layer of gradients. Moreover, Wainakh et al. [64] exploit both the angle and magnitude of gradients to identify the labels. The angle shows whether a label is contained in the batch, while the magnitude indicates the number of duplicate labels. Dang et al. [9] present a more powerful label leakage attack, which can be applied to both image classification and speech recognition tasks. Beyond the FedSGD algorithm, this study also considers

recovering labels under multiple local iterations within the FedAvg algorithm.

Model Training for Gradient Generation

To obtain the generated gradients, the attacker needs to feed the initialized dummy data and labels into the model. Based on the error between the model outputs and the labels, the gradients of weights can be calculated through backward propagation. In the setting of distributed learning, the global model can be viewed as a white box, which means the model structure and weights are known. However, the depth of training network structures and the batch size of training data can implicitly affect the results of data recovery.

Network Model. Convolutional neural networks (CNN) or multilayer perceptron (MLP) are generally adopted as training networks for computer vision. Intuitively, the deeper the network is, the more parameters it contains. This raises two serious issues. First, the computational complexity is greatly increased, which makes it difficult or impossible for the optimization process to converge. Second, even if the procedure converges, there may exist multiple locally optimal solutions, resulting in a significant difference in the ground-truth value. So far, Geiping et al. [18] can recover the raw data on ResNet152 [24], although only a few images can be recognized. Yin et al. [78] achieve data recovery of high-resolution images on ResNet50, and display relatively better results. [29, 3] also investigate the relationship between the convergence states of model training and the errors of reconstruction.

Batch Training. For regular training, learning with mini-batch data can decrease the number of iterations and reduce the fluctuation of accumulated errors. However, this greatly increases the difficulty of GIAs. Given an observation of gradients, the problem of recovering original data is equivalent to the decomposition of averaged summation. Using the vanilla optimization approach in Eq. (2.1), Zhu et al. [88]

can only perform data recovery for a maximum batch size of 8. With the assistance of various regularization terms, Yin et al. [78], Huang et al. [26] all can recover the original private data with a batch size of over 30.

Data Update through Gradient Matching

The process of gradient matching measures the difference between generated gradients and ground-truth gradients and then calculates the update of dummy inputs. Essentially, the procedure for data recovery can be analogized to supervised learning, which means the ground-truth gradients are similar to a high-dimensional "label", while the dummy data and labels are the parameters to be learned in the optimization process. There are several important parts for gradient matching, the first is to obtain the gradients of a victim, and the next is proposing an effective method to minimize the distance.

Disaggregation. Considering the secure aggregation rules [5, 16], the adversaries can only observe a summation of the gradients. To perform gradient matching, it is necessary to decompose the individual gradients from the aggregation. Similar to the decomposition problem of mini-batch recovery, this is also a challenging problem. Lam et al. [36] first focus on this issue and formulate it as a matrix factorization problem. By leveraging the participant information acquired from device analytics, additional constraints contribute to the solution. If G_{agg} represents the aggregated gradients, the factorization problem can be solved by finding the participant vector p_k for each client k:

$$\begin{aligned} \text{Find} \quad p_k \quad \text{s.t.} \quad & \text{Null}(G_{\text{agg}}^{\mathsf{T}})\, p_k = 0 \\ & C_k p_k - c_k = 0 \\ & p_k \in \{0,1\}^n \end{aligned} \tag{2.3}$$

where $\text{Null}(\cdot)$ calculates the kernel of a matrix, and C_k specifies the participated rounds of client k.

Loss Function. Generally, an attacker is assumed to have direct access to the gradients of a victim. Thus, these studies focus on improving their algorithms to decrease the difference between gradients and recover more realistic data. The enhancement consists of optimization metrics and regularization terms. To measure the distance between generated gradients and ground-truth gradients, Euclidean distance (i.e., ℓ_2 norm) is a frequently used loss function [88, 69, 36]. However, Wang et al. [66] discover that gradients with large values dominate data recovery at the early stages. They hence propose a weighted Gaussian kernel as the distance metric. Furthermore, Geiping et al. [18] observe that the direction of gradients plays a more important role than magnitude, and substitute the ℓ_2 cost function with cosine similarity:

$$\arg \min_{\mathbf{x}' \in [0,1]^n} 1 - \frac{\langle \nabla \mathrm{W}', \nabla \mathrm{W} \rangle}{\|\nabla \mathrm{W}'\| \, \|\nabla \mathrm{W}\|}, \tag{2.4}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors and $\|\cdot\|$ represents the ℓ_2 norm of a vector. $\nabla \mathrm{W}$ and $\nabla \mathrm{W}'$ are the ground-truth gradients and generated gradients with respect to dummy data, respectively.

Regularization. To recover more realistic data from the batch, some auxiliary regularization terms are inserted into the cost functions. These regularization terms can be divided into two categories, one that constrains the fidelity of images, and the other that revises the position of the main object. Fidelity regularization [18, 78] steers the reconstructed data from impractical images, including total variation norm (TV), ℓ_2 norm and batch normalization (BN) [50, 43]. Group consistency regularization is presented by [78], which jointly considers multiple random seeds for initialization, and calculates the averaged data $\mathbb{E}(\mathbf{x}'_{g \in G})$ as reference. Any candidate whose recovery deviates from the "consensus" image of the group will be penalized.

$$\begin{cases} \mathcal{R}_{\text{fidel}}(\mathbf{x}') = \alpha_{\text{tv}} \mathcal{R}_{\text{TV}}(\mathbf{x}') + \alpha_{\ell_2} \mathcal{R}_{\ell_2}(\mathbf{x}') + \alpha_{\text{bn}} \mathcal{R}_{\text{BN}}(\mathbf{x}') \\ \mathcal{R}_{\text{group}}(\mathbf{x}', \mathbf{x}'_{g \in G}) = \alpha_{\text{group}} \left\| \mathbf{x}' - \mathbb{E}(\mathbf{x}'_{g \in G}) \right\|^2 \end{cases} \tag{2.5}$$

Applying $\mathcal{R}_{\text{fidel}}$ and $\mathcal{R}_{\text{group}}$ regularization terms can indeed support reconstructing

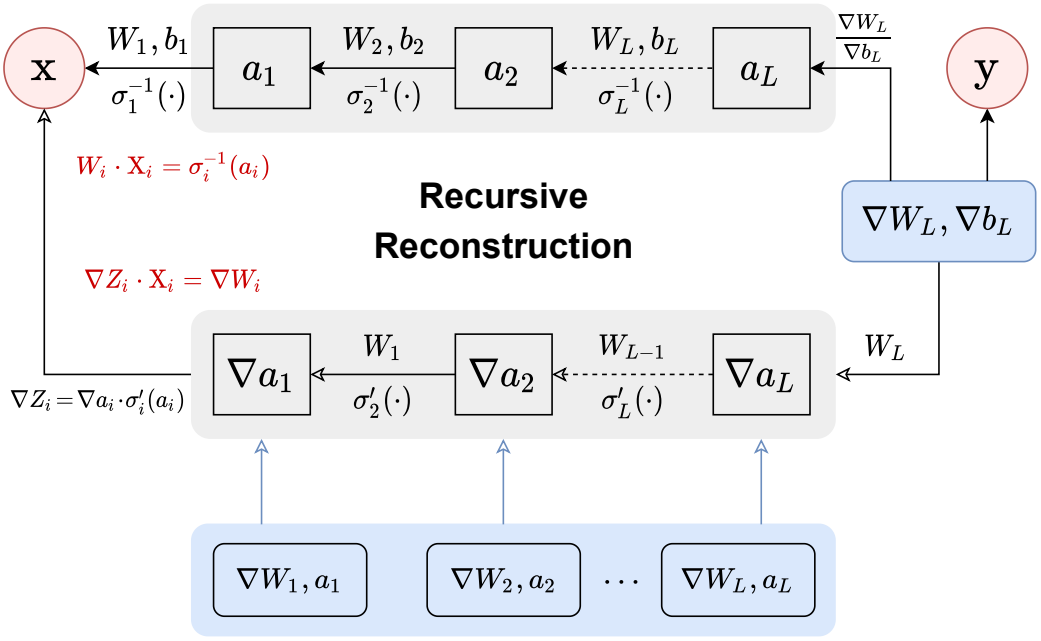


Figure 2.2: Reconstruction workflow of recursion-based GIAs. The process includes data, models and gradients, which are represented in different colors. (**Red**: Reconstructed data, labels or gradients of the dummy inputs. **Gray**: Global model shared in distributed learning, whose parameters and structure are known. **Blue**: Ground-truth gradients and generated gradients for data recovery.)

more realistic image data on complex datasets (e.g., ImageNet) under mini-batch training. Additionally, the GIAs are not overly sensitive to initialization.

2.2.2 Recursion-based Data Recovery

In recursion-based data recovery, the attacker can recursively calculate the input of each layer by finding the optimal solution with minimized error. Phong et al. [55] first discover that the input of a perceptron can be directly recovered from $x_k = \nabla \mathrm{W}_k / \nabla b$. This conclusion is later generalized to fully connected (FC) layers or MLP, as long as bias terms exist. Following this idea, Fan et al. [13] convert a convolutional layer into an FC layer by stacking the filters, and then utilize the above relationship. However,

they neglect the feature of reused weights in the convolutional layers, which is different from that of FC layers. To recover the image data of the first convolutional layer, Zhu and Blaschko [87] combine the forward and backward propagation and formulate the problem as solving a system of linear equations. The essence of such data leakage is that: the feature map and the gradient of kernels in the first convolutional layer have direct involvement with the original data. Extending to the i-th layer, it is possible to recover the input vector $\mathbf{x}_i$ by solving:

$$\begin{cases} \mathrm{W}_i \, \mathbf{x}_i = \mathrm{Z}_i \\ \nabla Z_i \, \mathbf{x}_i = \nabla \mathrm{W}_i \end{cases} \tag{2.6}$$

where Z_i and $\nabla \mathrm{Z}_i$ represent the feature map and its gradients.

Denoting the neuron outputs and activation function of each layer as a_i and $\sigma_i(\cdot)$, then we have the relationships of $\mathrm{Z}_i = \sigma_i^{-1}(a_i)$ and $\nabla \mathrm{Z}_i = \nabla a_i \cdot \sigma_i'(a_i)$. Hence, it is possible to recover the original data by recursively starting from the FC layer to the convolutional layer. Chen and Campbell [7] also propose a generic framework by combining multiple optimization problems under different situations. The workflow of the recursion-based framework is depicted in Fig. 2.2.

Compared to the iteration-based approaches, these recursion-based attacks have the following characteristics.

Non-initialization. Different from iterative optimization, a recursion-based attack can directly recover the original image without initialization or generating dummy inputs. Since the time of recovery is proportional to the square of the number of pixels, the current image resolution that can be recovered does not exceed 32×32. Furthermore, the extraction of labels refers to the above-mentioned works in Section 2.2.1.

Model Structure. The networks for experiments include only convolutional and fully connected layers. The pooling layers or shortcut connections in ResNet are not

considered, which would cause an accumulation of errors layer by layer, and make the reconstructed data extremely biased. However, these attacks can not deal with mini-batch training, which is a major difference compared to iteration-based attacks.

Linear Solving. As formulated in Eq. (2.6), the original data can be recursively solved by constructing linear equations. The feature maps and their gradients can be derived from the model weights and corresponding gradients. It is worth mentioning that such an attack depends on the integrity of gradients. Once the gradients are perturbed, the noisy solution will make the restored results completely unrecognizable.

2.3 Gradient Inversion Defenses

In this section, we summarize the emerging defense strategies from perspectives of data obscuration, model improvement, and gradient protection. A client can either obscure the private images from the data source, enhance the structure of network models, or protect the gradients before sharing. The main contributions of these defenses are described in Table 2.2.

2.3.1 Obscuration of Original Data

Since the target of GIAs is to recover the training data of a victim, an ideal defense strategy is to directly protect the raw data before training. We expect that the private input is difficult to reconstruct while the model utility does not degrade too much. Zhang et al. [80] propose the method of MixUp for data augmentation, where virtual training samples are generated by linearly combining a pair of data and labels. These generated examples can not only improve the accuracy of the training model but also "aggregate" the original data. Based on MixUp, Huang et al. [25] introduce the idea of cryptography to protect the data using one-time private keys. In particular,

Table 2.2: Summary and classification of existing defenses against GIAs.

Surface	Method	Study	Key Contribution
Data	MixUp	[80]	Data enhancement by linearly combining the inputs
	InstaHide	[25]	Encrypt the MixUp data with one-time secret keys
	Pixelization	[14, 15]	Perturb the raw data with pixelization-based method
Model	Dropout	[86]	Add an additional dropout layer before the classifier
	Local iters	[69]	Share gradients after multiple local training iterations
	Architecture	[87]	Reduce the number of convolutional kernels properly
Gradients	Aggregation	[79]	Apply Homomorphic Encryption to protect gradients
		[39]	Utilize Secure Multi-Party Computation to aggregate
	Perturbation	[61]	Perturb data representation layer and maintain utility
		[70]	Add adaptive noise with differential privacy guarantee
	Compression	[63]	Compress the smaller values in gradients to zero
		[30]	Transmit the sign of gradients for model updates

a portion of images from both private and public datasets are randomly selected for combination, and then the pixels are flipped according to the keys. This lightweight approach prevents an attacker from recovering the training data and ensures the usability of the data. Pang et al. [52] also mixup the input with other clean samples to improve the adversarial robustness of training models. In addition, Fan [15] protect the images with pixelization and Gaussian blur approaches, which can be used not only for distributed training but also for data publication, such as crowd-sourcing.

2.3.2 Improvement of Training Model

As for training models, in addition to increasing the depth of the neural network or training with a mini-batch (Section 2.2.1), we introduce some newly presented but effective approaches. In general GIAs, it is assumed that the gradients are sent after one round of local training. Wei et al. [69] propose to schedule and control the number of local training iterations before gradient sharing, which makes it more difficult to reconstruct the private data. Experiments demonstrate that the success rates of data recovery have dropped by more than 60% when performing 10 local iterations. Modifications to the network structure can also defend against GIAs. Zheng [86] propose to add a simple dropout layer between the encoder and the classifier to solve the problem of overfitting. During the training process, there may be certain neurons with larger activation values, indicating the features of training data are overly memorized. If a proportion of neurons are randomly pruned, then it is possible to mitigate the privacy inference attacks. Considering that different network models may have different risks of privacy leakage, Zhu and Blaschko [87] present a rank-based security analysis. Such a method indicates that the more filters in a convolutional layer, the better the data recovery will be. Similarly, Geiping et al. [18] have also mentioned that it is impossible to recover the original data from gradients if the dimensionality of model parameters is lower than that of input data. This conclusion inspires us to appropriately reduce the number of parameters while ensuring the performance of the model.

2.3.3 Protection from Gradient Sharing

In distributed learning, since model updates are performed based on gradient exchanging, the straightforward privacy-preserving approach is to protect the gradients. Summarizing the existing studies, we divide them into *aggregation*, *perturbation* and *compression*-based defense strategies.

Cryptography-based methods can generally guarantee the security and privacy of individual gradients without compromising their utility. [5, 39] use secure Multi-Party Computation (MPC) to compute the summation result of model updates. [55, 32, 79] implement Homomorphic Encryption (HE) to carry out operations on the ciphertext space for gradient aggregation. Even if an attacker steals the information through man-in-the-middle (MITM) attacks, the adversary cannot decrypt it to obtain the ground-truth gradients. However, these approaches not only require modifications to the training architecture but also exponentially increase the computation time, bandwidth and data storage, which are not suitable for large-scale distributed learning.

Gradient perturbation is another frequently used approach for privacy protection. Studies like [62, 68, 76] propose to add Gaussian noise into the transmitted gradients with the guarantee of Differential Privacy (DP). He et al. [22] essentially reveal how iterative training impacts privacy, and establishes the relationship between generalization and privacy-preserving. Sun et al. [61] find the key to GIAs lies in the data representation layer, and only perturb the gradient values in this critical layer. Wei et al. [70] propose a method with dynamically adjustable noise that can achieve high resilience against GIAs. Except for injecting noise, Zhu et al. [88] discover that some compression methods, originally used to reduce communication overhead, can also be used to prevent data recovery. Vogels et al. [63] propose to prune the smaller values to zero by a certain percentage, and Karimireddy et al. [30] only transmit the sign of gradients. These methods can resist attacks to a certain extent while maintaining the performance of the training model.

2.4 Challenges and Opportunities

After reviewing the existing GIAs and their corresponding defenses, we have identified some challenges and opportunities for improvement. Building a secure, trustworthy and robust FL system requires a comprehensive understanding of potential threats

and vulnerabilities. Only by fully investigating the attack surfaces can we effectively design and implement robust defense strategies. Therefore, we point out the problems or challenges of previous GIAs in the following aspects.

1. One significant challenge lies in recovering training labels. As discussed in Section 2.2.1, label recovery is pivotal for reconstructing original training samples. While methods like DLG [88] demonstrate limited success in simultaneously recovering both data and labels in small-batch, few-category classification scenarios. However, in large-batch, multi-category classification tasks, no method adopts such a strategy to optimize both variables at the same time. Existing approaches predominantly assume known labels, concentrating solely on data reconstruction, such as [18, 21]. Notably, related research [38] underscores the critical importance of accurate labels, highlighting the significant degradation in data recovery performance when labels are incorrect. Although methods like [84, 78, 9, 64] aim to analytically recover labels from shared gradients, they still face several limitations. For instance, [84] is restricted to restoring single data points, which [78, 64] only function effectively with specific activation functions (Sigmoid or ReLU). Moreover, [78, 9] require the unrealistic assumption that all training labels are unique. These constraints demonstrate that existing label recovery techniques struggle to generalize to more realistic FL scenarios.

2. The second challenge revolves around exploiting gradient relationships. As detailed in Section 2.2.2, existing recursion-based GIAs primarily focus on mining correlations between gradients themselves (gradient-to-gradient) and gradients with respect to data (gradient-to-data). For instance, PPDL-AHE [55] leverages gradients of the fully connected layer's weight and bias to directly reconstruct input features. Moreover, R-GAP [87] and COPA [7] utilize relationships within both forward and backward propagation: input-weight-output connections during forward propagation and input-weight gradient-output gradient connections during backpropagation. These relationships are then exploited to construct a

system of linear equations for recursively recovering input samples. Although primarily effective for batch size 1 scenarios, these studies underscore the crucial role of gradient correlations in privacy leakage. However, we believe current research in this domain remains inadequate. The ability to infer unknown or unshared gradients from known ones based on established correlations could empower attackers to extract sensitive information and potentially breach privacy. Therefore, we posit that effectively harnessing the power of gradient correlations holds significant potential for uncovering novel and unexpected attack vectors.

3. The third challenge is to reconstruct data in a targeted manner. As summarized in Table 2.1, while current optimization-based GIAs strive to recover increasingly complex data (e.g., ImageNet [12]) trained on deeper neural networks and larger batch sizes, few studies explore methods for efficiently retrieving specific target data. If approaches like [18, 78, 21] are used to extract target data, the entire batch must be recovered before filtering for desired data points based on proximity to the targets. However, as presented by [17], GradInversion [78] can only reconstruct 28% of training data when applied on a batch size of 48 for a ResNet50 model, which is far from efficient and effective. Furthermore, from an attacker's standpoint, recovering specific target data maximizes his potential gain or impact. Applying such a targeted attack strategy within FL could lead to system-wide abuse and pose a significantly greater privacy risk than existing GIAs. Consequently, investigating targeted attacks capable of directly recovering desired training data from gradients is crucial.

In summary, we identify three key aspects for exploration within current research on GIAs: label information inference, gradient relationship exploitation, and targeted data reconstruction. Addressing these challenges will not only enable the development of more sophisticated attacks but also deepen our understanding of privacy threats arising from gradient leakage in FL.

2.5 Chapter Summary

In this chapter, we provide a comprehensive review of recent advances in Gradient Inversion Attacks (GIAs), including both offensive and defensive methodologies. To the best of our knowledge, this is the first attempt to systematically categorize GIAs using a novel taxonomy and elucidate the primary steps involved in deploying such attacks within a Federated Learning system. We also reclassify representative defense strategies employed to mitigate data recovery risks. Following a thorough examination of existing research on GIAs, we identify several unresolved challenges warranting further investigation, including label information inference, gradient relationships exploitation, and targeted data reconstruction. We will delve into specific improvements and contributions stemming from these challenges in subsequent chapters.

Chapter 3

Posterior Probability-Based Label Recovery Attack from Gradients in Federated Learning

3.1 Introduction

Federated Learning (FL) has become a popular paradigm for training machine learning models in privacy-sensitive applications [44, 77]. In FL, the clients compute gradients on their local devices and then send the gradients to the server for aggregation and global model update. Since the private data is preserved on the client side, FL is supposed to offer more privacy protection than the centralized learning paradigm. However, a category of attacks called *Gradient Inversion* has shown that the shared gradients can be exploited to reconstruct the training data [88, 18, 78]. Moreover, some analytical attacks can recover the labels from gradients of a classification model by analyzing the relationship between the gradients and the labels [84, 64, 42]. However, none of these works explain the nature of label recovery or exhibit the applicability to other classification problems. We hence raise the following

key questions: (i) *What is the essence of label leakage from gradients?* And (ii) *How to implement a generalized attack for label recovery?*

In this chapter, we explore and answer these two questions from both theoretical and practical perspectives. In particular, starting from the *focal loss* function, we first derive an important relationship among the gradients, labels and posterior probabilities in a concise form. This conclusion can be applied to a variety of loss functions, which reveals the connection between the gradients and the labels in a classification model. Then we explain the fundamental reason for our findings from the *exponential family* of distributions. We show that the gradient with respect to logits is the expectation of the target labels, which provides a convenient way to reduce computation costs but opens a "backdoor" for label leakage. Finally, we propose a generalized attack for label recovery by estimating the posterior probabilities of the target batch from an auxiliary dataset. The key insight is based on our empirical observation that the positive (negative) samples of a class have approximate probability distributions. By fitting the auxiliary dataset into the global model, we can estimate the target posterior probabilities, and then recover the labels of a specified class by substituting the gradients and the posterior probabilities into the derived formula.

Our main contributions are summarized as follows:

- For the first time, we investigate the root cause of label leakage from gradients, and find the gradient with respect to the logits is only related to the posterior probabilities and the target labels in various loss functions, such as focal loss and binary cross-entropy loss.
- We explain the intrinsic reason for our findings from the perspective of the exponential family, and conclude that the combination of cross-entropy loss and Softmax or Sigmoid activation function opens a "backdoor" for the label restoration attacks.
- We evaluate our attack on a variety of FL settings and classification variants,

and demonstrate that it outperforms the prior attacks in terms of *Class-level Accuracy (ClsAcc)* and *Instance-level Accuracy (InsAcc)*.

3.2 Related Work

Here we first review some works most related to ours, including Federated Learning, Gradient Inversion Attacks, and existing label recovery attacks.

3.2.1 Federated Learning

Federated Learning (FL) is a privacy-preserving machine learning paradigm that enables multiple clients to collaboratively train a global shared model without collecting their private data [44, 77]. In FL, the clients train the shared model locally and then send the model update (i.e., the gradient of the model parameters) to the server. The server aggregates the uploaded gradients from the selected clients and updates the global model. The training procedure is repeated until the global model converges. Since the private data is preserved on the client side, FL is widely used in privacy-sensitive applications such as finance [41, 58] and healthcare [75, 51].

3.2.2 Gradient Inversion Attacks

Zhu et al. [88] initially propose Gradient Inversion Attacks (GIAs), which can reconstruct the training data $\boldsymbol{x}$ and corresponding labels $\boldsymbol{y}$ from the shared gradients in FL. An honest-but-curious attacker generates a batch-averaged dummy gradient by fitting the global model with a batch of dummy data, and then iteratively updates the dummy data to minimize the distance between the dummy gradient and the target gradient. Geiping et al. [18] use cosine similarity as the error function and add total variation as a regularization term to improve the quality of reconstruction. Yin et al.

[78] present group consistency regularization to enhance the restoration of the object locations in the images. Jeon et al. [28] leverage a generative model pre-trained on the same data distribution to improve the quality of the recovered images. Moreover, recent studies also propose inversion attacks in other tasks, such as natural language processing [20, 2] and speech recognition [10]. The success of these attacks is based on an underlying assumption that the gradient is an approximate bijection of the training data. Thus, decreasing the gap between gradients equals optimizing the dummy data towards the ground-truth sensitive data.

3.2.3 Analytical Label Recovery Attacks

Zhao et al. [84] propose an analytical label attack named iDLG, which can directly infer the label from $\nabla \boldsymbol{W}$ of the classification layer. They derive that the gradients with respect to the logits $\boldsymbol{z}_j$ equal to $\sigma(\boldsymbol{z}_j) - 1$ if j is the target index c of the one-hot label, and $\sigma(\boldsymbol{z}_j)$ if $j \neq c$, where $\sigma(\cdot)$ denotes the Softmax function. When the model uses a non-negative activation, such as ReLU or Sigmoid, $\nabla \boldsymbol{W}_c$ consists of negative values, while the other rows are positive. Thus, the attacker can extract the label by simply comparing the signs of the gradient $\nabla \boldsymbol{W}$. However, iDLG only applies to single-batch labels, and the activation of the model must be non-negative.

Wainakh et al. [64] exploit the direction and magnitude of $\nabla \boldsymbol{W}$ to determine how many instances of each class are in the target batch. They formulate the problem as $\sum_{i=1}^{M} \nabla \boldsymbol{W}_j = \lambda_j m + s_j$, where λ_j is the number of batch labels of the j-th class, m is the impact factor related to the input features, and s_j is a class-specific offset caused by misclassification. Using known data and labels, impact m and offset s_j can be estimated from multiple sets of gradients, and then λ_j can be calculated.

Ma et al. [42] transform the label recovery problem into solving a system of linear equations. For each class j, they regard $\sigma(\boldsymbol{z}_j) - 1$ and $\sigma(\boldsymbol{z}_j)$ as the coefficients of the target label and the other labels, respectively. By constructing these coefficients into

a matrix $\boldsymbol{A}$, they can solve the label vector $\boldsymbol{y}$ from the equation $\boldsymbol{Ay} = \nabla \boldsymbol{b}$, where $\nabla \boldsymbol{b}$ is the gradient with respect to the bias term of the last layer. However, none of these works explain the essence of label leakage from gradients or address the issue of whether the label attacks can apply to other classification variants.

3.3 Preliminaries

3.3.1 Focal Loss in Multi-class Classification

According to the definition of *binary focal loss* in [40], we extend it into multi-class scenarios. In a multi-class classification task using cross-entropy (CE) Loss, the CE loss can be written as follows:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{i=1}^{K} y_i \log(p_i) = -\sum_{i=1}^{K} \log(p_i^{y_i}) = \begin{cases} -\log(p_1) & \text{if } y_1 = 1 \\ -\log(p_2) & \text{if } y_2 = 1 \\ \quad\vdots & \\ -\log(p_K) & \text{if } y_K = 1, \end{cases} \tag{3.1}$$

where $\boldsymbol{y}$ is the one-hot label vector, $\boldsymbol{p}$ is the predicted probability vector, and K is the number of classes. For any class i, we use p_t to represent the confidence degree of the model's prediction as the following:

$$p_t = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{otherwise,} \end{cases} \tag{3.2}$$

where $t = i$. To be consistent with the original focal loss in [40], we use t to represent the class index instead of i, and t is identical to i.

To solve class imbalance, focal loss assigns an auto-determined weight $(1 - p_t)^\gamma$ and a pre-determined weight α_t to each class t. Finally, we define the focal loss for multi-class classification tasks as:

$$\mathcal{L}_{\mathrm{FL}}(p_t) = -\sum_{t=1}^{K} \alpha_t (1 - p_t)^\gamma \log(p_t). \tag{3.3}$$

The focal loss has two hyperparameters: α_t and γ. The former is the weight of the target class, and the latter is the focusing parameter that controls the degree of class imbalance.

3.3.2 Definition of Class-wise Probabilities

In a multi-class classification problem, each instance in the dataset belongs to one of several classes. We denote the total number of classes as K and a particular class as $k \in K$. In this context, we can define positive and negative samples for class k.

- **Positive Samples** ($\boldsymbol{x}_k^+$): The positive samples of class k satisfy that: $\boldsymbol{x}_k^+ = \{\boldsymbol{x}^{(i)} : y_c^{(i)} = k\}$, where $\boldsymbol{x}^{(i)}$ is the input of the i-th data, $y_c^{(i)}$ is the true class label of the corresponding data, and c is the class index.
- **Negative Samples** ($\boldsymbol{x}_k^-$): Similarly, the negative samples of class k satisfy that: $\boldsymbol{x}_k^- = \{\boldsymbol{x}^{(i)} : y_c^{(i)} \neq k\}$. In summary, the negative samples are associated with all but one $(K-1)$ of the other classes.

According to the positive and negative samples, we can then get the positive and negative probability for class k.

- **Positive Probability** (p_k^+): When a positive instance is fed into the model, the predicted probability of class k is termed the positive probability. Since the Softmax activation function is used in the output layer, the output posterior probability $\boldsymbol{p}^+$ is a vector of length k. Therefore, the positive probability for class k can be expressed as p_k^+.
- **Negative Probability** (p_k^-): Similarly, when a negative sample is input into the model, the k-th element of the output probability vector represents the negative probability, denoted as p_k^-. It is essential to note that any negative sample associated with the other $(K-1)$ classes contributes to p_k^-.

When using an auxiliary dataset to estimate the probabilities of the target training batch in FL, we denote the estimated positive and negative probabilities as $\hat{p}_k^+$ and $\hat{p}_k^-$, respectively.

In a batch size of B samples, we aim to recover the labels of each instance within the batch, i.e., $\boldsymbol{y} = [\sum_{i=1}^{B} y_1^{(i)}, \sum_{i=1}^{B} y_2^{(i)}, \cdots, \sum_{i=1}^{B} y_B^{(i)}]$. Since this is a multi-class classification problem, the ground-truth labels $\boldsymbol{y}$ can also be represented by the occurrences of each class: $\boldsymbol{y} = [n_1, n_2, \cdots, n_K]$, where n_k is the number of samples belonging to class k and K is the number of total classes.

The number of class-wise labels at class k can be defined as: $n_k = \sum_{i=1}^{B} \delta(y_c^{(i)} = k)$. Here, n_k is the number of samples belonging to class k, B is the batch size, $y_c^{(i)}$ is the true class label of the i-th instance in the batch, and $\delta(\cdot)$ denotes the Kronecker delta function, which equals 1 if the condition inside is true and 0 otherwise.

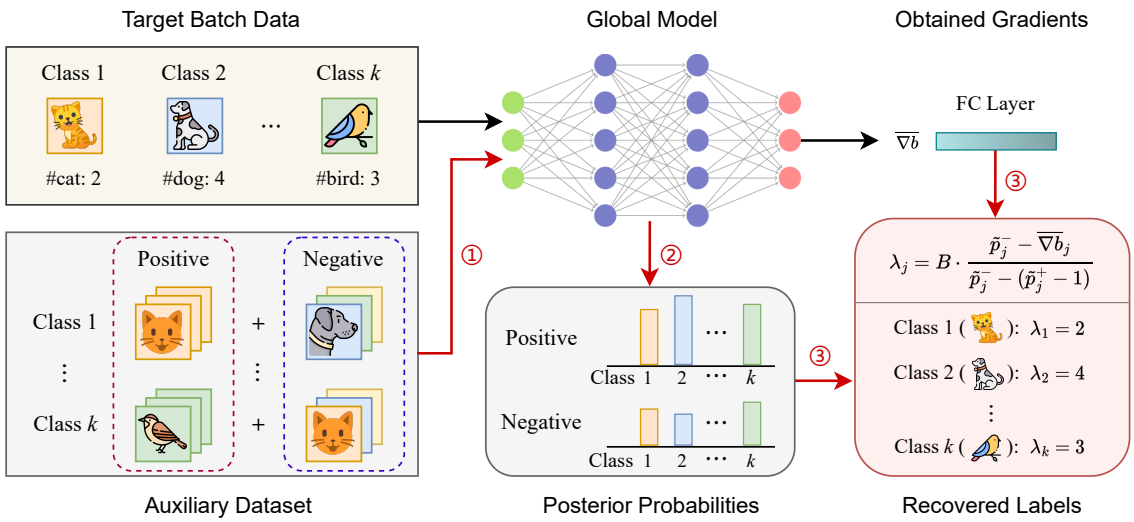


Figure 3.1: Workflow of our proposed label recovery attack. ① Acquire an auxiliary dataset and divide it into positive and negative subsets for each class. ② Fit the data into the global model for estimating the posterior probabilities of the target batch. ③ Recover the batch labels by substituting the gradients and the posterior probabilities into the derived formula.

3.3.3 Problem Formulation

For a K-class classification task in FL, the FedSGD [44] algorithm is used to train the global model $\boldsymbol{\theta}_g$. At a given iteration, a victim client v trains the model with its local batch data $\boldsymbol{x}$ and labels $\boldsymbol{y}$, where $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}_v$. Here, $\mathcal{D}_v$ denotes the data distribution of client v. Then, the client calculates the batch-averaged gradient $\nabla\boldsymbol{\theta}_g$ of the model parameters and sends it to the server. As the honest-but-curious FL server, we aim to recover the batch labels $\boldsymbol{y}$ from the shared gradient $\nabla\boldsymbol{\theta}_g$. The knowledge we have includes the global model $\boldsymbol{\theta}_g$, the shared gradient $\nabla\boldsymbol{\theta}_g$, and an auxiliary dataset $\boldsymbol{x}_a$ with the same data distribution as $\mathcal{D}_v$.

In our label attack, we mainly utilize the gradient with respect to the bias b in the last fully connected layer, i.e., $\overline{\nabla\boldsymbol{b}}$, to recover the target batch labels $\boldsymbol{y}$. From the auxiliary dataset $\boldsymbol{x}_a$, we randomly sample a portion of instances $\hat{\boldsymbol{x}}_a$, ensuring that the number of instances in each class equals a predefined threshold. Then we divide $\hat{\boldsymbol{x}}_a$ into positive samples $\hat{\boldsymbol{x}}_j^+$ and negative samples $\hat{\boldsymbol{x}}_j^-$ for each class j. By fitting the model $\boldsymbol{\theta}_g$ with these samples, we can obtain the averaged posterior probabilities $\hat{\boldsymbol{p}}^+$ and $\hat{\boldsymbol{p}}^-$ of the positive and negative samples, respectively. By substituting $\overline{\nabla\boldsymbol{b}}$, $\hat{\boldsymbol{p}}^+$ and $\hat{\boldsymbol{p}}^-$ into our derived formula in Section 3.4.1, we can recover the batch labels $\hat{\boldsymbol{y}}$. The workflow of our label attack for any victim client v is illustrated in Fig. 3.1.

3.4 Essence of Label Leakage

In this section, we investigate the root cause of label leakage from gradients in FL. We first derive a generalized relationship between the gradient $\nabla\boldsymbol{z}$ and the labels $\boldsymbol{y}$, which can be applied to various classification tasks. Then we explain the intrinsic reason for these findings from the perspective of the exponential family. We conclude that the combination of cross-entropy loss and Softmax is intended to reduce the amount of computation, but opens a backdoor to potential attacks.

3.4.1 Generalized Expression of Gradients

Without loss of generality, we consider a multi-class classification task using focal loss as the loss function and Softmax as the activation function. Focal loss is an extension of the cross-entropy loss, which is proposed to solve the problem of class imbalance [40]. The definition of focal loss in multi-class scenarios can be represented as:

$$\mathcal{L}_{\mathrm{FL}}(p_t) = -\sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} \log p_t, \tag{3.4}$$

where p_t is the categorical probability of target class t, α_t is the weight of the target class, γ is the focusing parameter that controls the degree of class imbalance, and K is the number of classes.

For the t-th class, $p_t = p_i$ if $i = t$, and $p_t = 1 - p_i$ if $i \neq t$. Here, p_i is the Softmax probability of the i-th class[1], and y_i is the corresponding label. If we set $\gamma = 0$ and $\alpha_t = 1$, the focal loss is equivalent to the cross-entropy loss $\mathcal{L}_{\mathrm{CE}}(p_i) = -\sum_{t=i}^{K} y_i \log p_i$. Under the same setting, if $K = 2$, $y \in \{0, 1\}$ and the activation is Sigmoid[2], the focal loss is also equivalent to the binary cross-entropy loss $\mathcal{L}_{\mathrm{BCE}}(p) = -y \log p - (1 - y) \log (1 - p)$. Thus, starting from the analysis of focal loss allows us to derive a more general conclusion that can be applied to the other classification variants.

Theorem 1 (Gradient of Focal Loss)**.** *For a K-class classification task using the **focal loss** function and **Softmax** activation, we can derive that the gradient of logit z_j as follows:*

$$\nabla_{z_j} \mathcal{L}_{FL} = \sum_{t=1}^{K} \Phi(\alpha_t, p_t, \gamma) \cdot (p_j - \delta_{tj}), \tag{3.5}$$

where $\Phi(\alpha_t, p_t, \gamma) = \alpha_t (1-p_t)^{\gamma} \left(1 - \gamma \frac{p_t \log p_t}{1-p_t}\right)$ and $\forall t \in K$, we have $\Phi(\alpha_t, p_t, \gamma) \geq 0$. Besides, δ_{tj} is the Kronecker delta, which equals 1 if $t = j$ and 0 otherwise.

Proof. According to Eq. (3.4), we substitute the last p_t with its Softmax formula

[1] p_i is an instance-wise probability, while p_t is a class-wise probability corresponding to the label.

[2] A special case of Softmax with one neuron: $\frac{e^{z_1}}{e^{z_1}+e^{z_2}} = \frac{1}{1+e^{-(z_1-z_2)}} = \frac{1}{1+e^{-z}}$, where $z = z_1 - z_2$.

$p_t = \frac{e^{z_t}}{\sum_{k=1}^{K} e^{z_k}}$, and obtain the transformed focal loss function:

$$\begin{aligned}
\mathcal{L}_{\mathrm{FL}} &= -\sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} \log p_t \\
&= -\sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} \log \frac{e^{z_t}}{\sum_{k=1}^{K} e^{z_k}} \\
&= \sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} \log \sum_{k=1}^{K} e^{z_k} - \sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} z_t.
\end{aligned}$$

Let $\hbar = (1-p_t)^{\gamma}$, then we can deduce the gradient of logit z_j as follows:

$$\begin{aligned}
\nabla_{z_j} \mathcal{L}_{\mathrm{FL}} &= \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} \log \sum_{k=1}^{K} e^{z_k} + \sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} p_j - \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} z_t - \alpha_j (1-p_j)^{\gamma} \\
&= \sum_{t=1}^{K} \alpha_t \frac{\partial \hbar}{\partial z_j} \left(\log \sum_{k=1}^{K} e^{z_k} - z_t \right) + \sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} (p_j - \delta_{tj}) \\
&= \sum_{t=1}^{K} \alpha_t (1-p_t)^{\gamma} \left(1 - \gamma \frac{p_t \log p_t}{1-p_t} \right) (p_j - \delta_{tj}) \\
&= \sum_{t=1}^{K} \Phi(\alpha_t, p_t, \gamma) \cdot (p_j - \delta_{tj}).
\end{aligned}$$

□

From Theorem 1, we find that $\nabla_{z_j} \mathcal{L}_{\mathrm{FL}}$ is a summation of K items, where each term is a product of $\Phi(\alpha_t, p_t, \gamma)$ and $(p_j - \delta_{tj})$. The item $\Phi(\alpha_t, p_t, \gamma)$ can be regarded as the weight of the t-th class, and $(p_j - \delta_{tj})$ indicates the distance between the Softmax probability of the j-th class and the target categorical expectation at the t-th class. In particular, an interesting observation can be made from the latter item is that $(p_j - \delta_{tj})$ is only negative when $t = j$, and positive otherwise, which supports the following conclusions of different loss functions.

From this generalized relationship, we can also derive the gradient of logits $\boldsymbol{z}$, i.e., $\nabla \boldsymbol{z}$, in other classification variants by setting different values for α_t or γ, and choosing different label embeddings or activation functions. We summarize the commonly used

loss functions, argument settings and the corresponding gradients in Table 3.1. We can find that the gradient $\nabla \boldsymbol{z}$ is only related to the posterior probabilities $\boldsymbol{p}$ and the target labels $\boldsymbol{y}$. This finding reveals the connection between the gradient $\nabla \boldsymbol{z}$ and the target labels $\boldsymbol{y}$, which is the key to label recovery attacks. In the rest of this chapter, we mainly focus on the multi-class classification task using the Softmax function and cross-entropy loss, where the gradient is denoted as $\nabla \boldsymbol{z}$, the posterior probability is denoted as $\boldsymbol{p}$, and the target label is denoted as $\boldsymbol{y}$.

Table 3.1: Relationships between different loss function and its gradient $\nabla \boldsymbol{z}$.

Loss function	$\boldsymbol{\gamma}$	$\boldsymbol{\alpha}$	**Label**	**Activation**†	**Gradient** $\boldsymbol{\nabla z}$
Focal Loss	-	-	one-hot	Softmax (τ)	$\frac{1}{\tau}\Phi(\alpha_c, p_c, \gamma)(\boldsymbol{p}-\boldsymbol{y})^{\ddagger}$
Cross-entropy Loss	0	1	-	Softmax (τ)	$\frac{1}{\tau}(\boldsymbol{p}-\boldsymbol{y})$
Binary Cross-entropy Loss	0	1	binary	Sigmoid	$p-y$

† τ denotes the temperature parameter in Softmax for softness control.

‡ c is the target index of the one-hot label $\boldsymbol{y}$.

3.4.2 Explanation from Exponential Family

As seen from Table 3.1, item $(\boldsymbol{p}-\boldsymbol{y})$ exists in different combinations of target labels and activation functions. In particular, if $\Phi(\alpha_c, p_c, \gamma)$ in focal loss is treated as a constant, then gradient $\nabla \boldsymbol{z}$ of each loss function is dominated by $(\boldsymbol{p}-\boldsymbol{y})$. To unveil the essential reason for this phenomenon, we explain it from the perspective of the *exponential family* [1]. The exponential family is a class of probability distributions that has the following representation:

$$f_{\mathrm{x}}(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left[\,\boldsymbol{\eta}(\boldsymbol{\theta}) \cdot \boldsymbol{T}(\boldsymbol{x}) - A(\boldsymbol{\theta}) + B(\boldsymbol{x})\,\right], \tag{3.6}$$

where $\boldsymbol{\theta}$ is the parameter, $\boldsymbol{\eta}(\boldsymbol{\theta})$ is the canonical parameter, $\boldsymbol{T}(\boldsymbol{x})$ is the sufficient statistic, $A(\boldsymbol{\theta})$ is the log-partition function, and $B(\boldsymbol{x})$ is the base measure.

Multi-class Classification in Exponential Form

Take the multi-class classification task as an example, we can build the probability $p(\boldsymbol{x}|\boldsymbol{\theta})$ from the categorical distribution as follows:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{k=1}^{K} \theta_k^{x_k} = \exp\left[\log\left(\prod_{k=1}^{K} \theta_k^{x_k}\right)\right] = \exp\left[\sum_{k=1}^{K} x_k \log \theta_k\right], \tag{3.7}$$

where K is the number of categories, $x_k \in \{0, 1\}$ and $x_k = 1$ if x belongs to the k-th category, and θ_k denotes the probability when $x_k = 1$.

Since $\sum_{k=1}^{K} x_k = 1$, we can also derive that $\sum_{k=1}^{K} \theta_k = 1$. Replacing x_K with the first $(K-1)$ items of x, we can further express the probability $p(\boldsymbol{x}|\boldsymbol{\theta})$ as follows:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{\theta}) &= \exp\left[\sum_{k=1}^{K-1} x_k \log \theta_k + \left(1 - \sum_{k=1}^{K-1} x_k\right) \log \theta_K\right] \\ &= \exp\left[\sum_{k=1}^{K-1} x_k \log \theta_k - \sum_{k=1}^{K-1} x_k \log \theta_K + \log \theta_K\right] \\ &= \exp\left[\sum_{k=1}^{K-1} x_k \log \frac{\theta_k}{\theta_K} + \log \theta_K\right]. \end{aligned} \tag{3.8}$$

If we set $\boldsymbol{\eta}(\boldsymbol{\theta}) = [\log \frac{\theta_1}{\theta_K}, \cdots, \log \frac{\theta_{K-1}}{\theta_K}]$, $\boldsymbol{T}(\boldsymbol{x}) = [x_1, \cdots, x_{K-1}]$, $A(\boldsymbol{\theta}) = -\log \theta_K$, and $B(\boldsymbol{x}) = 0$, we can rewrite the above equation as:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left[\boldsymbol{\eta}(\boldsymbol{\theta}) \cdot \boldsymbol{T}(\boldsymbol{x}) - A(\boldsymbol{\theta}) + B(\boldsymbol{x})\right]. \tag{3.9}$$

So we can conclude that the categorical distribution belongs to the exponential family.

Derivation of Softmax and Cross-entropy Loss

From Eq. (3.9), we obtain $\boldsymbol{\eta}_k = \boldsymbol{\eta}(\theta_k) = \log \frac{\theta_k}{\theta_K}$, which can be transformed into: $\theta_k = e^{\eta_k} \cdot \theta_K$. Incorporating the characteristic of $\sum_{k=1}^{K} \theta_k = 1$, we summarize the

K items on both sides of the equation and deduce: $\theta_K \cdot \sum_{k=1}^{K} e^{\eta_k} = \sum_{k=1}^{K} \theta_k = 1$. Hence, θ_K can be represented as $\theta_K = \frac{1}{\sum_{k=1}^{K} e^{\eta_k}}$. Substituting θ_K into the relationship between θ_k and η_k, we finally derive the Softmax function as:

$$\theta_k = e^{\eta_k} \cdot \theta_K = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}}. \tag{3.10}$$

Combining Eq. (3.7) and Eq. (3.10), we can derive the log-likelihood of the categorical distribution as follows:

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}) = \log p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} x_k \log \theta_k = \sum_{k=1}^{K} x_k \log \left(\frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} \right). \tag{3.11}$$

In the multi-class classification task, $\boldsymbol{x}$ is a one-hot vector ($\sum_{k=1}^{K} x_k = 1$), and $\boldsymbol{\eta}$ is the logit vector. Thus, we can derive the expression of the cross-entropy loss function from the log-likelihood function as follows:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{x}, \boldsymbol{\theta}) = -\ell(\boldsymbol{\theta}; \boldsymbol{x}) = -\sum_{k=1}^{K} x_k \log \left(\frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} \right). \tag{3.12}$$

Therefore, we find that Softmax has a very strong connection with cross-entropy loss, which can be derived from the categorical distribution in the exponential family. This property interprets why the combination of Softmax and cross-entropy loss is widely used in multi-class classification tasks.

Gradient of Canonical Parameter $\boldsymbol{\eta}$

Given that $A(\boldsymbol{\theta}) = -\log \theta_K$ and $\theta_K = \frac{1}{\sum_{j=1}^{K} e^{\eta_j}}$, we then calculate the derivative of $A(\boldsymbol{\theta})$ with respect to η_k, i.e., $\nabla_{\eta_k} A(\boldsymbol{\theta}) = \frac{e^{\eta_k}}{\sum_{j=1}^{K} e^{\eta_j}} = \theta_k$. We proceed to derive the gradient of the log-likelihood $\ell(\boldsymbol{\theta}; \boldsymbol{x})$ with respect to the canonical parameter η_k as:

$$\nabla_{\eta} \ell(\boldsymbol{\theta}; \boldsymbol{x}) = \boldsymbol{T}(\boldsymbol{x}) - \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}) = \boldsymbol{T}(\boldsymbol{x}) - \boldsymbol{\theta}. \tag{3.13}$$

This formula not only exhibits an important feature of the exponential family but also discloses why the combination of Softmax and cross-entropy loss has the relationships

in Table 3.1. For the exponential family, when parameter $\boldsymbol{\theta}$ is determined, the change of $\ell(\boldsymbol{\theta}; \boldsymbol{x})$ with respect to $\boldsymbol{\eta}$ is determined only by $\boldsymbol{T}(\boldsymbol{x})$ and independent of other information about the samples. For a multi-class classification task, the sufficient statistic $\boldsymbol{T}(\boldsymbol{x})$ is the target label $\boldsymbol{y}$, the parameter $\boldsymbol{\theta}$ denotes the posterior probabilities $\boldsymbol{p}$, and the canonical parameter $\boldsymbol{\eta}$ implies the logits $\boldsymbol{z}$. Therefore, we can naturally obtain that $\nabla \boldsymbol{z} = \boldsymbol{p} - \boldsymbol{y}$, which is consistent with the previous conclusion.

3.4.3 Further Explanation of Label Leakage

The standard form of exponential family is given by Eq. (3.6). We know that the likelihood is the joint probability of all samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ given the parameter $\boldsymbol{\theta}$:

$$\begin{aligned} L(\boldsymbol{\theta}; \boldsymbol{x}) &= f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N | \boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} f(\boldsymbol{x}_i | \boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} \exp\left[\, \boldsymbol{\eta}(\boldsymbol{\theta}) \cdot \boldsymbol{T}(\boldsymbol{x}_i) - A(\boldsymbol{\theta}) + B(\boldsymbol{x}_i) \,\right] \\ &= \exp\left[\boldsymbol{\eta}(\boldsymbol{\theta}) \cdot \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i) - N A(\boldsymbol{\theta}) + \sum_{i=1}^{N} B(\boldsymbol{x}_i) \right]. \end{aligned} \tag{3.14}$$

If we take the logarithm of the likelihood function, we get the log-likelihood function:

$$\begin{aligned} \ell(\boldsymbol{\theta}; \boldsymbol{x}) &= \log L(\boldsymbol{\theta}; \boldsymbol{x}) \\ &= \boldsymbol{\eta}(\boldsymbol{\theta}) \cdot \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i) - N A(\boldsymbol{\theta}) + \sum_{i=1}^{N} B(\boldsymbol{x}_i). \end{aligned} \tag{3.15}$$

For the exponential family, parameter $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ are reversible. Hence, the derivative of canonical parameter $\boldsymbol{\eta}$ is denoted as:

$$\nabla_{\boldsymbol{\eta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i) - N \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}). \tag{3.16}$$

For the categorical distribution, we have $\nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}) = \boldsymbol{\theta}$, $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{x}$ and $\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}, \boldsymbol{x}) =$

$-\ell(\boldsymbol{\theta}; \boldsymbol{x})$. Consequently, we can derive the final expression as follows:

$$\begin{aligned} \nabla_{\boldsymbol{\eta}} \mathcal{L}_{\mathrm{CE}}(\boldsymbol{\theta}, \boldsymbol{x}) &= -\nabla_{\boldsymbol{\eta}} \ell(\boldsymbol{\theta}; \boldsymbol{x}) \\ &= N \nabla_{\boldsymbol{\eta}} A(\boldsymbol{\theta}) - \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i) \\ &= N\boldsymbol{\theta} - \sum_{i=1}^{N} \boldsymbol{x}_i. \end{aligned} \tag{3.17}$$

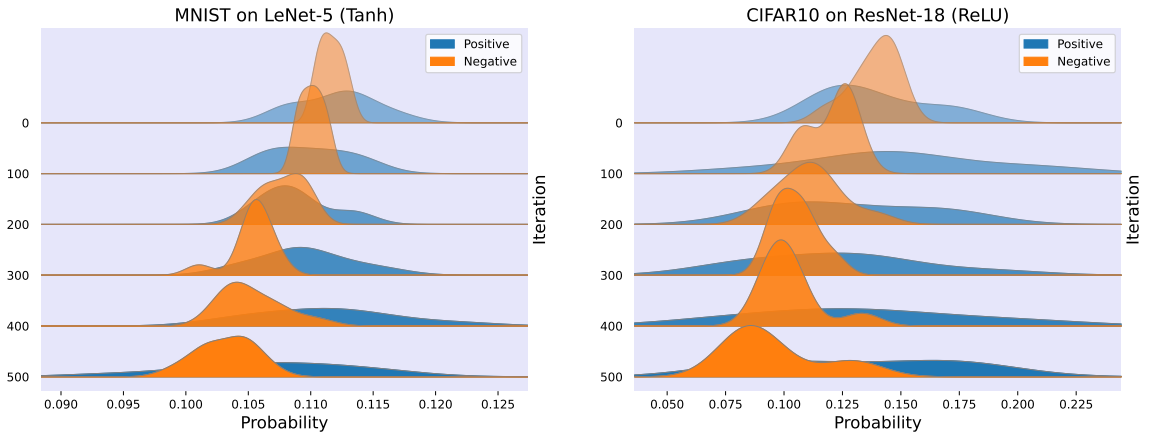
From this formula, we can observe that the characteristic of the exponential family gives an efficient way for CE loss to calculate the loss of canonical parameter $\boldsymbol{\eta}$ by just doing a subtraction rather than complex calculations. In the multi-class scenario, after substituting $\boldsymbol{\theta}$ with Softmax probability $\boldsymbol{p}$, $\boldsymbol{\eta}$ with logits $\boldsymbol{z}$, and $\boldsymbol{x}$ with target label $\boldsymbol{y}$, we can derive $\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{CE}}$ (when $N = 1$) as follows:

$$\nabla_{\boldsymbol{z}} \mathcal{L}_{\mathrm{CE}}(\boldsymbol{p}, \boldsymbol{y}) = \boldsymbol{p} - \boldsymbol{y}. \tag{3.18}$$

In summary, the combination of CE loss and Softmax is derived from the exponential family, which can reduce the amount of computation. However, it also opens a "backdoor" for potential threats, such as label leakage from gradients.

3.5 Label Recovery Attack

In this section, we propose our label recovery attack. We first present an important observation that the positive and negative samples of a class have approximate probability distributions. Based on this insight, we exploit an auxiliary dataset for estimating the posterior probabilities of the target batch. Finally, we propose an analytical method that can directly recover the number of labels of each class by substituting the probabilities and gradients into the derived formula.



(a) LeNet (Tanh) on MNIST.

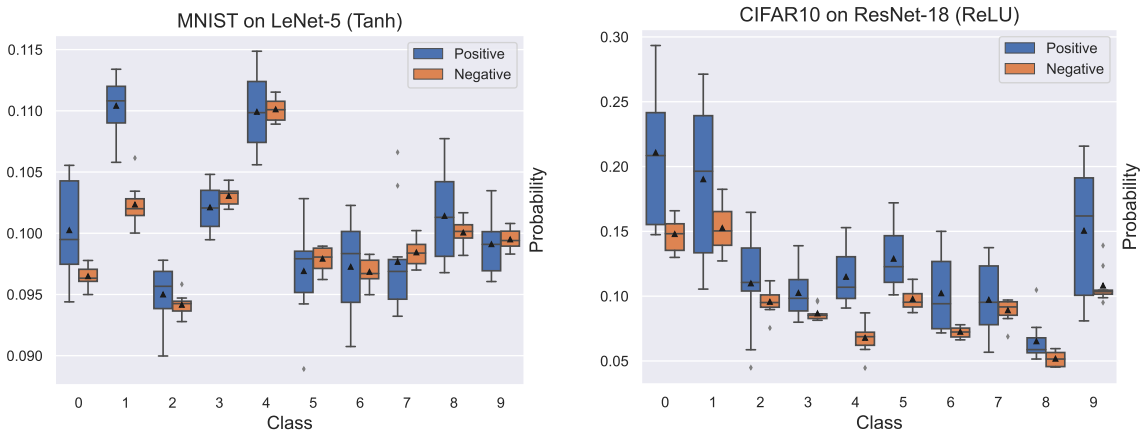
(b) ResNet18 (ReLU) on CIFAR-10.

Figure 3.2: Posterior probability distribution of positive and negative samples in different iterations of model training. The LeNet model is trained on MNIST, and the ResNet18 model is trained on CIFAR-10. We randomly select class 4 in MNIST and class 7 in CIFAR-10 for demonstration. The x-axis represents the probability value, and the y-axis represents the density.

3.5.1 Our Key Observation

In a multi-class classification task using a neural network, the model first outputs the logits $\boldsymbol{z}$ according to forward propagation, and then normalizes the logits into probabilities $\boldsymbol{p}$ through the Softmax function. By analyzing these posterior probabilities, we empirically observe that different positive and negative samples in the same class have approximate probability distributions.

We carry out the experiments on MNIST and CIFAR-10 datasets, which are trained on the LeNet [37] and ResNet18 [24] models, respectively. To eliminate the influence of the activation function, we choose Tanh for LeNet and ReLU for ResNet18. For each class, we treat the data belonging to this category as positive samples, and the others as negative samples. Then we aggregate the positive and negative samples from each class during the training procedure to show the correlations and variations



(a) LeNet (Tanh) on MNIST.

(b) ResNet18 (ReLU) on CIFAR-10.

Figure 3.3: Posterior probability distribution of different classes. We select the iterations of 100 for MNIST and 200 for CIFAR-10, respectively. The x-axis represents the class index, and the y-axis represents the probability value.

in their corresponding probabilities. For ease of demonstration, we only display the results of the first 500 training iterations.

Fig. 3.2 shows the ridgeline plots of probability distributions from a randomly selected class (class 4 in MNIST and class 7 in CIFAR-10) in different iterations. It can be seen that in the initial stages, the positive and negative probabilities are extremely close and have almost the same mean value. As the training progresses, the negative probabilities gradually decrease, while the positive probabilities slowly increase. Although the variance of the probabilities starts to increase, the mean values of the positive or negative probabilities remain within a small range.

We also exhibit the box plots of the probability distributions of all the classes in Fig. 3.3, whose training iteration is 100 for MNIST and 200 for CIFAR-10, respectively. It is shown that the positive or negative probabilities of each class are gathered around a certain value, although the values are slightly different. For some easily distinguishable classes, the positive and negative probabilities are already separated, such as class 1 in MNIST and class 4 in CIFAR-10.

We can interpret the above observations from the model's representation capability. For an untrained model, it cannot discriminate which class the instance belongs to, other than random guesses. Thus, the posterior probabilities of the positive and negative samples are almost the same, equal to $\frac{1}{K}$. As training proceeds, the model gradually learns the data distribution and can distinguish the positive or negative samples in each class. Although the degree of learning varies moderately for different classes, a robust model can give similar confidence scores (i.e., posterior probabilities) for the same class. This explains why the positive or negative samples per class have similar probability distributions.

3.5.2 Analytical Label Recovery

Based on the observation in Section 3.5.1, we can estimate the posterior probabilities of the target batch from an auxiliary dataset, whose data distribution is the same as the training data. Hence, we denote the estimated positive and negative probabilities of the j-th class as $\hat{p}_j^+$ and $\hat{p}_j^-$, respectively. Combined with the conclusions in Section 3.4.1, we can derive the following theorem for restoring the batch labels λ_j for each class j.

Theorem 2 (Label Recovery Formula)**.** *Having* ***an auxiliary dataset*** *with the same distribution of training data, we can recover the class-wise labels λ_j in the target batch according to the averaged gradient $\overline{\nabla b}_j$ and the estimated posterior probabilities $\hat{p}_j^+$ and $\hat{p}_j^-$ as follows:*

$$\lambda_j = B \cdot \frac{(\hat{p}_j^- - y_j^-) - \overline{\nabla b}_j / \hat{\varphi}_j}{(\hat{p}_j^- - y_j^-) - (\hat{p}_j^+ - y_j^+)}, \tag{3.19}$$

where y_j^+ and y_j^- are the pre-set label embeddings of class j, $\hat{\varphi}_j = \frac{1}{\tau}\Phi(\alpha_j, \hat{p}_j^+, \gamma)$ is an coefficient related to the j-th class, and B is the batch size.

Proof. Since $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, we can deduce that $\nabla \boldsymbol{b} = \nabla \boldsymbol{z}$ and $\overline{\nabla b}_j = \overline{\nabla z}_j$. We expand the averaged gradient $\overline{\nabla z}_j$ as a summation of B terms and replace the posterior

probability $p_j^{(n)}$ of each sample n with its estimated probabilities $\hat{p}_j^+$ and $\hat{p}_j^-$. Because $\Phi(\alpha_j, p_j^{(n)}, \gamma)$ is only related to the positive samples of the j-th class, we can replace $p_j^{(n)}$ with $\hat{p}_j^+$. So we have:

$$\hat{\varphi}_j = \frac{1}{\tau}\Phi(\alpha_j, \hat{p}_j^+, \gamma) = \frac{1}{\tau}\alpha_j(1-\hat{p}_j^+)^\gamma\left(1 - \gamma\frac{\hat{p}_j^+ \log \hat{p}_j^+}{1-\hat{p}_j^+}\right).$$

Assume that the first λ_j samples belong to the j-th class, and the rest $(B-\lambda_j)$ samples belong to other classes. Then from the first row of Table 3.1, we can derive that:

$$\begin{aligned}
\overline{\nabla b}_j &= \frac{1}{B}\sum_{n=1}^{B}\nabla b_j^{(n)} \\
&= \frac{1}{B}\left\{\sum_{n=1}^{\lambda_j}\varphi_j^{(n)}\left(p_j^{(n)} - y_j^{(n)}\right) + \sum_{n=\lambda_j+1}^{B}\varphi_j^{(n)}\left(p_j^{(n)} - y_j^{(n)}\right)\right\} \\
&\approx \frac{1}{B}\left\{\lambda_j\,\hat{\varphi}_j\left(\hat{p}_j^+ - y_j^+\right) + (B-\lambda_j)\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right)\right\}.
\end{aligned}$$

Therefore, we can finally derive that:

$$\lambda_j = B\cdot\frac{\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right) - \overline{\nabla b}_j}{\hat{\varphi}_j\left(\hat{p}_j^- - y_j^-\right) - \hat{\varphi}_j\left(\hat{p}_j^+ - y_j^+\right)} = B\cdot\frac{(\hat{p}_j^- - y_j^-) - \overline{\nabla b}_j/\hat{\varphi}_j}{(\hat{p}_j^- - y_j^-) - (\hat{p}_j^+ - y_j^+)}.$$

□

Specifically, the label embeddings are pre-defined by the FL protocol, which could be one-hot labels or smoothed labels. For one-hot labels, we have $y_j^+ = 1$ and $y_j^- = 0$. For smoothed labels, we have $y_j^+ = 1-\varepsilon$ and $y_j^- = \frac{\varepsilon}{K-1}$, where ε is the smoothing factor. Since $\Phi(\alpha_j, p_j, \gamma)$ is determined by p_j, we can use $\hat{p}_j^+$ for replacement and obtain $\hat{\varphi}_j$. By substituting the gradient $\overline{\nabla b}_j$, estimated coefficient $\hat{\varphi}_j$, positive probabilities $\hat{p}_j^+$, $\hat{p}_j^-$ and label embeddings y_j^+, y_j^- into the above formula, we can directly recover the number of labels λ_j for each class j.

3.6 Experiments

In this section, we evaluate the performance of our label recovery attack. We first introduce the experimental setup, evaluation metrics and baselines, and then show the experimental results. Moreover, we conduct ablation studies to analyze the effectiveness of our attack under different settings, such as batch size, loss function, auxiliary data size, and training iterations.

3.6.1 Experimental Settings

Datasets, Models and Activations

We evaluate our attack on three datasets, including MNIST [37], CIFAR-10 [34] and CIFAR-100 [34]. These datasets are widely used in FL research and cover a variety of classification tasks, such as handwritten digit recognition, object recognition and image classification. We choose the LeNet [37], VGG16 [59] and ResNet50 [24] as the training models for the above datasets, respectively. In addition, we select a bunch of activation functions, including Sigmoid, Tanh, ReLU [47], ELU [8] and SELU [33], to verify the universality of our attack.

Evaluation Metrics

To quantitatively evaluate the performance of label recovery, we use the following two metrics: (1) *Class-level Accuracy* (ClsAcc): the accuracy measures the proportion of correctly recovered classes; (2) *Instance-level Accuracy* (InsAcc): the accuracy measures the proportion of correctly recovered labels in the target batch. In particular, both of these two metrics are realized through Jaccard similarity, which is defined as follows:

$$J(\hat{\mathbf{y}}, \mathbf{y}) = \frac{|\hat{\mathbf{y}} \cap \mathbf{y}|}{|\hat{\mathbf{y}} \cup \mathbf{y}|} = \frac{|\hat{\mathbf{y}} \cap \mathbf{y}|}{|\hat{\mathbf{y}}| + |\mathbf{y}| - |\hat{\mathbf{y}} \cap \mathbf{y}|},$$

where $\hat{\mathbf{y}}$ and $\mathbf{y}$ denote the sets of recovered labels (or classes) and ground-truth labels (or classes) of the training batch.

Evaluation Baselines

Since iDLG [84] only applies to single batch training and non-negative activation functions, we exclude it from the comparison. We mainly compare our attack with LLG [64] and iLRG [42], which do not restrict the batch size or activation function. For LLG, we generate the dummy data with the same number as the target batch size and average the results of 10 runs. Since LLG and iLRG are all designed for untrained models, we mainly compare our attack with them in the untrained setting to reach a fair comparison.

3.6.2 Comparison with Baselines

To exhibit the versatility of our attack, we compare it with the baselines in 3 different groups of settings. We set the batch size to 32 for MNIST and CIFAR-10, and 256 for CIFAR-100. Without loss of generality, we assume that the training data of each class is uniformly distributed, and the auxiliary dataset is randomly sampled from the validation dataset with 100 samples per class. Furthermore, since the baselines are designed for untrained models, we also use initialized models for comparison to be fair. We run each experiment 20 times and report the average results in Table 3.2.

From the results, we can see that our attack performs better than the baselines and even achieves 100% ClsAcc and 100% InsAcc in most of the scenarios. In addition, the evaluation results also illustrate that compared with the dataset and model structure, the activation function has a greater impact on the performance of all label recovery attacks. This could be explained by the fact that some activation functions, like SELU, produce high variance, which causes the probability distribution of positive and negative samples from the same class to diverge significantly. Therefore, the

Table 3.2: Comparison of our attack with the baselines on diverse scenarios.

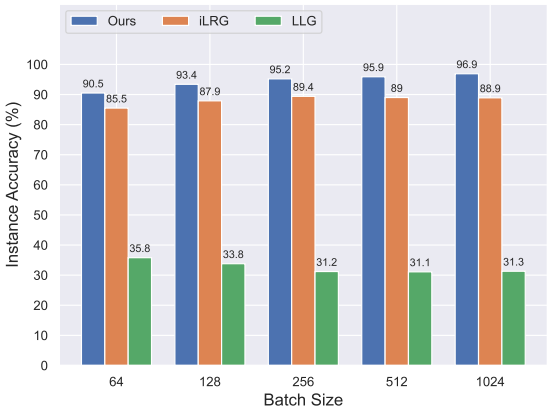
Dataset	Model	Activation	LLG		iLRG		Ours	
			ClsAcc	InsAcc	ClsAcc	InsAcc	ClsAcc	InsAcc
MNIST	LeNet	Sigmoid	0.954	0.973	0.946	0.880	**1.000**	**1.000**
		Tanh	0.506	0.163	**1.000**	**1.000**	**1.000**	**1.000**
CIFAR-10	VGG16	ReLU	0.995	0.997	**1.000**	**1.000**	**1.000**	**1.000**
		ELU	0.965	0.979	**1.000**	**1.000**	**1.000**	**1.000**
CIFAR-100	ResNet50	ReLU	0.937	0.952	**1.000**	**1.000**	**1.000**	**1.000**
		SELU	0.028	0.005	0.922	0.832	**0.968**	**0.951**

attack performance of SELU is worse than that of ReLU and ELU. Nevertheless, our attack still shows good results, which demonstrates its effectiveness and universality.

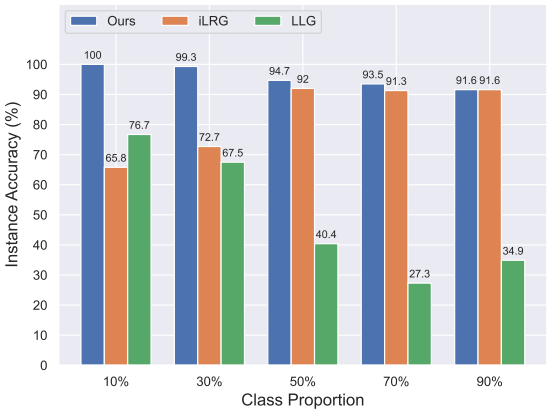
3.6.3 Comparison of Various FL Settings

From Table 3.2, it is shown that the attack baselines have the best performance for CIFAR-10 on VGG16 with ReLU activation. So we chose this scenario to analyze the effects of batch size and class imbalance. The batch size varies from 64 to 1024, which is closer to a realistic FL scenario. Class imbalance is a prevalent issue in FL, typically brought on by the unequal distribution of data across various clients. We compare the class proportions from 10% to 90% to simulate the class imbalance. Before launching the attacks, we train the classification model for one epoch with a learning rate of 0.001.

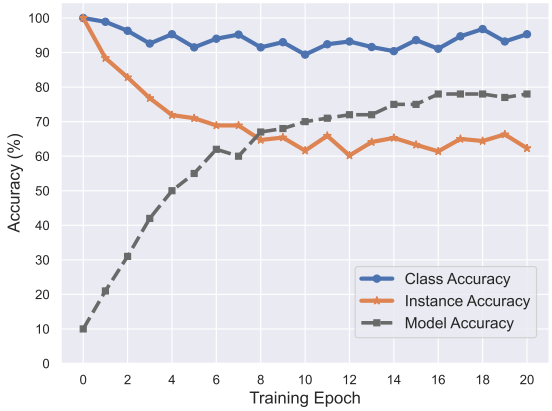
It is shown in Fig. 3.4 that our label recovery is robust to various batch sizes and class imbalance ratios, which maintains over 90% accuracy in all of these settings. As the batch size increases, the InsAcc of our attack gradually improves, which is



(a) Different batch sizes.



(b) Different class imbalance ratios.



(c) Different training iterations.

Figure 3.4: Comparison of label recovery performance against baselines across various FL settings. The batch size varies from 64 to 1024, the class imbalance ratios range from 10% to 90%, and we show the initial 20 iterations in the first training epoch.

mainly because the larger the batch size, the more robust the estimation of the averaged posterior probabilities. In addition, since we have the prior distribution of the training data, we can constrain and regularize the estimated labels to improve the success rate of label recovery.

3.6.4 Ablation Studies

Temperatures and Label Smoothing

We conduct ablation studies to analyze the effectiveness of our attack under different classification variants and scales of auxiliary data. Table 3.3 shows the InsAcc of our attack on an untrained model with the focal loss and cross-entropy loss under different hyper-parameters. The results indicate that our attack is robust to these variants, which can achieve 100% InsAcc in all of these settings.

Table 3.3: Comparison of our attack with classification variants.

Loss function	**Temperature τ**		**Label smoothing ε**	
	0.8	1.2	0.1	0.25
Focal Loss	1.000	1.000	1.000	1.000
Cross-entropy Loss	1.000	1.000	1.000	1.000

Attack on Focal Loss

We present additional experiments conducted on focal loss. We mainly test the parameters of τ, γ and ε on an untrained model, and average the experiments over 10 trials. Focusing on temperature τ, we present several cases where the accuracy is not 100%. In addition, by varying γ and ε on these settings, the accuracies are not

Table 3.4: Label recovery accuracies on focal loss ($\gamma = 2$, $\epsilon = 0$).

τ	**MNIST (LeNet)**		**CIFAR-10 (ResNet18)**		**CIFAR-100 (ResNet50)**	
	Our ClsAcc	Our InsAcc	Our ClsAcc	Our InsAcc	Our ClsAcc	Our InsAcc
0.5	0.980	0.906	0.990	0.960	1.000	1.000
0.75	0.980	0.945	1.000	0.994	1.000	1.000
0.9	0.990	0.954	1.000	1.000	1.000	1.000
1.25	0.990	0.983	1.000	1.000	1.000	1.000
1.5	1.000	0.998	1.000	1.000	1.000	1.000

affected. Table 3.4 shows the ClsAcc and InsAcc of our attack on the focal loss with different temperatures τ (batch size=64, activation=ReLU).

We have observed that the temperature parameter, τ, significantly impacts smaller datasets. When τ is smaller, the space of logits expands, complicating the estimation of batch posterior probabilities. Consequently, as τ decreases, label accuracy deteriorates. For the large datasets with more classes, such as CIFAR-10, the logits space is hardly influenced by changing different τ.

Scale of Auxiliary Data

Moreover, we also show the attack performance with different scales of auxiliary data in Fig. 3.5. For an untrained model, the attack performance is not sensitive to the scale of auxiliary data per class, which can achieve 100% InsAcc in all of these settings. For the training models, our attack is slightly affected by the scale but still maintains a reasonable accuracy. This manifests that estimating the posterior probabilities from external data is a robust solution.

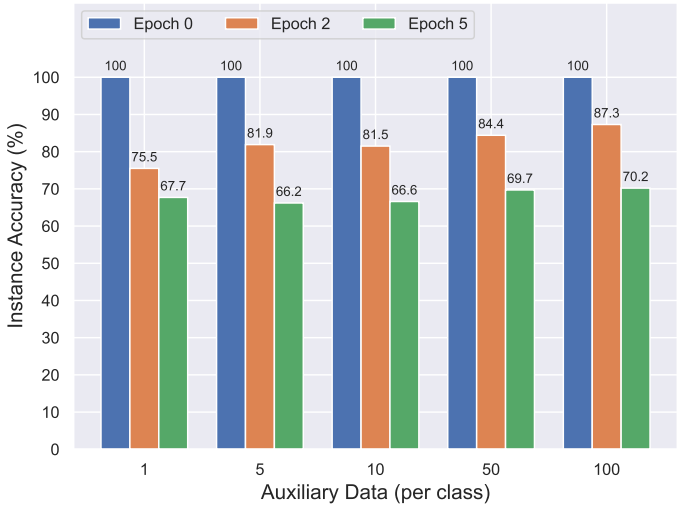


Figure 3.5: Instance accuracy with different scales of auxiliary data.

Auxiliary Data with Distribution Shift

To address concerns about significant distribution shifts, we conducted supplementary experiments, averaging results over 10 trials.

In the first set of experiments, we alternately used CIFAR-10 and CINIC-10 [11] as the training and auxiliary datasets. CINIC-10 extends CIFAR-10 by incorporating down-sampled ImageNet images. While there is some overlap in the distributions due to similar object categories, there is a notable bias in the data features. We employed an untrained VGG16 model with a batch size of 64.

Similarly, the second set of experiments involves alternating between MNIST and Fashion-MNIST (F-MNIST) [74] as the training and auxiliary datasets. Despite both datasets having 10 classes, the objects they represent are entirely different. MNIST dataset contains a lot of handwritten digits, while F-MNIST represents the article of clothing. Employing an untrained LeNet model with a batch size of 64, the results are presented in the table below.

This phenomenon can be explained as follows: In the initial phase of model training,

Table 3.5: Distribution shift between training dataset and auxiliary dataset.

Training Data	**Auxiliary Data**	**Model**	**Our ClsAcc**	**Our InsAcc**
CIFAR-10	CINIC-10	VGG16	1.000	1.000
CINIC-10	CIFAR-10	VGG16	1.000	1.000
MNIST	F-MNIST	LeNet	1.000	0.969
F-MNIST	MNIST	LeNet	1.000	0.948

the model cannot differentiate between samples from each class, assigning similar output probabilities to all fitted samples (i.e., $1/K$). For different training datasets like F-MNIST and MNIST, the model projects input figures to similar output probabilities with slight variations. This benefits our label recovery attack as it becomes easier to estimate the posterior probabilities of the target batch.

Given that F-MNIST is more intricate than MNIST, utilizing MNIST as the auxiliary dataset poses challenges in accurately estimating the probability distribution of batch training samples. This difficulty accounts for the marginal decrease in InsAcc. However, since CIFAR-10 and CINIC-10 exhibit similarities, there is no difference in InsAcc. The outcomes of these experiments illustrate that if an attacker initiates an attack during the early training stages, having an auxiliary dataset with an identical distribution to the training dataset may not be essential.

3.7 Chapter Summary

This study explores potential threats of Federated Learning from the perspective of label recovery. We theoretically derive relationships between gradients and labels in various classification tasks, identifying the root cause of label recovery from gradients

in FL. Building on our key observation that positive (or negative) samples within a class have approximate probability distributions, we propose an analytical method to recover batch labels from estimated posterior probabilities. Extensive experiments on diverse datasets and models demonstrate the effectiveness and universality of this attack. For future work, we consider designing a defense mechanism to mitigate label leakage, inspired from our theoretical analysis.

Chapter 4

Building Gradient Bridges: Label Leakage from Restricted Gradient Sharing in Federated Learning

4.1 Introduction

Federated Learning (FL) is a privacy-preserving technology that enables multiple clients to collaboratively train a machine learning model without sharing their private data [44, 6]. In each communication round, the FL server sends the global model to each client, who uses their private data to compute gradients w.r.t.[1] the model parameters and then transmits these gradients back to the server. The server aggregates the contributions from all clients and updates the global model for the next round of training. This collaborative process is repeated until the training loss of the global model converges to a satisfactory state.

Despite its privacy-enhancing nature, recent studies [67, 69, 82] have shown that the

[1] w.r.t. stands for with respect to.

shared gradients in FL provide a novel attack surface for privacy leakage. In particular, adversaries can reveal the label distribution of a client's private training data by inspecting the gradients [9, 19, 64, 83]. This label leakage can lead to more severe privacy threats, such as data reconstruction attacks [88, 21] that recover the training data of the victim client, membership inference attacks [65] that reveal whether a training instance belongs to the victim client's dataset, and attribute inference attacks [45] that infer the values of sensitive attributes or properties of the victim client's training data.

To mitigate the above label leakage, the majority of existing defenses focus on encrypting the gradients with Homomorphic Encryption [55, 79] or protecting the gradients with Differential Privacy [26, 71]. However, these defenses either incur huge computational overhead or degrade the performance of the training model. To balance the trade-off between privacy and utility, recent studies have proposed dedicated lightweight defense strategies to protect critical gradients. These strategies include homomorphically encrypting the gradients in the last fully connected (FC) layer [60], removing the bias parameters from the FC layers [57], and locally updating the FC layers during FL training process [46]. From the perspective of raw label information contained in the gradients, these lightweight defenses aim to restrict gradient sharing, especially within the final FC layer, which is the most vulnerable layer for leakage of label distributions. Although these defense strategies appear to provide privacy without compromising utility, we argue that they offer a false sense of security.

To this end, we propose a novel label restoration attack, named Gradient Bridge (GDBR), to restore the label distribution from the limited gradient information in FL. The key insight behind GDBR is to construct a bridge between the obtained gradients and the training labels. By exploring the relationship between the layer-wise gradients and the model parameters, we first trace the flow of gradients in the bottom layers of the model. Specifically, the gradient w.r.t. the input features of each layer can be recursively deduced from the starting layer at the bottom. Then, we

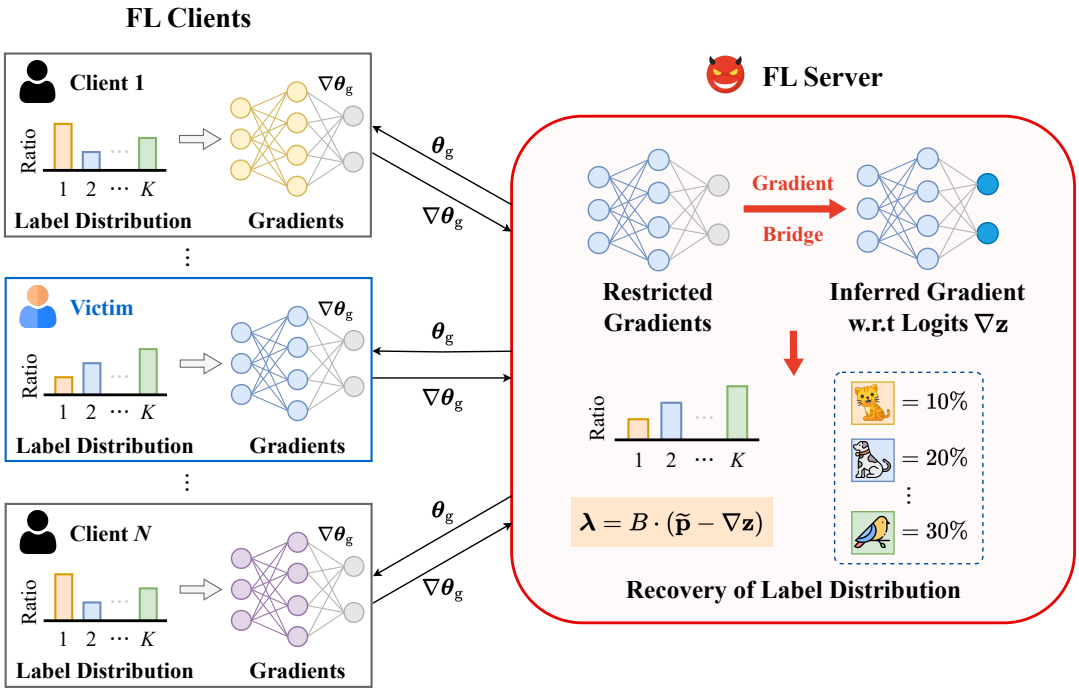


Figure 4.1: Illustration of our Gradient Bridge (GDBR) attack. The gray parts in gradients denote the unshared information, while the colored parts depict the shared gradients. GDBR first infers the gradient w.r.t. output logits (in sky blue) from the obtained gradients and then recovers the label distribution of the victim client from the derived equation (in orange).

derive the gradient w.r.t. the logits in a batch-averaged form, which contains the label information. Finally, we present the formulation of label restoration by associating the derived gradient with the observable variables. We conduct extensive experiments on various image datasets and classification models to validate the effectiveness and robustness of our GDBR attack. The experiment results show that more than 80% of the labels in the batch training data can be accurately recovered, regardless of the model architectures, dataset complexities, batch sizes, and class distributions. Even if the shared gradients are pruned or perturbed, GDBR can still recover the label distribution with acceptable degradation in performance. Fig. 4.1 illustrates our proposed GDBR attack. Our main contributions are summarized as follows:

- We propose a novel label recovery attack, called Gradient Bridge (GDBR), designed to recover the label distribution from the restricted gradient information in FL systems.
- We build a series of bridges between the shared gradients and the target labels by tracing the flow of gradients in the bottom layers, and derive the equations for label restoration.
- Extensive experiments show that GDBR is effective and robust in restoring the label distribution of private training data. Moreover, GDBR can still recover a certain amount of labels even when additional defenses are applied.

4.2 Related Work

In this section, we introduce the related works on gradient inversion attacks, analytical label recovery attacks, and representative lightweight defense strategies in FL.

4.2.1 Gradient Inversion Attacks

Gradient inversion attacks [82] are designed to recover private training data by exploiting the transmitted gradients in FL. Zhu et al. [88] propose the DLG attack, which iteratively optimizes dummy data and labels to minimize the discrepancy between the generated gradients and the ground-truth gradients. Geiping et al. [18] enhance this attack by using cosine similarity as the loss function and introducing total variance regularization to improve the quality of the reconstructed images. Yin et al. [78] devise group consistency regularization to better restore the locations of main objects in images. Zhu and Blaschko [87] present a recursive-based attack that recovers the input features of each layer by solving a system of linear equations. Furthermore, recent studies have also investigated inverting the gradients in other collaboration

tasks, such as natural language processing [20, 2] and speech recognition [10].

4.2.2 Analytical Label Recovery

In addition to data leakage via solving optimization problems, label recovery attacks are also proposed to effectively infer the label distribution, such as the percentages of each class or the critical attributes of the training data.

Zhao et al. [84] introduce iDLG, which discovers the correlation between the gradient of the last FC layer and the one-hot label. Yin et al. [78] propose GI to recover the labels by selecting the rows with the minimum values in the gradient of the last FC layer. Dang et al. [9] present RLG to distinguish the target class by separating the columns of the gradient w.r.t. weights in the output layer. Geng et al. [19] propose ZLG to estimate the posterior probabilities and extracted features of the last layer from auxiliary data to restore class-wise labels. Wainakh et al. [64] design LLG by utilizing both the direction and magnitude of the gradient w.r.t. weights in the FC layer to determine the overall impact factor and class-wise offset, and then restore the labels one by one. Ma et al. [42] devise iLRG to derive the embedding features and Softmax probabilities of the output layer and solve linear equations for label recovery.

However, all these attacks rely on the gradients of the final FC layer. Specifically, iDLG, GI and RLG can only recover non-repeating labels of the training data, while iRLG requires the gradient w.r.t. the bias term for feature restoration.

4.2.3 Lightweight Defense Strategies

To preserve label privacy, several dedicated but lightweight defense strategies have been proposed to mitigate the leakage of private labels from uploaded gradients in FL. Sotthiwat et al. [60] propose Partially Encrypted Multi-Party Computation (PE-MPC) to encrypt the gradients in the FC layer closest to the data source. This

ensures that the gradients in the output layer are neither accessible nor observable to the adversary. For some attacks that exploit the bias gradient to restore data or labels, such as [42, 55, 17], Scheliga et al. [57] present to remove the bias term from all the FC layers to entirely avoid such kind of leakage. Additionally, Mei et al. [46] propose the personalized FL training method FedVF, which allows each participant to locally update the parameters of the classification layers. FedVF seeks to balance the model performance and privacy protection by preventing the attacker from accessing the critical gradients in these layers.

4.3 Preliminary

In this section, we introduce the preliminaries of this work, including the inference of a single one-hot label, the gradients in typical layers, and our threat model in FL.

4.3.1 Inference of Single One-Hot Label

In a classification task with C classes, the cross-entropy loss function $\mathcal{L}$ is widely used to measure the distance between the predicted Softmax probabilities $\mathbf{p} \in \mathbb{R}^C$ and the target one-hot labels $\mathbf{y} \in \mathbb{R}^C$. The cross-entropy loss is defined as:

$$\mathcal{L} = -\sum_{i=1}^{C} \mathrm{y}_i \log(\mathrm{p}_i) = -\log \frac{\exp(\mathrm{z}_c)}{\sum_{j=1}^{C} \exp(\mathrm{z}_j)},$$

where c is the index of the true class, z_i is the i-th element of the output logits $\mathbf{z}$, and $\mathrm{p}_i = \frac{\exp(\mathrm{z}_i)}{\sum_{j=1}^{C} \exp(\mathrm{z}_j)}$.

In the backward propagation, the gradient of the loss function $\mathcal{L}$ w.r.t. the output logits z_i can be formulated as:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathrm{z}_i} &= -\frac{\partial \log \exp(\mathrm{z}_c)}{\partial \mathrm{z}_i} + \frac{\partial \log \sum_{j=1}^{C} \exp(\mathrm{z}_j)}{\partial \mathrm{z}_i} \\ &= -\delta_{ic} + \frac{\exp(\mathrm{z}_i)}{\sum_{j=1}^{C} \exp(\mathrm{z}_j)},\end{aligned}$$

where δ_{ic} is the Kronecker delta function, which is equal to 1 if $i = c$ and 0 otherwise. It is obvious to observe that $y_i = \delta_{ic}$, and δ_{ic} just denotes the one-hot label. Therefore, the gradient w.r.t. the output logits $\mathbf{z}$ can be rewritten as:

$$\nabla \mathbf{z} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} = \mathbf{p} - \mathbf{y}. \tag{4.1}$$

The attacker can infer the one-hot label $\mathbf{y}$ by exploiting the gradient $\nabla \mathbf{z}$ and the estimated probabilities $\mathbf{p}$.

4.3.2 Gradients in Typical Layers

Here, we introduce the gradients in typical layers, including the fully connected (FC) layer, the convolutional (Conv) layer, and the ReLU activation layer. For simplicity, we will ignore the bias items in the following discussion.

FC Layer

The forward propagation of an FC layer can be expressed as $\mathbf{z} = \mathbf{W}\mathbf{x}$, where $\mathbf{z} \in \mathbb{R}^N$ is the output, $\mathbf{W} \in \mathbb{R}^{N \times M}$ is the weight matrix, and $\mathbf{x} \in \mathbb{R}^M$ is the input.

In the backward propagation, the gradient of the loss function $\mathcal{L}$ w.r.t. the weight matrix $\mathbf{W}$ is calculated as:

$$\nabla \mathbf{W} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{W}} = \nabla \mathbf{z} \mathbf{x}^\top, \tag{4.2}$$

where $\nabla \mathbf{z}$ is the gradient w.r.t. the layer output $\mathbf{z}$.

Similarly, the gradient of the loss function $\mathcal{L}$ w.r.t. the input feature $\mathbf{x}$ can be derived as follows:

$$\nabla \mathbf{x} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \mathbf{W}^\top \nabla \mathbf{z}. \tag{4.3}$$

Conv Layer

The weights in a Conv layer are also called kernels, which are shared across different locations within the input. The kernel is generally a 4D tensor, denoted as $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times H_{\text{k}} \times W_{\text{k}}}$, where C_{in} and C_{out} are the numbers of input and output channels, and H_{k} and W_{k} are the height and width of the kernel, respectively.

The forward pass can be represented as $\mathbf{Z} = \mathbf{W} * \mathbf{X}$, where $\mathbf{Z} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}}$ is the output, $\mathbf{X} \in \mathbb{R}^{C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$ is the input, and $*$ denotes the convolution operation. Unfolding the process of convolutional operation, a specific element $\mathrm{Z}_{k,i,j}$ in the output $\mathbf{Z}$ is calculated as follows:

$$\begin{aligned} \mathrm{Z}_{k,i,j} &= \sum_{c=1}^{C_{\text{in}}} \sum_{h=1}^{H_{\text{k}}} \sum_{w=1}^{W_{\text{k}}} \mathrm{W}_{k,c,h,w} \cdot \mathrm{X}_{c,i+h-1,j+w-1} \\ &= \sum_{c,h,w} \mathrm{W}_{k,c,h,w} \cdot \mathrm{X}_{c,i+h-1,j+w-1}, \end{aligned}$$

where k is the index of the output channel, and i and j are the indices of the height and width of the output, respectively. In addition, we use the Einstein summation convention, such as $\sum_{c,h,w}$, to simplify the notation.

In the backward pass, the gradient w.r.t. element $\mathrm{W}_{k,c,h,w}$ in the kernel $\mathbf{W}$ can be expressed as:

$$\begin{aligned} \nabla \mathrm{W}_{k,c,h,w} &= \sum_{i=1}^{H_{\text{out}}} \sum_{j=1}^{W_{\text{out}}} \nabla \mathrm{Z}_{k,i,j} \cdot \mathrm{X}_{c,i+h-1,j+w-1} \\ &= \sum_{i,j} \nabla \mathrm{Z}_{k,i,j} \cdot \mathrm{X}_{c,i+h-1,j+w-1}. \end{aligned} \tag{4.4}$$

ReLU Layer

ReLU is the most commonly used activation function in deep neural networks. ReLU activation function is defined as $\mathrm{a} = \mathrm{ReLU}(\mathrm{z}) = \max(0, \mathrm{z})$, where a is the output, z is the input, and $\max(\cdot)$ is the maximum operation.

The derivative of the output a w.r.t. the input z is given by:

$$\sigma'(\mathrm{z}) = \frac{\mathrm{da}}{\mathrm{dz}} = \begin{cases} 1, & \text{if } \mathrm{z} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

In the backward propagation, the gradient w.r.t. an element z_k in the input $\mathbf{z}$ can be deduced as:

$$\nabla \mathrm{z}_k = \frac{\partial \mathcal{L}}{\partial \mathrm{a}_k} \cdot \sigma'(\mathrm{z}_k) = \nabla \mathrm{a}_k \cdot \begin{cases} 1, & \text{if } \mathrm{z}_k > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

4.3.3 Threat Model

In this study, we assume an honest-but-curious attacker in FL, who follows the FL protocol but attempts to extract private information from the shared gradients. We mainly focus on the FedSGD [44], which is a typical FL algorithm in other inversion attacks [88, 78, 19, 64]. The attacker aims to recover the private labels of the clients' training data by analyzing the shared gradients. The attacker can launch GDBR in a white-box setting, where the attacker has full knowledge of the model architecture and parameters. Moreover, the attacker can also leverage some auxiliary information, such as testing datasets, to enhance the attack performance. In our attack scenario, the gradients of the final FC layer(s) in the model are not fully shared. The client will only transmit the first layer's gradients in the bottom layer stacking. Without loss of generality, we assume that the attacker will launch the GDBR attack at the early stage of the FL training process.

4.4 Gradient Bridge (GDBR)

In this section, we introduce the proposed Gradient Bridge (GDBR) attack against FL. We first present the overview of the GDBR. Then, we provide the theoretical foundation of GDBR, including the correlation between layer-wise gradients and the

derived gradient w.r.t. output logits. Finally, we provide the formulation for label recovery by utilizing the estimated features and Softmax probabilities.

4.4.1 Overview

Our GDBR aims to recover the labels of training data in a victim's local dataset by exploiting the gradient information shared in FL collaboration. Different from previous attacks, the adversary in GDBR cannot obtain the complete gradient information from a client's update. In particular, the gradient from the last FC layer is always unavailable to the semi-honest server in FL. In GDBR, only one layer of gradient information is required for label recovery, which could be:

- Gradients of middle layers in the MLP [56] model;
- Gradients of the first or second FC layer in the LeNet [37], AlexNet [35] or VGG [59] model;
- Gradients the last Conv layer in the ResNet [24] model, where the average pooling layer is assumed to be replaced by a Conv layer whose output size is 1×1.

Based on the gradient obtained from any of the above layers, the GDBR attack first builds the gradient bridge that connects the given layer to the output logits. Then, GDBR reconstructs the batch-averaged gradient w.r.t. the output logits. Finally, the attack infers the one-hot labels by associating the estimated probabilities with the gradient w.r.t. the logits.

4.4.2 Correlation Between Layer-wise Gradients

In the following, we explore the correlations between the gradients in typical layers, such as FC, Conv and ReLU layers, or the layer stacks. These correlations are essential

for GDBR to recover the labels from the gradients in FL.

Lemma 1. *In an FC layer, let* $\nabla\mathbf{x}$*,* $\nabla\mathbf{W}$*, and* $\nabla\mathbf{z}$ *represent the gradients w.r.t. the input* $\mathbf{x}$*, the weight* $\mathbf{W}$*, and the output* $\mathbf{z}$*, respectively. Then the following relationships hold:*

$$\nabla\mathbf{x}\mathbf{x}^\top = \mathbf{W}^\top\nabla\mathbf{W} \tag{4.6}$$

$$\nabla\mathbf{z}\mathbf{z}^\top = \nabla\mathbf{W}\mathbf{W}^\top \tag{4.7}$$

$$\nabla\mathbf{z}\odot\mathbf{z} = \mathrm{diag}(\nabla\mathbf{W}\mathbf{W}^\top), \tag{4.8}$$

where $\odot$ *denotes the element-wise product, and* $\mathrm{diag}(\cdot)$ *denotes the diagonal elements of a matrix.*

Proof. If we multiply $\mathbf{x}^\top$ on both sides of Eq. (4.3), and then substitute Eq. (4.2) into it, we can derive:

$$\nabla\mathbf{x}\mathbf{x}^\top = \mathbf{W}^\top\nabla\mathbf{z}\mathbf{x}^\top = \mathbf{W}^\top\nabla\mathbf{W}.$$

Likewise, if we multiply $\mathbf{W}^\top$ on both sides of Eq. (4.2), then we can obtain the following equation:

$$\nabla\mathbf{W}\mathbf{W}^\top = \nabla\mathbf{z}\mathbf{x}^\top\mathbf{W}^\top = \nabla\mathbf{z}\mathbf{z}^\top.$$

Since $\nabla\mathbf{z}\mathbf{z}^\top$ is a symmetric matrix, the diagonal elements of it are equal to the element-wise product of the output $\mathbf{z}$ and its gradient $\nabla\mathbf{z}$. So we also have:

$$\nabla\mathbf{z}\odot\mathbf{z} = \mathrm{diag}(\nabla\mathbf{z}\mathbf{z}^\top) = \mathrm{diag}(\nabla\mathbf{W}\mathbf{W}^\top),$$

where $\odot$ denotes the element-wise product. □

Lemma 2. *In an FC layer, given the weight matrix* $\mathbf{W}$ *and the input gradient* $\nabla\mathbf{x}$*, the output gradient* $\nabla\mathbf{z}$ *can be derived as follows:*

$$\nabla\mathbf{z} = (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}. \tag{4.9}$$

Proof. From the derived Eq. (4.6), we can individually denote the gradient $\nabla\mathbf{W}$ by multiplying the inverse of $\mathbf{W}\mathbf{W}^\top$ and $\mathbf{W}$ on both sides of the equation, that is:

$$\nabla\mathbf{W} = (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}\mathbf{x}^\top.$$

Combining with Eq. (4.2), we can transfer the role of $\mathbf{x}^\top$ by multiplying $\mathbf{x}$ and the inverse of $\mathbf{x}^\top\mathbf{x}$ on both sides of the above equation, and deduce the gradient $\nabla\mathbf{z}$ as:

$$\begin{aligned}\nabla\mathbf{z} &= (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}\mathbf{x}^\top\mathbf{x}(\mathbf{x}^\top\mathbf{x})^{-1} \\ &= (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}.\end{aligned}$$

□

Note: Lemma 1 and 2 present the relationships between the gradients w.r.t. the input, output and weight in an FC layer. Gradient $\nabla\mathbf{z}$ can be inferred by using: (1) gradient $\nabla\mathbf{x}$, or (2) gradient $\nabla\mathbf{W}$ and output $\mathbf{z}$.

Lemma 3. *In a Conv layer, let $\nabla\mathbf{W}_k$ and $\nabla\mathbf{Z}_k$ represent the gradients w.r.t. the k-th kernel $\mathbf{W}_k$ and the output $\mathbf{Z}_k$, respectively. Then the following relationship holds:*

$$\langle\nabla\mathbf{W}_k, \mathbf{W}_k\rangle_F = \begin{cases} \nabla\mathbf{Z}_k \odot \mathbf{Z}_k, & \text{if } \mathbf{Z}_k \in \mathbb{R}, \\ \langle\nabla\mathbf{Z}_k, \mathbf{Z}_k\rangle_F, & \text{otherwise}, \end{cases} \tag{4.10}$$

where $\odot$ denotes the element-wise product, and $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product that calculates the sum of the element-wise product of two tensors or matrices.

Proof. If we multiply $\mathrm{W}_{k,c,h,w}$ on both sides of Eq. (4.4), and then sum over c, h, and

w, we can derive:

$$\begin{aligned}
&\sum_{c,h,w} \nabla \mathrm{W}_{k,c,h,w} \cdot \mathrm{W}_{k,c,h,w} \\
&\quad = \sum_{c,h,w} \sum_{i,j} \nabla \mathrm{Z}_{k,i,j} \cdot \mathrm{X}_{c,h+i-1,w+j-1} \cdot \mathrm{W}_{k,c,h,w} \\
&\quad = \sum_{i,j} \nabla \mathrm{Z}_{k,i,j} \cdot \left(\sum_{c,h,w} \mathrm{W}_{k,c,h,w} \cdot \mathrm{X}_{c,i+h-1,j+w-1} \right) \\
&\quad = \sum_{i,j} \nabla \mathrm{Z}_{k,i,j} \cdot \mathrm{Z}_{k,i,j}.
\end{aligned}$$

So we can simplify the above equation as $\langle \nabla \mathbf{W}_k, \mathbf{W}_k \rangle_{\mathrm{F}} = \langle \nabla \mathbf{Z}_k, \mathbf{Z}_k \rangle_{\mathrm{F}}$. When the shape of the output $\mathbf{Z}_k$ is 1×1, that is, $\mathbf{Z}_k$ is a scalar, we can represent the result using the element-wise product $\odot$, so that $\langle \nabla \mathbf{W}_k, \mathbf{W}_k \rangle_{\mathrm{F}} = \nabla \mathbf{Z}_k \odot \mathbf{Z}_k$. □

Note: Lemma 3 provides the relationship between the gradients w.r.t. the kernel and the output in a Conv layer. When the output size is 1×1, the Frobenius inner product can be replaced by the element-wise product.

Lemma 4. *In a ReLU layer, let* $\nabla \mathbf{z}$ *and* $\nabla \mathbf{a}$ *represent the gradients w.r.t. the input* $\mathbf{z}$ *and output* $\mathbf{a}$*, respectively. Then the following relationship holds:*

$$\nabla \mathbf{z} \odot \mathbf{z} = \nabla \mathbf{a} \odot \mathbf{a}, \tag{4.11}$$

where $\odot$ *denotes the element-wise product, and* $\nabla \mathbf{Z} \odot \mathbf{Z} = \nabla \mathbf{A} \odot \mathbf{A}$ *holds for both* $\mathbf{Z}$ *and* $\mathbf{A}$ *being matrices or tensors.*

Proof. By multiplying z_k on both sides of Eq. (4.5), for each element k, we can deduce the following relationship:

$$\nabla \mathrm{z}_k \cdot \mathrm{z}_k = \nabla \mathrm{a}_k \cdot \max(0, \mathrm{z}_k) = \nabla \mathrm{a}_k \cdot \mathrm{a}_k.$$

The relationship can be easily extended to the entire vector $\mathbf{z}$, and we have $\nabla \mathbf{z} \odot \mathbf{z} = \nabla \mathbf{a} \odot \mathbf{a}$ for each element in this ReLU layer. It is the same for the matrix or tensor case. □

Theorem 3. *In a single stack of an FC layer followed by a ReLU activation (FC-ReLU), the gradient* $\nabla\mathbf{a}$ *can be inferred from the following relationships:*

$$\nabla\mathbf{a} = \begin{cases} (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}, & \text{if } \nabla\mathbf{x} \text{ is given,} \\ \operatorname{diag}(\nabla\mathbf{W}\mathbf{W}^\top) \oslash \mathbf{a}, & \text{if } \nabla\mathbf{W} \text{ is given,} \end{cases} \tag{4.12}$$

where $\oslash$ *denotes the element-wise division, and all elements in* $\mathbf{a}$ *are assumed to be non-zero in the second case.*

Proof. Combining Lemma 2 and Lemma 4, we can establish the connection between the activated output gradient $\nabla\mathbf{a}$ and the input gradient $\nabla\mathbf{x}$ in the FC-ReLU stack as follows:

$$\nabla\mathbf{a} = \nabla\mathbf{z} \odot \mathbf{z} \oslash \mathbf{a} = (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x} \odot \mathbf{z} \oslash \mathbf{a}.$$

Since we have assumed that all elements in $\mathbf{a}$ are non-zero, it is obvious that $\mathbf{a} = \mathbf{z}$ holds from the definition of ReLU activation. In this case, $\mathbf{z} \oslash \mathbf{a}$ can be simplified as $\mathbf{1}$, where all the elements in $\mathbf{1}$ are equal to 1. Therefore, we can simplify the above equation, that is:

$$\nabla\mathbf{a} = (\mathbf{W}\mathbf{W}^\top)^{-1}\mathbf{W}\nabla\mathbf{x}.$$

Combining Eq. (4.8) and Eq. (4.11), there is another way to infer the gradient $\nabla\mathbf{a}$ when the gradient $\nabla\mathbf{W}$ and the output $\mathbf{a}$ are given. The derivation of $\nabla\mathbf{a}$ can be expressed as:

$$\nabla\mathbf{a} = \operatorname{diag}(\nabla\mathbf{W}\mathbf{W}^\top) \oslash \mathbf{a},$$

where all elements in $\mathbf{a}$ are assumed to be non-zero. □

Note: Theorem 3 illustrates two ways to derive the input gradient $\nabla\mathbf{a}$ in a single FC-ReLU stack. The gradient $\nabla\mathbf{a}$ can be either deduced from (1) the gradient $\nabla\mathbf{x}$, or (2) the gradient $\nabla\mathbf{W}$ and the activation $\mathbf{a}$.

Theorem 4. *In a single stack of a Conv layer followed by a ReLU activation (Conv-ReLU), the output gradient $\nabla\mathbf{A} \in \mathbb{R}^N$ can be inferred from the following equations:*

$$\begin{cases} \nabla\mathbf{A}_k &= \langle \nabla\mathbf{W}_k, \mathbf{W}_k \rangle_F \oslash \mathbf{A}_k, \\ \nabla\mathbf{A} &= [\nabla\mathbf{A}_1, \nabla\mathbf{A}_2, \cdots, \nabla\mathbf{A}_K]^\top, \end{cases} \tag{4.13}$$

where $\oslash$ denotes the element-wise division, and each activated output $\mathbf{A}_k$ is assumed to be non-zero.

Proof. According to Lemma 3 and Lemma 4, we have $\nabla\mathbf{Z} \odot \mathbf{Z} = \nabla\mathbf{A} \odot \mathbf{A}$ and $\nabla\mathbf{Z} \odot \mathbf{Z} = \langle \nabla\mathbf{W}, \mathbf{W} \rangle_{\mathrm{F}}$ in this Conv-ReLU stack, where $\mathbf{Z}$ and $\mathbf{A}$ are the outputs of the Conv layer and the ReLU layer, respectively. We mainly consider that the shape of $\mathbf{A}_k$ is 1×1, that is, $\mathbf{A}_k \in \mathbb{R}$ and $\nabla\mathbf{A} \in \mathbb{R}^N$, where N is the number of kernels in the Conv layer. Thus, for the k-th kernel, we can derive the gradient $\nabla\mathbf{A}_k$ as:

$$\nabla\mathbf{A}_k = \langle \nabla\mathbf{W}_k, \mathbf{W}_k \rangle_{\mathrm{F}} \oslash \mathbf{A}_k,$$

where $\mathbf{A}_k$ is assumed to be non-zero. Finally, we can deduce the gradient $\nabla\mathbf{A}$ by concatenating all the $\nabla\mathbf{A}_k$ together. □

Note: When $\mathbf{A} \in \mathbb{R}^N$, Theorem 4 presents the gradient $\nabla\mathbf{A}$ in a single Conv-ReLU stack, which can be inferred by using the gradient $\nabla\mathbf{W}$ and the activation $\mathbf{A}$ (all elements are supposed to be non-zero).

In deep neural networks, the bottom layers of a classification model are usually composed of Conv-ReLU-FC-ReLU stacks or FC-ReLU stacks. We consider a more general structure that L layers of such stacks are cascaded in the bottom layers. The first stack is an FC-ReLU or Conv-ReLU, and the last stack is an FC layer without the ReLU activation. The other stacks are all FC-ReLU. We use superscripts $[l]$ to denote the layer index, where $l \in \{1, 2, \cdots, L\}$. This structure is common in the LeNet, AlexNet, VGG, and ResNet models. In summary, the layers that we analyze include:

- 1 FC-ReLU or Conv-ReLU stack ($l = 1$);
- $(L-2)$ FC-ReLU stacks ($l \in \{2, 3, \cdots, L-1\}$);
- 1 FC output layer without ReLU activation ($l = L$).

Combining the previous Theorem 3 and Theorem 4, we can infer the gradient $\nabla \mathbf{A}_k^{[1]}$ in the first stack as follows:

$$\nabla \mathbf{a}_k^{[1]} = \begin{cases} \operatorname{diag}(\nabla \mathbf{W}_k^{[1]} \mathbf{W}_k^{[1]\top}) \oslash \mathbf{a}_k^{[1]}, & \text{if FC,} \\ \langle \nabla \mathbf{W}_k^{[1]}, \mathbf{W}_k^{[1]} \rangle_{\mathrm{F}} \oslash \mathbf{a}_k^{[1]}, & \text{if Conv,} \end{cases} \tag{4.14}$$

where $\nabla \mathbf{W}_k^{[1]}$ can be directly obtained from the shared gradients in FL. In the design of GDBR, all we need is the gradient $\nabla \mathbf{W}_k^{[1]}$ in the first stack of the bottom layers.

For the l-th layer, $l \in \{2, 3, \cdots, L-1\}$, we can recursively deduce the gradients $\nabla \mathbf{a}^{[l]}$ and $\nabla \mathbf{x}^{[l+1]}$ from Theorem 3 as:

$$\begin{cases} \nabla \mathbf{x}^{[2]} = \nabla \mathbf{a}^{[1]} &= [\nabla \mathbf{a}_1^{[1]}, \nabla \mathbf{a}_2^{[1]}, \cdots, \nabla \mathbf{a}_K^{[1]}]^\top \\ \nabla \mathbf{x}^{[3]} = \nabla \mathbf{a}^{[2]} &= (\mathbf{W}^{[2]} \mathbf{W}^{[2]\top})^{-1} \mathbf{W}^{[2]} \nabla \mathbf{x}^{[2]} \\ &\vdots \\ \nabla \mathbf{x}^{[l+1]} = \nabla \mathbf{a}^{[l]} &= (\mathbf{W}^{[l]} \mathbf{W}^{[l]\top})^{-1} \mathbf{W}^{[l]} \nabla \mathbf{x}^{[l]}. \end{cases} \tag{4.15}$$

Finally, according to Lemma 2, we can derive the gradient $\nabla \mathbf{z}^{[L]}$ in the last FC layer as the following equation:

$$\nabla \mathbf{z}^{[L]} = (\mathbf{W}^{[L]} \mathbf{W}^{[L]\top})^{-1} \mathbf{W}^{[L]} \nabla \mathbf{x}^{[L]}. \tag{4.16}$$

We name the above derivations as the *Gradient Bridge* that connects the gradient w.r.t. the first FC or Conv layer to the gradient w.r.t. the output logits. In this scenario, only the gradient $\nabla \mathbf{W}^{[1]}$ of the first stack in the bottom layers and the global model $\boldsymbol{\theta}_{\mathrm{g}}$ are required for inferring the gradient $\nabla \mathbf{z}^{[L]}$ of the output logits. In

FL collaboration, the model weights $\mathbf{W}^{[l]}$ in each layer are known to the server, and the activated features $\mathbf{a}^{[1]}$ can also be estimated by fitting some auxiliary data into the model. Our finding of the gradient bridge breaks the limitation of the restricted gradient sharing in FL, which is the foundation of the GDBR attack.

Note: Given the gradient $\nabla\mathbf{W}^{[1]}$ of the first FC or Conv in the bottom layers, the Gradient Bridge can be built to infer the gradient $\nabla\mathbf{z}^{[L]}$ w.r.t. the output logits of the global model.

4.4.3 Derivation of Batch-Averaged Gradients

Assumption 1. *We assume that the activation features $\mathbf{a}^{[1]}$ in the first stack of the bottom layers exhibit similar Gaussian distributions across different dimensions for various training samples. Specifically, the features $\mathbf{a}^{[1](n)}$ for the n-th sample in a batch of B can be approximated by some auxiliary data, denoted as $\widetilde{\mathbf{a}}^{[1]}$. Consequently, we have:*

$$\widetilde{\mathbf{a}}^{[1]} \approx \mathbf{a}^{[1](n)}, \quad \forall n \in \{1, 2, \cdots, B\}.$$

Combining Eq. (4.14) and Assumption 1, we can deduce the batch-averaged gradient $\overline{\nabla\mathbf{a}}^{[1]}$ by utilizing $\widetilde{\mathbf{a}}^{[1]}$ to approximate the features $\mathbf{a}^{[1](n)}$. So the averaged gradient $\overline{\nabla\mathbf{a}}^{[1]}$ is given by the following equations:

$$\begin{aligned}
\overline{\nabla\mathbf{a}_k}^{[1]} &= \frac{1}{B}\sum_{n=1}^{B}\nabla\mathbf{a}_k^{[1](n)} \\
&\approx \begin{cases} \left(\frac{1}{B}\sum_n \operatorname{diag}(\nabla\mathbf{W}_k^{[1](n)}\mathbf{W}_k^{[1]\top})\right) \oslash \widetilde{\mathbf{a}}_k^{[1]} \\ \left(\frac{1}{B}\sum_n \langle\nabla\mathbf{W}_k^{[1](n)}, \mathbf{W}_k^{[1]}\rangle_{\mathrm{F}}\right) \oslash \widetilde{\mathbf{a}}_k^{[1]} \end{cases} \\
&= \begin{cases} \operatorname{diag}(\overline{\nabla\mathbf{W}}_k^{[1]}\mathbf{W}_k^{[1]\top}) \oslash \widetilde{\mathbf{a}}_k^{[1]}, & \text{if FC}, \\ \langle\overline{\nabla\mathbf{W}}_k^{[1]}, \mathbf{W}_k^{[1]}\rangle_{\mathrm{F}} \oslash \widetilde{\mathbf{a}}_k^{[1]}, & \text{if Conv}. \end{cases}
\end{aligned}$$

For the l-th layer ($l \in \{2, 3, \cdots, L-1\}$), the batch-averaged gradient $\overline{\nabla\mathbf{a}}^{[l]}$ can be

obtained as follows:

$$\begin{aligned}\overline{\nabla \mathbf{a}}^{[l]} &= \frac{1}{B}\sum_{n=1}^{B} \nabla \mathbf{a}^{[l](n)} \\ &= (\mathbf{W}^{[l]}\mathbf{W}^{[l]\top})^{-1}\mathbf{W}^{[l]}\left(\frac{1}{B}\sum_{n=1}^{B}\nabla \mathbf{x}^{[l](n)}\right) \\ &= (\mathbf{W}^{[l]}\mathbf{W}^{[l]\top})^{-1}\mathbf{W}^{[l]}\overline{\nabla \mathbf{x}}^{[l]}.\end{aligned}$$

Similarly, the batch-averaged gradient w.r.t. the logits $\mathbf{z}^{[L]}$ in the output layer can be derived in the same way as above.

In summary, for the batch-averaged setting, the gradients for each stack of the bottom layers can be recursively inferred using the following equations:

$$\overline{\nabla \mathbf{a}}_k^{[1]} \approx \begin{cases} \operatorname{diag}(\overline{\nabla \mathbf{W}}_k^{[1]}\mathbf{W}_k^{[1]\top}) \oslash \widetilde{\mathbf{a}}_k^{[1]}, & \text{if FC,} \\ \langle \overline{\nabla \mathbf{W}}_k^{[1]}, \mathbf{W}_k^{[1]} \rangle_{\mathrm{F}} \oslash \widetilde{\mathbf{a}}_k^{[1]}, & \text{if Conv.} \end{cases} \tag{4.17}$$

$$\begin{cases} \overline{\nabla \mathbf{x}}^{[2]} = \overline{\nabla \mathbf{a}}^{[1]} &= [\overline{\nabla \mathbf{a}}_1^{[1]}, \overline{\nabla \mathbf{a}}_2^{[1]}, \cdots, \overline{\nabla \mathbf{a}}_K^{[1]}]^\top \\ \overline{\nabla \mathbf{x}}^{[3]} = \overline{\nabla \mathbf{a}}^{[2]} &= (\mathbf{W}^{[2]}\mathbf{W}^{[2]\top})^{-1}\mathbf{W}^{[2]}\overline{\nabla \mathbf{x}}^{[2]} \\ &\vdots \\ \overline{\nabla \mathbf{x}}^{[l+1]} = \overline{\nabla \mathbf{a}}^{[l]} &= (\mathbf{W}^{[l]}\mathbf{W}^{[l]\top})^{-1}\mathbf{W}^{[l]}\overline{\nabla \mathbf{x}}^{[l]}. \end{cases} \tag{4.18}$$

$$\overline{\nabla \mathbf{z}}^{[L]} = (\mathbf{W}^{[L]}\mathbf{W}^{[L]\top})^{-1}\mathbf{W}^{[L]}\overline{\nabla \mathbf{x}}^{[L]}. \tag{4.19}$$

4.4.4 Label Recovery from Inferred Gradients

Assumption 2. *We assume that the Softmax probabilities* $\mathbf{p}$ *of the model output exhibit similar Gaussian distributions across different classes for various training samples. Specifically, the probabilities* $\mathbf{p}^{(n)}$ *for the n-th sample in a batch of size B can*

be approximated by some auxiliary data, denoted as $\widetilde{\mathbf{p}}$*. Consequently, we have:*

$$\widetilde{\mathbf{p}} \approx \mathbf{p}^{(n)}, \quad \forall n \in \{1, 2, \cdots, B\}.$$

Combining Eq. (4.1) in Preliminary, the batch-averaged gradient $\overline{\nabla \mathbf{z}}^{[L]}$ can be represented as:

$$\overline{\nabla \mathbf{z}}^{[L]} = \frac{1}{B} \sum_{n=1}^{B} \nabla \mathbf{z}^{[L](n)} = \frac{1}{B} \sum_{n=1}^{B} \left(\mathbf{p}^{(n)} - \mathbf{y}^{(n)} \right). \tag{4.20}$$

In the batch-averaged setting, we use $\boldsymbol{\lambda}$ to denote the sum of the one-hot labels for the batch of B samples. Each element in $\boldsymbol{\lambda}$ is an integer, which represents the number of samples that have the corresponding class label in the batch. By substituting $\boldsymbol{\lambda}$ into Eq. (4.20) and use $\widetilde{\mathbf{p}}$ in Assumption 2 to approximate the probabilities $\mathbf{p}^{(n)}$, we can derive $\boldsymbol{\lambda}$ as:

$$\boldsymbol{\lambda} = \sum_{i=1}^{B} \mathbf{y}^{(i)} = B \cdot \left(\widetilde{\mathbf{p}} - \overline{\nabla \mathbf{z}}^{[L]} \right). \tag{4.21}$$

Finally, we can infer the number of samples for each class in the batch by solving $\boldsymbol{\lambda}$ in Eq. (4.21). Gradient $\overline{\nabla \mathbf{z}}^{[L]}$ is the output of the gradient bridge, which has been provided in Eq. (4.19).

4.5 Experiments

In this section, we conduct extensive experiments on various datasets and models to evaluate the effectiveness of GDBR in recovering labels. We first validate the reasonability of the assumptions in Section 4.4. Then, we compare the performance with prior baseline methods and analyze the influence of different factors on GDBR, such as batch sizes, different layers, class distributions, etc. Finally, we present the performance of GDBR against two defense mechanisms, which are gradient pruning and noise perturbation.

4.5.1 Experimental Setups

Datasets and Models

We evaluate the GDBR attack on five benchmark datasets and five popular neural network models. Here are the details of the datasets and models.

- MNIST [37] contains 60k training and 10k testing gray-scale digit images of size 28×28 with 10 classes.
- SVHN [49] contains 73k training and 26k testing color digit images of size 32×32 with 10 classes.
- CIFAR-10/100 [34] contain 50k training and 10k testing color images of size 32×32 with 10/100 classes.
- ImageNet-1K [12] contains 1.28M training and 50k validation color images of size 224×224 with 1,000 classes.

We employ five neural network models with different architectures, including MLP (Multi-Layer Perceptron) [56], LeNet [37], AlexNet [35], VGG series [59], and ResNet series [24]. In particular, the MLP model consists of 6 FC layers with [2048, 1024, 512, 256, 128, 64] hidden units. All the above models are implemented using the PyTorch framework [54].

Implementation Details

We consider an FL system with N participants, where one of them is designated as the victim. Each client trains the model using its private dataset with the FedSGD algorithm [44]. During each training iteration, a client randomly selects B samples from its dataset to create a mini-batch, with the default batch size set to 64.

For the ResNet series models, we replace the average pooling layer with a convolutional layer. We use PyTorch's default initialization for the layers preceding the average pooling. For the bottom layers, we initialize the weights using a uniform distribution between 0.01 and 0.2, and we ignore the bias terms for simplicity. By default, we use the uniform distribution to initialize the weights of the bottom layers, and we provide the gradient of the penultimate layer to GDBR.

To estimate the input features of the penultimate layer and the output probabilities of the model, we utilize the validation or testing dataset as an auxiliary dataset. We sample a total of 1,000 samples from the auxiliary data, distributing them evenly across all the classes. To eliminate experimental randomness, we repeat each experiment 20 times and report the average result.

All the experiments are conducted using PyTorch 2.0.0 and CUDA 12.2 on a workstation equipped with an Intel i9-10900K CPU @ 3.70GHz, 64GB of RAM, and an NVIDIA GeForce RTX 4090 (24GB) GPU.

Baselines and Evaluation Metrics

We compare our proposed GDBR with two state-of-the-art methods, ZLG [19] and LLG [64]. ZLG leverages the relationship between the labels and the gradient $\nabla \mathbf{W}^{[L]}$ in the final FC layer to recover labels, while LLG exploits the direction and magnitude of $\nabla \mathbf{W}^{[L]}$ to restore the labels of the batch samples. However, gradient $\nabla \mathbf{W}^{[L]}$ is not directly accessible in our attack scenario. For a fair comparison, we provide part of the ground-truth gradient $\nabla \mathbf{W}^{[L]}$ to both ZLG and LLG for label recovery. Moreover, we provide the same subset of auxiliary data to GDBR and the baselines to estimate the features of the penultimate layer and the output probabilities of the model.

To quantify the performance of label restoration, we adopt two metrics: *Instance-level Accuracy* (InsAcc) and *Class-level Accuracy* (ClsAcc) [83]. InsAcc evaluates the accuracy of recovering the ground-truth labels of individual data samples, while

ClsAcc assesses the accuracy of recovering the ground-truth classes to which the data samples belong. All the results are reported as percentages (%).

4.5.2 Verification of Assumptions

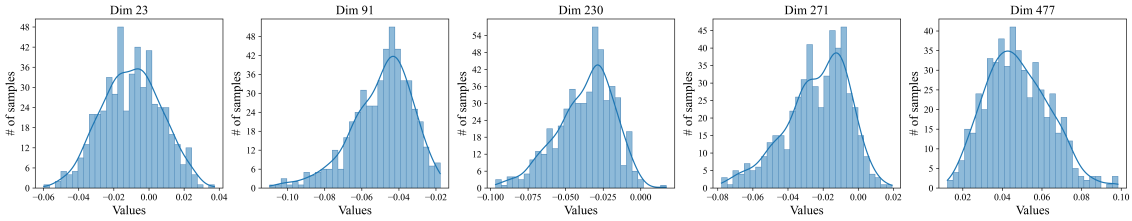
We first verify the rationalization of Assumption 1 and 2 by conducting experiments on the ResNet18 model trained on the CIFAR-100 dataset. We select five random dimensions within input features and five 10-fold classes to visualize the distributions of the features and the probabilities in the final FC layer.

The findings, shown in Fig. 4.2, indicate that at the early stage of training, the extracted features and the output probabilities of different samples exhibit similar distributions. This observation supports the use of auxiliary data to estimate the hidden features and the output probabilities of the model, which are essential for GDBR to build the gradient bridge.

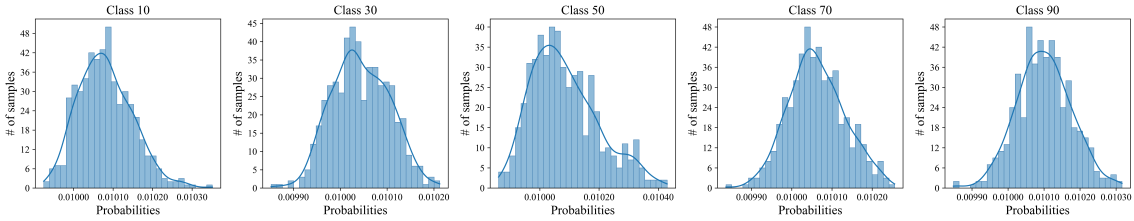
4.5.3 Comparison with Baselines

Next, we compare GDBR with the baselines across various combinations of datasets and models. The model-dataset pairs are as follows: LeNet on MNIST, AlexNet on SVHN, VGG11 on CIFAR-10, and ResNet18 on CIFAR-100. We set the batch sizes ranging from 2 to 512, with the batch data randomly sampled from a subset of all classes in the training datasets. To provide the weight gradient in the last layer to ZLG and LLG, we shuffle the column elements of the matrix.

The results, displayed in Fig. 4.3 and Fig. 4.4, show that GDBR outperforms ZLG and LLG on both InsAcc and ClsAcc metrics in almost all cases. Even with access to part of the ground-truth gradient, ZLG and LLG still struggle to recover the labels of the target batch. In contrast, GDBR can effectively recover the labels with high accuracy, demonstrating the effectiveness of our method.

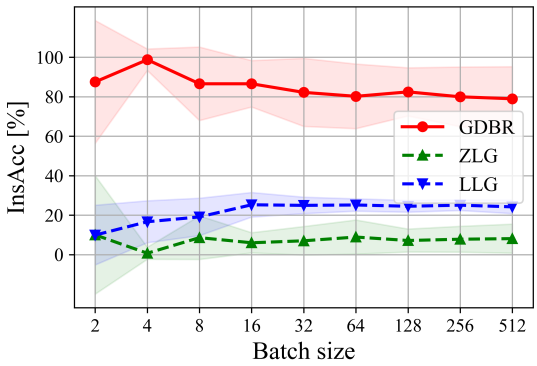


(a) Input feature distributions of the FC layer in ResNet18 ($B = 512$). The shown dimensions are randomly selected, and the values in each dimension approximately follow a Gaussian distribution whose mean can be estimated from similar training samples.

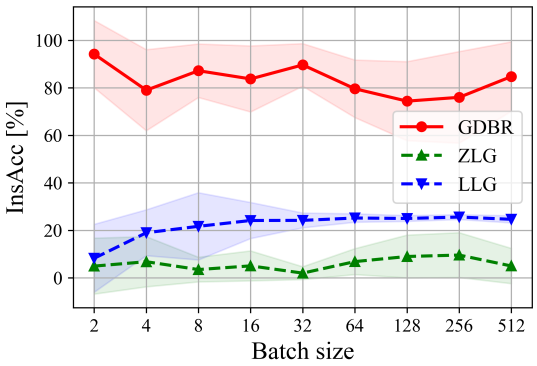


(b) Softmax probability distributions of ResNet18 model ($B = 512$). Five 10-fold classes are chosen for illustration, and the probabilities of different classes exhibit similar Gaussian distributions, with mean values close to 0.01, across various training samples.

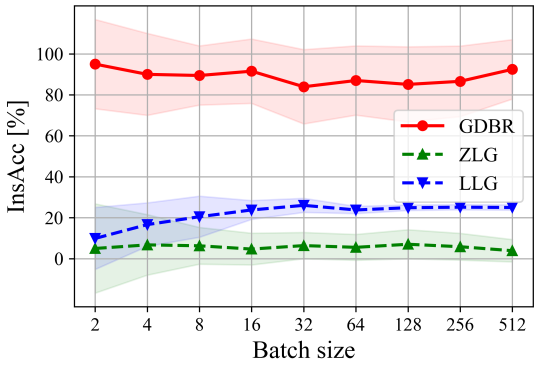
Figure 4.2: The distributions of input features and output probabilities of the FC layer in ResNet18, which is trained on the CIFAR-100 dataset. The x-axis represents the values of features or probabilities, and the y-axis represents the number of samples.



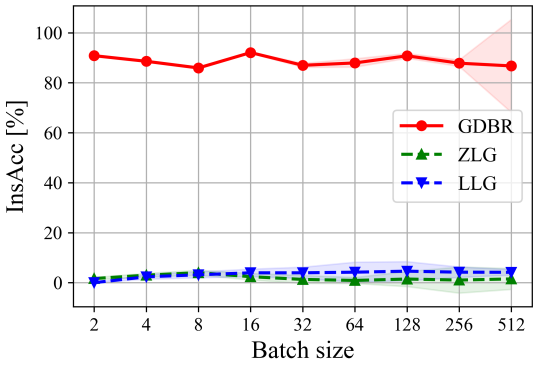
(a) InsAcc on MNIST.



(b) InsAcc on SVHN.

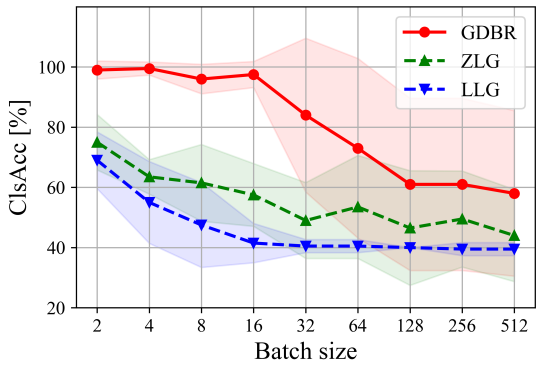


(c) InsAcc on CIFAR-10.

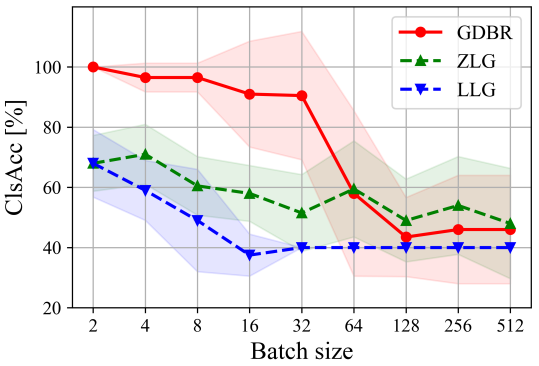


(d) InsAcc on CIFAR-100.

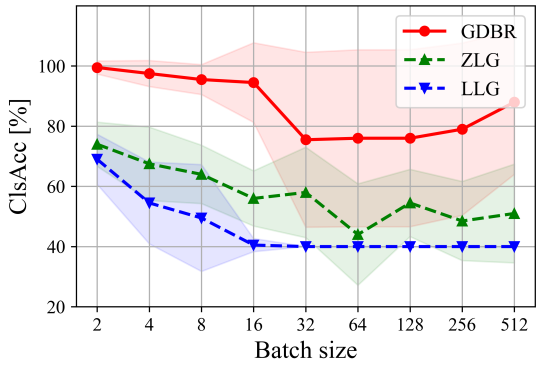
Figure 4.3: Comparison of GDBR with baselines on InsAcc across different datasets and batch sizes. The experiments are performed on four model-dataset pairs: LeNet on MNIST, AlexNet on SVHN, VGG11 on CIFAR-10, and ResNet18 on CIFAR-100, with batch sizes ranging from 2 to 512. The batch data is randomly sampled from a subset of training classes.



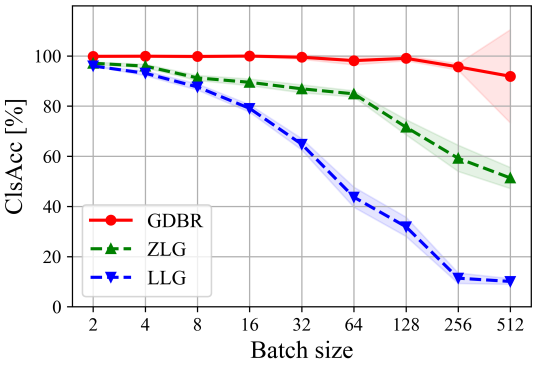
(a) ClsAcc on MNIST.



(b) ClsAcc on SVHN.



(c) ClsAcc on CIFAR-10.



(d) ClsAcc on CIFAR-100.

Figure 4.4: Comparison of GDBR with baselines on ClsAcc across different datasets and batch sizes. The experiments are performed on four model-dataset pairs: LeNet on MNIST, AlexNet on SVHN, VGG11 on CIFAR-10, and ResNet18 on CIFAR-100, with batch sizes ranging from 2 to 512. The batch data is randomly sampled from a subset of training classes.

4.5.4 Analysis of Different Factors

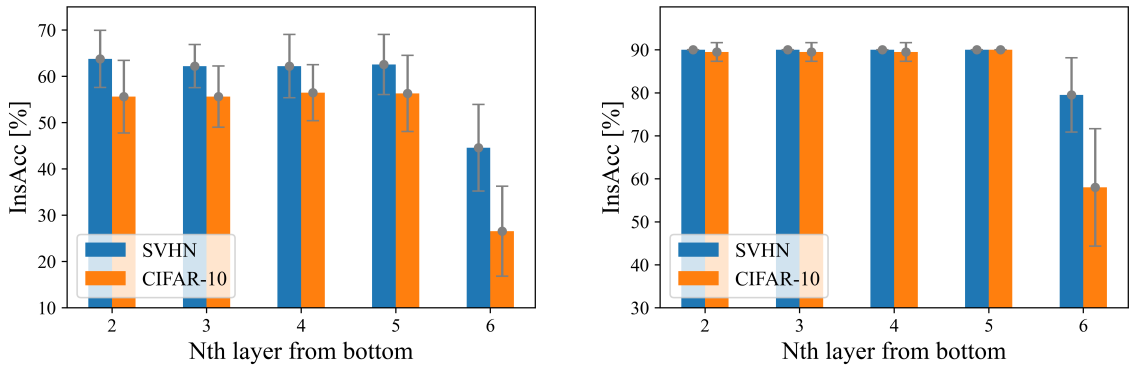
In this subsection, we investigate the influence of various factors on the performance of GDBR. These factors include the gradients from different layers, the class distribution of the batch data, the use of auxiliary data or dummy data, and the gradient simulation and model initialization modes.

Effect of Gradients from Different Layers

We investigate the influence of gradients from different layers provided to GDBR. The experiments are conducted on a 6-layer MLP model using the SVHN and CIFAR-10 datasets, with the batch data randomly sampled from the training datasets. As shown in Fig. 4.5, the gradient of the first layer (6th from bottom) has the worst results. This is because the features of the first layer are more scattered and directly affected by the training data, making it harder to estimate and build an accurate bridge to the output logits. Additionally, due to the more complex nature of CIFAR-10 compared to SVHN, GDBR's performance on CIFAR-10 is inferior to that on SVHN. The gradients from other layers exhibit similar performance, indicating the robustness of GDBR when applied to different layers.

Effect of Batch Class Distribution

We next examine the impact of different batch class distributions on the performance of GDBR. We evaluate GDBR with five different batch class distributions: *random*, *uniform*, *single*, *subclassed*, and *imbalanced*, and report the InsAcc. The experiments are conducted on different datasets and models, and the results are shown in Table 4.1. The findings indicate that GDBR performs well in all settings. Although the performance is slightly affected by the data distribution, GDBR consistently achieves high accuracy in recovering the labels of the target batch data. Notably, batch data



(a) InsAcc on 6-layer MLP.

(b) ClsAcc on 6-layer MLP.

Figure 4.5: Comparison of utilized gradients from different layers in a 6-layer MLP model, trained on MNIST and CIFAR-10.

containing only one class yields better results than mixed classes, likely because the variables are more concentrated and easier to estimate. These analyses underscore the robustness of GDBR against different data distributions.

Effect of Auxiliary Data vs. Dummy Data

We further compare the effect of using auxiliary data versus dummy data in the GDBR. As mentioned in the Implementation Details, we use the validation or testing dataset as auxiliary data, selecting 1,000 samples evenly distributed across all classes. For dummy data, we generate the data from a standard Gaussian distribution. The evaluation results are shown in Table 4.2. For simple datasets like MNIST, GDBR can achieve even higher InsAcc with dummy data. However, for the other color image datasets, GDBR performs better with auxiliary data. The auxiliary data provides more accurate information to estimate the extracted features and the output probabilities, helping GDBR recover the labels more accurately. Nonetheless, when auxiliary data is unavailable, GDBR can still achieve comparable performance by exploiting dummy data. We also find that the initialized training model lacks feature

Table 4.1: Comparison of different class distributions within the batch training data across various datasets and models.

Dataset	**Model**	**InsAcc** [%]				
		R	U	S	SC	IM
MNIST	LeNet	80.7	83.0	86.3	85.0	93.5
SVHN	AlexNet	85.6	81.6	81.7	85.2	90.1
CIFAR-10	VGG11	84.1	81.2	89.6	90.0	97.0
CIFAR-100	ResNet18	89.8	88.5	94.3	87.9	90.4
ImageNet	ResNet50	95.2	97.0	98.8	90.0	91.0

*R: random, U: uniform, S: single, SC: subclassed, IM: imbalanced.

extraction capabilities, resulting in similar embeddings for different samples in the feature space and making it easier to recover the labels.

Effect of Gradient Simulation and Model Initialization

Then, we investigate the impact of gradient simulation modes and model initialization modes on the effectiveness of GDBR. The simulation methods are proposed to provide a portion of the ground-truth gradient to ZLG and LLG. Specifically, we consider the following simulation methods.

- *Stats*: Using the statistics of the ground-truth gradient to generate similar distributions for simulation.

- *Stats (col)*: Using the column-wise statistics of the gradient to generate similar distributions for each column.

Table 4.2: Comparison of label recovery using auxiliary data vs. dummy data across various datasets and models.

Dataset	**Model**	**InsAcc** [%]		**ClsAcc** [%]	
		Aux	Dummy	Aux	Dummy
MNIST	LeNet	$81. \pm 5.$	$83. \pm 6.$	$98. \pm 4.$	$95. \pm 5.$
SVHN	AlexNet	$86. \pm 5.$	$80. \pm 4.$	$99. \pm 3.$	$98. \pm 4.$
CIFAR-10	VGG11	$84. \pm 6.$	$85. \pm 7.$	$97. \pm 5.$	$99. \pm 2.$
CIFAR-100	ResNet18	$90. \pm 4.$	$85. \pm 3.$	$98. \pm 2.$	$97. \pm 2.$
ImageNet	ResNet50	$95. \pm 2.$	$92. \pm 2.$	$99. \pm 1.$	$98. \pm 2.$

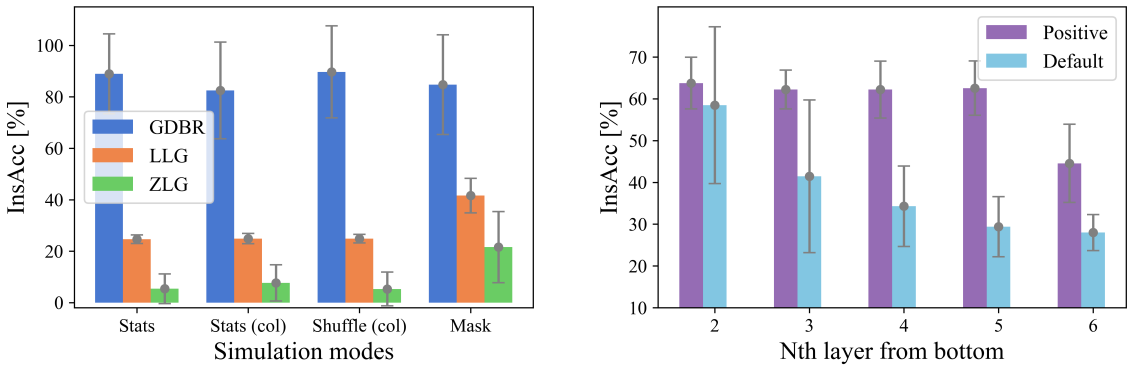
*Aux: auxiliary data, Dummy: dummy data.

- *Shuffle (col)*: Shuffling the column-wise elements of the ground-truth gradient to simulate the observed gradient.

- *Mask*: Masking 50% of the gradient and replacing with mean values of the masked elements for simulation.

As shown in Fig. 4.6a, although the simulation of *Mask* has slightly better performance, GDBR outperforms the baselines with any of the gradient simulation methods.

We also compare the influence of different model initialization modes on GDBR. As introduced in the Implementation Details, we use a uniform distribution between 0.01 and 0.2 to initialize the weights of the bottom layers. This modification ensures that all the features for estimation are positive. The default initialization of PyTorch adopts Kaiming uniform [23], which uses values between the negative and positive bounds. In addition, we replace zero values in the features with the mean of the non-zero elements. We use a 6-layer MLP model trained on SVHN to compare the



(a) Gradient simulation modes.

(b) Model initialization modes.

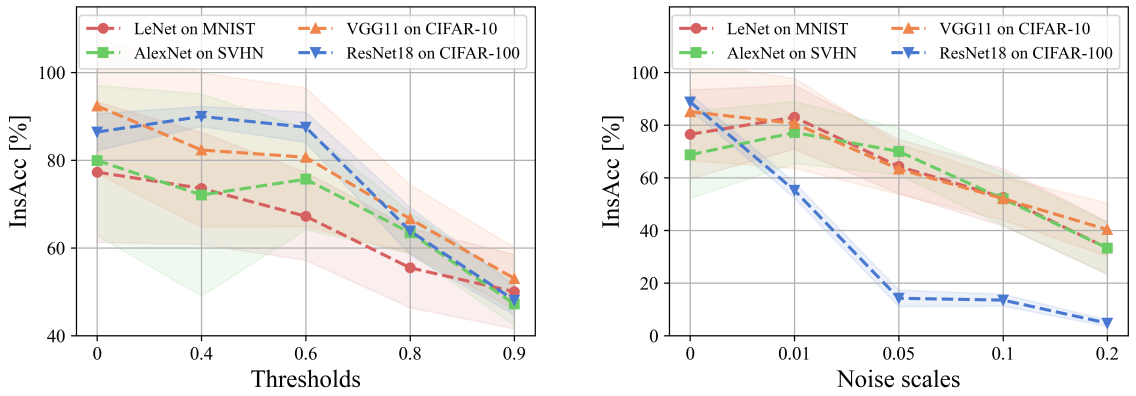
Figure 4.6: Comparison of gradient simulation modes for baselines and model initialization modes. Results for gradient simulation modes are obtained using a VGG11 model trained on CIFAR-10. Model initialization mode experiments are conducted with a 6-layer MLP trained on SVHN.

performance of GDBR with these two initialization modes, utilizing gradients from different layers. The results, shown in Fig. 4.6b, indicate that GDBR achieves higher accuracy with the positive initialization. However, the performance gap is acceptable, suggesting that GDBR is robust to different initialization modes.

4.5.5 Performance against Defense Mechanisms

Finally, we evaluate the performance of GDBR against two typical defense mechanisms: gradient pruning and noise perturbation. We compare the performance of GDBR with different gradient pruning thresholds in the range of [0, 0.9] and noise scales in the range of [0, 0.2]. The experiments are conducted on various datasets and models, and the results are shown in Fig. 4.7.

The results demonstrate that the performance of GDBR deteriorates significantly under a high pruning threshold (e.g., $\geq$0.9) or a high noise scale (e.g., $\geq$0.2). Notably, the Resnet18 model trained on CIFAR-100 is more sensitive to noise than the other



(a) Gradient pruning ([0, 0.9]).

(b) Noise perturbation ([0, 0.2]).

Figure 4.7: Comparison of gradient pruning and noise perturbation across various groups of datasets and models. The threshold of gradient pruning ranges from 0 to 0.9, and the scale of noise perturbation ranges from 0 to 0.2.

models. This increased sensitivity is caused by the accumulation of noise in deducing the gradient w.r.t. $\mathbf{a}_k$ in the penultimate layer. According to $\nabla \mathbf{a}_k = \langle \nabla \mathbf{W}_k, \mathbf{W}_k \rangle_{\mathrm{F}} \oslash \mathbf{a}_k$, the noise in $\nabla \mathbf{W}_k$ is amplified by the Frobenius inner product. Overall, GDBR can still recover the labels with high accuracy under moderate defense mechanisms, highlighting the necessity of developing more robust defense strategies.

4.6 Chapter Summary

In this study, we propose GDBR, a label recovery attack that reconstructs the label distribution of the victim client's private training data from the limited gradient information shared in FL. By exploring the correlation between the gradients and the model parameters, GDBR builds a bridge between the shared gradients and the target labels and presents the formulation for label restoration. Extensive experiments demonstrate that GDBR can accurately recover at least 80% of the class-wise batch labels in different FL settings. However, GDBR mainly focuses on the early training

stage of FL collaboration. In the future, we plan to investigate the effectiveness of GDBR in the trained models and explore effective defenses against GDBR.

Chapter 5

GradFilt: Class-wise Targeted Data Reconstruction from Gradients in Federated Learning

5.1 Introduction

Federated Learning (FL) is a distributed machine learning paradigm that allows multiple clients to collaboratively train a global model without sharing their data [44]. In FL, clients train the shared model with its local data and then send the gradient of the model parameters to the server for aggregation. In this way, FL enables the model training without collecting private data, which has been applied in privacy-sensitive applications such as finance and healthcare [77, 41, 75].

However, recent studies have proposed Gradient Inversion Attacks (GIAs) to restore the training data and labels from the shared gradients [88, 26, 82]. Typically, the role of the attacker is performed by the FL server who is curious about the client's private data. The attacker can recover the training data by optimizing the dummy data to minimize the distance between the dummy and ground truth gradients [88, 18].

From a data recovery perspective, existing GIAs can be broadly classified into two categories. The first category focuses on passively reconstructing the entire training data in the batch from gradients by adding regularization terms or using generative models [18, 28]. However, the reconstruction performance is limited and often results in blurry, distorted images. In contrast, the second category strives to restore high-quality instances by proactively modifying the model parameters or structures [17, 4]. However, the recovery is stochastic, and the attacker cannot control which instances can be recovered. From an attacker's viewpoint, who aims to maximize the attack benefits and efficiency, the preference lies in reconstructing only the sensitive data of interested classes, as it holds greater value than other samples.

In this work, we propose Gradient Filtering (GradFilt), a targeted attack to reconstruct the training data of sensitive or desired classes from gradients. Our insight is to filter out the gradients of non-target data and retain the gradients of the target data in the mini-batch. GradFilt includes four main phases: *Gradient Separation*, *Label Restoration*, *Gradient Calibration*, and *Data Reconstruction*.

We first modify the parameters of the final fully connected (FC) layer to create a filter that preserves only the gradients of the target classes. Then, we restore the sum of one-hot labels of the target data from the batch-averaged gradients. After that, we calibrate the gradients in the final FC layer and scale the gradients in the preceding layers. Finally, according to the number of target data in the batch, we either solve an optimization problem to reconstruct the target data or analytically recover the data by inserting an FC layer in the first layer.

We evaluate GradFilt on various image datasets and classification models, covering datasets of different complexities and models of varying depths. It is worth noting that GradFilt can achieve 100% success rates in restoring the labels of the target mini-batch. The experimental results demonstrate that GradFilt can effectively reconstruct the desired data with higher accuracies compared to the existing GIAs. Our contributions are summarized as follows:

- We propose GradFilt to reconstruct the training data of the target classes from shared gradients in FL. The key insight is to filter out the gradients of non-target data and retain those of target data in the mini-batch.
- Specifically, GradFilt consists of four core functionalities: gradient separation, label restoration, gradient calibration, and data reconstruction. By integrating these functionalities, GradFilt can effectively reconstruct the target data.
- We evaluate GradFilt on various image datasets and classification models. The results show that GradFilt can restore the batch labels with 100% success rates and reconstruct the target data with higher accuracies than existing GIAs.

5.2 Related Work and Motivation

The majority of GIAs are designed for honest-but-curious FL scenarios, where the server is curious about the client's private data. In this context, the attacker strictly follows the FL protocol and only exploit the shared gradients to recover the training data. Current research in this area focuses on improving the quality of the recovered data by using cosine similarity as the loss function[18], introducing total variation as regularization[18, 78], leveraging a generative model pre-trained on the prior distribution [28, 2]. In addition, [87, 31] leverage the relationship between model parameters and gradients to efficiently and analytically reconstruct the training data. However, the performances of these attacks are limited when facing larger batch sizes or deeper model architectures. Fowl et al. [17] point out that [78] can only reconstruct 28% of training data when applied on a batch size of 48 for a ResNet50 model.

Moreover, recent studies have proposed GIAs for malicious adversaries in FL, where the attacker can arbitrarily modify the model parameters or structures to effectively recover the training data. Fowl et al. [17] propose a stochastic attack by exploiting the relationship between the gradients and input features of FC layers, which randomly

recover the training samples by adding an FC layer next to the input. Boenisch et al. [4] devise trap weights in FC layers as half positive and negative, with slightly larger negative magnitudes. Wen et al. [72] modify the parameters of FC layers to amplify gradients associated with specific samples. Pasquini et al. [53] propose a method to circumvent secure aggregation mechanisms by distributing deliberately crafted and inconsistent model parameters to participants during a training round. Zhao et al. [85] introduce Loki, an attack method capable of simultaneously targeting multiple clients and leaking identifiable data directly related to them, even in the presence of both FedAvg algorithm and secure aggregation protocols.

However, these data reconstruction attacks are inherently stochastic, meaning they rely on random processes and probabilistic methods. As a result, they lack precision and control over the specific samples that can be successfully recovered. This unpredictability poses a significant limitation, as attackers cannot selectively target or guarantee the retrieval of particular data points, leading to data that yields low utility or cannot be effectively used for further downstream attacks.

To address these limitations, we propose GradFilt to reconstruct the training data of desired classes from gradients. Unlike existing stochastic methods, GradFilt offers precise control over the reconstruction process, enabling attackers to selectively focus on specific classes of interest. This targeted approach not only enhances the utility of the recovered data but also increases its applicability for further downstream attacks, thereby overcoming the unpredictability and inefficiency of previous techniques.

5.3 Preliminaries

In this section, we introduce the preliminaries of this work, including the classification tasks in FL, the gradients with respect to the input features and output logits in the last FC layer, the batch-averaged gradients, and our proposed threat model.

5.3.1 Classification Tasks in FL

In FL, we concentrate on image classification tasks. Clients collaboratively train a shared machine learning model, denoted as $\boldsymbol{\theta}$. This global model consists of L layers: the first $L-1$ layers extract features, and the final layer is a fully connected (FC) layer that classifies the features. The raw output of the model is the logits $\boldsymbol{z} \in \mathbb{R}^C$, where C is the number of classes. These logits are then transformed into Softmax probabilities, $\boldsymbol{p} = \text{Softmax}(\boldsymbol{z}) \in \mathbb{R}^C$. The category with the highest probability in these Softmax probabilities determines the model's prediction. We use the cross-entropy (CE) loss function to measure the difference between the predicted probabilities $\boldsymbol{p}$ and the ground truth one-hot label $\boldsymbol{y} \in \mathbb{R}^C$, which is defined as:

$$\mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{i=1}^{C} y_i \log p_i = -\log p_c, \tag{5.1}$$

where c is the index of the ground truth class.

5.3.2 Gradients of the Final FC Layer

According to the definition of the Softmax function, the probability of the c-th class is given by $p_c = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}}$. The CE loss can be rewritten as:

$$\begin{aligned} \mathcal{L}_{\text{CE}}(\boldsymbol{p}, \boldsymbol{y}) &= -\log \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}} \\ &= -z_c + \log \sum_{j=1}^{C} e^{z_j}. \end{aligned} \tag{5.2}$$

The gradient of the CE loss with respect to the i-th logits, i.e., z_i, is given by:

$$\begin{aligned} \nabla z_i = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_i} &= -\frac{\partial z_c}{\partial z_i} + \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \\ &= -\mathbb{I}(i = c) + p_i \\ &= \begin{cases} -1 + p_i, & \text{if } i = c, \\ p_i, & \text{otherwise,} \end{cases} \end{aligned} \tag{5.3}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function that outputs 1 if the condition is true and 0 otherwise. Therefore, we can derive the gradient with respect to logits $\boldsymbol{z}$ as:

$$\nabla \boldsymbol{z} = \boldsymbol{p} - \boldsymbol{y}. \tag{5.4}$$

In the final FC layer, the forward propagation process can be represented as $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{C \times M}$ is the weight matrix, $\boldsymbol{a} \in \mathbb{R}^{M}$ is the input features, and $\boldsymbol{b} \in \mathbb{R}^{C}$ is the bias vector. To calculate the gradient of features $\boldsymbol{a}$ and weights $\boldsymbol{W}$, we follow the chain rule and deduce the gradients as follows:

$$\nabla \boldsymbol{a} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \boldsymbol{a}} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{a}} = \boldsymbol{W}^{\top} \nabla \boldsymbol{z}. \tag{5.5}$$

$$\nabla \boldsymbol{W} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \boldsymbol{W}} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = \nabla \boldsymbol{z} \boldsymbol{a}^{\top}. \tag{5.6}$$

5.3.3 Batch-averaged Gradients

In this study, we focus on the FedSGD algorithm [44] in FL systems. Each client u trains the global model $\boldsymbol{\theta}$ on its local dataset during each communication round r. After training, the client then sends its batch-averaged gradients, $\overline{\nabla \boldsymbol{\theta}}_{r,u}$, to the server. The batch-averaged gradient is calculated as:

$$\overline{\nabla \boldsymbol{\theta}}_{r,u} = \frac{1}{B} \sum_{n=1}^{B} \nabla \boldsymbol{\theta}_{r,u}^{(n)}, \tag{5.7}$$

where B is the batch size, and $\nabla \boldsymbol{\theta}_{r,u}^{(n)}$ is the gradient computed on the n-th sample in the mini-batch. The server then aggregates the gradients from all the clients to update the global model for the next round $r+1$. To safeguard against gradient poisoning attacks, the server performs an inspection of each client's updated gradient before aggregation. For simplicity, we will omit the round index r and the client index u in the following discussion, i.e., $\overline{\nabla \boldsymbol{\theta}} = \frac{1}{B} \sum_{n=1}^{B} \nabla \boldsymbol{\theta}^{(n)}$, as the attack can be launched on any client at any communication round.

5.3.4 Threat Model

We assume a *malicious* FL server that seeks to recover sensitive information from the participating clients. Specifically, we model the server's objective as recovering private training data belonging to a *target class*[1], denoted by t, where $t \in \{1, 2, \cdots, C\}$. We refer to data points associated with this target class as *target data* and denote them as $\boldsymbol{x}_t$. Conversely, the remaining classes constitute the *non-target classes*, represented by o, where $o = \{1, 2, \cdots, C\} \backslash t$. Data samples corresponding to non-target classes are referred to as *non-target data* and denoted as $\boldsymbol{x}_o$.

The malicious server possesses comprehensive knowledge about the FL system. It not only understands the meaning behind each data category – for example, in CIFAR-10 dataset [34], it knows that class 1 represents "airplane" and class 2 corresponds to "automobile" – but also has access to the shared model's architecture and parameters. This intimate understanding grants the adversary the power to directly modify the model, a crucial capability for achieving its objective: manipulating gradient updates to extract sensitive information. This modification selectively nullifies the gradients associated with non-target data, denoted as $\nabla\boldsymbol{\theta}'(\boldsymbol{x}_o)$. The isolation of gradients with respect to target data, $\nabla\boldsymbol{\theta}'(\boldsymbol{x}_t)$, enables their independent analysis. Therefore, the batch-averaged gradient $\overline{\nabla\boldsymbol{\theta}'}$ is transformed into a pure representation of the target data's gradients, which can be represented using the following equation:

$$\overline{\nabla\boldsymbol{\theta}'} = \frac{1}{B}\left(\sum_{i=1}^{|\boldsymbol{x}_t|}\nabla\boldsymbol{\theta}'^{(i)} + \sum_{j=1}^{|\boldsymbol{x}_o|}\nabla\boldsymbol{\theta}'^{(j)}\right) = \frac{1}{B}\sum_{i=1}^{|\boldsymbol{x}_t|}\nabla\boldsymbol{\theta}'^{(i)}, \tag{5.8}$$

where $|\boldsymbol{x}_t|$ and $|\boldsymbol{x}_o|$ are the numbers of target and non-target data in the batch.

Then, the adversary can either invert the manipulated gradients $\overline{\nabla\boldsymbol{\theta}'}$ to reconstruct the target data $\boldsymbol{x}_t$ by solving an optimization problem, or analytically restore some random samples by exploiting the characteristics of an FC layer.

[1]Without loss of generality, we consider the attack targets a single class (i.e., $|t| = 1$). It is easy to extend this scenario to multiple target classes, where $t \subseteq \{1, 2, \cdots, C\}$.

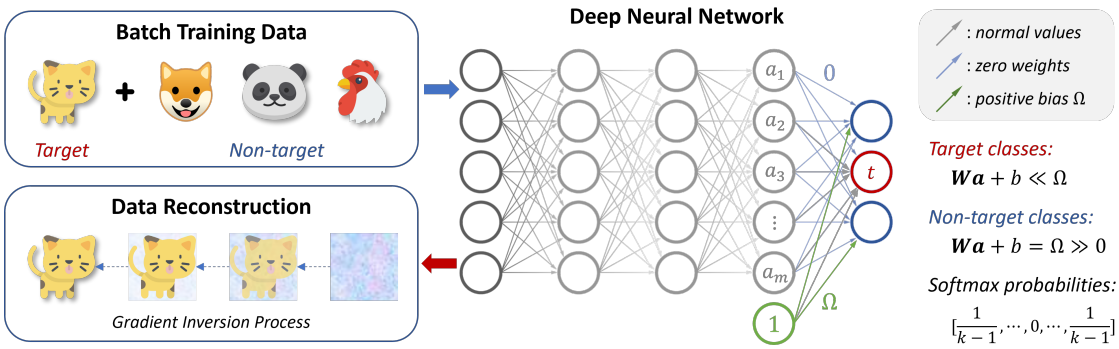


Figure 5.1: Illustration of our proposed GradFilt attack. According to the purpose of GradFilt, the adversary aims to recover the target data from its interested classes. The adversary modifies the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ of the final FC layer to embed a filter within the model, which retains the gradients of the target class and nullifies the gradients of non-target classes. Based on the preserved gradients associated with the target data, the adversary then reconstructs the original data through gradient inversion. In the final layer, the weights of non-target classes are set to zero, and the biases of these classes are assigned large positive values. The weights and biases of the target class are retained and scaled by a factor α.

5.4 Methodology of GradFilt

In this section, we introduce our GradFilt attack. We first present the overview of this targeted attack, and then delve into the details of the several main phases: *Gradient Separation*, *Label Restoration*, *Gradient Calibration* and *Data Reconstruction*. We provide a comprehensive explanation of the attack's workflow and the mechanism of model manipulation. The workflow of GradFilt is illustrated in Fig. 5.1 and the algorithm is summarized in Algorithm 1.

5.4.1 Overview of GradFilt

Building upon the threat model presented in Section 5.3.4, we introduce GradFilt, a method designed to reconstruct the training data of the target class determined by the adversary in FL. The attack leverages carefully manipulated model parameters, denoted as $\boldsymbol{\theta}'$, which form a modified version of the original global model $\boldsymbol{\theta}$. GradFilt operates in the following four main phases and is summarized in Algorithm 1.

1. **Gradient Separation**: The parameters of the final FC layer are strategically modified to create a "filter" that retains the gradients associated with the target class while effectively nullifying the gradients calculated from non-target classes. This manipulation allows for the isolation of gradients with respect to the target data within the batch-averaged gradients.

2. **Label Restoration**: Combining the batch-averaged bias gradients $\overline{\nabla \boldsymbol{b}'}$ with the manipulated probabilities $\boldsymbol{p}'$, the adversary can restore the sum of one-hot labels (denoted as $\boldsymbol{\lambda}$) of the target data. These recovered labels can not only be utilized to reveal the number of target data in the batch but also reveal the distribution of the important data across the classes.

3. **Gradient Calibration**: The gradients in the final FC layer are mixed with both target data and non-target data, and the gradients from previous layers are averaged across the entire batch. This calibration process allows for the extraction of the target gradients in the final layer and scales the gradients in the preceding $L-1$ layers accordingly.

4. **Data Reconstruction**: The filtered gradients, now enriched with information solely from the target data, are then inverted either formulating an optimization problem to reconstruct the original target data or analytically recovering the data by inserting an FC layer in the first layer of the model.

Algorithm 1 Algorithm of GradFilt attack.

Input: Global model $\boldsymbol{\theta}$, target classes t, non-target classes o, number of classes C, scale α, threshold β

Output: Reconstructed target data $\boldsymbol{x}_t^*$

Gradient Separation:

In the final fully connected (FC) layer, set all rows of $\boldsymbol{W}$ corresponding to classes o to zero and assign large values Ω to the indices of o in $\boldsymbol{b}$

Scale the row of $\boldsymbol{W}$ corresponding to class t by a factor α

Send the manipulated model $\boldsymbol{\theta}'$ to the participating client

Label Restoration:

Receive the batch-averaged gradients $\overline{\nabla \boldsymbol{\theta}'}$ from the client

Derive the manipulated probabilities $\boldsymbol{p}' = [\frac{1}{C-1}, \cdots, 0, \cdots, \frac{1}{C-1}]^\top$

Restore the sum of one-hot labels $\boldsymbol{\lambda}$ from $\boldsymbol{p}'$ and $\overline{\nabla \boldsymbol{b}'}$ using Eq. (5.14)

Gradient Calibration:

Determine the number of target data in the batch: $|\boldsymbol{x}_t| = \sum_{i \in t} \boldsymbol{\lambda}_i$

Extract the batch-averaged gradients $\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)$ and $\overline{\nabla \boldsymbol{W}'}(\boldsymbol{x}_t)$ for target data $\boldsymbol{x}_t$ in the final FC layer using Eq. (5.17) and Eq. (5.18)

Scale gradient $\overline{\nabla \boldsymbol{\theta}'}(\boldsymbol{x}_t)$ in the previous $L-1$ layers by $\frac{B}{|\boldsymbol{x}_t|}$ using Eq. (5.22)

Data Reconstruction:

if $|\boldsymbol{x}_t| \leq \beta$:

    Solve the optimization problem in Eq. (5.23) to reconstruct $\boldsymbol{x}_t^*$

else:

    Insert an FC layer in the first layer to analytically recover $\boldsymbol{x}_t^*$

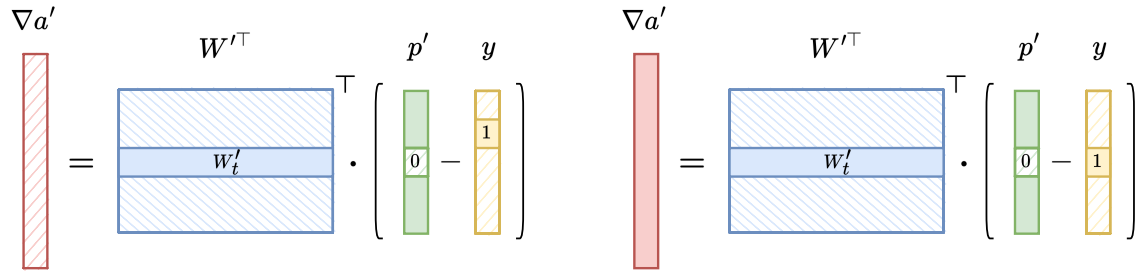
return $\boldsymbol{x}_t^*$

5.4.2 Gradient Separation

Control of Gradient $\nabla \boldsymbol{a}^{[L]}$

To differentiate the gradient of target data $\boldsymbol{x}_t$ from non-target data $\boldsymbol{x}_o$, we propose to control the gradient with respect to the input features $\nabla \boldsymbol{a}^{[L]}$ in the last FC layer. According to the chain rule and backpropagation, gradients for model parameters $\boldsymbol{\theta}$ are calculated progressively from the output layer to the input. Setting $\nabla \boldsymbol{a}^{[L]}$ to zero can effectively halt gradient flow through previous layers, essentially vanishing their gradients. Conversely, preserving $\nabla \boldsymbol{a}^{[L]}$ ensures that gradients propagate through all layers of the model during backpropagation.

This insight forms the basis of our method, GradFilt: selectively nullifying gradients for non-target data while preserving those of target data in batch-averaged gradients. In essence, we aim to achieve $\nabla \boldsymbol{a}^{[L]}(\boldsymbol{x}_o) = \mathbf{0}$ for non-target data and retain $\nabla \boldsymbol{a}^{[L]}(\boldsymbol{x}_t)$ for target data. We achieve this by manipulating the parameters of the last FC layer, i.e., the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$.



(a) Gradient $\nabla \boldsymbol{a}'$ of non-target data.

(b) Gradient $\nabla \boldsymbol{a}'$ of target data.

Figure 5.2: Illustration of controlling gradient $\nabla \boldsymbol{a}'$ in the final FC layer by manipulating the weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$. The light-colored rectangles with dashed lines represent zero values, while the dark-colored rectangles indicate normal non-zero elements. The key insight is to make the zero rows within $\boldsymbol{W}'$ and $\boldsymbol{p}'$ complementary to each other, ensuring that only the gradients of the target class are preserved.

Parameter Manipulation

Before explaining how to adjust the parameters of the global model, we first examine the calculation of $\nabla \boldsymbol{a}^{[L]}$ in the last layer for a single data point. For simplicity, we will omit the layer index L in the following discussion. By combining Eq. (5.4) and Eq. (5.5), we can derive the gradient of the input features $\nabla \boldsymbol{a}$ as:

$$\nabla \boldsymbol{a} = \boldsymbol{W}^\top \nabla \boldsymbol{z} = \boldsymbol{W}^\top (\boldsymbol{p} - \boldsymbol{y}). \tag{5.9}$$

Despite the uncertainty in the one-hot label $\boldsymbol{y}$ within Eq. (5.9), we can strategically control the gradient $\nabla \boldsymbol{a}$ by manipulating both the weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$. The manipulation of bias parameters directly influences the calculated probabilities $\boldsymbol{p}$. For example, a large positive value assigned to a specific bias b_k will significantly increase the corresponding probability, making it close to 1. Fig. 5.2 visually illustrates this gradient manipulation for both non-target and target data. The light-colored rectangles with dashed lines represent zero values, while the dark-colored rectangles indicate normal non-zero elements.

Specifically, we set the rows of $\boldsymbol{W}'$ corresponding to non-target classes to zero: $\boldsymbol{W}'_k = \boldsymbol{0}$ $(k \notin t)$. Conversely, the values for the row pointing to the target class are retained. Simultaneously, we influence the Softmax probabilities by assigning large bias values $\boldsymbol{b}'_k = \Omega \gg 0$ to all non-target classes $(k \notin t)$, and force their probabilities to zero. This complementary manipulation effectively zeros out the gradient $\nabla \boldsymbol{a}$ except when the one-hot label corresponds to the target class, at which point $\nabla \boldsymbol{\theta}'$ is non-zero.

Additionally, an important benefit of this modification is that it maximizes the distance between the manipulated Softmax probabilities $\boldsymbol{p}'$ and the one-hot label $\boldsymbol{y}$ for target data. As illustrated in Fig. 5.2, the difference $\boldsymbol{p}'_t - \boldsymbol{y}_t$ reaches its maximum value, i.e., -1, for the target samples, highlighting this divergence. To further amplify the magnitude of the gradient $\nabla \boldsymbol{a}'$, we introduce a scaling factor α $(\alpha \geq 1)$ to the rows of $\boldsymbol{W}'$ corresponding to the target class. This adjustable factor provides fine-grained

control over the gradient's magnitude, ultimately influencing the effectiveness of data reconstruction during the subsequent inversion process. Consequently, the manipulated gradient $\nabla \boldsymbol{a}'$ can be expressed as:

$$\nabla \boldsymbol{a}' = -\alpha \boldsymbol{W}_t'^{\top}, \tag{5.10}$$

where $\boldsymbol{W}_t'$ denotes the row of $\boldsymbol{W}'$ corresponding to the target class t. In the context of batch-averaged settings, the averaged gradient $\overline{\nabla \boldsymbol{a}'}$ can be calculated as:

$$\overline{\nabla \boldsymbol{a}'} = \frac{1}{B} \sum_{i=1}^{|\boldsymbol{x}_t|} \nabla \boldsymbol{a}'^{(i)} = -\frac{\alpha |\boldsymbol{x}_t|}{B} \boldsymbol{W}_t'^{\top}. \tag{5.11}$$

where $|\boldsymbol{x}_t|$ is the number of target data in the mini-batch.

Note: We leverage the manipulated gradient $\nabla \boldsymbol{a}$ to selectively filter out gradients of non-target data $\boldsymbol{x}_o$ and retain those of target data $\boldsymbol{x}_t$. Specifically, we set the rows of the weight matrix $\boldsymbol{W}'$ corresponding to non-target classes to $\mathbf{0}$ and assign large values Ω to indices of non-target classes in $\boldsymbol{b}'$.

5.4.3 Label Restoration

From the forward propagation $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{a} + \boldsymbol{b}$, we observe that the probabilities of non-target classes converge uniformly to $\frac{1}{C-1}$ when bias values of these indices are set to large positive numbers Ω. This allows us to have the deterministic manipulated probabilities $\boldsymbol{p}'$, which can be expressed as:

$$\boldsymbol{p}' = \text{Softmax}(\boldsymbol{z}') = \left[\frac{1}{C-1}, \cdots, 0, \cdots, \frac{1}{C-1} \right]^{\top}. \tag{5.12}$$

Combining Eq. (5.4) with $\nabla \boldsymbol{b}' = \nabla \boldsymbol{z}'$, we can reconstruct the original one-hot label $\boldsymbol{y}$ from the manipulated probabilities $\boldsymbol{p}'$ as follows: $\boldsymbol{y} = \boldsymbol{p}' - \nabla \boldsymbol{b}'$. In a batch-averaged scenario, this relationship becomes:

$$\overline{\nabla \boldsymbol{b}'} = \frac{1}{B} \sum_{n=1}^{B} \nabla \boldsymbol{b}'^{(n)} = \frac{1}{B} \sum_{n=1}^{B} \left(\boldsymbol{p}^{(n)} - \boldsymbol{y}^{(n)} \right) = \boldsymbol{p}' - \frac{1}{B} \boldsymbol{\lambda}, \tag{5.13}$$

where $\boldsymbol{\lambda}$ is the sum of one-hot labels for the batch of B samples. Each element in $\boldsymbol{\lambda}$ is an integer, representing the number of samples that have the corresponding class label in the batch. Then we can directly restore the one-hot labels of the mini-batch training data from the following equation:

$$\boldsymbol{\lambda} = B \cdot (\boldsymbol{p}' - \overline{\nabla \boldsymbol{b}'}). \tag{5.14}$$

From Eq. (5.14), we can determine the number of target data in the mini-batch, i.e., $|\boldsymbol{x}_t| = \sum_{i \in t} \boldsymbol{\lambda}_i$. When there are multiple target classes, where $|t| > 1$, the probability equals $\frac{1}{C-|t|}$ for non-target classes and 0 for target classes. The above label restoration process is crucial for the subsequent gradient inversion step, ensuring that the target data points are correctly identified for reconstruction.

Note: By modifying the parameters of the final FC layer, GradFilt can restore the sum of one-hot labels $\boldsymbol{\lambda}$ for the mini-batch samples: $\boldsymbol{\lambda} = B \cdot (\boldsymbol{p}' - \overline{\nabla \boldsymbol{b}'})$. This recovery process is crucial for subsequent gradient inversion.

5.4.4 Gradient Calibration

While we have successfully isolated the gradients of target data in the first $L-1$ layers, the gradients of non-target data still contribute to the final FC. To obtain the gradients of target data, we need to extract them from the batch-averaged gradients $\overline{\nabla \boldsymbol{W}'}$ and $\overline{\nabla \boldsymbol{b}'}$ in this layer. To achieve this goal, we introduce a gradient calibration process that selectively retrieves the target data gradients.

According to Eq. (5.6), we can derive the gradient of modified weight matrix $\boldsymbol{W}'$ for a single training data point as:

$$\nabla \boldsymbol{W}' = \nabla \boldsymbol{z}' \boldsymbol{a}^\top = (\boldsymbol{p}' - \boldsymbol{y}) \boldsymbol{a}^\top. \tag{5.15}$$

For the t-th row of $\nabla \boldsymbol{W}'$, denoted as $\nabla \boldsymbol{W}'_t$, we can derive that $\nabla \boldsymbol{W}'_t = (\boldsymbol{p}'_t - \boldsymbol{y}_t) \boldsymbol{a}^\top$, where $\boldsymbol{p}'_t$ becomes zero after our parameter manipulation. When $i \in \text{IndexSet}(\boldsymbol{x}_t)$,

it is clear that $\nabla \boldsymbol{W}_t'^{(i)} = -\boldsymbol{a}^{\top(i)}$. Conversely, when $i \notin \text{IndexSet}(\boldsymbol{x}_t)$, the gradient $\nabla \boldsymbol{W}_t'^{(i)} = \mathbf{0}$. Leveraging this observation, we can recover the averaged input feature embeddings of the target data $\overline{\boldsymbol{a}}^{\top}$, from the weight gradient $\overline{\nabla \boldsymbol{W}_t'}$ as follows:

$$\overline{\boldsymbol{a}}^{\top}(\boldsymbol{x}_t) = -\frac{1}{|\boldsymbol{x}_t|}\sum_{i=1}^{|\boldsymbol{x}_t|} \nabla \boldsymbol{W}_t'^{(i)} = -\frac{B}{|\boldsymbol{x}_t|}\,\overline{\nabla \boldsymbol{W}_t'}. \tag{5.16}$$

In Eq. (5.12), we have already obtained the manipulated probabilities $\boldsymbol{p}'$ for all the training data. Combining this with Eq. (5.4) and $\nabla \boldsymbol{b}' = \nabla \boldsymbol{z}'$, we can derive the averaged gradient of the bias vector for the target data $\overline{\nabla \boldsymbol{b}'}$ as:

$$\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t) = \frac{1}{|\boldsymbol{x}_t|}\sum_{i=1}^{|\boldsymbol{x}_t|} \nabla \boldsymbol{b}'^{(i)} = \frac{1}{|\boldsymbol{x}_t|}\sum_{i=1}^{|\boldsymbol{x}_t|} \left(\boldsymbol{p}'^{(i)} - \boldsymbol{y}^{(i)}\right) = \boldsymbol{p}' - \mathbb{I}(j=t), \tag{5.17}$$

where $\mathbb{I}(j=t)$ is an indicator function that equals 1 if $j=t$ and 0 otherwise, and $j \in \{1, 2, \cdots, C\}$ denotes the index of the class.

Finally, we can calibrate the obtained batch-averaged weight gradient $\overline{\nabla \boldsymbol{W}'}$ by associating $\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)$ in Eq. (5.17) with the derived feature embeddings $\overline{\boldsymbol{a}}(\boldsymbol{x}_t)$ in Eq. (5.16), which can be expressed as:

$$\overline{\nabla \boldsymbol{W}'}(\boldsymbol{x}_t) = \overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)\,\overline{\boldsymbol{a}}^{\top}(\boldsymbol{x}_t) = -\frac{B}{|\boldsymbol{x}_t|}\,\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)\,\overline{\nabla \boldsymbol{W}_t'}. \tag{5.18}$$

In the scenario of multiple target classes (where $|t| > 1$), we represent the sum of all rows corresponding to target classes as $\sum_{k\in t} \overline{\nabla \boldsymbol{W}_k'}$. Consequently, the derived feature embeddings $\overline{\boldsymbol{a}}^{\top}$, the averaged bias gradient $\overline{\nabla \boldsymbol{b}'}$, and the calibrated weight gradient $\overline{\nabla \boldsymbol{W}'}$ of target data are calculated as follows:

$$\overline{\boldsymbol{a}}^{\top}(\boldsymbol{x}_t) = -\frac{1}{|\boldsymbol{x}_t|}\sum_{i=1}^{|\boldsymbol{x}_t|}\sum_{k\in t} \nabla \boldsymbol{W}_k'^{(i)} = -\frac{B}{|\boldsymbol{x}_t|}\sum_{k\in t} \overline{\nabla \boldsymbol{W}_k'} \tag{5.19}$$

$$\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t) = \frac{1}{|\boldsymbol{x}_t|}\sum_{i=1}^{|\boldsymbol{x}_t|}\sum_{k\in t} \left(\boldsymbol{p}'^{(i)} - \boldsymbol{y}^{(i)}\right) = \boldsymbol{p}' - \frac{1}{|\boldsymbol{x}_t|}\sum_{k\in t} \mathbb{I}(j=k) \tag{5.20}$$

$$\overline{\nabla \boldsymbol{W}'}(\boldsymbol{x}_t) = \overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)\,\overline{\boldsymbol{a}}^{\top}(\boldsymbol{x}_t) = -\frac{B}{|\boldsymbol{x}_t|}\,\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)\sum_{k\in t} \overline{\nabla \boldsymbol{W}_k'}. \tag{5.21}$$

Therefore, by applying gradient separation and calibration, we can determine the batch-averaged gradients $\overline{\nabla \boldsymbol{\theta}'}$ corresponding to the target data $\boldsymbol{x}_t$. However, these received gradients in the previous $L-1$ layers represent an average across the entire batch. Based on Equation (5.8), we scale them by a factor $\frac{B}{|\boldsymbol{x}_t|}$ to obtain the gradients of the target data for subsequent data reconstruction, which can be expressed as:

$$\overline{\nabla \boldsymbol{\theta}'}(\boldsymbol{x}_t)^{[l]} = -\frac{B}{|\boldsymbol{x}_t|}\overline{\nabla \boldsymbol{\theta}'}^{[l]}, \quad \text{for } l = 1, 2, \cdots, L-1. \tag{5.22}$$

Note: We extract the batch-averaged gradients $\overline{\nabla \boldsymbol{b}'}(\boldsymbol{x}_t)$ and $\overline{\nabla \boldsymbol{W}'}(\boldsymbol{x}_t)$ of target data by calibrating the gradients of the final FC layer. This calibration process leverages the manipulated probabilities $\boldsymbol{p}'$ and the derived feature embeddings $\overline{\boldsymbol{a}}^{\top}(\boldsymbol{x}_t)$ to infer the gradients of the target data.

5.4.5 Data Reconstruction

Having successfully filtered out non-target data and retained target data gradients, we move on to recovering the target data from the batch-averaged gradients $\overline{\nabla \boldsymbol{\theta}'}(\boldsymbol{x}_t)$. We employ two distinct strategies for this purpose.

1. **Optimization-based Data Reconstruction**. We formulate an optimization problem that seeks to minimize the distance between the reconstructed data and its original counterpart, thereby recovering the target data.

2. **Analytical Data Reconstruction**. We leverage the gradients in the FC layer to analytically restore the target data by inserting an FC layer into the first layer of the global shared model. This approach enables direct gradient inversion for reconstructing the target data.

To determine the reconstruction method, we employ a pre-defined threshold β. Specifically, if the number of the target samples $|\boldsymbol{x}_t|$ is less than or equal to β, we opt for optimization-based reconstruction; otherwise, we select analytical reconstruction.

Optimization-based Data Reconstruction

In the optimization-based approach, we aim to recover the original target data $\boldsymbol{x}_t$ by solving an optimization problem that minimizes the distance between the obtained target gradients $\overline{\nabla\boldsymbol{\theta}'}(\boldsymbol{x}_t)$ and the dummy gradients $\nabla\boldsymbol{\theta}'(\boldsymbol{x}_t)$ that calculated from the dummy data $\boldsymbol{x}^*$. The optimization problem can be formulated as:

$$\arg\min_{\boldsymbol{x}_t^* \in [0,1]^n} 1 - \frac{\langle \overline{\nabla\boldsymbol{\theta}'}(\boldsymbol{x}_t^*),\, \overline{\nabla\boldsymbol{\theta}'}(\boldsymbol{x}_t)\rangle}{\|\overline{\nabla\boldsymbol{\theta}'}(\boldsymbol{x}_t^*)\|\, \|\overline{\nabla\boldsymbol{\theta}'}(\boldsymbol{x}_t)\|}, \tag{5.23}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, and $\|\cdot\|$ represents the Euclidean norm. The dummy data $\boldsymbol{x}_t^*$ is initialized with random Gaussian noise and optimized iteratively to minimize the objective function in Eq. (5.23). The reconstructed target data $\boldsymbol{x}_t^*$ is obtained when the optimization process converges.

Analytical Data Reconstruction

Building upon the work presented in [17], we can analytically restore the target data by inserting an FC layer into the first layer of the model. In this FC layer, leveraging Eq. (5.6) and $\nabla\boldsymbol{b} = \nabla\boldsymbol{z}$, a single input data point $\boldsymbol{x}$ can be directly recovered from $\boldsymbol{x} = \nabla\boldsymbol{W}_k^\top \oslash \nabla\boldsymbol{b}_k$ based on the gradients of the k-th neuron. By carefully crafting the weights and biases, data with different mean values (e.g., brightness in images) can activate specific neurons in an ordering that corresponds to their numerical magnitude, from top to bottom. This concept can be formalized as follows:

$$(\overline{\nabla\boldsymbol{W}}_l^\top - \overline{\nabla\boldsymbol{W}}_{l-1}^\top) \oslash (\overline{\nabla\boldsymbol{b}}_l - \overline{\nabla\boldsymbol{b}}_{l-1}) = \boldsymbol{x}_s + \sum_{m=1}^{q} \boldsymbol{x}_m - \sum_{m=1}^{q} \boldsymbol{x}_m = \boldsymbol{x}_s, \tag{5.24}$$

where $\oslash$ denotes the element-wise division, $\boldsymbol{x}_s$ is a specific data point that activates the l-th neuron, and $\{\boldsymbol{x}_m\}_{m=1}^q$ that already activate the previous $l-1$ neurons. This analytical approach effectively reconstructs part of the target data $\boldsymbol{x}_t^*$, proving more efficient and computationally lighter than the optimization-based method, especially for numerous target data points.

5.5 Experiments

In this section, we evaluate the performance of GradFilt. We begin by introducing our experimental setup, including the datasets and models, baselines, and evaluation metrics. Then, we present the attack's results for label restoration, gradient calibration, and data reconstruction. For data reconstruction, we provide a detailed analysis of both optimization-based and analytical approaches. We conduct extensive experiments across various FL settings and display visualization results to illustrate the data reconstruction process and recovered images.

5.5.1 Experimental Setup

Datasets and Models

We evaluate GradFilt on four benchmark image datasets and corresponding neural network models. Here are the details of the datasets in our experiments.

- Fashion-MNIST (F-MNIST) [74] contains 60k training and 10k testing gray-scale clothing images of size 28×28 with 10 classes.
- SVHN [49] contains 73k training and 26k testing color digit images of size 32×32 with 10 classes.
- CIFAR-10/100 [34] contain 50k training and 10k testing color images of size 32×32 with 10/100 classes.
- ImageNet-1K [12] contains 1.28M training and 50k validation color images of size 224×224 with 1,000 classes.

Our experiments utilize several neural network architectures: LeNet [37], ConvNet, VGG11 [59], and models from ResNet series [24]. Specifically, ConvNet consists of 10

layers: eight 3×3 convolutional layers with Batch Normalization (BN) [27], followed by one max-pooling layer and one FC layer. We employ VGG11 for CIFAR-10, ResNet18 for CIFAR-100, and ResNet50 for ImageNet datasets.

Implementation Details

We consider an FL system with N participants, one of which is targeted as the victim. Each client trains the model on its own private data using the FedSGD algorithm [44]. In each training iteration, a client randomly selects B samples to form a mini-batch. Because GradFilt can be applied to any communication round, we focus our attack on the initial stages of training. The threshold β for selecting the reconstruction method is set to 8. All experiments are conducted using PyTorch 2.4.0 [54] and CUDA 12.2 on a workstation equipped with an Intel i9-10900K CPU @ 3.70GHz, 64GB RAM, and an NVIDIA GeForce RTX 4090 (24GB) GPU.

Baselines and Evaluation Metrics

We compare our proposed GradFilt with two baselines: *Inverting Gradients* (IG) [18] and *Deep Leakage from Gradients* (DLG) [88]. IG employs cosine similarity as its error function, Adam as the optimizer, and Total Variation (TV) as a regularization term. It runs for a maximum of 12,000 iterations. DLG uses Euclidean distance as its error function, L-BFGS as the optimizer, and runs for a maximum of 600 iterations.

We evaluate the quality of reconstructed images using three common metrics: *Mean Squared Error* (MSE), *Peak Signal-to-Noise Ratio* (PSNR), and *Learned Perceptual Image Patch Similarity* (LPIPS) [81]. MSE measures pixel-wise differences between the reconstructed and original images. PSNR assesses pixel-wise similarity, indicating higher quality with larger values. LPIPS evaluates perceptual similarity, capturing how visually alike the reconstructed and original images appear. Lower MSE and LPIPS scores, along with a higher PSNR, signify better reconstruction quality. All

Table 5.1: Label restoration performance of GradFilt across various batch sizes ($B \in \{2, 8, 32, 128, 512\}$), different datasets, and diverse models. GradFilt achieves perfect accuracy (100% InsAcc and 100% ClsAcc) in all these FL settings.

Dataset	**Model**	**Batch Size B**				
		2	8	32	128	512
F-MNIST	LeNet	100%	100%	100%	100%	100%
SVHN	ConvNet	100%	100%	100%	100%	100%
CIFAR-10	VGG11	100%	100%	100%	100%	100%
CIFAR-100	ResNet18	100%	100%	100%	100%	100%
ImageNet	ResNet50	100%	100%	100%	100%	100%

results are presented as average values across the target data or mini-batch.

Furthermore, we evaluate label restoration performance using two metrics: *Instance-level Accuracy* (InsAcc) and *Class-level Accuracy* (ClsAcc) [83]. InsAcc measures the accuracy of correctly recovering the true labels for individual data samples. ClsAcc assesses the accuracy of assigning data samples to their correct classes. All results are reported as percentages (%).

5.5.2 Label Restoration

We evaluate GradFilt's label restoration performance across five diverse datasets and corresponding models: F-MNIST (LeNet), SVHN (ConvNet), CIFAR-10 (VGG11), CIFAR-100 (ResNet18), and ImageNet (ResNet50). Experiments are conducted with varying batch sizes ($B \in \{2, 8, 32, 128, 512\}$) to assess the robustness of GradFilt. As shown in Table 5.1, GradFilt achieves perfect label restoration accuracy (100% InsAcc

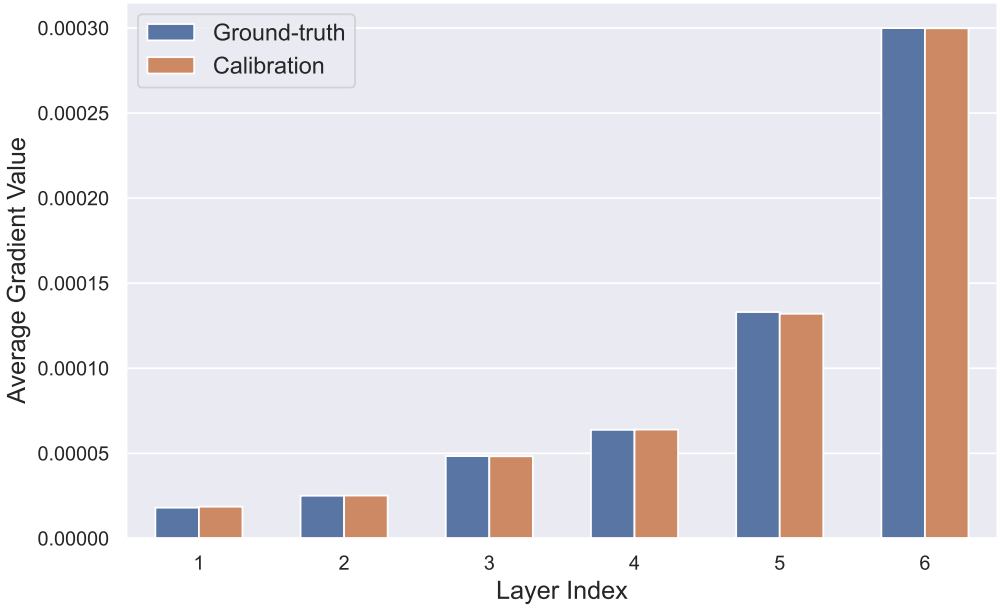


Figure 5.3: Layer-wise errors between the calibrated gradients and true target gradients on the CIFAR-10 dataset using the ConvNet model. The errors are quantified by calculating the Euclidean distance between corresponding gradients in each layer, and then averaged over the number of parameters within the layer.

and 100% ClsAcc) across all FL settings. This outcome, while remarkable, is expected given the parameter manipulation results in identical Softmax probabilities for all training data: target class probability $p'_t = 0$ and non-target class probability $p'_k = \frac{1}{C-|t|}$. This deterministic output, both for manipulated probabilities $\boldsymbol{p}'$ and potentially other variables, highlights the vulnerability of FL systems to privacy breaches. Our findings demonstrate GradFilt's efficacy in restoring mini-batch labels regardless of dataset, model architecture, or batch size.

5.5.3 Gradient Calibration

Before inverting the "filtered" gradients to reconstruct the target data, we evaluate the gradient calibration process described in Section 5.4.4. To validate the accuracy of

this calibration, we compute the layer-wise errors between the calibrated gradients and the true target gradients. We conduct experiments on the CIFAR-10 dataset using the ConvNet model, selecting the first six layers for demonstration. The results are visualized in Figure 5.3. As evident from the figure, the layer-wise errors between the calibrated and target gradients are exceedingly small. This finding strongly suggests that the calibration process is highly effective in isolating the gradients specific to the target data from the batch-averaged gradients. Our findings demonstrably showcase the feasibility of GradFilt in extracting these precise gradients associated with the target data, laying a solid foundation for subsequent data reconstruction.

5.5.4 Optimization-based Data Reconstruction

To evaluate the effectiveness of our proposed GradFilt method for data reconstruction, we first focus on scenarios where the number of target data points is less than or equal to the predefined threshold β. We compare the reconstruction quality of GradFilt against IG and DLG, where the experiments are conducted on the CIFAR-10 dataset using the ConvNet model. Four distinct FL settings are involved, each with a varying number of target data points ($|\boldsymbol{x}_t| \in \{1, 2, 4, 8\}$) and corresponding batch sizes ($B \in \{2, 4, 8, 16\}$). The results are summarized in Table 5.2.

It is shown that GradFilt consistently achieves the lowest MSE, the highest PSNR, and the lowest LPIPS scores across all experimental settings. Notably, the PSNR scores of images reconstructed by GradFilt exceeds 18 under the first three settings. This indicates a high degree of similarity between the reconstructed images and the original images, suggesting that the reconstruction process preserves essential visual details with only minor noise or color discrepancies. This clearly demonstrates the superior reconstruction capabilities of GradFilt compared to the baseline methods. Furthermore, it is observed that as the number of target data samples increases, the overall reconstruction quality for both GradFilt and the baselines degrades. This

Table 5.2: Comparison of data reconstruction quality between GradFilt and the baselines (IG and DLG) on the CIFAR-10 dataset using the ConvNet model. The results are reported with varying numbers of target data points ($|\boldsymbol{x}_t| \in \{1, 2, 4, 8\}$) and batch sizes ($B \in \{2, 4, 8, 16\}$). The best performance are highlighted in bold.

Method	**Metric**	**Number of Target Data (Batch Size)**			
		$\lvert\boldsymbol{x}_t\rvert = 1$ ($B = 2$)	$\lvert\boldsymbol{x}_t\rvert = 2$ ($B = 4$)	$\lvert\boldsymbol{x}_t\rvert = 4$ ($B = 8$)	$\lvert\boldsymbol{x}_t\rvert = 8$ ($B = 16$)
GradFilt	MSE ↓	**0.0057**	**0.0113**	**0.0159**	**0.0347**
	PSNR ↑	**22.41**	**19.54**	**18.11**	**14.85**
	LPIPS ↓	**0.07**	**0.10**	**0.18**	**0.26**
IG [18]	MSE ↓	0.0174	0.0460	0.0369	0.0491
	PSNR ↑	18.30	14.60	14.99	13.54
	LPIPS ↓	0.10	0.19	0.19	0.31
DLG [88]	MSE ↓	0.1423	0.1104	0.1093	0.1237
	PSNR ↑	8.47	9.61	9.71	9.26
	LPIPS ↓	0.46	0.41	0.37	0.42

decline in performance can be attributed to the increased complexity of reconstructing multiple targets simultaneously.

To further demonstrate GradFilt's versatility, we evaluate its performance on a variety of datasets and model architectures. These experiments utilize the same settings as those presented in Table 5.1, with the number of target data points set to 4 and 8, within the batch sizes of 16 and 32. Importantly, the effectiveness of GradFilt is largely independent of batch size. This highlights a key advantage: accurate target data recovery can be achieved regardless of the batch size used during FL training. As shown in Table 5.3, GradFilt consistently achieves lower MSE, higher PSNR, and

Table 5.3: Ablation study of data reconstruction quality using GradFilt on various datasets and model architectures. The results are reported with $|\boldsymbol{x}_t| = 4$ and $|\boldsymbol{x}_t| = 8$ target data points that included in the batch size B of 16 and 32.

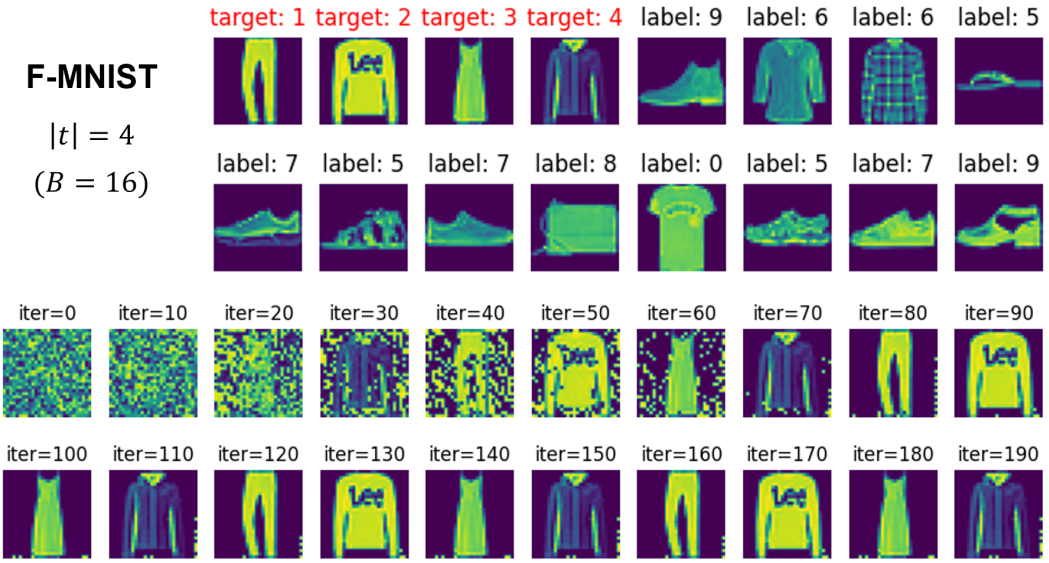
Dataset	**Model**	$\|\boldsymbol{x_t}\| = \mathbf{4}$ $(B = 16)$			$\|\boldsymbol{x_t}\| = \mathbf{8}$ $(B = 32)$		
		MSE	PSNR	LPIPS	MSE	PSNR	LPIPS
F-MNIST	LeNet	0.0015	39.25	0.0008	0.0026	35.94	0.0019
SVHN	ConvNet	0.0094	20.50	0.13	0.0153	18.94	0.17
CIFAR-10	VGG11	0.0416	14.09	0.19	0.0669	12.59	0.17
CIFAR-100	ResNet18	0.0168	18.49	0.22	0.0237	16.52	0.23
ImageNet	ResNet50	0.0695	12.11	0.71	0.1000	10.26	0.76

lower LPIPS scores across these diverse settings. While GradFilt demonstrates strong performance across different datasets and models, it is noteworthy that reconstruction quality tends to decrease as dataset complexity increases and model capacity grows. For instance, the F-MNIST dataset paired with the LeNet model yields the highest reconstruction quality, producing images visually indistinguishable from the originals. This can be attributed to the relative simplicity of both the dataset and the model, which facilitates accurate target data recovery. Conversely, the ImageNet dataset combined with the ResNet50 model exhibits the lowest reconstruction quality due to its inherent complexity and the sophisticated nature of the ResNet50 architecture. These factors pose greater challenges for the reconstruction process.

To provide a visual illustration of the data reconstruction process facilitated by GradFilt, we selected two representative scenarios:

1. **CIFAR-100 (ConvNet)**: One target data point from class 66 is chosen for demonstrating data reconstruction.

(a) Data reconstruction process on CIFAR-100 dataset trained with ConvNet model. One target sample from class 66 is selected for reconstruction.



(b) Data reconstruction process on F-MNIST dataset trained with LeNet model. Four target samples from classes 1 to 4 are simultaneously reconstructed.

Figure 5.4: Illustration of data reconstruction process using GradFilt. The top two rows showcase the original batch images (red label for targets), while the bottom two rows present the corresponding reconstructed images obtained at each iteration.

2. **F-MNIST (LeNet)**: Four samples, each belonging to classes 1 to 4, are selected for simultaneous reconstruction.

Figure 5.4 presents the visualization results of data reconstruction process over the initial 200 iterations. In the case of F-MNIST, one artifact sample is displayed per iteration, showcasing the iterative nature of the optimization-based method employed by GradFilt. Our experimental results demonstrate that GradFilt effectively recovers all target samples within a batch when data complexity and model depth are relatively low. This highlights the GradFilt's ability to accurately reconstruct individual data points even within a multi-sample setting.

5.5.5 Analytical Data Reconstruction

To address scenarios with a larger number of target data points, we introduce an analytical reconstruction method as a complement to the optimization-based approach. This method is activated when the number of target samples $|\boldsymbol{x}_t|$ exceeds the threshold β. The analytical reconstruction technique involves inserting an FC layer into the first layer of the model. Gradients within this FC layer are then utilized to reconstruct the target data. We employ 128 bins for data recovery by default, corresponding to 128 hidden neurons in the inserted FC layer. The specific model architecture is less crucial in this context, as our primary goal is to recover instances from the gradients produced by this inserted FC layer. We evaluate the performance of this analytical reconstruction approach on settings with $|\boldsymbol{x}_t| = 50$ and $|\boldsymbol{x}_t| = 100$ across four groups of datasets and models. The results are summarized in Table 5.4.

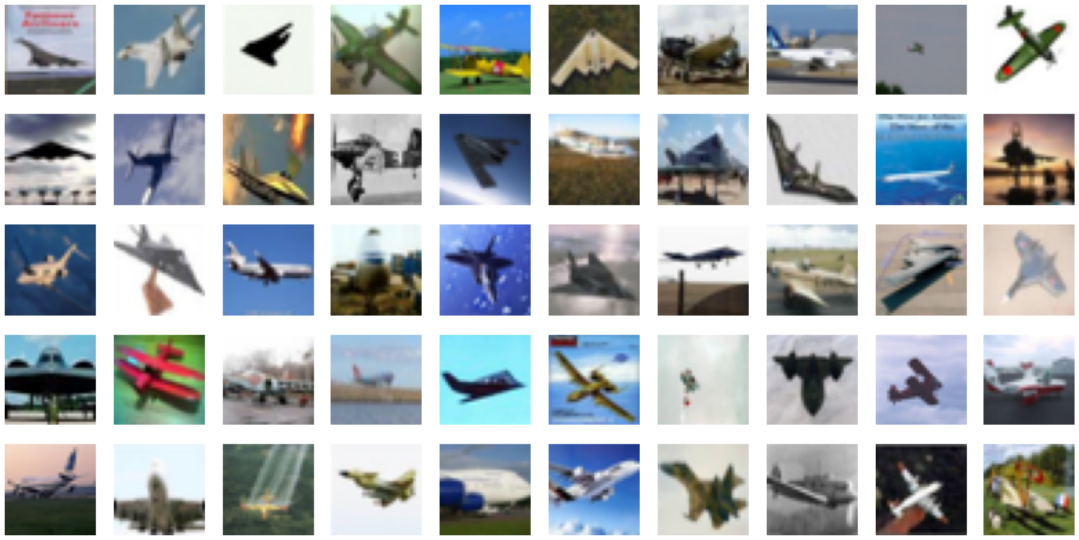
In Table 5.4, we use the *Hits score* to represent the percentage of successfully reconstructed target samples compared to the total number of ground-truth target data. While the reconstruction process inherently exhibits a degree of randomness, GradFilt consistently achieves high Hits scores across all evaluated datasets and models. This demonstrates GradFilt's effectiveness in extracting target data. Importantly, the ana-

Table 5.4: Analytical data reconstruction performance of GradFilt on settings with $|\boldsymbol{x}_t| = 50$ and $|\boldsymbol{x}_t| = 100$ target samples across various datasets and models. The Hits score represents the percentage of successfully reconstructed target data.

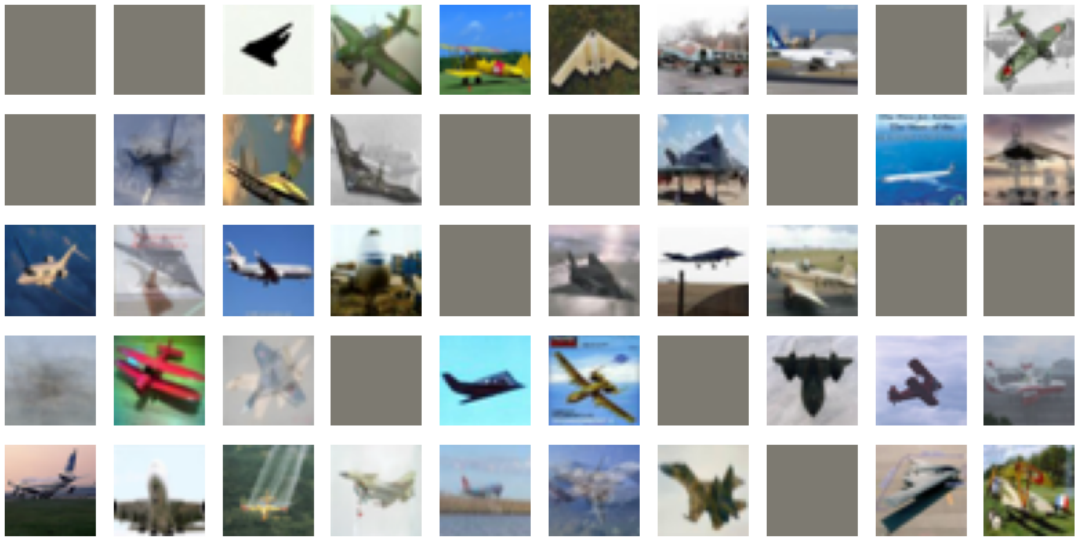
Dataset	$\|\boldsymbol{x}_t\| = 50$				$\|\boldsymbol{x}_t\| = 100$			
	Hits	MSE	PSNR	LPIPS	Hits	MSE	PSNR	LPIPS
SVHN	0.82	0.0256	76.31	0.10	0.59	0.0342	39.16	0.22
CIFAR-10	0.74	0.0356	64.65	0.16	0.66	0.0355	51.50	0.21
CIFAR-100	0.84	0.0189	86.71	0.10	0.68	0.0556	52.98	0.21
ImageNet	0.78	0.0227	71.59	0.25	0.72	0.0298	60.27	0.29

lytical approach yields significantly superior image quality compared to optimization-based reconstruction. The PSNR scores for all experiments exceed 39, indicating that the recovered images are nearly indistinguishable from the original images. However, as the number of target data points increases, both the Hits score and overall reconstruction quality tend to decrease. This observation highlights the limitations of the analytical approach when dealing with numerous targets and underscores the need for adaptive strategies, such as the threshold β, to ensure optimal performance.

To further illustrate the efficacy of analytical reconstruction mechanism, we provide visualizations of the results obtained from CIFAR-10 and ImageNet datasets when $|\boldsymbol{x}_t| = 50$. For CIFAR-10, we select the first category (airplane) as the target class, while for ImageNet, we choose the second class (white shark). The visualizations are presented in Figure 5.5 and Figure 5.6, respectively. Observing these figures reveals that GradFilt effectively reconstructs the target data with high fidelity. Reconstructed images closely resemble their original counterparts, demonstrating the accuracy of the analytical approach. Gray-scale images represent instances where reconstruction is not entirely successful, and are filled with zeros.

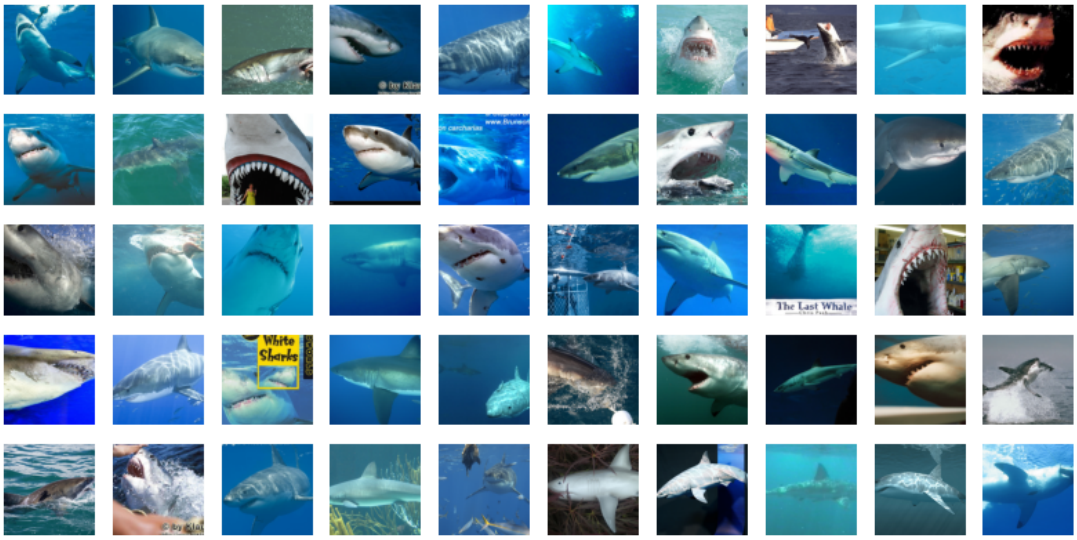


(a) Ground-truth images of the target class (airplane) from CIFAR-10 training batch with $|\boldsymbol{x}_t| = 50$.

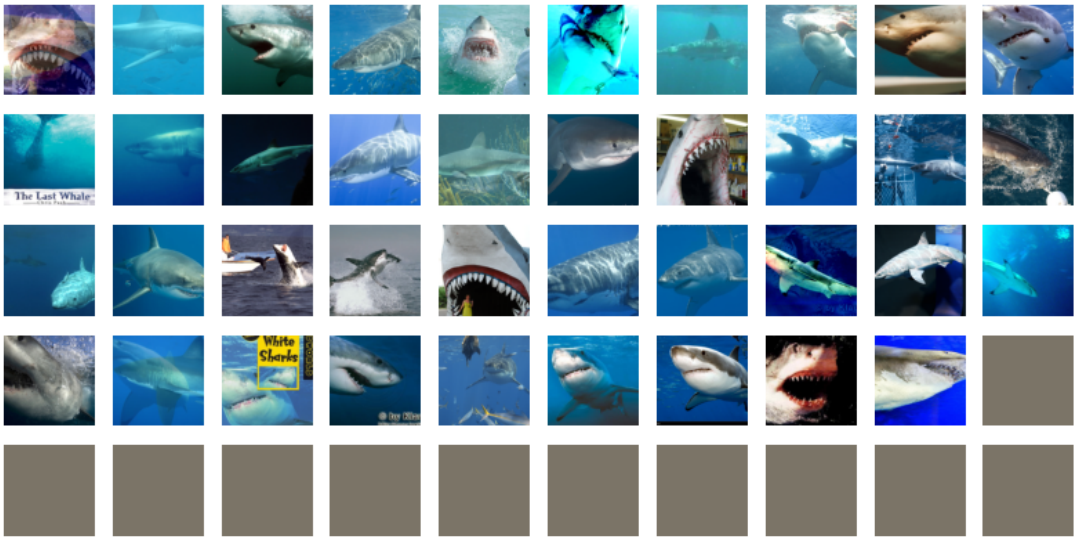


(b) Analytical reconstruction of airplane targets from CIFAR-10, where 37 images are successfully reconstructed ($|\boldsymbol{x}_t^*| = 37$).

Figure 5.5: Illustration of the analytical data reconstruction performance of GradFilt on airplane targets from CIFAR-10 dataset. 37 out of 50 target images are successfully reconstructed, while the remaining 13 are represented in gray-scale to indicate unsuccessful reconstruction attempts.



(a) Ground-truth images of the target class (white shark) from ImageNet training batch with $|\boldsymbol{x}_t| = 50$.



(b) Analytical reconstruction of white shark targets from ImageNet, where 39 images are successfully reconstructed ($|\boldsymbol{x}_t^*| = 39$).

Figure 5.6: Illustration of the analytical data reconstruction performance of GradFilt on targets like white shark from ImageNet dataset. 39 out of 50 target images are successfully reconstructed, while the remaining 11 are represented in gray-scale to indicate unsuccessful reconstruction attempts.

5.6 Chapter Summary

In this work, we propose GradFilt to reconstruct training samples of target classes in FL collaboration by leveraging the carefully crafted gradients. GradFilt embeds a "filter" inside the model to retain only the gradients of the target classes by modifying the parameters of the final fully connected (FC) layer. By exploiting these gradients, GradFilt can effectively reconstruct the target data with high fidelity. Our evaluation demonstrates that GradFilt outperforms existing gradient leakage attacks in terms of data reconstruction quality. Future work will focus on exploring defense mechanisms against GradFilt to enhance the privacy and security of FL systems.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we investigate the privacy leakage vulnerabilities arising from gradient-based attacks in Federated Learning (FL). We first summarize three key challenges often overlooked by existing attack methods: label information leakage, layer-wise gradient exploitation, and targeted data recovery. To address these critical issues, we propose three novel and advanced gradient-induced attacks.

To efficiently recover training labels, we introduce a generalized label recovery attack by estimating the posterior probabilities from auxiliary data. We also reveal the essence of label leakage from gradients and explain the reasons for such findings from the perspective of the exponential family. Then, we further expose privacy breaches through correlated layer-wise gradients with our Gradient Bridge (GDBR) attack. GDBR reconstructs the gradient of the model's output logits, enabling restoration of the label distribution even under restricted gradient sharing. Finally, to reconstruct the training data for specific classes, we design Gradient Filtering (GradFilt). This method effectively recovers desired samples with higher accuracy by strategically modifying parameters within the classification layer.

These meticulously crafted attacks expose fundamental vulnerabilities in current FL systems. By exploiting the lack of analysis regarding relationships between gradients and data, attackers can infer critical gradients and deduce sensitive information about training labels. Furthermore, the absence of robust anomaly detection mechanisms for the globally shared model leaves FL susceptible to malicious manipulation by a compromised server. By shedding light on these attack surfaces, we hope to inspire the development of more secure and privacy-preserving FL systems in the future.

6.2 Limitation

Despite our research shedding light on several vulnerabilities within FL systems, it is important to acknowledge the limitations inherent in our study.

Firstly, our investigation primarily focuses on the FedSGD algorithm, which is considered the most representative and potentially problematic algorithm in FL. However, we did not extend our analysis to more advanced algorithms such as FedAvg. The FedAvg algorithm may mitigate the effectiveness of certain attacks due to the reduced interaction between users and attackers. This aspect presents an opportunity for future research to explore.

Secondly, our study is centered around classification tasks and models. While classification tasks are predominant in machine learning, there are other types of tasks such as regression and generative tasks that we did not cover. Our focus on classification tasks aligns with the mainstream approach of most GIAs. Investigating other tasks and models may uncover additional intriguing findings.

Furthermore, our attacks require auxiliary datasets to achieve optimal results. Although common datasets can often be sourced from the internet or public repositories, there may be niche categories with non-IID distributions for which auxiliary data is not readily available. This limitation could impact the effectiveness of the attacks.

However, given the uncertainty surrounding the capabilities of potential attackers, providing reasonable auxiliary datasets allows us to reveal the most severe scenarios of information leakage and privacy breaches in FL systems.

6.3 Future Work

This thesis investigates the vulnerabilities of FL systems to gradient-induced privacy leakage, demonstrating the potential for adversaries to extract sensitive information. While our work makes significant strides in understanding these risks, numerous avenues remain open for future exploration in this rapidly evolving field. We outline several promising directions for further investigation below.

To safeguard labels, incorporating noise injection techniques like noise labels or label differential privacy during training presents a promising direction. This approach not only obfuscates sensitive information within labels, preventing attackers from inferring local data distributions or attributes, but also effectively mitigates the reconstruction of original training samples from shared gradients.

Given the inherent correlations between gradients, sensitive gradients can be identified through careful analysis. Subsequently, these sensitive gradients can be preserved at each layer using cryptography-based methods, such as Homomorphic Encryption. This approach can achieve lightweight and efficient privacy protection without significantly impacting the performance of FL models.

To enhance the security of globally shared models in FL, participating clients can implement anomaly detection mechanisms. On one hand, clients can collaborate to verify the consistency of the global model. On the other hand, individual clients can compute gradients using a subset of their local datasets and analyze these gradients for anomalies in terms of both direction and magnitude. This multi-faceted approach helps identify potential tampering or manipulation of the FL shared model.

Furthermore, the risk of privacy leakage from gradients can be indirectly quantified by analyzing the relationship between client training data and calculated gradients. Any discernible correlation suggests a vulnerability to potential leakage. Metrics like mutual information or sensitivity analysis provide a rigorous means of measuring this relationship, revealing how gradient leakage risk is interconnected with factors like model status and underlying data distribution.

References

[1] Erling Bernhard Andersen. Sufficiency and exponential families for discrete sample spaces. *Journal of the American Statistical Association*, 65(331):1248–1255, 1970.

[2] Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. Lamp: Extracting text from gradients with language model priors. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, pages 7641–7654, 2022.

[3] Mislav Balunović, Dimitar Iliev Dimitrov, Robin Staab, and Martin Vechev. Bayesian framework for gradient leakage. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, pages 1–16, 2022.

[4] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *Proceedings of the 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199, 2023.

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191, 2017.

[6] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. In *Proceedings of the 2nd Machine Learning and Systems (MLSys)*, pages 374–388, 2019.

[7] Cangxiong Chen and Neill D. F. Campbell. Understanding training-data leakage from gradients in neural networks for image classifications. In *Workshop on Privacy in Machine Learning in Conjunction with NeurIPS 2021 (PriML-NeurIPS-21)*, 2021.

[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, pages 1–14, 2016.

[9] Trung Dang, Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, Peter Chin, and Françoise Beaufays. Revealing and protecting labels in distributed training. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pages 1727–1738, 2021.

[10] Trung Dang, Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, Peter Chin, and Françoise Beaufays. A method to reveal speaker identity in distributed asr training, and how to counter it. In *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4338–4342, 2022.

[11] Luke N Darlow, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 2009 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009.

[13] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, Chang Liu, Chee Seng Chan, and Qiang Yang. Rethinking privacy preserving deep learning: How to evaluate and thwart privacy attacks. In *Federated Learning: Privacy and Incentive*, pages 32–50. Springer, 2020.

[14] Liyue Fan. Image pixelization with differential privacy. In *Proceedings of the 2018 IFIP Annual Conference on Data and Applications Security and Privacy*, pages 148–162, 2018.

[15] Liyue Fan. Differential privacy for image publication. In *Workshop on the Theory and Practice of Differential Privacy in Conjunction with CCS 2019 (TPDP-CCS-19)*, pages 1–4, 2019.

[16] Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Helen Möllering, Thien Duc Nguyen, Phillip Rieger, Ahmad-Reza Sadeghi, Thomas Schneider, Hossein Yalame, et al. Safelearn: Secure aggregation for private federated learning. In *Proceedings of the 2021 IEEE Security and Privacy Workshops (SPW)*, pages 56–62, 2021.

[17] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

[18] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, pages 16937–16947, 2020.

[19] Jiahui Geng, Yongli Mou, Feifei Li, Qing Li, Oya Beyan, Stefan Decker, and Chunming Rong. Towards general deep leakage in federated learning. In *Workshop on Trustable, Verifiable and Auditable Federated Learning in Conjunction with AAAI 2022 (FL-AAAI-22)*, pages 1–8, 2022.

[20] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering private text in federated learning of language models. *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, pages 8130–8143, 2022.

[21] Ali Hatamizadeh, Hongxu Yin, Holger R Roth, Wenqi Li, Jan Kautz, Daguang Xu, and Pavlo Molchanov. Gradvit: Gradient inversion of vision transformers. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10021–10030, 2022.

[22] Fengxiang He, Bohan Wang, and Dacheng Tao. Tighter generalization bounds for iterative differentially private learning algorithms. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 802–812, 2021.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[25] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 4507–4518, 2020.

[26] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pages 7232–7241, 2021.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[28] Jiwnoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pages 29898–29908, 2021.

[29] Xiao Jin, Pin-Yu Chen, Chia-Yi Hsu, Chia-Mu Yu, and Tianyi Chen. Cafe: Catastrophic data leakage in vertical federated learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pages 994–1006, 2021.

[30] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 3252–3261, 2019.

[31] Sanjay Kariyappa, Chuan Guo, Kiwan Maeng, Wenjie Xiong, G Edward Suh, Moinuddin K Qureshi, and Hsien-Hsin S Lee. Cocktail party attack: Breaking aggregation-based privacy in federated learning using independent component analysis. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 15884–15899, 2023.

[32] Jinsu Kim, Dongyoung Koo, Yuna Kim, Hyunsoo Yoon, Junbum Shin, and Sungwook Kim. Efficient privacy-preserving matrix factorization for recommendation

via fully homomorphic encryption. *ACM Transactions on Privacy and Security (TOPS)*, 21(4):1–30, 2018.

[33] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 972–981, 2017.

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto*, 2009.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of the 26th Conference on Neural Information Processing Systems (NeurIPS)*, pages 1–9, 2012.

[36] Maximilian Lam, Gu-Yeon Wei, David Brooks, Vijay Janapa Reddi, and Michael Mitzenmacher. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 5959–5968, 2021.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Zhaohua Li, Le Wang, Guangyao Chen, Muhammad Shafq, et al. A survey of image gradient inversion against federated learning. *Authorea Preprints*, 2022.

[39] Dragos Lia and Mihai Togan. Privacy-preserving machine learning using federated learning and secure aggregation. In *Proceedings of the 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6, 2020.

[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[41] Guodong Long, Yue Tan, Jing Jiang, and Chengqi Zhang. Federated learning for open banking. In *Federated Learning: Privacy and Incentive*, pages 240–254. Springer, 2020.

[42] Kailang Ma, Yu Sun, Jian Cui, Dawei Li, Zhenyu Guan, and Jianwei Liu. Instance-wise batch label restoration via gradients in federated learning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, pages 1–15, 2023.

[43] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5188–5196, 2015.

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282, 2017.

[45] Shagufta Mehnaz, Sayanton V Dibbo, Roberta De Viti, Ehsanul Kabir, Björn B Brandenburg, Stefan Mangard, Ninghui Li, Elisa Bertino, Michael Backes, Emiliano De Cristofaro, et al. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security)*, pages 4579–4596, 2022.

[46] Yuan Mei, Binbin Guo, Danyang Xiao, and Weigang Wu. Fedvf: Personalized federated learning based on layer-wise parameter updates with variable frequency. In *Proceedings of the 2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–9, 2021.

[47] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted

boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[48] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papemoti, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*, pages 866–882, 2021.

[49] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *Workshop on Deep Learning and Unsupervised Feature Learning in Conjunction with NeurIPS 2011 (DLUFL-NeurIPS-11)*, pages 1–9, 2011.

[50] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[51] Dinh C Nguyen, Quoc-Viet Pham, Pubudu N Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(3):1–37, 2022.

[52] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, pages 1–14, 2020.

[53] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2429–2443, 2022.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, pages 1–12, 2019.

[55] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security (TIFS)*, 13(5):1333–1345, 2018.

[56] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pages 318–362. MIT Press, 1987.

[57] Daniel Scheliga, Patrick Mäder, and Marco Seeland. Combining variational modeling with partial gradient perturbation to prevent deep gradient leakage. *arXiv preprint arXiv:2208.04767*, 2022.

[58] Geet Shingi. A federated learning based approach for loan defaults prediction. In *Proceedings of the 2020 International Conference on Data Mining Workshops (ICDMW)*, pages 362–368, 2020.

[59] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[60] Ekanut Sotthiwat, Liangli Zhen, Zengxiang Li, and Chi Zhang. Partially encrypted multi-party computation for federated learning. In *Proceedings of the 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 828–835, 2021.

[61] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. Soteria: Provable defense against privacy leakage in federated learning from representation perspective. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9311–9319, 2021.

[62] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the 3rd ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys)*, pages 61–66, 2020.

[63] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, pages 14269–14278, 2019.

[64] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-level label leakage from gradients in federated learning. In *Proceedings of the 22nd Privacy Enhancing Technologies Symposium (PETS)*, pages 227–244, 2022.

[65] Xiaodong Wang, Longfei Wu, and Zhitao Guan. Graddiff: Gradient-based membership inference attacks against federated distillation with differential comparison. *Information Sciences*, 658:120068, 2024.

[66] Yijue Wang, Jieren Deng, Dan Guo, Chenghong Wang, Xianrui Meng, Hang Liu, Caiwen Ding, and Sanguthevar Rajasekaran. Sapag: A self-adaptive privacy attack from gradients. *arXiv preprint arXiv:2009.06228*, 2020.

[67] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from feder-

ated learning. In *Proceedings of the 2019 IEEE Conference on Computer Communications (INFOCOM)*, pages 2512–2520, 2019.

[68] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security (TIFS)*, 15:3454–3469, 2020.

[69] Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. A framework for evaluating gradient leakage attacks in federated learning. In *Proceedings of the 25th European Symposium on Research in Computer Security (ESORICS)*, pages 545–566, 2020.

[70] Wenqi Wei, Ling Liu, Yanzhao Wut, Gong Su, and Arun Iyengar. Gradient-leakage resilient federated learning. In *Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 797–807, 2021.

[71] Wenqi Wei, Ling Liu, Jingya Zhou, Ka-Ho Chow, and Yanzhao Wu. Securing distributed sgd against gradient leakage threats. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 34(7):2040–2054, 2023.

[72] Yuxin Wen, Jonas A Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 23668–23684, 2022.

[73] Spencer Wheatley, Thomas Maillart, and Didier Sornette. The extreme risk of personal data breaches and the erosion of privacy. *The European Physical Journal B*, 89:1–12, 2016.

[74] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel im-

age dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[75] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5:1–19, 2021.

[76] He Yang. H-fl: A hierarchical communication-efficient and privacy-protected architecture for federated learning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 479–485, 2021.

[77] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[78] Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16337–16346, 2021.

[79] Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC)*, pages 493–506, 2020.

[80] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, pages 1–13, 2018.

[81] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[82] Rui Zhang, Song Guo, Junxiao Wang, Xin Xie, and Dacheng Tao. A survey on gradient inversion: Attacks, defenses and future directions. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5678–5685, 2023.

[83] Rui Zhang, Song Guo, and Ping Li. Posterior probability-based label recovery attack in federated learning. In *Workshop on Privacy Regulation and Protection in Machine Learning in Conjunction with ICLR 2024 (PML-ICLR-24)*, 2024.

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

[85] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP)*, pages 1287–1305, 2024.

[86] Yanchong Zheng. Dropout against deep leakage from gradients. *arXiv preprint arXiv:2108.11106*, 2021.

[87] Junyi Zhu and Matthew Blaschko. R-gap: Recursive gradient attack on privacy. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, pages 1–17, 2021.

[88] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, pages 14774–14784, 2019.