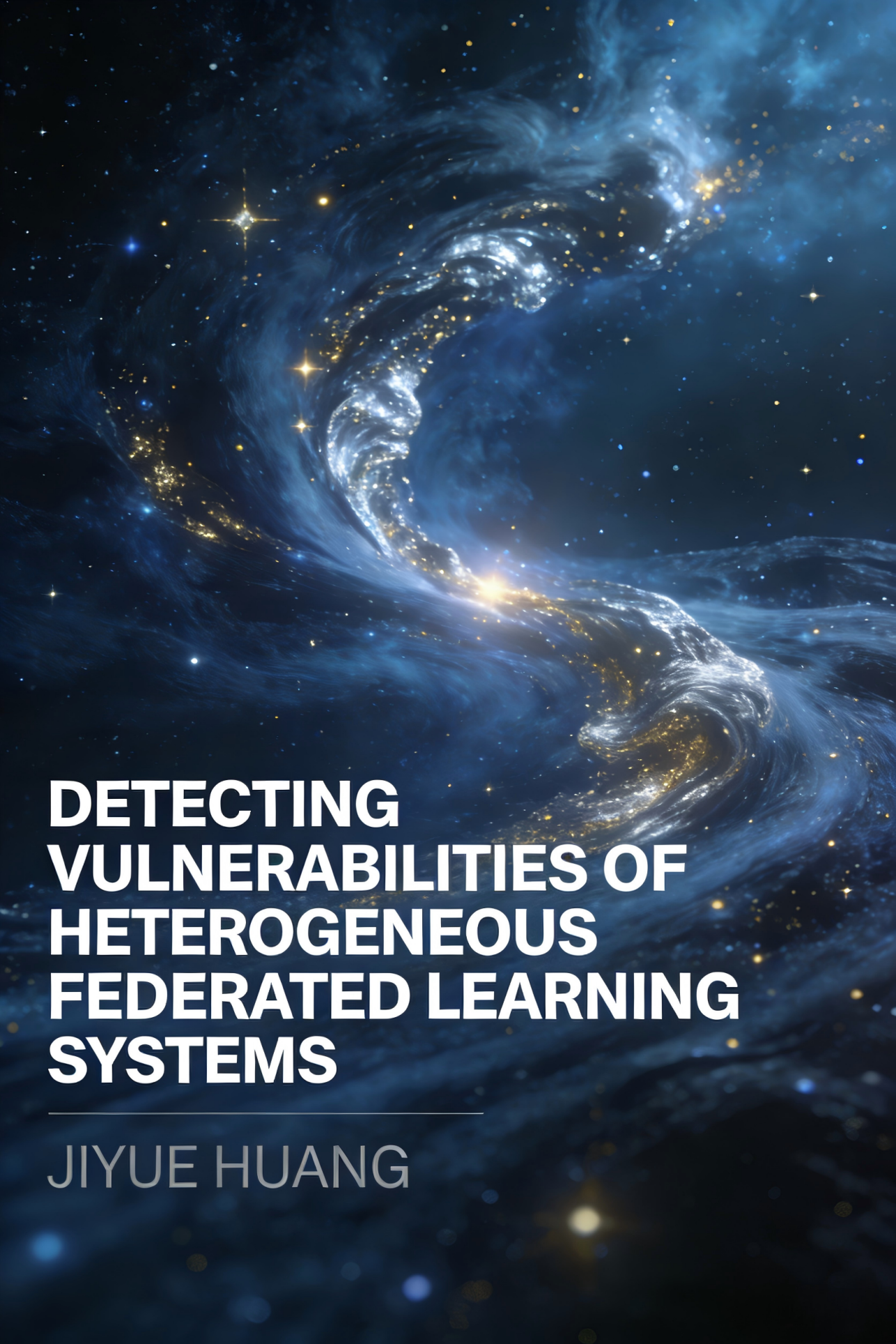


DETECTING VULNERABILITIES OF HETEROGENEOUS FEDERATED LEARNING SYSTEMS

JIYUE HUANG

DETECTING VULNERABILITIES OF HETEROGENEOUS FEDERATED LEARNING SYSTEMS

DETECTING VULNERABILITIES OF HETEROGENEOUS FEDERATED LEARNING SYSTEMS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus, Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates,
to be defended in public on Friday, 13 March 2026, 10:00.

by

Jiyue HUANG

This dissertation has been approved by the (co)promotors.

Composition of the promotion committee:

Rector Magnificus,	chairperson
Em. prof. dr. ir. D.H.J. Epema,	TU Delft, promotor
Prof. dr. Y. Chen,	TU Delft / University of Neuchatel, Switzerland, copromotor
Prof. dr. S. Roos,	RPTU Kaiserslautern-Landau, Germany, promotor
Independent Members:	
Prof. dr. G. Smaragdakis	TU Delft
Prof. dr. S. Picek,	TU Delft / Radboud University / University of Zagreb, Croatia
Prof. dr. M. Humbert,	University of Lausanne, Switzerland
Dr. S. Ben Mokhtar,	CNRS, France
Prof. dr. M.T.J. Spaan	TU Delft, *reserve member*

Keywords: Federated learning, Client selection, Privacy-preserving learning, Untargeted attack, Data-free attack, Multi-server attack, Gradient inversion attack

Printed by: DelftrainPrint

Cover by: ChatGPT and Doubao AI

ISBN 978-94-6518-264-3

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Dao can be known, but it may not be the well-known truth.

Laozi

CONTENTS

SUMMARY

Federated learning (FL) has emerged as an important paradigm in distributed machine learning, enabling collaborative model training across decentralized devices while preserving data privacy. FL's privacy-preserving nature – where raw data remains on local devices and only model updates are shared – has made it suitable in sensitive domains like healthcare and finance. However, the decentralized framework introduces fundamental challenges that threaten its reliability and adoption. Data heterogeneity, security threats, and privacy leakage risks create critical vulnerabilities that demand robust solutions.

To study such vulnerabilities, this thesis considers two kinds of parties: the clients and the servers. Clients act as data owners that perform localized computations and share only model parameters, thereby preserving raw data privacy, yet they introduce vulnerabilities through potential malicious behaviors (e.g., data/model poisoning attacks) or unreliable contributions due to data quality. In contrast, the server, while facilitating model convergence through aggregation, poses inherent privacy risks by potentially inferring sensitive client information from shared gradients, even without direct data access. These two parties create a dual-threat landscape: clients may compromise model performance through adversarial manipulations, while servers break confidentiality via reconstruction methods.

This thesis investigates these challenges through four key research questions, combining theoretical analysis with empirical validation to advance FL robustness. The first challenge stems from the inherent data heterogeneity in FL systems, where clients possess non-identically distributed data that can significantly impact model convergence. The second vulnerability involves security threats from malicious clients who may poison the training process. The third investigates privacy risk where servers could reconstruct sensitive data from model updates, especially under data reuse for training, violating FL's core privacy guarantees. The fourth study extends the privacy leakage concerns to emerging federated generative models.

In Chapter 2, we focus on client selection in heterogeneous FL environments. Traditional approaches often overlook participants with unique data distributions, leading to decreased model performance. We explore the role of specialized clients with unique data distributions and develop adaptive strategies to optimize their participation. We propose an adaptive selection strategy that dynamically evaluates the feature distances based on the data characteristics of the clients' models and the global model, ensuring optimal participation patterns across different learning phases.

In Chapter 3, we study untargeted attacks with limited knowledge or resources in FL systems. Unlike existing work that assumes strong adversary capabilities, we consider more realistic scenarios where attackers lack training data or update visibility to benign clients. The proposed attack framework demonstrates how carefully crafted synthetic

updates can effectively compromise model performance. We complement this chapter with a defense mechanism that identifies and mitigates such specific attacks.

In Chapter 4, we investigate privacy risks in multi-server FL environments, where clients participate in multiple concurrent learning tasks and reuse their local training data. We analyze how gradient updates from different tasks can be integrated to break data privacy, developing theoretical bounds on the relationship between data reuse and privacy leakage. Our proposed attack method leverages collusion between servers to significantly improve reconstruction quality compared to single-server scenarios.

In Chapter 5, we explore privacy vulnerabilities in federated diffusion models, which act as increasingly important high-quality generative models. While most FL privacy research focuses on involving external knowledge, we demonstrate how the unique training of diffusion models creates a new privacy attack paradigm. Our proposed two-stage inversion method successfully reconstructs training images despite the additional complexity introduced by the denoising process.

Finally, in Chapter 6, we summarize the conclusions made in this thesis, with regard to the research questions, and outline the limitations and future research directions.

SAMENVATTING

Federated learning (FL) is uitgegroeid tot een belangrijk paradigma in gedistribueerd machinaal leren, waarmee collaboratieve modeltraining over gedecentraliseerde apparaten mogelijk wordt gemaakt terwijl de dataprivacy behouden blijft. De privacybeschermende aard van FL - waarbij ruwe data lokaal op apparaten blijft en alleen modelupdates worden gedeeld - maakt het geschikt voor gevoelige domeinen zoals gezondheidszorg en financiën. Het gedecentraliseerde framework brengt echter fundamentele uitdagingen met zich mee die de betrouwbaarheid en adoptie ervan bedreigen. Dataheterogeniteit, beveiligingsdreigingen en privacyrisico's creëren kwetsbaarheden die robuuste oplossingen vereisen.

Om dergelijke kwetsbaarheden te bestuderen, beschouwt dit proefschrift twee soorten partijen: clients en servers. Clients fungeren als data-eigenaren die lokale berekeningen uitvoeren en alleen modelparameters delen, waardoor de privacy van ruwe data gewaarborgd blijft. Toch introduceren ze kwetsbaarheden door mogelijk kwaadwillend gedrag (zoals data/modelvergiftigingsaanvallen) of onbetrouwbare bijdragen door datakwaliteit. De server daarentegen, hoewel deze modelconvergentie faciliteert via aggregatie, vormt inherente privacyrisico's door mogelijk gevoelige clientinformatie af te leiden uit gedeelde gradients, zelfs zonder directe datatoegang. Deze twee partijen creëren een dubbel dreigingslandschap: clients kunnen modelprestaties compromitteren via adversariale manipulaties, terwijl servers vertrouwelijkheid schenden via reconstructiemethoden.

Dit proefschrift onderzoekt deze uitdagingen aan de hand van vier onderzoeksvragen, waarbij theoretische analyse wordt gecombineerd met empirische validatie om de robuustheid van FL te verbeteren. De eerste uitdaging komt voort uit de inherente dataheterogeniteit in FL-systemen, waarbij clients niet-identiek verdeelde data bezitten die modelconvergentie aanzienlijk kunnen beïnvloeden. De tweede kwetsbaarheid betreft beveiligingsdreigingen van kwaadwillende clients die het trainingsproces kunnen vergiftigen. Het derde onderzoek richt zich op privacyrisico's waarbij servers gevoelige data kunnen reconstrueren uit modelupdates, vooral bij hergebruik van trainingsdata, wat in strijd is met de kernprivacygaranties van FL. Ons vierde onderzoek breidt de privacyproblematiek uit naar opkomende federatieve generatieve modellen.

In Hoofdstuk 2 richten we ons op clientselectie in heterogene FL-omgevingen. Traditionele benaderingen negeren vaak deelnemers met unieke dataverdelingen, wat leidt tot verminderde modelprestaties. We onderzoeken de rol van gespecialiseerde clients met unieke dataverdelingen en ontwikkelen adaptieve strategieën om hun participatie te optimaliseren. We stellen een adaptieve selectiestrategie voor die dynamisch de featureafstanden evalueert op basis van de datakarakteristieken van de clientsmodellen en het globale model, waardoor optimale participatiepatronen over verschillende leerfasen gegarandeerd worden.

In Hoofdstuk 3 bestuderen we niet-gerichte aanvallen met beperkte kennis of middelen in FL-systemen. In tegenstelling tot bestaand werk dat uitgaat van sterke adversary-

capaciteiten, beschouwen we realistischer scenario's waarbij aanvallers geen trainingsdata of updatezichtbaarheid van goedaardige clients hebben. Het voorgestelde aanvalsframework demonstreert hoe zorgvuldig geconstrueerde synthetische updates modelprestaties effectief kunnen compromitteren. We complementeren dit hoofdstuk met een verdedigingsmechanisme dat dergelijke specifieke aanvallen identificeert en mitigeert.

In Hoofdstuk 4 onderzoeken we privacyrisico's in multi-server FL-omgevingen, waarbij clients deelnemen aan meerdere gelijktijdige leertaken en hun lokale trainingsdata hergebruiken. We analyseren hoe gradientupdates van verschillende taken geïntegreerd kunnen worden om dataprivacy te doorbreken, en ontwikkelen theoretische grenzen voor de relatie tussen datahergebruik en privacy-lekkage. Onze voorgestelde aanvalsmethode benut collusie tussen servers om reconstructiekwaliteit significant te verbeteren vergeleken met single-server scenario's.

In Hoofdstuk 5 exploreren we privacykwetsbaarheden in federatieve diffusiemodellen, die steeds belangrijkere hoogwaardige generatieve modellen vormen. Terwijl het meeste FL-privacyonderzoek zich richt op externe kennis, demonstreren we hoe de unieke training van diffusiemodellen een nieuw privacy-aanvalsparadigma creëert. Onze voorgestelde tweefasen-inversiemethode reconstrueert succesvol trainingsafbeeldingen ondanks de extra complexiteit van het denoisingsproces.

Tenslotte vatten we in Hoofdstuk 6 de conclusies van dit proefschrift samen met betrekking tot de onderzoeksvragen, en schetsen we de beperkingen en toekomstige onderzoeksrichtingen.

1

INTRODUCTION

Federated learning (FL) was introduced by Google researchers in 2016 to address privacy concerns in distributed machine learning, specifically targeting mobile applications like Google's Gboard. FL enables collaborative training on data from multiple parties without direct raw data sharing. The goal was to enable devices such as smartphones to collaboratively train machine learning models on the task of next-word prediction. This innovation was a result of the growing demand for privacy-preserving techniques, particularly as data privacy regulations like the General Data Protection Regulation (GDPR) were becoming more prominent. Over time, FL expanded beyond mobile applications and entered domains such as healthcare, finance, and autonomous systems. For example, in healthcare, it enables the training of models on sensitive medical data across hospitals without sharing patient records. In finance, FL is used to detect fraud across institutions without exposing proprietary customer data. In other words, it leverages the crowdsourcing of diverse data and the computational power of distributed devices, making it a scalable solution for training fast adaptive models on large, distributed datasets without sharing the raw data. Thus, as data privacy concerns and regulatory constraints grow, FL becomes a crucial technology for enabling secure and efficient machine learning applications across diverse domains.

There are two key parties in the context of FL systems: the data owners, i.e., clients, and the model aggregator, i.e., the server. Multiple clients (sometimes can be anonymous) are decentralized parties that participate in the training of a shared global model using their local data, such as smartphones, IoT devices, or organizational servers, each holding data that never leaves local devices. Clients iteratively train the model on their own datasets and send updates, such as gradient information, to a central server for aggregation. This decentralized approach helps to preserve data privacy but introduces challenges like varying computational capacities and unstable Internet connectivity. Moreover, clients may have heterogeneous data, which means the data of different owners varies a lot, e.g., data class and data quantity. It results in making the learning process more complicated and may affect the global model's performance unless handled properly.

In contrast to the clients, the server acts as the central coordinator that aggregates model updates from clients and manages the global model. The server's primary role is to initialize and distribute the machine learning model, collect updates from multiple clients, aggregate them, which often computes weighted averages, update the global model accordingly, and redistribute the updated model to the clients. Please note that aggregations and redistributions are repeated over multiple global rounds, e.g., hundreds or thousands. Thus, after each round of aggregation, the server distributes the updated model back to the clients for the next round of local training. The server must handle challenges such as ensuring authenticated, secure communication, mitigating the effects of data heterogeneity, and handling the scalability of multiple participants, as the number of clients might be very large. The FL models range from classifications, predictions, to generations, covering various modalities including image, text, audio, and time-series data.

Above, we introduced the general process of FL. However, one issue associated with a large number of participants is that some parties might be malicious (actively malicious or curious-but-honest). On the client side, malicious clients can intentionally change their data input or directly modify model updates. The goal could be degrading the overall performance of the global model, injecting a backdoor into the model, or just joining the learning without providing any real contribution. By submitting such manipulated updates, they pose significant threats to the system's security and reliability. Also, for the server side, the server has access to multiple updates from different clients, which may result in privacy attacks inferring confidential information, thus, breaking the initial advantage of protecting the data of FL. This privacy challenge is crucial for clients to consider. Additionally, there is the possibility of collusion attacks among clients or among servers, which can happen for both security and privacy attacks. These attacks are particularly dangerous because they exploit the decentralized nature of FL and are more difficult to detect, as the malicious behavior can be masked within normal client activity.

To create more secure and robust FL systems, this thesis explores the vulnerabilities concerning both the clients and the servers to improve the robustness. We investigate vulnerability-related issues of FL systems, which are specified by data heterogeneity, client security attacks, and the risk of privacy leakage by the server. We design and analyze novel methods that address the threats of these two kinds of parties: the server and the clients, respectively. In particular, for the clients, we design and analyze a method to effectively select suitable clients for round-wise participation, a method to attack the global model in a manner that requires less knowledge than state-of-the-art methods, as well as the corresponding defenses for the clients. As for the server, we analyze and develop a collusion data reconstruction method to break the local data privacy. We also extensively study this reconstruction in a different type of task, namely, generative models. We now introduce the background (Section 1.1, Section 1.2), vulnerabilities for the clients (Section 1.3), vulnerabilities for the server (Section 1.4), our research questions (Section 1.5), the structure of this thesis and its scientific contributions (Section 1.6), and our research methodology (Section 1.7).

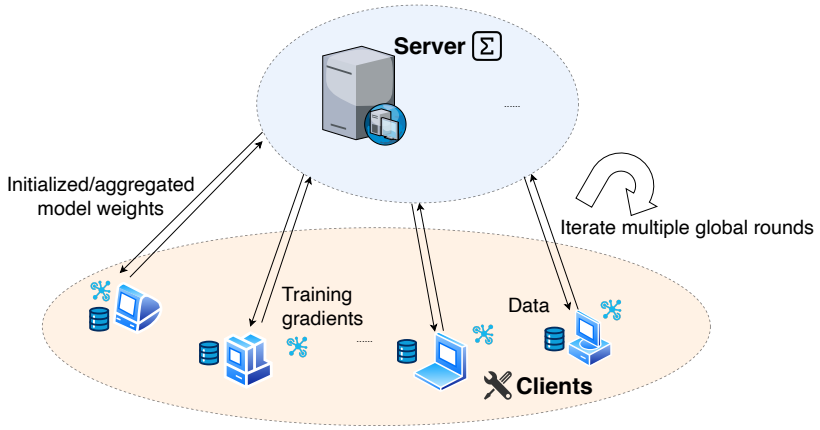


Figure 1.1: Federated learning with two kinds of parties: the servers (can be multiple) for model initialization, distribution, aggregation, and redistribution; the clients (multiple) as the data owners for training and model updating.

1.1. FEDERATED LEARNING

Federated learning is a distributed machine learning paradigm designed to train models across decentralized devices while keeping data in local devices. Unlike traditional approaches that centralize data in a single server, FL allows multiple participants to collaboratively train a shared model without exchanging their raw data. This decentralized approach enhances privacy and data security by minimizing the risk of exposing sensitive information about the raw data, as only model updates are communicated. The key components and functionalities are shown in Figure 1.1.

Specifically, the training process typically begins with the initialization of a global model by a central server, e.g, CNNs for image classification and LSTMs for text prediction. This model is then distributed to a set of selected clients [99], each of which trains the model locally using their own private data. The local training step involves running a few iterations of standard machine learning algorithms, such as gradient descent, on the client's dataset. Once the local training is complete, the clients send the updated model parameters (e.g., gradients or weights) back to the central server. Importantly, no raw data is ever shared and no clients are connected to each other, ensuring the privacy of each client's information. This local training phase may vary in length depending on the client's computational resources and data availability.

After receiving the local updates from clients, the server performs an aggregation step to update the global model. One of the most common aggregation methods is Federated Averaging (FedAvg) [96], where the server takes a weighted average of the clients' updates based on the size of their local datasets. This ensures that clients with larger amounts of data have a proportionally greater impact on the global model's updates. More advanced aggregation techniques may be used to handle data heterogeneity or defend against malicious updates. For instance, some methods include outlier detection or clipping extreme gradients to prevent any one client from intentionally or unintention-

ally influencing the model. Once aggregated, the updated global model is redistributed to the clients for another round of local training.

The process of local training, aggregation, and model distribution is repeated iteratively until the global model reaches the desired level of performance. Each iteration is often referred to as a global communication round, and the number of rounds depends on factors like model complexity, data distribution, and client participation. During training, some clients may drop out due to connectivity issues or lack of resources, but FL frameworks are designed to handle such scenarios through asynchronous updates or selective client participation, with the aim of converging to a high-quality global model. Over multiple global rounds, the global model gradually converges, leveraging the collective knowledge of all participating clients without compromising the privacy of individual data. This iterative process is key to achieving a well-trained model under the FL paradigm.

1.2. VULNERABILITIES IN FEDERATED LEARNING

This thesis researches various vulnerability issues in FL. Specifically, it is important to understand the three main aspects of distributed machine learning vulnerability: the challenges brought by diverse data distribution, the security risks of training many clients to gain a high-quality global model, and the privacy threats to clients' data from the server's view.

1.2.1. DATA HETEROGENEITY

Data heterogeneity is a fundamental challenge in FL, arising from the non-independent and identically distributed (*non-i.i.d.*) nature of data across clients. Unlike traditional machine learning, where data is centralized, federated learning operates in a decentralized environment where each client collects and stores data independently. This leads to significant variations in data distributions, including differences in feature spaces, data quality, and the quantity of data across clients. For example, in a FL system for predictive text on smartphones, the typing patterns and language preferences of users can vary widely, resulting in each client having a unique dataset [96]. This heterogeneity has a negative impact on the training process, compared with centralized training, as the global model must effectively generalize across these diverse datasets without access to the raw data, which can lead to challenges in model convergence and performance.

Addressing data heterogeneity in FL systems requires specialized techniques to ensure the global model remains robust and accurate. Strategies such as personalized models [29, 25], where the global model is fine-tuned for each client, and Federated meta-learning [137], which aims to create a model that can quickly adapt to new client data, have been developed to handle these variations. Aggregation methods also pay a crucial role in managing heterogeneity. Techniques like weighted averaging consider the size and relevance of each client's data during model updates [96]. Furthermore, algorithms are designed to identify and mitigate the impact of outlier data that could skew the global model.

Despite these efforts, data heterogeneity remains a critical area of research in FL, as it directly impacts the scalability and effectiveness of the system across real-world

applications where data diversity is inherent.

1.2.2. SECURITY THREATS

Malicious clients pose a significant threat to FL systems, where the training process relies on the cooperation and integrity of numerous participating entities owning data. This decentralization opens the door for adversarial participants to introduce harmful actions, such as data poisoning [135], where the training data of a client is maliciously pre-processed or biased by design, and model poisoning [118], where malicious clients intentionally submit incorrect or manipulated model updates to break the performance of the global model. These clients might even execute backdoor attacks [5] to skew the model in favor of specific outcomes, such as misclassifying certain data points to the wrong label. Detecting such malicious behavior is challenging because the server only receives the model updates, with no direct visibility into the raw data or local computations.

To mitigate the risks posed by malicious clients, FL systems incorporate various security measures and robust aggregation techniques. Secure aggregation protocols [10], for instance, ensure that the server can combine updates without learning any individual client's contribution, reducing the risk from adversarial attackers by malicious updates. Additionally, anomaly detection algorithms can identify suspicious behavior by monitoring the consistency and validity of client updates, flagging or excluding those that deviate significantly from expected patterns. Techniques such as Byzantine fault tolerance [9] are also employed, allowing the system to work correctly even when a portion of the clients behave adversarially, according to the pair-wised distance [98] or statistic features [157] compared with other clients.

Despite these defenses, the evolving stealthiness of attacks attracts ongoing research and development to enhance the robustness of FL systems against malicious clients, ensuring that the collaborative model training process remains secure and reliable.

1.2.3. PRIVACY LEAKAGES

While the server is designed to organize the collaborative model training without accessing raw data, it can still become a key point for privacy concerns despite it not having data. Although the server only receives aggregated model updates from clients, it is theoretically possible for an honest-but-curious or malicious server to infer sensitive information from these updates. For example, through model inversion attacks or by analyzing the gradients sent by clients, a server could potentially infer the membership (the participation of one data point) [26, 102, 149] of a specific data point, recover important features of the training dataset [97], or even reconstruct the original data [66, 31, 165]. These behaviors exactly violate privacy guarantees. This risk is particularly concerning in scenarios where the data is highly sensitive, such as in healthcare or financial applications, where even minimal data leakage could have severe consequences.

To address the potential privacy attacks conducted by the server, FL systems implement various privacy-preserving techniques. Differential privacy [27] is one popular method, where noise is added to the updates before they are sent to the server, making it much harder for the server to extract meaningful information about individual data points. The local training by the clients can also apply specific transformations of gra-

dients to hide precise information. Secure multi-party computation (SMPC) and homomorphic encryption (HE) are other advanced techniques that ensure data privacy even in the presence of a potentially untrustworthy server. These methods enable the server to perform necessary computations on encrypted data without ever accessing the data in its raw form.

Despite these defenses, the possibility of server-side privacy leakage remains a critical area of concern for privacy-preserving distributed learning. A variety of vulnerabilities and corresponding defenses introduced above pose potential threats to FL systems. Specifically, they are launched by either the clients, the server(s), or third parties. In this thesis, we focus on the most general cases: the clients and the servers.

1.3. ATTACKS AND DEFENSES FOR CLIENTS

Multiple state-of-the-art works have been proposed to demonstrate the damage to models caused by misbehaving participants in FL. It is worth mentioning that the attacks involved in this thesis are carried out during training time only by insider malicious participants (clients or server(s)). Threats are characterized according to the following criteria.

Security Attack Goal. Participants can maliciously contribute to FL frameworks for various goals ranging from provoking arbitrary damage to the system to targeted backdoor attacks. Attackers might try to prevent model convergence, degrade model accuracy, inject backdoors in the model, misclassify a certain type of input, or have access to the model without actually participating in the training process.

Number of Attackers. Adversarial behaviors can be carried out by individual participants separately or by multiple participants simultaneously. The latter can either be controlled by the same malicious party in order to bring more damage to the system (Sybil Attacks) or parties can collude to achieve a common adversarial goal by injecting different but coordinated malicious inputs, respectively.

Participants' Knowledge. The background knowledge of the attacker is a decisive factor of the attack impact. For instance, they may know other honest participants' training data or their training model parameters. They can be aware of the defense mechanism applied by the server to detect malicious activity, and so on.

Attack Duration. Some FL malicious behaviors may require to be carried out continuously through multiple rounds but with less perturbation between the benign and malicious updates to take effect. Normally, in this case, the attack is more stealthy. On the other hand, some adversarial goals are more straightforward to achieve, and thus, the attack can be carried out in a single round.

1.3.1. ATTACKS

Various attacks can undermine the effectiveness of the global model, such as free-riding, targeted, and untargeted attacks. A free-riding attack [35, 85, 86] is when clients participate in the FL process without contributing meaningful updates, essentially exploiting the benefits of the global model without computational costs or data privacy risks. This can slow down model convergence or even lead to lower model performance. In contrast, targeted attacks involve malicious clients intentionally manipulating their updates to bias the global model towards specific and harmful outcomes, such as causing the

model to misclassify certain types of data [92, 5], like misidentifying security threats in a cybersecurity application. Untargeted attacks [84, 33] are the most harmful and aim to degrade the overall performance of the global model without a specific target in mind. In these attacks, malicious clients may introduce random noise or incorrect gradients, causing general inaccuracies in the model's predictions. Both targeted and untargeted attacks pose significant threats to the reliability and robustness of FL systems.

In terms of the way to inject attack, data poisoning and model poisoning are two significant types of attacks that can attack the global model. Data poisoning happens when malicious clients introduce corrupted or biased data into their local training datasets. This manipulation aims to skew the model's learning process, leading to inaccurate or harmful predictions once the poisoned data influences the global model. For example, in a healthcare application, data poisoning could cause a model to misdiagnose a particular disease by altering the data it learns from. Model poisoning, on the other hand, involves directly manipulating the model updates—such as gradients or weights—sent to the central server. Malicious clients may modify these updates to introduce vulnerabilities or degrade the model's overall performance. Unlike data poisoning, which affects the model indirectly through malicious data, model poisoning directly attacks the model's parameters.

1.3.2. DEFENSES

To avoid attacks from the client's side, there are three impactful approaches: executing Byzantine robust aggregation methods, auditing abnormal gradients, and applying trusted hardware environments.

Byzantine Robust Aggregation. Byzantine robust aggregation is designed to safeguard the global model from the influence of malicious clients, often referred to as Byzantine clients. Clients operate independently and the server has limited visibility into their actions, some clients may behave adversarially or submit corrupted/low-quality updates, intentionally (e.g., data poisoning) or unintentionally (e.g., data heterogeneity). Byzantine robust aggregation methods aim to mitigate the impact of these unreliable clients by ensuring that the global model remains accurate and stable despite their presence. Techniques fall into two categories: statistic-based: median and trimmed mean, which filter out extreme or suspicious updates before they can affect the global model, and distance-based, e.g., Krum [9], mKrum [9], Bulyan [98], which determines the abnormality by pairwise distance of client updates. These approaches work by identifying and minimizing the influence of outlier updates, thereby protecting the integrity of the learning process.

Gradient Auditing. The purpose of this kind of protection mechanism is to detect and punish malicious behavior such as model poisoning or free-riding. In this case, the server is assumed to be trusted and it monitors statistical changes in model updates. The monitoring tries to point out suspicious updates and exclude them from the aggregation process or reduce their weights. Examples of such approaches are FoolsGold [132] and Gradient Norm Bounding [37].

Trusted Execution Environments. This a hardware-based protection mechanism that is mostly adapted to cross-silo [1] federated learning ecosystems where the local train-

[1] Cross-silo FL is an FL setting that involves a small number of relatively reliable clients, for example, multiple

ing code on the participants-side is implemented in a Trusted Execution Environment (TEE) such as Intel-SGX (e.g., [84]). This way, the code run by participants is attested by the server to make sure that the updates they send are not malicious.

1.4. ATTACKS AND DEFENSES FOR THE SERVER

Besides the crowd-sourcing clients, the server, which does not have data itself, may also be adversarial in the FL system. First, we introduce the four parts of the adversarial model. We then summarize the attacks and corresponding defense mechanisms, which are mostly applied by the client in this section.

Adversarial Goal. The goal of the servers is to steal private information from the clients instead of crafting the training model. The leakage of privacy is demonstrated by membership attacks, inference attacks, and data reconstruction attacks. These attacks pose different levels of leakage while sharing the same adversarial goal: breaking the privacy of the confidential training data of the clients by the server.

Number of Attackers. Similar to the adversarial clients, the number of attackers with regard to the server can be one or multiple. Generally, it is determined by the FL system architecture. Under the setting of single-server FL, the number of attackers is 1. However, under the setting of multiple-server FL, the number of attacks can be 1 (non-collusion among attackers) or more (with collusion).

Participants' Knowledge. The background knowledge of the attackers reflects the systematic settings. Normally, the honest-but-curious servers only have knowledge of global model parameters, the number of clients, and the updates transmitted by the clients, while the original data of the clients are confidential. The defense applied by the clients can be known or unknown. Further, active adversarial servers possess more knowledge but it is beyond the research of this thesis.

Attack Duration. The attack duration of the server attacks can be a single shot or multiple rounds, which can be either continuous rounds or separate rounds. Usually, multiple-round attacks are more effective than single-round attacks, due to the deeper extraction of clients' private information.

1.4.1. ATTACKS

As the server does not have original data for training, the honest-but-curious server may conduct privacy leakage attacks to infer confidential information from the clients [153, 178, 145]. Privacy concerns extend beyond the protection of raw data, as adversaries may exploit the system through various attacks, such as membership inference attacks [26], and data reconstruction attacks [66]. A membership attack is when an attacker attempts to determine whether a specific data point is part of a client's training dataset. This can lead to privacy threats, particularly in sensitive domains such as medical records. Inference attacks go a step further by attempting to infer sensitive attributes, e.g., financial risk preference, or information about the data based on the model updates received by the server, even if the raw data remains unknown. Data reconstruction attacks are the most dangerous, where attackers use gradients or model updates to reconstruct original data samples, completely violating the goal of FL to preserve privacy.

organizations collaborating to train a model.

1.4.2. DEFENSES

There are two possible ways to protect the server from privacy leakage attacks in FL. On the one hand, the client can implement local gradient transformation before sending it to the server. On the other hand, the basic FL protocol can be enhanced by prevention mechanisms that stop malicious behavior from occurring in the first place. In the following, we present some state-of-the-art mechanisms that were proposed to detect and prevent malicious contributions in FL frameworks.

Gradient Sparsification. This protection mechanism limits the effect of the privacy leakage attacks by pruning gradients that have a small magnitude, this is also referred to as gradient compression. It has been shown in a paper [88] that gradients can be compressed up to a factor of 300 while maintaining the same model accuracy. This approach was initially proposed to reduce communication bandwidth in distributed learning but was shown in [92] to be an effective way to protect against privacy leakage with a reasonable tradeoff between the accuracy-loss and protection-level.

Differential Privacy. Initially, differentially-private FL was proposed to reduce information leakage about local users' data [40]. However, since adding noise to user updates bounds their influence over the joint model, some state-of-the-art works [5, 92] considered using differential privacy as a protection mechanism to limit the damage caused by security attacks. This approach works by first clipping abnormal (e.g., amplified) and potentially malicious updates, then adding Gaussian or Laplacian noise to them. This simply reduces the impact of such attacks but does not entirely eliminate them. Also, adding user-level noise potentially reduces the accuracy of the trained models.

1.5. RESEARCH QUESTIONS

This thesis aims to provide an in-depth exploration of solutions for enhancing internal robustness in the presence of misbehaving clients and the server in modern federated learning systems, with enhanced scenarios such as extreme data distribution, less knowledge required for attack, and collusion attacks. The overarching research question addressed in this thesis is:

How can we detect the vulnerabilities of federated learning systems in order to improve their robustness?

To answer this overarching research question, we explore both the server(s) and the clients through the following related research questions. Specifically, we define two questions for the client, related to data heterogeneity and attacks on security, as well as two questions for privacy leakage by the server side in federated learning systems.

[RQ1] What is an effective client selection strategy for federated learning when specific data is exclusively owned by only a few clients? In practice, there exist multiple types of data heterogeneity. We bring up a common but overlooked scenario where specific data are owned and controlled by a few clients in the system called Mavericks. We are curious about the convergence performance of the FL system in the presence of Maverick. Considering the mainstream client selection methods based on the contribution of federated clients, we aim to investigate whether distance-based selection enhances efficient training in the presence of heterogeneous clients.

[RQ2] How can malicious clients launch an untargeted attack on federated learning systems without training data? Clients in the FL systems are multiple individuals

and are sometimes anonymous during the collaborative training process. Thus, it is difficult to guarantee the security of the system in such a crowdsourcing scenario. The most harmful attacks on the system are untargeted attacks, which destroy the functionality totally. Existing untargeted attacks require the adversary with large local data or the ability to eavesdrop on the benign updates. We aim to explore whether it is possible to launch a successful untargeted attack without the attacker having raw data and accessing the benign updates.

[RQ3] How can the servers collaboratively reconstruct the data from the clients who reuse data for multiple training processes? Due to the distributed nature, data owned by clients may be utilized in multiple training tasks (e.g., multiple servers) to gain more benefits for the clients. However, no existing work has studied and quantified the privacy leakage risk brought by joining multiple FL tasks with the same data. Our goal is to quantify the privacy leakage risk of data reuse, in terms of data reconstruction, in multi-server systems. Additionally, we aim to propose the aggregation method for multiple servers when negotiating the optimization direction for reconstruction.

[RQ4] How can a server apply a data reconstruction attack against generative models of federated learning? Generative models have played a crucial role in artificial intelligence systems, where diffusion models are one of the emerging generative models that train to reverse the process of adding noise to image samples from the training set. The training of diffusion models naturally fits FL. The reason is that the original centralized training of diffusion models use random sampling for inputting separate data samples, which can easily be processed in a distributed manner. As the general FL based on classification and regression models is vulnerable to data reconstruction attacks by the server, we are wondering whether the generative model server is also capable of recovering the training data of the clients. Thus, our goal is to design a gradient inversion attack that reconstructs original data by approximating the gradients calculated by training, against federated diffusion models.

1.6. THESIS OUTLINE AND CONTRIBUTIONS

In the next four technical chapters of this thesis (Chapters 2–5), we address our four research questions (RQ 1-4 of Section 1.5). Figure 1.7 illustrates the two types of vulnerability issues covered by these chapters. Figure 1.2 shows these issue within the clients and the servers. We now illustrate the contents and contributions of each chapter.

[Chapter 2] Federated learning under heterogeneous clients. In this chapter, we address RQ1 by designing a client selection method to simulate a real-world example application case of client heterogeneity. To achieve fast convergence with heterogeneous clients, selecting those who can contribute effectively is essential. A key but overlooked case, Mavericks, represents a certain data distribution (e.g., children's hospitals with pediatric cardiology data). The chapter highlights the importance of Mavericks' contributions, which the common contribution-based method, *Shapley Value*, has underestimated. Thus, we introduce FEDEMD, an adaptive strategy leveraging the Wasserstein distance, which accelerates convergence by prioritizing Mavericks when beneficial. FEDEMD improves neural network convergence speed and maintains consistent performance across varying levels of heterogeneity.

The research objective and key contribution are illustrated in Figure 1.3. This chapter

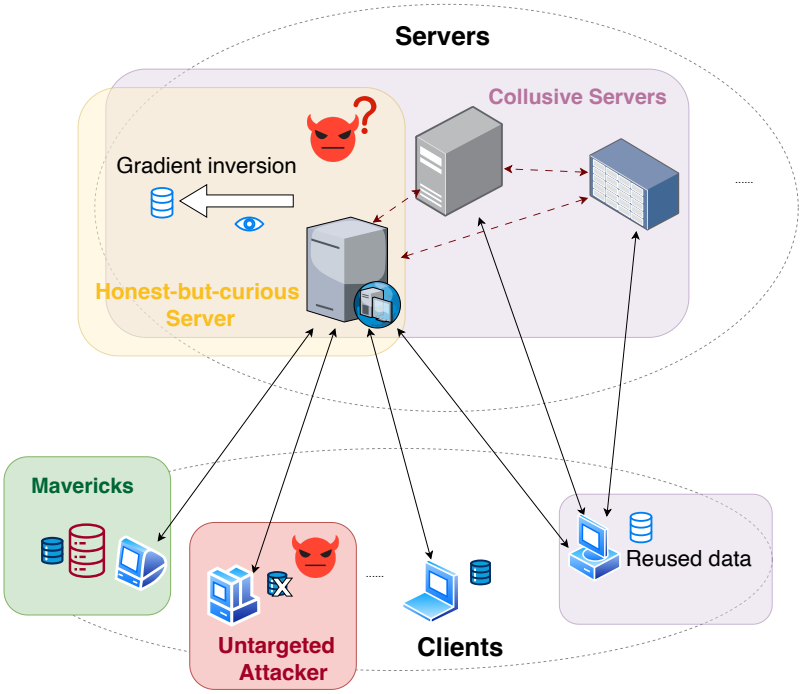


Figure 1.2: Overview of the proposed vulnerability issues covered by all chapters.

is based on the following two publications, where the former is a workshop paper and the latter is the extended version as a full conference paper:

Jiyue Huang, Chi Hong, Yang Liu, Lydia Y. Chen, and Stefanie Roos. "Tackling mavericks in federated learning via adaptive client selection strategy." In International Workshop on Trustable, Verifiable and Auditable federated learning in Conjunction with AAAI, vol. 2023. 2022.

Jiyue Huang, Chi Hong, Yang Liu, Lydia Y. Chen, and Stefanie Roos. "Maverick matters: Client contribution and selection in federated learning." In 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 269-282. Cham: Springer Nature Switzerland, 2023.

[Chapter 3] Data-free Untargeted Attack in Federated Learning. In this chapter, we address RQ2 by designing two variants of a data-free adversary to launch an untargeted attack in the training process of FL systems. Attacks on FL can greatly reduce model quality. However, current untargeted attacks are often unrealistic because they assume attackers know every benign client update or have access to large training datasets. This chapter introduces a data-free untargeted attack (DFA) that generates adversarial models without eavesdropping or needing large training data. We propose two variants,

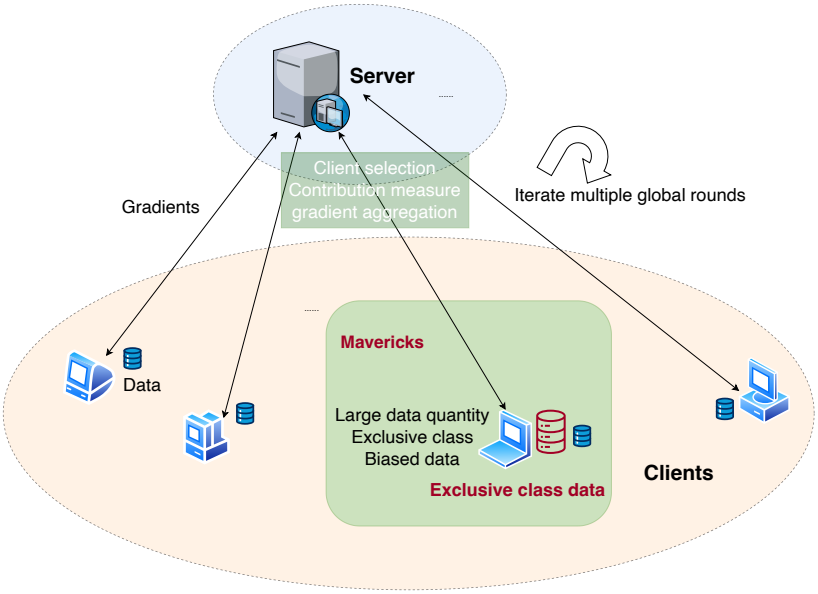


Figure 1.3: Overview of the proposed Federated Earth Mover's Distance (FEDEMD) in Chapter 2.

DFA-R and DFA-G, which balance stealthiness and effectiveness. DFA-R minimizes global model prediction confidence, while DFA-G directs the model output toward a specific class. Experiments on three different datasets show that DFA achieves similar or even higher attack success rates under existing defense mechanisms than state-of-the-art attacks.

To counter these attacks, we introduce REFD, a defense that detects biased or low-confidence updates using a reference dataset. REFD outperforms current defenses by filtering malicious updates and preserving high global model accuracy.

The research objective and key contribution are also illustrated in Figure 1.4. This chapter is based on the following publication:

Jiyue Huang, Zilong Zhao, Lydia Y. Chen, and Stefanie Roos. "Fabricated Flips: Poisoning federated learning without Data." In 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 274-287. IEEE, 2023.

[Chapter 4] Privacy Risk of Data Reuse in Multi-server Federated Learning In this chapter, we address RQ3 by proposing a gradient inversion attack that includes a colluded attack under a multi-server FL system. As shown by various privacy attacks, the model updates shared by the clients to the server leak information about the dataset. While privacy in single data use is well-studied, users often provide the same data for multiple tasks. We focus on data reuse in scenarios with multiple colluding servers, whether for the same or different training tasks. We introduce Collusion Gradient In-

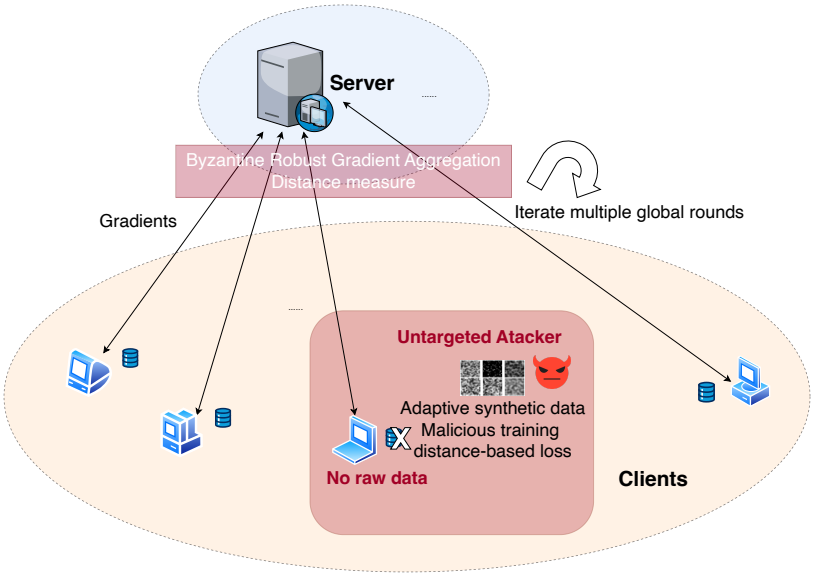


Figure 1.4: Overview of the proposed Data-Free Untargeted Attack (DFA) in Chapter 3.

version (CGI), an attack that combines gradients from repeated data use to reconstruct the original data. We analyze a theoretical bound on how privacy leakage grows with data reuse for the same task. Additionally, we show that Nash bargaining games effectively determine aggregation weights across tasks. Our experiments confirm that CGI improves the quality of reconstructed images compared to single-server attacks, both with and without defense mechanisms.

The research objective and key contribution are also illustrated in Figure 1.5. This chapter is based on the following publication:

Jiyue Huang, Lydia Y. Chen, and Stefanie Roos. "On Quantifying the Gradient Inversion Risk of Data Reuse in federated learning Systems." In the 43rd International Symposium on Reliable Distributed Systems (SRDS), 2024.

[Chapter 5] Gradient Inversion Attack in Federated Diffusion Models In this chapter, we address RQ4 by studying the key difference between classification models and diffusion models in terms of privacy. We present our fusion optimization method of Gradient Inversion of Diffusion Models (GIDM) consisting of two phases. Training effective diffusion models require large amounts of real data, typically distributed across multiple parties. Due to privacy concerns, these parties often cannot share raw data, limiting the use of diffusion models. We propose that diffusion models can be trained federatedly, with data kept on-premise. Our main focus is on evaluating privacy risks through gradient inversion attacks to reconstruct training data. We introduce a two-phase optimization method, GIDM, where the first phase uses the diffusion model as prior knowledge

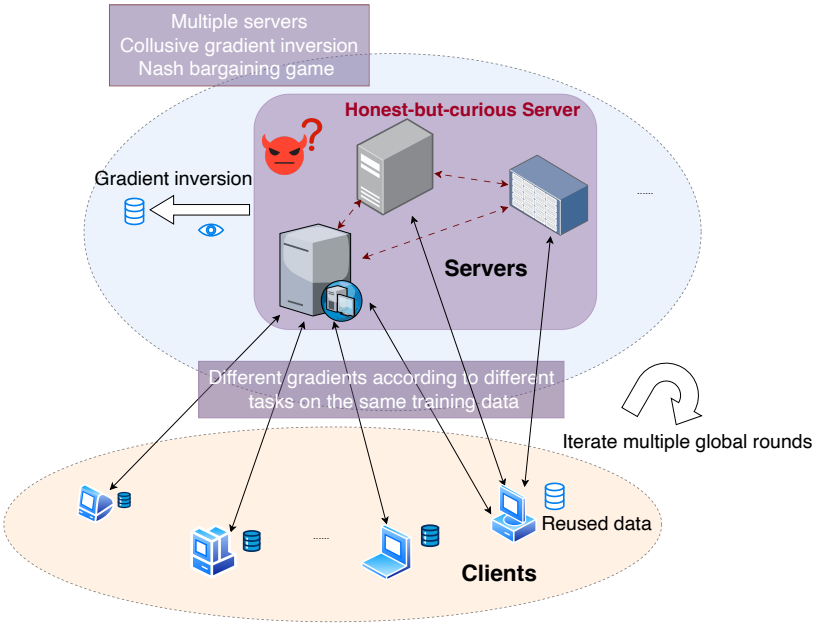


Figure 1.5: Overview of the proposed Collusion Gradient Inversion (CGI) in Chapter 4.

to limit the inversion search space, followed by pixel-level fine-tuning. GIDM nearly replicates the original images. Given that the original data, the sampling step, and the noise are all unknown to the adversarial server, we design a triple optimization method to adapt and optimize all three factors simultaneously.

The research objective and key contribution are also illustrated in Figure 1.6. This chapter is based on the following publication:

Jiyue Huang, Chi Hong, Lydia Y. Chen, and Stefanie Roos. "Gradient Inversion of Federated Diffusion Models." In the 20th International Conference on Availability, Reliability and Security (ARES), 2025.

In summary, we address four key research questions (RQ1–RQ4) related to the vulnerability of federated learning systems. Chapter 2-5 address critical aspects of data heterogeneity, privacy, and security in FL systems. Our proposed methods provide insights into the vulnerability of different FL systems and introduce innovative solutions, like FEDEMD for effective client selection in heterogeneous environments, data-free untargeted attacks with novel defense mechanisms, and gradient inversion attacks for multi-server systems and federated diffusion models. These contributions highlight the importance of adaptive strategies to enhance convergence speed, maintain privacy, and protect against emerging threats in FL. Overall, the chapters emphasize the need for robust, privacy-preserving techniques in complex, real-world FL scenarios.

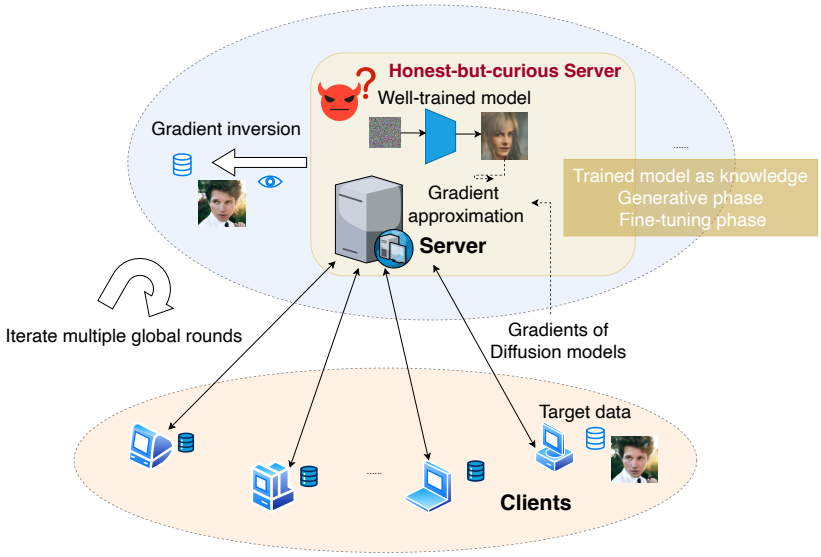


Figure 1.6: Overview of the proposed Gradient Inversion on Diffusion Models (GIDM) in Chapter 5.

1.7. RESEARCH METHODOLOGY

As we research this domain, the two primary research methodologies applied are the following:

- **Theoretical Methods:** This research methodology explores the theoretical foundations of how federated learning models manage global model convergence under disturbance. Researchers use mathematical tools such as formal proofs to examine the inherent limitations and capabilities of these models in heterogeneous clients' environments and quantify the privacy leakage bound. This theoretical analysis provides valuable insights into the guarantees and trade-offs associated with various data distributions, which provides a deeper understanding of their effectiveness and limitations.
- **Experimental Methods:** This approach focuses on evaluating the effectiveness of proposed methods to enhance the robustness of heterogeneous federated learning models against privacy and security threats. Researchers conduct extensive experiments using established benchmarks to measure the performance of their methods. These studies typically involve diverse datasets (including both real data and synthetic data) and adversarial techniques to simulate systems with malicious clients. By examining the model's performance across varying levels and patterns of attackers, researchers can compare and improve their techniques, ensuring their methods are robust and effective in handling malicious clients and servers.

In this thesis, we explore both experimental and theoretical aspects, addressing each research question by designing, implementing, analyzing, proving, and evaluating the

1

RQ	Method	Source Code Repository	Implemented Parties for Attack
1	FEDEMD	[57]	Single server, multiple clients
2	DFA, REFD	[55]	Single server, multiple clients
3	CGI-S, CGI-D	[54]	multiple servers, single client
4	GIDM	[56]	Single server, single client

Table 1.1: An overview of source code for each research question (proposed method) in this thesis.

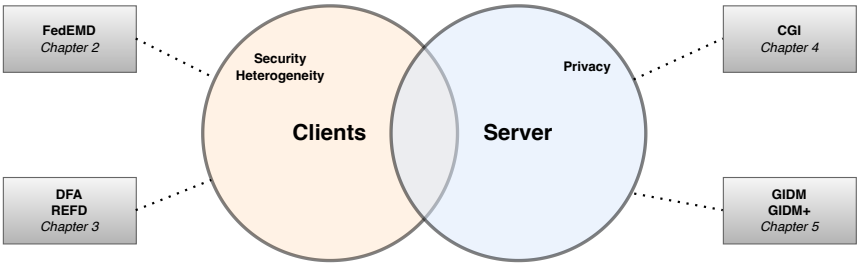


Figure 1.7: The methods proposed in the thesis and their relation to each chapter with respect to the research questions.

impact of adversaries to enhance the robustness of FL systems. Specifically, on the theoretical side, Chapter 2, which explores RQ1, aims at achieving high accuracy and fast convergence. It analyzes the fairness of contribution evaluation and then derives the theoretical convergence bound for the proposed client selection method. We first provide a clear definition of the Maverick, associated with key properties. These assist the analysis of this application case. To derive the convergence bound, we follow the assumptions of related studies, e.g., *L-smooth* and μ*-strongly convex* on the objective functions. In order to study the heterogeneity, our theoretical analysis also includes the factor to represent the heterogeneity degree. Chapter 4, which explores RQ3, aims at analyzing the reconstruction bound according to the number of data reuses that quantifies the privacy leakage and applies game theory for aggregation. We discuss the reconstruction bound by different loss functions of the FL network, i.e., linear or polynomial (reflecting one kind of convolutional neural network).

On the experimental side, we implement all our proposed methods in Python, the leading language for machine learning research. We explore PyTorch as the primary tools and libraries for implementing our algorithms. We include at least three replication experiments to mitigate the impact of randomness brought by common machine learning training processes, e.g., stochastic gradient descent. To develop our method, we initialize our FL code base as the foundation for easy comparison and adaption (details see Table 1.1). Subsequently, we implement and deploy our method on top of the code base. The code base for this thesis is a simulation system of a distributed machine learning framework, instead of physically distributed devices. Note that in this simulation, we execute the training and aggregation on the same machine, while demonstrating the ac-

Dataset	Task Type	Number of Samples	Number of Classes	RQ
MNIST	classification	60K	10	1, 3
Fashion-MNIST	classification	60K	10	1, 2
CIFAR-10	classification	50K	10	1, 2
CIFAR-100	classification	50K	100	3
STL-10	classification	500	10	1, 3
SVHN	classification	60K	10	2
LFW	classification	13K	2	3
CelebA	generation	202K	N.A.	4
LSUN-bedroom	generation	120K	N.A.	4

Table 1.2: An overview of public datasets used for each research question (proposed method) in this thesis.

tions of the clients and the server alternatively and sequentially. The reason is that the research questions and methods of this thesis only consider clients' data and updates by algorithm, rather than hardware manipulation or communicational eavesdropping. By this means, we are able to focus on the vulnerability study in terms of data with a large number of clients, e.g., 100.

Each chapter includes a detailed description of the experiments conducted. Our evaluation is carried out by Alienware- Aurora-R13 with Ubuntu 20.04. The machine is equipped with 64G memory, GeForce RTX 3090 GPU, and 16-core Intel i9 CPU. Each of the 8 P-cores has two threads, hence each machine contains 24 logical CPU cores in total. In addition to implementing and designing the methods, we thoroughly evaluate the performance of our proposed approaches on a diverse set of public and widely used datasets in deep learning. Table 1.2 provides a comprehensive overview of the datasets employed in each research question. For each research question, we evaluate with *non-i.i.d.* data to simulate real-world data distributions. The level of heterogeneity varies according to different tasks. To further assess the performance of our proposed methods, we evaluate their performance using the most common metrics: test accuracy for the global model, attack success rate, and defense pass rate for security attacks by clients, and standard image similarity metrics (SSIM, MSE, LPIPS, PSNR) for privacy threats by servers. We promote the reproducibility of our proposed methods and experiments by providing detailed explanations of the methods in each chapter.

This work advances FL robustness by addressing critical vulnerabilities in both client-server dynamics, offering practical solutions for secure, privacy-preserving collaborative learning in real-world applications. The proposed methods balance performance, security, and scalability, shedding light on future robust adoption of FL in sensitive domains.

2

FEDERATED LEARNING UNDER HETEROGENEOUS CLIENTS

Federated learning enables collaborative learning between parties, called clients, without sharing the original and potentially sensitive data. To ensure fast convergence in the presence of such heterogeneous clients, it is imperative to timely select clients who can effectively contribute to learning. A realistic but overlooked case of heterogeneous clients are Mavericks, who monopolize the possession of certain data types, e.g., children's hospitals possess most of the data on pediatric cardiology. In this chapter, we address the importance of Mavericks and tackle its challenges by exploring two types of client selection strategies. First, we show theoretically and through simulations that the common contribution-based approach, Shapley Value*, underestimates the contribution of Mavericks and is hence not effective as a measure to select clients. Then, we propose* FEDEMD*, an adaptive strategy with competitive overhead based on the Wasserstein distance, supported by a proven convergence bound. As* FEDEMD *adapts the selection probability such that Mavericks are preferably selected when the model benefits from improvement on rare classes, it consistently ensures fast convergence in the presence of different types of Mavericks. Compared to existing strategies, including* Shapley Value*-based ones,* FEDEMD *improves the convergence speed of neural network classifiers with FedAvg aggregation by 26.9% and its performance is consistent across various levels of heterogeneity.*

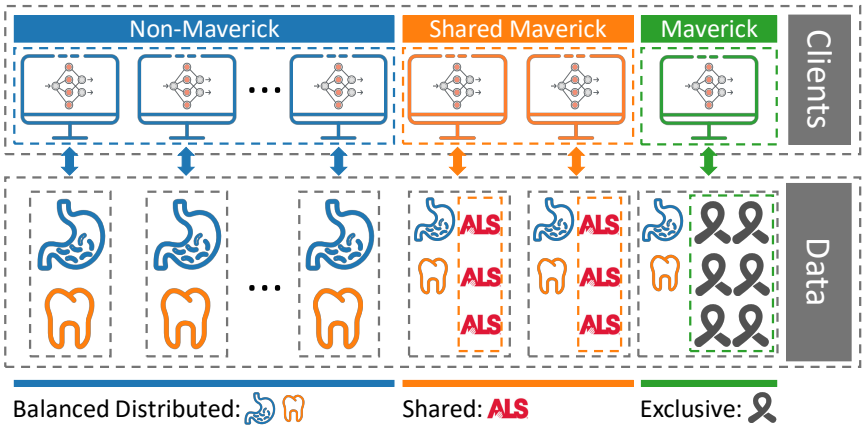


Figure 2.1: Mavericks (or Shared Mavericks within a small group) are specific data owners that exclusively own both rare class data and general class data.

2.1. INTRODUCTION

Federated Learning enables clients (either individuals or institutes who own data) to collaboratively train a global machine learning models by exchanging locally trained models instead of data [96, 91]. Thus, Federated Learning allows the training of models when data cannot be transferred to a central server and is hence often a suitable alternative for medical research and other domains, such as finance, with high privacy requirements. The effectiveness of FL, in terms of accuracy and convergence, highly depends on how the local models are selected and aggregated.

In FL, clients tend to own heterogeneous datasets [80] rather than identically and independent distributed (*i.i.d.*) ones. The prior art has recently addressed the challenge of heterogeneity from either the perspective of skewed distribution [179] or skewed quantity [140] among all clients. However, a common real-world scenario, where one or a small group of clients monopolize the possession of a certain class, is universally overlooked. For example, in the widely used image classification benchmark, Cifar-10 [72], most people can contribute images of cats and dogs. However, deer images are bound to be owned by comparably few clients. We call these types of clients *Mavericks*. Another relevant example, shown in Fig. 2.1, arises from learning predictive medicine from clinics who specialize in different conditions, e.g., AIDS and Amyotrophic Lateral Sclerosis, and own data of exclusive disease types. Without involving Mavericks into the training, it is impossible to achieve high accuracy on the classes for which they own the majority of all training data, e.g., rare diseases.

Given its importance, it is not well understood when to best involve Mavericks in FL training, because the effectiveness of FL, in terms of accuracy and convergence, highly depends on how those local models are selected and aggregated. The existing client selection[1] considers either the contribution of local models [13] or difference of data distributions [99]. The contribution-based approaches select clients based on contribution

[1] Note that here we only discuss selection on statistical challenges, the selections considering system resources, e.g., unreliable networks are left for other works.

scores preferring clients with higher scores [34], whereas the distance-based methods choose clients based on the pairwise feature distance. Both types of selection methodologies have their suitable application scenarios and it is hard to weigh the benefits of one over the other in general.

In this chapter, we aim to effectively select Mavericks in FL so that users are able to collaboratively train an accurate model in a low number of communication rounds.We first explore *Shapley Value* as a contribution metric for client selection. Although *Shapley Value* is shown to be effective in measuring contribution for the *i.i.d.* case, it is unknown if it can assess the contribution of Mavericks and effectively involve them via the selection strategy. Moreover, we propose FEDEMD, which selects clients based on Wasserstein distance [4] of the global distribution and current distribution. As FEDEMD adapts the selection probability such that Mavericks are preferably selected when the model benefits from improvement on rare classes, it consistently ensures the fast convergence in the presence of different types of Mavericks.

Our main **contributions** for this work can be summarized as follows. *i)* We explore the effectiveness of both contribution-based and distance-based selection strategies for Mavericks. *ii)* Both our theoretical and empirical results show that the contribution of clients with skewed data or very large data quantity is measured below average by *Shapley Value*. *iii)* We propose FEDEMD, a novel adaptive client selection based on the Wasserstein distance, derive a convergence bound, and show that it significantly outperforms SOTA selection methods in terms of convergence speed across different scenarios of Mavericks.

2.2. RELATED STUDIES

Contribution Measurement. Although the self-reported contribution evaluation [34] is easy to implement, it is fragile to dishonest parties. Besides, existing work on contribution measurement can be categorized into two classes: *i)* local approach: clients exchange the local updates, i.e., model weights or gradients, and measure the contribution of each other, e.g., by creating a reputation system [68], and *ii)* global approach: all clients send all their model updates to the *federator* who in turn aggregates and computes the contribution via the marginal loss [110, 144]. Prevailing examples of globally measuring contribution are Influence [110] and *Shapley Value* [144, 121]. The prior art demonstrates that *Shapley Value* can effectively measure the client's contribution for the case when client data is *i.i.d.* or of biased quantity [121]. A work [141] has proposed federated *Shapley Value* to capture the effect of participation order on data value. The experimental results indicate that *Shapley Value* is less accurate in estimating the contribution of heterogeneous clients than for *i.i.d.* cases. However, there is no rigorous analysis on whether *Shapley Value* can effectively evaluate the contribution from heterogeneous users with skewed data distributions.

Client Selection. Selecting clients within a heterogeneous group of potential clients is key to enabling fast and accurate learning based on high data quality. The state-of-the-art client selection strategies focus on the resource heterogeneity [107, 60] or data heterogeneity [13, 15, 80]. In case of data heterogeneity, which is the focus of our work, selection strategies [15, 41, 13] gain insights on the distribution of clients' data and then select them in specific manners. Goetz et. al [41] apply active sampling and Cho et. al

[15] use Power-of-Choice to favor clients with higher local loss. TiFL [13] considers both resource and data heterogeneity to mitigate the impact of stragglers and skewed distributions. TiFL applies a contribution-based client selection by evaluating the accuracy of selected participants each round and chooses clients of higher accuracy. FedFast [99] chooses classes based on clustering and achieves fast convergence for recommendation systems. One recently work [95] focuses on reduce wall-clock time for convergence under high degrees of system and statistical heterogeneity. However, there is no selection strategy that addresses the Maverick scenario.

2.3. FEDERATED LEARNING WITH MAVERICKS

In this section, we first formalize a Federated Learning framework with Mavericks. Then we rigorously analyze the contribution of clients based on *Shapley Value* and argue that the contribution of Mavericks is underestimated by the *Shapley Value*, which leads to a severe selection bias and a suboptimal integration of Mavericks into the learning process.

Suppose there are a total of N clients in a federated learning system. We denote the set of possible inputs as $\mathscr{X}$ and the set of L class labels as $\mathscr{Y} = \{1, 2, ..., L\}$. Let $f : \mathscr{X} \rightarrow \mathscr{P}$ be a prediction function and ω be the learnable weights of the machine learning tasks, the objective is then defined as: $\min \mathscr{L}(\boldsymbol{\omega}) = \min \sum_{l=1}^{L} p(y=l)\mathbb{E}_{\boldsymbol{x}|y=l}\left[\log f_l(\boldsymbol{x}, \boldsymbol{\omega})\right]$.

The training process of a FL system has the following steps[2]:

1. INITIALIZATION. Initialize global model $\boldsymbol{\omega}_0$ and distribute it to the available clients, i.e., a set $\mathscr{C}$ of N clients.

2. CLIENT SELECTION. Enumerate the K clients $\mathscr{C}(\pi, \boldsymbol{\omega}_r)$, selected in round r with selection strategy π, by $C_1, \ldots, C_K$.

3. UPDATE AND UPLOAD. Each client C_k selected in round r computes local updates $\boldsymbol{\omega}_r^k$ and the *federator* aggregates the results. Concretely, with η being the learning rate, C_k updates their weights in the r-th global round by: $\boldsymbol{\omega}_r^k = \boldsymbol{\omega}_{r-1} - \eta \sum_{l=1}^{L} p^k(y=l)\nabla_{\boldsymbol{\omega}}\mathbb{E}_{\boldsymbol{x}|y=l}\left[\log f_l(\boldsymbol{x}, \boldsymbol{\omega}_{r-1})\right]$.

4. AGGREGATION. Client updates are aggregated to one global update. The most common aggregation method is quantity-aware FedAvg, defined as follows with n^k indicating the data quantity of C_k: $\boldsymbol{\omega}_r = \sum_{k=1}^{K} \frac{n^k}{\sum_{k=1}^{K} n^k} \boldsymbol{\omega}_r^k$.

To facilitate our discussions, we also define the following:
Local Distribution: the array of all L class quantities $\mathscr{D}^i(y=l), l \in \{1, .., L\}$ owned by client C_i.
Global Distribution: the quantity of all clients' data by class as $\mathscr{D}_g = \sum_{i=1}^{N} \mathscr{D}^i(y=l), l \in \{1, .., L\}$.
Current Distribution at R: by summing up the class quantity of all clients' data reported, which have been chosen up to round R as: $\mathscr{D}_{\rfloor}{}^R = \sum_{t=1}^{R} \sum_{C_k \in \mathscr{K}^t} \mathscr{D}^{C_k}$.

[2] Here we assume all the clients are honest. Since we focus on the statistical challenge, the impact of unreliable networking and insufficient computation resources is ignored.

Definition 1 (Maverick)**.** Let Y_{Mav} be the set of class labels that are owned by Mavericks. In the extreme case, there is one Maverick but it might also be a small set of Mavericks jointly owning the same class. For a client C_k, let q_k^{Mav} be the fraction of C_k's data that has label Y_{Mav}. Then:

$$q_k^{Mav} = \begin{cases} 1-\epsilon, \text{if } C_k \text{ is a Maverick} \\ \epsilon, \text{if } C_k \text{ is not a Maverick.} \end{cases} \tag{2.1}$$

An exclusive Maverick is one client that owns one or more classes exclusively. A shared Maverick is a small group of clients who jointly own one class exclusively. That is:

$$D_i = \begin{cases} \{\{x_l, y_l\}^i_{l \in Y_{Mav}}, \{x_l, y_l\}^i_{l \notin Y_{Mav}}\}, \text{if } C_i \text{ is a Maverick} \\ \{x_l, y_l\}^i_{l \notin Y_{Mav}}, \text{if } C_i \text{ is not a Maverick,} \end{cases} \tag{2.2}$$

where D_i denotes the dataset for C_i, $\{x_l, y_l\}^i$ denotes the dataset in C_i with label l.

In the rest of this chapter, we assume the global distribution organized by the server's preprocessing has high similarity with the real-world (test dataset) distribution, which is balanced, so that data $\{x_l, y_l\}_{l \notin Y_{Mav}}$ are evenly distributed across all parties, whereas $\{x_l, y_l\}_{l \in Y_{Mav}}$ either belong to one exclusive Maverick or are evenly distributed across all shared Maverick parties. We focus our analysis on exclusive Mavericks since shared Maverick are a straightforward extension. Based on the assumptions above, we obtain the following properties for Mavericks.

Property 1. Because the data distribution is balanced, Mavericks have a larger data quantity than non-Mavericks. Concretely, let n^n be the data quantity of a non-Maverick. Let n^m be the quantity for Mavericks, then $n^m = ((N/m-1) \times Y_{Mav} + L) \times n^n$, where m is the number of Mavericks.

Property 2. Assume $N > 2$, the KL divergence of a Maverick's data to the normalized global distribution is expected to be larger than for a non-Maverick due to their specific distribution, i.e., $D_{KL}(\mathscr{P}_\}||\mathscr{P}_\Updownarrow) > D_{KL}(\mathscr{P}_\}||\mathscr{P}_\setminus)$, where $\mathscr{P}_\Updownarrow$, $\mathscr{P}_\setminus$ are the data distribution with class labels for Maverick and non-Maverick, where $\mathscr{P}_\}$ denotes for global distribution.

Definition 2 (*Shapley Value*)**.** Let $\mathscr{K} = \mathscr{C}(\pi, \boldsymbol{\omega}_r)$ denote the set of clients selected in a round including C_k, $\mathscr{K} \setminus \{C_k\}$ denote the set $\mathscr{K}$ without C_k. *Shapley Value* of C_k is:

$$SV(C_k) = \sum_{S \subseteq \mathscr{K} \setminus \{C_k\}} \frac{|S|!(|\mathscr{K}| - |S| - 1)!}{|\mathscr{K}|!} \delta C_k(\mathscr{S}). \tag{2.3}$$

Here we let $\delta C_k(S)$ be the Influence [110]. Influence can be defined on loss, accuracy, etc., here we apply the most commonly used loss-based Influence written as $Inf_S(C_k)$ for set C_k.

Lemma 1. *Based on Shapley Value in Eq. 2.3, the difference of Maverick C_m's and non-Maverick C_n's Shapley Value is:*

$$\begin{aligned} SV(C_m) - SV(C_n) = \frac{1}{|\mathcal{K}|!}\Big(& (|\mathcal{K}|-1)!(\mathcal{L}(C_m) - \mathcal{L}(C_n)) \\ & + \sum_{S \subseteq S_-} |S|!(|\mathcal{K}|-|S|-1)!(Inf_S(C_m) - Inf_S(C_n)) \\ & + \sum_{S \subseteq S_+} |S|!(|\mathcal{K}|-|S|-1)!(Inf_S(C_m) - Inf_S(C_n))\Big), \end{aligned} \tag{2.4}$$

with $S_- = \mathcal{K} \setminus \{C_n, C_m\}$, $S_+ = \mathcal{K} \setminus \{C_n, C_m\} \cup C_M$, $C_M \in \{C_n, C_m\}$. Note that we simplify $Inf_{S \cup C_i}(C_i)$ as $Inf_S(C_i)$ for readability.

Comparison of *Shapley Value* and Influence: Rather than considering Influence for the complete set of K clients, Eq. 2.4 only considers Influence on a subset S. However, our derivations for Influence are independent from the number of selected clients and remain applicable for subsets S, meaning that indeed the second and the third term of Eq. 2.4 are negative. Similarly, the first term is negative as the loss for clients only owning one class is higher. However, *Shapley Value* obtains higher values for *i.i.d.* clients with large data sets than Influence since $\mathcal{L}(C_m) - \mathcal{L}(C_n)$ increases if the distance between C_m's distribution and the global distribution is small, in line with a previous work [58].

Property 3. *Shapley Value* and Influence share the same trend in contribution measurement for Mavericks.

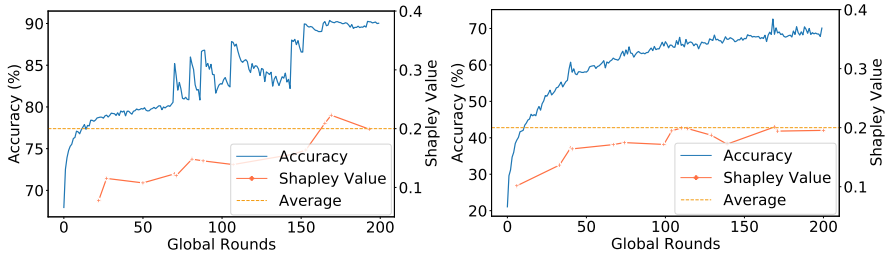
Theorem 1. Let C_m and C_n be a Maverick and a non-Maverick client, respectively, and denote by $SV_t(C_k)$ the Shapley value of C_k in round r. Then $SV_1(C_m) < SV_1(C_n)$ and $SV_t(C_m)$ converges towards $SV_t(C_n)$.

We present the empirical evidences of how one or multiple Mavericks are measured by *Shapley Value*. We here focus on single exclusive Mavericks and leave multiple Mavericks, shared and exclusive, for our in-depth experimental evaluation in the supplementary material. We use Fashion-MNIST (Fig. 3.6a) and Cifar-10 (Fig. 3.6b) as learning scenarios and use random client selection with FedAvg.

Fig. 2.2 shows the global accuracy and the relative *Shapley Value* during training, with the average relative *Shapley Value* of the 5 selected clients out of 50 indicated by the dotted line. The contribution is only evaluated when a Maverick is selected. Looking at Fig.(3.6a), (3.6b), The *Shapley Value* of the Maverick indeed increases over time but remains below average until round 160, providing concrete evidence of **Theorem. 1**. Furthermore, the accuracy increases when a Maverick is selected, indicating that Mavericks contribute highly to improving the model. Thus, assigning Mavericks a lower contribution measure is unreasonable, especially in the early stage of the learning process. All of the empirical results are consistent with our theoretical analysis.

2.4. FEDEMD: AN ADAPTIVE STRATEGY FOR CLIENT SELECTION WITH MAVERICKS

In this section, we propose a novel adaptive client selection algorithm FEDEMD, which enables FL systems with Mavericks to achieve faster convergence compared with SOTA



(a) FMNIST-Maverick (b) Cifar-10-Maverick

Figure 2.2: Global model accuracy and relative *Shapley Value* measured for Mavericks during training on FMNIST and Cifar-10.

methods, including *Shapley Value*-based ones. The key idea is to assign a higher probability for selecting Maverick clients initially to accelerate convergence; later we reduce the selection probability to avoid skewing the distribution towards Maverick classes. To measure the differences in data distributions, we adopt Wasserstein distance (EMD) [4], which is used to characterize weight divergence in FL [174]. The Wasserstein distance (EMD) is defined as:

$$\mathrm{EMD}\,(P_r, P_\theta) = \inf_{\gamma\in\Pi} \sum_{x,y} \|x-y\|\gamma(x,y) = \inf_{\gamma\in\Pi} \mathbb{E}_{(x,y)\sim\gamma}\|x-y\|, \tag{2.5}$$

where $\Pi(P_r, P_\theta)$ represents the set of all possible joint probability distributions of P_r, P_θ. $\gamma(x,y)$ represents the probability that x appears in P_r and y appears in P_θ.

Algorithm 1: FEDEMD Clients Selection

Data: $\mathscr{D}^i$ for $i \in 1, 2, ..., N$.
Result: $\mathscr{K}$: selected participants.
1 **Set:** distance coefficient $\beta > 0$;
2 initialize probability $Proba^1$;
3 initialize current distribution $\mathscr{D}_c^1$;
4 $\mathscr{D}_g \leftarrow \sum_{i=1}^{N} \mathscr{D}^i$;
5 calculate $\widetilde{emd}_g$ by Eq. 2.7;
6 **for** *round* $r = 1, 2, ..., R$ **do**
7 $\quad \mathscr{K}^r = rand(K, \mathscr{C}, Proba^r)$
8 $\quad \mathscr{D}_c^{r+1} \leftarrow \mathscr{D}_c^r + \sum_{C_k\in\mathscr{K}}^{r} \mathscr{D}^{C_k}$;
9 $\quad$ calculate $\widetilde{emd}_c^r$ by Eq. 2.8;
10 $\quad$ **for** *client* $i = 1, ..., N$ **do**
11 $\quad\quad$ update $Proba^{r+1}$ by Eq. 2.6

Overview The complete algorithm is shown in Alg. 1, we here summarize the different components that make up the algorithm. ***i) Data Reporting and Initialization*** (Line 1–3): Clients report their data quantity so that the *federator* is able to compute the global data size array $\mathscr{D}_\}$ and initialize the current size array $\mathscr{D}_\rfloor^\infty$.

ii) Dynamic Weights Calculation (Line 4–11): In this key step, we utilize a light-weight measure based on EMD to calculate dynamic selection probabilities over time, which achieve faster convergence, yet avoid overfitting, concretely we compute

$$Proba^r = softmax(\widetilde{emd}_g - t\beta\widetilde{emd}_c^r) \tag{2.6}$$

where $Proba_i^r$ is the probability for selecting C_i in round r. β is a coefficient to weigh the global and current distance and shall be adapted for different initial distributions, i.e., different dataset and distribution rules. $\widetilde{emd}_g$ and $\widetilde{emd}_c^r$ are the normalized EMDs between the global/current and local distributions (Line 5, 9), namely

$$\widetilde{emd}_g = Norm([EMD(\mathscr{D}_\}, \mathscr{D}^i)\big|_{i\in\{1,...,N\}}]), \tag{2.7}$$

which is constant through the learning process as long as the local distribution of clients stays the same. The larger $\widetilde{emd}_g$ is, the higher the probability $Proba_i^r$ that a client C_i is selected to increase model accuracy (Line 11), since C_i brings more distribution information to train $\boldsymbol{\omega}_r$. However, for convergence, a smaller $\widetilde{emd}_c$ is preferred in selection, so that $\widetilde{emd}_c$ depends on the round r:

$$\widetilde{emd}_c^r = Norm([EMD(\mathscr{D}_c^r, \mathscr{D}^i)\big|_{i\in\{1,...,N\}}]), \tag{2.8}$$

where $\mathscr{D}_c^r$ is the accumulated $\mathscr{D}^\rangle$ of selected clients over rounds (Line 8). Let l denote one class randomly chosen by the *federator* except for the Maverick class from $\mathscr{D}$, here we apply normalization: $Norm(emd, \mathscr{D}) = \frac{emd}{\sum_{i=1}^N \mathscr{D}^i(y=l)/N}$.

iii) Weighted Random Client Selection (Line 7): At each round r, we select clients based on a probability distribution characterized by the dynamic weights [28] $Proba^r$:

$$\mathscr{K}^r = rand(K, \mathscr{C}, Proba^r). \tag{2.9}$$

Sampling K out of N clients based on $Proba^r$ has a complexity of $O(K\log(N/K))$, so comparably low. Thus, Mavericks with larger global distance and smaller current distance initially are preferred to be selected. The decrease of probability for selecting Mavericks elaborates based on the global and current distances changes over the learning procedure. As r increases, so does the impact of the current distance based on Eq. 2.6, reducing the probability of selecting a Maverick, as intended.

Convergence Analysis: To derive the convergence bound, we follow the setting of [81]. We let F_k be the local objective of client C_k and define $F(\boldsymbol{\omega}) \triangleq \sum_{k=1}^N p_k F_k(\boldsymbol{\omega})$, where p_k is the weight of client C_k when doing the aggregation. We have the FL optimization framework $\min_{\boldsymbol{\omega}} F(x) = \min_{\boldsymbol{\omega}} \sum_{k=1}^N p_k F_k(\boldsymbol{\omega})$. We make the *L-smooth* and *μ-strongly convex* assumptions on the functions $F_1, ..., F_N$ [81, 105]. Let T be the total number of SGDs in a client, E be the number of local iterations of each client in each round. t is used to index the SGDs in each client. Thus, the relationship between E, t and global round r is $r = \lfloor t/E \rfloor$. F^* and F_k^* are the minimum values of F and F_k. $\Gamma = F^* - \sum_{k=1}^N p_k F_k^*$ is used to represent the degree of heterogeneity. We obtain:

Theorem 2. Let ξ_t^k be a sample chosen from the local data of each client. For $k \in [N]$, assume that:

$$\mathbb{E}\left\| \nabla F_k(\boldsymbol{\omega}_t^k, \xi_t^k) - F_k(\boldsymbol{\omega}_t^k) \right\|_2^2 \leq \sigma_k^2, \tag{2.10}$$

and

$$\mathbb{E}\left\|F_k(\boldsymbol{\omega}_t^k, \xi_t^k)\right\|_2^2 \leq G^2. \tag{2.11}$$

Then let $\epsilon = \frac{L}{\mu}$, $\gamma = \max\{8\epsilon, E\}$ and the learning rate $\eta_t = \frac{2}{\mu(\gamma+t)}$. We have the following convergence guarantee for Algorithm 1.

$$\mathbb{E}[F(\boldsymbol{\omega}_T)] - F^* \leq \frac{\epsilon}{\gamma + T - 1}\left(\frac{2(\Psi+\Phi)}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\left\|\boldsymbol{\omega}_1 - \boldsymbol{\omega}^*\right\|_2^2\right), \tag{2.12}$$

where $\Psi = \sum_{k=1}^{N}(Proba_k^{\lfloor T/E \rfloor})^2 {\sigma_k}^2 + 6L\Gamma + 8(E-1)^2G^2$ and $\Phi = \frac{4}{K}E^2G^2$.

Since all the notations except T in Expression (2.12) are constants, we have $O(\frac{1}{T})$ convergence rate for the algorithm where $\lim_{T\to\infty}\mathbb{E}[F(\boldsymbol{\omega}_T)] - F^* = 0$.

2.5. EXPERIMENTAL EVALUATION

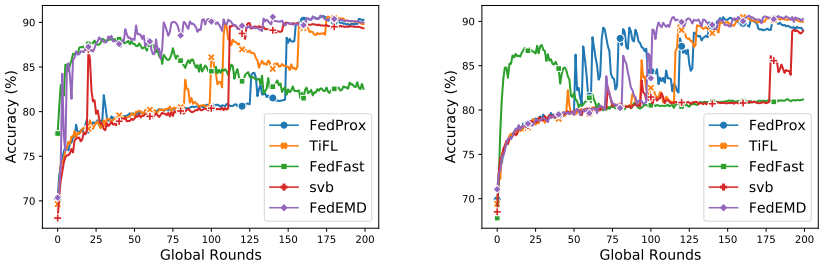
In this section, we comprehensively evaluate the effectiveness and convergence of FEDEMD in comparison to *Shapley Value*-based selection and SOTA baselines. The evaluation considers both exclusive and shared Mavericks.

Datasets and Classifier Networks We use public image datasets: *i)* Fashion-MNIST [151] for bi-level image classification; *ii)* MNIST [77] for simple and fast tasks that require a low amount of data; *iii)* Cifar-10 [72] for more complex task such as colored image classification; *iv)* STL-10 [18] for applications with small amounts of local data for all clients. We note that light-weight neural networks are more applicable for FL scenarios, where clients typically have limited computation and communication resources [99]. Thus, here we apply light-weight CNNs for all datasets.

Federated Learning System The system considered has 50 participants with homogeneous computation and communication resources and 1 *federator*. At each round, the *federator* selects 10% of clients using different client selection algorithms. The *federator* uses average or quantity-aware aggregation to aggregate local models from selected clients. We set one local epoch for both aggregations to enable a fair comparison of the two aggregation approaches. Two types of Mavericks are considered: exclusive and shared Mavericks with up to 3 Mavericks. We demonstrate the case of single Maverick owning an entire class of data in most of our experiments.

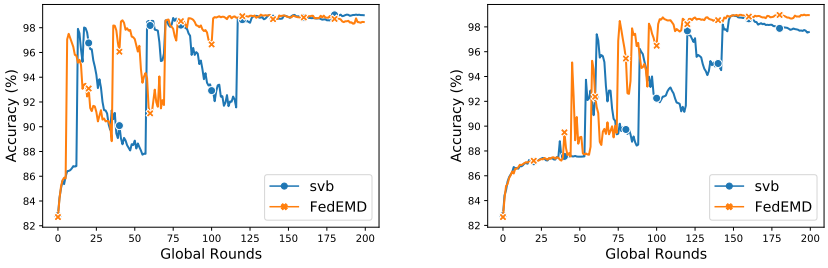
Evaluation Metrics *i)* Global test accuracy for all classes; *ii)* Source recall for classes owned by Mavericks exclusively; *iii) R*@99: the number of communication rounds required to reach 99% of test accuracy of random selection results; *iv)* Normalized *Shapley Value* ranging between $[0, 1]$ to measure the contribution of Mavericks.

Baselines We consider four selection strategies: Random [96], *Shapley Value*-based, FedFast [99], and TiFL [13][3] under both average and quantity-aware aggregation methods. Further, in order to compare with state-of-the-art solutions for heterogeneous FL that focus on the optimizer, we evaluate FedProx [80] as one of the baselines.



(a) FMNIST-quantity (b) FMNIST-average

Figure 2.3: Comparison on FEDEMD with baselines.



(a) FMNIST-quantity (b) FMNIST-average

Figure 2.4: Comparison on FEDEMD over different β.

2.5.1. FEDEMD IS EFFECTIVE FOR CLIENT SELECTION

Fig. (2.3a),(2.3b) show global accuracy over rounds. First, we focus on the comparison between the contribution-based SVB and our proposed distance-based FEDEMD. FEDEMD achieves an accuracy close to the maximum almost immediately for FedAvg while SVB requires about 100 rounds (72 and 104 rounds for R@99 for SVB and FEDEMD). For average aggregation, both client selection methods have a slower convergence but FEDEMD still only requires about half the number of rounds to achieve the same high accuracy as SVB. Indeed, SVB fails in reaching R@99 within 200 rounds. The reason is that SVB rarely selects the Maverick in the early phase, as the Maverick has a below-average *Shapley Value*. We can also see the superiority of FEDEMD among results presented for the baselines in the figures. The detailed analysis will be discussed together with Tab. 2.1 below.

We evaluate the effects of the hyper-parameter β in Fig. (2.4a),(2.4b). The server can apply a preliminary client selection simulation before training based on the self-reported data size array. FEDEMD works best when the average probability of selecting Maverick

[3] We focus on their client selection and leave out other features, e.g., communication acceleration in TiFL. We apply distribution mean clustering for FedFast following the setting in their paper.

Table 2.1: Convergence rounds of selection strategies in $R@99$ Accuracy, under average and quantity-aware aggregation (Every result is averaged over three runs and is marked with standard deviation among all of the replication results).

Dataset	Average Aggregation																	
	Random			FedProx			TiFL			FedFast			SVB			FEDEMD		
MNIST	133	±	44.47	118	±	8.50	111	±	21.66	>200	±	NA	147	±	52.50	**99**	±	24.70
Fashion-MNIST	144	±	51.47	135	±	20.59	140	±	8.62	>200	±	NA	**103**	±	56.00	131	±	37.29
Cifar-10	141	±	6.11	164	±	15.00	147	±	10.97	>200	±	NA	184	±	9.24	**140**	±	15.13
STL-10	122	±	49.94	186	±	4.36	125	±	57.50	171	±	16.74	190	±	3.06	**96**	±	4.93
Dataset	Quantity-aware Aggregation																	
	Random			**FedProx**			**TiFL**			**FedFast**			**SVB**			**FEDEMD**		
MNIST	72	±	29.26	51	±	8.19	84	±	37.99	>200	±	NA	49	±	2.52	**40**	±	5.57
Fashion-MNIST	111	±	37.75	92	±	12.12	146	±	38.18	>200	±	NA	80	±	40.13	**80**	±	10.79
Cifar-10	143	±	26.29	144	±	39.46	120	±	9.45	174	±	9.50	132	±	26.50	**107**	±	10.58
STL-10	180	±	0.58	179	±	6.24	>200	±	NA	153	±	34.88	181	±	10.97	**95**	±	2.65

is within $[1/N-\epsilon, 1/N+\epsilon]$ based on our observation experiments, where $\epsilon > 0$ is a task-aware small value. In our example with Fashion-MNIST, we choose β equal to 0.008, 0.009 and 0.01, with the results displayed in Fig. 2.4. These three values all satisfy the average probability above with $\epsilon \geq 0.002$. The results shows that all of the 3 numbers work for Fashion-MNIST, verifying the effectiveness of FEDEMD for various values of the hyper-parameter. However, there are also values of β that are not suitable, e.g, $\beta = 0.1$ for which the Maverick is selected too rarely.

Comparison with baselines. We summarize the comparison with the state-of-the-art methodologies in Tab. 2.1. The reported $R@99$ is averaged over three replications. Note that we run each simulation for 200 rounds, which is mostly enough to see the convergence statistics for these lightweight networks. The rare exceptions when 99% maximal accuracy is not achieved for random selection are indicated by > 200.

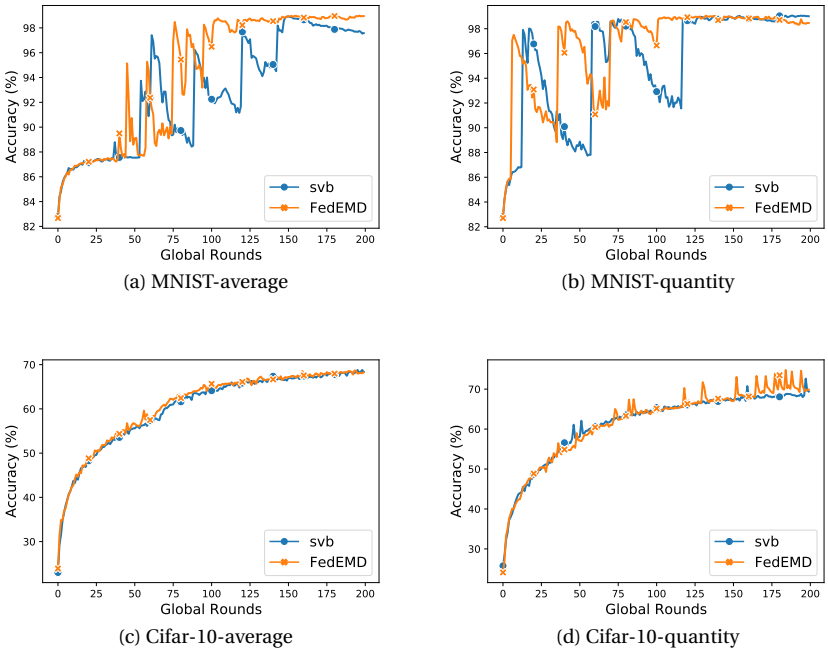
Due to its distance-based weights, FEDEMD almost consistently achieves faster convergence than all other algorithms. The reason for this result is that FEDEMD enhances the participation of the Maverick during the early training period, speeding up learning of the global distribution. For most settings, the difference in convergence rounds is considerable and clearly visible.

The only exceptions are easy tasks with simple averaging rather than weighted, e.g., Fashion-MNIST with average aggregation, which indicates our distribution-based selection method is especially useful for data size-aware aggregation and more complex tasks. Quantity-aware aggregation nearly always outperforms plain average aggregation as its weighted averaging assigns more impact to the Maverick. While such an increased weight caused by larger data size can lead to a decrease in accuracy in the latter phase of training, Mavericks are rarely selected in the latter phase by FEDEMD, which successfully mitigates the effect and achieves a faster convergence.

In order to demonstrate the comparison of FEDEMD and SVB across multiple datasets, here we also provide the experimental results with MNIST and Cifar-10, which is in line with our conclusion of Fashion-MNIST in Fig. 2.4 for better convergence performance of FEDEMD.

2.5.2. FEDEMD WORKS FOR MULTIPLE MAVERICKS

We explore the effectiveness of FEDEMD on both types of Mavericks: exclusive and shared Mavericks.



(a) MNIST-average (b) MNIST-quantity

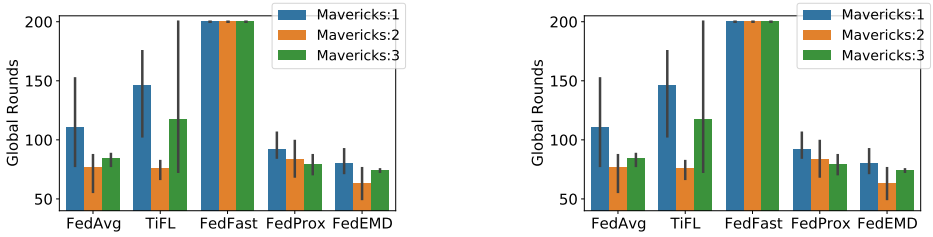
(c) Cifar-10-average (d) Cifar-10-quantity

Figure 2.5: Comparison on FEDEMD with SVB.

We vary the number of Mavericks between one and three and use the Fashion-MNIST dataset. The Maverick classes are 'T-shirt', 'Trouser', and 'Pullover'. Results are shown with respect to $R@99$.

Fig. (2.6a) illustrates the case of multiple exclusive Mavericks. For exclusive Mavericks, the data distribution becomes more skewed as more classes are exclusively owned by Mavericks. FEDEMD always achieves the fastest convergence, though its convergence rounds increase slightly as the number of Mavericks increases, reflecting the increased difficulty of learning in the presence of skewed data distribution. FedFast's K-mean clustering typically results in a cluster of Mavericks and then always includes at least one Maverick. In some initial experiments, we found that constantly including a Maverick hinders convergence, which is also reflected in FedFast's results. TiFL outperforms FedAvg with random selection for multiple Mavericks. However, TiFL's results differ drastically overruns due to the random factor in its local computations. Thus, TiFL is not a reliable choice for Mavericks. Comparably, FedProx tends to achieve the best performance among the SOTA algorithms but still exhibits slower convergence than FEDEMD as higher weight divergence entails a higher penalty on the loss function.

For shared Mavericks, a higher number of Mavericks indicates a more balanced distribution. Similar to the exclusive case, FEDEMD has the fastest convergence and FedFast again trails the others. The improvement of FEDEMD over the other methods is less visible due to the limited advantage of FEDEMD on balanced data. A higher number of



(a) Exclusive Mavericks

(b) Shared Mavericks

Figure 2.6: Convergence rounds $R@99$ for multiple Mavericks.

Mavericks resembles the case of *i.i.d.*. Random performs the most similar to FEDEMD for shared Mavericks, as random selection is best for *i.i.d.* scenarios. Note that the standard deviation of FEDEMD is smaller, implying a better stability.

2.6. CONCLUSION

Client selection is key to successful FL as it enables maximizing the usefulness of different diverse datasets. In this chapter, we highlighted that existing schemes fail when clients have heterogeneous data, in particular if one class is exclusively owned by one or multiple Mavericks. We first explore *Shapley Value*-based selection, theoretically showing its limitations in addressing Mavericks. We then propose FEDEMD that encourages the selection of diverse clients at the opportune moment of the training process, with guaranteed convergence. Evaluation results on multiple datasets across different scenarios of Mavericks show that FEDEMD reduces the communication rounds needed for convergence by 26.9% compared to the state-of-the-art client selection methods.

3

DATA-FREE UNTARGETED ATTACKS IN FEDERATED LEARNING

Attacks on Federated Learning can severely reduce the quality of the generated models and limit the usefulness of this emerging learning paradigm that enables on-premise decentralized learning. However, existing untargeted attacks are not practical for many scenarios as they assume that i) the attacker knows every update of benign clients, or ii) the attacker has a large dataset to locally train updates imitating benign parties.

*In this paper, we propose a data-free untargeted attack (*DFA*) that synthesizes malicious data to craft adversarial models without eavesdropping on the transmission of benign clients at all or requiring a large quantity of task-specific training data. We design two variants of* DFA*, namely* DFA-R *and* DFA-G*, which differ in how they trade off stealthiness and effectiveness. Specifically,* DFA-R *iteratively optimizes a malicious data layer to minimize the prediction confidence of all outputs of the global model, whereas* DFA-G *interactively trains a malicious data generator network by steering the output of the global model toward a particular class. Experimental results on Fashion-MNIST, Cifar-10, and SVHN show that* DFA*, despite requiring fewer assumptions than existing attacks, achieves similar or even higher attack success rate than state-of-the-art untargeted attacks against various state-of-the-art defense mechanisms. Concretely, they can evade all considered defense mechanisms in at least 50% of the cases for CIFAR-10 and often reduce the accuracy by more than a factor of 2.*

Consequently, we design REFD*, a defense specifically crafted to protect against data-free attacks.* REFD *leverages a reference dataset to detect updates that are biased or have a low confidence. It greatly improves upon existing defenses by filtering out malicious updates and achieves high global model accuracy.*

3.1. INTRODUCTION

Federated learning [154, 106] enables distributed training of machine learning models, e.g., multi-class image classifiers, without sharing the raw data. *Clients* train models locally and the overall model, called the global model, is an aggregation of these local models. The training proceeds in multiple rounds: in each round, *the central server* provides a global model that clients use to initialize their local models. They then train on their local dataset and provide updates to the central server, who aggregates these updates to a new global model for the next round. In this manner, models requiring personal data such as information about medical or financial conditions can be obtained without explicit privacy violations. Recently, FL has been applied to domains such as detection of credit card fraud [176, 175], cybersecurity center operations [70], and medical relation extraction [128].

A downside of preventing the central server from accessing local data is that it limits the ability to detect misbehavior. Adversarial clients may reduce the quality of the model by manipulating the data they train on [135] or their local model directly [32]. Cross-device [69, 44] FL, which allows arbitrary parties to join the distributed training, is especially vulnerable as attackers can easily infiltrate the system. The attack can be untargeted, i.e., aiming for an overall accuracy degradation of the trained global model. It can also be targeted, i.e., only supposed to affect certain input, e.g., inject backdoors that lead to wrong model output from input data with a certain chosen feature [5]. In this chapter, we focus on untargeted attacks, as they are far-reaching denial-of-service attacks. In cross-device FL, attackers may run such a denial-of-service attack to undermine a competing company from getting meaningful models after their users. Furthermore, when machine learning as a service [109] is extended to include FL [71], untargeted attacks aiming to cause losses for a service provider are to be expected, similar to current denial-of-service attacks on Amazon Web Services and Github [1].

There have been a number of untargeted attacks on FL [32, 7, 118]. Yet, some attacks [7, 118] assume that the adversary is aware of all of the updates that benign clients send. It is unclear how they can practically obtain such knowledge as clients only share the updates with the benign central server and communication can be encrypted to prevent eavesdropping from the adversary. While not all attacks require benign updates, attacks that can succeed without this knowledge requires that the attacker has a considerable amount of training data to train substitute benign updates [32]. Although this assumption is realistic for common tasks, e.g., image classification of common pets, the possession of such data is much less likely for special-purpose tasks, e.g., classification of rare disease based on detailed medical data [111].

In this chapter, we consider whether it is actually necessary to have real data (or benign updates). One may expect that in cross-device FL, it is relatively easy to obtain data as everyone can join, which might indicate that everyone can have data. However, there exist scenarios where admission is not restricted to a predefined group because there are few parties that can contribute and it is not known who they are. For instance, for a study on the lives of people with a rare disease, it might not be possible to access medical records on who has the disease, so it makes sense to just publicly ask for participation. Furthermore, not requiring parties to identify before joining allows them to participate

[1] https://www.a10networks.com/blog/aws-hit-by-largest-reported-ddos-attack-of-2-3-tbps/

anonymously, possibly using tools like Tor [24] to send in their updates without having to fear that they reveal that they have a certain medical condition, which could increase their insurance premium or prevent them from gaining employment. In such a scenario, it is also hard to corrupt participating clients and use their data, as the identity of the clients is not known. Even if the learning task is such that is easy to obtain data, e.g., a software company aiming to build a model on how users interact with their tool, it is still additional overhead for the attackers, e.g., they have to either use the tool themselves or obtain data from a real user. Thus, even if the attacker can get data, the question of whether they *have to* or can skip the overhead of data acquisition is essential as without data acquisition, it is more likely that attacks can be automated and run at scale against many FL learning tasks.

We design a novel Data-Free Attack (DFA) and evaluate it on the example of image classification. The goal of the attack is to reduce the overall accuracy of the model through the injection of malicious model updates based on synthetic images. In each round, the attacker first generates malicious images by making use of the received global model and then trains the local adversarial model using those images paired with a randomly chosen class $\tilde{Y}$. We design two variants of DFA, DFA-R and DFA-G, which steer the global model to classify images to either have low confidence or to classify incorrectly. Our first attack variant, DFA-R, generates synthetic local data by adding a filte**R** layer to the training. This data generation optimizes towards local synthetic data that is ambiguous according to the current global model, i.e., the current global model should output each of the L possible classes with equal probability. A local model corresponding to such data diverts the global model and reduces classification accuracy. In contrast, our second attack, DFA-G, iteratively trains a **G**enerator that should produce synthetic images that are not from a specific randomly chosen class $\tilde{Y}$. We then assign these images with class label $\tilde{Y}$ and train on the resulting dataset, thus implicitly combining synthetic data generation with label flipping for poisoning.

To improve stealthiness for both attacks, we add a regularization term to the loss function of the classifier that steers the update generation such that updates are not detected as outliers and hence not removed by defenses. DFA thus stealthily bypasses the defense by ensuring that the deviation to the global model follows similar patterns as benign updates.

In our evaluation, we determine the attack success rate, i.e., the decrease in model accuracy caused by the attack, and the rate at which our attackers pass the defense. We evaluate different levels of data heterogeneity by assigning data to clients according to the Dirichlet distribution, which is a common model for heterogeneous real-world distributions [148]. DFA-R and DFA-G reduce the accuracy of the trained model by a factor of 2 for most settings, even if defenses are applied. In comparison to state-of-the-art attacks, DFA-R and DFA-G achieve similar results, despite having weaker assumptions. Indeed, for most scenarios, our attacks perform slightly better than the existing attacks.

Having shown that data-free attacks have severe impact on the accuracy of FL, we propose a defense strategy, REFD, which aims to defend against DFA-G and DFA-R by leveraging a reference dataset at the server. Based on this reference dataset, the central server determines whether a received model update is biased toward a certain class, which is typical for DFA-G, or shows a low confidence, which is typical for DFA-R. It

Attacks	LIE[7]	Fang[32]	Min-Max[118]	Min-Sum[118]	FEDEMD (ours)
No benign updates needed	✗	✓/✗	✓/✗	✓/✗	✓
Defense-agnostic	✓	✗	✓	✓	✓
No raw data needed	✓	✓/✗	✓/✗	✓/✗	✓
Heterogeneity considered	✗	✓	✓	✓	✓
Attack categories	Statistic	Statistic	Statistic	Statistic	Optiminzation

Table 3.1: Applicability of targeting scenarios and categories of five attacks (four state-of-the-art baselines and ours): ✓refers to applicable, ✗refers to not applicable, and ✓/✗means the successful attack requires either information available.

combines these two factors into a novel defense score, termed $\mathscr{D}$-score. Our evaluation results show that REFD successfully defends against the proposed data-free attacks, achieving accuracies that are close to the accuracy achieved in the absence of both attacks and defenses.

3.2. BACKGROUND AND RELATED WORK

3.2.1. FEDERATED LEARNING PRIMER

As a distributed machine learning framework, FL systems consist of a set of N clients and a central server. The global training process considers R consecutive rounds indexed by the round number t. After model initialization by the server, each client i (for $i = 1, 2, ..., N$) trains a local model based on their own real data without sharing the raw data. The server iteratively aggregates models/gradients submitted from clients and distributes the aggregated model to the clients until reaching global model convergence. As clients can be offline or unresponsive, only a subset of them usually submits updates.

In this chapter, we focus on image classification tasks with L classes. Let D_i be the local dataset of client i and F be the objective function for the classification task. The client i updates its local model weights based on the global model $\boldsymbol{w}(t)$ by:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}(t) - \eta \frac{\partial F(\boldsymbol{w}(t), D_i)}{\partial \boldsymbol{w}(t)}, \tag{3.1}$$

where η is the global uniformed learning rate.

For aggregating models of $K \leq N$ clients, the predominant method for attack-free scenarios is FedAvg [96], which aggregates the new global model as a weighted average of the submitted local models , i.e.,

$$\boldsymbol{w}(t) = \sum_{i=1}^{K} \frac{n_i}{\sum_{k=1}^{K} n_k} \boldsymbol{w}_i(t), \tag{3.2}$$

where n_i is the number of training samples of client i. However, the above algorithm is not robust under attacks [**backdoor:conf/aistats/BagdasaryanVHES20**, 32, 118, 152], hence defenses for securing the aggregation against maliciously crafted updates (also called robust aggregation methods) have been developed.

3.2.2. EXISTING ATTACKS IN FL SYSTEMS

FL empowers clients by leaving the training to them and not revealing the local data. However, as a consequence, FL systems are vulnerable to malicious behaviors. Attacks can happen during the **training time** [7, 118, 5, 32, 152] or **inference time** [158, 125, 102]. For the inference-time attacks, attackers aim to infer private data [102]. They may even reconstruct the private local training data [158]. In this chapter, we focus on training-time attacks where attackers participate in the training. We classify the training-time attacks from two perspectives: *i*) the attack objectives and *ii)* the attacked component of the FL system, e.g., data or model.

There are three attack objectives for training-time attacks: **Free-riding** [85, 35] is used to obtain the global model without contributing data and computation. **Targeted attacks** [5, 152] aim to decrease the model accuracy for specific data, e.g., data with designed triggers. **Untargeted attacks** [7, 32, 118], in contrast, aim to decrease the general accuracy of the model.

There are four state-of-the-art untargeted attacks, namely LIE [7], Fang [32] as well as Min-Max and Min-Sum [118], which are two variants of the same attack idea. We summarize their key differences in Table 3.1. All attacks require knowledge of the models of benign clients, real data, or knowledge of any defenses applied by the server. Some of them, like Min-Max, are flexible in that they can work with either benign updates or real data but they need at least one of the two, which we indicate by ✓/✗ in the respective rows in the table. In terms of methods, all existing attacks rely on statistical methods or heuristics to construct the malicious updates by shifting the mean of benign updates without being detected. Concretely, LIE [7] calculates the mean and standard deviation of all of the benign updates and then shifts the true mean by changing the value in one direction in such a manner that it is within the range that is considered acceptable by the defense. Shejwalkar et. al [118] further improve LIE by adapting the scaling factor z of the weighted sum as well as extending the standard deviation to the sign and unit vector of the gradient. By such means, the maximum distance (or the sum of squared distances for Min-Sum) of the malicious gradient from all the benign gradients is upper bounded. Note that while the authors [118] propose a number of attacks according to different levels of adversarial knowledge, we only compare to the Min-Max attack, which is the strongest in their paper. Fang et. al [32] propose an attack that steers global model parameters in the opposite direction of the benign updates and ensures its stealthiness through the knowledge of the exact defense. Aforementioned untargeted attacks, except LIE, are evaluated in junction with the heterogeneous data, which is attribute skewed [20, 127] or label skewed [139, 161, 87].

Moreover, attacks can also be categorized by the component which attacks act upon: data or model. During the training time, the adversary may inject malicious data with dirty labels or data to train the local model, e.g., label flipping [135] and trigger injection [5, 152]. For example, backdooring[5] is executed by injecting trigger-based malicious samples [5, 162] into the local training dataset. DBA [152] then extends the study [5] to bypass Sybil defenses such as FoolsGold [38]. Modeling poisoning [7, 118, 32, 5, 152] manipulates the submitted model rather than merely adopting malicious data to train, e.g., submit updates of the reversed sign of training gradient [32]. Generally, model poisoning attacks require sophisticated technical capabilities such as eavesdropping and

sufficient computation resources.

None of the existing attacks can deal with an attacker that does not have data unless they can observe the communication in plaintext. Our attack uses a generator, as do other attacks but for highly different scenarios or goals: attacking centralized learning [169, 164], attacking privacy [170, 125], or mimicking prototypical samples of the other participants' training set with the goal of targeted attacks [167, 168].

3.2.3. EXISTING DEFENSE MECHANISM IN FL

We here focus on defense mechanisms for FL, as algorithms designed for centralized learning (e.g., [14, 79] are not directly applicable in FL. To tackle the attacks on FL systems, existing defense strategies can be conducted either on the server-side [9, 98, 157], or the client-side [129, 173, 100]. Server-side defenses are effective against both targeted and untargeted attacks due to the access to all model updates, whereas the state-of-the-art client-side defenses are merely shown to be effective against targeted attacks. As we are concerned with untargeted attacks, we hence focus on server-side defenses.

Generally, there are three categories of server-side defenses: *i) Sybil defenses* aim at detecting Sybil attackers who are controlled by one entity and submit similar updates. For example, *FoolsGold* [38] identifies Sybils based on the diversity of client contributions using cosine similarity of client updates. *ii) Statistic defenses* curate the aggregated model by computing the statistics of every parameter across multiple updates. *Median* [157] utilizes the median value of all updates for each parameter whereas Trimmed mean (*TRmean*) [157] excludes the minimum and maximum value from the average of each parameter. *iii) Outlier detection* [9, 98] removes updates based on the pairwise distances of returned models. Higher distance implies that data owned by a client is of low quality or unrelated to the training task. *Krum* [9] only uses one update sent from the client whose cumulative distance of updates to the other updates is the lowest, taking the squared L2 Norm as a metric. *mKrum* [9] extends this idea by choosing multiple updates. *Bulyan* [98] first selects updates using *mKrum* and further computes the trimmed mean of the selected gradients.

3.3. AN OPTIMIZATION-BASED DATA-FREE ATTACK

In this section, we first introduce the threat model for our work. Then we propose our data-free attack (DFA) with two variants to generate malicious data inputs and local model updates: DFA-R and DFA-G.

3.3.1. THREAT MODEL

We assume that communication between clients and the central server uses encrypted and authenticated channels, which prevent eavesdropping and manipulation of data during transmission. As a consequence, attackers are unaware of benign client updates. Benign clients always follow the protocol whereas malicious clients may arbitrarily deviate. All attackers may submit the same update. We add these assumptions for simplicity as we can easily circumvent Sybil defenses by adding small perturbation noise, as shown in the related work [5]. The central server applies a defense mechanism, which is not known to the clients.

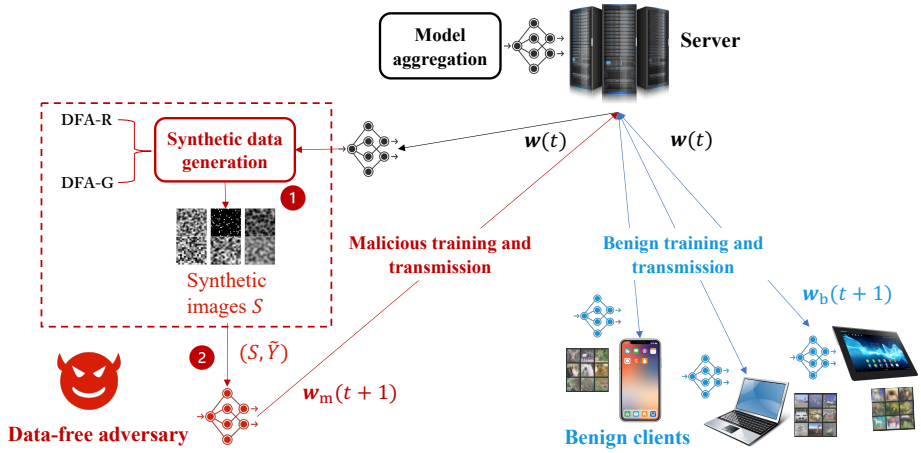


Figure 3.1: The framework of our proposed data-free attack (DFA) without knowing benign updates and owning raw data.

We focus on cross-device FL, which means that anyone can join and at the same time there is client selection each round. Furthermore, the adversary inserts their own clients in the system rather than corrupt other clients. Corrupting other clients requires knowing the identities of other clients, which is not explicitly shared in cross-device FL. In the absence of anonymous communication, the adversary could obtain the identities only from observing network traffic but the ability to observe network traffic is restricted to internet service providers and other parties, so we do think it is more realistic to assume that the adversary does not know the other clients and hence also cannot easily corrupt them.

Additionally, we assume that malicious parties do not have any data so as to enhance the versatility of the adversary. In practice, the difficulty of obtaining data varies between tasks. It is reasonable to assume that there are tasks relying on rare data that an attacker cannot easily obtain. We assume that all computations are executed by one adversarial party, who then sends the updates to individual malicious clients.

Objectives: The overall objective of an untargeted attack in Federated Learning is reducing the accuracy of the global model maintained by the central server. As a part of achieving this objective, clients need to craft malicious updates that bypass the applied defense.

Capabilities: First, we assume that the number of malicious users controlled by the adversary in the system does not exceed 50% of the total clients. It seems implausible that a defense can overcome a higher number of attackers as defenses typically need a reference for benign behavior. The attacker cannot break cryptographic primitives. More generally, it is computationally bounded so that it cannot solve NP-complete or NP-hard problems. Otherwise, they can arbitrarily control the communication and computation of the malicious clients but not of any other parties in the system.

Knowledge: Neither the defense algorithm nor benign updates are known to the adversary. As the attacker also does not have data, the only knowledge of the adversary is

the classification task in general, i.e., the number of classes, which is necessarily accessible as the server distributes the model.

3.3.2. ATTACK OPTIMIZATION FRAMEWORK

The overall framework of our proposed attack DFA is illustrated in Fig. 3.1. The server first distributes the current global model (classifier) $\boldsymbol{w}(t)$ to all of the clients. The benign clients truthfully follow the protocol and send the trained model $\boldsymbol{w}_b(t+1)$ back to the server. Malicious clients send the adversarial model $\boldsymbol{w}_m(t+1)$ instead. Then the server aggregates the submitted updates according to the deployed defense. As attackers do not have real data or benign updates, intuitively, the most obvious approach to attack is to directly change $\boldsymbol{w}(t)$.

We experimented with using random weights but the attack was detected almost always. Concretely, only 2.62% and 6.57% of all updates submitted by malicious clients with random model weights bypassed the *mKrum* defense for Fashion-MNIST and Cifar-10, respectively. For the *Bulyan* defense, the attack only bypassed the defense in 3.27% of the cases for Fashion-MNIST and always failed for Cifar-10. As manipulating the model directly does not seem a promising approach, we optimize the generation of synthetic malicious images according to $\boldsymbol{w}(t)$ and then use it to train the local adversarial model every round. The attack process consists of the following two steps.

1. Malicious image generation. We propose two optimization methods to synthesize malicious images based on different optimization methods, objectives of adversarial models and, importantly, the feedback of the global model. The first method is **DFA-R**, which introduces an additional **filter input layer**[2] and optimizes it from a dummy image with the objective to reduce the confidence on all outputs of the global model. Our second method, **DFA-G**, designs a **generator network** to synthesize malicious images such that the output of the global model biases toward a randomly chosen class. As such, the generated noisy images paired with incorrect labels are applied to malevolently update the current model. The details on DFA-R and DFA-G are discussed in the following subsections.

2. Adversarial classifier training with distance-based loss. In this step, the attacker uses synthetic data as generated by step 1 to train the classifier $\boldsymbol{w}_m(t+1)$. The optimization problem of the attack then becomes $\min_{\boldsymbol{w}_i} F(\boldsymbol{w}_i, S)$, where S is the generated image set. In order to enhance the stealthiness and hence pass the unknown defense, we propose to train with a distance-based loss function $\min_{\boldsymbol{w}_i}(F(\boldsymbol{w}_i, S) + \mathscr{L}_d)$ with regularization term $\mathscr{L}_d$ to enhance stealthiness (detailed illustration in Sec. 3.3.5). The size of S, $|S|$, is a hyper parameter of our attack framework that depends on the task. In the evaluation, we find that using a similar number of images as benign clients results in an effective attack. The adversary can estimate the size during training based on the aggregated results of the global model and the duration that other clients require for training.

3.3.3. DFA-R SYNTHETIC DATA GENERATION

When constructing the synthetic dataset S, DFA-R aims to aggressively lower the confidence of all outputs of the global model by introducing a malicious filter layer(i.e., a

[2] Such a layer has the same input dimension as the original image.

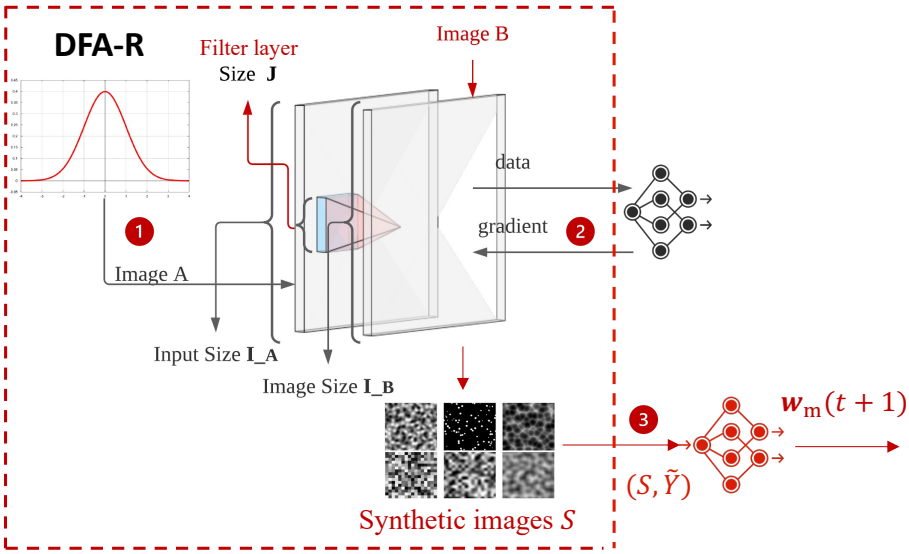


Figure 3.2: Synthetic data generation process of DFA-R.

convolutional layer). DFA-R optimizes this image layer such that per-class probability output of global model is equally low, i.e., $Y_D = [\frac{1}{L}, \frac{1}{L}, ..., \frac{1}{L}]$, where L is the total number of classes. Such data is bound to confuse the global model. Fig. 3.2 depicts the optimization procedure to find $|S|$ malicious images iteratively via two steps: *i)* generating the malicious image through mapping a random dummy image via a filter layer [76], and *ii)* optimizing the filter layer by minimizing cross-entropy loss of the global model between the predicted class probabilities and Y_D.

Concretely, we first generate a random image A (size $a \times a$), with each pixel being drawn from a uniform distribution, and apply the filter layer to transform it into an image B(size $b \times b$). In this manner, we train a mapping from randomness to images that have the desired properties. The size of image B is the same as the real image. We let this convolutional layer have kernel size $J \times J$, i.e., the square filter layer between image A and B in Fig. 3.2. After being filtered from the convolution layer, the image B is then classified by the current global model. The attack works for various network structures and datasets, e.g., Alexnet, VGG on other image datasets, as long as the relation between input and output are maintained. Concretely, for stride size St and padding size P [82], we require $a = b \times (St + 1) - 2P + J$. The attack can be extended to other tasks, e.g., text processing, by replacing the filter model and using a Seq2Seq model [133] instead of a convolutional layer. In this manner, the random text (mapped from random values to a dictionary) is filtered by the Seq2Seq model and fed in the text processing network, similar to Fig. 3.2.

To optimize the convolutional layer that results in ambiguous Y_D, we first consider the dummy image A, the filter layer, synthetic image B, and the global model as one big classification problem. Its training objective is to minimize the cross-entropy loss of pre-

dicted probabilities of image B and Y_D, such that the model cannot predict classes reliably. Different from the regular training of a classification problem, we keep certain parts of the model and input constant. Specifically, the model weights of the global model and the image A are static. Otherwise, without keeping A static, we would need to re-train whenever we change the randomness. Keeping the number of trainable parameters to a minimum, we optimize the efficiency of the attack. The only trainable parameters here are the parameters of the filter layer. It takes E epochs to train this convolutional layer. Upon finishing training, the image B is one data instance of S. To increase the diversity of the training dataset S, for each FL training round, we repeat the above process for $|S|$ times to construct S.

3.3.4. DFA-G SYNTHETIC DATA GENERATION

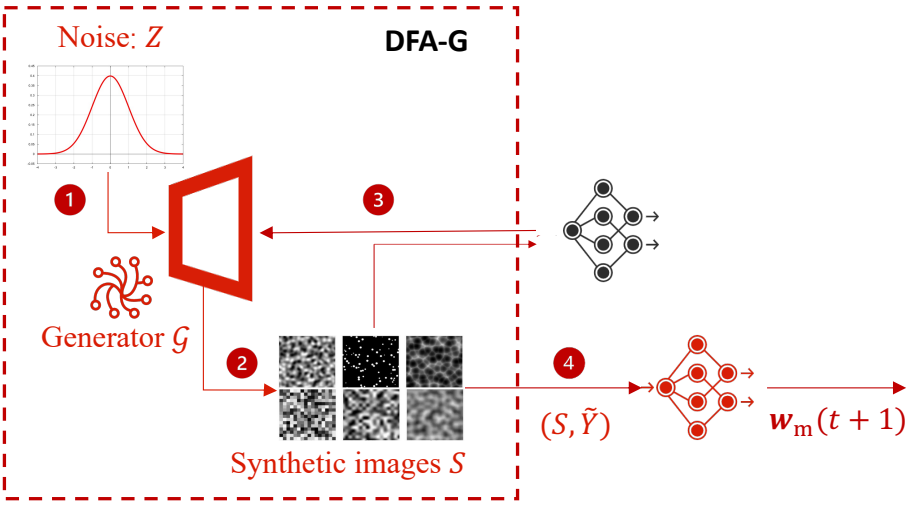


Figure 3.3: Synthetic data generation process of DFA-G.

In contrast to DFA-R, DFA-G synthesizes images through a generator network, which misleads the global classifier to confidently make incorrect classifications. In order to do so, we generate images that are not supposed to be from a class $\tilde{Y}$ but classify all of them as $\tilde{Y}$, which is a randomly chosen label and never changes through the training procedure. The training/optimization of the generator network is through the feedback of the global model, i.e., we assume that the classification provided by the global model is the correct classification of the synthetic image. Typically, benign training minimizes the cross-entropy of the prediction and true label so that the model can output an accurate prediction. However, as our goal is to reduce the model accuracy, we maximize the cross-entropy of the prediction and $\tilde{Y}$ to train $\mathcal{G}$, steering the generated images away from $\tilde{Y}$. As DFA-R, DFA-G works for various network structures, with input and output size needing to match the training data.

The training procedure for the generator is shown in Fig. 3.3. We first draw a random noise vector Z from the Gaussian distribution and input Z into a generator $\mathcal{G}$ to

synthesize malicious images. We use the same random seed over multiple rounds so that the trained generator is able to consistently produce synthetic data different from class $\tilde{Y}$, as our training goal is to optimize the mapping from the generated vector to the targeted synthetic data. The network structure of the generator is a transpose convolution neural network (TCNN), which outputs task-specific image size data $S = \mathcal{G}(Z)$. The size of real data can be obtained from $\boldsymbol{w}(t)$. Specifically, we use a lightweight TCNN of two transposed convolutional layers and one convolutional layer following the structure of the popular WGAN paper [4]. The model parameter of the generator $\mathcal{G}$ is randomly initialized before training, denoted as $\boldsymbol{\theta}$. As the generator aims at synthesizing images that differ from the chosen class $\tilde{Y}$, the objective function of $\mathcal{G}$ is $\max_{\boldsymbol{\theta}} F(\boldsymbol{w}(t), (S, \tilde{Y}))$ where S is generated from $\boldsymbol{\theta}$. After training $\mathcal{G}$ locally for E epochs until convergence, the synthetic images are leveraged to train $\boldsymbol{w}_m(t+1)$. When it comes to other tasks, e.g., text processing, the generator is a recurrent neural network such as GRU [17] in order to generate random texts rather than just random numbers.

In summary, differences between the two variants are *i)* the optimization network, *ii)* the objective functions, *iii)* the use of $\tilde{Y}$, and *iv)* the randomness in their inputs.

3.3.5. DISTANCE-BASED REGULARIZATION

State-of-the-art defenses in a FL system are mainly based on the pairwise distances among multiple updates. In order to bypass the defense mechanisms, we introduce a distance-based regularization term when training the adversarial classifier with the aim to further enhance stealthiness of the adversarial model updates. The concrete regularization term is

$$\mathcal{L}_d = \|\boldsymbol{w} - \boldsymbol{w}(t)\|_2 - \|\boldsymbol{w}(t) - \boldsymbol{w}(t-1)\|_2. \tag{3.3}$$

In Eq. 3.3, the first term refers to the weight differences of the adversarial update and the current model. Analogously, the second term refers to the difference between the current model and the model of the previous round. This term varies over rounds and the optimization variable is the model parameter $\boldsymbol{w}$. We add $\mathcal{L}_d$ to the vanilla cross-entropy loss in the objective function of the adversarial classifier to avoid extremely high differences in model changes over rounds, which could be easily detected. Thus, in both DFA-R and DFA-G, we guide the training such that the differences in weights are similar to the ones in previous rounds, as achieved by using the two most recent global models.

3.4. EXPERIMENTAL EVALUATION

We empirically evaluate the effectiveness of our proposed data-free untargeted attack, on three commonly used classification benchmarks. We compare the attack success rate and defense pass rate for various settings, for four state-of-the-art defenses and in comparison to three existing attacks. All of the results reported in this section are averaged over three runs. The source code of DFA is provided in github.

3.4.1. EXPERIMENT SETUP

The experimental setups are organized into the following categories. These components jointly define a realistic and reproducible federated learning environment for evaluation. Each category is specified in detail.

FL system. Our FL system considered contains 100 clients. In typical real-world FL systems, some of the clients could be offline or unavailable temporarily and hence not all of them might be able to participate in the whole training process. Thus, as in previous work [5, 96, 118], 10 of the available clients are selected uniformly at random each round. Clients train the classifier locally for one epoch. For the main results, we assume that the adversary can compromise 20% of the clients following [118, 32], unless stated otherwise, and further evaluate 10% and 30% in Sec. 3.4.5. Lower percentages of attackers have been shown to be ineffective [119].

Datasets and networks. In this work, we consider three datasets. *Fashion-MNIST* [151] consists of a training set of 60,000 and a test set of 10,000 fashion-related images. Each instance is a 28 × 28 grayscale image. *Cifar-10* [73] contains 50,000 training images of 3-channel RGB images and 10,000 of test images. *SVHN* [104] includes 73257 digit images for training and 26032 for testing. All digits have 32 × 32 pixels. All datasets have 10 classes in total. For Fashion-MNIST and Cifar-10, the images are evenly distributed over classes. SVHN in slightly imbalanced in class distribution. The total number of images used to train in this chapter is reduced to 10% for Fashion-MNIST and Cifar-10 but maintain the original size for SVHN. For Fashion-MNIST and Cifar-10, the data are chosen uniformly at random in order to model real-world scenarios that full data may not be available during the whole training. This amount is verified to be sufficient for training on Cifar-10 and Fashion-MNIST [6].To determine the $|S|$ and show hyperparameter sensitivity, we run initial experiments varying $|S|$ from 20, 50, and 100 based on knowing 50 samples per client for Cifar-10. We found that DFA is able to achieve similar attack success rate. Indeed, sometimes a lower $|S|$ had a higher ASR, e.g., for DFA-G on Fashion-MNIST with $\beta = 0.5$, $|S| = 20$ has higher attack success rate than $|S| = 50$. As they all succeed in attacking, we use the results of 50 in the chapter to keep consistency. For these three datasets, we use representative neural networks with 2 (for Fashion-MNIST) and 6 (Cifar-10 and SVHN) convolutional layers connected with 1 and 2 densely-connected layers, respectively, to map the inputs and outputs[3].

Defense mechanisms. Four state-of-the-art defenses are evaluated in our work: *mKrum*, *TRmean*, *Bulyan* and *Median*. We do not apply *Krum* since *mKrum* interpolates between *Krum* and averaging, thereby allowing the trade-off between the resilience properties and the convergence speed [9].

Data heterogeneity. To emulate a heterogeneous distribution, we assign data to clients according to the commonly used Dirichlet distribution. It emulates a real-world data distribution and the degree of heterogeneity is governed by the hyperparameter β [148], indicating the level of heterogeneity. In Sec. 3.4.4, we vary β from 0.1 to 0.9 in order to demonstrate our effectiveness for different degrees of data heterogeneity. Higher β means a lower degree of data heterogeneity. For our experiments, except for Sec. 3.4.4, we choose $\beta = 0.5$, as in the prior work [148, 51].

Hardware. Our FL emulator is based on Pytorch and we run experiments on a machine running Ubuntu 20.04, with 32 GB memory, a GeForce RTX 2080 Ti GPU and an Intel i9 CPUs with 10 cores (2 threads each).

[3] We use shallow networks to simplify evaluation, consistent with [118], higher accuracy can be achieved with deep nets.

3.4.2. EVALUATION METRICS.

We utilize two main metrics to evaluate the effectiveness of our attack. *i) Attack success rate (ASR)* is defined by:

$$ASR = \frac{acc - acc_m}{acc} \times 100\%, \tag{3.4}$$

i.e., the decrease of accuracy caused by attacks. Specifically, it is the difference between the global accuracy acc without attacks and defenses and the maximum accuracy acc_m of the global model during one experiment with attacks. Attack success rate specifies the effectiveness of an attack strategy through the decrease in accuracy. The Higher, the better.

ii) Defense pass rate (DPR) is a metric to measure the stealthiness of an attack. In this chapter, it is defined by the proportion of attackers who have passed the defense (N_p) from all of the randomly selected attackers (N_s):

$$DPR = \frac{N_p}{N_s} \times 100\%. \tag{3.5}$$

DPR as defined above requires that defenses select updates for aggregation rather than computes statistics on all updates. Thus, as detailed Sec. 3.2.3, DPR can only be computed for *mKrum* and *Bulyan*, but not for *TRmean* and *Median*. High *DPR* is better.

Table 3.2: Attack success rate (*ASR*) and the maximum accuracy (acc_m) accordingly under attacks on Dirichlet distribution. $\beta = 0.5$. (The accuracy without attacks and defenses acc for Fashion-MNIST, Cifar-10 and SVHN is 82, 50, and 86, respectively [7]).

		Fang		**LIE**		**Min-Max**		**DFA-R**		**DFA-G**	
Dataset	**Defense**	acc (%)	*ASR* (%)	acc (%)	*ASR* (%)	acc (%)	*ASR* (%)	acc (%)	*ASR* (%)	acc (%)	*ASR* (%)
Fashion-MNIST	*mKrum*	73.5	**10.37**	72.7	**11.34**	67.3	**17.93**	52.6	**35.85**	64.3	**21.59**
	Bulyan	68.1	**16.91**	75.0	**8.54**	56.8	**30.73**	70.8	**13.66**	59.8	**27.07**
	TRmean	30.9	**62.32**	59.9	**26.95**	37.8	**53.90**	21.9	**73.29**	51.3	**37.44**
	Median	61.1	**25.49**	73.4	**10.49**	62.0	**24.39**	62.0	**24.39**	60.9	**25.73**
Cifar-10	*mKrum*	34.1	**31.80**	33.5	**33.00**	27.8	**44.40**	24.6	**50.80**	24.4	**51.20**
	Bulyan	28.4	**43.30**	31.4	**37.20**	21.2	**57.60**	22.2	**55.60**	21.7	**56.60**
	TRmean	13.9	**72.20**	13.1	**73.80**	12.6	**74.80**	14.4	**71.20**	12.5	**75.00**
	Median	24.5	**51.00**	37.0	**26.00**	24.9	**50.20**	24.7	**50.60**	23.8	**52.40**
SVHN	*mKrum*	81.85	**4.83**	80.87	**5.97**	28.03	**67.41**	60.33	**29.85**	26.36	**69.35**
	Bulyan	73.03	**15.08**	50.18	**41.65**	19.66	**77.14**	19.30	**77.56**	42.70	**50.35**
	TRmean	19.59	**77.22**	46.76	**45.63**	42.54	**50.53**	42.06	**51.09**	41.13	**52.17**
	Median	59.67	**30.62**	83.63	**2.76**	43.38	**49.56**	68.08	**20.84**	65.09	**24.31**

Baselines. We are the first to propose data-free untargeted attacks. So there is no direct baseline to compare to. To demonstrate the effectiveness, we compare our results with the three state-of-the-art attacks **LIE** [7], **Fang** [32] and **Min-Max** [118] that require knowledge of benign updates or real data. We make the following choices regarding the parametrization of the defenses. As defenses are unknown to the attacker in our scenario, we implement the version of the Min-Max attack that is designed for unknown defenses and achieves the best results. For the Fang attack, the original paper assumed knowledge of the defense. We here use the version of the Fang attack that assumes *TRmean* or *Median* as the defense, which is the only source code provided by the

authors. Otherwise, we use the parameters that produced the best results in the original papers.

3.4.3. COMPARISON WITH BASELINES

***ASR* and *DPR*.** Our main results for the attack success rate and defense pass rate are shown in Tab. 3.2 and Fig. 3.4. Among all of the baseline methods, Min-Max attack is the most successful attack, with high *ASR* even on low *DPR*. In general, our experimental evaluation demonstrates that the proposed data-free attack strategies, DFA-R and DFA-G, are able to achieve similar or even slightly higher attack success rate than the baseline attacks, which require full knowledge of benign updates or a large quantity of raw data. DFA-G outperforms DFA-R in terms of *DPR* for most results on different datasets, which shows its stealthiness. During the first rounds of training, the attack is relatively weak as the global model does not yet provide a good enough model to generate effective poisoning, as indicated by our experimental results. Once the model converges, the polished model guides the attack and the attack success increases.

Specifically, from the results of Fashion-MNIST, DFA-R is better than DFA-G and all baselines when *mKrum* and *TRmean* are used to defend. *Bulyan* rejects on average more updates while *Median* merely includes the median of each model parameter from all of the clients. Both make it hard to inject malicious data into the model, leading to the low pass rate for DFA-R and hence higher effectiveness of the more stealthy DFA-G. Correspondingly, DFA-R performs better *mKrum* and *TRmean* as they allow it to pass the defense more frequently.

On the other hand, DFA-G performs well for Cifar-10 due to the fact that training Cifar-10 networks with more layers (parameters) results in slower convergence so that it favours attacks that continuously circumvent the defenses. Also, the use of 3-channel RGB data increases the diversity of benign updates. As a consequence, the level of uncertainty is generally higher during training, so that it becomes easier to pass the defense as the benign updates are not consistent enough to act as a reference point that can be used to detect malicious images. For the same reason, *DPR* of both DFA-G and DFA-R is higher on Cifar-10 than on Fashion-MNIST. However, Fang and Min-Max are not more successful on Cifar-10. Min-Max, which is aware of benign updates and hence can adapt to different datasets, already integrates dataset-specific behavior that allows it to adapt to Fashion-MNIST's low diversity. Fang rarely passes defenses, regardless of the dataset. The results of *ASR* for Fang without knowing the exact defense is consistent with the original results [32, 118].

For SVHN, both DFA-R and DFA-G achieve competitive *ASR* compared with the baselines. The only exception is Median where Min-max clearly outperforms our attacks. The result can be explained by the complexity of SVHN. SVHN is more complex than Fashion-MNIST, so it benefits from the additional knowledge Min-Max leverages to craft the update. In contrast, Cifar-10 also has a higher complexity, but experiments show that *Median* has a low accuracy for Cifar-10 even in the absence of attacks if there is data heterogeneity, as it does not include important information in the model. So it does not make so much of a difference which attack is applied. The *DPR* of DFA is lower than most other attacks for SVHN, in contrast to the other datasets. The results show that the effectiveness of the attacks depends on a combination of dataset and applied defense.

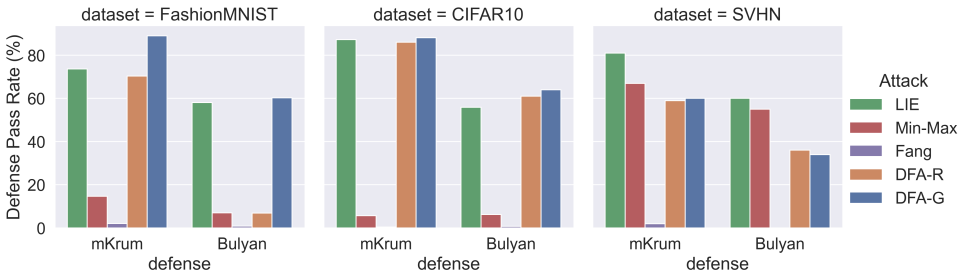


Figure 3.4: Defense pass rate (*DPR*) on Dirichlet distribution. $\beta = 0.5$ for Fashion-MNIST, Cifar-10 and SVHN.

We now consider the baseline attacks in more detail. LIE appears to be weaker than other attacks since it applies only a minor static shift to the mean of benign updates in order to pass defenses. This results in LIE's high *DPR* but it limits its attack effectiveness. In contrast, Min-Max attack trains (maximizes) the scale of the shifting from the mean of benign updates each round so as to enhance effectiveness, especially under heterogeneous data. This the reason why it achieves good *ASR* even with low *DPR*. The few times it overcomes the defense are sufficient for the crafted malicious updates to permanently damage the model. Fang attack has the least *DPR*, as it steers the global model parameters to the reverse direction. It is even more easily detected by the defenses than Min-Max, to the extent that the attack effectiveness is severely reduced.

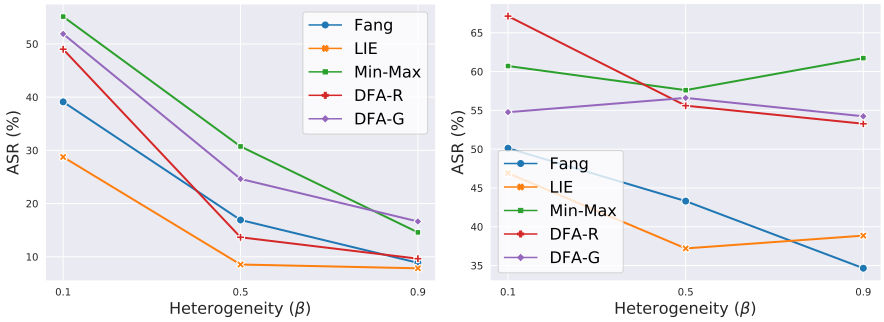
3.4.4. DATA HETEROGENEITY LEVEL

We evaluate the impact of different levels of data heterogeneity on the *ASR* of attacks. Specifically, we choose $\beta = 0.1$ as the most heterogeneous case while $\beta = 0.9$ is the least heterogeneous case. Fig. 3.5 displays the results for Fashion-MNIST and Cifar-10 when *Bulyan* is used as a defense, which is a defense our attacks usually do not achieve the highest attack success rate, as can be seen from Tab. 3.2. In general, the effectiveness for all attacks increases with an increased level of data heterogeneity, since more heterogeneity means that the benign updates are more diverse and hence detection of outliers is harder.

The global model accuracy decreases on more heterogeneous data without attacks. This is consistent with the intuitive expectation that data of higher heterogeneity in an FL system results in poorer global accuracy within the same number of training rounds.

From Fig. 3.5, we can observe that for the aggressive *Bulyan* defense, the Min-Max attack achieves mostly the best performance among all of the attacks. Attacks with full knowledge of benign updates as well as adaptive weights for maliciously shifting the mean is expected to work better. That is especially true under aggressive defenses because in contrast to our attacks, Min-Max has access to information necessary to ensure their updates are less suspicious than others. Yet, thanks to the enhanced stealthiness, DFA-G outperforms Min-Max when data is less heterogeneously distributed among clients. Accordingly, DFA-R achieves the best results when $\beta = 0.1$ on Cifar-10 dataset. In this scenario, the requirement of stealthiness is the least for all of the six scenarios because Cifar-10, as discussed above, has more diverse updates and the high degree of hetero-

geneity further increases the diversity, making it hard to detect outliers. Additionally, the *ASR* of LIE and Fang attack decreases drastically with decreased heterogeneity. LIE attack adds a static minor shift to the true mean as it is designed to attack independent and identical distribution scenarios. For more heterogeneous updates, LIE attack is more likely to pass the defense and have an impact. Fang attack usually requires knowledge of the defense; in the absence of this knowledge, it fares better when its behavior is harder to be detected. The results on SVHN dataset show similar trends with regard to data heterogeneity. As for Cifar-10, the *ASR* for $\beta = 0.9$ may exceeds the *ASR* for $\beta = 0.5$, e.g., DFA-G on SVHN has an *ASR* of 71.68% and 50.35% for $\beta = 0.9$ and $\beta = 0.5$, respectively.



(a) Fashion-MNIST (b) Cifar-10

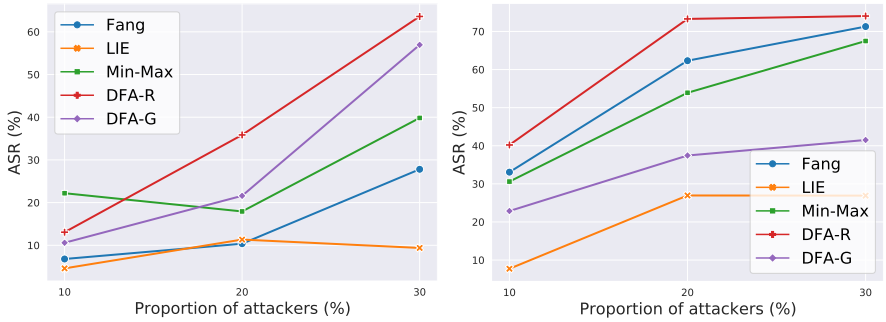
Figure 3.5: ASR(%) for different methods of attacks under different levels of training data heterogeneity on Fashion-MNIST and Cifar-10 dataset.

3.4.5. DIFFERENT PROPORTION OF ATTACKERS

In this section, we demonstrate the applicability of our proposed attack for different numbers of attackers. In order to show our effectiveness, we choose *TRmean*, which is a statistic-based defense, and *mKrum*, which is a distance-based defense, for our experimental results presented in Fig. 3.6. The results are evaluated on the Fashion-MNIST dataset. We vary the attacker proportion from 10% to 30% as we do not expect the attackers of an FL learning system to exceed 30%. To create heterogeneous data, we follow the Dirichlet distribution with $\beta = 0.5$ as in Tab. 3.2.

The results of Fig. 3.6 demonstrate the consistent effectiveness of DFA compared with other attacks. The more attackers we have, the higher the attack success rate, as one expects. Yet, DFA achieves the highest attack success rate, compared to other attacks.

DFA-R usually has the best performance, with the exception of 10% on *mKrum*, where Min-Max attack has the best *ASR*. Indeed, it is easier to defend against a smaller number of attacks in the simpler dataset. As Min-max has more knowledge about the benign updates, it is then able to send in more malicious models than DFA in this easy-to-defend case.



(a) *mKrum* (b) *TRmean*

Figure 3.6: ASR(%) for different methods of attacks under different proportions of attackers on *mKrum* and *TRmean* defense.

GENERATOR TRAINING EPOCHS

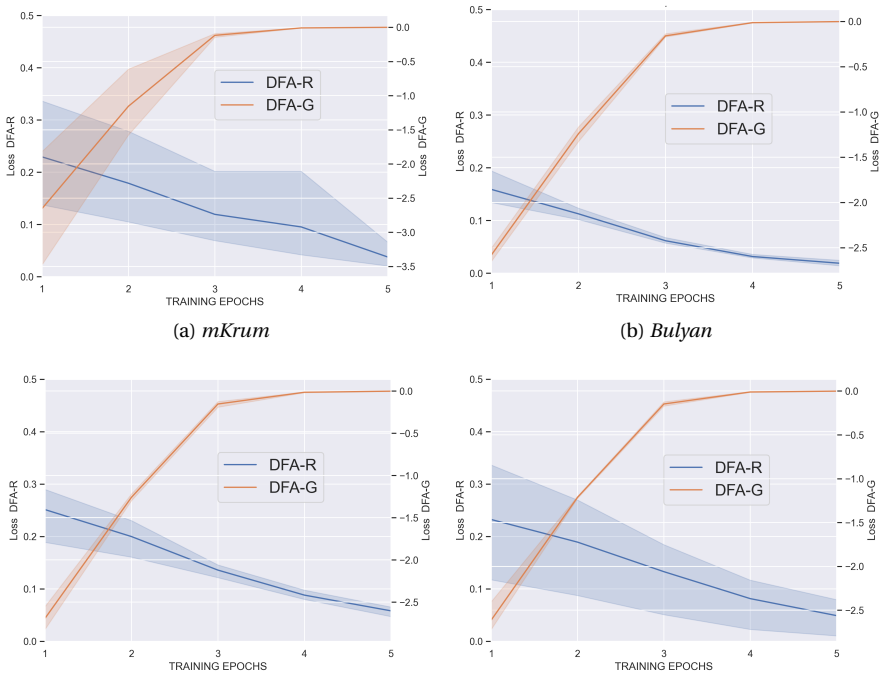
Here, we empirically investigate the convergence towards the optimal loss, where DFA-R is minimizing its loss but DFA-G is maximizing it. Fig. 3.7 shows the results for Fashion-MNIST on all four defenses. It can be clearly seen that the local training for generating malicious images converges to a local optimum. For both of our proposed attacks, DFA-R and DFA-G, we only need a few epochs to train. For DFA-R, E is 5 for Fashion-MNIST, and $E = 10$ for Cifar-10 and SVHN as Fashion-MNIST is easier to train.

COMPARISON WITH NON-TRAINING APPROACH

Given that the training converges fast, we also investigate the impact of training in comparison to just using a randomly initialized filter layer for DFA-R and a randomly initialized generator for DFA-G without any updating over rounds. As explained, according to the definition, *DPR* is measured only on *mKrum* and *Bulyan* defenses. We hence report the results for *TRmean* and *Median* as "N/A". The maximum accuracies without attack are the same as Tab. 3.2.

The results can be seen in Tab. 3.3 and confirm that training according to the current global model is indeed necessary. For DFA-R, training a single layer aims at generating images that confuse the global model. Without the training step, the injection of DFA-R is less malicious. Thus, *ASR* usually decreases without training, except for Fashion-MNIST with *Bulyan* defense. This observation is due to the fact that training DFA-R reduces the stealthiness of the attack by focusing on effectiveness and hence DFA-R passes *Bulyan* more often without training. This is consistent with our results in Fig. 3.4 that *Bulyan* significantly reduces the *DPR* of DFA-R.

When it comes to DFA-G, training helps to enhance stealthiness. The impact can be clearly seen from the results for *DPR* in Tab. 3.3, especially for *Bulyan*. Only for a relatively lenient defense like *mKrum*, the training has little additional impact as *DPR* is already high without training. These results also reflect the minor increase of *DPR* from Fashion-MNIST to Cifar-10 dataset for *mKrum* in Fig. 3.4.



(a) *mKrum* (b) *Bulyan*

(c) *TRmean* (d) *Median*

Figure 3.7: Local training process of both DFA-G and DFA-R on Fashion-MNIST.

IMPACT OF THE REGULARIZATION TERM

In this part, we conduct an ablation study for our proposed distance-based loss, which adds a regularization term to the original cross-entropy loss function. Tab. 3.4 shows both *ASR* and *DPR* with and without the regularization term on Fashion-MNIST. For DFA-R, the effectiveness of the regularization term is more apparent for *mKrum*. However, for DFA-G, the increase is most notable for *Bulyan*. This is because the regularization term in the less stealthy DFA-R is insufficient for passing *Bulyan* whereas it is what enables DFA-G to pass *Bulyan* frequently. In contrast, DFA-G does not require the regularization term for *mKrum* as it already passes without extra regularization.

SYNTHETIC VS REAL DATA

In order to demonstrate the effectiveness of our malicious synthetic data, we compare the *ASR* of our attacks to a version of the attack that uses real data, i.e., we use a set of real images instead of the synthetic image set S. We assign the number of real images owned by the attackers under the same Dirichlet distribution as for benign users. The results for the four defenses on both datasets are shown in Fig. 3.8 with stripped visualization. "Real-data" in the figure refers to the results of *ASR* using real data paired with the uniformly chosen label $\tilde{Y}$ to train $\boldsymbol{w}(t)$ with distance-based loss as described in Sec. 3.3 similarly for the synthetic data. Fig. 3.8 shows the effectiveness of our malicious

Table 3.3: *ASR* and *DPR* for (non-)training approach where "Static" refers to non-training way with only randomly initialized. "Fashion" and "Cifar" is short for Fashion-MNIST and Cifar-10 datasets.

		Static		**Trained**	
Attack	**Defense**	ASR(%)	DPR(%)	ASR(%)	DPR(%)
DFA-R Fashion	*mKrum*	18.17	87.78	35.85	70.33
	TRmean	37.20	N/A	73.29	N/A
	Bulyan	23.66	57.50	13.66	6.86
	Median	21.22	N/A	24.39	N/A
DFA-G Fashion	*mKrum*	17.07	88.33	21.59	89.02
	TRmean	30.73	N/A	37.44	N/A
	Bulyan	24.88	65.26	27.07	69.33
	Median	22.44	N/A	25.73	N/A
DFA-R Cifar	*mKrum*	50.00	85.20	50.80	86.04
	TRmean	71.14	N/A	71.20	N/A
	Bulyan	56.00	60.98	55.65	61.05
	Median	48.60	N/A	50.60	N/A
DFA-G Cifar	*mKrum*	38.60	56.46	51.20	88.14
	TRmean	71.40	N/A	75.00	N/A
	Bulyan	47.80	37.35	56.60	63.99
	Median	50.60	N/A	52.40	N/A

Table 3.4: *ASR* and *DPR* for ablation test of the regularization term proposed by our distance-based loss.

		without regularization		**with regularization**	
Attack	**Defense**	ASR(%)	DPR(%)	ASR(%)	DPR(%)
DFA-R	*mKrum*	17.68	41.92	35.85	70.33
	TRmean	58.78	N/A	73.29	N/A
	Bulyan	10.73	3.32	13.66	6.86
	Median	23.72	N/A	24.39	N/A
DFA-G	*mKrum*	20.98	87.34	21.59	89.02
	TRmean	31.71	N/A	37.44	N/A
	Bulyan	22.32	60.27	27.07	69.33
	Median	23.78	N/A	25.73	N/A

synthetic data generated by DFA-R and DFA-G as *ASR* outperforms the case of using real images. That is expected because our synthetic images are specifically constructed such that the attack is very effective but at the same time stealthy. Thus, even if data is present at the attacker, a data-free attack can be the better choice. Consequently, it is usually not necessary for the attacker to invest the overhead of obtaining data.

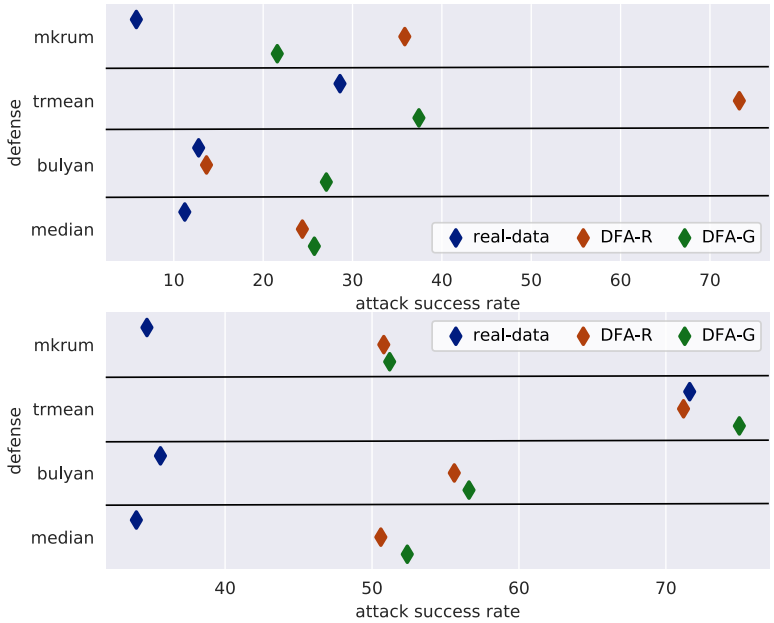


Figure 3.8: Comparison of *ASR* (%) of real data and synthetic data by DFA-R and DFA-G with four defenses on Fashion-MNIST and Cifar-10.

3.5. DEFENSE FOR DFA: REFD

Based on the results of the previous section, our attack is highly effective against known defenses. Yet, the attack might not withstand defenses that are crafted with data-free attacks in mind. Thus, in this section, we design and evaluate a novel defense that specifically addresses the reasons why existing defenses are insufficient.

Let us first state why existing defenses fail. *mKrum* and *Bulyan* reject updates that differ greatly from others. Our regularization term ensures that our updates do not differ too much from the global model from the previous round, which at least once the model starts to converge is close to the models submitted by benign clients. Statistical methods lose information about the distribution but medians or trimmed means also shift easily without the need for an attacker to provide outlier data [7].

3.5.1. DESIGN OF REFD

The defense is designed with data-free attacks in mind and overcomes the drawbacks of existing defenses. We rely on a reference dataset $\mathcal{D}_r$ to detect unusual classification

patterns. Upon receiving an updated model from clients, the server uses that model and executes the model inference on $\mathscr{D}_r$, i.e., they compute predicted class probabilities. We design a novel statistic, D-score, which identifies the model updates whose outputs have either biased prediction or low confidence. To evaluate the effectiveness of REFD, we apply REFD on both our proposed attacks and the state-of-the-art attacks, for both balanced and heterogeneous data distributions.

Assumptions. REFD is a server-side defense. REFD requires that the server owns a small reference set $\mathscr{D}_r$ of real data with correct labels. The quantity of each class label is assumed to be balanced.

The design core of REFD is the D-score for each model update received: a low score indicates a high risk of the update being malicious and results in the server rejecting the update. The D-score is computed based on two parts: the balance value and the confidence value of updates. The balance value determines how balanced the outputs from the updated model are, to detect updates that are biased toward a specific class, such as DFA-G, LIE [7], and Min-Max [118]. The confidence value measures the confidence in predicting a class and rejects updates that result in low confidence, which are the objective of DFA-R and Fang [32].

Before explaining those values, we first explain background notations. $\boldsymbol{w}_i$ is defined as the updated classifier model received from the client i, which maps the data input into two kinds of output. Specifically, $\boldsymbol{w}_i^p(\cdot)$ maps the data into the per class probability vector, and $\boldsymbol{w}_i^h(\cdot)$ maps the data into the per class one-hot encoding vector. Both vectors have a length of L, corresponding to the number of classes.

Balance value. To tackle the attack type that causes bias in classification, we define the balance value B_i for the updated model of client i as the inverse of the standard deviation of the class label distribution:

$$B_i = \begin{cases} \dfrac{1}{std(A_i)}, \text{if } std(A_i) \neq 0 \\ 1, \text{if } std(A_i) = 0 \end{cases} \tag{3.6}$$

where $std(\cdot)$ is the standard deviation over all class labels and A_i consists of the aggregated number of predicted labels of each class. For instance, A_i for Cifar-10, is a set of 10 values, i.e., the number of predicted samples per class. To compute A_i, we first apply $\boldsymbol{w}_i^h$ on each sample in D_r and aggregate them. Overall, the non-biased output prediction from benign clients results in a better balanced A_i across all classes and thus a higher value of B_i. The adversarial parties on the other hand should have a lower value of B_i.

Confidence value. This value quantifies the average confidence when using the updated model on $\mathscr{D}_r$. Specifically, for a data sample j in $\mathscr{D}_r$, we let the confidence of applying classifier of client i, M_{ij} as the biggest element of the output probability vector $\boldsymbol{w}_i^p(\mathscr{D}_r(j))$. We thus define the confidence value, V_i, as:

$$V_i = \frac{1}{|\mathscr{D}_r|} \sum_{j=1}^{|\mathscr{D}_r|} M_{ij}. \tag{3.7}$$

Low confidence values indicate higher risks of being adversarial updates. The potential drawback of using V_i to detect adversarial behavior is that low confidence may also appear in the early training epochs of honest clients.

D-Score. For each client update, we combine the balance value, B, and the confidence value, V, to detect a wide range of adversarial behaviors. Motivated by F_β Score [116], we define the D-Score to evaluate the quality of an intermediate training model from client i as:

$$D - Score = (1+\alpha^2) \times \frac{B_i \times V_i}{\alpha^2 B_i + V_i}, \tag{3.8}$$

where α is a hyper-parameter to weigh the importance between balance value and confidence value. It can be set as a specific value according to what the central server knows or suspects about the executed attack. It can also be adaptive and learned over epochs, but we consider this out-of-scope for the chapter and a good avenue for future work. Instead, we set $\alpha = 1$ to represent the equal importance of B_i and V_i. If the predictions are perfectly balanced and have high confidence, we have a D-Score of 1. When B_i is reduced while V_i stays constant, the D-Score is reduced, mirroring the increased bias. Analogously for V_i, a lower value for V_i leads to a lower D-Score to indicate the lower confidence. Moreover, as we designed the defense with data-free attacks in mind, we expect it to work better for those than for other attacks, for which defenses already exist.

Removing attackers. After calculating the D-Score for each update of a given round, the server rejects the updates with the X lowest D-Scores. The server then excludes them for aggregation. The method is the same as used by *mKrum* [9]). X is determined by the server's assumptions about the fraction of attackers, i.e., the more attackers they expect, the higher they choose X.

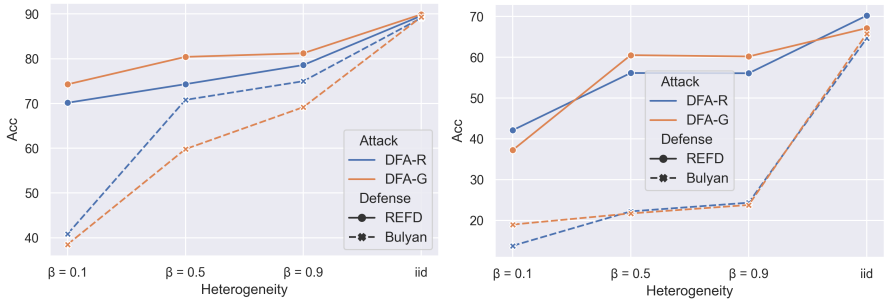
3.5.2. EVALUATION FOR REFD

Experimental settings. To evaluate the effectiveness of REFD, we compare the accuracy of the global model in the presence of the new defense. We use the full test set for the respective dataset in the presented results but also experimented with smaller reference datasets (1000 images instead of 10000) and found no significant difference. Hence, smaller datasets can be used to increase efficiency and lower the requirements in terms of data availability at the server side. However, the reference set has to be balanced among class labels to compute the balance value reliably. REFD is evaluated on both Fashion-MNIST and Cifar-10 datasets. Additionally, our experiments include four different levels of data heterogeneity: independent and identical distributed (*i.i.d*) and three heterogeneity levels ($\beta = 0.1, 0.5, 0.9$, where $\beta = 0.1$ indicates the highest level of heterogeneity) as in Sec. 3.4.4. We also evaluate the impact of different levels of data heterogeneity since defenses are sensitive to the heterogeneity, especially for distance-based defenses. Intuitively, a higher level of training data heterogeneity makes defense more difficult. The robustness of a defense in the presence of high data heterogeneity is important to various real-world application scenarios.

As in other works [32, 118], we set the proportion of attackers in the system to be 20% and $X = 2$. We compare REFD against *Bulyan*, the most effective SOTA defense for our attack.

Results for defense DFA.

From Fig. 3.9, we see that REFD significantly outperforms *Bulyan*. The advantage of our defense is obvious when the heterogeneity of the data is high. For $\beta = 0.1$ and Fashion-MNIST, REFD achieves an accuracy of more than 70% when the attack is DFA-



(a) Fashion-MNIST (b) Cifar-10

Figure 3.9: Accuracy(%) for REFD on Fashion-MNIST and Cifar-10 datasets with different levels of data heterogeneity, compared with the maximum accuracy under *Bulyan* defense.

R and close to 75% for DFA-G. In contrast, the accuracy of *Bulyan* is only around 40%. *Bulyan* is relatively effective for *i.i.d* data, achieving a similar value as REFD for both attacks.

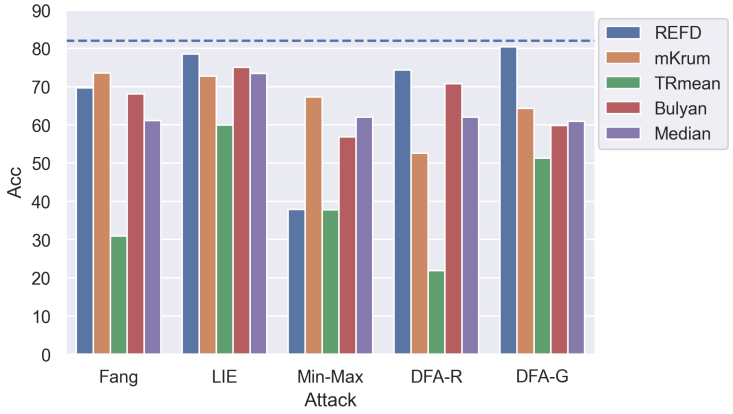
The results on Cifar-10 confirm the superiority of REFD. As noted in Sec. 3.4, the accuracy on Cifar-10 is generally lower. Yet, REFD is clearly the more effective defense. In the presence of data heterogeneity, the accuracy is at least twice as high for REFD than *Bulyan*.

For both datasets, the advantage of REFD is least pronounced for the *i.i.d* setting. For *i.i.d*, the benign updates are very similar, hence making it barely possible for our attacks to deviate without being detected. Thus, there is little difference between defenses. The results for Fashion-MNIST and Cifar-10 differ in that the final global model accuracy difference between REFD and *Bulyan* is larger for Cifar-10. The data complexity of Cifar-10 is higher, increasing the difficulty of defending so that our specifically designed defense shows a more pronounced advantage.

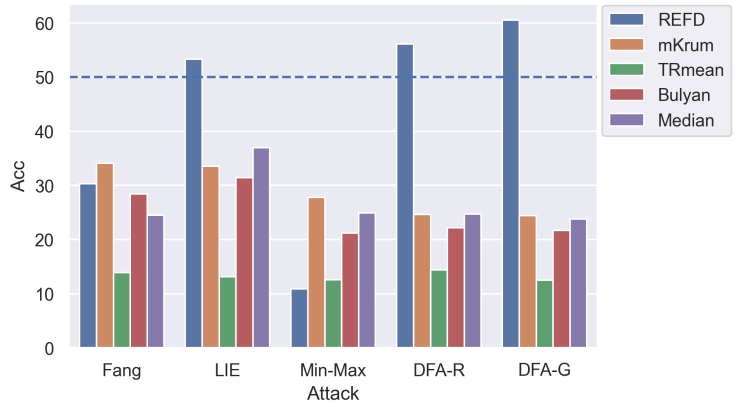
The achieved accuracy is close to the accuracy achieved without attacks and defenses. For instance, for $\beta = 0.5$, the accuracy without attacks and defenses is 82% for Fashion-MNIST, which is less than 2% higher than the accuracy for REFD for DFA-G. For DFA-R, the attack decreases the accuracy by about 10%. For Cifar-10, the accuracy with and without attacks is almost equal. Indeed, the accuracy with attack and defense is insignificantly higher for some settings, which is likely due to randomness.

Defending against other attacks. REFD is designed with DFA in mind, as our goal is to show that we can defend against data-free attacks. The design does not necessarily work against all attacks. Here, we establish whether the defense can nevertheless defend against Fang, LIE, and Min-Max attack and compare it with the results for DFA.

The results are reported in Fig. 3.10. We also follow the setting of 20% attacking proportion and compare the maximum global model accuracy for the state-of-the-art defenses and REFD. We include the baseline accuracy with no attack and no defense as the dashed line. From Fig. 3.10, we can see that REFD has a good defending performance in general. However, it is not always the best among the state-of-the-art defenses. Specifically, for the LIE attack, REFD gets the best defending performance. LIE shifts the



(a) Fashion-MNIST



(b) Cifar-10

Figure 3.10: Comparison of Accuracy(%) for defenses with state-of-the-art attacks on Fashion-MNIST and Cifar-10 datasets.

true statistical features of the benign updates, which can easily be caught by the balance value B of REFD. Moreover, REFD also protects the model from the Fang attack, where it achieves the second-best ranking on both datasets. REFD works well for Fang since Fang updates malicious models in the opposite direction, which causes low confidence, i.e., low V. However, REFD is less effective against Min-Max than other defenses, as Min-Max's scaling technique should not affect balance and confidence value much. In summary, REFD protects well against data-free attacks presented in this chapter and can also protect against other attacks. However, it is not a generic defense and hence should be applied in combination with other defense mechanisms. It is also interesting to note that with RefD, the global model accuracy can even be higher than the baseline on Cifar-10. This result implies that RefD has benefits in the presence of data heterogeneity in comparison to FedAVG.

3.5.3. OVERHEAD ANALYSIS FOR DEFENSE

The defense does not add any communication, so merely the computation complexity is affected. The defense first evaluates the local update of each client for each image in the reference dataset, so the cost is $\mathcal{O}(|\mathcal{D}_r|K)$ times the cost of evaluating the update. Furthermore, the D-Score needs to be computed, which is linear in $\mathcal{O}(|\mathcal{D}_r|)$ as we compute the standard derivation (for B) and the maximum (for V) of $\mathcal{O}(|\mathcal{D}_r|)$ values. Last, we determine the clients with the smallest values, which has complexity $\mathcal{O}(K)$. Overall, evaluating updates is of a lower complexity than training new models, so the overhead is not prohibitive and can be reduced by using a smaller set $\mathcal{D}_r$.

3.6. CONCLUSION

We propose DFA, the first data-free untargeted attack on FL. Our results confirm that data-free attacks can be similar or even more effective than other attacks that require data or benign updates, due to generating synthetic images to train on that are particularly useful at steering the model in the wrong directions. Furthermore, we design a defense strategy REFD that effectively protects against the proposed DFA and existing attacks by leveraging the statistics of model outputs in predicting reference data. In the future, we want to explore DFA on different data types, e.g., text, and check whether combining synthetic and real data in an attack can improve attack effectiveness and to what extent data is needed in a defense.

4

Privacy Risk of Data Reuse in Multi-server Federated Learning

Federated learning enables clients to collaboratively learn models without revealing their local data. However, the shared model updates still reveal information about the data set, as indicated by a number of attacks on privacy. While privacy in the context of single data use is well-studied, users may provide the same data for multiple tasks. Hence, we focus on the case when data is re-used in the presence of multiple colluding servers, either the same or the different training tasks. We develop Collusive Gradient Inversion (CGI), an attack that combines multiple gradients computed on the same data to reconstruct the original data. The theoretical bound on how privacy leakage increases with the number of re-use is analyzed for the same task reconstruction. We then show that Nash bargaining games are effective in determining aggregation weights while integrating contributions from different tasks. We experimentally validate the increased quality of the reconstructed image in comparison to single-server reconstruction, both with and without defense mechanisms.

4.1. INTRODUCTION

Federated learning [59, 155] collaboratively trains a machine learning model without sharing the raw data. As a consequence, it enables learning from distributed sensitive data, such as medical [112] or financial [64] data that cannot be shared due to the serious implications of data leakage for users. A typical FL architecture consists of a single server and multiple clients, who jointly learn a single task over multiple global rounds and only exchange the intermediate model updates instead of raw data.

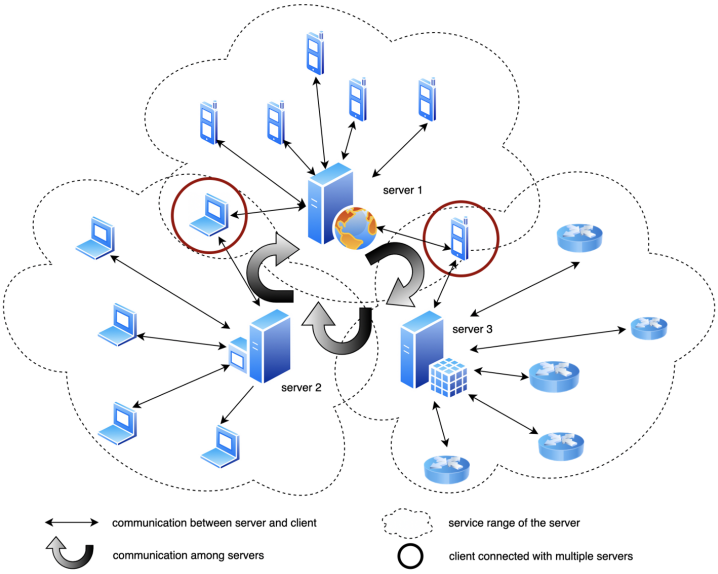


Figure 4.1: Multi-server Federated learning with data-reusing clients submitting updates to multiple servers. The tasks offered by the servers may be the same or different.

In this setting, data is only used to train one task provided by the single server. It has been shown that even contributing to single-server FL by exchanging model updates is vulnerable to a number of attacks on privacy. In particular, data reconstruction attacks, also called gradient inversion attacks [39, 165] aim to reconstruct the raw data, completely defeating the purpose of FL, which originally aims to hide this exact data. However, for a single server to successfully reconstruct high-quality data usually requires external knowledge such as a well-trained generative adversarial network [31, 66] or data owned by the server that highly resembles the target user data [146]. Furthermore, recent single-server gradient inversion attacks are limited as they can only handle small batch sizes [178, 172, 159], with training always being executed on one batch at a time. Indeed, attacks on batch sizes of above 48 have only been proven successful even when the attacker had external knowledge [66, 83, 31].

Till now, no work has studied the increased privacy risk of gradient inversion caused by data reuse in the context of FL, which is common in practice. A server might train multiple tasks, either the same task, i.e., to evaluate whether results vary, or different tasks, by reusing the same data [11]. Examples of such scenarios include self-driving cars [65].

In addition to considering the same server with multiple different tasks, the emerging multi-server architecture [89, 43] has multiple collaborative servers that first coordinate the learning for clients within geographical vicinity and then aggregate their local results to improve latency and reduce communication overhead. There can be clients that are within the vicinity of two or more servers and hence send updates to all of them, by reusing the same local data, as illustrated in Fig. 4.1. Furthermore, research confirms that client selection affects the model performance drastically [16], meaning that different servers (e.g., companies) may choose to train on different but overlapping sets of clients for the same or similar tasks.

Thus, in this chapter, we focus on investigating the risk of data reconstruction in repeated usage of images for FL. While our scenario includes both single server and multiple colluding servers, we generalize by associating each task with one (virtual) server, resulting in the *Multi-server Federated Learning* (MSFL) setting described in Fig. 4.1. We assume that the servers are honest-but-curious and collude in running a data reconstruction attack.

Our multi-server **C**ollusive **G**radient **I**nversion (CGI) attack initializes dummy data randomly and then iteratively adapts the dummy data such that it resembles the original data. Concretely, for one specific global training round, all servers receive task gradients from a common client. They then compute the inversion gradients based on the losses of task gradients and dummy gradients, which are obtained by back-propagating through the dummy data. These inversion gradients are then aggregated by one of the colluding servers to update the dummy image. For the aggregation, there are two variants of CGI: CGI-S considers the case when the learning tasks are the same and CGI-D is the case when they are different. If the tasks are the same, the aggregation simply averages over the gradients. If the tasks are different, the updating direction and gradient magnitude can differ as well, which may prevent the convergence of the attack optimization. We model the process of agreeing on weights for different-task gradient aggregation as a Nash bargaining game. After the aggregation, the dummy data is updated and the process repeated until a maximal number of iterations is reached or the distance between dummy data and real data is extremely small.

We extensively evaluate CGI on four datasets and compare it against four optimization-based state-of-the-art single-server reconstruction attacks in the presence of four common privacy defense mechanisms. To assess how well our reconstructed images resemble the original data, we consider four common similarity measures for (image) data (MSE, LPIPS, PNSR, and SSIM) to cover various definitions for similarity. For all metrics, CGI outperforms the baseline methods considerably, especially for large datasets with complex deep neural networks, where single-server algorithms fail. For shallow nets with small data input, CGI-S usually requires fewer iterations than the existing approaches, frequently reconstructing images that closely resemble the original according to all four metrics in 10 iterations. Batch sizes of up to 64 can be handled, an increase of 33% in comparison to single-server attacks.

We are the first to conduct the impact of data reuse on gradient inversion attacks, establishing both a novel model for covering such attacks and providing the first results. We further determine a theoretical bound on how the privacy leakage increases with the number of servers and design the first aggregation methods for a collusive attack based

on Nash bargaining game.

4.2. MULTI-SERVER FEDERATED LEARNING SYSTEM

In this section, we first introduce background knowledge on Federated Learning and data reconstruction attacks. Then, we model multi-server FL systems in the context of existing frameworks, followed by popular privacy leakage defense for FL. Last, our threat model is described in detail.

4.2.1. FEDERATED LEARNING

In vanilla single-server FL, each client $C_i \in \mathscr{C}, i \in [1, N]$, utilizes its local data to train the model initialized by the single server $\mathscr{S}$ without sharing the data $D_i = \{(x_n^i, y_n^i)\}_{n=1}^{|D_i|} \in \mathbb{R}^{|D_i|}$ itself. The updated local models are aggregated by the server and distributed iteratively for multiple global rounds until convergence. Denote the model at the r^{th} global training round as ω^r, and let η be the global learning rate, so

$$\omega_i^{r+1} = \eta F(\omega^r, D_i), \tag{4.1}$$

where F is the function to update the task model, e.g. a classification task or a generative task. Using the common FedAvg aggregation [96] weighted by data quantity, the server aggregates the models from multiple clients as:

$$\omega^{r+1} = \frac{\sum_{i=1}^{N} |D_i| \omega_i^{r+1}}{\sum_{i=1}^{N} |D_i|}, \tag{4.2}$$

The training process continues for R global rounds.

4.2.2. GRADIENT INVERSION

Recent studies show that an honest-but-curious server is able to reconstruct raw training data from clients based on their submitted gradients [178, 172, 39]. This process of reconstruction is called gradient inversion. Let $\mathscr{L}$ be the loss function, then the gradient of client C_i is:

$$g_i = \nabla \mathscr{L}(\omega^r, D_i) = \eta \frac{\partial F(\omega^r, D_i)}{\partial \omega^r}. \tag{4.3}$$

We here consider two-dimensional images of width I and height J, with K indicating the number of possible pixel colors. To reconstruct D_i, $D_i \in \mathbb{R}^{I \times J \times K}$, the attacker first initializes the dummy image $\tilde{D}_i \in \mathbb{R}^{I \times J \times K}$. Similarly, the dummy gradient is computed as

$$\tilde{g}_i = \nabla \mathscr{L}(\omega^r, \tilde{D}_i). \tag{4.4}$$

As the server already knows the model ω, to reconstruct the data, the gradient inversion attack minimizes the distance of received and dummy gradient:

$$\min Dist(\tilde{g}_i, g_i) = Dist(\nabla \mathscr{L}(\omega^r, \tilde{D}_i), g_i), \tag{4.5}$$

by optimizing $\tilde{D}_i$. The distance measure $Dist$ for two gradients is usually L2-Norm or cosine similarity. Modern learning paradigms use batched data to train models. Generally, the reconstruction of batched (size B) data regards the input as a $B \times I \times J \times K$ data and also use the same dummy data size.

4.2.3. SYSTEM MODEL

In this chapter, we look at a general system where a client may contribute the same data to multiple training tasks proposed by different servers, referred to as Multi-server FL. Concretely, we have a set of servers $\mathscr{S}$ with $|\mathscr{S}| > 1$ and a set of tasks $\mathscr{T}$ such that each server S_k manages one task T_k. There is a set of clients $\mathscr{C}$ with subsets $\mathscr{C}_k$ such that $\mathscr{C}_k$ consists of all clients that contribute to task T_k. We assume that at least one client contributes to more than one task. If a client joins two tasks T_k and T_j, we can have $T_j = T_k$ or $T_j \neq T_k$. We call the first case **Same-Task MSFL** (sMSFL) and the second case **Different-Task MSFL** (dMSFL). Note that different tasks still require that the client can reuse the same dataset. For instance, the same face images could be used to predict gender as well as sexuality.

We assume that parties are incentivized to participate in multiple tasks through rewards, e.g., the server may provide monetary compensation. There are a number of strategies for assigning rewards in the context of FL [163, 22, 75]. As our work is orthogonal to the topic of reward allocation, we do not integrate reward allocation in our evaluation but merely assume that it is sufficient to encourage participation.

4.2.4. DEFENSES AGAINST INVERSION

The goal of defense mechanisms is to *i)* ensure system resilience against privacy leakage attacks, *ii)* without significantly affecting the model accuracy on the learning tasks [130]. State-of-the-art defenses are conducted by individual clients without collaboration. Specifically, defenses can be classified into gradient perturbation [156, 131, 130, 3], where gradients are transmitted after local perturbation, or input perturbation [166, 63], which for instance linearly mixing up images from the same dataset before local training. Among the above two, gradient perturbation is more commonly used due to its efficiency and global model accuracy.

4.2.5. THREAT MODEL

We assume encrypted and authenticated communication channels between clients and servers. We assume all the clients to be honest and the servers to be honest-but-curious. Servers collude, meaning they share the updates they receive. To conduct gradient inversion attacks, colluding servers are not required to share all updates, attacks on a single-round update are sufficient.

Capabilities: The servers do not have real data. Moreover, the servers' computational resources are bounded, they can hence only execute (probabilistic) algorithms that are polynomial in the input.

Knowledge: We assume that servers can identify clients, e.g., by IP address, and can hence determine whether a client contributes to multiple tasks. The servers also naturally own the global model and is able to access the model parameters.

Objective Let C_i be a client contributing to multiple tasks. Without loss of generality, let $T_1, \ldots, T_K$ be the tasks that C_i contributes to. Given local models $(\omega_{i,k})_{k=1..K}$ or gradient updates $(g_{i,k}(\cdot))_{k=1..K}$, which are trained on the same local dataset D_i of client C_i, the servers $\mathscr{S}$ design an inversion method to reconstruct client's training data D_i based on the gradients in global round r.

4.3. RELATED WORK

Based on the gradients shared by clients in FL, a server is able to break the user-level privacy of clients [178, 46], which unexpectedly leaks local data information to the server. On the highest level, gradient privacy leakage attacks can be classified into *feature leakage* or *data leakage*. Feature leakage attacks infer merely specific features or statistics about the original data [48, 120, 97] instead of fully reconstructing the data. In this chapter, we focus on data leakage attacks, which are stronger as they infer the actual data the model was trained on. Data leakage attacks can be classified into recursion-based and optimization-based attacks.

Optimization-based attacks. The reconstruction process is formulated as an iterative optimization problem for dummy data. The earliest Deep Leakage from Gradient (DLG) attack [178] reconstructs the dummy data and the label at the same time. Optimization on gradients requires the global model to be second-order differentiable. There are three directions to improve gradient inversion: *i)* changing optimizer/objective function, *ii)* integrating prior information, or *iii)* adding more external knowledge. The difference between *ii)* and *iii)* lies in the origin of the knowledge. For *i)*, DLG-Adam [153] is a variant of DLG, which changes the optimizer from L-BFGS to Adam to improve the performance for larger global model networks. InvG [39] changes the distance metric from using L2 Norm to cosine similarity to fit industrial realistic scenario in deep and non-smooth global model architectures. For *ii)*, iDLG [172] infers labels from the gradient rather than optimizing the dummy label to improve the efficiency and accuracy of DLG. This method is limited to batch size 1 as it cannot infer batch labels. InvG [39] adds total variation [114] prior knowledge, with a theoretical guarantee for inversion on linear fully-connected network. As for *iii)*, GradInversion [159] adds a batch normalization regularizer to iDLG as extra knowledge to achieve reconstruction for large batch size of up to 48. GIAS [66] utilizes prior data distributions and improves the reconstruction quality of GradInversion using a pre-trained generative adversarial model (GAN). GGL [83] and GIFD [31] further fine-tune the use of such external GANs. AGIC [153] approximates gradient updates to simplify the procedure and leverage gradients from multiple epochs.

In summary, most optimization-based inversion attacks only work on small batch sizes. Moreover, guarantees that inversion is possible are only given for linear networks [39]. Our focus in this chapter is to uncover the increased risk of re-using the same data to join multiple learning tasks and is independent of research on external knowledge, which can also be used to improve our attack. Therefore, we mainly compare our work with *i)* and *ii)*.

Recursion-based attacks. The key idea is to exploit the implicit relationships among the input data, model parameters, and gradients of each layer to find a mapping solution the minimal error. The attack was first constructed for shallow nets [108] and then extended to convolutional neural networks by converting the fully connected layers to convolutional layers [30]. To provide theoretical guarantees of reconstruction, R-GAP [177] analyzes and provides a closed-form recursive procedure. Overall, recursion-based methods do not work on models with pooling layers or batch sizes larger than 1. Moreover, they are sensitive to gradient perturbations and thus are ineffective in the presence of defenses [31].

4.4. COLLUSIVE INVERSION FOR SAME-TASK

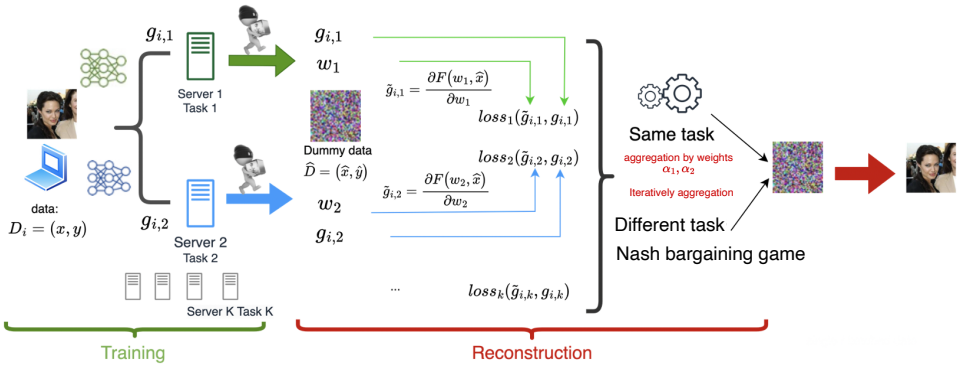


Figure 4.2: Collusive data reconstruction attack for Multi-server FL. The client contributes the same local data to multiple learning tasks (e.g., Task 1 and Task 2 at two different servers in the figure), and the servers reconstructs the data from a dummy image by minimizing the loss of the dummy, based on the received gradients $g_{i,k}$. Aggregation varies depending on whether the tasks are the same or different.

We now introduce our solution for multi-server gradient inversion protocol as **Collusive Gradient Inversion** (CGI), which reconstructs the re-used data. Fig. 4.2 illustrates the two phases of our attack: *i)* training phase, when the client submits gradients to multiple servers, and *ii)* reconstruction phase, where servers collude to optimize the dummy data. The two variants of our attack are CGI-S for sMSFL and CGI-D for dMSFL. The difference between CGI-S and CGI-D is the way to aggregate updates from multiple servers. In this section, we start with the more straightforward CGI-S, where all servers learn the same task independently. We describe the collusive inversion process and analyze it theoretically. Our theoretical results show that q servers are able to reconstruct data if $\nabla\mathscr{L}$ is a polynomial of degree q, meaning that the more servers collude, the more complex networks can be handled.

4.4.1. COLLUSIVE INVERSION STEPS

Alg. 2 lists the steps of our attack for both CGI-S and CGI-D. Multiple servers reconstruct client data merely from received gradients collusively. Every server calculates the dummy gradient on the same dummy data, according to each client's gradients and one master server node executes the gradient aggregation. The master server can be selected randomly between the servers. If there is only one server offering multiple tasks, the server trivially acts as the master. Note that the inversion is launched at one round r and we skip the coefficient r for simplicity.

In the **training phase**, client C_i contributes to multiple learning tasks by submitting trained gradients (Line 12). Thus, each server S_k receives gradients $g_{i,k} = \nabla\mathscr{L}_k(\omega_k, D_i)$. In order to launch an inversion attack, at the beginning of **reconstruction phase**, the master server generates the common random dummy data $\tilde{D}$, which is of the same size as D (Line 13), and distribute it to collusive servers. At the same time, the aggregation weights for collusion are initialized. After receiving the task gradient $g_{i,k}$, each server S_k calculates the dummy gradients by inputting $\tilde{D}_i$ to the most recent global model ω_r,

Algorithm 2: Colluded gradient inversion for MSFL, with CGI-S for sMSFL and CGI-D for dMSFL.

Data: ω_k, $g_{i,k}$, μ, $\mathscr{L}_k(\cdot)$ for server $k \in 1,2,...,K$, for client C_i.
Result: $\tilde{D}_i \sim D_i$: reconstructed data, which is similar to the original data D_i.

12 calculate $g_{i,k} = \nabla \mathscr{L}_k(\omega_k, D_i), \forall k \in K$;
13 initialize dummy data $\tilde{D}_i$, aggregation weights $\sum_k \alpha_k = 1$;
14 **for** *iteration* τ *= 1, 2, ..., T* **do**
15 calculate $\tilde{g}_{i,k} = \nabla \mathscr{L}_k(\omega_k, \tilde{D}_i), \forall k \in K$;
16 calculate the distances $Dist(g_{i,k}, \tilde{g}_{i,k})$.
17 use back propagation to get the gradients g_k for $\tilde{D}_i$ respectively.
18 **if** D_i *is reused by same task* **then**
19 assign $\alpha_k = 1/K$ as in Sec. 4.4;
20 **if** D_i *is reused by different task* **then**
21 **for** *epoch e = 1, ..., E* **do**
22 optimize α_k by Nash bargaining game in Sec. 4.5;
23 aggregates gradient $\tilde{g} = \Delta \tilde{D}_i = \sum_{k=\{1,...,K\}} \alpha_k g_k$;
24 update dummy data $\tilde{D}_i^{\tau+1} = \tilde{D}_i^{\tau} + \mu \Delta \tilde{D}_i^{\tau}$.

as $\tilde{g}_{i,k} = \nabla \mathscr{L}_k(\omega_k, \tilde{D}_i)$ (Line 15). The goal of updating $\tilde{g}_{i,k}$ is to approximate $g_{i,k}$. As illustrated in [39], cosine similarity is better suited to deep and non-smooth global model architectures, which are commonly used for real-world industrial applications. Thus, we apply the cosine similarity distance $Dist(g_{i,k}, \tilde{g}_{i,k})$ [153, 39] as the loss of updating $\tilde{D}_i$ (Line 16). As different servers have different losses, the gradients g_k from multiple servers need to be aggregated. In the context of same-task MSFL, servers are learning the same task and the same network structure. Thus, we do not expect large differences in magnitudes and optimization directions of the gradients. Thus, for CGI-S, we assign the aggregation weight $\alpha_k = 1/K$ to all servers (Line 19). On the other hand, for CGI-D, we assign weights according to Nash bargaining game (Line 22), as described in detail in Sec. 4.5.

The weights assigned for each task are used afterward for weighted aggregation (Line 23). Then, the dummy data of iteration $\tau+1$ is updated using the learning rate μ (Line 24). The dummy data update is conducted multiple times until a convergence criterion is reached. In Alg. 2, we repeat the update a fixed number of rounds for simplicity but alternative methods to determine convergence can be integrated trivially.

We analyze the bounded reconstruction output. Additionally, we show via experiments that this simple combination of loss functions works well for effectively reconstructing client data.

4.4.2. ANALYSIS ON RECONSTRUCTION

We analyze whether our approach provides any guarantees on approximating the original image. Given the loss function of $\mathscr{L}(\cdot)$, the real gradient from C_i for task T_k is $g_{i,k} = \nabla \mathscr{L}(\omega_k, D_i)$ and the dummy gradient aggregated is $\tilde{g}_k = \nabla \mathscr{L}(\omega_k, \tilde{D}_i)$. We want to

i) prove that $\tilde{D}_i$ approximates D_i and ii) provide a bound on the tightness of the approximation. We first consider models with a linear mapping function and then extend our analysis to polynomial functions. Finally, we discuss how the results for the polynomial setting translate to more general CNN networks.

For brevity, we ignore the indices k and i when they are apparent from the context and consider a single data sample $x \in D$. Input data can be different dimensions, e.g., 1-dimension for common speech signals and 2-dimensions for common image data). Here we reshape x into a 1-dimension (if not the original 1D) array by concatenation.

LINEAR $\nabla\mathscr{L}(\cdot)$

The widely adopted fully-connected neural networks generally contain linear convolutional layers and fully-connected layers. Generally, biases can be added after the multiplication, i.e., an additional column in the weight matrix.

Proposition 1. Let $x \in \mathbb{R}^n$, $\nabla\mathscr{L}(\omega, x) \in \mathbb{R}^p$ and $p > n$. We assume there is no zero gradient conditions. If $\nabla\mathscr{L}(\cdot)$ is linear, x can be uniquely reconstructed from the network gradient $\nabla\mathscr{L}(\omega, x)$.

Prop. 1 holds for single-server FL [39] and directly translates to the multi-server setting.

POLYNOMIAL $\nabla\mathscr{L}(\cdot)$

convolutional neural networks with hidden layers can be approximated by polynomial expressions, which will be discussed in more detail at the end of the section. Considering x is used for the training of K servers, gradients are denoted by $\nabla\mathscr{L}(\omega_1, x), \nabla\mathscr{L}(\omega_2, x), ...,$ $\nabla\mathscr{L}(\omega_K, x)$, since for same task we can simply ignore the k index for $\nabla\mathscr{L}(\cdot)$.

Proposition 2. Let $x \in \mathbb{R}^n$, $\nabla\mathscr{L}(\omega_k, x) \in \mathbb{R}^p$ and $p > n$. We assume there are no zero gradient conditions. If $\nabla\mathscr{L}(\cdot)$ is polynomial with the highest term of order q, q non-linearly dependent gradients of servers from $\mathscr{S}$ are able to reconstruct x within q^n limited range.

The result shows that we can approximate one image within tight bounds. In practice, images are usually trained in batches. Reconstructing batched data by the mentioned optimization-based inversion increases n, i.e., the approximation becomes less tight. Therefore, the above analysis also hints at why optimization-based gradient inversion is limited by the number of images (batch size) reconstructed simultaneously [153, 172, 178].

APPROXIMATING CNNS

According to approximation theory [136], a CNN consisting of input layer, hidden layer and output layer can be approximated by a polynomial of one variable. Adding a nonlinear activation function to a CNN and increasing the network depth only increases the degree of freedom of the neural network. Polynomials can still approximate such neural networks with a bounded error [23]. Moreover, in real applications, the accuracy of the approximation depends on the specific target distribution. If the target distribution is relatively simple, the neural networks degenerate into a polynomial function under regularized sparse learning [94]. Thus, even for generic neural networks, Prop. 2 is relevant

as it indicates that in many cases, we can approximate the input data well. We also empirically demonstrate the effectiveness of introducing multiple servers for reconstructing data for complex neural networks, e.g., ResNet [45].

4.5. COLLUSIVE INVERSION FOR DIFFERENT-TASK

Here we introduce the collusive gradient inversion method for dMSFL as CGI-D. Note that CGI-D can also be applied if the tasks are the same. However, it increases computational complexity considerably and is hence inefficient than CGI-S. Compared to sMSFL in Sec. 4.4, collusion on different network structures (dMSFL) brings additional challenges for the reconstruction attack. When calculating the loss based on the distances, the value is accumulated from every parameter of the network. Thus, there can be a large difference in magnitudes between different servers, i.e., the learning task of the largest net dominates the training. There may also exist conflicts of optimization direction proposed by different servers, hindering convergence. Thus, a simple averaging of gradients does not work in this case.

Consequently, as illustrated in Fig. 4.2, for CGI-D, we design a novel aggregation method for gradients g_k but keep the training, output, and other components of the reconstruction phase consistent with CGI-S. Motivated by game theory [36] we propose to model the aggregation of multiple gradients through a Nash bargaining game, i.e., a cooperative game in which the servers negotiate to reach the agreement on the direction of inversion gradients. In order to apply the underlying results from game theory and obtain a unique result, during the game, we assume that servers prefer their own gradients to dominate the results. Note that the goal of the servers remains to reconstruct images, they merely assume the selfish behavior common for games to reach an agreement that fulfills the guarantees needed.

4.5.1. NASH BARGAINING GAMES

In a Nash bargaining game, generally, there are K players who bargain over a fixed amount of resources such that the product of their respective utility is maximized. All players are selfish and have the freedom to make their own decisions independently. There are two types of outcomes for such a game, namely reaching an agreement or not. There are multiple solutions when there is an agreement but only one when there is a disagreement. The agreements are elements from a convex closed subset of $\mathbb{R}^K$, which is called the feasibility set. The disagreement is what players are guaranteed to receive if they cannot come to a mutual agreement. Prior to the starting of the game, each player knows the number of players and their utility functions, denoted as $f_k \forall k$, as well as the utility of disagreement, which is usually pre-determined by the system.

In the case of agreement, all players agree on how they split the total resources, i.e., the result is a vector specifying the proportional allocation of resources to players, $\mathbf{a} = (a_1, \dots a_k)$, where $0 \leq a_k \leq 1$, and $\sum_k a_k = 1$. Let $\mathscr{A}$ be the set of all such vectors. The utility payoff of player k is thus $u_k = f_k(\mathbf{a})$. If they fail to agree on any alternative, there is a fixed disagreement outcome, $\mathbf{s}$, with the payoff of player k being $d_k = f_k(\mathbf{s})$.

A solution to a Nash bargaining game should satisfy the following Nash axioms (details see [101]): *i)* Pareto optimality; ii) Symmetry; *iii)* Independence of irrelevant alternatives and *iv)* Invariance to affine transformation. Nash proved that a solution that

satisfies all axioms exists and is unique. It is equivalent to solving:

$$\begin{aligned} &\max_{\mathbf{a}} \prod_k (u_k - d_k). \\ &\text{s.t. } \mathbf{a} \in \mathscr{A}, \forall k : u_k > d_k \end{aligned} \quad . \tag{4.6}$$

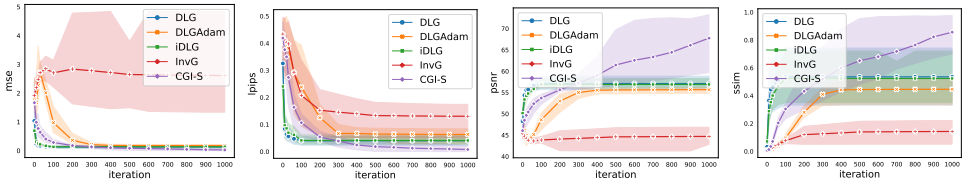
Then, the set of utilities $\{u_k^* | k = 1, ..., K\}$ is a Nash bargaining solution if it solves Eq. 4.6. In other words, the objective of this game is to find $\mathbf{a}$ to maximize the product of the payoff difference between the agreement and disagreement, $u_k - d_k$, for all players. Two constraints need to be considered simultaneously are *i)* $\mathbf{a} \in \mathscr{A}$, and *ii)* utility payoff from the agreement should be greater than disagreement, $u_k > d_k$. Optimizing Eq. 4.6 is equivalent to maximizing the sum of $log(u_k - d_k)$. Depending on the complexity of the utility function, f, one can either solve it numerically or through a closed-form solution.

4.5.2. NASH BARGAINING SOLUTION FOR CGI-D

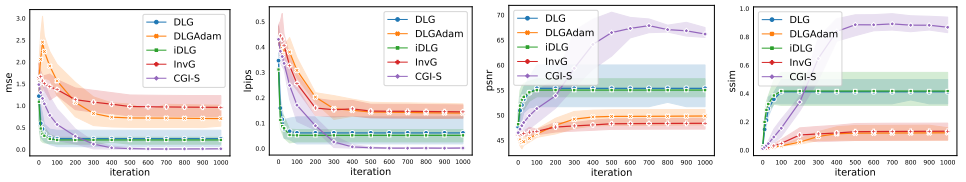
We follow the Nash bargaining game to model our collusive gradient inversion with K servers, including the utility function design, disagreement point, and fulfillment of axioms. The server k bargains for an aggregation weight α_k to influence the direction of dummy data update $\tilde{g} = \sum_k \alpha_k g_k$, which sums over the weighted gradients of all servers, $\alpha_k g_k$. The objective of the game is to reach an agreement on the weight allocation for all tasks, $\mathbf{a} = (\alpha_1, \ldots, \alpha_k)$, where $\alpha_k \in [0, 1]$ and $\sum_k \alpha_k = 1$. As every server aims at maximizing the similarity (represented by cosine similarity or scalar product) between its local gradient and the aggregated dummy gradient, we define the utility function for server k as $u_k = \tilde{g}^{\mathrm{T}} g_k \, \forall k$. Another key modeling component is the disagreement point. When servers do not reach an agreement on the aggregation weight, we let $d_k = 0 \, \forall k$, meaning that servers keep the dummy data as in the previous iteration to avoid optimizing in the wrong direction. As our global learning objectives exhibit the same form as a multi-task learning game, we follow the argument of [103] that the four axioms are satisfied and our game is equivalent to Eq. 4.6. Putting everything together, we can thus write the following optimization to determine α_k for any given iteration of CGI-D

$$\begin{aligned} &\max \prod_k \tilde{g}^{\mathrm{T}} g_k \\ &\text{s.t. } \sum_k \alpha_k g_k = \tilde{g}, \sum_k \alpha_k = 1. \end{aligned} \tag{4.7}$$

Moreover, we need to consider one additional constraint specific to the gradient inversion for images. The dummy gradient update, $\tilde{g}$, is bounded in a certain range, determined by the data source. For example, if the source data is an 8-bit binary encoded value, then the pixel-wise color range is $[0, 255]$. Alternatively, the n-channel images pixel value can also be used to bound the range of dummy gradient update. Thus, to solve this gradient inversion bargaining game of Eq. 4.7, we first take the log of the objective function and optimize the summation of the log. Afterward, we find a solution using the concave-convex procedure [103].

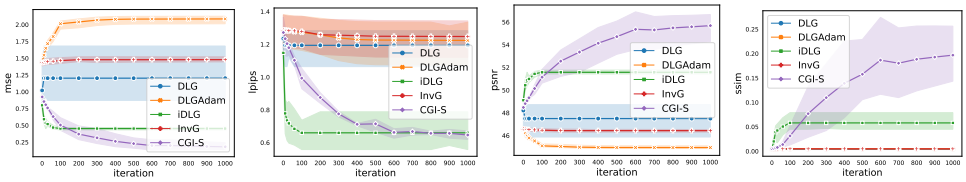


(a) *Cifar100-Vgg11*-MSE ↓ (b) *Cifar100-Vgg11*-LPIPS ↓ (c) *Cifar100-Vgg11*-PSNR ↑ (d) *Cifar100-Vgg11*-SSIM ↑



(e) (f) (g) (h)

Cifar100-ResNet20-MSE ↓ *Cifar100-ResNet20*-LPIPS ↓ *Cifar100-ResNet20*-PSNR ↑ *Cifar100-ResNet20*-SSIM ↑



(i) *STL10-ResNet34*-MSE ↓ (j) *STL10-ResNet34*-LPIPS ↓ (k) *STL10-ResNet34*-PSNR ↑ (l) *STL10-ResNet34*-SSIM ↑

Figure 4.3: Comparing the reconstruction results for 2 servers same-task with single-server baselines; metrics are MSE, SSIM, PSNR and LPIPS on *Cifar-100 (LeNet, ResNet-20)* and *STL-10 (ResNet-34)* datasets. Averaged over 3 runs.

4.6. EMPIRICAL EVALUATION

In this section, we first introduce our evaluation settings in detail. Then, we quantify the advantage that attackers have when data is re-used for both sMSFL and dMSFL, even in the presence of a defense mechanism.

4.6.1. EXPERIMENTAL SETUPS

GENERAL SETTINGS

We evaluate our CGI-S and CGI-D on *Cifar-100 [74], lfw [53], STL-10 [19]* and *Mnist [21]* datasets with LeNet [77], ResNet [45] and Vgg [122] neural networks. To demonstrate the quality of image data reconstruction, **MSE [8], PSNR [50], SSIM [143]** and **LPIPS [171]** are applied.

BASELINES

As this chapter aims to reveal the increased privacy risk for a client of contributing local data to multiple tasks, it is necessary that we keep the other variables consistent. We compare our proposed CGI with optimization-based inversion methods without external knowledge, concretely: DLG [178], iDLG [172], DLG-Adam [153], and InvG [39].

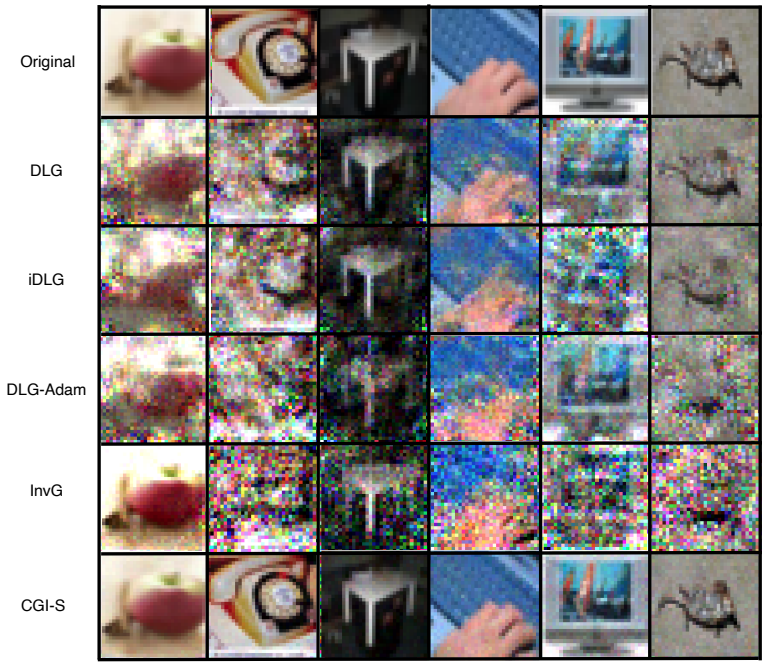


Figure 4.4: Reconstructed *CIFAR-100* images from ResNet-20 with 1000 iterations for five baselines and CGI-S.

TASKS

For most experiments, we utilize 2 servers as the minimum collusive version. For sMSFL, we follow the same setting for every server as the original single-server classification on each dataset. For dMSFL, we mainly perform 2 different tasks on the same dataset. For *Cifar-100*, one server is conducting a 100-class classification task while the other server is for 20-class, based on the two benchmark labels. Similarly, for *lfw* and *STL-10*, they identify 5749 different people/binary gender classification, and original 10-class/2-class animal or transportation respectively. Using the above settings, we validate the effectiveness of collusive inversion based on our Nash bargaining game. We vary the number of servers from 2 to 5 with increasingly fine-grained label distinctions while keeping the optimizer, distance metric, learning rate and other settings consistent with single-server settings.

DEFENSES

Following the related works [31, 90, 61], we select four representative defenses with the following settings. *i) Noising:* Gaussian Noise with standard deviation 0.1; *ii) Clipping:* Gradient clipping with a clip bound of 4; *iii) Sparsification:* Gradient sparsification in a sparsity of 90; and *iv) Soteria:* Soteria with a pruning rate of 80%.

OTHER SETTINGS

All the baseline methods are for a single server with a single task. Thus, we compare the baselines with CGI-S for sMSFL. For Vgg-11 and ResNet, we train 1000 iterations for

inversion while 500 for LeNet. We evaluate the batch size as 2^k, where $k \in \{1, 2, ..., 6\}$, and the detailed comparison for batched data is in Sec. 4.6.2. Our reconstruction of batched data focuses on the data leakage risk and does not consider the alignment of batched data order, which is practical and is consistent with related studies [153, 66, 83, 31].

4.6.2. SAME TASK MSFL (SMSFL)

COMPARISON WITH BASELINES

We start by discussing the results for 2 servers under sMSFL, then show that additional servers increase the risk of a successful reconstruction attack. Fig 4.3 shows the inversion progress over iterations using the 4 evaluation metrics: MSE, LPIPS, PSNR and SSIM on datasets *Cifar-100* and *STL-10*. For *Cifar-100*, we use VGG-11, ResNet-20, and ResNet-34 to demonstrate the impact of the network size. We also visualize the reconstructed images from *Cifar-100* as examples in Fig. 4.4.

Fig. 4.3 demonstrates that our proposed CGI-S outperform baseline methods overall, exhibiting higher PSNR/SSIM and lower MSE/LPIPS. Notably, our CGI-S shows clear advantages in terms of PSNR and SSIM, with less distinctly differences in terms of MSE and LPIPS. This difference arises from the fact that PSNR, SSIM, and LPIPS focus on image quality assessment, while MSE treats all pixel values equally, potentially leading to differing perceptual outcomes. LPIPS, specifically, adapts its evaluation based on learned features and can be dataset-sensitive, showing a smaller range of values in our study (e.g., 0–0.5 for ResNet on *Cifar-10*). Moreover, our CGI-S yield significantly improved results with more complex networks like ResNet-20 on *Cifar-100* and ResNet-34 on *STL-10*. However, the performance gap narrows with smaller networks such as Vgg-11 on *Cifar-100*. Additionally, experiments with LeNet on *Cifar* show minimal differences in final similarities between CGI-S and single-server attacks, but CGI-S demonstrates higher efficiency. This suggests that for simpler tasks where baselines achieve successful reconstruction, CGI-S benefits from enhanced efficiency, guided by bounded reconstruction probabilities in Sec. 4.4.

In terms of the dataset, the results are very consistent in the sense that the ranking the attacks in terms of achieved similarity usually results in the same ranking for all datasets. Concretely, CGI-S is mostly the best-performing attack, followed by iDLG. Convergence-wise, DLG and iDLG sometimes may converge faster than CGI-S but they do not achieve the same reconstruction quality. The faster convergence is due to the use of the L-BFGS optimizer by DLG and iDLG, which is known for fast convergence. DLGAdam and InvG are consistently of a lower performance than CGI-S, regardless of the number of iterations. By visualization in Fig. 4.4, we show the perceptual similarity between the reconstructed data with the original data, which is consistent with Fig. 4.3, demonstrating the high privacy risk of data reuse, even if there are only 2 servers.

Additionally, we also evaluate the performance of reconstructing at different phases of the training, i.e., different round numbers r. The conclusion is consistent with the single-server case [153]: It is easier to launch the attack in the early training phase where gradients are not affected by the previous global models and hence updates from other users.

According to Sec. 4.4, the reconstruction performance also depends on the original size of image and network size. Comparing 32×32 images taken from *Cifar-100* with

96 × 96 sampled from *STL-10*, in Fig. 4.4 and Fig. 4.5, we find that single-server inversion methods fail on larger images with complex networks such as ResNet. However, with the same optimizer and distance function, collusive servers are able to reconstruct high-resolution images without extra external knowledge. As stated earlier, using multiple gradients reduces the searching space for optimization, which has more impact when the search space is big due to complex models or large images size. Another observation worth mentioning is that even with collusive servers, larger data input is difficult to be reconstructed for small networks, where the gradient information is limited, as indicated by the notable better image quality for CGI-S with ResNet in comparison to CGI-S with LeNet in Fig. 4.5 (e) and (f).

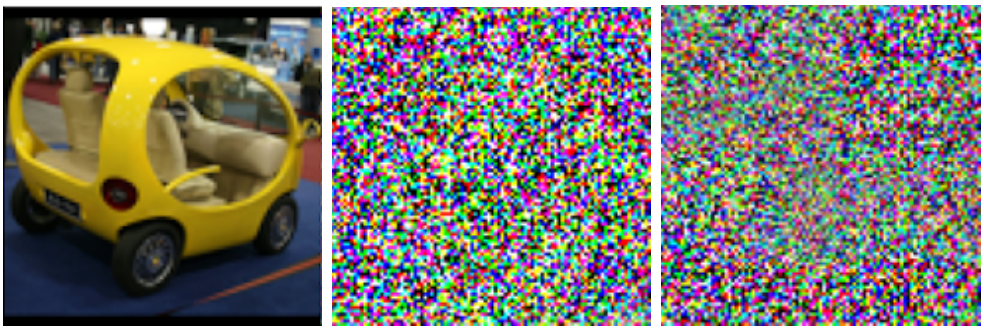
Table 4.1: Results for inversion from dMSFL on *Cifar-100*, *lfw* and *STL* dataset with LeNet and ResNet averaged over 5 runs. We compare our CGI-D with weights assigned to a "Single" server and "Random" weights assigned for each task.

		MSE ↓			LPIPS ↓			PSNR ↑			SSIM ↑		
Dataset	**Iter**	Single	Random	CGI-D	Single	Random	CGI-D	Single	Random	CGI-D	Single	Random	CGI-D
Cifar-100	25	0.428	0.190	**0.004**	0.154	0.105	**4.3e-3**	51.819	55.336	**72.547**	0.249	0.413	**0.969**
	50	0.182	0.526	**1.4e-3**	0.067	0.035	**9.9e-6**	55.532	60.924	**86.619**	0.463	0.695	**0.999**
LeNet	75	0.084	0.020	**9.6e-6**	0.040	0.009	**3.0e-7**	58.890	65.204	**98.290**	0.624	0.842	**0.999**
	100	0.040	0.009	**6.3e-6**	0.018	0.003	**1.7e-7**	62.071	68.579	**100.13**	0.755	0.932	**0.999**
lfw	25	0.043	1.307	**8.7e-5**	0.048	0.278	**3.5e-5**	61.813	46.968	**88.727**	0.550	0.067	**0.969**
	50	0.002	0.018	**3.3e-8**	0.001	0.009	**9.9e-6**	75.806	65.668	**122.894**	0.965	0.855	**0.999**
LeNet	75	8.1e-5	6.0e-6	**3.0e-8**	2.7e-5	1.2e-6	**7.7e-9**	89.060	100.381	**123.316**	0.998	**0.999**	**0.999**
	100	4.4e-6	1.1e-7	**3.0e-8**	1.3e-6	1.7e-6	**7.7e-7**	101.742	117.687	**123.316**	**0.999**	**0.999**	**0.999**
STL-10	300	0.242	0.223	**0.184**	0.048	**0.039**	0.047	54.300	54.638	**55.481**	0.214	0.265	**0.301**
	600	0.201	0.200	**0.119**	0.039	0.044	**0.033**	55.109	55.124	**57.386**	0.314	0.299	**0.415**
ResNet20	900	0.100	0.150	**0.068**	0.034	0.034	**0.024**	58.127	56.382	**59.802**	0.515	0.375	**0.586**
	1800	0.067	0.109	**0.005**	0.015	0.036	**0.001**	59.890	57.738	**70.921**	0.614	0.491	**0.927**

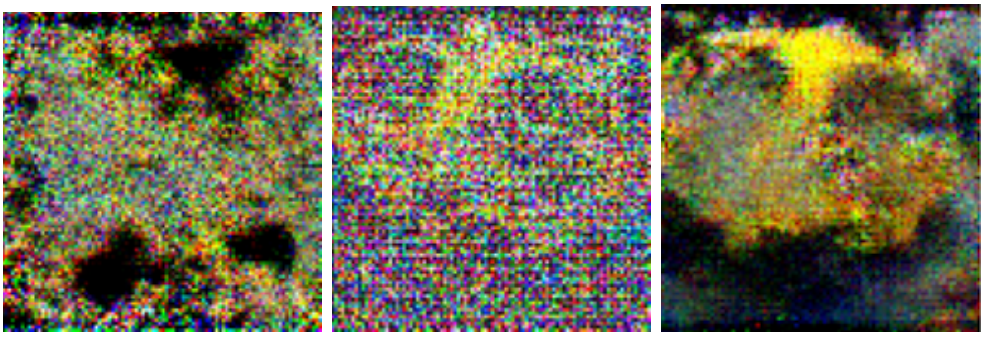
DIFFERENT BATCH SIZES

A common issue for optimization-based inversion is that the algorithm is too slow to converge with low quality reconstructed data for large batch size B as batched data can have $B!$ different permutations [178]. Here, we use *Mnist* dataset and LeNet with batch sizes of $2^k, k \in \{1,2,...,6\}$, where the largest successful reconstruction by optimization without external knowledge is 48 in related studies. We empirically show that multiple servers allow for higher batch sizes for the same reason as why they do well with complex networks and large images: they narrow down the search space of dummy data optimization through intersecting multiple search spaces. The result is in line with the theoretical results of Sec. 4.4 that the increased number of servers entails that (approximate) reconstruction is possible in more complex settings. We utilize DLG as the single-server baseline, since for iDLG, the predicted single label does not work on batched data, where one batch can contain different labels. The results are displayed in Fig. 4.6, showing the final image of 300 iterations for batch size 4 and 8 and 1000 iterations for batch size 64. Single-server attacks still work for up to a batch size of 4 but start failing at a batch size of 8. CGI-S results in relatively similar images even for a batch size of 64, which is as similar quality visually with the related study on large batch sizes, e.g., 48 [39].

IMAGE SIZE V.S. NETWORK SIZE



(a) original (b) DLG ResNet (c) DLG-Adam ResNet



(d) iDLG ResNet (e) CGI-S LeNet (f) CGI-S ResNet

Figure 4.5: Reconstruction visulization on high-resolution image *STL-10* (size 96 × 96) with different network sizes, comparing CGI-S and baselines.

According to Sec. 4.4, the reconstruction performance also depends on the original size of image and network size. The results of Fig. 4.3 only covers successful reconstruction with varying network on 32 × 32 *Cifar-100*. Here we apply high resolutional, i.e., 96 × 96, input data sample from *STL-10*, to compare the reconstruction results between single/multi-server and shallow/deep network. The original image and the reconstructed results are shown in Fig. 4.5. From Fig. 4.5, we can see that although single-server inversion methods work on some low resolutional data input, e.g., 32 × 32 *Cifar-100* image in Fig. 4.4, it fails on the more complex deep neural network of ResNet. However, with the same optimizer and distance function, collusive servers are able to reconstruct high resolutional image without extra external knowledge. As stated earlier, using multiple gradients reduces the searching space for optimization, which has more impact when the search space is big due to complex models or large images size. Another observation worthy mention is that even with collusive servers, larger data input is difficult to be reconstructed for small networks, where the gradient information is limited, as indicated by the notable better image quality for CGI-S with ResNet in comparison to CGI-S with LeNet in Fig. 4.5 (e) and (f).

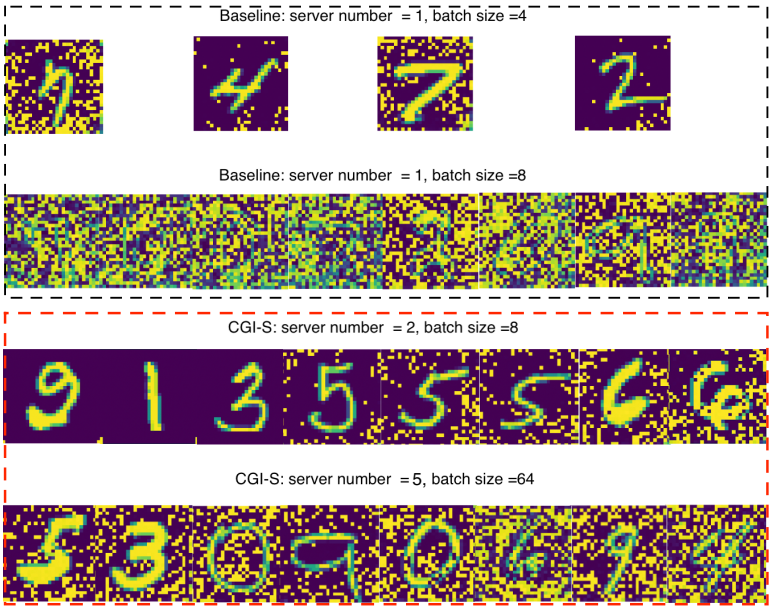


Figure 4.6: Performance for different batch sizes, up to 64, on Mnist.

PERFORMANCE IN PRESENCE OF DEFENSE

Table 4.2: PSNR ↑ results under defense mechanisms for sMSFL on Cifar-100 with ResNet-20 and 1000 iterations.

	Defense strategy			
Method	Noising [1]	Clipping [1]	Sparsification [3]	Soteria [130]
DLG [178]	51.485	50.579	50.743	50.911
iDLG [172]	50.978	52.003	50.513	49.894
DLG-Adam [153]	50.094	48.634	47.993	49.141
InvG [39]	42.531	47.595	47.619	48.086
CGI-S	**63.086**	**66.176**	**66.396**	**67.438**

We furthermore evaluate the effectiveness of reconstructing client data under different defense strategies. The PSNR comparison results on *Cifar-100* dataset are shown in Tab. 4.2. Note that PSNR without defense is shown in Fig. 4.3g for the 500^{th} iteration. According to PSNR, we see that each defense decreases the reconstructing quality in general. Among the baselines, CGI-S is able to reconstruct images with higher PSNR for all four defenses compared to single-server attacks, demonstrating its effectiveness in the presence of defenses.

4.6.3. DIFFERENT-TASK MSFL (DMSFL)

COMPARISON WITH BASELINES

As we are the first to evaluate the attack in the multi-server setting, there is no existing baseline for different tasks. We hence compare our CGI-D with 2 settings: *i) Single* server, where only one randomly chosen task gradient is applied to update the dummy data and ignore the gradients from other tasks; *ii) Random* weights, where each iteration a randomly chosen weight vector is applied for gradient aggregation of servers. The results of MSE, LPIPS, PSNR and SSIM averaged with 5 replications are given in Tab. 4.1. We train each inversion for multiple iterations determined by the network size and report the results at iterations 25, 50, 75, 100 for LeNet and iterations 200, 400, 600, and 800 for ResNet-34. For more iterations, MSE and LPIPS decrease while PSNR and SSIM increases, showing better reconstruction performances. According to the results, it is evident that *Random* outperforms *Single* in general, which means that even without sophisticated weight assignment, multiple servers lead to lower privacy. Yet, both *Single* and *Random* fluctuate and do not converge monotonously. This is in line with our motivation to design gradient aggregation based on Nash bargaining game, namely that randomly or averagely integrating different tasks may suffers from gradient conflicts, which affect convergence. As a comparison, our CGI-D utilizes Nash bargaining game to coordinate the aggregation, so that each server negotiates on the updating direction, resulting in the clearly better results than those achieved by the baselines.

IMPACT OF DUMMY INITIALIZATION

Inverting gradients starts from the initialized dummy data. Hence, we also consider how the initialization method affects our results. Four initialization methods are considered: *i)* standard norm distribution, where each pixel is generated as $\tilde{D}_0 \sim N(0,1)$; *ii)* the initialized images follows uniform distribution on the interval $[0,1)$; *iii)* random integers in the range of pixel color encoding, e.g., $[0,255]$ for 8-bit; and *iv)* real images chosen from the same dataset but not the image we aim to reconstruct. Fig. 4.7 displays the results for CGI-D on *lfw* dataset on LeNet. Random integers do not lead to good results as the training procedure takes float/double values. Using a real image from a similar data distributions means that the optimization may get stuck in local optimum and hence also does not lead to good results in terms of reconstruction. Float type initialization from $[0,1)$ based on either standard norm or uniform distribution differ only slightly, shown in the first and second row of Fig. 4.7 on reconstruction. A similar conclusion with regard to dummy initialization holds for CGI-S on sMSFL.

IMPACT OF THE NUMBER OF DIFFERENT TASKS

We now increase the number of servers. The *Cifar-100* dataset is chosen since the 100 classes can be grouped such that we have 2-class, 10-class, 5-class and 20-class labels, resulting in total of 5 classification tasks when including the original 100 classes. From Fig. 4.8, we can see that generally, the effectiveness of the reconstruction, measured in terms of PSNR, increases with the number of servers. However, there is hardly any difference between 4 servers and 5 servers, indicating diminishing returns for a higher number of servers.

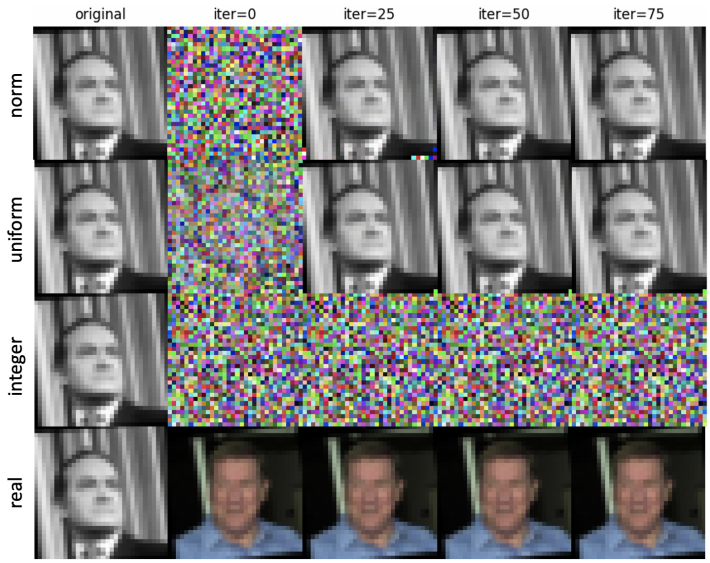


Figure 4.7: Impact of different initialization methods for CGI-D on *lfw* human face dataset using LeNet: random noise of standard norm / uniform distribution, random integers encoded as 8-bit and real image from the same dataset.

Table 4.3: PSNR ↑ results under defense mechanisms for dMSFL on Cifar-100 with ResNet-20.

	Defense strategy			
Method	Noising [1]	Clipping [1]	Sparsification [3]	Soteria [130]
Single	50.648	52.337	50.694	51.460
Random	49.947	51.131	50.063	50.842
CGI-D	**51.184**	**64.366**	**63.047**	**66.576**

PERFORMANCE IN PRESENCE OF DEFENSE

We apply the same four defenses as for sMSFL. The results, displayed in Tab. 4.3, confirm that CGI-D outperforms the baselines in the presence of defenses. Note that the advantage is considerable for all defenses but Noising, where CGI-D is only slightly better than the baselines, indicating that the defense is quite effective. Compared to CGI-S (Tab. 4.2), we find that the results for CGI-D are a bit worse, so it is harder to evade defenses when the tasks are different and aggregation is hence more challenging.

4.7. CONCLUSION

We are the first to analyze the risk of data reconstruction when a client repeatedly uses their data to contribute to multiple tasks. Both analytical and empirical results indicate that our attacks, CGI-S and CGI-D, are considerably more impactful than attacks in the single-server setting, even in the presence of defense mechanisms. We demonstrate the

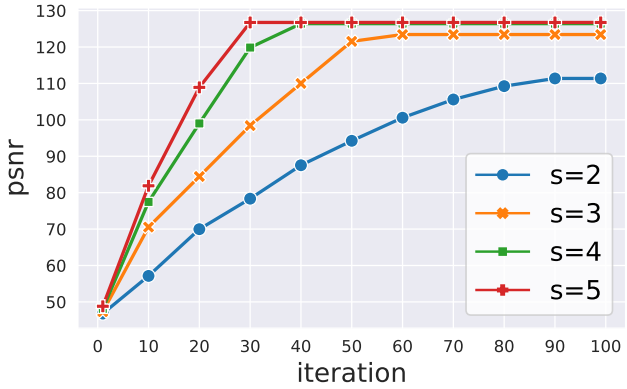


Figure 4.8: Impact of the number of servers for dMSFL on PSNR on *Cifar-100* with LeNet.

superiority of our attack through better performance on large image sizes, large batch sizes, and more complex deep neural networks.

5

Gradient Inversion Attack in Federated Diffusion Models

Diffusion models are becoming the most prevalent generative models, producing exceptionally high-quality image data through a stochastic process of diffusion steps based on Gaussian noise. Recent studies explore the federated training of diffusion models, enabling the collaborative training of a model without clients sharing raw data. We demonstrate that even without direct sharing of the data, the shared gradients of federated diffusion models already leak sensitive information about the raw data.

In this chapter, we first argue that diffusion models can easily be trained in a federated manner, without the raw data leaving premises. Our main contribution then lies in evaluating the privacy leakage of the shared model updates by developing gradient inversion attacks to reconstruct training data. We design the first gradient inversion attack GIDM for diffusion, which can reconstruct the training data from the shared model updates. GIDM is a two-phase fusion attack that is both efficient and effective. In its first phase, GIDM leverages the trained diffusion model itself as prior knowledge to constrain the inversion search (latent) space, followed by a second phase of pixel-wise fine-tuning. Different from existing inversion attacks on the classification models, inverting diffusion models presents new challenges, most notably that the noise term and randomly sampled diffusion step are not known to the attacker but are required for the reconstruction. To tackle this challenge, we propose a joint triple-optimization algorithm to approximate the raw data, sampling step, and noise term simultaneously. GIDM is shown to be able to reconstruct images almost identical to the original ones and clearly outperforms baselines, i.e., GIDM without the second phase and state-of-the-art attacks on classifiers adapted to diffusion.

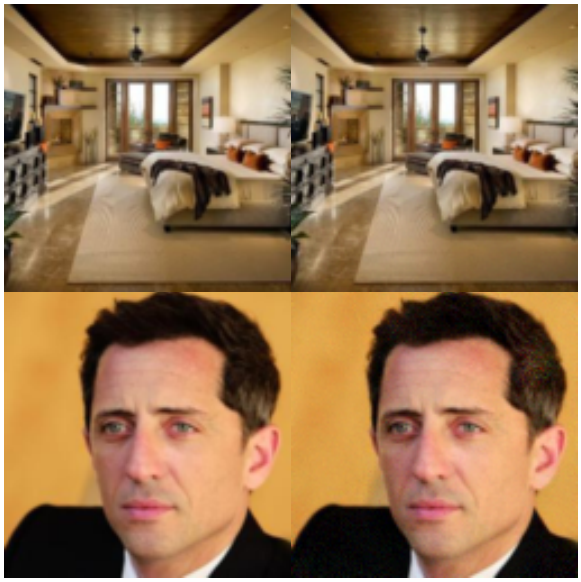


Figure 5.1: Original training image (left) v.s. Recovered images (right) by our proposed GIDM on diffusion models: reconstructing almost identical to the original image of 128 × 128 size.

5.1. INTRODUCTION

The emergence of likelihood-based diffusion models empowers probabilistic models to generate high-quality data, such as image and video data [124, 49, 113]. Diffusion models are trained by finding the reverse Markov transitions that maximize the likelihood of the training data. In practice, the training is done by gradually adding noise to and denoising the images over multiple steps.

However, training high-quality diffusion models usually requires a large amount of data. Practically, such data may be owned by different parties. Following data privacy regularization such as HIPAA [2] and GDPR [138], data owners are not allowed to share such data if it contains personally identifiable information, which includes, in particular, sensitive data such as medical records. As a consequence, prior works argue that diffusion models need to be trained in a distributed manner without sharing raw data, e.g., in a federated learning manner [117, 78]. Indeed, the design of the Denoising Diffusion Probabilistic Model (DDPM) [49], the most common architecture for training diffusion models, enables distribution trivially: in each round, the algorithm independently samples a random Gaussian noise $\boldsymbol{\epsilon}$ for data sample(s) $\boldsymbol{x}_0$ at each sampling step t. Thus, there exist multiple designs for federated diffusion, which — while differing in the details of their implementation — all share the following approach: at each global round of training, multiple distributed data owners (clients) train the diffusion sub-models based on their local data. Then, the central server, which connects to every client, aggregates their gradients before returning the aggregated gradient to the clients for further training until convergence is reached.

Such a federated diffusion method successfully avoids direct data sharing and enables the server to obtain the intermediate training gradients and the diffusion model in the end. In this paper, we consider the privacy leakage caused by the shared gradients. While there have been privacy attacks on diffusion models, they do not consider gradients. For example, relying on inferring information from the trained model, e.g., error comparison of the forward process posterior estimation [26, 117], adversaries are able to launch membership inference attacks [52, 26, 149] or attribute inference attacks.

Although studies on federated training classifiers demonstrate that the server is able to invert the client's raw data from their gradients [178, 153, 39, 172], we argue that existing attacks cannot be easily transferred to diffusion models. The key idea of previous attacks is to reconstruct images from dummy data by using the gradients to constrain the search space. In the context of classification, label information needs to be present or reconstructed as part of the attack. In contrast, diffusion does not have labels; rather, it requires the sampling step t during the training process and the noise prediction $\boldsymbol{\epsilon}$ in each step, so that a novel approach is needed.

In this paper, we systematically study the data reconstruction risk when training federated diffusion models. We first define the common factors of federated training methods for diffusion models, according to related studies. Afterwards, we design our attack, which relies on diffusion-specific information that other scenarios lack. Specifically, at the end of the training process, the server owns a trained diffusion model, which we can use to constrain the search space for the reconstructed images. Using generative models has been explored in prior studies on inversion attacks on classification [66, 31] but these studies rely on external models.

Concretely, to reconstruct images that resemble the training data, we thus design a fusion optimization, GIDM, that includes two phases. The generating phase maps the dummy data into a narrow latent space to optimize in-distribution images by utilizing the diffusion model as prior knowledge. Then the fine-tuning phase further optimizes the similarity between the dummy and real gradients to update the dummy data generated during the first phase of the attack.

Approximating the gradient of diffusion models also requires the knowledge of $\boldsymbol{\epsilon}$ and t, which may only be known to the clients, a key challenge of inversion attacks on diffusion models. To solve this, the two phases of GIDM include a novel triple-optimization for dummy data, $\boldsymbol{\epsilon}$, and t. Specifically, the triple-optimization includes three independent optimizers for the dummy data, the noise, and the sampling step to refine the joint training. By coordinating the three optimizers with updating intervals, we are able to recover images without knowing the exact values of $\boldsymbol{\epsilon}$ and t.

Our proposed GIDM is able to recover images almost identical to the original data up to size 128×128, as shown in Fig. 5.1. Our evaluation further shows that the fine-tuning phase is indeed required and GIDM cannot reliable reconstruct images without using the triple-optimization. For comparison, we adapt two attacks for image classifiers, DLG [178] and InvG [39], to the scenario of diffusion. Note that as these attacks do not consider $\boldsymbol{\epsilon}$ and t, we weaken the adversarial model for them by providing the concrete parameter values. Despite these attacks having more information, which would not be available in a real-world setting, GIDM outperforms them in terms of four key image similarity metrics.

We summarize our main contributions as follows. We present the first gradient inversion attack GIDM on diffusion models, leveraging three key novel ideas: the use of the trained diffusion model to constrain the search space; a two-phase attack consisting of a phase based on the existing diffusion model, and a fine-tuning phase; a triple-optimization method that jointly reconstructs the data, the sampling step, and the noise parameter. Our evaluation shows that GIDM can successfully reconstruct images of a large size and highlights the impact of the different attack components through ablation studies.

5.2. RELATED WORK

Diffusion Models and Privacy. Diffusion models employ a two-step process: First, they deconstruct the training data structure step by step in a forward manner. Second, they master the reconstruction of the structure from noise in a reverse process. The Denoising Diffusion Probabilistic Model (DDPM) [49] introduces a stable and efficient implementation of diffusion for high-quality image synthesis. DDPM relies on a forward process without learnable parameters while employing simplified Gaussian noise in the reverse phase. Further variants of diffusion models such as DDIM [124], Stable Diffusion [113], and Imagen [115] improve the sampling efficiency or involve deep language understanding for text-to-image generation. However, well-trained diffusion models have been shown to be vulnerable to privacy attacks, i.e., information leakage on the training data. Recent studies on privacy concerns of diffusion models mainly focus on membership inference attacks [26, 52, 149] or training data memorizing attacks [123, 12]; both of which are executed on the trained model. None of the studies has addressed the data reconstruction from the gradients of diffusion models.

Privacy Defenses of Federated Learning. In order to enhance system privacy against privacy leakage attacks and not significantly reduce model accuracy [156, 3, 47, 130], current defenses are primarily conducted individually by clients, falling into two categories: gradient perturbation and input perturbation. Gradient perturbation [156, 131, 27, 1, 3, 47, 126, 62], preferred for its efficiency and maintaining global model accuracy, involves transmitting perturbed gradients. In contrast, input perturbation, such as mixing images before local training, is less common [166]. Prominent gradient perturbation defenses include differential private stochastic gradient descent [27], which adds noise and clips gradients to limit sensitivity [1]. Gradient sparsification accelerates training by setting small gradient entries to zero [3], differing from dropout by removing small entries rather than randomly selecting them [62, 47, 126]. Soteria proposes a fully-connected defense layer to perturb data representation, crucial for preventing inversion attacks while preserving Federated Learning performance [130]. Overall, defense efficacy depends on parameters like noise level, clipping bound, sparsity, and pruning rate, which need to be carefully chosen to balance model quality and privacy.

Gradient Inversion. As the first practical gradient inversion attack for classifiers, Deep Leakage from Gradients (DLG) reconstructs data and label simultaneously by directly approximating gradients from the dummy data input [178]. DLG tends to reconstruct images of low quality and cannot deal with large training batches. To strengthen DLG, one line of work improves DLG by developing different optimizers [153, 39], distance metrics [39], or integrating direct features [172], e.g., they first infer labels before recon-

structing. The other line of work focuses on leveraging external knowledge for inversion. Such knowledge includes prior data distributions for more accurate embeddings [42], adding batch normalization regularizers to manage larger batches [159], applying pre-trained generator models to ensure high-quality reconstructed data [66, 31, 83] and utilizing auxiliary datasets [147]. GGDM iteratively refines the noise via a pre-trained unconditional DDPM as a guidance, but also targeting invert classifiers. These advanced attacks [31, 66, 159, 67] successfully integrate additional information to improve the attack effectiveness. However, such knowledge can also be further integrated to our proposed GIDM. Thus, our direct comparison baselines are DLG and InvG.

Apart from classification models, inversion can be applied to Generative Adversarial Networks (GANs), where the attacks aim to invert a generated image back into the latent space of a pre-trained GAN model [150], i.e., reconstructing the latent code instead of the training images. Currently, there exist no inversion attack studied for inverting diffusion models.

5.3. METHODOLOGY

In this section, we first introduce preliminaries on federated diffusion models, highlighting the common components of existing frameworks. Then, we propose our inversion attack GIDM, which leverages the trained diffusion model as prior knowledge for constrained optimization, consisting of two phases. The generative phase improves the quality of a dummy image to achieve fast convergence, followed by a fine-tuning phase to increase the pixel-wise similarity. For both phases, we argue that the prior art of gradient inversion attacks does not apply to diffusion models due to the inability to handle more unknown factors, namely the noise ϵ and sampling step t, according to the DDPM training strategy. Thus, we design a triple-optimization algorithm to infer the original image, ϵ, and t simultaneously.

5.3.1. FEDERATED DIFFUSION MODEL PRELIMINARIES

We consider a federated image generation task following the standard DDPM diffusion models [49], which aims to optimize the weighted variational bound:

$$L(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right], \tag{5.1}$$

where $\mathbf{x}_0 \in \mathbb{R}^m$ are training samples of dimension m = width × height × color, $L(\cdot)$ is the point-wise loss function, θ denotes the diffusion model network parameters and $\bar{\alpha}_t$ is a hyper-parameter controlling the forward noising process. Note that the sampling step t is chosen uniformly between 1 and T and we follow the definition [49] that $\boldsymbol{\epsilon}_\theta$ is a function approximator intended to predict the Gaussian noise $\boldsymbol{\epsilon}$ added to the image $\mathbf{x}_t$ of step t.

There are K clients serving as data owners who are responsible for diffusion model training. The k^{th} federated learning client owns the local real dataset $X_k, k \in \{1, 2, ..., K\}$, which is not shared with others. Each client reports the gradient:

$$\nabla_\theta \left\|\boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{\theta_k}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_k, t\right)\right\|^2,$$

Algorithm 3: Federated Diffusion Model Training

Input: The number of clients K, number of global training round R, local datasets $X_k, k \in [1,K]$, diffusion steps T, global learning rate η.
Initialize model θ
for $r = 1,2,...,R$ **do**
  for $k = 1,2,...,K$ **do**
    $t \sim \text{Uniform}(\{1,...,T\})$
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    $\mathbf{x}_0 \sim X_k$
    $g_k = \nabla_{\theta_k} \left\| \boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{\theta_k} \left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_k, t \right) \right\|^2$
  end
  $g = 1/K \sum_{k=1}^{K} g_k$
  Update $\theta = \theta - \eta g$
end
Return the trained θ^* (θ from the last round)
Result: the trained θ^*

for the locally sampled data $\mathbf{x}_0$. We use R global training rounds. The single server $\mathcal{S}$, which does not own any data itself, aggregates, usually by computing their average, and distributes the aggregated model in each global training round.

In contrast to other federated learning models, e.g., classification tasks, diffusion models require sampling the parameters $\boldsymbol{\epsilon}$ and t. We assume that they are sampled by the clients who own the data during the training process. Consequentially, the server does not know which parameters were chosen by each client. In the evaluation, we compare this setting to a less privacy-preserving variant where the server chooses $\boldsymbol{\epsilon}$ and t and distributes them to the clients in Sec. 5.4.4 and Sec. 5.4.5. Both settings can be implemented as equivalent optimization problems to centralized diffusion models. The general training process is given in Alg. 3.

5.3.2. THREAT MODEL

Our threat model considers the federated server $\mathcal{S}$ as the adversary to reconstruct the input training data X_k of the target client C_k. The threat model is as follows.

Objective. The adversarial server aims to recover the input data X_k trained by client C_k based on the gradient:

$$g_k = \nabla_\theta \left\| \boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{\theta_k} \left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_k, t \right) \right\|^2,$$

during a specific global training round r. As the inversion can be executed at one given global round, we drop the index r for simplicity. The attack is successful if the recovered image $\hat{X}$ is almost identical to X_k.

Capability. We assume that the honest-but-curious server does not have access to the real data of data owners. Moreover, the servers' computational resources are limited, so it cannot, e.g., break cryptographic primitives.

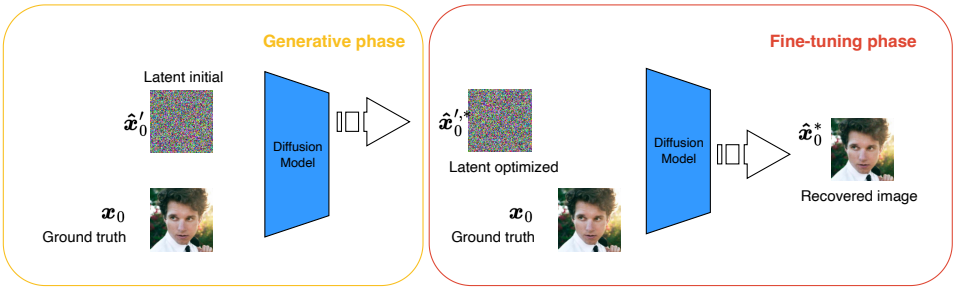


Figure 5.2: The workflow of two phases of GIDM: Generative phase to optimize the latent code, while the fine-tuning phase continues to improve the pixel-wise similarity between the recovered images and the original ones to achieve both efficiency and effectiveness.

Knowledge. To recover input data at a specific round, we assume $\mathscr{S}$ naturally owns the global model and can access the model parameters and the submitted gradient of all clients. However, the initialization of ϵ and the sampling step of t are unknown to the adversary unless stated otherwise.

5

5.3.3. TWO-PHASE INVERSION WITH DIFFUSION PRIOR

To assess the privacy leakage of federated diffusion, we propose the first gradient inversion attack GIDM: an honest-but-curious server reconstructs a victim client's data using the gradients submitted by the client. We model the inversion process as an optimization problem that iteratively modifies the dummy data by minimizing the distance between known and approximated gradients, as proposed by the inversion attacks for federated classifier training [178]. Following Alg. 3, when client k computes the gradient for the training data $\mathbf{x}_0$, the gradient is:

$$g_k = \nabla_{\theta_k} \left\| \boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{\theta_k} \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_k, t \right) \right\|^2.$$

Note that since a gradient inversion attack can be launched at any specific global training round for any client, we drop the round and client indexes r and k in the following. Assuming that θ is second-order differentiable, we suppose that the dummy data $\hat{\mathbf{x}}_0$ is an approximation of $\mathbf{x}_0$ if $\hat{g} \sim g$, where $\hat{g}$ is the dummy gradient calculated based on $\hat{\mathbf{x}}_0$. Thus, our gradient inversion objective turns to:

$$\min_{\hat{x}_0} Dist(\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t} \hat{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2, g), \tag{5.2}$$

where $Dist(\cdot)$ is a distance metric for two gradients.

Without constraining the search space, the difficulty of inversion increases exponentially with the image size [178]. We leverage the final trained diffusion model, which the server naturally possesses as it is the result of the training process, to constrain the search space. In contrast to prior work, no external pre-trained model is required. Concretely, we propose a constrained fusion model consisting of two phases: the generative phase and the fine-tuning phase. First, the generative phase leverages prior knowledge of the

trained final diffusion model, $\mathscr{D}_{\theta^*}: \mathbb{R}^m \to \mathbb{R}^m$, where θ^* is the diffusion model parameter. Then, the fine-tuning phase improves the dummy image from the generative phase by efficiently minimizing the pixel-wise similarity. The overall workflow is summarized in Fig. 5.2, where the first generative phase outputs the optimized latent code as the input to the fine-tuning phase. The fine-tuning phase then output the final recovered images.

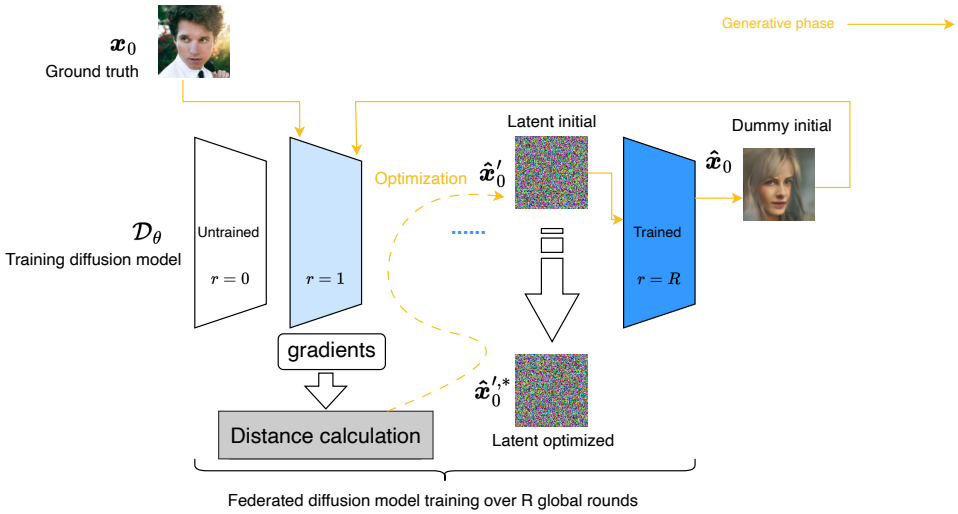


Figure 5.3: We utilize one 128 × 128 image from *CelebA* as an example to show every intermediate result above. The generative phase executes 5000 iterations in this example, finally outputting the high-quality latent code which is able to sample resembled images from the trained diffusion models.

Generative phase. We aim to generate images that resemble the training dataset as the starting point for dummy data optimization by gradient approximation. Our approach is to map the original search space into a narrow latent space with constraints (based on prior knowledge). The details of the generative phase is presented in Fig. 5.3. Let the latent code $\hat{\mathbf{x}}'_0$ [142] be of the same dimension as the dummy data $\hat{\mathbf{x}}_0$ from Eq. 5.2 and $\hat{\mathbf{x}}_0 = \mathscr{D}_\theta(\hat{\mathbf{x}}'_0, t)$. We execute multiple iterations to optimize $\hat{\mathbf{x}}'_0$, as described below, so that the gradient computed on $\hat{\mathbf{x}}_0$ has a small distance to the real gradient. Thus, instead of directly updating the dummy data, we perform a latent space search over $\hat{\boldsymbol{x}}'_0$, which is the input of $\mathscr{D}_\theta$, outputting $\hat{\mathbf{x}}_0$. That is:

$$\hat{\mathbf{x}}'^{,*}_0 = \mathscr{D}_\theta\left(\underset{\hat{\mathbf{x}}'_0}{\arg\min}\, Dist(\delta, g), t\right),$$

with:

$$\delta = \nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathscr{D}_\theta(\hat{\mathbf{x}}'_0, t) + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2.$$

Fine-tuning phase. The generative phase is able to generate high-quality data, yet, indirectly optimizing $\hat{\mathbf{x}}_0$ does not guarantee pixel-wise similarity. Moreover, each iteration of optimizing the latent code $\hat{\mathbf{x}}'_0$ requires T sampling steps of the trained diffusion

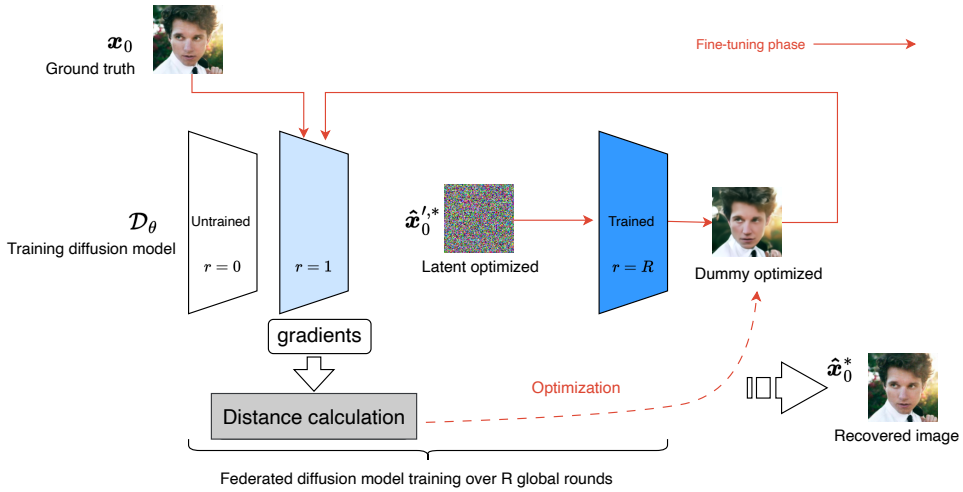


Figure 5.4: We utilize one 128 × 128 image from *CelebA* as an example to show every intermediate result above. The fine-tuning phase consists of 1500 iterations. The latent optimized $\hat{\mathbf{x}}_0'^{,*}$ is the output of the generative phase.

5

model, e.g., $T = 1000$ in DDPM, which is computationally expensive. Therefore, our fine-tuning phase executes direct optimizing of $\hat{\mathbf{x}}_0$ following Eq. 5.2 based on the output of the generative phase $\hat{\mathbf{x}}_0'^{,*}$ to increase the pixel-wise similarity. The details of the finet-tuning phase is presented in Fig.5.4.

The output of the generative phase, i.e., the intermediate dummy data, generated by the optimized latent code $\hat{\mathbf{x}}_0'^{,*}$, is then optimized by the fine-tuning phase. From Fig. 5.4, we see that the generative phase is already able to reconstruct a high-quality image that overall resembles the original picture while the pixel-wise similarity with the original data is low. As in the comparison example, The output of the generative phase differs from the original data in hair, face, and background. By integrating the generative phase and the fine-tuning phase, we recover high-quality data efficiently and effectively with high pixel-wise similarity.

Within both phases, calculating the gradients requires the knowledge of private $\boldsymbol{\epsilon}$ and t. We first conduct an exploratory experiment to determine whether inversion methods for classifiers adapted for diffusion models enable successful attacks. The result, presented in Sec. 5.4.4, demonstrates that adapted existing attacks are unable to recover training samples when applied to diffusion models due to the large search space of the diffusion optimization procedure. This motivates the novel design for gradient approximation on diffusion models.

5.3.4. TRIPLE-OPTIMIZATION

To enable inversion without knowing $\{\boldsymbol{\epsilon}, t\}$, we optimize three different parameters simultaneously while taking their design principles and differences into consideration. Concretely, we design a triple-optimization method to refine the coordination of the three independent optimizations, namely, of $\mathbf{x}_0$, $\boldsymbol{\epsilon}$, and t by determining the approxi-

mations $\hat{\mathbf{x}}_0$, $\hat{\boldsymbol{\epsilon}}$, and $\hat{t}$, respectively. Note that all three optimizations are based on back-propagating with the goal approximating the gradient.

Optimizing $\hat{\boldsymbol{\epsilon}}$. In DDPM, we compute the distribution of the noisy sample after t iterations of the forward process in closed-form by:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}\right).$$

The number of iterations in the forward process is set to a large T, and the variance levels $\beta_t \in (0,1)$ increase linearly (ranging from 10^{-4} to 0.02), which means that $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_t)$ approximates 0 for large t. Thus, the latent distribution is a Gaussian distribution ϵ sampled locally by the client, i.e., we should have $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. To follow such a Gaussian distribution, i.e., reduce the probability of sampling unlikely values, when implementing stochastic gradient descent requires a small learning rate η_ϵ to update $\hat{\boldsymbol{\epsilon}}$. Also, following Eq. 5.1, training diffusion models is to predict the noise added during the forward process and the training performs well without changing the noise term during every iteration of inversion. Thus, we utilize an interval updating strategy for $\hat{\boldsymbol{\epsilon}}$. The updating is achieved by:

$$\hat{\boldsymbol{\epsilon}}^{i+1} = \hat{\boldsymbol{\epsilon}}^{i} - \eta_\epsilon \frac{\partial Dist(\hat{g}, g)}{\partial \hat{\boldsymbol{\epsilon}}^{i}}.$$

Optimizing $\hat{t}$. In contrast to $\boldsymbol{\epsilon}$ and $\mathbf{x}_0$, which are floating tensors, t is an discrete integer in $\{1,\ldots,T\}$. To find the optimal t by stochastic gradient descent, we initialize a $1 \times T$ auxiliary vector following the uniform distribution: $\hat{\boldsymbol{t}} \sim \text{Uniform}(\mathbf{0}, \boldsymbol{J})$ where $\boldsymbol{J}$ is a vector of ones. In each iteration of optimization for approximating the dummy gradient to the real gradient, $\hat{\boldsymbol{t}}$ is updated by the learning rate η_t. The inferred $\hat{t}$ is chosen as the index of the maximum element in the vector after softmax transition: $\hat{t} = \arg\max(softmax(\hat{\boldsymbol{t}}))$. $\hat{t}$ is updated at each inversion iteration:

$$\hat{t}^{i+1} = \hat{t}^{i} - \eta_t \frac{\partial Dist(\hat{g}, g)}{\partial \hat{t}^{i}}.$$

Optimizing $\hat{\mathbf{x}}_0$. Replacing $\boldsymbol{\epsilon}$ and t in Eq. 5.2 with $\hat{\boldsymbol{\epsilon}}$ and $\hat{t}$, we perform the optimization of $\hat{\mathbf{x}}_0$ starting with a random initialized image $\hat{\mathbf{x}}_0 \sim \text{Uniform}(\mathbf{0}, 2^p \boldsymbol{I})$, given that the source data is a p-bit binary encoded value. With each iteration, we update $\hat{\mathbf{x}}_0$ by back-propagating the distance between $\hat{g}$ and g. We set the learning rate of the dummy data as η_x. $\hat{\mathbf{x}}_0$ is updated at each inversion iteration:

$$\hat{\mathbf{x}}_0^{i+1} = \hat{\mathbf{x}}_0^{i} - \eta_x \frac{\partial Dist(\hat{g}, g)}{\partial \hat{\mathbf{x}}_0^{i}}.$$

The optimizers are coordinated based on an optimization interval S. After every S iterations of updating $\hat{\mathbf{x}}_0$ and $\hat{t}$, we perform S iterations of $\hat{\boldsymbol{\epsilon}}$, $\hat{\mathbf{x}}_0$ and $\hat{t}$ simultaneously.

In summary, the triple-optimization gradient inversion coordinates three independent optimization processes to perform data reconstruction for the practical federated diffusion models without knowledge of the private noise and step sampled by each client.

5.4. EXPERIMENTAL EVALUATION

We design a set of experiments to study the impact and effectiveness of gradient inversion attacks on diffusion models, as well as the ablation studies. The experimental setups, baselines, evaluation metrics, employment details, and results are presented in this section.

5.4.1. SETUPS

Datasets. Unless stated otherwise, our experiments use two datasets : *Celeb-A* [93] and *LSUN-Bedroom* [160], both with images resized to 128×128 since the original images from these datasets are of varying sizes. Both datasets are trained using the standard DDPM model [49] as by Alg. 3 with 50 rounds. The optimization interval of GIDM for $\hat{\boldsymbol{\epsilon}}$ optimizer is set to be $S = 50$. For all three optimizations of our triple-optimization, we use Adam as the optimizer. We choose $Dist(\cdot)$ to be the L2-Norm [178] distance, as the distance calculated by cosine similarity is large in our model, which may cause exploding gradients during training.

Hardware. For hardware, we conducted our experiments using an Alienware Aurora R13 running Ubuntu 20.04. This system boasts 64GB of memory, a GeForce RTX 3090 GPU, and a 16-core Intel i9 CPU. With each of its 8 P-cores supporting two threads, the machine houses a total of 24 logical CPU cores.

Hyperparameters. The project of this paper is based on Pytorch 2.3.0. In our gradient inversion attackers, we apply Adam as the optimizer for both $\boldsymbol{\epsilon}$, t and the dummy image. The learning rate used in this paper is 0.01 without any scheduler for learning rate decay. The diffusion model for our experiments is the commonly used DDPM, which uses UNet to implement the sampling of each step. For our topic, the server can reconstruct the data from each client. Thus, we experiment on one random client for this work, following the settings of the baseline works [153, 178].

5.4.2. BASELINES WITH ADAPTATION

Since we are the first to study the gradient inversion attack on diffusion models, there is no direct baseline to compare to. State-of-the-art inversion attacks are designed for classifiers. Thus, we compare GIDM with adapted versions of the attacks that are compatible with diffusion models: DLG-dm [178] and InvG-dm [39] with Adam optimizer. Please note that without knowing $\{\boldsymbol{\epsilon}, t\}$, DLG-dm and InvG-dm are **not** able to invert images similar to the original. As a consequence, we provide $\{\boldsymbol{\epsilon}, t\}$ for these methods, giving them additional information that GIDM does not have. Specifically, for the same training input of images, the initialization works the same for DLG-dm and InvG-dm as for DLG and InvG. The backpropagation for calculating the gradients of diffusion models requires the additional inputs of $\boldsymbol{\epsilon}$ and t, which does not apply to the baseline DLG and InvG. Thus, for DLG-dm and InvG-dm on diffusion models, we randomly sample $\boldsymbol{\epsilon}$ following a Gaussian distribution of the same size of the dummy image and t from $[1, T]$ following a uniform distribution. They are kept constant over all inversion iterations. When it comes to the distance metric, we keep L2-Norm for DLG-dm and cosine similarity for InvG-dm, as for DLG and InvG.

5.4.3. METRICS

In evaluating the gradient inversion attack, which aims to reconstruct data that closely matches the original client data, we employ four metrics to gauge the resemblance between the reconstructed and real data: Mean Squared Error (MSE), Structural Similarity Index (SSIM)[143], Peak Signal-to-Noise Ratio (PSNR)[50], and Learned Perceptual Image Patch Similarity (LPIPS) score [171]. Statistically, *MSE* calculates the average squared difference on a pixel level between the reconstructed and original images:

$$\mathrm{MSE}\left(q_1, q_2\right) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \left(q_1(i,j) - q_2(i,j)\right)^2,$$

where $q_1(i,j)$ and $q_2(i,j)$ indicate the original and recovered images, respectively. *PSNR* relates this *MSE* to the maximum pixel values, concretely, considering the ratio of the maximal value to *MSE* in a logarithmic manner:

$$\mathrm{PSNR} = 10 \log_{10} \frac{R^2}{\mathrm{MSE}},$$

where R is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. For more modern visual assessments, *SSIM* mimics the human visual system to measure the structural variance of images based on luminance, contrast, and structure:

$$SSIM = \frac{2\mu_{x_r}\mu_{x_x} + c_1 2\sigma_{x_x x_x} + c_2}{\mu_{x_r}^2 + \mu_{x_z}^2 + c_1\sigma_{x_r}^2 + \sigma_{x_y}^2 + c_2},$$

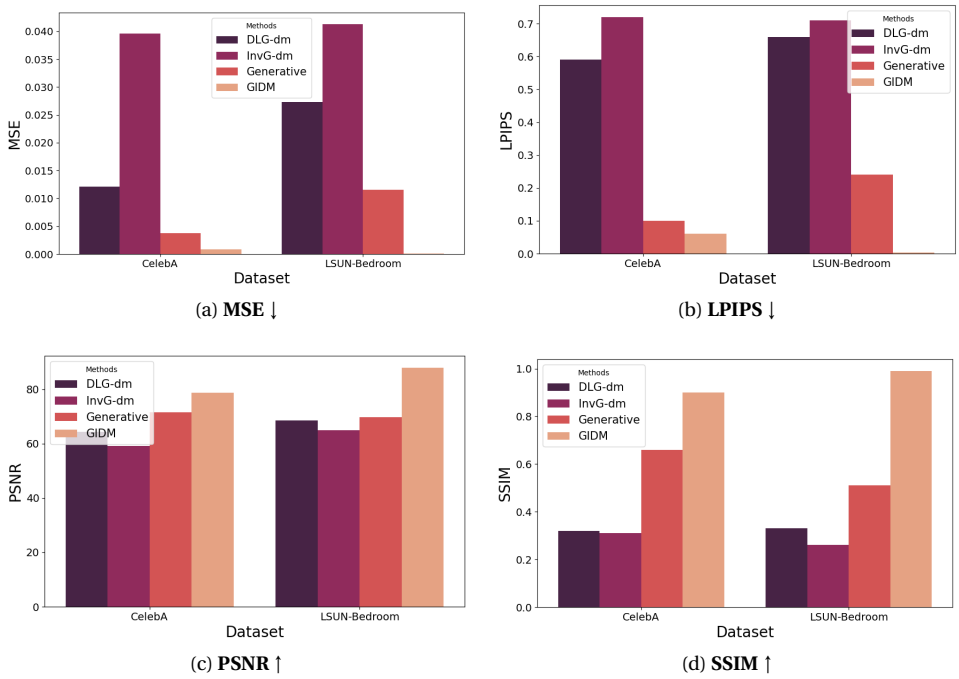
where μ_{x_r} and μ_{x_g} represent the means of the ground truth and the generated image, respectively. Accordingly, $\sigma_{\hat{x}_r}$ and σ_{x_g} are the standard deviations of $\hat{x}_r$ and x_g. Moreover, σ denotes the covariance between both images, while c_1 and c_2 are constants set to avoid instability.

LPIPS determines the perceptual similarity between the original and reconstructed images by learning the inverse mapping from the generated image back to the original. For MSE and LPIPS, lower values indicate greater similarity, whereas for SSIM and PSNR, higher values signify closer resemblance.

5.4.4. FINAL RECONSTRUCTED IMAGES

The performance of our GIDM is assessed by the similarity between the final recovered image and the original image used during training. To demonstrate our inversion effectiveness, we report the final MSE, LPIPS, PSNR, SSIM results in Fig. 5.5. We also visualize examples of the final reconstructed images in Fig. 5.6 for *CelebA* and *LSUN-bedroom*. For GIDM, we assume $\{\boldsymbol{\epsilon}, t\}$ is sampled by the client and randomly initialized: $\hat{\boldsymbol{\epsilon}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\hat{t} \sim \mathrm{Uniform}(\{1, ..., T\})$. Recall that for the baselines, $\{\boldsymbol{\epsilon}, t\}$ is assumed to be known. "Generative" refers to only the output of the generative phase before the fine-tuning phase, with $\{\boldsymbol{\epsilon}, t\}$ known.

From Fig. 5.5, it is evident that GIDM outperforms baseline methods consistently for all four evaluation metrics, demonstrating superior reconstructing results. Yet, our generative phase does not always recover better images than DLG-dm and InvG-dm, which



(a) **MSE** ↓ (b) **LPIPS** ↓

(c) **PSNR** ↑ (d) **SSIM** ↑

Figure 5.5: Inversion results of diffusion models on *CelebA* and *LSUN-Bedroom*. We compare our GIDM (unknown $\{\boldsymbol{\epsilon}, t\}$) with DLG-dm, InvG-dm, and our generative phase result (all three known $\{\boldsymbol{\epsilon}, t\}$). ↓ stands for the lower the better, ↑ for the higher the better.

can be explained by the difference in goals in terms of similarity. The four evaluation metrics compute the quality of similarity based on pixels. In contrast, "Generative" optimizes the latent space to conduct indirect inversion. Thus, the high-quality output (semantically similar and clear) from the diffusion model may result in lower pixel-wise similarity than baselines, which appear comparably blurred in Fig. 5.6. Adding the fine-tuning phase resolves the issue and achieves better performance in terms of the metrics despite using less information.

Fig. 5.6 visualizes and compares the recovered images on both datasets. GIDM successfully reconstructs high-quality images from the gradients that are nearly indistinguishable from the ground truth perceptually. As a comparison, DLG-dm and InvG-dm, which are designed for classifiers, fail to recover images resembling the ground truth. Specifically, they are only able to create images of similar color palettes as the original data without recreating the original object, let alone high-quality details. This meets our expectations since they conduct pixel-wise optimization. These methods suffer from exponentially increased difficulty when the image size is large.

When it comes to the difference between "Generative" and GIDM, we observe that our generative phase can already recover good approximations of the original image. Generally, the main color, object outline, and positions after the generative phase are al-

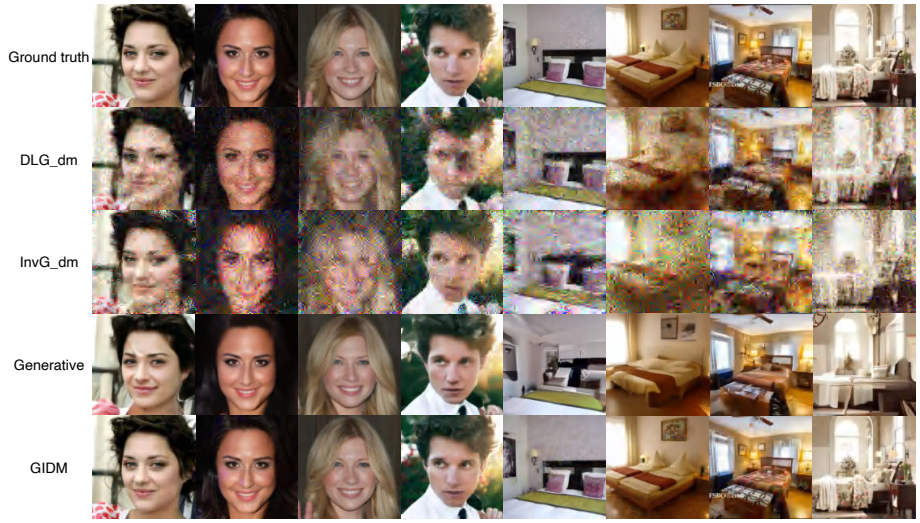


Figure 5.6: Visualization of recovered images comparing with baselines on *CelebA* and *LSUN-Bedroom* datasets. "Generative" in the figure is the output of our generative phase before the fine-tuning phase. For GIDM, we assume that the server does not know $\{\boldsymbol{\epsilon}, t\}$, while the baselines assume known $\{\boldsymbol{\epsilon}, t\}$.

most identical to the ground truth, though there are still minor differences in the details of the images. For example, the face and hair shape, the background texture, or sometimes the makeup color is different from the ground truth for the *CelebA* human-facial dataset. The fine-tuning phase adjusts the generated images by direct gradient approximation, which results in successful reconstruction.

To summarize, gradient inversion of diffusion models is possible. Using the trained diffusion model to constrain the search space is highly effective. Still a fine-tuning phase is required to indeed recover high-quality images. With this phase, GIDM clearly outperforms the adapted baselines, despite using less knowledge.

5.4.5. INTERMEDIATE INVERSION OUTPUTS OF GIDM

We now analyze the quality of the reconstructed images during the optimization process. We use example images from the *CelebA* dataset, both for known and unknown $\{\boldsymbol{\epsilon}, t\}$.

Fig. 5.7 shows the intermediary results for our example image. It highlights that both phases are necessary for the inversion process. For both known and unknown $\{\boldsymbol{\epsilon}, t\}$, we observe gradual improvements, with the color palette and the profile details getting more similar to the original image in each step.

For the generative phase, the initialized random Gaussian noise can directly output a distinguishable image of a human face. However, this image shows an arbitrary human face, with no strong similarity to the original image. Thus, we have constrained the search space to images of human faces, which now enables finding one particular human face. We hence see that leveraging the trained diffusion model as a prior is highly effective in narrowing down the set of potential results.

After calculating the dummy gradient based on the previous image, our back prop-

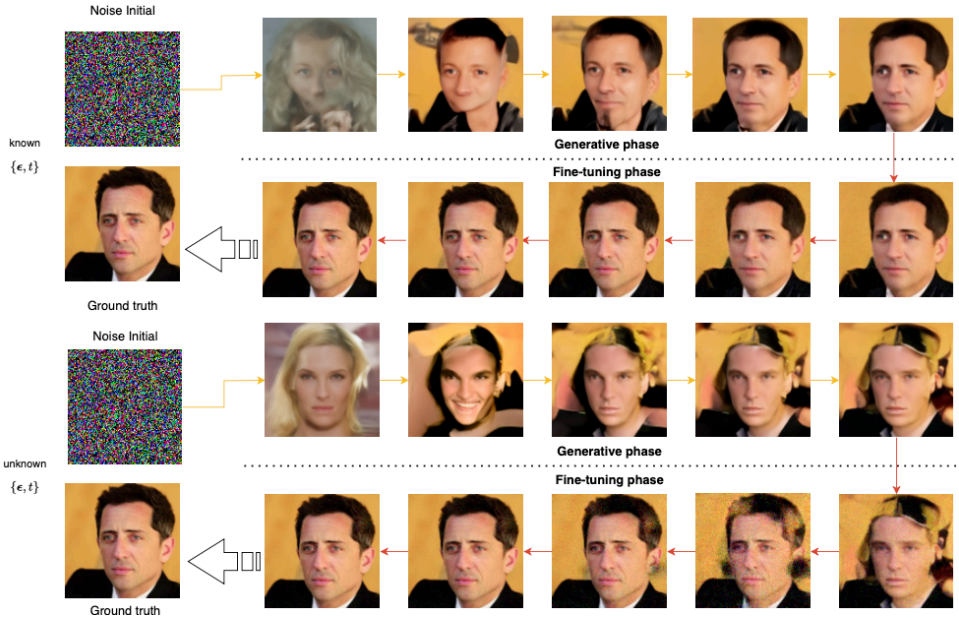


Figure 5.7: Step-by-step gradient inversion intermediate results visualization for known and unknown $\{\boldsymbol{\epsilon}, t\}$. The generative phase is marked by yellow arrows while the fine-tuning phase is red arrows. We initialize the latent code using 128×128 Gaussian noise. The generative phase is trained for $r_g = 5000$ and the fine-tuning phase for $r_f = 1500$ iterations. The same number n_i of images is shown for both phases, with the iterations between images being r_g/n_i and r_f/n_i for the generative and fine-tuning phases, respectively.

agation adjusts the Gaussian noise so that it gradually generates an image of a similar color palette and object profiles. One interesting finding is that GIDM can reconstruct similar colors at an early stage of the training. In contrast, object outlines, such as the hair, converge slowly and gradually. Upon reaching the 3000-th iteration of the generative phase, the recovered image has been gradually started resembling the original image but does not mirror all the details of the original. Note that increasing the number of iterations to 10,000 does still not result in a fully recovered image, further motivating the need for the fine-tuning phase.

The fine-tuning phase starts with the output of the generative phase and we train 1500 iterations. Following the red arrows, the hair outline, and background start to approximate the original data. The fine-tuning phase does not utilize the diffusion model for optimizing alternative space. Thus, the reconstruction is efficient by not unnecessarily executing diffusion model sampling at each optimization iteration as in the generative phase.

5.4.6. THE IMPACT OF KNOWING t OR ϵ ON GIDM

Our results in Sec. 5.4.4 consider $\boldsymbol{\epsilon}$ and t initialized together by the client, which is unknown to the server. Here we provide an ablation study on the impact of knowing one factor and keeps the other one unknown repectively. We start by discussing the impact of known and unknown t. Hence, we assume that $\boldsymbol{\epsilon}$ is known in this section. We use an image from *CelebA* as an example and show the process of optimization in Fig. 5.8. From the steps, it can be observed that when t is known to the server, the optimization process is smooth. This demonstrates that generating the changing t as the input of $\mathscr{D}(\hat{\boldsymbol{x}}_0, t)$ influences the approximation during the optimization process of t. However, even without knowing t, the 128×128 images can be recovered in the end due to the less randomness than $\boldsymbol{\epsilon}$.

Figure 5.8: Step-by-step gradient inversion intermediate results visualization, comparing known or unknown t. We initialize the latent code by 128×128 Gaussian noise. We present some of the representative intermediate outputs through the whole process. The total number of inversion iterations is 4000.

As a comparison, we also evaluate the impact of known or unknown ϵ for the reconstruction. Here, it is also assumed that t is known to the server (adversary). The results are shown in Fig. 5.9. Similarly, in the end, our GIDM is shown to be able to reconstruct images that resembles the original (ground truth in the figure). However, the intermediate output shows different way approaching the end. In general, the middle steps looks more natural perceptually with ϵ known by the server. We could observe smooth changes towards the final results. Additionally, each output demonstrates high quality in terms of both the color blocks and facial profiles (figures above). As a comparison, unknown ϵ causes some deformed images, represented by both color and profiles (figures below). When it comes to the difference between Fig. 5.9 and Fig. 5.8, we find that t influences more on the image content (e.g., natural undistorted face) while ϵ controls more on the image clarity (blurness and random dots).

5.4.7. EFFECTIVENESS OF TRIPLE-OPTIMIZATION

In this part, we conduct an ablation study to showcase the significance of our joint triple-optimization. We apply the two-phase fusion model to compare the final recovered image without optimizing $\hat{t}$ or $\hat{\boldsymbol{\epsilon}}$. An example reconstruction can be seen in Fig. 5.10.

In Fig. 5.10, we can first observe that it is indeed necessary to optimize the three factors jointly. Without such a joint optimization, the reconstructed images do not resemble the original image closely. The difference in importance between the two terms is notable. Concretely, when $\hat{t}$ is not optimized, the resulting image is very close to random

Figure 5.9: Step-by-step gradient inversion intermediate results visualization, comparing known or unknown ϵ. We initialize the latent code by 128 × 128 Gaussian noise. We present some of the representative intermediate outputs through the whole process. The total number of inversion iterations is 4000.

Figure 5.10: The reconstruction results of without optimizing $\hat{t}$ (random sampled), without optimizing $\hat{\boldsymbol{\epsilon}}$ (random sampled) and triple-optimization on an example of *CelebA*.

pixels. In contrast, when $\hat{t}$ is optimized but $\hat{\boldsymbol{\epsilon}}$ not, the reconstructed images is clearly that of a human face, though the details are not reconstructed.

Although $\hat{t}$ is only a single value that has a limited range, e.g., integer values between 0 and 999 for the classical DDPM, it decides the specific position on the Markov chain to train on. It guides the training process, similar to the label of classification tasks, which has a huge impact on the gradients. Thus, without having $\hat{t}$ resemble the parameter used during training, we have a very noisy image.

Without optimizing the noise term $\hat{\boldsymbol{\epsilon}}$, which is a float tensor with more randomness, the image quality remains low but the attack is able to recover most color palettes and some outline information. We argue that there might be two reasons. First, $\hat{\boldsymbol{\epsilon}}$ is introduced as a prediction target for training, which is less sensitive to incorrect sampling than $\hat{t}$, which directly instructs the noising process through the position of the chain. Second, the noise term is designed to follow a Gaussian distribution, avoiding unlikely values during inversion avoids extreme differences. In summary, our triple-optimization is indeed required.

5.5. CONCLUSION

In this paper, we are the first to study gradient inversion attacks on diffusion models. Leveraging the trained diffusion models as prior knowledge to constrain the search space, we propose an attack GIDM consisting of two phases. In the first phase, we use the trained diffusion model to approximate the image before fine-tuning the image to resemble the origin on a pixel-by-pixel basis in the second phase. The key challenge in designing GIDM is that the noise and sampling step are not known to the attacker, which we solved by the use of a joint optimization algorithm. Our reconstruction results are impressive, almost perfectly recreating images up to 128×128 in size.

In the future, we aim to develop a defense mechanism against the inversion attack, e.g., by using differential privacy [27]. It is unclear how differential privacy can be integrated in such a manner that the quality of the trained diffusion model is not severely impacted.

6

CONCLUSION

The primary aim of this thesis is to develop mechanisms for enhancing the robustness of federated learning systems. In the face of several vulnerabilities, we have considered the privacy and heterogeneity of the clients and the security of the server(s), and four related research questions are introduced. We have concluded by examining how such proposed mechanisms can improve the system's robustness under different clients and servers with attacking scenarios. In this chapter, we summarize the conclusions and discuss the limitations and future directions of this thesis.

6.1. CONCLUSIONS

This thesis detects essential vulnerabilities of federated learning systems by involving adversarial or heterogeneous clients and honest-but-curious servers. Specifically, four key contributions are made to address the identified research questions (RQ1-RQ4), advancing both theoretical understanding and practical solutions for federated learning robustness:

1. **Optimization under Client Heterogeneity (RQ1).** In Chapter 2, we introduced the importance of client selection in FL as a critical factor for maximizing the utility of diverse clients. We have shown that existing schemes struggle when faced with heterogeneous data, particularly when one or more classes are exclusively owned by Mavericks. We first explored Shapley value-based selection and theoretically demonstrated its limitations in addressing the challenge posed by Mavericks. We then proposed FEDEMD, a method that encourages the selection of diverse clients at the optimal stage of the training process, ensuring guaranteed convergence. We conclude that our evaluation of multiple datasets and scenarios involving Mavericks shows that FEDEMD reduces the number of communication rounds needed for convergence.

2. **Data-free Adversarial Vulnerability (RQ2).** In Chapter 3, we present DFA, the first data-free untargeted attack in FL. We demonstrated that data-free attacks can be

as effective (or even more effective than as) other attacks that require data or benign updates, primarily because synthetic images generated for training are particularly effective at steering the model in the wrong direction. Furthermore, we designed a defense strategy, REFD, that effectively protects against the proposed DFA as well as existing attacks, by leveraging the statistics of model outputs to predict reference data. We conclude that synthetic data can also be used for launching effective untargeted attacks in FL systems, raising our attention to system security, especially under practical FL scenarios where clients are anonymous.

3. **Privacy Risks by Data Reuse (RQ3).** In Chapter 4, we introduce the first analysis of the risk of data reconstruction when a client repeatedly uses their data to contribute to multiple tasks. We have shown, through both analytical and empirical results, that our attacks, CGI-S and CGI-D, are significantly more impactful than attacks in a single-task setting, even in the presence of defense mechanisms. We conclude that our attack outperforms others, particularly on larger image sizes, larger batch sizes, and more complex deep neural networks.

4. **Generative Model Vulnerabilities (RQ4).** In Chapter 5, we study gradient inversion attacks on diffusion models, marking the first exploration of this area. We have shown that when an adversarial server knows the Gaussian noise and sampling step used during training, our proposed GIDM effectively optimizes a dummy image through a generative phase followed by a fine-tuning phase. Our experiments demonstrate the critical role of both phases in recovering high-resolution data, enabling successful reconstruction where other attacks fail.

In summary, the findings from these chapters demonstrate how federated learning systems can be strengthened against various client and server-side vulnerabilities, providing a comprehensive approach to enhancing the robustness of FL systems under diverse conditions. We derive the following two high-level conclusions for both sides:

1. **The Evolving Role of Data**. This thesis challenges traditional assumptions about data's role in federated learning, discussing its dual identity as both for improvement and for vulnerability. While client heterogeneity and synthetic data generation advance model training, they also bring fundamental threats: client selection strategies must balance efficiency with effectiveness, and synthetic inputs designed for preserving privacy empower adversaries. Our work demonstrates that the expanding usage of data (e.g., in generative models or cross-task reuse) enlarges the attack surface, demanding systems that natively integrate robustness and privacy into their design. These findings highlight that future distributed learning frameworks must treat data not as a static resource but as a dynamic and challenging element in distributed machine learning.

2. **The Trade-Offs upon Design**. Distributed machine learning systems are within a constrained design space, where advances in one dimension (e.g., convergence speed, privacy, or robustness) often compromise another. Through theoretical and empirical analysis, this thesis discusses these trade-offs, showing how client

heterogeneity increases vulnerability for attackers, while defenses against data reconstruction reduce model utility. Also, we show that traditional adversaries fail to consider risks arising from more practical aspects (e.g., data reuse or generative training), promoting solutions that address interdependencies between performance and security. The broader implication is clear: federated learning's next generation must move beyond isolated optimizations and design frameworks where trade-offs are explicitly modeled and managed as a core requirement of system architecture.

6.2. LIMITATIONS

While this thesis explores significant aspects of enhancing the robustness of federated learning systems, limitations remain. We clarify the limitations of each chapter in the following:

1. In Chapter 2, like most experimental research works, the proposed solutions, such as client selection strategies, primarily focus on constrained experimental settings and may not fully capture the complexity of real-world, large-scale federated learning environments, where the behavior of clients and servers can vary significantly. Also, the effectiveness of these strategies may decrease in scenarios with extreme heterogeneity.

2. In Chapter 3, the defense design targets DFA specifically. To enhance the generalization, REFD needs to consider when active adversarial behaviors evolve over time. The scalability and computational overhead of some proposed solutions, especially when DFA involves adaptive strategies and complex attack models, could pose challenges when applied to large-scale federated systems.

3. In Chapter 4, while the analysis of privacy risks, such as gradient inversion attacks and data reuse, provides concrete insights, it is based on certain assumptions about system architecture, noise parameters, and model structures that may not always hold in practical implementations. The evaluated defenses, such as those based on statistical modeling, may require further refinement to ensure they can handle more sophisticated attack methods and real-world deployment challenges.

4. In Chapter 5, the study focuses mainly on specific vulnerabilities of gradient inversion on diffusion models, with concrete solutions, but no high-level strategy for designing such solutions was extensively summarized, e.g., intuitions for other types of federated tasks. Additionally, although this work is a new field in gradient inversion of diffusion models themselves, more adapted baselines can be included to demonstrate the conclusions.

6.3. FUTURE DIRECTIONS

This thesis has provided an exploratory analysis of the solution space of FL robustness under different settings. Therefore, there is still much to be explored beyond what is presented in this thesis. We now provide several promising directions for future work related to each chapter:

1. In Chapter 2, we propose a selection strategy for the FL settings, including Mavericks. Future research should focus on further refining and expanding adaptive client selection strategies, particularly in environments characterized by extreme data heterogeneity. Investigating how these strategies can be integrated with other advanced techniques, such as personalized FL models [134] or differential privacy [1], could enhance both the efficiency and security of FL systems. Additionally, exploring the impact of Mavericks in real-world applications and extending FEDEMD's applicability to other types of learning tasks, such as reinforcement learning or unsupervised learning, could provide deeper insights and broader utility. Future studies might also consider developing more sophisticated models that can dynamically adjust to varying levels of client participation and data availability, ensuring optimal performance even in more complex and diverse federated learning scenarios.

2. In Chapter 3, we propose an untargeted data-free attack. We could explore adaptive defense mechanisms that dynamically respond to evolving attack strategies, particularly in scenarios where attackers continuously modify their approaches to bypass defenses like REFD. Additionally, investigating the integration of REFD with other defense techniques, such as differential privacy or anomaly detection, could further improve the security and privacy of FL models. There is also a need for more extensive real-world testing and benchmarking of these defense strategies across a wider array of datasets and FL applications. Finally, research could focus on creating proactive measures that predict potential vulnerabilities in FL systems, reducing the need for reactive defenses and keeping models secure against new types of attacks.

3. In Chapter 4, we discover privacy leakage of data reuse when a data source contributes to a multi-server system. We could focus on developing more robust defense mechanisms that can protect against gradient-based reconstruction across various tasks. Additionally, this research can be integrated with Chapter. 5 to extend the conclusion to different tasks and models for generality. Expanding this research to include different types of data, such as text or time-series data, and testing in more complex, real-world FL environments would also be valuable. Finally, there is a need for theoretical advancements that provide stronger guarantees of privacy preservation, even when data is reused extensively in FL systems.

4. In Chapter 5, we show how the original data is exposed for the training of Federated diffusion models without the presence of external knowledge from another well-trained generative model. Building on the vulnerabilities exposed by gradient inversion attacks as GIDM, future work could explore the robustness of these models against more complex and adaptive adversarial strategies that could help strengthen their resilience. Expanding the research to include a broader range of data types and more diverse federated settings would also provide deeper insights into potential weaknesses. One finding worth mentioning is that while uncovering the relationship between the gradients and the real data, we noticed the implicit link between the random noise input and the sampled data output during the dif-

fusion process. This could assist in understanding the diffusion process concerning both generative capability and stability.

In summary, the future directions of this thesis lie in three main parts: 1) detecting vulnerabilities over a broader range of data types and different scenarios of distributed machine learning systems; 2) designing defense mechanisms to enhance the robustness of the systems, targeting more advanced adversaries; 3) improving practical attack methods considering the adversarial behaviours from a third-party (external) and hardware integration. The research on detecting the vulnerability of distributed machine learning systems assists in building robust systems with the development of machine learning systems. Future work will progress the functionality, scalability, and reliability in a mutually reinforcing manner.

Acknowledgements

To those who supported me with love,
to those who challenged me with truth,
to Chi, who walked beside me in silence,
to Michael Jackson—
this work is also yours.

For the questions, the doubts,
for the undying spark within my heart,
for the stars that shone through the long nights,
for Delft and Haarlem,
I remain deeply grateful—
for this life.

BIBLIOGRAPHY

[1] Martín Abadi et al. “Deep Learning with Differential Privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016*. ACM, 2016, pp. 308–318.

[2] Accountability Act. “Health insurance portability and accountability act of 1996”. In: *Public law* 104 (1996), p. 191.

[3] Alham Fikri Aji and Kenneth Heafield. “Sparse Communication for Distributed Gradient Descent”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*. Association for Computational Linguistics, 2017, pp. 440–445.

[4] Martín Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223.

[5] Eugene Bagdasaryan et al. “How to backdoor federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2020, pp. 2938–2948.

[6] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. “Shaping the learning landscape in neural networks around wide flat minima”. In: *Proc. Natl. Acad. Sci. USA* 117.1 (2020), pp. 161–170.

[7] Gilad Baruch, Moran Baruch, and Yoav Goldberg. “A Little Is Enough: Circumventing Defenses For Distributed Learning”. In: *Conference on Neural Information Processing Systems*. 2019, pp. 8632–8642.

[8] Peter J Bickel and Kjell A Doksum. *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. CRC Press, 2015.

[9] Peva Blanchard et al. “Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent”. In: *Conference on Neural Information Processing Systems*. 2017, pp. 119–129.

[10] Kallista A. Bonawitz et al. “Practical Secure Aggregation for Privacy-Preserving Machine Learning”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*. ACM, 2017, pp. 1175–1191.

[11] Keith Bonawitz et al. “Towards federated learning at scale: System design”. In: *Proceedings of machine learning and systems* 1 (2019), pp. 374–388.

[12] Nicholas Carlini et al. “Extracting Training Data from Diffusion Models”. In: *32nd USENIX Security Symposium, USENIX Security 2023*. USENIX Association, 2023, pp. 5253–5270.

[13] Zheng Chai et al. “TiFL: A Tier-Based Federated Learning System”. In: *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. 2020, pp. 125–136.

[14] Jian Chen et al. “De-Pois: An Attack-Agnostic Defense against Data Poisoning Attacks”. In: *IEEE Trans. Inf. Forensics Secur.* 16 (2021), pp. 3412–3425.

[15] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. “Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies”. In: *arXiv preprint arXiv:2010.01243* (2020).

[16] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. “Towards understanding biased client selection in federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 10351–10375.

[17] Junyoung Chung et al. “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *CoRR* abs/1412.3555 (2014).

[18] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics (AISTATS)*. JMLR Workshop and Conference Proceedings. 2011.

[19] Adam Coates, Andrew Y. Ng, and Honglak Lee. “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011*. Vol. 15. JMLR Proceedings. JMLR.org, 2011, pp. 215–223.

[20] Gregory Cohen et al. “EMNIST: an extension of MNIST to handwritten letters”. In: *CoRR* abs/1702.05373 (2017).

[21] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[22] Yongheng Deng et al. “Improving Federated Learning With Quality-Aware User Incentive and Auto-Weighted Model Aggregation”. In: *IEEE Trans. Parallel Distributed Syst.* 33.10 (2022), pp. 4515–4529.

[23] Ronald DeVore, Boris Hanin, and Guergana Petrova. “Neural network approximation”. In: *Acta Numerica* 30 (2021), pp. 327–444.

[24] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004.

[25] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. “Personalized Federated Learning with Moreau Envelopes”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[26] Jinhao Duan et al. “Are Diffusion Models Vulnerable to Membership Inference Attacks?” In: *International Conference on Machine Learning, ICML 2023*. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 8717–8730.

[27] Cynthia Dwork. “The Differential Privacy Frontier (Extended Abstract)”. In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009*. Vol. 5444. Lecture Notes in Computer Science. Springer, 2009, pp. 496–502.

[28] Pavlos S Efraimidis and Paul G Spirakis. “Weighted random sampling with a reservoir”. In: *Information Processing Letters* 97.5 (2006), pp. 181–185.

[29] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. “Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[30] Lixin Fan et al. “Rethinking Privacy Preserving Deep Learning: How to Evaluate and Thwart Privacy Attacks”. In: *Federated Learning - Privacy and Incentive*. Vol. 12500. Lecture Notes in Computer Science. Springer, 2020, pp. 32–50.

[31] Hao Fang et al. “GIFD: A Generative Gradient Inversion Method with Feature Domain Optimization”. In: *IEEE/CVF International Conference on Computer Vision, ICCV 2023*. IEEE, 2023, pp. 4944–4953.

[32] Minghong Fang et al. “Local Model Poisoning Attacks to Byzantine-Robust Federated Learning”. In: *USENIX Security Symposium*. 2020, pp. 1605–1622.

[33] Minghong Fang et al. “Local model poisoning attacks to Byzantine-robust federated learning”. In: *arXiv preprint arXiv:1911.11815* (2019).

[34] Shaohan Feng et al. “Joint Service Pricing and Cooperative Relay Communication for Federated Learning”. In: *2019 IEEE iThings and GreenCom and IEEE Cyber and CPSCom and SmartData, iThings/GreenCom/CPSCom/SmartData 2019*. IEEE, 2019, pp. 815–820.

[35] Yann Fraboni, Richard Vidal, and Marco Lorenzi. “Free-rider attacks on model aggregation in federated learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2021, pp. 1846–1854.

[36] Drew Fudenberg and Jean Tirole. *Game theory*. MIT press, 1991.

[37] Clement Fung, Chris J M Yoon, and Ivan Beschastnikh. “Mitigating sybils in federated learning poisoning”. In: *arXiv preprint arXiv:1808.04866* (2018).

[38] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. “The Limitations of Federated Learning in Sybil Settings”. In: *23rd International Symposium on Research in Attacks, Intrusions and Defenses*. 2020, pp. 301–316.

[39] Jonas Geiping et al. “Inverting Gradients - How easy is it to break privacy in federated learning?” In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*. 2020.

[40] Robin C Geyer, Tassilo Klein, and Moin Nabi. “Differentially private federated learning: A client level perspective”. In: *arXiv preprint arXiv:1712.07557* (2017).

[41] Jack Goetz et al. “Active federated learning”. In: *arXiv preprint arXiv:1909.12641* (2019).

[42] Zenghao Guan et al. “GIE : Gradient Inversion with Embeddings”. In: *IEEE International Conference on Multimedia and Expo, ICME 2024*. IEEE, 2024, pp. 1–6.

[43] Dong-Jun Han et al. “FedMes: Speeding Up Federated Learning With Multiple Edge Servers”. In: *IEEE J. Sel. Areas Commun.* 39.12 (2021), pp. 3870–3885.

[44] Andrew Hard et al. “Federated learning for mobile keyboard prediction”. In: *arXiv preprint arXiv:1811.03604* (2018).

[45] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*. IEEE Computer Society, 2016, pp. 770–778.

[46] Xinlong He et al. “Enhance membership inference attacks in federated learning”. In: *Comput. Secur.* 136 (2024), p. 103535.

[47] Geoffrey E. Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *CoRR* abs/1207.0580 (2012).

[48] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. “Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*. ACM, 2017, pp. 603–618.

[49] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*. 2020.

[50] Alain Horé and Djemel Ziou. “Image Quality Metrics: PSNR vs. SSIM”. In: *20th International Conference on Pattern Recognition, ICPR 2010*. IEEE Computer Society, 2010, pp. 2366–2369.

[51] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. “Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification”. In: *CoRR* abs/1909.06335 (2019).

[52] Hailong Hu and Jun Pang. “Loss and Likelihood Based Membership Inference of Diffusion Models”. In: *Information Security - 26th International Conference, ISC 2023*. Vol. 14411. Lecture Notes in Computer Science. Springer, 2023, pp. 121–141.

[53] Gary B. Huang et al. “Learning to Align from Scratch”. In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*. 2012, pp. 773–781.

[54] Jiyue Huang. *Collusive Gradient Inversion*. `https://github.com/GillHuang-Xtler/CGI`. 2023.

[55] Jiyue Huang. *Data-free Untargeted attack*. `https://github.com/GillHuang-Xtler/DFA`. 2022.

[56] Jiyue Huang. *Diffusion Distillation*. `https://github.com/GillHuang-Xtler/Diffusion_inversion`. 2024.

[57] Jiyue Huang. *Mavericks*. `https://github.com/GillHuang-Xtler/Mav_text`. 2021.

[58] Jiyue Huang et al. “An Exploratory Analysis on Users’ Contributions in Federated Learning”. In: *arXiv preprint arXiv:2011.06830* (2020).

[59] Jiyue Huang et al. “Fabricated Flips: Poisoning Federated Learning without Data”. In: *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Network, DSN 2023*. IEEE, 2023, pp. 274–287.

[60] Tiansheng Huang et al. “An Efficiency-boosting Client Selection Scheme for Federated Learning with Fairness Guarantee”. In: *IEEE Transactions on Parallel and Distributed Systems* (2020).

[61] Yangsibo Huang et al. “Evaluating Gradient Inversion Attacks and Defenses in Federated Learning”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*. 2021, pp. 7232–7241.

[62] Yangsibo Huang et al. “Evaluating Gradient Inversion Attacks and Defenses in Federated Learning”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*. 2021, pp. 7232–7241.

[63] Yangsibo Huang et al. “InstaHide: Instance-hiding Schemes for Private Distributed Learning”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 4507–4518.

[64] Ahmed Imteaj and M. Hadi Amini. “Leveraging asynchronous federated learning to predict customers financial distress”. In: *Intell. Syst. Appl.* 14 (2022), p. 200064.

[65] Joel Janai et al. “Computer vision for autonomous vehicles: Problems, datasets and state of the art”. In: *Foundations and Trends® in Computer Graphics and Vision* 12.1–3 (2020), pp. 1–308.

[66] Jinwoo Jeon et al. “Gradient Inversion with Generative Image Prior”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*. 2021, pp. 29898–29908.

[67] Xiao Jin et al. “Cafe: Catastrophic data leakage in vertical federated learning”. In: *Advances in neural information processing systems* 34 (2021), pp. 994–1006.

[68] Jiawen Kang et al. “Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory”. In: *IEEE Internet Things J.* 6.6 (2019), pp. 10700–10714.

[69] Sai Praneeth Karimireddy et al. “Breaking the centralized barrier for cross-device federated learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28663–28676.

[70] Ekaterina Khramtsova et al. “Federated Learning For Cyber Security: SOC Collaboration For Malicious URL Detection”. In: *40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020*. IEEE, 2020, pp. 1316–1321.

[71] Nicolas Kourtellis, Kleomenis Katevas, and Diego Perino. “Flaas: Federated learning as a service”. In: *Proceedings of the 1st workshop on distributed machine learning*. 2020, pp. 7–13.

[72] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[73] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[74] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[75] Tra Huong Thi Le et al. “An Incentive Mechanism for Federated Learning in Wireless Cellular Networks: An Auction Approach”. In: *IEEE Trans. Wirel. Commun.* 20.8 (2021), pp. 4874–4887.

[76] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. “Deep learning”. In: *Nat.* 521.7553 (2015), pp. 436–444.

[77] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324.

[78] Daixun Li et al. “FedDiff: Diffusion Model Driven Federated Learning for Multi-Modal and Multi-Clients”. In: *IEEE Trans. Circuits Syst. Video Technol.* 34.10 (2024), pp. 10353–10367.

[79] Huiying Li et al. “Blacklight: Scalable Defense for Neural Networks against Query-Based Black-Box Attacks”. In: *31st USENIX Security Symposium, USENIX Security 2022*. USENIX Association, 2022, pp. 2117–2134.

[80] Tian Li et al. “Federated Optimization in Heterogeneous Networks”. In: *Proceedings of Machine Learning and Systems (MLsys)*. 2020.

[81] Xiang Li et al. “On the convergence of fedavg on non-iid data”. In: *ICLR* (2020).

[82] Zewen Li et al. “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Trans. Neural Networks Learn. Syst.* 33.12 (2022), pp. 6999–7019.

[83] Zhuohang Li et al. “Auditing Privacy Defenses in Federated Learning via Generative Gradient Leakage”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*. IEEE, 2022, pp. 10122–10132.

[84] David Lie and Petros Maniatis. “Glimmers: Resolving the privacy/trust quagmire”. In: *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. 2017, pp. 94–99.

[85] Jierui Lin, Min Du, and Jian Liu. “Free-riders in Federated Learning: Attacks and Defenses”. In: *CoRR* abs/1911.12560 (2019).

[86] Jierui Lin, Min Du, and Jian Liu. “Free-riders in Federated Learning: Attacks and Defenses”. In: *CoRR* abs/1911.12560 (2019).

[87] Tao Lin et al. “Ensemble Distillation for Robust Model Fusion in Federated Learning”. In: *Conference on Neural Information Processing Systems*. 2020.

[88] Yujun Lin et al. “Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training”. In: *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[89] Lumin Liu et al. “Client-Edge-Cloud Hierarchical Federated Learning”. In: *2020 IEEE International Conference on Communications, ICC 2020*. IEEE, 2020, pp. 1–6.

[90] Sheng Liu, Zihan Wang, and Qi Lei. “Data Reconstruction Attacks and Defenses: A Systematic Evaluation”. In: *arXiv preprint arXiv:2402.09478* (2024).

[91] Shunjian Liu, Xinxin Feng, and Haifeng Zheng. “Overcoming Forgetting in Local Adaptation of Federated Learning Model”. In: *Advances in Knowledge Discovery and Data Mining - 26th Pacific-Asia Conference, PAKDD 2022*. Vol. 13280. Lecture Notes in Computer Science. Springer, 2022, pp. 613–625.

[92] Yang Liu, Zhihao Yi, and Tianjian Chen. “Backdoor attacks and defenses in feature-partitioned collaborative learning”. In: *CoRR* abs/2007.03608 (2020).

[93] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015*. IEEE Computer Society, 2015, pp. 3730–3738.

[94] Christos Louizos, Max Welling, and Diederik P. Kingma. “Learning Sparse Neural Networks through L_0 Regularization”. In: *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[95] Bing Luo et al. “Tackling System and Statistical Heterogeneity for Federated Learning with Adaptive Client Sampling”. In: *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022*. IEEE, 2022, pp. 1739–1748.

[96] Brendan McMahan et al. “Communication-Efficient Learning of Deep Networks from Decentralized Data”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1273–1282.

[97] Luca Melis et al. “Exploiting Unintended Feature Leakage in Collaborative Learning”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019*. IEEE, 2019, pp. 691–706.

[98] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. “The Hidden Vulnerability of Distributed Learning in Byzantium”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. 2018, pp. 3518–3527.

[99] Khalil Muhammad et al. “Fedfast: Going beyond average for faster training of federated recommender systems”. In: *Proceedings of the 26th ACM International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1234–1242.

[100] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. “Toward Robustness and Privacy in Federated Learning: Experimenting with Local and Central Differential Privacy”. In: *CoRR* abs/2009.03561 (2020).

[101] John Nash. “Two-person cooperative games”. In: *Econometrica: Journal of the Econometric Society* (1953), pp. 128–140.

[102] Milad Nasr, Reza Shokri, and Amir Houmansadr. “Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019*. IEEE, 2019, pp. 739–753.

[103] Aviv Navon et al. “Multi-Task Learning as a Bargaining Game”. In: *International Conference on Machine Learning, ICML 2022*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 16428–16446.

[104] Yuval Netzer et al. “Reading digits in natural images with unsupervised feature learning”. In: (2011).

[105] Hung T Nguyen et al. “Fast-convergent federated learning”. In: *IEEE Journal on Selected Areas in Communications* 39.1 (2020), pp. 201–218.

[106] Thien Duc Nguyen et al. “DÏoT: A Federated Self-learning Anomaly Detection System for IoT”. In: *IEEE International Conference on Distributed Computing Systems*. 2019, pp. 756–767.

[107] Takayuki Nishio and Ryo Yonetani. “Client selection for federated learning with heterogeneous resources in mobile edge”. In: *IEEE International Conference on Communications (ICC)*. 2019, pp. 1–7.

[108] Le Trieu Phong et al. “Privacy-Preserving Deep Learning via Additively Homomorphic Encryption”. In: *IEEE Trans. Inf. Forensics Secur.* 13.5 (2018), pp. 1333–1345.

[109] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. “Mlaas: Machine learning as a service”. In: *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE. 2015, pp. 896–902.

[110] Adam Richardson, Aris Filos-Ratsikas, and Boi Faltings. “Rewarding High-Quality Data via Influence Functions”. In: *CoRR* abs/1908.11598 (2019).

[111] Nicola Rieke et al. “The Future of Digital Health with Federated Learning”. In: *CoRR* abs/2003.08119 (2020).

[112] Nicola Rieke et al. “The future of digital health with federated learning”. In: *NPJ digital medicine* 3.1 (2020), p. 119.

[113] Robin Rombach et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*. IEEE, 2022, pp. 10674–10685.

[114] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[115] Chitwan Saharia et al. “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022*. 2022.

[116] Yutaka Sasaki et al. “The truth of the F-measure”. In: *Teach tutor mater* 1.5 (2007), pp. 1–5.

[117] Aditya Shankar et al. “SiloFuse: Cross-silo Synthetic Data Generation with Latent Tabular Diffusion Models”. In: *CoRR* abs/2404.03299 (2024).

[118] Virat Shejwalkar and Amir Houmansadr. “Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning”. In: *Annual Network and Distributed System Security Symposium*. 2021.

[119] Virat Shejwalkar et al. “Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning”. In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 1354–1371.

[120] Reza Shokri et al. “Membership Inference Attacks Against Machine Learning Models”. In: *2017 IEEE Symposium on Security and Privacy, SP 2017*. IEEE Computer Society, 2017, pp. 3–18.

[121] Rachael Hwee Ling Sim et al. “Collaborative machine learning with incentive-aware model rewards”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 8927–8936.

[122] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[123] Gowthami Somepalli et al. “Diffusion Art or Digital Forgery? Investigating Data Replication in Diffusion Models”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023*. IEEE, 2023, pp. 6048–6058.

[124] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

[125] Mengkai Song et al. “Analyzing User-Level Privacy Attack Against Federated Learning”. In: *IEEE J. Sel. Areas Commun.* 38.10 (2020), pp. 2430–2444.

[126] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958.

[127] Hang Su et al. “Multi-view Convolutional Neural Networks for 3D Shape Recognition”. In: *IEEE International Conference on Computer Vision*. 2015, pp. 945–953.

[128] Dianbo Sui et al. “FedED: Federated Learning via Ensemble Distillation for Medical Relation Extraction”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*. Association for Computational Linguistics, 2020, pp. 2118–2128.

[129] Jingwei Sun et al. “FL-WBC: Enhancing Robustness against Model Poisoning Attacks in Federated Learning from a Client Perspective”. In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*. 2021, pp. 12613–12624.

[130] Jingwei Sun et al. “Soteria: Provable Defense Against Privacy Leakage in Federated Learning From Representation Perspective”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*. Computer Vision Foundation / IEEE, 2021, pp. 9311–9319.

[131] Xu Sun et al. “meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting”. In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3299–3308.

[132] Ziteng Sun et al. “Can You Really Backdoor Federated Learning?” In: *CoRR* abs/1911.07963 (2019).

[133] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[134] Alysa Ziying Tan et al. “Towards personalized federated learning”. In: *IEEE transactions on neural networks and learning systems* 34.12 (2022), pp. 9587–9603.

[135] Vale Tolpegin et al. “Data Poisoning Attacks Against Federated Learning Systems”. In: *Computer Security - ESORICS 2020*. Vol. 12308. Springer, 2020, pp. 480–501.

[136] Lloyd N Trefethen. *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[137] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, and Thorsteinn S. Rögnvaldsson. “Personalized Federated Learning with Contextual Modulation and Meta-Learning”. In: *Proceedings of the 2024 SIAM International Conference on Data Mining, SDM 2024*. SIAM, 2024, pp. 842–850.

[138] Paul Voigt and Axel Von dem Bussche. “The eu general data protection regulation (gdpr)”. In: *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.

[139] Hongyi Wang et al. “Federated Learning with Matched Averaging”. In: *International Conference on Learning Representations*. 2020.

[140] Lixu Wang et al. “Addressing Class Imbalance in Federated Learning”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI, IAAI, EAAI*. AAAI Press, 2021, pp. 10165–10173.

[141] Tianhao Wang et al. “A Principled Approach to Data Valuation for Federated Learning”. In: *Federated Learning*. 2020, pp. 153–167.

[142] Ze Wang et al. “Binary Latent Diffusion”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023*. IEEE, 2023, pp. 22576–22585.

[143] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Trans. Image Process.* 13.4 (2004), pp. 600–612.

[144] Shuyue Wei et al. “Efficient and Fair Data Valuation for Horizontal Federated Learning”. In: *Federated Learning*. 2020, pp. 139–152.

[145] Yuxin Wen et al. “Fishing for User Data in Large-Batch Federated Learning via Gradient Magnification”. In: *International Conference on Machine Learning, ICML 2022*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 23668–23684.

[146] Ruihan Wu et al. “Learning To Invert: Simple Adaptive Attacks for Gradient Inversion in Federated Learning”. In: *Uncertainty in Artificial Intelligence, UAI 2023*. Vol. 216. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2293–2303.

[147] Ruihan Wu et al. “Learning To Invert: Simple Adaptive Attacks for Gradient Inversion in Federated Learning”. In: *Uncertainty in Artificial Intelligence, UAI 2023*. Vol. 216. Proceedings of Machine Learning Research. PMLR, 2023, pp. 2293–2303.

[148] Steven H. Wu et al. “Estimating error models for whole genome sequencing using mixtures of Dirichlet-multinomial distributions”. In: *Bioinform.* 33.15 (2017), pp. 2322–2329.

[149] Yixin Wu et al. “Membership Inference Attacks Against Text-to-image Generation Models”. In: *CoRR* abs/2210.00968 (2022).

[150] Weihao Xia et al. “GAN Inversion: A Survey”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.3 (2023), pp. 3121–3138.

[151] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[152] Chulin Xie et al. “DBA: Distributed Backdoor Attacks against Federated Learning”. In: *8th International Conference on Learning Representations*, 2020.

[153] Jin Xu et al. “AGIC: Approximate Gradient Inversion Attack on Federated Learning”. In: *41st International Symposium on Reliable Distributed Systems, SRDS 2022*. IEEE, 2022, pp. 12–22.

[154] Qiang Yang et al. “Federated Machine Learning: Concept and Applications”. In: *ACM Trans. Intell. Syst. Technol.* 10.2 (2019), 12:1–12:19.

[155] Qiang Yang et al. “Federated Machine Learning: Concept and Applications”. In: *ACM Trans. Intell. Syst. Technol.* 10.2 (2019), 12:1–12:19.

[156] Xucheng Ye et al. “Accelerating CNN Training by Pruning Activation Gradients”. In: *Computer Vision - ECCV 2020 - 16th European Conference Part XXV*. Vol. 12370. Lecture Notes in Computer Science. Springer, 2020, pp. 322–338.

[157] Dong Yin et al. “Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates”. In: *Proceedings of the International Conference on Machine Learning*. Vol. 80. 2018, pp. 5636–5645.

[158] Hongxu Yin et al. “See Through Gradients: Image Batch Recovery via GradInversion”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*. Computer Vision Foundation / IEEE, 2021, pp. 16337–16346.

[159] Hongxu Yin et al. “See Through Gradients: Image Batch Recovery via GradInversion”. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*. Computer Vision Foundation / IEEE, 2021, pp. 16337–16346.

[160] Fisher Yu et al. “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop”. In: *CoRR* abs/1506.03365 (2015).

[161] Mikhail Yurochkin et al. “Bayesian Nonparametric Federated Learning of Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. 2019, pp. 7252–7261.

[162] Syed Zawad et al. “Curse or Redemption? How Data Heterogeneity Affects the Robustness of Federated Learning”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. 2021, pp. 10807–10814.

[163] Yufeng Zhan et al. “A Learning-Based Incentive Mechanism for Federated Learning”. In: *IEEE Internet Things J.* 7.7 (2020), pp. 6360–6368.

[164] Chaoning Zhang et al. “Data-free Universal Adversarial Perturbation and Black-box Attack”. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*. IEEE, 2021, pp. 7848–7857.

[165] Chi Zhang et al. “Generative Gradient Inversion via Over-Parameterized Networks in Federated Learning”. In: *IEEE/CVF International Conference on Computer Vision, ICCV 2023*. IEEE, 2023, pp. 5103–5112.

[166] Hongyi Zhang et al. “mixup: Beyond Empirical Risk Minimization”. In: *6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net, 2018.

[167] Jiale Zhang et al. “PoisonGAN: Generative Poisoning Attacks Against Federated Learning in Edge Computing Systems”. In: *IEEE Internet Things J.* 8.5 (2021), pp. 3310–3322.

[168] Jiale Zhang et al. “Poisoning Attack in Federated Learning using Generative Adversarial Nets”. In: *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 13th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2019*. IEEE, 2019, pp. 374–380.

[169] Jie Zhang et al. “Towards Efficient Data Free Blackbox Adversarial Attack”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*. IEEE, 2022, pp. 15094–15104.

[170] Jingwen Zhang et al. “GAN Enhanced Membership Inference: A Passive Local Attack in Federated Learning”. In: *2020 IEEE International Conference on Communications, ICC 2020*. IEEE, 2020, pp. 1–6.

[171] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 586–595.

[172] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. “iDLG: Improved Deep Leakage from Gradients”. In: *CoRR* abs/2001.02610 (2020).

[173] Lingchen Zhao et al. “Shielding Collaborative Learning: Mitigating Poisoning Attacks Through Client-Side Detection”. In: *IEEE Trans. Dependable Secur. Comput.* 18.5 (2021), pp. 2029–2041.

[174] Yue Zhao et al. “Federated Learning with Non-IID Data”. In: *CoRR* abs/1806.00582 (2018).

[175] Zilong Zhao et al. “Fed-TGAN: Federated Learning Framework for Synthesizing Tabular Data”. In: *CoRR* abs/2108.07927 (2021).

[176] Wenbo Zheng et al. “Federated Meta-Learning for Fraudulent Credit Card Detection”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*. ijcai.org, 2020, pp. 4654–4660.

[177] Junyi Zhu and Matthew B. Blaschko. “R-GAP: Recursive Gradient Attack on Privacy”. In: *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

[178] Ligeng Zhu, Zhijian Liu, and Song Han. “Deep Leakage from Gradients”. In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 2019, pp. 14747–14756.

[179] Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. “Data-Free Knowledge Distillation for Heterogeneous Federated Learning”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12878–12889.

CURRICULUM VITÆ

Jiyue HUANG

05-11-1995 Date of birth in Sichuan, China.

EDUCATION

The following list of education only consists of higher education leading up to this thesis, while omitting primary study, exchange experience, individual courses, and summer schools.

2020–present Ph.D Computer Science
Thesis Title: Detecting Vulnerabilities of Heterogeneous Federated Learning Systems
Delft University of Technology, The Netherlands

2017–2020 M.Phil Computer Science
Thesis Title: Research on Incentive Mechanism of Federated Learning based on Verifiable Client Contribution
Peking University, China

2013–2017 B.Sc Electric Engineering
Thesis Title: Algorithm Design and Implementation for Traffic Sign Detection
Tianjin Univerisity, China

WORK EXPERIENCE

2019 Deep Learning Researcher
Tencent
Shenzhen, China

LIST OF PUBLICATIONS

1. 🕮 **J. Huang**, C. Hong, L.Y. Chen, and S. Roos. "Gradient inversion of federated diffusion models." In *20th International Conference on Availability, Reliability and Security (ARES)*, 2025.

2. C. Hong, **J. Huang**, L.Y. Chen, and S. Roos. "Single-fold Distillation for Diffusion models." In *37th Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, 2025.

3. 🕮 **J. Huang**, L.Y. Chen, and S. Roos. "On Quantifying the Gradient Inversion Risk of Data Reuse in Federated Learning Systems." In *43rd International Symposium on Reliable Distributed Systems (SRDS)*, pp. 235-247, 2024.

4. C. Hong, **J. Huang**, R. Birke, and L.Y. Chen. "Exploring and Exploiting Data-Free Model Stealing." In *35th Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pp. 20-35, 2023.

5. 🕮 **J. Huang**, Z. Zhao, L.Y. Chen, and S. Roos. "Fabricated flips: Poisoning federated learning without data." In *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 274-287, 2023.

6. Z. Zhao*, **J. Huang***, L.Y. Chen. and S. Roos, "Defending against free-riders attacks in distributed generative adversarial networks." In *25th International Conference on Financial Cryptography and Data Security (FC)*, pp. 200-217, 2023.

7. J. Xu, C. Hong, **J. Huang**, L.Y. Chen, and J. Decouchant. "Agic: Approximate gradient inversion attack on federated learning." In *41st International Symposium on Reliable Distributed Systems (SRDS)*, pp. 12-22, 2022.

8. 🕮 **J. Huang**, C. Hong, Y. Liu, L.Y. Chen, and S. Roos. "Maverick matters: Client contribution and selection in federated learning." In *27th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pp. 269-282, 2023.

9. **J. Huang**, R. Talbi, Z. Zhao, S. Bouchenak, L.Y. Chen, and S. Roos. "An exploratory analysis on users' contributions in federated learning." In *second IEEE international conference on trust, privacy and security in intelligent systems and applications (TPS-ISA)*, pp. 20-29, 2020.

🕮 Included in this thesis.

* Equal contribution.


TUDelft