8-8-2023

Towards Privacy Preservation of Federated Learning in Artificial Intelligence of Things

Zuobin Xiong

Towards Privacy Preservation of Federated Learning in Artificial Intelligence of Things

by

Zuobin Xiong

Under the Direction of Wei Li, Ph.D. and Zhipeng Cai, Ph.D.

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2023

ABSTRACT

Under the need of processing huge amounts of data, providing high-quality service, and protecting user privacy in Artificial Intelligence of Things (AIoT), Federated Learning (FL) has been adopted as a promising technique to facilitate its broad applications. Although the importance of developing privacy-preserving FL has attracted lots of attention in different aspects, the existing research is still far from perfect in real applications. In this dissertation, we propose three privacy-related research accordingly towards three realistic weaknesses of federated learning in the AIoT scenarios, which solve the problems of private data inference, private data generation, and private data deletion in different stages of data life. First, to solve the privacy inference problem of traditional FL, we design a dual differentially private FL mechanism to achieve privacy preservation efficiently for both server side and local clients. In particular, our proposed method focuses on FL with non-independent identically distributed (non-i.i.d.) data distribution and gives theoretical analysis on privacy leakage as well as algorithm convergence. The second problem is to generate heterogeneous data privately in FL. To handle this challenging problem, we design a distributed generative model framework that can learn a powerful generator in hierarchical AIoT systems. Thirdly, we investigate a newly emerged machine unlearning problem, which is to remove a data point and its influence from the trained machine learning model with efficiency and effectiveness. Moreover, as the very first work on exact federated machine unlearning in literature, we design a quantization based method, which can remove unlearned data from multiple clients with significantly higher speed-up. All of the proposed methods are evaluated on different datasets, and the results output by our models express superiority over existing baselines.

INDEX WORDS: Federated learning, Privacy preservation, Differential privacy, Generative adversarial networks, Machine unlearning

Towards Privacy Preservation of Federated Learning in Artificial Intelligence of Things

by

Zuobin Xiong

Committee Chair:	Wei Li
Committee:	Zhipeng Cai
	Yingshu Li
	Yanqing Wang

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2023

DEDICATION

This dissertation is dedicated to my parents Shengli Xiong and Hong Wang for their absolute support and endless love. It is impossible for me to finish the five years of Ph.D. study without their love and encouragement.

This dissertation will be defended on June 20th of the year 2023 as the dedicated 8th anniversary gift for my dear wife Hongyue Fu, who has been staying with me in the United States in the past five years, and supporting me in the past eight years with whole heart.

ACKNOWLEDGMENTS

I am blessed to have the precious chance to pursue my Ph.D. degree in the Department of Computer Science at Georgia State University. The past five years I spent in Georgia State University with the lovely people here is a significant condiment in my life and will truly have an impact on the future dish of mine. I would never have been able to accomplish my dissertation without the guidance of my excellent advisors, help of my group members, and support from my friends.

First of all, I would like to show my deepest gratitude to my academic advisors Dr. Wei Li and Dr. Zhipeng Cai. They provide me with an excellent climate for research and give me tons of opportunities to improve myself. They always inspire me when I feel frustrated, and provide valuable suggestion and guidance when I am trapped in my research. Thanks to their help, I can focus on my research and gain a lot of skills on it. Dr. Wei Li and Dr. Cai not only has taught me the thorough methodologies to conduct my academic career, but also inspired me to pursue my life and achieve self-realization. Their hard-working, open-mind, and warm-hearted quality has set a good example for me to follow in my future career and I will always keep the attitudes in my life. It is very grateful and a great honor to have Dr. Yingshu Li and Dr. Yanqing Wang in my committee, who give me great supports for my Ph.D. study and spend valuable time to participate in my dissertation defense.

I also wish to express my gratitude to my former M.S. advisor Dr. Qilong Han, who initially introduced me to the research world of computer science, and provided me an

opportunity to advance myself with Dr. Wei Li and Dr. Zhipeng Cai at Georgia State University. I also appreciate the help and support from Dr. Kejia Zhang, Dr. Lijie Li, Dr. Haiwei Pan, Dr. Dan Lu, and many other professors who kindly helped me in China when I was on the voyage in research and life.

Besides, I have many thanks go to colleagues in my group and in the department, including those who have already graduated: Dr. Yan Huang, Dr. Madhuri Siddula, Dr. Saide Zhu, Dr. Jishen Yang, and Dr. Peng Wang; and those who are still working towards degree: Akshita Maradapu Vera Venkata Sai, Euiseong Ko, Kainan Zhang, Bingyi Xie, Honghui Xu, Chenyu Wang, Guangxi Lu, Jinkun Han, Kaiyang Li, and Yixian Chen. The group meeting and research discussion with them are the important sources of my research energy and the leisure time with them are the important sources of my happy life.

Special thanks for my dear friends Shubin Wu, Anqi Lu, and Shuo Sun, who helped my life in U.S. They shed lights in my heart every time when I feet tired and nostalgic.

Last but not least, it is a pleasure to thank everybody who made this very dissertation possible, as well as express my apologies that I could not mention personally one by one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
IoT	Internet of Things
FL	Federated Learning
SMC	Secure Multi-party Computation
DP	Differential Privacy
GDPR	General Data Protection Regulation
CCPA	California Consumer Privacy Act
MIA	Membership Inference Attack
IID	Independent Identically Distributed

CHAPTER 1
INTRODUCTION

1.1 Background of Federated Learning

The explosive progress and widespread deployment of Internet of Things (IoT) are being leveraged to advanced ubiquitous data sensing and collection across every corner in our life. In addition, with the growing demand for high-quality customized services by IoT users, IoT is desired to be endowed with more powerful learning capacities by Artificial Intelligence (AI) to process the enormous amount of data in data-hungry applications [1, 2, 3, 4]. During the past decade, machine learning has been typically applied in a centralized manner via collecting the generated data to a central server, which delivers excellent performance on real tasks. For example, in the computer vision field [5], image classification and object detection are improved to high accuracy. In natural language processing [6], understanding and analyzing text data makes our work and communication easier, where a popular representative is machine translation software, such as Google Translate. More importantly, some combined machine learning techniques are reshaping our life. Inspired by AlphaGo, Bard, and ChatGPT, machine learning can make better plan or decision for us, and the advancement of self-driving technology may free people from tired driving [7, 8, 9].

Currently, two different modes of deploying machine learning are prevalent, i.e., centralized model vs. federated mode. In a centralized mode, no matter where the data is generated, a centralized server will collect the data, train specific learning models on it, and then provide prediction or query API for applications to use. Although the central-

ized mode is easy to use, accurate in performance, and generalized well, there are many consequences along with, especially from security and privacy perspectives. For instances, (i) since all data and the model are stored in one place, the single point failure threatens the centralized storage and model easily once attackers are able to access the server; (ii) in a large centralized system, there will be too much data upload from devices to the server. The data transmission for such an amount of data brings expensive cost to communication system; (iii) from the view point of data owner, with users' ever-increasing privacy awareness and governments' sophisticated privacy regulations, it becomes harder to encourage users to contribute their valuable private data to central storage for processing. All of these above issues raise unprecedented challenges for machine learning in a centralized mode.

Fortunately, the advent of distributed learning technologies provide us with promising solutions, among which federated learning (FL) [10] is one of the most eye-catching paradigms. In FL, geographically distributed participants collaboratively learn a global model over their local datasets by sharing their local outputs for aggregation, which significantly reduce communication cost (*e.g.*, bandwidth and transmission time) and mitigate privacy leakage from participants' raw data. In concrete, FL differs traditional centralized learning mode in three ways: (i) the FL server only holds a global model instead of the entire dataset, which decreases the data leakage rick of single point failure; (ii) since the structure is changed, the communication between FL server and clients only contains model parameters, where the transmission payload is reduced a lot; and (iii) at local client side, each local client trains their own model on their private local data, preventing privacy leakage for one more step.

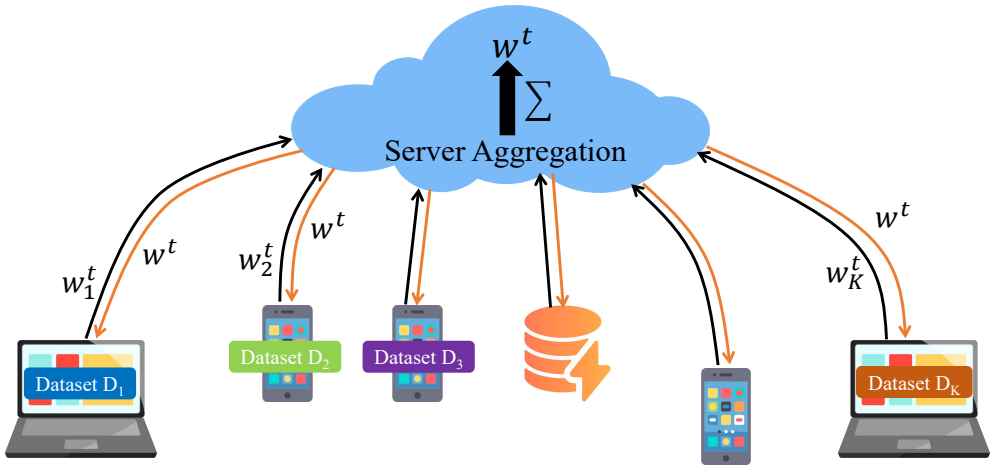


Figure 1.1 The framework of federated learning in AIoT.

Moreover, the presence of FL not only provides a distributed learning solution but also suits Artificial Intelligence of Things (AIoT) perfectly. In an AIoT system, there are large numbers of smart devices with different data distribution [11, 12, 13, 14, 15], which calls a **federated** structure to unit the them for collaboration. Besides, to utilize these IoT data to achieve better performance in applications, the **learning** techniques are necessary. Thus, federated learning and artificial intelligence of things suit each other seamlessly.

1.2 Privacy Issues in Federated Learning

Recently, the machine learning based services and data-driven applications are immersive in our daily life [16, 17], overwhelming from personal devices, government facilities, to even medical agents. Our private data involved in these scenarios, no matter willing or not, is enduring serious privacy leakage [18, 19, 20, 21, 22, 23, 24, 25, 26]. It is under these urgent needs, federated learning is proposed to handle the privacy issues amongst the diverse machine learning and artificial intelligence scenarios. However, though FL separates data

and models with privacy consideration, it still can not achieve perfect privacy protection due to the existence of different attacks in various applications [27, 28]. As pointed out by prior research, FL still suffers malicious attack aiming at breaking system security and privacy [29, 30, 31]. These attacks can be categorized by different levels: (i) at the server side, an honest-but-curious server may play the role of an insider attacker, trying to inference private data and information of local clients; (ii) during the model transmission stage, an outsider attacker can eavesdrop or hijack model parameters, which can be used as prior knowledge to threaten system security and user privacy; and (iii) due to the large numbers of local clients, malicious participants could hide themselves and try to steal private information about the system and user. Therefore, in federated learning, privacy preservation still has a long way to go.

In this dissertation, we mainly focus on three privacy-related research problems in federated learning with consideration of common operations during usage of federated learning, including inference, generation, and deletion. Firstly, we investigate the problem that honest-but-curious server and malicious participants can threaten data privacy during federated learning process. Thus, our proposed approach is to design a private federated learning mechanism that can decrease privacy inference performance. The challenges in this problem are how to identify the privacy leakage and achieve a balance the model utility and data privacy of the proposed mechanism.

Secondly, towards some realistic applications that need data in FL server, we explore how to generated heterogeneous multi-source local data privately at the server side. This research

is caused by variety of AIoT devices and applications, where the data heterogeneity appears in different scenarios. Our approach is to design a federated generative mechanism, allowing server to generate private heterogeneous data from all local clients. Surely, generating data privately for heterogeneous data is the key challenge of this problem.

Thirdly, we consider a newly emerged problem that is to protect privacy of user data during data deletion process in FL. This research corresponds to dynamic property of IoT devices, where devices could dynamically remove their data or quit a system due to different circumstances. In order to handle the problem, our solution is to devise a data removal mechanism through machine unlearning techniques, in which unlearning data in FL setting in exact manner is a big challenge.

1.3 Dissertation Organization

This dissertation is organized as follows. Related works of above topics in federated learning are introduced in Chapter 2, in which we summarize the existing literature in three perspectives, including attack and protection of federated learning, generative model in hetero-generation, and machine unlearning in different settings. Then, we investigate each of the branch for private data inference, private data generation, and private data deletion with proposed methods dual differentially private federated learning, federated hetero-generation, and federated machine unlearning in Chapter 3, Chapter 4, and Chapter 5, respectively. Thereafter, a series of promising future work is proposed towards the security and privacy of federated learning in Chapter 6. Finally, this dissertation is concluded in Chapter 7.

CHAPTER 2
RELATED WORKS

2.1 Attacks and Protection of Federated Learning

2.1.1 Attacks on Federated Learning

As FL has attracted more and more attentions of research and applications, various vulnerabilities of FL models have been explored to launch attacks, mainly including inference attack [32, 33, 29] and poisoning attack [34, 35, 36]. To learn local users' data privacy, Melis *et al.* [32] deveoped membership inference attack by using non-zero gradients of the embedding layer of a deep natural language processing model. A Generative Adversarial Networks (GAN)-based active inference attack was designed by Hitaj *et al.* [33] to generate targeted private samples of the victim client. In [29], authors reviewed the privacy leakage in FL and then developed an inference attack model via using each layer's gradient of the target model. Data poisoning attack of [34] modified the training data through flipping data label and changing features or small regions. In [35], model poisoning attack embedded a global backdoor trigger in FL models, which is achieved by inserting hidden backdoors into a subset of local clients before updating to the server. In [36], by modeling the interactions between training loss and attack performance as an adversarial min-max game, the authors designed model poisoning attack to bypass the poisoning detection tool of FL systems. However, most of the attack models are experiment-oriented and lack theoretical analysis on the attack factors and performance.

2.1.2 Protection of Federated Learning

To protect data privacy in FL, secure multi-party computation (SMC) [37] and differential privacy (DP) [38] are commonly used solutions. Although SMC offers a strong security guarantee, the complicated computation protocols yield potentially un-affordable overheads for small devices, such as mobile devices. Existing works incorporate DP into FL from different aspects [39, 40, 41, 42]. McMahan *et al.* [39] introduced the first DP-based FL proposal for protecting the privacy of a recurrent language model. In [40], an asynchronous FL was designed for mobile edge computing in urban informatics using local differential privacy to protect the privacy of self-driving vehicles. Agarwal *et al.* [41] studied the optimal communication cost with binomial mechanism for FL under certain DP conditions. In [42], DP-based noise was added twice for data privacy in FL – the first time is after training local client models and before updating local model parameters, and the second time is during the process of parameter aggregation. But, all of these current works only focus on FL with i.i.d. scenario.

2.2 Generative Models in Hetero-Generation

2.2.1 Generative Models on Multi-Source Data

Generative adversarial networks (GANs) [43] originally focuses on data generation in a single dataset. In generative model on multi-source data scenarios, two or more data sources are provided to train variants of GANs, which can be partitioned into conditional generation and joint generation. The conditional generation methods aim at learning a conditional dis-

tribution of one data source given other data source(s) as the additional information, which is prevalent in image-to-image translation [44, 45, 7], domain adaptation [46, 47, 48], domain generalization [49, 50, 51], *etc.* Zhu *et al.* [44] design the cycleGAN for image-to-image translation between multiple domains, where each generator takes source domain data as input and outputs conditional data to its target domain. Later, pix2pix [45] is developed as a universal image-to-image translation framework that transfers one image to its desired domain. Xiong *et al.* [7] devise a conditional GAN with privacy mechanism, which can translate original camera data into a privacy-preserving format. StarGAN [47] adopts a bidirectional generator to learn the mapping from source domain to target domain, as well as the inverse direction, which can successfully transfer human face among different attributes. Similar work is concurrently conducted by Li *et al.* [48] who use multiple generators for domain adaptation on each domain. Le *et al.* [52] explore the mutual information between data sources to reduce the space for data generation, therefore improving the fidelity of generated data. In [53, 54], conditional generation is applied on MRI medical data to adapt with multi-contrast image synthesis. Differently, the joint generation methods try to learn the joint distribution of multiple data sources. CoGAN [55] is the first work to learn a joint distribution without any tuple of paired images. As an improvement, Mao and Li [56] use the domain label as a condition and then train only a single conditional generator to represent the joint distribution of two sources without using paired data. Then, RadialGAN [46] is developed to learn the joint distribution of multiple data sources by training multiple autoencoder-based generator on each source, which enhances the data augmentation quality

and solve the missing data issues. Pu *et al.* [57] propose the JointGAN that adopts multiple generators and one discriminator to learn the joint distribution from multiple marginal samples.

However, all the existing generative models on multiple data sources are centralized mode, which limits their applicability on geographically distributed data sources.

2.2.2 Generative Models in Distributed Settings

For generative models in distributed setting, Generative Adversarial Parallelization [58] is the first work applying GANs in a distributed setting, where each GAN model is trained on a distributed dataset. Similar idea is adopted by Hardy *et al.* [59] in Gossip GAN, but the generators and discriminators are gossiped among neighborhood for adapting the global distribution. Durugkar *et al.* [60] and Hardy *et al.* [61] simplify the previous structure via changing multiple generators to one while keeping multiple discriminators for training, which can provide the aggregated discriminator loss to the generator for better generation quality. Along with this line, in [62], authors consider the distributed data generation on multiple datasets, where the multiple discriminators are weighted averaged according to their loss values. Since the federated learning [63] paradigm was proposed, it has been leveraged to develop various GANs [64, 65, 66, 67, 68], in which the minor difference among these methods is whether to aggregate generators [64], discriminators [67], or both [65, 66, 68]. However, a high communication cost for these generative models is transmitting model parameters iteratively. To circumvent, reformed methods in [69] and [70] set a central generator and update it with the loss value of local discriminators, where the data volume transmitted

between server and local clients is much smaller than transferring models. But their solutions are not specified for non-i.i.d data, which lacks of a solid ground for landing a practical application.

These above methods work well on a single data source but are not applicable on multiple heterogeneous data sources.

2.3 Machine Unlearning in Different Settings

2.3.1 Exact Machine Unlearning

The concept of "machine unlearning" was first introduced in [71] and then formally defined by [72], for which existing works can be categorized into two branches, *i.e.*, exact unlearning and approximate unlearning, according to their efficiency and effectiveness.

Exact unlearning requires the distribution of unlearned model should be exactly same as the distribution of model that is retrained on the dataset without the deleted data. Since this requirement is hard to be achieved in complicated models, exact unlearning strategies are mainly designed on simple machine learning systems. Cao and Yang [71] first designed unlearning algorithms for statistical query based machine learning models, such as Naive Bayesian classifier and SVM, where the strategies are used to maintain model statistics at learning stage and update parameter information when unlearning data comes. Then, Ginart *et al.* [72] proposed the first unlearning method for unsupervised learning k-means cluster algorithm, which adopts stability and divide-and-conquer to improve unlearning efficiency. In [73], a SISA framework was designed to reduce unlearning time through sharding,

isolation, slicing, and aggregation, of which the idea is to split a dataset to small parts so that the retraining time is reduced. Similarly, Aldaghri *et al.* [74] adopted ensemble learning to split training dataset into disjoint shards by coding matrix. When unlearning is performed, the unlearned data is removed from coded shards, and the corresponding model is retrained. Following this, [75, 76] conducted unlearning algorithms on random forests algorithm, where they used a similar idea to adjust the structure of decision tree such that the retrained subtree can be minimized. In the above works, the exact unlearning strategies realize the unlearning requirement by retraining a part of models, which can guarantee unlearning effectiveness but consume too many resources, reducing unlearning efficiency.

2.3.2 Approximate Machine Unlearning

On the contrary, approximate unlearning prefers unlearning efficiency to unlearning effectiveness. Different from exact unlearning, the distribution of approximately unlearned model is similar to the distribution of retrained model with much less unlearning time. Instead of retraining a part of model, approximate unlearning methods perform a post-processing on the learned models to obtain an approximation of the fully retrained models. In the current works, there two main methods of processing a model, *i.e.*, gradient based methods [77, 78, 79, 80, 81] and hessian based methods [82, 83, 84, 85]. Wu *et al.* proposed a DeltaGrad [77] algorithm to achieve approximate unlearning by updating the trained model with stored gradients when certain data points are removed. Similarly, in [78], the gradients of each training batch is memorized, and the unlearning method simply removes the parameter updates of the deleted data batch from the learned model. Neel *et al.* [80] and

Ullah *et al.* [81] imported statistical indistinguishability and algorithm stability respectively to unlearn data via gradient descent with provable approximation. To cover the adversarial scenario where a user deliberately deletes data under specific distribution, Gupta *et al.* proposed adaptive machine unlearning that can handle arbitrary model classes and training methodologies. On the other hand, the authors of [82, 85] proposed differentially private data removal mechanisms, which can unlearn data from the learned model by hessian matrix, and Golatkar [83] focused on unlearning a specific class label from deep networks. Considering the computational cost of hessian matrix, Izzo *et al.* [84] found a sublinear algorithm to speed up unlearning from linear models efficiently. In summary, approximate unlearning is faster than exact unlearning but fails to improve model utility, such as accuracy and model difference. So, it is challenging to design an exact and efficient unlearning method as there is still an open question.

CHAPTER 3
PRIVATE DATA INFERENCE IN FEDERATED LEARNING

3.1 Challenges and Contributions

The explosive progress and widespread deployment of Internet of Things (IoT) are being leveraged to advanced ubiquitous data sensing and collection across every corner in our life. In addition, with the growing demand for high-quality customized services by IoT users, IoT is desired to be endowed with more powerful learning capacities by Artificial Intelligence (AI) to process the enormous amount of data in data-hungry applications, such as smart city, smart transportation, and smart healthcare [86, 87, 88]. During past decades, machine learning methods have been typically trained in a centralized manner via collecting all the generated data to a central server, which performs well for accuracy but fail to satisfy the needs of Artificial Intelligence of Things (AIoT) due to its essential flaws: (i) data collection for such an amount of data brings expensive cost to communications; (ii) single point failure threats the centralized storage and model easily once attackers are able to access the server; and (iii) with users' ever-increasing privacy awareness and governments' sophisticated privacy regulations, it becomes harder to encourage users to contribute their valuable private data to central storage and processing. All of these above issues raise unprecedented challenges for machine learning in AIoT – how to effectively and efficiently learn information from massive data without unexpected privacy leakage?

Fortunately, the advent of distributed learning technologies provide us with promising solutions, among which federated learning (FL) [10] is one of the most eye-catching paradigms.

In FL, geographically distributed participants collaboratively learn a global model over their local datasets by sharing their local outputs for aggregation, which significantly reduce communication cost (*e.g.*, bandwidth and transmission time) and mitigate privacy leakage from participants' raw data. However, though FL separates data and models with privacy consideration, it is far away from perfect privacy protection. As pointed out by prior research, FL still suffers malicious attack aiming at stealing private information, such as membership privacy [29], model privacy [30], and attribute privacy [31]. On the other hand, in order to resist the threats of privacy leakage, secure multi-party computation (SMC) [37] and differential privacy (DP) [38, 39, 40, 41, 42, 89] have been widely employed to design various privacy-preserving FL schemes. Nevertheless, the existing works on privacy threats and privacy protection in FL are limited to the impractical assumption that clients' datasets are independent identically distributed (i.i.d.). Until now, only few works on FL consider the non-i.i.d. data, and none of them is related to privacy issue. In fact, the generated data in AIoT usually has highly skewed distribution and even belongs to different data domains; that is, clients are likely to own non-i.i.d. datasets.

For the purpose of better protecting private information, this section intends to fill the gap of investigating privacy-preserving FL under non-i.i.d. scenario. Our research endeavor starts with a comprehensive and deep analysis on the issue of privacy leakage in the original FL system by taking into account both passive and active privacy inference attack. Through our theoretical proof, the performance upper bound of privacy inference in FL is obtained, and the influence of FL scenario (including clients' data size and the difference of data

distribution, *etc.*) on such a performance upper bound is clearly analyzed. Furthermore, a Dual Differentially Private FL (2DP-FL) is elaborately designed to defend privacy inference attack while guaranteeing a convergence upper bound. Particularly, with the flexible noise addition, our 2DP-FL mechanism can meet the different needs for privacy protection. In the real-data experiments, the feasibility of our considered inference attack, the effectiveness of our 2DP-FL mechanism, and the superiority of our 2DP-FL mechanism over the state-of-the-art are well validated. The contributions of this section are summarized as follows:

- To our best knowledge, this is the first work to theoretically analyze privacy leakage in FL with non-i.i.d. setting, in which the upper bound of inferring privacy is obtained.
- To defend privacy inference attack, we propose a DP-based FL mechanism, 2DP-FL.
- Our 2DP-FL mechanism is proved to be effective with a convergence upper bound.
- Intensive experiments are conducted to evaluate our theoretical analysis on privacy leakage as well as the advantages of 2DP-FL mechanism in achieving convergence, protecting privacy, and maintaining model accuracy.

3.2 System Model & Problem Formulation

FL is a distributed learning paradigm that allows geographically distributed participants to follow a common training procedure with the same objective and loss functions to build a federated model on a server using their local datasets. The federated model parameter is learnt by aggregating local participants' model parameters through FedAvg algorithm [10]:

$w_t^f = \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} w_t^k$, where w_t^f and w_t^k are the federated model parameter and client k's model parameter, respectively, n^k is the size of client k's dataset, and $|\mathcal{D}| = \sum_{k=1}^{K} n^k$.

As shown in Fig. 3.1, in this work, we consider that a federated learning model $\mathcal{F}$ is trained for C classes on the dataset (X, Y), where X is the feature space, and $Y = \{1, \dots C\}$ is the set of all class labels. The classifier in FL is reversible, such as liner regression and logistic regression. The goal of federated learning is to obtain an optimized model parameter w_t^f that minimizes the loss function in Eq. (3.1).

$$L(w_t^f) \triangleq \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} \sum_{i=1}^{C} p^k(y=i) \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_t^f)], \tag{3.1}$$

where $\mathcal{F}_i$ denotes the probability of a data point belonging to the i-th class of Y.

It is worth noticing that in this work, all clients' local datasets are non-i.i.d, *i.e.*, D^k is non-i.i.d. For the expected training goal, each client k optimizes his local model parameter w_t^k to minimize the loss function $L^k(w_t^k)$ on the local dataset that follows distribution p^k, *i.e.*,

$$\begin{aligned} L^k(w_t^k) =& \mathbb{E}_{(x,y)\sim p^k}[\sum_{i=1}^{C} \mathbf{1}_{y=i} \log \mathcal{F}_i(x, w_t^k)] \\ =& \sum_{i=1}^{C} p^k(y=i) \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_t^k)]. \end{aligned} \tag{3.2}$$

To obtain the optimal parameter w_t^k, gradient descent-based method is used to solve the

Table 3.1 Summary of main notations used in Chapter 3

Notation	Definition
X	Feature space of data
Y	Label space of data
L	Loss function of federated model
L^k	Loss function of k-th local client model
∇L	Gradient of Loss function L
∇L^k	Gradient of Loss function L^k
p	Global data distribution of X
p^k	Data distribution of k-th client
$\mathcal{F}$	Learning function of model
$\mathcal{F}_i$	i-th digit of the learning function $\mathcal{F}$
K	Number of local clients
D^k	Training dataset on k-th local client
n^k	The number of data in D^k
w_t^f	Model parameter of federated model at time t
w_t^k	Model parameter of local client k at time t
$\tilde{w}_t^f$	DP federated model at t when server noise is 0
$\tilde{w}_t^k$	DP model of client k at t when server noise is 0
$\dot{w}_t^f$	DP federated model in our method
$\dot{w}_t^k$	DP model of client k in our method
N_t^f	DP noise added by server at time t
N_t^k	DP noise added by local client at time t

optimization problem iteratively with the following equation:

$$\begin{aligned} w_t^k &= w_{t-1}^k - \eta \nabla_w L^k(w_{t-1}^k) \\ &= w_t^k - \eta \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_{t-1}^k)], \end{aligned} \tag{3.3}$$

where η is the selected learning rate.

Theoretically, four common assumptions are considered to facilitate performance analysis on FL [90, 42, 91].

1. **Bounded and Unbiased Gradient.** The gradient of each local loss function $\nabla L^k(w)$

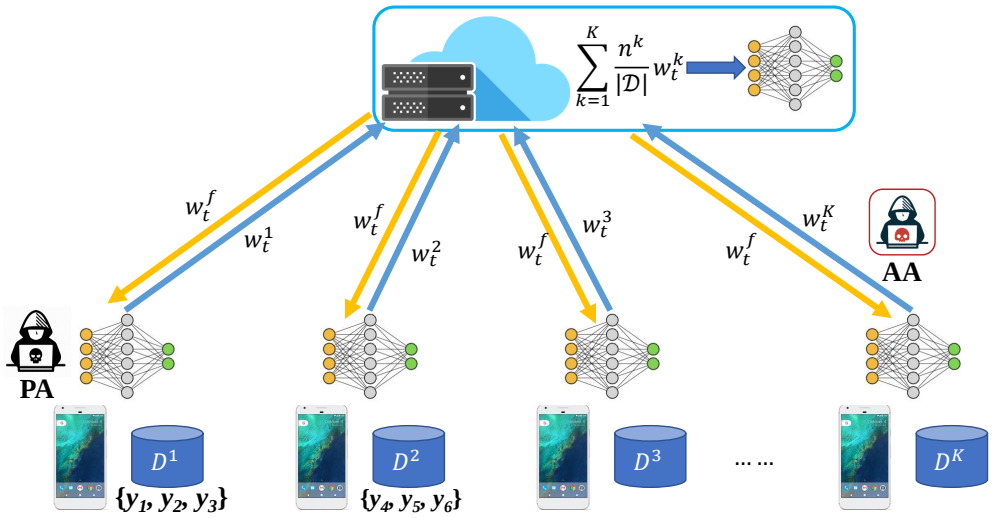


Figure 3.1 The system model of federated learning.

is bounded, and the estimator of global loss function's gradient $\nabla L(w)$ is unbiased:

$$\|\nabla L^k(w)\| \leq G; \nabla L(w) = \mathbb{E}\{\nabla L^k(w)\}. \tag{3.4}$$

2. **Lipschitz Continuity.** The global loss function $L(w)$ is Lipschitz continuous:

$$\|L(w) - L(\bar{w})\| \leq \lambda \|w - \bar{w}\|, \tag{3.5}$$

where λ is Lipschitz constant.

3. **Lipschitz Continuous Gradient.** The gradient of each local loss function $L^k(w)$ is Lipschitz continuous:

$$\|\nabla L^k(w) - \nabla L^k(\bar{w})\| \leq \mu \|w - \bar{w}\|, \tag{3.6}$$

where μ is Lipschitz constant.

4. **Polyak-Lojasiewicz (PL) inequality.** The global loss function $L(w)$ has strong convexity and satisfies Polyak-Lojasiewicz (PL) inequality:

$$\tau(L(w) - L(w^*)) \leq \frac{1}{2}\|\nabla L(w)\|^2, \tag{3.7}$$

where w^* is the optimal model parameter.

3.3 Privacy Leakage of Federated Learning

3.3.1 Privacy Leakage Analysis

Although every client and the server in FL cannot directly access others' local data, private information can still be inferred from the shared model parameters. Especially, at the end of FL, every client holds w_t^f that contains the information about other clients and can be used to infer other clients' privacy via passive attack and/or active attack.

Typically, a classifier $\mathcal{C}$ is represented as a parametric function: $\mathcal{C}(x, w) = y$. If there exists an inverse function, we can compute $x = \mathcal{C}^{-1}(y, w)$. As a result, given the model parameter w and output label y of $\mathcal{C}$, the corresponding input x can be inferred. Under this situation, any client k in FL is able to learn other clients' private information when knowing $\mathcal{C}^{-1}$ and w_t^f. For example, as a type of preimage privacy attack, model inversion attack [92] can recover the input data in FL.

Theorem 1. *Given a classifier $\mathcal{C}$ and an output label y, if $\mathcal{C}$ is reversible and Lipschitz continuous, the distance between the real input x and the inferred input x' has an upper*

bound: $\|x-x'\| \leq \lambda\|w-w'\|$*, where* λ *is the Lipschitz constant,* w *is the real model parameter,* w' *is the parameter used for inference attack.*

Proof. For a reversible discriminative model $\mathcal{C}$, there are $\mathcal{C}(x,w) = y$ and $x = \mathcal{C}^{-1}(y,w)$. Accordingly, the inference result is $x' = \mathcal{C}^{-1}(y,w')$. From the *Lipschitz* continuity of $\mathcal{C}$, we have $\|x - x'\| = \|\mathcal{C}^{-1}(y,w) - \mathcal{C}^{-1}(y,w')\| \leq \lambda\|w - w'\|$. □

Furthermore, Theorem 1 can be extended to a generic attack scenario. Any honest-but-curious client k can infer other clients' data by analyzing w_t^f and/or w_t^k. What's worse, the inference performance of any client k has an upper bound, which is demonstrated in Theorem 2 and Theorem 3.

Theorem 2. *Given* K *clients in federated learning, each client* k*'s local dataset has a size* n^k *and a distribution* p^k*. If* $\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x,w)]$ *is* $\alpha_{x|y=i}$*-Lipschitz for* $\forall i \in Y$*, and each local model parameter* w_{mT}^k *is updated to the server every* m *local iterations, then the distance between the federated model parameter* w_{mT}^f *and any target local model parameter* w_{mT}^j *after* T *updates is upper bounded by Eq.* (3.8)*:*

$$\begin{aligned} \|w_{mT}^f - w_{mT}^j\| \leq \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}[(b^k)^m \|w_{m(T-1)}^j - w_{m(T-1)}^k\| \\ + \eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\|(\sum_{l=0}^{m-1}(b^k)^l g_{max}(w_{mT-1-l}^j))], \end{aligned} \tag{3.8}$$

where $g_{max}(\cdot)$ *is the maximal gradient of model parameter.*

Proof. To prove this theorem, there are two cases for consideration: $m = 1$ and $m > 1$.

When $m = 1$, Eq. (3.9) can be obtained.

$$
\begin{aligned}
&\|w_{mT}^{f} - w_{mT}^{j}\| = \|\sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} w_{mT}^{k} - w_{mT}^{j}\| \qquad (3.9)\\
=&\|\sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}(w_{mT-1}^{k} - \eta \sum_{i=1}^{C} p^k(y=i)\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_{mT-1}^{k})])\\
&- w_{mT-1}^{j} + \eta \sum_{i=1}^{C} p^j(y=i)\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_{mT-1}^{j})]\|\\
\leq&\|\sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} w_{mT-1}^{k} - w_{mT-1}^{j}\|\\
&+ \|\eta \sum_{i=1}^{C} p^j(y=i)\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_{mT-1}^{j})]\\
&- \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}\eta \sum_{i=1}^{C} p^k(y=i)\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w_{mT-1}^{k})]\|.
\end{aligned}
$$

For the first term at the right side of the inequality in Eq. (3.9), we have $\|\sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} w_{mT-1}^{k} - w_{mT-1}^{j}\| = \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}\|w_{mT-1}^{k} - w_{mT-1}^{j}\|$. The second term at the right side of the inequality in

Eq. (3.9) can be rewritten as:

$$
\begin{aligned}
&\|\eta \sum_{i=1}^{C} p^j(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&- \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} \eta \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-1})]\| \\
=&\eta \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} \| \sum_{i=1}^{C} p^j(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&- \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-1})]\| \\
=&\eta \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} \| \sum_{i=1}^{C} p^j(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&- \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&+ \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&- \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-1})]\| \\
\leq&\eta \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} [\sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&+ \sum_{i=1}^{C} p^k(y=i) (\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&- \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-1})])].
\end{aligned}
\tag{3.10}
$$

Since $\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x,w)]$ is $\alpha_{x|y=i}$-Lipschitz for $\forall i \in Y$, Eq. (3.10) can be written as:

$$\begin{aligned}
&\eta \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} [\sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-1})] \\
&+ \sum_{i=1}^{C} p^k(y=i) \alpha_{x|y=i} \|w^j_{mT-1} - w^k_{mT-1}\|] \\
\leq &\eta \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} [\sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| g_{max}(w^j_{mT-1}) \\
&+ \sum_{i=1}^{C} p^k(y=i) \alpha_{x|y=i} \|w^j_{mT-1} - w^k_{mT-1}\|],
\end{aligned} \tag{3.11}$$

where $g_{max}(\cdot)$ is the largest gradient of weight matrix w^j_{mT-1}.

By combining Eq. (3.10) and Eq. (3.11), Eq. (3.9) can be equivalently simplified as Eq. (3.8), *i.e.*,

$$\begin{aligned}
&\|w^f_{mT} - w^j_{mT}\| \\
\leq &\sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|} [(1 + \eta \sum_{i=1}^{C} p^k(y=i) \alpha_{x|y=i}) \|w^j_{mT-1} - w^k_{mT-1}\| \\
+ &\eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| g_{max}(w^j_{mT-1})].
\end{aligned}$$

That is, this theorem holds when m=1.

When $m > 1$, additional analysis is addressed below.

$$
\begin{aligned}
&\|w^j_{mT-1} - w^k_{mT-1}\| \\
=&\|w^j_{mT-2} - \eta \sum_{i=1}^{C} p^j(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-2})] \\
&- w^k_{mT-2} + \eta \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-2})]\| \\
\leq&\|w^j_{mT-2} - w^k_{mT-2}\| \\
&+ \eta \| \sum_{i=1}^{C} p^k(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^k_{mT-2})] \\
&- \sum_{i=1}^{C} p^j(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^j_{mT-2})]\|.
\end{aligned}
$$

Following the same analysis in Eq. (3.11), we get Eq. (3.12).

$$
\begin{aligned}
&\|w^j_{mT-1} - w^k_{mT-1}\| \\
\leq&(1 + \eta \sum_{i=1}^{C} p^k(y=i) \alpha_{x|y=i}) \|w^j_{mT-2} - w^k_{mT-2}\| \\
&+ \eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| g_{max}(w^j_{mT-2}).
\end{aligned} \tag{3.12}
$$

Let $b^k = (1 + \eta \sum_{i=1}^{C} p^k(y=i) \alpha_{x|y=i})$. Since b^k is a constant greater than 1, Eq. (3.12) is

rewritten as:

$$
\begin{aligned}
&\|w_{mT-1}^{j} - w_{mT-1}^{k}\| \qquad (3.13)\\
\leq & b^k \|w_{mT-2}^{j} - w_{mT-2}^{k}\| + \eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| g_{max}(w_{mT-2}^{j}) \\
\leq & (b^k)^2 \|w_{mT-3}^{j} - w_{mT-3}^{k}\| + \\
& \eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| (g_{max}(w_{mT-2}^{j}) + b^k g_{max}(w_{mT-3}^{j})) \\
& \ldots \\
\leq & (b^k)^{m-1} \|w_{m(T-1)}^{j} - w_{m(T-1)}^{k}\| + \\
& \eta \sum_{i=1}^{C} \|p^j(y=i) - p^k(y=i)\| (\sum_{l=0}^{m-2} (b^k)^l g_{max}(w_{mT-2-l}^{j}))
\end{aligned}
$$

By substituting Eq. (3.13) into Eq. (3.8), the theorem holds when $m > 1$. Therefore, the theorem is proved. □

Theorem 2 states that the distance between the federated model parameter w_{mT}^{f} and the target client j's model parameter w_{mT}^{j} is upper bounded by two factors: (i) the difference of data distribution between client j and other clients; and (ii) the maximum gradient value of client j during training. When an honest-but-curious client intends to attack client j using w_t^f, the inference error $\|x - x'\|$ should also be restricted by the above two factors according to Theorem 1. Moreover, this finding can be used to improve privacy protection against attackers' inference when the data distribution is known.

Theorem 3. *Given K clients in federated learning, each client k's local dataset has a size*

n^k and a distribution p^k. If $\nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x,w)]$ is $\alpha_{x|y=i}$-Lipschitz for $\forall i \in Y$, and each local model parameter w^k_{mT} is updated every m local iterations, then the distance between any two local model parameters, w^u_{mT} and w^v_{mT}, after T updates is upper bounded by Eq. (3.14)*:*

$$\begin{aligned} &\|w^u_{mT} - w^v_{mT}\| \leq (b^v)^m \|w^v_{m(T-1)} - w^u_{m(T-1)}\| \\ &+ \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| (\sum_{l=0}^{m-1} (b^v)^l g_{max}(w^u_{mT-1-l})). \end{aligned} \tag{3.14}$$

Proof.

$$\begin{aligned} &\|w^u_{mT} - w^v_{mT}\| \\ =&\|w^u_{mT-1} - \eta \sum_{i=1}^{C} p^u(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^u_{mT-1})] \\ &- w^v_{mT-1} + \eta \sum_{i=1}^{C} p^v(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^v_{mT-1})]\| \\ \leq&\|w^u_{mT-1} - w^v_{mT-1}\| \\ &+ \|\eta \sum_{i=1}^{C} p^v(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^v_{mT-1})] \\ &- \eta \sum_{i=1}^{C} p^u(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^u_{mT-1})]\|. \end{aligned} \tag{3.15}$$

For the second term at the right side of inequality in Eq. (3.15), we have

$$\begin{aligned}
&\|\eta \sum_{i=1}^{C} p^v(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^v_{mT-1})] \\
&- \eta \sum_{i=1}^{C} p^u(y=i) \nabla_w \mathbb{E}_{x|y=i}[\log \mathcal{F}_i(x, w^u_{mT-1})]\| \\
\leq &\eta \| \sum_{i=1}^{C} p^v(y=i) \alpha_{x|y=i} \| w^v_{mT-1} - w^u_{mT-1} \| \\
&+ \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| g_{max}(w^u_{mT-1}).
\end{aligned} \tag{3.16}$$

Combining Eq. (3.15) and Eq. (3.16), we get a new inequality:

$$\begin{aligned}
&\|w^u_{mT} - w^v_{mT}\| \\
\leq &(1 + \eta \sum_{i=1}^{C} p^v(y=i) \alpha_{x|y=i}) \|w^v_{mT-1} - w^u_{mT-1}\| \\
&+ \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| g_{max}(w^u_{mT-1}).
\end{aligned} \tag{3.17}$$

For simplicity, we denote $b^v = (1 + \eta \sum_{i=1}^{C} p^v(y=i)\alpha_{x|y=i})$ and obtain the following result:

$$
\begin{aligned}
&\|w_{mT}^u - w_{mT}^v\| \\
\leq & b^v \|w_{mT-1}^v - w_{mT-1}^u\| + \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| g_{max}(w_{mT-1}^u) \\
\leq & (b^v)^2 \|w_{mT-2}^v - w_{mT-2}^u\| + \\
& \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| (g_{max}(w_{mT-1}^u) + b^k g_{max}(w_{mT-2}^u)) \\
& \ldots \\
\leq & (b^v)^m \|w_{m(T-1)}^v - w_{m(T-1)}^u\| + \\
& \eta \sum_{i=1}^{C} \|p^v(y=i) - p^u(y=i)\| (\sum_{l=0}^{m-1} (b^v)^l g_{max}(w_{mT-1-l}^u)).
\end{aligned}
$$

□

Theorem 3 implies that the distance between two local clients' model parameters, w_{mT}^u and w_{mT}^v, is upper bounded by two factors: (i) the difference of data distribution between clients u and v; (ii) the maximum gradient of the honest-but-curious client u during training process. Accordingly, when client u acts as an attacker using his local model to perform inference attack towards client v, the inference error $\|x - x'\|$ is determined by the above two factors.

To sum up, from Theorem 2 and Theorem 3, inference attack can be implemented to learn privacy in FL under the non-i.i.d. setting, and the attack performance is influenced by the difference of data distribution. Moreover, passive attackers (*e.g.* an honest-but-curious

client k holding w_t^f and w_t^k) can steal preimage privacy with easy implementation, while active attackers (*e.g.*, a malicious client k and an external malicious attacker) can reveal both preimage privacy and membership privacy but requires a more powerful capacity to collect prior knowledge (*e.g.,* a victim's model for white-box attack). More details about the attack scenarios are addressed in the following two subsections.

3.3.2 Passive Attack

To collaboratively train a global model in FL, all clients should achieve some consensus, such as the same model structure, loss function and similar data domain, which can be used as the prior knowledge for an honest-but-curious client to perform passive inference attack. According to Theorem 2 and Theorem 3, the honest-but-curious client only needs to analyze the received global model parameter and/or his local model parameter without tampering training rules or bringing negative impact on learning accuracy.

In the example of passive attack (PA) in Fig. 3.1, client 1 wants to infer the features of $\{y_4, y_5, y_6\}$ of client 2. Ideally, the best way is using w_t^2 to get $x = \mathcal{C}^{-1}(y, w_t^2)$. Unfortunately, it is hard or impossible for an honest-but-curious client in FL to obtain w_t^2. Instead, client 1 uses w_t^1 and/or w_t^f for inference. That is, client 1 learns client 2's private information via $x' = \mathcal{C}^{-1}(y, w_t^1)$ and/or $x' = \mathcal{C}^{-1}(y, w_t^f)$. The attack performance $\|x - x'\|$ is upper bounded by either $\|w_t^1 - w_t^2\|$ or $\|w_t^f - w_t^2\|$ as analyzed in Theorem 1, Theorem 2 and Theorem 3.

3.3.3 Active Attack

Besides the honest-but-curious clients in passive attack, there may be active attackers aiming at stealing privacy from benign clients of FL. An active attacker could be either an external adversary or a participant of FL. Unlike the passive attackers who only hold their own model parameters, the active attackers usually have stronger power to acquire more prior knowledge and resources (*e.g.*, hijacking transmitted parameters, eavesdropping information exchange, and compromising local clients), leading to severe privacy leakage in FL.

As shown in Fig. 3.1, the active attacker (AA) has the ability to access a victim client k's model parameter w_t^k and/or the aggregated model parameter w_t^f. With w_t^k and w_t^f in hand as a white box, the active attacker can launch three kinds of privacy inference attack. (i) **Instance-level membership attack** on D^k with w_t^k and $\mathcal{D}$ with w_t^f, in which the attacker can easily use the target model as a white box to learn whether a specific datapoint x is in the target model's training dataset [93]. (ii) **Model inversion attack** on client k with w_t^k and the entire system with w_t^f, which is a white box attack and is similar to the passive attack in Section 3.3.2. (iii) **Client-level membership attack** on a target client by consistently analyzing w_t^f to infer whether the target client joins the training process of FL or not, which causes serious consequences when the target client holds identity-related data in FL. For example, in a FL system that is trained on mobile phone trajectory datasets, the trajectory data is user-dependent and can be used to infer other private information like sex, address, and occupation, *etc.* [94, 18].

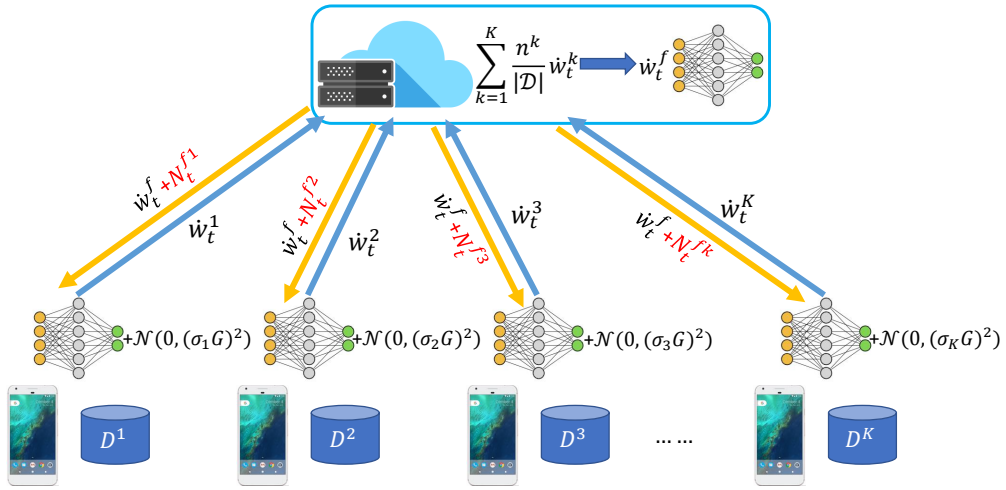


Figure 3.2 The framework of our proposed 2DP-FL mechanism.

3.4 Dual Differentially Private Federated Learning

For passive attack, the root cause of successfully inferring any victim client j' parameter w_t^j from w_t^k and/or w_t^f by any honest-but-curious client k is that the learning model overfits the training dataset. The stronger overfitting, the more accurate inference results. Since DP can introduce randomized noise into training process, extend the generalization capability, and reduce the overfitting [95], it is an effective solution to relieve privacy inference attack. Besides, DP is applicable to defend active inference attack. As active attack is essentially a white-box attack on the victim client's parameter w_t^j and/or the federated model parameter w_t^f, injecting randomized noise into the training process can conceal private information.

In light of the above analysis, we propose a novel mechanism, named "Dual Differentially Private Federated Learning (2DP-FL)", in which DP-based noise is added when training $\dot{w}_t^k$ and before downloading $\dot{w}_t^f$ as illustrated in Fig. 3.2. Adding noise into $\dot{w}_t^k$ can perturb the model parameters to resist data-level membership privacy attack and model inversion

attack, and adding noise into $\dot{w}_t^f$ can defend client-level membership privacy attack and model inversion attack [39, 96].

At the beginning, the initialized global model is distributed to all local clients by setting $w_t^k = \dot{w}_0^f$. Within time slot t, each client k independently trains his own local dataset D^k by minimizing the loss function L^k, in which a random batch data B is picked and the gradient $g(x)$ is calculated based on each data point $x \in D^k$. To bound the gradient contribution of each x, we clip the gradient with a predefined upper bound G, average all gradients in B, and add a scaled gaussian noise, $N_t^k \sim \mathcal{N}(0, (\sigma_k G)^2)$, to achieve DP at the client side. Then, the local model $\dot{w}_t^k$ with DP protection is updated by gradient descent method for each client k. After receiving the local model parameter $\dot{w}_{t+1}^k$ from the selected clients, the server performs FedAvg algorithm to get $\dot{w}_{t+1}^f$. Then, the federated model $\dot{w}_{t+1}^f$ is updated and distributed to all local clients, in which a noise $N_t^{fk} \sim \mathcal{N}(0, \frac{(\sigma S)^2}{U})$ is added into $\dot{w}_t^f$ for each client k. With the perturbed model parameter $\dot{w}_{t+1}^f$, each client k can update $\dot{w}_{t+2}^k$ in time slot $(t+1)$. The pseudo-code of the operations at the clients and the server is described in Algorithm 1.

In the original FL system, $\|w_{mT}^j - w_{mT}^k\|$ is gradually reduced with the increase of T because both w_{mT}^j and w_{mT}^j are trained based on the commonly shared federated model parameter w_{mT}^f, leading to an improved performance of inferring client j's privacy at the side of client k. However, when the server distributed noise N_t^{fk} is equal for all clients, which is similar as most existing works [38, 39], the privacy still can not be protected. On the contrary, in our 2DP-FL system, every client k receives a different perturbed model

Algorithm 1 Dual Differentially Private Federated Learning (2DP-FL)

Require: Total iteration T for FL, number of clients K, selected client U, initialized model $\dot{w}_{t=0}^{f}$

Ensure: 2DP-FL model $\dot{w}_{T}^{f}$

t=0
for $k \in \{1, 2, \dots K\}$ **do**
  $w_t^k = \dot{w}_0^f$
end for
while $t < T$ **do**
  for $k \in \{1, 2, \dots K\}$ **do**
    take a random batch B from D^k with probability $p = \frac{|B|}{|D^k|}$
    compute gradient for each $x \in B$, $g(x) \leftarrow \nabla_w L^k(w_t^k, x)$
    clip gradient, $\bar{g}(x) \leftarrow g(x)/\max(1, \frac{\|g(x)\|_2}{G})$
    add noise, $\tilde{g} \leftarrow \frac{1}{|B|}(\sum_{x \in B} \bar{g}(x) + \mathcal{N}(0, (\sigma_k G)^2)$
    update local model, $\dot{w}_{t+1}^k \leftarrow w_t^k - \eta \tilde{g}$
  end for
  $\dot{w}_{t+1}^f \leftarrow \sum_{k=1}^{K} \frac{|D^k|}{|\mathcal{D}|} \dot{w}_{t+1}^k$
  for $k \in \{1, 2, \dots K\}$ **do**
    $w_{t+1}^k \leftarrow \dot{w}_{t+1}^f + \mathcal{N}(0, \frac{(\sigma S)^2}{U})$
  end for
end while

$(\dot{w}_t^f + N_t^{fk})$ from the server, which is helpful to relieve the reduction of $\|w_{mT}^j - w_{mT}^k\|$ and thus enhance the difficulty of privacy inference. Moreover, the noise addition at the server in 2DP-FL is flexible and can be pre-determined according to the different application requirements. For examples, when $N_t^{fj} = N_t^{fk} \neq 0$ for $j, k \in [1, K]$, a same DP noise is added into the federated model for distribution, which is a common method used in the current works; when $N_t^{fk} = 0$ for $k \in [1, K]$, the DP noise is only added into the clients' local model parameters, and the corresponding federated model is denoted by $\tilde{w}_t^f$ for presentation in the following analysis.

For privacy-preserving FL, the model accuracy is another important concern as adding too much noise into a model would inevitably reduce learning performance. The elegant

design of 2DP-FL lies in the flexible setting of N_t^{fk} that can meet various privacy protection needs with negligible impact on model accuracy, which is analyzed in Theorem 4.

Theorem 4. *In our 2DP-FL mechanism, the difference between $\dot{w}_t^f$ and $\tilde{w}_t^f$ is negligible,* i.e., $\|\tilde{w}_t^f - \dot{w}_t^f\| = 0$.

Proof. From the setting of N_t^k at each client k and N_t^{fk} at the server, we have the following equation:

$$
\begin{aligned}
\|\tilde{w}_t^f - \dot{w}_t^f\| =& \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}(\dot{w}_{t-1}^f - \eta[\nabla_w L^k(\dot{w}_{t-1}^f) + N_t^k]) \\
& - \sum_{k=1}^{K} \frac{n^k}{|\mathcal{D}|}(\dot{w}_{t-1}^f + N_t^{fk} - \eta[\nabla_w L^k(\dot{w}_{t-1}^f + N_t^{fk}) + N_t^k]) \\
=& \mathbb{E}\{\|(\dot{w}_{t-1}^f - \eta[\nabla_w L^k(\dot{w}_{t-1}^f) + N_t^k]) \\
& - (\dot{w}_{t-1}^f + N_t^{fk} - \eta[\nabla_w L^k(\dot{w}_{t-1}^f + N_t^{fk}) + N_t^k])\|\} \\
=& \mathbb{E}\{\|N_t^{fk} + \eta[\nabla_w L^k(\dot{w}_{t-1}^f + N_t^{fk}) - \nabla_w L^k(\dot{w}_{t-1}^f)]\} \\
\overset{1}{\leq}& 0 + \eta\mathbb{E}\{\|\mu \cdot N_t^{fk}\|\} \\
=& 0
\end{aligned}
$$

The inequality 1 holds because N_t^{fk} and N_t^k are both normal distribution noise with mean value 0. □

Theorem 4 tells that with various noise N_t^{fk}, the performance of model $\dot{w}_t^f$ is still nearly the same as that of model $\tilde{w}_t^f$.

Before analyzing the convergence of 2DP-FL, an important conclusion is introduced in Theorem 5.

Theorem 5. *For the federated model $\tilde{w}^f$, $\mathbb{E}\{\|L(\tilde{w}^f_{t+1}) - L(\tilde{w}^f_t)\|\}$ is upper bounded by the following inequality:*

$$\begin{aligned}\mathbb{E}\{\|L(\tilde{w}^f_{t+1}) - L(\tilde{w}^f_t)\|\} \leq & \gamma_1 \mathbb{E}\{\|\nabla L(\tilde{w}^f_t)\|^2\} \\ & + \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}^f_t)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\}.\end{aligned} \tag{3.18}$$

Proof. First, the federated model $\tilde{w}^f_t$ at t-th step can be represented as $\tilde{w}^f_t = \sum_1^K P^k(w^k_t + N^k_t)$, where $P^k = \frac{n^k}{|\mathcal{D}|}$ is the weight of client k, w^k_t is client k's local model parameter without noise, and N^k_t is the injected noise in client k at time t-th step. According to the property of Lipschitz continuous gradient, we have

$$L^k(\tilde{w}^f_{t+1}) \leq L^k(\tilde{w}^f_t) + \nabla L^k(\tilde{w}^f_t)(\tilde{w}^f_{t+1} - \tilde{w}^f_t) + \frac{\mu}{2}\|\tilde{w}^f_{t+1} - \tilde{w}^f_t\|^2.$$

By taking the expectation of both sides, Eq. (3.19) is obtained.

$$\begin{aligned}&\mathbb{E}\{\|L(\tilde{w}^f_{t+1}) - L(\tilde{w}^f_t)\|\} \\ &\leq \mathbb{E}\{\nabla L(\tilde{w}^f_t)(\tilde{w}^f_{t+1} - \tilde{w}^f_t)\} + \frac{\mu}{2}\mathbb{E}\{\|\tilde{w}^f_{t+1} - \tilde{w}^f_t\|^2\},\end{aligned} \tag{3.19}$$

Particularly, $\tilde{w}_{t+1}^f - \tilde{w}_t^f$ can be expressed by:

$$\begin{aligned}\tilde{w}_{t+1}^f - \tilde{w}_t^f =& \sum_1^K P^k(w_{t+1}^k + N_{t+1}^k) - \tilde{w}_t^f \\ =& \sum_1^K P^k(-\eta \nabla L^k(\tilde{w}_t^f)) + N_{t+1},\end{aligned} \tag{3.20}$$

where $N_{t+1} = \sum_{k=1}^K P^k N_{t+1}^k$. Similarly, for $\|\tilde{w}_{t+1}^f - \tilde{w}_t^f\|$, we have

$$\begin{aligned}\|\tilde{w}_{t+1}^f - \tilde{w}_t^f\| =& \|\sum_{k=1}^K P^k(w_{t+1}^k + N_{t+1}^k) - \tilde{w}_t^f\| \\ =& \|\sum_{k=1}^K P^k(w_{t+1}^k - \tilde{w}_t^f) + N_{t+1}\| \\ \leq& \mathbb{E}\{\|w_{t+1}^k - \tilde{w}_t^f\|\} + \|N_{t+1}\| \\ =& \|\eta \nabla L(\tilde{w}_t^f)\| + \|N_{t+1}\|.\end{aligned} \tag{3.21}$$

Then we substitute Eq. (3.20) and Eq. (3.21) into Eq. (3.19) to get the following inequality:

$$\begin{aligned}&\mathbb{E}\{\|L(\tilde{w}_{t+1}^f) - L(\tilde{w}_t^f)\|\} \\ \leq& \mathbb{E}\{\nabla L(\tilde{w}_t^f)(\sum_1^K P^k(-\eta \nabla L^k(\tilde{w}_t^f)) + N_{t+1})\} \\ &+ \frac{\mu}{2}\mathbb{E}\{[\|\eta \nabla L(\tilde{w}_t^f)\| + \|N_{t+1}\|]^2\} \\ =& (-\eta + \frac{\mu\eta^2}{2})\mathbb{E}\{\|\nabla L(\tilde{w}_t^f)\|^2\} + (1 + \mu\eta)\mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| \\ &+ \frac{\mu}{2}\mathbb{E}\{\|N_{t+1}\|^2\}.\end{aligned} \tag{3.22}$$

For simplicity, let $\gamma_1 = -\eta + \frac{\mu\eta^2}{2}$, $\gamma_2 = 1 + \mu\eta$ and $\gamma_3 = \frac{\mu}{2}$. Thus, Eq. (3.22) can be equivalently rewritten as Eq. (3.18):

$$\begin{aligned}\mathbb{E}\{\|L(\tilde{w}_{t+1}^f) - L(\tilde{w}_t^f)\|\} \leq & \gamma_1 \mathbb{E}\{\|\nabla L(\tilde{w}_t^f)\|^2\} \\ & + \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\}.\end{aligned}$$

□

Theorem 6. *The convergence upper bound of our proposed 2DP-FL method after T iterations is given by Eq.* (3.23) *when $\eta \in (0, \frac{2}{\mu}]$, or Eq.* (3.24) *when $\eta \in (\frac{2}{\mu}, +\infty)$.*

$$\mathbb{E}\{\|L(\dot{w}_T^f) - L(w^{f^*})\|\} \leq (1 + 2\tau\gamma_1)^T C_0 + \frac{\gamma_3 \omega^2 T \log\frac{1}{\delta}}{2\tau\gamma_1\epsilon^2}, \tag{3.23}$$

$$\mathbb{E}\{\|L(\dot{w}_T^f) - L(w^{f^*})\|\} \leq (\frac{1}{2\tau} + \gamma_1)G^2 + \frac{\gamma_3 \omega^2 T \log\frac{1}{\delta}}{\epsilon^2}, \tag{3.24}$$

where $C_0 = \|L(\tilde{w}_0^f) - L(w^{f^*})\|$ represents the initialization quality of federated model.

Proof. From Theorem 4, there is

$$\begin{aligned}& \mathbb{E}\{\|L(\dot{w}_T^f) - L(w^{f^*})\|\} \\ =& \mathbb{E}\{\|L(\dot{w}_T^f) - L(\tilde{w}_T^f) + L(\tilde{w}_T^f) - L(w^{f^*})\|\} \\ \leq& \mathbb{E}\{\|L(\tilde{w}_T^f) - L(w^{f^*})\|\}.\end{aligned} \tag{3.25}$$

With Eq. (3.25) and Theorem 5, we can get the following inequality:

$$\mathbb{E}\{\|L(\tilde{w}_{t+1}^f) - L(w^{f^*})\|\} \leq \mathbb{E}\{\|L(\tilde{w}_t^f) - L(w^{f^*})\|\} \tag{3.26}$$
$$+ \gamma_1 \mathbb{E}\{\|\nabla L(\tilde{w}_t^f)\|^2\} + \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\}.$$

When $\eta \in (0, \frac{2}{\mu}]$, $\gamma_1 \leq 0$. According to Polyak-Lojasiewicz inequality, we have

$$\gamma_1 \mathbb{E}\{\|\nabla L(w)\|^2\} \leq 2\tau\gamma_1 \mathbb{E}\{\|L(w) - L(w^*)\|\}. \tag{3.27}$$

The result of substituting Eq. (3.27) into Eq. (3.26) is

$$\begin{aligned}
&\mathbb{E}\{\|L(\tilde{w}_{t+1}^f) - L(w^{f^*})\|\} \qquad (3.28)\\
\leq &\mathbb{E}\{\|L(\tilde{w}_t^f) - L(w^{f^*})\|\} + 2\tau\gamma_1 \mathbb{E}\{\|L(\tilde{w}_t^f) - L(w^{f^*})\|\}\\
&+ \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\}\\
= &(1 + 2\tau\gamma_1)\mathbb{E}\{\|L(\tilde{w}_t^f) - L(w^{f^*})\|\}\\
&+ \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\}\\
\leq &(1 + 2\tau\gamma_1)^2 \mathbb{E}\{\|L(\tilde{w}_t^f) - L(w^{f^*})\|\}\\
&+ (1 + 2\tau\gamma_1)[\gamma_2 \mathbb{E}\{\|N_t\|\}\|\nabla L(\tilde{w}_{t-1}^f)\| + \gamma_3 \mathbb{E}\{\|N_t\|^2\}]\\
&+ \gamma_2 \mathbb{E}\{\|N_{t+1}\|\}\|\nabla L(\tilde{w}_t^f)\| + \gamma_3 \mathbb{E}\{\|N_{t+1}\|^2\},
\end{aligned}$$

where $N_{t+1} = \sum_{k=1}^{K} P^k N_{t+1}^k$. Since the noise N_t follows normal distribution, $\mathbb{E}\{\|N_t\|\} = 0$

and $\mathbb{E}\{\|N_t\|^2\} = \sigma^2$. Therefore, we can rewrite Eq. (3.28) at time T to be

$$\begin{aligned} &\mathbb{E}\{\|L(\tilde{w}_T^f) - L(w^{f^*})\|\} \\ \leq &(1+2\tau\gamma_1)^T \mathbb{E}\{\|L(\tilde{w}_0^f) - L(w^{f^*})\|\} + \gamma_3\sigma^2 \sum_0^{T-1}(1+2\tau\gamma_1)^T \\ \leq &(1+2\tau\gamma_1)^T C_0 + \frac{\gamma_3\sigma^2}{2\tau\gamma_1}. \end{aligned} \tag{3.29}$$

where $\sigma = \omega\sqrt{T\log\frac{1}{\delta}}/\epsilon$ is the noise scale used in moments accountant [97]. Thus, by combining Eq. (3.25) and Eq. (3.29) the convergence upper bound in Eq. (3.23) is proved.

When $\eta \in (\frac{2}{\mu}, +\infty)$, we obtain $\gamma_1 > 0$ and Eq. (3.30).

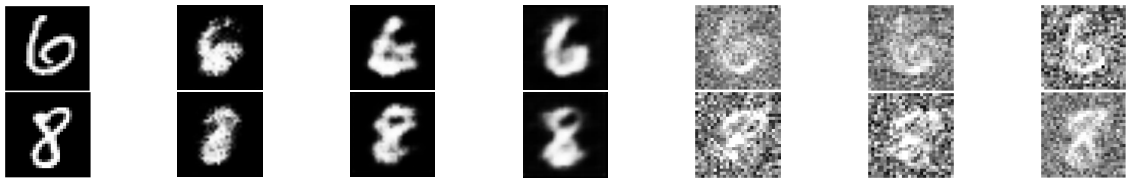
$$\gamma_1 \mathbb{E}\{\|L(w) - L(w^*)\|\} \leq \frac{\gamma_1}{2\tau}\mathbb{E}\{\|\nabla L(w)\|^2\}. \tag{3.30}$$

Substituting Eq. (3.30) and Eq. (3.26) into Eq. (3.25), we can prove the upper bound in Eq. (3.24). □

Notice that in real data experiments, the learning rate η is always set to be a small scalar around 10^{-4} [97], which yields a negative γ_1 and leads to the convergence bound in Eq. (3.23).

3.5 Experiments

In this section, intensive real-data experiments are carried out to validate our analysis on privacy leakage in FL and our 2DP-FL mechanism.



(a) (b) (c) (d) (e) (f) (g)

Figure 3.3 Visual results of attack on MNIST dataset under different scenarios.

MNIST contains 10 classes (*i.e.*, $0-9$) for classification problem and is adopted in our experiments. Considering the FL system under non-i.i.d. setting, we distribute the dataset to different clients according to their class labels, ensuring the assigned local datasets follow different types of distribution. The whole dataset is divided into 15 non-overlapping data buckets and 5 overlapping data buckets, each of which contains data associated with a pre-determined label group. Specifically, for the non-overlapping buckets, there are 15 label groups including $\{0,1\}$, $\{2,3\}$, $\{4,5\}$, $\{6,7\}$, $\{8,9\}$, $\{0\}$, $\{1\}$, ..., $\{9\}$; and for the overlapping buckets, there are 5 label groups including $\{0,1,2,3\}$, $\{2,3,4,5\}$, $\{4,5,6,7\}$, $\{6,7,8,9\}$, and $\{8,9,0,1\}$. Each client can get one or multiple data buckets for local distribution configuration.

The experiments consist of two parts, including privacy leakage analysis and defense performance evaluation. In the analysis of privacy leakage, we show the feasibility of privacy inference attack towards the original FL system by visualizing the data recovery results. To investigate the defense performance of our 2DP-FL mechanism, the convergence, attack accuracy, and classification accuracy are evaluated. Moreover, to the best of our knowledge, the state-of-the-art scheme termed "NbAFL" [42] realizes privacy-preserving FL with the idea similar to our 2DP-FL and is taken as a baseline for performance comparison. All

experiments are performed on a Linux server with Intel(R) Xeon CPU E5-1607, 16 GB memory, and the NVIDIA GeForce RTX 2080 GPU with 11 GB memory, and the common used machine learning library Pytorch, Pysyft and OpenCV are adopted.

3.5.1 Analysis of Privacy Leakage in FL

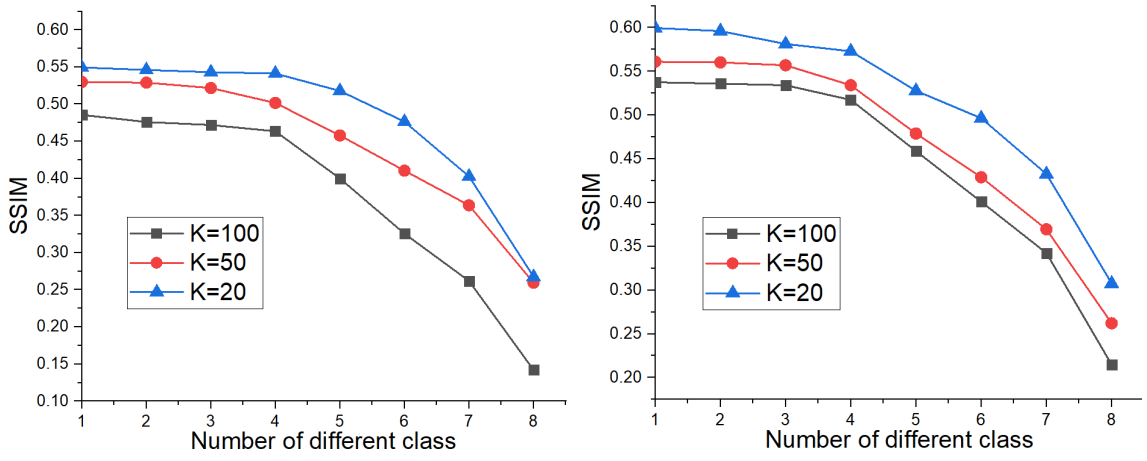
From the analysis in Section 3.3, we know that an honest-but-curious client in FL can work as a passive attacker to infer the privacy of a victim client's data class by using his local model and/or federated model. Taking model inversion attack as a case of attack scenario, such a passive attacker aims to recover an unseen class's common features of the victim client whose dataset has a different distribution. In our experiments, to infer the features of unseen classes in the victim's dataset that holds class labels $\{5, 6, 7, 8, 9\}$, the passive attacker trains his own dataset with class labels $\{0, 1, 2, 3, 4\}$ and implements model inversion attack using his local model parameter w_t^k and/or the federated model parameter w_t^f.

The results of privacy inference attack are visualized in Fig. 3.3. Fig. 3.3a shows the original images with labels 6 and 8 from MNIST dataset. Fig. 3.3b and Fig. 3.3c display the results of passive inference attack with the attacker's local model parameter and the federated model parameter in the original FL, respectively, from which we can see that both attack results expose some feature information of the target classes. Compared with Fig. 3.3b, the recovered images in Fig. 3.3c is closer to the original one. This is because the difference between global data distribution and victim data distribution is more similar, which is in line with our analysis in Section 3.3 – a smaller difference of data distribution leads to a higher upper-bound of attack performance. When the victim's model is used in

active inference attack (*i.e.*, the victim's model is captured and used as a white box), as shown in Fig. 3.3d, the reconstructed image is more clear, demonstrating the feasibility of active attack. Fig. 3.3e, Fig. 3.3f and Fig. 3.3g present the recovered images of passive attack with local model, passive attack with federated model, and active attack under our 2DP-FL mechanism, in which it is hard to perceive that the digit labels of recovered images are 6 and 8. With the protection of our 2DP-FL mechanism, the visual quality of images recovered by model inversion attack is significantly reduced. Moreover, we can see that the results of Fig. 3.3e and Fig. 3.3f are worse than the result of Fig. 3.3g, because active attack uses white-box to perform inference that tends to have smaller model error.

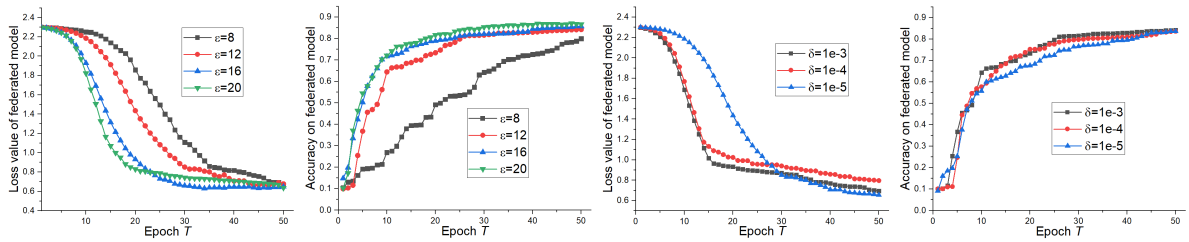
To evaluate the attack performance varying with data distribution, we plot the attack results in Fig. 3.4 by changing the difference of data distribution between the attacker and the victim, where x-axis represents the number of different classes between the attacker's dataset and the victim's dataset, and the y-axis denotes the similarity between the original and the recovered images. More specifically, more different classes between the attacker's dataset and the victim's dataset results in a larger difference between data distribution. The image similarity is measured by Structure Similarity Index Metric (SSIM) with a range of $[0, 1]$, where 0 means totally different and 1 means exactly the same.

Fig. 3.4a depicts the attack performance when the attacker's local model parameter w_t^k is used. According to the Fig. 3.4a, the value of SSIM decreases as the number of different classes is increasing, which is consistent with our theoretical analysis in Section 3.3. Meanwhile, Fig. 3.4a shows the impact of total number of client, K, on privacy leakage in



(a) Passive attack of local model (b) Passive attack on global model

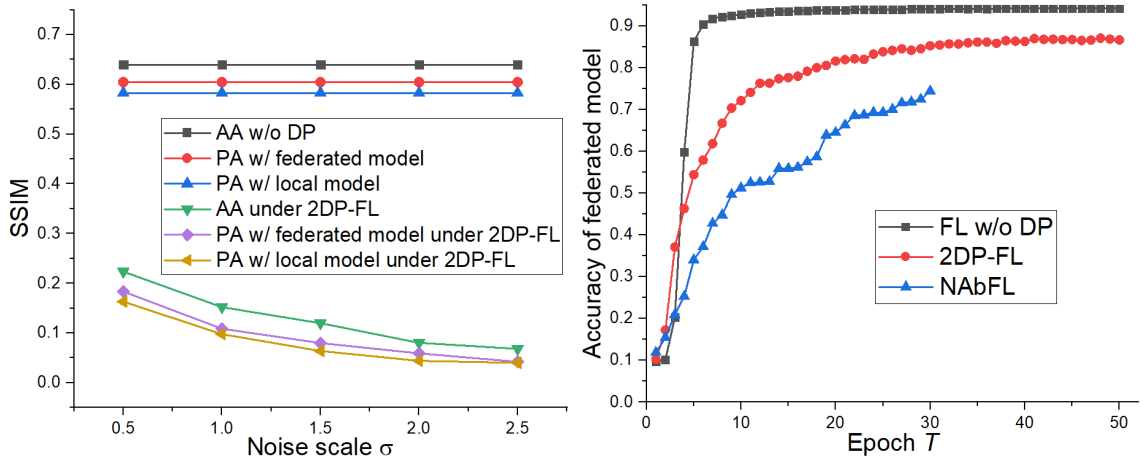
Figure 3.4 Attack performance vs. data distribution difference.



(a) Loss of federated model vs. ϵ (b) Accu. of federated model vs. ϵ (c) Loss of federated model vs. δ (d) Accu. of federated model vs. δ

Figure 3.5 Evaluation on convergence of 2DP-FL mechanism.

FL. As K becomes larger, the amount of each client's private information included in the federated model is reduced as the contribution of each client' local model to the federated model is reduced, mitigating privacy leakage in FL. When the global model parameter w_t^f by the attacker for privacy inference, similar observations can be obtained from Fig. 3.4b.



(a) Attack performance vs. σ (b) Baseline comparison

Figure 3.6 Evaluation on privacy and utility of 2DP-FL mechanism.

3.5.2 2DP-FL Evaluation

According to the analysis in Section 3.4, our proposed 2DP-FL mechanism can defend inference attack while benefiting a good data utility. We design different experiments in this subsection to deeply investigate the performance of 2DP-FL mechanism from the aspects of learning convergence, privacy protection, and data utility.

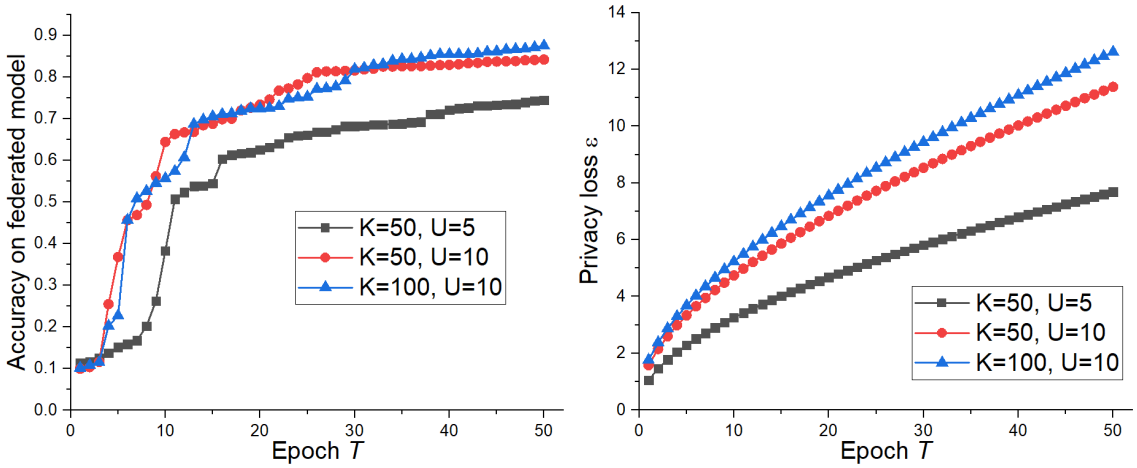
First, to evaluate the convergence of our 2DP-FL mechanism, we make the following settings for Fig. 3.5: the number of clients is K=50, the number of selected clients is U=10, and the number of training epochs is T=50. Additionally, ϵ and δ are the privacy parameters of (ϵ, δ)-DP, where larger ϵ and δ mean less privacy protection.

As shown in Fig. 3.5a, with the increase of T (*i.e.*, the number of training epoch), the decrease of loss value becomes smaller and smaller, gradually reaching a stable loss value, which reflects the convergence trend of 2DP-FL scheme from experimental perspective. Thus

we can conclude that our 2DP-FL mechanism converges when T is large enough, which is consistent with the convergence analysis in Theorem 6. On the other hand, a larger ϵ (*i.e.*, a higher privacy budget) leads to a faster convergence, because higher privacy budget implies weaker privacy protection with less injected noise during training process. Fig. 3.5b also confirms the convergence from the viewpoint of model accuracy, in which the accuracy increases with epoch T and stays stable after a certain threshold. In addition, the impact of δ on convergence is evaluated in Fig. 3.5c and Fig. 3.5d. Similar to ϵ, larger δ results in faster convergence, less privacy protection, and higher accuracy. For example, the loss value of $\delta = 10^{-3}$ shows the fastest decrease and is the first to convergence in Fig. 3.5c as well as reaches the highest accuracy in Fig. 3.5d, because it sacrifices more privacy for maintaining utility. From the results of Fig. 3.5, it also demonstrates that the maintenance of data utility is achieved at the price of privacy protection.

Then, we investigate attack performance in terms of SSIM through changing the noise scale σ of DP. The adopted FL setting is: K=50, U=10, T=50, and δ=0.001. In Fig. 3.6a, AA represents active attack, and PA represents passive attack. When our 2DP-FL is implemented for privacy protection, the attack performance is drastically reduced. The noise scale σ along x-axis denotes the privacy protection level. With the increase of σ, the attack performance is gradually reducing, which shows that our privacy protection works as expected.

Furthermore, we compare the classification accuracy of the original FL without DP, the baseline NbAFL mechanism, and our 2DP-FL mechanism and show the experimental results



(a) Impact of K and U on accuracy

(b) Impact of K and U on privacy

Figure 3.7 The impact of K and U.

in Fig. 3.6b. Especially, we fix $\epsilon = 20$ for NbAFL and 2DP-FL when DP is taken into account. Obviously, the accuracy of the original FL without DP is the best as no noise is added for privacy protection. The accuracy of our 2DP-FL is better than that of NbAFL. In particular, our 2DP-FL becomes convergent within the given ϵ range and achieves an accuracy of 85%. However, for NbAFL, when $T = 31$, the exhausted value of ϵ exceeds the given budget 20 and only gets an accuracy of 74%. From the comparison, we obtain two critical findings: (i) when the number of epochs is the same, 2DP-FL costs a smaller ϵ for better privacy protection; and (ii) when privacy budget is fixed, 2DP-FL can run more epochs for better accuracy. Thus, we can conclude that our 2DP-FL mechanism outperforms NbAFL in terms of classification accuracy and privacy protection.

3.5.3 Impact of K and U

In a FL system, besides privacy related parameters, hyper-parameters such as the number of client K and the number of selected client U also play important roles in fine-tuning the systems. Under our 2DP-FL mechanism, we evaluate the influence of K and U for further investigating hyper-parameter strategies in the FL system. In Fig. 3.7a, when the number of client is fixed at K=50, the federated model with a larger value of U reaches a faster convergence and a higher accuracy as a higher participant radio (represented by U/K) is helpful to enhance the training performance of FL systems; while for the same U (*i.e.*, U=10), a larger of K implies more clients' local datasets are available to the FL system, improving training performance.

The overall privacy loss of the FL system, indicated by ϵ, is exhibited in Fig. 3.7b. When K is equal, a smaller value of U can help reduce privacy loss, because with a smaller value of U, a larger noise is added to the clients' datasets (see line 15 of Algorithm 1). If U remains the same, a larger value of K causes an increased privacy loss. As more clients' local datasets are available for federated learning, it is more possible to reveal privacy of the FL system via data correlation, increasing the risk of privacy loss. In a summary, the learning accuracy and privacy loss of our privacy-preserving FL mechanism 2DP-FL are dependent on the values of K and U, which can be exploited to balance the trade-off between learning accuracy and privacy loss in real applications.

3.6 Summary

For the first time in literature, this chapter rigorously analyzes the issue of privacy leakage and proves the performance upper bound of privacy inference attack in FL with non-i.i.d. data. This analysis motivates us to develop a novel mechanism, 2DP-FL, for preserving private information with ensuring differential privacy. Besides, the noise addition in 2DP-FL can be flexibly set according to different application requirements, and the upper-bounded convergence of 2DP-FL can guarantee its learning performance. Through extensive experiments, the results of our theoretical analysis and the effectiveness of our 2DP-FL mechanism can be confirmed.

CHAPTER 4
PRIVATE DATA GENERATION IN FEDERATED LEARNING

4.1 Challenges and Contributions

Generative models have become a popular research direction in the field of deep learning thanks to its compelling ability to generate realistic data plausibly drawn from an existing data distribution. Different from training classifiers, a generative model can generate unlimited synthetic data to fulfill expected tasks by using the trained generator once it has been trained. So far, the breakthrough brought by generative models has rapidly produced a revolutionary impact on different fields, and this impact has already flourished in various real applications in the Artificial Internet of Things (AIoT). In IoT environments, a variety of devices are interconnected to generate, collect, share, and process heterogeneous data for data-driven applications [98]. With the help of the generative models, the generated data can facilitate the procedures of data collection and process in different aspects, such as reducing tedious data collection time, imputing missing data, augmenting data quality, and detecting abnormal samples. In [99, 100, 101, 102], generative models have been used for traffic data generation, traffic modeling, traffic prediction, and traffic control in smart cities. Generative Adversarial Networks (GANs)-based models have been designed to generate and analyze medical data in smart medication systems [103, 69, 70]. In video surveillance systems, many generative models are designed to support object recognition [104], movement capture [105], anomaly detection [106], and super resolution tasks [107].

Yet, most of the existing works implement the centralized generative models, which first

collect data from IoT devices to a central server and then train generative models to achieve generation goals. These centralized generative models may be vulnerable to the issues of single point failure and privacy leakage [108, 109, 110]. Moreover, users' willingness to share data with a central server may be decline because of their privacy concerns, increasing the difficulty in data collection and hindering the further development of IoT applications [111, 112, 113]. On the other hand, transmitting such a massive amount of data to a central server brings expensive communication cost to IoT. To break down the obstacles of privacy concerns and communication cost, designing distributed generative models should be a better solution. The integration of distributed generative models into IoT can benefit individuals and society in many aspects. For examples, in smart health, distributed generative models can be used to synthesize the pattern of a special tumor's MRI/CT scanning result without privacy violation, where the synthetic data can be used for data visualization and training dataset for other tasks; due to the existence of unaccessible data, pre-training a distributed generative model to collect features from underlying datasets can help model developers understand the dataset, such as checking the data sanity, detecting bias in the dataset, and debugging misclassified samples [67, 114]; and when multiple datasets on IoT devices are non-i.i.d. distributed, learning a distributed generative model can learn a mixed distribution and generate data with more diversity [115].

Currently, only limited works have developed the distributed generative models but overlook the following crucial issues in practical IoT scenarios: (i) most of the existing works adopt the federated learning style that needs to upload large-size model parameters, which

burdens limited network resources; (ii) all of the existing works mainly focus on i.i.d. data for model training without studying non-i.i.d. data scenarios; and (iii) no work considers the heterogeneity of data domain in different IoT devices. Therefore, how to design a distributed generative model to realize data generation effectively and efficiently in IoT is still a challenging problem.

To solve this problem, we design a novel distributed generative model framework, taking into account the essential properties of IoT devices, including wide geographic distribution, low computation power, non-i.i.d. data, and heterogeneous data domains [116, 117, 118]. The hierarchical architecture of our proposed framework contains a cloud layer on the top, an edge layer in the middle, and a local device layer at the bottom, which can offload the computation cost in different layers. With respect to the data distribution and correlation in different IoT devices, the problem of distributed data generation is studied in two scenarios: (i) feature related scenario, where the IoT devices and the edge-side generators are trained to capture homogeneous data distribution; and (ii) label related scenario, where domain condition is used to train the IoT devices and the edge-side generators. The local training results are transmitted to the cloud to global aggregation. The multi-fold contributions of this chapter are summarized as follows.

- Based on the features of IoT applications, we design a three-layer hierarchical framework to deploy distributed generative models, which is the first work to consider multi-source heterogeneous data for distributed data generation, to our best knowledge.

- According to the data scenarios in real IoT applications, we propose two distributed

generative models for multi-source data generation under our proposed hierarchical framework.

- We devise synchronous and asynchronous updating strategies for training the generators on the edges, which can be adopted by different application requirements.
- Intensive experiments are conducted on different datasets from multiple data domains, which can illustrate the performance of our data generation models compared with the state-of-the-arts.

4.2 System Model

As shown in Fig. 4.1, our distributed generative model framework is built as a three-layer hierarchical structure consisting of IoT devices at bottom layer, edge servers in the middle layer, and a cloud server at the top layer. Particularly, each edge server is co-located at a base station covering a local region, in which the edge server and the covered IoT devices form a local community. For the purpose of data generation, a discriminator is deployed in every IoT device, a community generator is deployed in every edge server, and a global generator is deployed in the cloud server. Let $\mathbb{K}$ be the set of local communities, G_k be the community generator of community $k \in \mathbb{K}$, $\mathbb{J}_k$ be the set discriminators in community k, and D_{kj} be the discriminator of IoT device $j \in \mathbb{J}_k$ in community k.

The training process of our distributed generative model consists of two stages, (i) local community training for updating G_k at the edge servers, and (ii) global training for updating G at the cloud server. Specifically, the global training is performed by I_{gl} iterations,

each of which contains I_{lo} iterations of local community training. In each iteration of local community training, with the help of D_{kj} ($j \in \mathbb{J}_k$), G_k is updated in a distributed manner to learn the data distribution of local community k. After I_{lo} iterations of the local community training, the parameters of community generator G_k ($k \in \mathbb{K}$) is sent to the cloud server for constructing G via global aggregation, which completes one iteration of the global training. Then, the aggregated G is distributed to all communities for updating G_k during the local community training thereafter. It is worth mentioning that the real raw data is captured and stored in IoT devices locally without being shared. Therefore, the global generator G can be constructed via the cooperation of edge servers and IoT devices without privacy leakage from the real raw data. Different from most of the existing methods [67, 64, 65, 66, 68] using federated learning to obtain GANs, our local community training structure has a single generator and multiple distributed discriminators. Besides, compared with those federated-style GANs where models' parameters are transmitted between cloud server and local clients, the exchanged information in our local community training process only contains a few batches of data and the loss function value of community generators, which can reduce communication cost and achieve efficient data exchange.

Due to the closely distributed locations, the data within a local community usually has the same features or class labels. While, considering the data distribution and correlation in different communities, the data generation problem in this chapter can be discussed in two types of scenarios: (i) **feature related scenario**, in which the data of different communities has the same features but different labels; and (ii) **label related scenario**, in which the

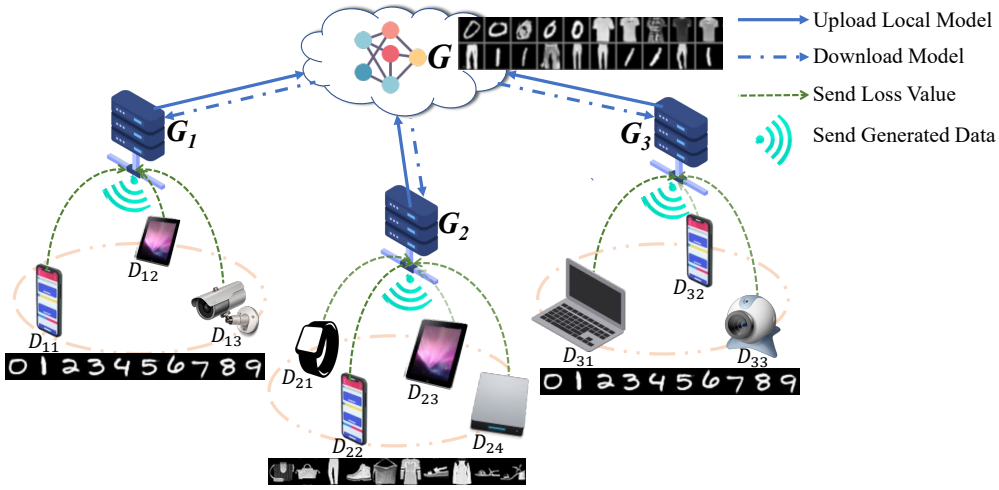


Figure 4.1 An example of the feature related data generation.

data of different communities has the same labels but different features. The details of data generation under these two scenarios are illustrated in Section 4.3 and Section 4.4.

4.3 Feature Related Data Generation

In the feature related scenario, our objective is to learn a mixed data distribution of all communities via the global generator G in the cloud server, where the datasets from different communities have the same features but different labels. As shown in the illustrative example of Fig. 4.1, the three communities have the same feature (*i.e.*, gray scale pixel value in [0, 1]) and different data labels (*i.e.,* digit labels in $\{0, 1, \ldots, 9\}$ and cloth labels in $\{t-shirts, shoes, \ldots, pants\}$). Finally, the global generator in the cloud server can generate the complete dataset with all labels including digits and clothes. This scenario commonly exists in real IoT applications. For example, medical data of a surgical hospital and a cancer hospital may cover different disease classes. Combining the data from these two hospitals through our framework can help generate a complete medical dataset. The design of the

generators and the discriminators, and their interactions is detailed in the following.

4.3.1 Generator

In our framework, there are two types of generators, *i.e.*, the community generator G_k and the global generator G. Similar to the original GANs, the community generator G_k produces two batches of data $G_k(z_d)$ and $G_k(z_g)$ with latent vector z_d and z_g drawn from $\mathcal{N}(0,1)$. Then, these data is sent to local devices, where $G_k(z_d)$ is served as fake data to update D_{kj} as shown in Eq. (4.2), and $G_k(z_g)$ is employed to calculate the loss function value $\mathcal{L}_{D_{kj}}(G_k)$ of G_k on D_{kj} for back propagation as shown in Eq. (4.3).

In the cloud server, the global generator G has the same network structure as G_k, and its parameters are aggregated from all the community generators via Eq. (4.1).

$$G = \sum_{k \in \mathbb{K}} \frac{\exp(\mathcal{L}_{D_*}(G_k))}{\sum_{k' \in \mathbb{K}} \exp(\mathcal{L}_{D_*}(G_k))} G_k, \tag{4.1}$$

where the loss function $\mathcal{L}_{D_*}(G_k)$ can be computed by either $\mathcal{L}_{D_{sync}}$ in Eq. (4.4) or $\mathcal{L}_{D_{async}}$ in Eq. (4.5) as specified in Section 4.3.3.

4.3.2 Discriminator

Corresponding to community generator G_k, in each iteration of the local community training, the discriminator D_{kj} performs two-step operations: (i) discriminator updating and (ii) loss function computation. At the first step, D_{kj} is trained via the IoT device's real local data x that follows distribution $p_{kj}(x)$ and the fake data $G_k(z_d)$ that is generated by G_k, which is

expressed in Eq. (4.2).

$$\begin{aligned}\max_{D_{kj}} \mathcal{L}(D_{kj}) =& \mathbb{E}_{x\sim p_{kj}(x)}[\log D_{kj}(x)]+ \\ & \mathbb{E}_{z_d\sim p_z(z)}[\log(1-D_{kj}(G_k(z_d)))].\end{aligned} \tag{4.2}$$

After D_{kj} is updated, $G_k(z_g)$ is input to D_{kj} to calculate the loss function value of G_k as follows,

$$\mathcal{L}_{D_{kj}}(G_k) = \mathbb{E}_{z_d\sim p_z(z)}[\log(1-D_{kj}(G_k(z_g)))]. \tag{4.3}$$

Next, the loss value $\mathcal{L}_{D_{kj}}(G_k)$ is sent back to G_k for the purpose of calculating gradients and updating G_k.

4.3.3 Updating Strategy

Considering the diversity of IoT devices (*e.g.,* laptops and smart phones) and the requirement of IoT applications, either synchronous or asynchronous methods can be adopted to update G_k during our local community training process.

Synchronous Updating. In the synchronous manner, G_k is updated after collecting the loss value $\mathcal{L}_{D_{kj}}(G_k)$ from all the local discriminators within the community k during each updating period, where the length of a unit updating period can be pre-determined by the system. Unlike traditional aggregation methods [69, 70] that adopt average aggregation of all collected values, our synchronous aggregation can work well when the local distribution $p_{kj}(x)$ is non-i.i.d. The aggregated loss $\mathcal{L}_{D_{sync}}(G_k)$ is computed as follows:

$$\mathcal{L}_{D_{sync}}(G_k) = \sum_{j\in\mathbb{J}_k} \frac{e^{\mathcal{L}_{D_{kj}}(G_k)}}{\sum_{j'\in\mathbb{J}_k} e^{\mathcal{L}_{D_{kj}}(G_k)}} \mathcal{L}_{D_{kj}}(G_k). \tag{4.4}$$

In Eq. (4.4), a larger value of $\mathcal{L}_{D_{kj}}(G_k)$ means the generated data $G_k(z_d)$ is more realistic as judged by D_{kj}, thus a bigger weight is assigned to $\mathcal{L}_{D_{kj}}(G_k)$ for aggregation. The effectiveness of such an exponential weighted aggregation method has been verified in the existing works [60, 62] on non-i.i.d. data. After $\mathcal{L}_{D_{sync}}(G_k)$ is obtained, the community generator G_k can be updated accordingly through the gradient ascending method.

Asynchronous Updating. Since the IoT devices may have different capacities, such as computation rate and transmission power, there might be some stragglers who reduce time efficiency of generator updating process if the synchronous method is adopted. To overcome the weakness of the synchronous method, we propose an asynchronous aggregation method to update G_k.

Basically, in the asynchronous manner, G_k is updated immediately once a loss function value is received from a local discriminator so that there is no need to wait the slowest discriminators for updating. Notice that with respect to G_k, the received loss function value with a smaller degree of staleness has more contritions to accurate updating. Via taking into account the impact of reception time on the updating performance, the "staleness" of $\mathcal{L}_{D_{kj}}(G_k)$ is defined to be $s_{kj} = T_{kj}^{rec} - T_k^{lat}$, where T_{kj}^{rec} is the time when $\mathcal{L}_{D_{kj}}(G_k)$ is received by G_k, and T_k^{lat} is the latest updating time of G_k. Accordingly, G_k can be updated with obtained loss function values asynchronously through Eq. (4.5).

$$\mathcal{L}_{D_{async}}(G_k) = \frac{1}{|\mathbb{M}_k|} \sum_{j \in \mathbb{M}_k} e^{-s_{kj}} \mathcal{L}_{D_{kj}}(G_k), \tag{4.5}$$

where $\mathbb{M}_k$ is the set of local IoT devices whose loss function values are received by G_k at the same time. As indicated in Eq. (4.5), a smaller degree of staleness is assigned a bigger

Algorithm 2 Feature Related Data Generation

Require: the size of $\mathbb{K}$, the size of $\mathbb{J}_k$ in each community k, the number of global iterations I_{gl}, and the number of local iterations I_{lo}

Ensure: the global generator G Initialize G, G_k, and D_{kj} with $j \in \mathbb{J}_k$

1: **for** $t = 0$ to I_{gl} **do** {global training}
2: **for** community $k \in \mathbb{K}$ in parallel **do**
3: $G_k \leftarrow$ **Local_Community_Training**($\mathbb{J}_k$, I_{lo})
4: **end for**
5: Aggregate G based on Eq. (4.1)
6: Distribute G back to communities
7: **end for**
8: **Local_Community_Training**($\mathbb{J}_k$, I_{lo}):
9: **for** $t = 0$ to I_{lo} **do** {local training}
10: Generate $G_k(z_d)$ and $G_k(z_g)$ and send to D_{kj} for $j \in \mathbb{J}_k$
11: **for** $j \in \mathbb{J}_k$ **do**
12: Update D_{kj} via Eq. (4.2)
13: Calculate loss value $\mathcal{L}_{D_{kj}}(G_k)$ based on Eq. (4.3)
14: **end for**
15: **if** Synchronous updating method is adopted **then**
16: Update G_k via Eq. (4.4)
17: **end if**
18: **if** Asynchronous updating method is adopted **then**
19: Update G_k via Eq. (4.5)
20: **end if**
21: **end for**
22: **return** G

weight on the aggregated loss value for updating G_k.

4.3.4 Training Process

In this subsection, we present the whole picture of the training process of distributed data generation in the feature related scenario. At the beginning of the global training process, the parameters of G, G_k, and all the discriminators are initialized with Xavier [119]. In each global training iteration, G_k is updated via I_{lo} iterations of the local community training as shown in lines 9-17 of Algorithm 2 and then is sent to the cloud server for aggregation.

Specifically, in each community k, G_k generates two batches of data and sends them to the discriminators within the same local community. The discriminators update weight parameters based on $G_k(z_d)$ and calculate the loss values based on $G_k(z_g)$ as feedback to G_k as described in Section 4.3.2. After this step, the loss value $\mathcal{L}_{D_{kj}}(G_k)$ is sent to the edge servers for updating G_k, where the aggregated loss can be $\mathcal{L}_{D_{sync}}$ in the synchronous updating method or $\mathcal{L}_{D_{async}}$ in the asynchronous updating method, which can be pre-determined by system or application requirements.

After local community training stops, G_k is sent to the cloud server for the global aggregation that follows Eq. (4.1). At the end of each global training iteration, G is distributed back to local communities, preparing for the next iteration of global training process. After I_{gl} iterations of the global training process, G is output for feature related data generation. As shown in Algorithm 2, the value of I_{gl} determines the iterations of updating G, and the value of I_{lo} decides the iterations of updating D_{kj} and G_k within each global training iteration. Obviously, when I_{gl} (*resp.* I_{lo}) is increased, the training result of G (*resp.* G_k) will become better with a longer training duration. Therefore, I_{gl} and I_{lo} can be set by considering the trade-off between training result and training time.

4.4 Label Related Data Generation

In the label related scenario, the distributed datasets in different communities have different feature distribution but the same labels; that is, the data in different communities belongs to different domains. Mathematically, we can formulate the problem of data generation in the

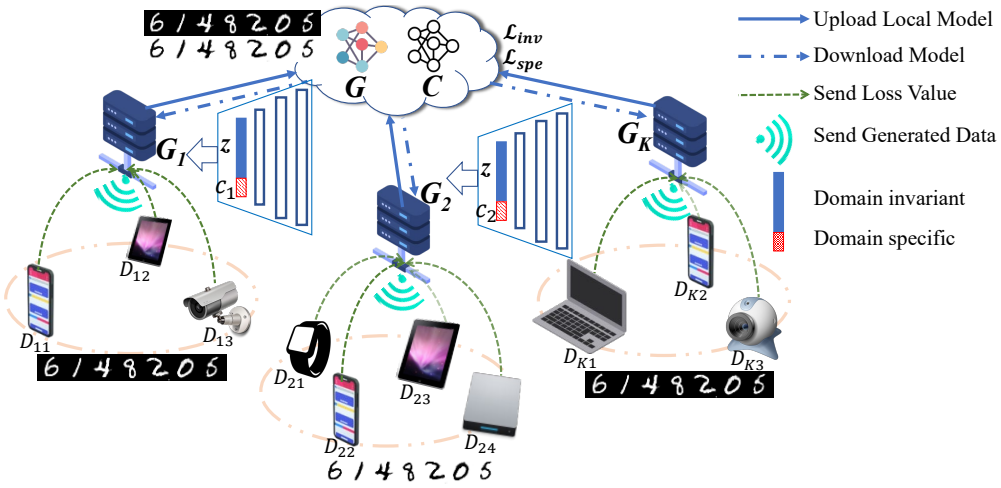


Figure 4.2 An example of the label related data generation.

label related scenario as the problem of multi-domain data generation, which is similar to domain adaptation problem [120]. In vehicular networks, camera data captured by vehicles in one community and LiDAR data in another community may have the same object labels [8], which is similar to our label related scenario.

Take Fig. 4.2 as an example, where the data in the three communities shares the same digit labels $\{0, 1, 2, \ldots, 9\}$ but differs in data features, such as content style, background color, and luminance. The global generator G in the cloud can generate data from different domains at the same time. Therefor, for the label related scenario, our goal is to train a global generator G in the cloud server to produce various data from different data domains. To this end, a domain classifier C is added in the cloud server to distinguish the original source of the generated data under our framework as presented in Fig. 4.2. In the following sections, details of label related generation will be introduced, including generators, discriminators, domain information, and training process.

4.4.1 Generator

In the label related scenario, the generator is expected to generate multi-domain data distribution under various domain conditions, where in each community k, the domain condition, denoted by c_k, indicates the origin of the generated data. To achieve this purpose, the model of cGANs [121] is adopted to embed the latent vector z as the domain invariant of all data domains (*e.g.,* the common class labels in $\{0, 1, 2, \ldots, 9\}$) and to embed the domain condition c_k as the domain specific of each data domain (*e.g.,* content style, background color, and luminance).

Specifically, the community generator G_k takes the latent vector z and the domain specific condition c_k as its inputs for data generation. During the local community training process, two batches of fake data, $G_k(z_d, c_k)$ and $G_k(z_g, c_k)$, are generated by G_k and sent to D_{kj} to update the discriminator and to calculate the loss function value $\mathcal{L}_{D_{kj}}(G_k)$ for back propagation.

In the cloud server, the global generator G has the same structure as G_k and is also aggregated by community generators as shown in Eq. (4.1). Meanwhile, the domain classifier C can help G separate the generated data space according to the domain conditions c_k. We defer the details of domain invariant, domain specific, and domain classifier to Section 4.4.3.

4.4.2 Discriminator

The discriminators are set with two major operations, including discriminator update and loss function computation.

For discriminator update, D_{kj} is trained by the real data, $x \sim p_{kj}(x)$, and the generated data, $G_k(z_d, c_k)$, through Eq. (4.6), which is the same as the training process of cGANs

$$\begin{aligned}\max_{D_{kj}} \mathcal{L}(D_{kj}) =& \mathbb{E}_{x\sim p_{kj}(x)}[\log D_{kj}(x|c_k)]+ \\ & \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{kj}(G_k(z_d, c_k)|c_k))].\end{aligned} \tag{4.6}$$

After D_{kj} is updated, the loss function value of G_k, $\mathcal{L}_{D_{kj}}(G_k)$, is calculated based on $G_k(z_g, c_k)$ as follows,

$$\mathcal{L}_{D_{kj}}(G_k) = \mathbb{E}_{z\sim p_z(z)}[\log(1 - D_{kj}(G_k(z_g, c_k)|c_k))]. \tag{4.7}$$

Then, $\mathcal{L}_{D_{kj}}(G_k)$ is sent back to edge server to update G_k.

Remark. The updating process of G_k in the label related scenario can be performed using either synchronous or asynchronous updating methods as described in Section 4.3.3.

4.4.3 Domain Invariant and Specific

The same class labels of all communities' data is so-called "domain invariant" that normally exists in high-level representation space [56, 122]. For a generator that maps latent vectors to high-dimension space, the output of its first layer denotes its high-level representation. The distance between the first-layer output of G under two different domain conditions can be defined as the "domain invariant loss", which can guide the generator to produce the required high-level representation. Through minimizing this distance, the generated result $G(z, c_k)$ with different domain conditions are forced to possess the same information with respect to z (*i.e.*, the same labels).

Since the global generator G is obtained on cloud server, with the domain condition c_k ($k \in \mathbb{K}$), G can generate $G(z, c_k)$ falling into different data domains. Accordingly, we can obtain the corresponding first-layer output, $G^{1st}(z, c_k)$, to calculate the domain invariant loss via Eq. (4.8), where L_2 norm is adopted to measure the difference in the high-level representation space.

$$\mathcal{L}_{inv}(G) = \sum_{k \neq k' \in \mathbb{K}} \|G^{1st}(z, c_k) - G^{1st}(z, c_{k'})\|_2. \tag{4.8}$$

Thus, minimizing $\mathcal{L}_{inv}(G)$ is able to push G to decode similar high-level representations under different domain conditions, which can help embed domain invariant information into the latent vector z.

Domain specific is the special information that belongs to each domain. Here, the different domain condition c_k is used to represent domain specific. With the domain condition c_k, the generated data $G(z, c_k)$ is actually self-labeled, and thus can be used as the training dataset to fulfill a supervised domain classifier training. The domain classifier C in the cloud server is modeled as a multi-class classifier that predicts the generated data to corresponding data domains. Concretely, C is constructed as a neural network with considering cross entropy loss and is trained on the labeled dataset $\{G(z, c_k), c_k | k \in \mathbb{K}\}$, where c_k is used as the domain label. During the evaluation phase of C, we can use the loss value of C as the domain specific loss of G, which is computed in Eq. (4.9):

$$\mathcal{L}_{spe}(G) = \sum_{k \in \mathbb{K}} H(C(G(z, c_k)), c_k), \tag{4.9}$$

where $\mathcal{L}_{spe}(G)$ denotes the domain specific loss, and $H(\cdot, \cdot)$ represnts the cross entropy mea-

Algorithm 3 Label Related Data Generation

Require: the size of $\mathbb{K}$, the size of $\mathbb{J}_k$ in each community k, the number of global iterations I_{gl}, and the number of local iterations I_{lo}

Ensure: the global generator G Initialize G, C, G_k, and D_{kj} with $j \in \mathbb{J}_k$

for $t = 0$ to I_{gl} **do** {global training}
  for community $k \in \mathbb{K}$ in parallel **do**
    $G_k \leftarrow$ **Local_Community_Training**($\mathbb{J}_k$, I_{lo})
  end for
  Aggregate G based on Eq. (4.1)
  Train the domain classifier C on $\{G(z, c_k), c_k | k \in \mathbb{K}\}$
  Calculate $\mathcal{L}_{inv}(G)$ and $\mathcal{L}_{spe}(G)$ with C
  Distribute G,$\mathcal{L}_{inv}(G)$, and $\mathcal{L}_{spe}(G)$ back to communities
end for
Local_Community_Training($\mathbb{J}_k$, I_{lo}):
for $t = 0$ to I_{lo} **do** {local training}
  if $t \neq 0$ **then**
    Update G_k by minimizing: $\lambda_1 \mathcal{L}_{inv}(G) + \lambda_2 \mathcal{L}_{spe}(G)$;
  end if
  Generate $G_k(z_d, c_k)$ and $G_k(z_d, c_k)$ and send to D_{kj}
  for $j \in \mathbb{J}_k$ **do**
    Update D_{kj} via Eq. (4.6)
    Calculate loss value $\mathcal{L}_{D_{kj}}(G_k)$ based on Eq. (4.7)
  end for
  if Synchronous updating method is adopted **then**
    Update G_k via Eq. (4.4)
  end if
  if Asynchronous updating method is adopted **then**
    Update G_k via Eq. (4.5)
  end if
end for
return G

surement. Since G is expected to produce more realistic data based on the domain condition c_k, minimizing $\mathcal{L}_{spe}(G)$ can force G to embed domain specific semantics into c_k.

4.4.4 Training Process

The training process of distributed data generation in the label related scenario is shown

in Algorithm 3. Similar to Algorithm 2 in the feature related scenario, Algorithm 3 starts with the initialization of model parameters. In each global training iteration, all local communities proceed their local community training to update G_k with I_{lo} iterations (see lines 11-21). During each location training iteration, two batches of generated data are sent to the local IoT devices to update their discriminators and calculate loss function values. Then, the calculated loss function values are transmitted to the edge servers for updating G_k based on Eq. (4.4) in a synchronous manner or Eq. (4.5) in an asynchronous manner, which can be decided by the system and/or application requirements.

After G_k is trained and received by the cloud server, a weighted aggregation is performed to obtain G as shown in line 5 of Algorithm 3. Then, according to the description in Section 4.4.3, we can train the domain classifier C based on the dataset generated by G with different domain conditions, and calculate the domain invariant loss $\mathcal{L}_{inv}(G)$ and domain specific loss $\mathcal{L}_{spe}(G)$. Thereafter, G, $\mathcal{L}_{inv}(G)$, and $\mathcal{L}_{spe}(G)$ are sent back to community generators for local training iteration. Particularly, in line 13 of Algorithm 3, $\lambda_1 \in (0, 1]$ and $\lambda_2 \in (0, 1]$ are two pre-determined hyperparameters used as the weights of loss functions. When the global training process ends, G is output to generate multi-domain data with the domain condition c_k. In Algorithm 3, through the settings of I_{gl} and I_{lo}, we can adjust the trade-off between training result and training time for G and G_k.

4.5 Experiments

In this section, we deeply investigate the quality of the generated data from both visualization and statistic aspects through the comprehensive comparison with the state-of-the-art models.

4.5.1 Experiment Settings

System Structure. The network environment is simulated on one device, where each community has 1 edge server and 5 IoT devices. Considering the available real datasets, the number of community varies with the scenario settings, which is demonstrated as follows.

Datasets. For the feature related scenario, we choose two types data in the experiments: (i) simulated gaussian mixed data, where the datasets of all local communities have the same feature (*i.e.,* variance) but different mean values; and (ii) MNIST dataset [123] and Fashion-MNIST dataset [124], where both datasets have the same features but different class labels. For the label related scenario, we select three pairs of datasets, each of which contains the same labels or semantic information but belongs to different domains: (i) MNIST dataset and inverse MNIST dataset; (ii) sketch-photo handbag dataset [125]; and (iii) sketch-photo shoe dataset [126].

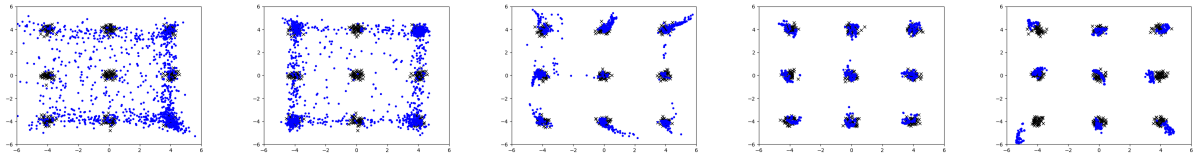
Data Distribution. Both the i.i.d. data distribution and the non-i.i.d. data distribution are considered for the feature related and the label related scenarios in our experiments. Accordingly, by combining the two types of data distribution and the two scenarios, we have the following four types of settings for evaluation:

(i) Feature related scenario with i.i.d. data. For the simulated gaussian mixed datasets, we set 9 communities as shown in Fig. 4.3, and each cluster of gaussian data is randomly assigned to a community. For the MNIST dataset and the Fashion-MNIST dataset, we set 3 communities, where we assign the Fashion-MNIST dataset randomly to one community, and split the MNIST dataset randomly to the remaining two communities. In this way, the datasets of the three communities have the same features but different labels.

(ii) Feature related scenario with non-i.i.d. data. Here, only the MNIST and Fashion-MNIST datasets are used because class labels are needed as indicator to split dataset into different devices such that the data distribution on local devices within each community is non-i.i.d. There are 3 communities, where Fashion-MNIST dataset is assigned to one community in non-i.i.d. manner, and MNIST dataset is assigned to the other two communities with non-i.i.d. partition. As a result, the datasets of the 5 IoT devices in each community following non-i.i.d. distribution.

(iii) Label related scenario with i.i.d. data. We set 3 communities in the whole network. For each of the three pairs of datasets, we assign one dataset randomly to the 5 IoT devices of one community, and split the other dataset randomly to 10 IoT devices of the remaining two communities.

(iv) Label related scenario with non-i.i.d. data. In this case, only the MNIST and inverse MNIST datasets are adopted because they are labeled datasets. We set 3 communities with MNIST dataset assigned to a community and inverse MNIST dataset allocated to the remaining two communities in non-i.i.d. manner, so that the data distribution on IoT devices



(a) Centralized GAN. (b) Federated GAN. (c) Multi-discriminator GAN. (d) FR data generation (sync). (e) FR data generation (async).

Figure 4.3 Generated data of different generative models on the simulated gaussian mixed dataset.

within community is non-i.i.d.

Baselines. Due to the different ideas and goals, we pick different baselines for feature related and the label related scenarios. In the feature related scenario, centralized GAN model, federated GAN (Fed GAN) [65], and multi-discriminator GAN (M-D GAN) [69] are used for performance comparison. In the label related scenario, RegGAN [56] and federated GAN are selected as the baseline models.

4.5.2 Feature Related Data Generation

Feature Related Scenario with I.I.D. Data. In this setting, data of different communities has the same features, and data inside the community is i.i.d. Fig. 4.3 displays the generated data of centralized GAN, Fed GAN, M-D GAN, and our feature related (FR) data generation model with synchronous and asynchronous updating methods, in which the black "×" markers denote the training data points coming from 9 communities, and the blue "·" markers are the generated data points. Thus, if the blue "·" can concentrate to the black "×" points closer, the generated data is more similar to the real data, indicating the better performance of generative models.

(a) Centralized GAN. (b) Federated GAN. (c) Multi-discriminator GAN. (d) FR data generation (sync). (e) FR data generation (async).

Figure 4.4 Generated data of different generative models on MNIST dataset and Fashion-MNIST dataset in i.i.d. setting.

In Fig. 4.3a, the generated data mainly concentrates around one black data cluster at the bottom right corner, which means only partial data distribution of entire training dataset is learned by centralized GAN. Centralized GAN cannot perform well on such gaussian mixed data due to mode collapse of original GAN model. Fed GAN averagely aggregates all the involved generators and discriminators on the server, producing the data that mainly concentrates around the black data clusters at the four corners. M-D GAN uses one generator to receive feedbacks from all discriminators for updating, which is a good way to learn the multi-source data distribution. But, as shown in Fig. 4.3c, it also generates a number of outliers that are far away from the black data points, because the discriminators work alone without information exchange among all communities. Our FR data generation model is like a combination of Fed GAN and M-D GAN, within each community, the GAN structure resembles to M-D GAN but uses an exponentially weighted aggregation; and in the cloud server, all community generators are exponentially weighted to obtain the global generator G so that the information among communities can be exchanged through distributing G back

to communities. From Fig. 4.3d, we find that almost all the generated data can concentrate on the 9 black data clusters accurately, reflecting the best generation performance when the synchronous updating method is used in our FR data generation model. While in Fig. 4.3e, when the asynchronous updating method is implemented, the majority of the generated data can concentrate on the 9 black data clusters closely, reaching a slightly worse generation performance than our FR data generation model with synchronous update but is still better than other baselines.

For experiments on MNIST and Fashion-MNIST, one community holds Fashion-MNIST data and the other two hold MNIST data. Since the generated data is expected to serve as training data in off-line/downstream applications, the utility of generative models can be evaluated by classification accuracy on its generated data. Other metrics for quality evaluation are also adopted, including Inception Score (IS) [127] and Frechet Inception Distance (FID) [128], which is common data generation tasks. The generated images are presented in Fig. 4.4, and the quantitative results are displayed in Table 4.1.

From the data of Fig. 4.4, centralized GAN only generates blurred partial Fashion-MNIST data, failing to generate data from the whole dataset because of model collapse. Fed GAN generates almost all labels, but some data is blurred and unclear, such as the 3rd and 5th rows in Fig. 4.4b, because the average aggregation strategy used in federated GAN may not able to collect all important feedbacks from local entities. Fig. 4.4c and Fig. 4.4d show that M-D GAN and our FR data generation model with synchronous update present similar results, since the structure of M-D GAN is similar to our local community design.

Table 4.1 Quantitative comparison of different models on MNIST and Fashion-MNIST in feature related scenario.

Setting	Metrics	Cent. GAN	Fed. GAN	M-D GAN	FR (sync)	FR (async)
i.i.d.	IS	3.386	4.215	4.228	**4.363**	3.208
	FID	19.84	17.14	14.69	**13.82**	52.24
	Accu.	82.21%	87.84%	90.60%	**90.65%**	53.17%
non-i.i.d	IS	3.386	2.281	2.839	**3.875**	2.922
	FID	19.84	74.80	96.12	**26.15**	77.18
	Accu.	82.21%	42.37%	45.22%	**86.05%**	51.20%

More details of performance comparison can be found from the statistics in Table. 4.1, which reflects our observation about Fig. 4.4. It can be seen that with respect to IS (the higher the better), FID (the lower the better), and accuracy (the higher the better), our FR data generation model with synchronous update outperforms the remaining four models.

Feature Related Scenario with non-I.I.D. Data. The generation results are depicted in Fig. 4.5 and the statistics are shown in Table 4.1. Centralized GAN in Fig. 4.5 and Fig. 4.4 outputs the same results since there is no difference for a centralized mode with i.i.d. and non-i.i.d. data. By comparing the results of the three distributed generation models in Fig. 4.4, Fig. 4.5, and Table 4.1, it is clear that our FR generation model with synchronous update can achieve the best generation performance with both i.i.d. and non-i.i.d. data. Especially, the performance improvement of our FR generation model with synchronous update is more significant under non-i.i.d. setting. The success of our generation model is credited to the following novel designs: (i) the three-layer hierarchical framework, where the community generator G_k in each community only needs to handle homogeneous data for a better generation performance; and (ii) the exponential parameter aggregation, which takes

(a) Centralized GAN. (b) Federated GAN. (c) Multi-discriminator GAN. (d) FR data generation (sync). (e) FR data generation (async).

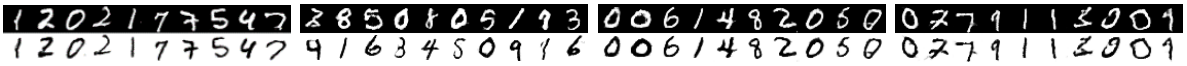
Figure 4.5 Generated data of different generative models on MNIST dataset and Fashion-MNIST dataset in non-i.i.d. setting.

the advantage of the community generators to build a powerful global generator to produce heterogeneous data from all communities with good data quality.

4.5.3 Label Related Data Generation

For the data generation problem in the label related scenario, this paper is the first one to propose distributed generative models, so there does not exist any comparable models. Instead, we build a distributed baseline model following the idea of federated GAN [65] for comparison. The structure of federated GAN (Fed. GAN) contains a cloud server and five local IoT devices. During each training iteration, each local device trains a GAN model and upload its generator and discriminator to the cloud server for aggregation. Then, the aggregated generator and discriminator in cloud server are distributed back to all IoT devices for the next training iteration.

Label Related Scenario with I.I.D. Data. The generated data of the MNIST, inverse MNIST datasets and the sketch-photo datasets are shown in Fig. 4.6 and Fig. 4.7, respectively.



(a) RegGAN. (b) Federated GAN. (c) LR data generation (sync). (d) LR data generation (async).

Figure 4.6 Generated data of different generative models on MNIST dataset and inverse MNIST dataset in i.i.d. setting.

(a) RegGAN. (b) Federated GAN. (c) LR data generation (sync). (d) LR data generation (async).

(e) RegGAN. (f) Federated GAN. (g) LR data generation (sync). (h) LR data generation (async).

Figure 4.7 Generated data of different generative models on (sketch-photo) Handbag dataset and Shoe dataset in i.i.d. setting.

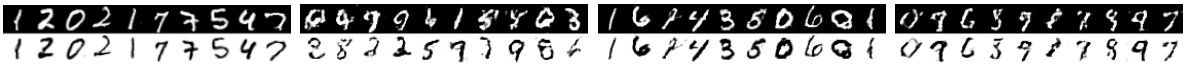
RegGAN outputs the best generated results compared with the remaining three distributed models because it is a centralized model. Among these three distributed models, the results of Fed. GAN and our LR data generation model with synchronous update have similar visual quality and statistical performance, which are clear and close to the generated results of RegGAN. On the other hand, our LR data generation model with asynchronous update performs a little bit worse than Fed. GAN and our LR data generation model with synchronous update. Notably, the significant difference between our LR data generation models with different update methods and Fed. GAN is whether the generated data in the same column has the same domain invariant. In Fig. 4.6a, Fig. 4.6c, and Fig. 4.6d, the generated data in each column always has the same class label. The reason that our LR data

Table 4.2 Quantitative comparison of different models on MNIST and Inverse MNIST in label related scenario.

	Setting	**i.i.d.**			**non-i.i.d.**		
Metrics	RegGAN	Fed. GAN	LR (sync)	LR (async)	Fed. GAN	LR (sync)	LR (async)
IS	**4.037**	3.360	3.971	3.041	2.651	3.495	2.452
FID	**26.11**	33.20	28.56	41.35	53.26	45.22	59.82

generation models can produce the same class label is that we use the domain classifier to minimize the domain invariant loss, which embeds the class label information into a domain invariant vector z. But, in Fig. 4.6b, the domain invariant (*i.e.*, class label) of two images in the same column are not identical. Fed. GAN does not deploy any domain classifiers in the cloud server except the parameter aggregation operation, and thus it can not guarantee the same class label with z during data generation. Also, for the generated data quality of our LR data generation model with different updating strategies, the synchronous method performs better than the asynchronous method, which is the same as the generation performance in the feature related scenario. Similar results on Handbag and Shoes datasets can be found in Fig. 4.7, where the generated data of our LR data generation model with different update methods have the same semantic information in each column, but the generated data of Fed. GAN does not. More statistical results about the generated data are presented in Table 4.2.

Label Related Scenario with non-I.I.D. Data. The experiments of label related data generation with non-i.i.d. data are only conducted on the MNIST dataset and the inverse MNIST dataset. The results of RegGAN in Fig. 4.6 and Fig. 4.8 are the same since there is no difference for a centralized mode with i.i.d. and non-i.i.d. data. The generated



(a) RegGAN. (b) Federated GAN. (c) LR data Generation (sync). (d) LR data Generation (async).

Figure 4.8 Generated data of different generative models on MNIST dataset and inverse MNIST dataset in non-i.i.d. setting.

results of Fed. GAN and our LR data generation models are shown in Fig. 4.8. Compared with Fed. GAN, our LR data generation model in the synchronous manner can still produce clear numbers from two different domains as shown in Fig. 4.8c, which is closer to the results of RegGAN and is better than the results of Fed. GAN in Fig. 4.8b. Similar to the results in i.i.d. setting, the generated data quality of our LR data generation model with asynchronous update is still a little bit worse. But from the view point of domain invariant, our LR data generation model with different update methods can produce the same labels in each column. Yet, in Fig. 4.8b, the generated data of Fed. GAN in the same column still has different labels, which demonstrate the effectiveness of the domain invariant loss function in our LR data generation models. The above observations validate that our LR data generation model work well on non-i.i.d. data with different update methods. In addition, more quantitative results of label related data generation in non-i.i.d. setting are provided in Table 4.2, where the data quality of our LR data generation model with synchronous update is not only better than Fed. GAN, but also very close to the performance of RegGAN even in such non-i.i.d. setting.

4.6 Summary

In this chapter, our aim is to solve the problem of distributed data generation with multiple heterogeneous data sources in IoT. To this end, we design a hierarchical distributed generative framework with the consideration of IoT features. Based on this framework, feature related data generation model and label related data generation model are proposed, which can solve the above data generation problems successfully in a synchronous manner or an asynchronous manner. Experiments are conducted on multiple datasets under different distribution settings to evaluate the performance of our models from both visualization and statistic aspects, which demonstrates the excellence of our models compared with the start-of-art baselines.

CHAPTER 5
PRIVATE DATA DELETION IN FEDERATED LEARNING

5.1 Challenges and Motivations

As well known, to advance the performance of machine learning models, a sufficient amount of data is indispensable to be collected from users and/or third parties. For examples, popular computer vision models are trained on images and videos posted by Facebook and Flickr users [129], many natural language processing models are trained on Amazon reviews [130], and micro-video recommendation systems are trained on Tiktok user data [131]. In a number of real applications, users provide their data to the service providers/platforms for model development in exchange of better service quality. Meanwhile, to protect users' data privacy, the "Right To Be Forgotten" is enforced by some regulations, such as General Data Protection Regulation (GDPR) [132] and California Consumer Privacy Act (CCPA) [133]. For instance, a user wants to delete part of search history, and a hospital requests to remove some patients' data. Thus, a practical and crucial question is that when users request to remove data from the services or platforms, what should the service providers do?

A straightforward method to deal with users' data removal requests is to delete the users' data from the databases. However, due to the memorization of machine learning models [134], the information of training data is memorized in model parameters and cannot be forgotten easily. Moreover, such naive data deletion can be explored by malicious attackers to infer users' private information in various ways, including model inversion attack [92], membership inference attack [33], reconstruction attack [135], *etc.* Therefore, how to correctly and

completely remove users' data from the learned models has become a challenging problem.

The rightful data removal in machine learning context, termed as "machine unlearning" [71], requires deleting data from training datasets as well as the impact of data in the learned models. Intuitively, retraining the machine learning models from scratch on the remaining databases sans the deleted data is a simple way to achieve unlearning objective, but full computation cost of retraining may not be affordable, especially on the models with millions of parameters [136]. Thus, designing a computation-efficient and time-saving unlearning method is the core focus of current machine unlearning works. So far, there are only a few works on machine unlearning but with different limitations, such as simple learning methods (linear regression [82, 84]), model-dependent methods (decision tree [76] and k-means cluster [72]). Besides, existing research on federated learning (FL) mainly focuses on improving unlearning efficiency with approximate unlearning but overlooks model utility (*e.g.,* model accuracy) after unlearning, which harms the performance of unlearned models.

Inspired by the limitations of existing unlearning methods, in this paper, we aim at designing an exact and efficient federated machine unlearning method in model-agnostic manner. First, to enable exact federated unlearning, we utilize the idea of α-quantization [137] to improve the stability of federated model and propose our quantized federated learning (Q-FL) algorithm, through which the quantized federated model can maintain unchanged before and after data deletion. More importantly, the quantized federated model and the model retrained from scratch on the remaining databases could be the same with a high probability, so that there is no need to retrain from scratch as long as the stability is held.

Based on the quantized federated model, we design an exact and efficient federated unlearning (Exact-Fun) algorithm that also can achieve decent unlearning efficiency and good model utility after unlearning. We highlight the contributions of this chapter as follows:

- To the best of our knowledge, this work is the first work to investigate the **exact** machine unlearning problem in the federated learning paradigm, which can be extended to different machine learning models (model agnostic).
- The quantized federated unlearning (Q-FL) algorithm is designed to enable exact federated unlearning with the guarantee of model convergence.
- The exact and efficient federated unlearning (Exact-Fun) algorithm is proposed to process users' data deletion requests with proved unlearning efficiency.
- A Newton's method based local model unlearning mechanism is developed in our approximate federated unlearning (Appro-Fun) algorithm, which is used for efficient and approximate data deletion.
- For the proposed Appro-Fun algorithm, we prove the indistinguishability between the unlearned model and the retrained model as well as the performance guarantee of the unlearned model.
- Both our Exact-Fun and Appro-Fun algorithms are evaluated on real datasets via intensive experiments, which validate the effectiveness and efficiency of our proposed algorithms compared with state-of-the-art.

Table 5.1 Summary of main notations used in Chapter 5

Notation	Meaning
$\mathcal{D}$	the training dataset of entire federated system
$\mathcal{D}_k$	the training dataset of client k
$\mathcal{U}_j$	the unlearning dataset submitted by client j
$\mathcal{D}^u$	$\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$
w^t	the federated model at iteration t
w_k^t	the local model of client k at iteration t
$\hat{w}^t$	the quantized federated model at iteration t
$\hat{w}^{u,t}$	the quantized federated model at iteration t after unlearning
$L(\cdot)$	the loss function of federated model
$L_k(\cdot)$	the loss function of client k's model
$l(\cdot,(x,y))$	the loss function of on a data instance (x,y)

This chapter is organized as follows. The system model is introduced in Section 5.2 and the problem of federated unlearning is formulated in Section 5.3. In Section 5.4 and Section 5.5, we detail our methodologies of exact federated unlearning and approximate unlearning, respectively. Then, our proposed algorithms are and evaluated in Section 5.6. Finally, we give a summary in Section 5.7.

5.2 System Model

In this section, we introduce the original framework of federated learning and present necessary notations in Table 5.1.

As an advanced distributed learning paradigm, FL allows a set of distributed clients $\mathcal{K} = \{1, 2, \ldots, K\}$ to collaboratively learn a global model on the federated server using their own local dataset $\mathcal{D}_k$ ($k \in \mathcal{K}$). In $\mathcal{D}_k$, each data instance is represented by (x, y), where $x \in \mathcal{X}$, $\mathcal{X}$ is the feature space of training data, $y \in \mathcal{Y}$, and $\mathcal{Y}$ is the set of ground truth labels. In a federated learning system, all the clients' local databases together form a global database $\mathcal{D} = \bigcup_{k \in \mathcal{K}} \mathcal{D}_k$. During each training iteration t, the goal of each local client k in

the system is to minimize a loss function as shown in Eq. (5.1).

$$L_k(w_k^t) = \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} l(w_k^t, (x, y)), \tag{5.1}$$

where L_k is the loss function of client k, w_k^t is the model parameter of client k at iteration t, $|\mathcal{D}_k|$ is the size of $\mathcal{D}_k$, and $l(w_k^t, (x, y))$ is the loss of model w_k^t on instance (x, y). Then, the clients' local models are uploaded to the server for aggregation, and the federated model parameter w^t of iteration t is calculated via FedAvg algorithm [10].

$$w^t = \sum_{k\in\mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k^t. \tag{5.2}$$

According to the loss function of local clients and the aggregation algorithm, the optimization objective of federated learning system can be formulated as Eq. (5.3).

$$\min_{w^t\in\mathcal{W}} L(w^t) = \sum_{k\in\mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} L_k(w^t), \tag{5.3}$$

where $\mathcal{W} \in \mathbf{R}^d$ is the d-dimension hypothesis space of model parameters. In a nutshell, a federated learning algorithm can be defined to be $\mathscr{A} : \mathcal{D} \to \mathcal{W}$, which takes $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$ as the input and outputs the federated model parameter w^t belonging to $\mathcal{W}$ as depicted in Fig. 5.1 and Algorithm 4.

In this chapter, we make assumptions on the considered federated learning as existing works, which can facilitate our analysis.

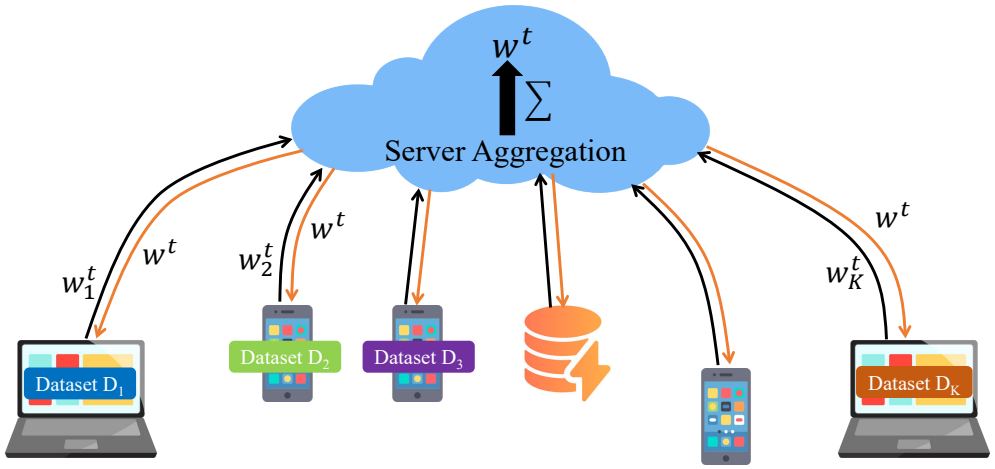


Figure 5.1 The framework of federated learning and unlearning.

Algorithm 4 Federated Learning Algorithm (FedAvg)

Input: the number of iteration T, the number of clients K, the size of minibatch B, learning rate η

Output: federated model $w = \xi^T$

Server executes:
initialize ξ^0
for iteration $t = 0$ to T **do**
  for client $k \in \mathcal{K}$ in parallel **do**
    $\xi_k^{t+1} \leftarrow \text{ClientUpdate}(k, \xi^t)$
  end for
  $\xi^{t+1} \leftarrow \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \xi_k^{t+1}$
end for
return $w = \xi^T$
Clients execute:
  ClientUpdate(k, ξ^t): // run on each client
    compute gradient $\nabla L_k(\xi^t)$ on $\mathcal{D}_k$
    update local model $\xi_k^{t+1} \leftarrow \xi^t - \eta \nabla L_k(\xi^t)$
    upload model ξ_k^{t+1} to server

1. **(Bounded and Unbiased Gradient)** $\forall w \in \mathcal{W}$, the stochastic gradient $\nabla L_k(w)$ has an upper bound G and is an unbiased estimator of federated loss function's gradient

$\nabla L(w)$ [90, 138]:

$$\|\nabla L_k(w)\| \leq G, \tag{5.4}$$

$$\nabla L(w) = \mathbb{E}\{\nabla L_k(w)\}. \tag{5.5}$$

2. **(Lipschitz Continuous Gradient)** $\forall w, w' \in \mathcal{W}$, the gradient of the loss function $L_k(\cdot)$ is Lipschitz continuous with $\mu > 0$ [42, 138]:

$$\|\nabla L_k(w) - \nabla L_k(w')\| \leq \mu \|w - w'\|. \tag{5.6}$$

3. **(Lipschitz Continuity)** $\forall w, w' \in \mathcal{W}$, the loss function $l(\cdot, (x, y))$ is Lipschitz continuous with $\iota > 0$ [42, 90]:

$$\|l(w, (x, y)) - l(w', (x, y))\| \leq \iota \|w - w'\|. \tag{5.7}$$

4. **(Strong Convexity)** $\forall w, w' \in \mathcal{W}$, the loss function $l(\cdot, (x, y))$ is strongly convex with $\tau > 0$ [42, 138]:

$$\begin{aligned} l(w, (x, y)) \geq l(w', (x, y)) &+ \nabla l(w', (x, y))^\top (w - w') \\ &+ \frac{\tau}{2} \|w - w'\|^2, \end{aligned} \tag{5.8}$$

where $\top$ is the transpose operation.

5. **(Hessian-Lipschitz)** The loss function $l(w,(x,y))$ is Hessian Lipschitz with constant M:

$$\|\nabla^3 l(w)\| \leq M, \forall w. \tag{5.9}$$

These assumptions are practical for common loss functions such as mean square error and cross entropy loss in liner models.

5.3 Problem Formulation

The problem of exact federated unlearning is mathematically formulated in Section 5.3. After a FL model is trained on the given training dataset, the model parameters are fixed and can be deployed for use in applications. When client $j \in \mathcal{K}$ would like to erase his/her data $\mathcal{U}_j \subset \mathcal{D}_j$ ($|\mathcal{U}_j| = m < |\mathcal{D}_j|$) from the trained federated model, he/she could submit an unlearning request to the server. Particularly, the clients hold disjoint private datasets locally, so the unlearned data submitted by each one is different. Besides the federated model, the federated learning algorithm $\mathscr{A}$ produces a set of meta-data $\mathcal{M}$ that is not necessarily used during prediction but useful in the unlearning procedure for computing gradients and intermediate results and other purposes. Accordingly, an unlearning algorithm can be defined as $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \to \mathcal{W}$, which takes the trained model $\mathscr{A}(\mathcal{D})$, the unlearning dataset $\mathcal{U}_j$, and the meta-data $\mathcal{M}$ as the inputs to update an unlearned model.

To process an unlearning request, $\mathcal{U}_j$ should be deleted from $\mathcal{D}_j$ (and $\mathcal{D}$), and the influence of $\mathcal{U}_j$ should be revoked from the trained federated model. Moreover, a successful exact unlearning algorithm should guarantee: (i) the unlearning cost (*e.g.,* time and computation) is less than the cost of training from scratch on the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$; and (ii) the distribution of unlearned model parameter is the same as the distribution of model parameters trained from scratch on $\mathcal{D}^u$. In this work, we focus on exact federated unlearning and approximate federated unlearning, and give their definition as follows.

Definition 1. (Exact Federated Unlearning). *Given an FL algorithm* $\mathscr{A} : \mathcal{D} \to \mathcal{W}$ *with clients set* $\mathcal{K}$*, and an unlearning request* $\mathcal{U}_j$*,* $j \in \mathcal{K}$*, the unlearning algorithm* $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \to \mathcal{W}$ *can exactly unlearn* $\mathcal{U}_j$ *from* $\mathscr{A}(\mathcal{D})$ *if*

$$\Pr[\mathscr{A}^u(\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \in \mathcal{W}] = \Pr[\mathscr{A}(\mathcal{D}^u) \in \mathcal{W}].$$

The definition means that the probability distributions of unlearned model and the re-trained model are equal.

Moreover, the definition of approximate federated unlearning is addressed as follows.

Definition 2. ((ϵ, δ)-Approximate Federated Unlearning). *Given an FL algorithm* $\mathscr{A} : \mathcal{D} \to \mathcal{W}$ *and an unlearning request* $\mathcal{U}_j$ *from client* j*, the unlearning algorithm* $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \to \mathcal{W}$ *is an* (ϵ, δ)*-approximate federated unlearning that unlearns* $\mathcal{U}_j$ *from*

$\mathscr{A}(\mathcal{D})$ *if*

$$\Pr[\mathscr{A}^u(\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \in \mathcal{W}] \leq e^\epsilon \cdot \Pr[\mathscr{A}(\mathcal{D}^u) \in \mathcal{W}] + \delta, \ \textit{and}$$

$$\Pr[\mathscr{A}(\mathcal{D}^u) \in \mathcal{W}] \leq e^\epsilon \cdot \Pr[\mathscr{A}^u(\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \in \mathcal{W}] + \delta.$$

In Definition 2, the distribution of the unlearned model parameters is indistinguishable from that of model parameters trained from scratch on the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$.

5.4 Exact Federated Unlearning

5.4.1 Quantization of Federated Learning

In the FL system, when the dataset $\mathcal{U}_j$ is deleted from $\mathcal{D}_j$ and $\mathcal{D}$ per client j's unlearning request, there may be changes in the final trained federated model. That is, the change of local user's dataset usually have influence on the trained model. As a result, it is hard to guarantee the exact equivalence on distribution between the unlearned model and the model trained from scratch on $\mathcal{D}^u$ as required in Definition 1. To overcome this challenge, stabilizing the FL algorithm becomes necessary to enable exact unlearning. In other words, the trained federated model is expected to have certain stability with respect to the local dataset, so that small changes on the local dataset should only cause a small or no change on the distribution of trained federated model parameters. In our problem, when a dataset $\mathcal{U}_j$ is requested to be unlearned, the trained federated model should not change too much.

If such changes can be evaluated efficiently during the unlearning stage, we can achieve the exact unlearning efficiently.

The way to reach stability in federated learning is quantization [137], where the aggregated parameters of the federated model are quantized to a discrete vertex in the hypothesis space of model parameters. The quantization operation $q(\alpha, w^t) = \hat{w}^t$ can map its continues input value w^t to a discrete value $\hat{w}^t$, which is expressed as follows:

$$\hat{w}^t = \alpha \cdot z^*, \text{ s.t. } z^* = \arg\min_{z \in \mathbf{Z}^d} \|w^t - \alpha \cdot z\|_2, \tag{5.10}$$

where $\mathbf{Z}^d$ is the d-dimensional integer space. For instance, in 1-dimension, $q(\alpha = 0.1, w^t = 0.62)$ maps w^t=0.62 to $\hat{w}^t$=0.6, which is like a rounding operation; and in 2-dimension, $q(\alpha = 0.5, w^t = [1.1, 2.7])$ maps w^t to the closest α vertex $\hat{w}^t = [1.0, 2.5]$.

By applying quantization, we first propose the quantized federated learning (Q-FL) algorithm as presented in Algorithm 5 to enable exact unlearning and then demonstrate our exact and efficient unlearning algorithm to unlearn a dataset $\mathcal{U}_j$. At the beginning of Q-FL, the server initializes model parameter w^0. The initialization is passed through quantization function $q(\alpha, \cdot)$ to get the quantized model $\hat{w}^0$, which is then distributed to all participated clients as their local models for computing ClientUpdate$(\cdot)$. The operation of clients is the same as that in the original FL described in Section 5.2, including computing gradients, updating local models, and uploading their updated local models to the server. After the server receives local updates and performs aggregation, a new federated model w^{t+1} is obtained. Next, quantization function is executed in Line 7 on the federated model w^{t+1} to get the

Algorithm 5 Quantized Federated Learning (Q-FL)

Input: the number of iterations T, the number of clients K, learning rate η, the granularity of quantization α

Output: quantized federated model $\hat{w}^T$

Server executes: initialize $\hat{w}^0 = q(\alpha, w^0)$
for iteration $t = 0$ to T **do**
  for client $k \in \mathcal{K}$ in parallel **do**
    $w_k^{t+1} \leftarrow$ **ClientUpdate**$(k, \hat{w}^t)$
  end for
  $w^{t+1} \leftarrow \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k^{t+1}$
  $\hat{w}^{t+1} = q(\alpha, w^{t+1})$
  save w^{t+1} and $\hat{w}^{t+1}$ on server; // as meta-data
end for
return $\hat{w}^T$
ClientUpdate$(k, \hat{w}^t)$: // run on each client
  compute gradient $\nabla L_k(\hat{w}^t)$ for $\mathcal{D}_k$
  update local model $w_k^{t+1} \leftarrow \hat{w}^t - \eta \nabla L_k(\hat{w}^t)$
  upload model w_k^{t+1} to server.

quantized model parameter $\hat{w}^{t+1} = q(\alpha, w^{t+1})$. Both the original federated model w^{t+1} and the quantized federated model $\hat{w}^{t+1}$ are stored on server as meta-data $\mathcal{M}$.

It is worth noticing that through quantization at server in each iteration t, the quantized federated model become stable as a constant with a high probability with respect to unlearning small datasets. Thus, $\mathcal{U}_j$ can be easily unlearned from the quantized FL model without complex re-computation and communication. Besides, the proposed Q-FL algorithm can not only support exact unlearning, but also preserve the model utility and convergence even if quantization operation disturbs its parameters. Hereafter, we first state a Lemma 1 and then use it to prove the convergence bound of proposed quantized federated learning (Q-FL) algorithm.

Lemma 1. *In the Q-FL of Algorithm 5, the loss value of quantized FL model between t-th*

iteration and $(t+1)$*-th iteration is bounded by the following inequality:*

$$\mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\} \leq \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \beta_2 \mathbb{E}\{\|N^{t+1}\|^2\}, \tag{5.11}$$

where $\beta_1 = -\eta + \frac{\mu\eta^2}{2}$ *and* $\beta_2 = \frac{\mu}{2}$.

Proof. The quantization function $\hat{w}^t = q(\alpha, w^t)$ essentially is a random noise perturbation. In each dimension of w^t, a random noise with uniform distribution $U(-\frac{\alpha}{2}, \frac{\alpha}{2})$ is added, which makes $\hat{w}^t$ be a perturbed result of w^t. Therefore, we have $\hat{w}^t = q(\alpha, w^t) = w^t + N^t$ and obtain Eq. (5.12).

$$\hat{w}^t = w^t + N^t = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k^t + N^t, \tag{5.12}$$

where $N^t \sim U(-\frac{\alpha}{2}, \frac{\alpha}{2})$ is the noise added in iteration t. According to the training process of gradient descent method, the local model of each client k is updated as

$$w_k^{t+1} = \hat{w}^t - \eta \nabla L_k(\hat{w}^t). \tag{5.13}$$

By combining Eq. (5.12) and Eq. (5.13), $\hat{w}^{t+1}$ can be computed as

$$\hat{w}^{t+1} = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} (\hat{w}^t - \eta \nabla L_k(\hat{w}^t)) + N^{t+1}. \tag{5.14}$$

Since the gradient of loss function $L_k(w)$ is Lipschitz continuous (see property 2 in Assumption), the loss function $L_k(w)$ is convex [139, 140]. We can construct a new convex function

$g(w) = \frac{\mu}{2} w^\top w - L_k(w)$ [141] and obtain its gradient $\nabla g(w) = \mu w - \nabla L_k(w)$. Because of the convexity of $g(w)$, there is

$$g(\hat{w}^{t+1}) \geq g(\hat{w}^t) + \nabla g(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t). \tag{5.15}$$

By substituting $g(w) = \frac{\mu}{2} w^\top w - L_k(w)$ into Eq. (5.15), we get Eq. (5.16).

$$\begin{aligned} &\frac{\mu}{2} \hat{w}^{t+1\top} \hat{w}^{t+1} - L_k(\hat{w}^{t+1}) \\ \geq &\frac{\mu}{2} \hat{w}^{t\top} \hat{w}^t - L_k(\hat{w}^t) + (\mu \hat{w}^t - \nabla L_k(\hat{w}^t))^\top (\hat{w}^{t+1} - \hat{w}^t). \end{aligned} \tag{5.16}$$

That is,

$$\begin{aligned} &L_k(\hat{w}^{t+1}) - L_k(\hat{w}^t) \\ \leq &\nabla L_k(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t) \\ &+ \big[\frac{\mu}{2} \hat{w}^{t+1\top} \hat{w}^{t+1} - \frac{\mu}{2} \hat{w}^{t\top} \hat{w}^t - \mu \hat{w}^{t\top} (\hat{w}^{t+1} - \hat{w}^t)\big] \\ \leq &\nabla L_k(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t) + \frac{\mu}{2} \|\hat{w}^{t+1} - \hat{w}^t\|^2. \end{aligned} \tag{5.17}$$

From the property 1 in Assumption, the gradient is bounded and unbiased, so Eq. (5.18) is obtained by taking expectation at both sides of Eq. (5.17).

$$\begin{aligned} \mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\} \leq &\mathbb{E}\{\nabla L(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t)\} \\ &+ \frac{\mu}{2} \mathbb{E}\{\|\hat{w}^{t+1} - \hat{w}^t\|^2\}. \end{aligned} \tag{5.18}$$

To estimate the upper bound of the right side in Eq. (5.18), we need to bound two items: $\hat{w}^{t+1}-\hat{w}^t$ and $\|\hat{w}^{t+1}-\hat{w}^t\|^2$. From Eq. (5.12), the difference between $\hat{w}^{t+1}$ and $\hat{w}^t$ is computed as follows:

$$\begin{aligned}\hat{w}^{t+1}-\hat{w}^t =& \sum_{k\in\mathcal{K}}\frac{|\mathcal{D}_k|}{|\mathcal{D}|}\left[\hat{w}_k^t-\eta\nabla L_k(\hat{w}^t)\right]+N^{t+1}-\hat{w}^t\\ =&\hat{w}^t-\eta\sum_{k\in\mathcal{K}}\frac{|\mathcal{D}_k|}{|\mathcal{D}|}\nabla L_k(\hat{w}^t)+N^{t+1}-\hat{w}^t\\ =&-\eta\sum_{k\in\mathcal{K}}\frac{|\mathcal{D}_k|}{|\mathcal{D}|}\nabla L_k(\hat{w}^t)+N^{t+1}\\ =&-\eta\nabla L(\hat{w}^t)+N^{t+1}. \end{aligned} \tag{5.19}$$

Then, for $\|\hat{w}^{t+1}-\hat{w}^t\|$, we have

$$\begin{aligned}\|\hat{w}^{t+1}-\hat{w}^t\| =&\|-\eta\nabla L(\hat{w}^t)+N^{t+1}\| \\ \leq&\|\eta\nabla L(\hat{w}^t)\|+\|N^{t+1}\|.\end{aligned} \tag{5.20}$$

Based on Eq. (5.19), Eq. (5.20), and Eq. (5.18), we can have

$$
\begin{aligned}
&\mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\} \\
\leq &\mathbb{E}\{\nabla L(\hat{w}^t)^\top(\hat{w}^{t+1} - \hat{w}^t)\} + \frac{\mu}{2}\mathbb{E}\{\|\hat{w}^{t+1} - \hat{w}^t\|^2\} \\
\leq &\mathbb{E}\{\nabla L(\hat{w}^t)^\top(-\eta\nabla L(\hat{w}^t) + N^{t+1})\} \\
&+ \frac{\mu}{2}\mathbb{E}\{(\|\eta\nabla L(\hat{w}^t)\| + \|N^{t+1}\|)^2\} \\
= &\mathbb{E}\{-\eta\|\nabla L(\hat{w}^t)\|^2 + (\nabla L(\hat{w}^t)^\top N^{t+1})\} \\
&+ \frac{\mu}{2}\mathbb{E}\{\eta^2\|\nabla L(\hat{w}^t)\|^2 + 2\eta\|\nabla L(\hat{w}^t)\|\|N^{t+1}\| + \|N^{t+1}\|^2\} \\
= &-\eta\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \mathbb{E}\{\|\nabla L(\hat{w}^t)N^{t+1}\|\} + \frac{\mu\eta^2}{2}\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} \\
&+ \mu\eta\mathbb{E}\{\|\nabla L(\hat{w}^t)\|\|N^{t+1}\|\} + \frac{\mu}{2}\mathbb{E}\{\|N^{t+1}\|^2\} \\
\overset{(i)}{=} &(-\eta + \frac{\mu\eta^2}{2})\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \frac{\mu}{2}\mathbb{E}\{\|N^{t+1}\|^2\}
\end{aligned} \tag{5.21}
$$

The equality (i) holds because the mean of noise N^t is 0. For simplicity, let $\beta_1 = -\eta + \frac{\mu\eta^2}{2}$ and $\beta_2 = \frac{\mu}{2}$. Lemma 1 is proved. □

Then, we can use Lemma 1 to prove Theorem 7.

Theorem 7. *The convergence upper bound of our proposed Q-FL Algorithm 5 is given by Eq.* (5.22) *when* $\eta \in (0, \frac{2}{\mu}]$ *and is given by Eq.* (5.23) *when* $\eta \in (\frac{2}{\mu}, \infty)$.

$$
\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (1 + 2\tau\beta_1)^t C^0 - \frac{\beta_2\alpha^2 d[1 - (1 + 2\tau\beta_1)^t]}{24\tau\beta_1}, \tag{5.22}
$$

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (\frac{1}{2\tau} + \beta_1)G^2 + \frac{\beta_2 \alpha^2 d}{12}, \tag{5.23}$$

where w^* is the optimal parameter of federated model, $C^0 = \|L(\hat{w}^0) - L(w^*)\|$ is the initialization quality of federated model, and η is the learning rate of local models.

Proof. From Lemma 1, we have

$$\begin{aligned}\mathbb{E}\{L(\hat{w}^{t+1}) - L(w^*)\} \leq & \mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \\ & + \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}_t)\|^2\} + \beta_2 \mathbb{E}\{\|N^{t+1}\|^2\}.\end{aligned} \tag{5.24}$$

The property of strong convexity implies Polyak-Lojasiewicz (PL) inequality:

$$\tau(l(w,(x,y)) - l(w^*,(x,y))) \leq \frac{1}{2}\|\nabla l(w,(x,y))\|^2, \tag{5.25}$$

which indicates that

$$2\tau(L(w) - L(w^*)) \leq \|\nabla L(w)\|^2. \tag{5.26}$$

When $\eta \in (0, \frac{2}{\mu}]$, $\beta_1 < 0$. By multiplying β_1 in both sides of Eq. (5.26), we have

$$\begin{aligned}&\beta_1 \|\nabla L(w)\|^2 \leq 2\tau\beta_1 \mathbb{E}\{(L(w) - L(w^*))\} \\ &\Rightarrow \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} \leq 2\tau\beta_1 \mathbb{E}\{(L(\hat{w}^t) - L(w^*))\}\end{aligned} \tag{5.27}$$

By substituting Eq. (5.27) into Eq. (5.24), the following result can be computed.

$$
\begin{aligned}
&\mathbb{E}\{L(\hat{w}^{t+1}) - L(w^*)\} \\
\leq &\mathbb{E}\{L(\hat{w}^{t}) - L(w^*)\} + 2\tau\beta_1\mathbb{E}\{(L(\hat{w}^{t}) - L(w^*))\} + \beta_2\mathbb{E}\{\|N^{t+1}\|^2\} \\
= &(1+2\tau\beta_1)\mathbb{E}\{L(\hat{w}^{t}) - L(w^*)\} + \beta_2\mathbb{E}\{\|N^{t+1}\|^2\} \\
\leq &(1+2\tau\beta_1)[(1+2\tau\beta_1)\mathbb{E}\{L(\hat{w}^{t-1}) - L(w^*)\} + \beta_2\mathbb{E}\{\|N^{t}\|^2\}] \\
&+ \beta_2\mathbb{E}\{\|N^{t+1}\|^2\} \\
\leq &(1+2\tau\beta_1)^2\mathbb{E}\{L(\hat{w}^{t-1}) - L(w^*)\} + (1+2\tau\beta_1)\beta_2\mathbb{E}\{\|N^{t}\|^2\} \\
&+ \beta_2\mathbb{E}\{\|N^{t+1}\|^2\} \\
&\cdots \\
\leq &(1+2\tau\beta_1)^{t+1}\mathbb{E}\{L(\hat{w}^{0}) - L(w^*)\} + \beta_2\mathbb{E}\{\|N^{t+1}\|^2\}\sum_{h=0}^{t}(1+2\tau\beta_1)^h \\
\leq &(1+2\tau\beta_1)^{t+1}\mathbb{E}\{L(\hat{w}^{0}) - L(w^*)\} + \frac{\beta_2\alpha^2 d}{12}\sum_{h=0}^{t}(1+2\tau\beta_1)^h \\
= &(1+2\tau\beta_1)^{t+1}C^0 - \frac{\beta_2\alpha^2 d[1-(1+2\tau\beta_1)^{(t+1)}]}{24\tau\beta_1}.
\end{aligned}
$$

When $\eta \in [\frac{2}{\mu}, \infty)$, we have $\beta_1 > 0$ and the following inequality.

$$
\mathbb{E}\{L(\hat{w}^{t}) - L(w^*)\} \leq \frac{1}{2\tau}\mathbb{E}\{\|\nabla L(\hat{w}^{t})\|^2\} \tag{5.28}
$$

By substituting Eq. (5.28) into Eq. (5.24), the result is

$$\begin{aligned}
&\mathbb{E}\{L(\hat{w}^{t+1}) - L(w^*)\} \\
&\leq \mathbb{E}\{L(\hat{w}^t) - L(w^*)\} + \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}_t)\|^2\} + \beta_2 \mathbb{E}\{\|N^{t+1}\|^2\} \\
&\leq \frac{1}{2\tau}\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}_t)\|^2\} + \beta_2 \mathbb{E}\{\|N^{t+1}\|^2\} \\
&\leq (\frac{1}{2\tau} + \beta_1)G^2 + \frac{\beta_2 \alpha^2 d}{12}.
\end{aligned}$$

Theorem 7 is proved. □

Theorem 7 states that even though our proposed Q-FL algorithm is obfuscated by quantization, the trained federated model can still converge.

5.4.2 Exact and Efficient Federated Unlearning

When our quantized federated learning algorithm terminates after T iterations, a federated model $\hat{w}^T$ is completely trained. With such a quantized federated model $\hat{w}^T$, exact unlearning process can be executed to unlearn $\mathcal{U}_j$ from the previously trained model $\hat{w}^T$ once the unlearning request is received from client j, and then the corresponding unlearned federated model $\hat{w}^{u,T}$ will be obtained.

According to Algorithm 5, in each iteration $t \in [0, T]$, the trained local model w_j^{t+1} of any client j is calculated as

$$w_j^{t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j|}[\sum_{(x,y)\in \mathcal{D}_j^u} \nabla l(w_j^t, (x, y)) + \sum_{(x,y)\in \mathcal{U}_j} \nabla l(w_j^t, (x, y))]. \tag{5.29}$$

When $\mathcal{U}_j$ is removed from client j's dataset $\mathcal{D}_j$, the updated model $w_j^{u,t+1}$ should be calcu-

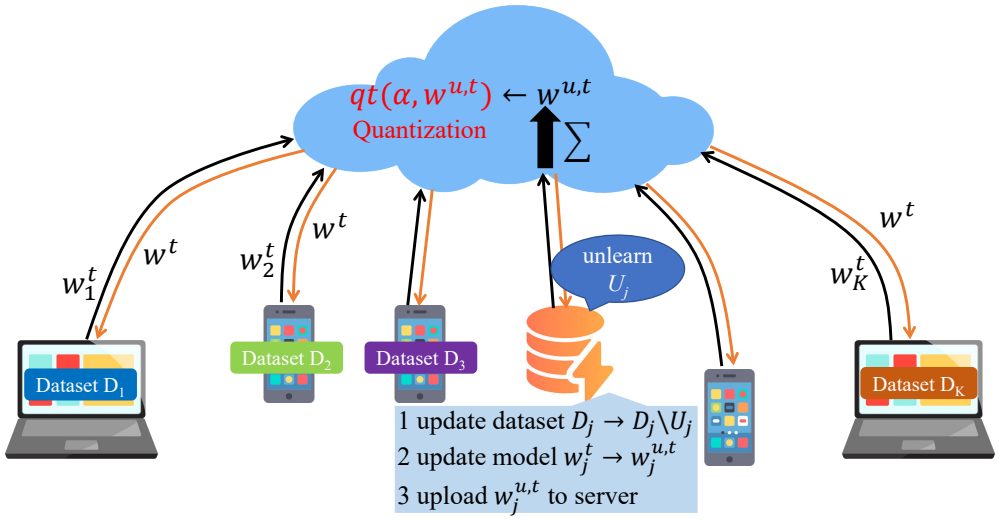


Figure 5.2 The framework of proposed Exact-Fun algorithm.

lated via Eq. (5.30) to unlearn $\mathcal{U}_j$.

$$w_j^{u,t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j^u|}[\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j^t, (x,y))]. \tag{5.30}$$

The difference between the trained local model w_j^{t+1} and the unlearned local model $w_j^{u,t+1}$ is only the gradients of data in $\mathcal{U}_j$. Thus, to get $w_j^{u,t+1}$ efficiently without computing the gradients of $\mathcal{D}_j^u$, we can directly remove the gradient of $\mathcal{U}_j$ from the previously trained local model w_j^{t+1}. By comparing Eq. (5.29) and Eq. (5.30), the rule of updating $w_j^{u,t+1}$ from w_j^{t+1} is given as

$$w_j^{u,t+1} = \frac{|\mathcal{D}_j|}{|\mathcal{D}_j^u|} w_j^{t+1} - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} w_j^t + \frac{\eta}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j^t, (x,y))]. \tag{5.31}$$

Notice that in Eq. (5.31), we have already got w_j^{t+1} and w_j^t in Q-FL Algorithm 5 as meta-data and only need to compute gradients for data points in $\mathcal{U}_j$. The change on the federated model caused by unlearning request may be small if the size of $\mathcal{U}_j$ is not too large. Moreover,

Algorithm 6 Exact and Efficient Federated Unlearning

Input: the number of iterations T, the number of clients K, the granularity of quantization α, the unlearning client j and its unlearning request $\mathcal{U}_j$
Output: the unlearned federated model $\hat{w}^{u,T}$

identify the unlearning client j and request $\mathcal{U}_j$
for iteration $t = 0$ to T **do**
  compute the gradient $\nabla L_j(w_j^t)$ of client j on $\mathcal{U}_j$
  update local model $w_j^{u,t}$ via Eq. (5.31)
  upload $w_j^{u,t}$ to server
  calculate $w^{u,t} = w^t - \frac{|\mathcal{D}_j^u|}{|\mathcal{D}|}(w_j^t - w_j^{u,t})$
  quantize $w^{u,t}$, $q(\alpha, w^{u,t}) = \hat{w}^{u,t}$
  if $\hat{w}^{u,t} = \hat{w}^t$ // deletion makes no changes **then**
    continue;
  else
    send $\hat{w}^{u,t}$ to all clients
    re-run Algorithm 5 on remaining dataset $\mathcal{D}^u$ with $\hat{w}^{u,t}$ as initialization for iterations in $[t, T]$
  end if
end for
return $\hat{w}^{u,T}$

with the quantized stability of Q-FL, the federated model can still be stable with a large probability (see Theorem 8).

The exact federated unlearning process for deleting $\mathcal{U}_j$ is presented in Fig. 5.2 and Algorithm 6. When client j submits an unlearning request, $\mathcal{U}_j$ is deleted from $\mathcal{D}_j$, and an updated local model $w_j^{u,t}$ is computed based on client j's trained local model at iteration t as shown in Eq. (5.31). Then, $w_j^{u,t}$ is uploaded to server to aggregate a new federated model $w^{u,t}$ that is quantized through quantization function $q(\alpha, \cdot)$ to produce $\hat{w}^{u,t}$. If the newly quantized federated model $\hat{w}^{u,t}$ is the same as the stored federated model $\hat{w}^t$, deleting $\mathcal{U}_j$ has no impact on the previously trained federated model $\hat{w}^t$, and retraining from scratch on $\mathcal{D}_j^u$ would output the same federated model. This implies that our unlearning method is exact.

On the contrary, if the newly quantized federated model $\hat{w}^{u,t}$ is different from the stored federated model $\hat{w}^t$, the stability of quantized federated model is broken, and retraining from the t-th iteration is needed to remove the influence of $\mathcal{U}_j$ from learned models in iteration t until terminated iteration T.

In our unlearning algorithm Exact-Fun, the major computation time lies in the retraining process (*i.e.,* Line 12 of Algorithm 6), which can be controlled by adjusting the quantization parameter α based on system requirements. A larger α brings a smaller retraining probability and a less retraining cost, but also a reduced model utility because of the increase of noise perturbation. In Theorem 8, we prove that the retraining probability of Algorithm 6 is a function of α and a proper α value can help realize efficient unlearning.

Theorem 8. *Assume the distance between the original federated model w^t and its unlearned federated model $w^{u,t}$ has an upper bound B, i.e., $\|w^t - w^{u,t}\| \leq B$ with $t \in [0, T]$. The probability Algorithm 6 needs retraining is given by Eq.* (5.32)*.*

$$\Pr(\hat{w}^{u,t} \neq \hat{w}^t) = \begin{cases} 1 - (\frac{\alpha}{2B})^d, & B \in [\alpha, \infty) \\ 1 - (1 - \frac{B}{2\alpha})^d, & B \in (0, \alpha) \end{cases} \tag{5.32}$$

where d is the dimension of model parameter space $\mathcal{W}$.

Proof. Without loss of generality, we suppose the model w^t is quantized to vertex α in 1-dimension space $\mathbf{R}$. Due to the property of quantization operation, w^t that is mapped to α should originally belong to the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$ with uniform distribution $U(\frac{\alpha}{2}, \frac{3\alpha}{2})$. Since

$\|w^t - w^{u,t}\| \leq B$, $w^{u,t}$ falls into the range $[w^t - B, w^t + B]$ after unlearning process. Thus, $\hat{w}^t = \hat{w}^{u,t}$ happens only if both w^t and $w^{u,t}$ fall into the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$, through which we can calculate the probability of $\hat{w}^t$ being equal to $\hat{w}^{u,t}$, *i.e.*, $\Pr(\hat{w}^{u,t} = \hat{w}^t)$.

On the other hand, $w^t, w^{u,t}, \hat{w}^t, \hat{w}^{u,t} \in \mathcal{W} \in \mathbf{R}^d$. The training process of learning algorithm is random, and the distribution of each dimension of parameters is independent. Thus, we can first calculate $\Pr(\hat{w}^{u,t} = \hat{w}^t)$ and $\Pr(\hat{w}^{u,t} \neq \hat{w}^t)$ in 1-dimension space $\mathbf{R}$ and then extend the calculation to d-dimension space $\mathbf{R}^d$ based on binomial distribution.

The range of w^t is $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$ with the length of α, and the range of $w^{u,t}$ is $[w^t - B, w^t + B]$ with the length of $2B$. According to relation between the length $2B$ and the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$, there are three cases for discussing whether $\hat{w}^{u,t}$ is in $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$.

(i) When $B \in [\alpha, \infty)$, for any $w^t \in [\frac{\alpha}{2}, \frac{3\alpha}{2}]$, the probability of unlearned model $w^{u,t}$ falls into $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$ is $\frac{\alpha}{2B}$ as B is large enough to cover the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$. So, $\Pr(\hat{w}^{u,t} = \hat{w}^t)$ in 1-dimension space $\mathbf{R}$ can be calculated as follows,

$$\Pr(\hat{w}^{u,t} = \hat{w}^t) = \int_{\frac{\alpha}{2}}^{\frac{3\alpha}{2}} \frac{1}{\alpha} \cdot \frac{\alpha}{2B} dw = \frac{\alpha}{2B}. \tag{5.33}$$

In d-dimension space $\mathbf{R}^d$, we have $\Pr(\hat{w}^{u,t} = \hat{w}^t) = (\frac{\alpha}{2B})^d$, because every dimension should satisfy the equality requirement. Thus, in d-dimension space $\mathbf{R}^d$, $\Pr(\hat{w}^{u,t} \neq \hat{w}^t) = 1 - (\frac{\alpha}{2B})^d$.

(ii) When $B \in [\frac{\alpha}{2}, \alpha)$, $\forall w^t \in [\frac{\alpha}{2}, \frac{3\alpha}{2} - B]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{B + w - \frac{\alpha}{2}}{2B}$; $\forall w^t \in [\frac{3\alpha}{2} - B, \frac{\alpha}{2} + B]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{\alpha}{2B}$; and $\forall w^t \in [\frac{\alpha}{2} + B, \frac{3\alpha}{2}]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{B + \frac{3\alpha}{2} - w}{2B}$. So, $\Pr(\hat{w}^{u,t} = \hat{w}^t)$ in

1-dimension space $\mathbf{R}$ is calculated as follows

$$\begin{aligned}\Pr(\hat{w}^{u,t}=\hat{w}^{t}) =&\frac{1}{\alpha}[\int_{\frac{\alpha}{2}}^{\frac{3\alpha}{2}-B}\frac{B+w-\frac{\alpha}{2}}{2B}dw+\int_{\frac{3\alpha}{2}-B}^{\frac{\alpha}{2}+B}\frac{\alpha}{2B}dw\\&+\int_{\frac{\alpha}{2}+B}^{\frac{3\alpha}{2}}\frac{B+\frac{3\alpha}{2}-w}{2B}dw]=1-\frac{B}{2\alpha}.\end{aligned}\tag{5.34}$$

Similarly, by extending to d-dimension space $\mathbf{R}^d$, we have $\Pr(\hat{w}^{u,t}\neq\hat{w}^{t})=1-(1-\frac{B}{2\alpha})^d$.

(iii) When $B\in(0,\frac{\alpha}{2})$, similar to the case in (ii), $\Pr(\hat{w}^{u,t}=\hat{w}^{t})$ can be calculated by

$$\begin{aligned}\Pr(\hat{w}^{u,t}=\hat{w}^{t}) =&\frac{1}{\alpha}[\int_{\frac{\alpha}{2}}^{\frac{\alpha}{2}+B}\frac{B+w-\frac{\alpha}{2}}{2B}dw+\int_{\frac{\alpha}{2}+B}^{\frac{3\alpha}{2}-B}\frac{2B}{2B}dw\\&+\int_{\frac{3\alpha}{2}-B}^{\frac{3\alpha}{2}}\frac{B+\frac{3\alpha}{2}-w}{2B}dw]=1-\frac{B}{2\alpha}.\end{aligned}\tag{5.35}$$

Thus, in d-dimension space $\mathbf{R}^d$, $\Pr(\hat{w}^{u,t}\neq\hat{w}^{t})=1-(1-\frac{B}{2\alpha})^d$.

Combining the cases (i), (ii), and (iii), Theorem 8 is proved. □

Theorem 8 implies that a larger quantization value α can reduce the retraining probability exponentially but may result in a worse convergence bound. The trade-off between efficiency and convergence should be designed carefully.

Remark. Our unlearning algorithm Exact-Fun processes one unlearning request each time. In real FL application, there may be multiple users submitting multiple unlearning requests, for which Exact-Fun can run multiple times to accomplish these unlearning requests one by one. Per the requirements of applications, the specific one-by-one implementation manner can be determined in a different ways, such as “first-come-first-serve” and “priority-

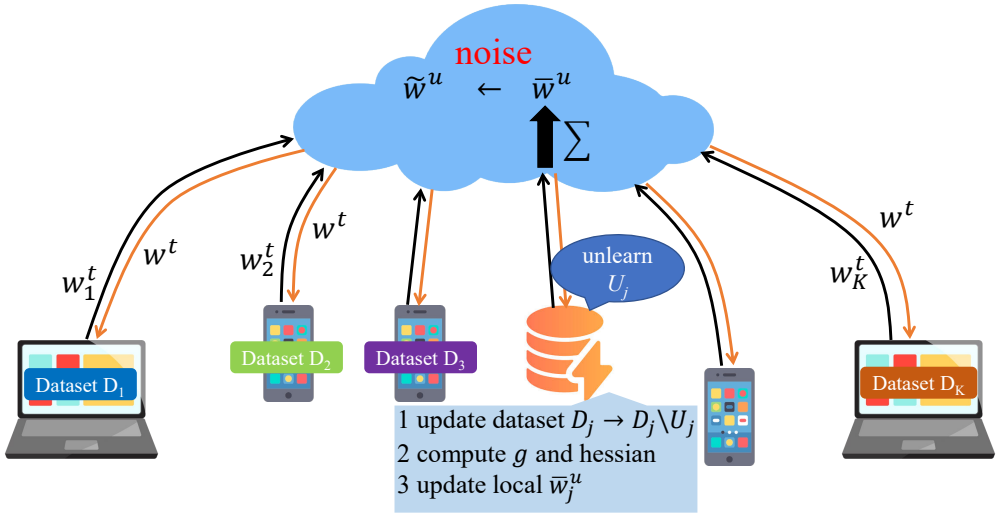


Figure 5.3 The framework of proposed Appro-Fun algorithm.

based service order".

5.5 Approximate Federated Unlearning

When processing an unlearning request $\mathcal{U}_j$, $\mathcal{U}_j$ should be deleted from $\mathcal{D}_j$ and $\mathcal{D}$, and the influence of $\mathcal{U}_j$ should be removed from the trained federated model. Moreover, a successful unlearning algorithm should have unlearning cost (*e.g.*, computation time) less than retraining cost from scratch on the remaining dataset $\mathcal{D}^u$. In order to achieve the approximate unlearning in the existing FL model, we propose the Appro-Fun algorithm and present its framework in Fig. 5.3. Appro-Fun consists of two major steps: (i) at the first step, the local clients who request unlearning remove their private data, compute gradient and hessian, and upload new unlearned local models to the server, which is described in Section 5.5.1; and (ii) at the second step, the server aggregates a new unlearned model and perturbs it with differentially private noise, which is presented in Section 5.5.2.

5.5.1 Local Model Unlearning

To process an unlearning request, the data points should be deleted from local dataset, and the influence of should be removed from the trained local and federated models. Without loss of generality, we illustrate Appro-Fun algorithm using a simple case where one local client submits one unlearning request at a time. For the case when each of multiple clients submits multiple unlearning requests, we can repeat Appro-Fun algorithm iteratively to implement data deletion requests one by one. Firstly, per client j's unlearning request, $\mathcal{U}_j$ is removed from client j's local dataset $\mathcal{D}_j$. Next, the trained local model w_j is used to unlearn the influence of $\mathcal{U}_j$ from it through Newton's method [142] to approximate the model retrained from scratch on dataset $\mathcal{D}_j^u = \mathcal{D}_j \setminus \mathcal{U}_j$.

Originally, the computation of Newton's method is a time consuming process because of the calculation of second-order derivation for hessian matrix. To reduce time cost, we compute the Newton's method in the following. Noticing that the loss function for client j on dataset $\mathcal{D}_j$ is formulated below,

$$L_j(w_j, \mathcal{D}_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j, (x, y)). \tag{5.36}$$

Rearranging Eq. (5.36) will give us Eq. (5.37) as follows

$$\begin{aligned} |\mathcal{D}_j| \cdot L_j(w_j, \mathcal{D}_j) =& \sum_{(x,y)\in\mathcal{D}_j^u} l(w_j, (x, y)) + \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x, y)) \\ =& |\mathcal{D}_j^u| \cdot L_j^u(w_j, \mathcal{D}_j^u) + |\mathcal{U}_j| \cdot L_j(w_j, \mathcal{U}_j), \end{aligned} \tag{5.37}$$

where $L_j^u(w_j, \mathcal{D}_j^u)$ is the loss value on dataset $\mathcal{D}_j^u$ and $L_j(w_j, \mathcal{U}_j)$ is the loss value on removed dataset $\mathcal{U}_j$.

Since the local model w_j is a trained local optimizer on $\mathcal{D}_j$, the gradient $\nabla L_j(w_j, \mathcal{D}_j)$ is approximately zero. Thus, we can calculate the approximation of gradient $\nabla L_j^u(w_j, \mathcal{D}_j^u)$ with respect to w_j in Eq. (5.38).

$$\nabla L_j^u(w_j, \mathcal{D}_j^u) = -\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} \cdot \nabla L_j(w_j, \mathcal{U}_j). \tag{5.38}$$

In this way, we only need to perform gradient computation for those removed data in $\mathcal{U}_j$, which is much less than original method calculating on all remaining data.

Furthermore, the hessian matrix of loss function on $\mathcal{D}_j^u$ can be calculated with the removed data $\mathcal{U}_j$ as well. The second order derivative of Eq. (5.37) can be written as

$$\begin{aligned} |\mathcal{D}_j| \cdot \nabla^2 L_j(w_j, \mathcal{D}_j) =& |\mathcal{D}_j^u| \cdot \nabla^2 L_j^u(w_j, \mathcal{D}_j^u) \\ &+ |\mathcal{U}_j| \cdot \nabla^2 L_j(w_j, \mathcal{U}_j). \end{aligned} \tag{5.39}$$

Accordingly, we have the hessian matrix H of $L_j^u(w_j, \mathcal{D}_j^u)$ as

$$H = \frac{|\mathcal{D}_j|}{|\mathcal{D}_j^u|} \cdot \nabla^2 L_j(w_j, \mathcal{D}_j) - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} \cdot \nabla^2 L_j(w_j, \mathcal{U}_j). \tag{5.40}$$

Because the first term $\nabla^2 L_j(w_j, \mathcal{D}_j)$ can be calculated offline before unlearning process starts, the unlearning process only needs to calculate the second-order derivative for $\nabla^2 L_j(w_j, \mathcal{U}_j)$,

Algorithm 7 Approximate Federated Unlearning (Appro-Fun)

Input: the trained model w from Algorithm 4, the number of clients K, the unlearning request $\mathcal{U}_j$, parameters ϵ and δ to calculated σ
Output: the unlearned federated model $\tilde{w}^u$

1: **local client executes**:
2: compute gradient $\nabla L_j^u(w_j, \mathcal{D}_j^u)$ as Eq. (5.38)
3: compute hessian matrix H as Eq. (5.40)
4: update local model $\bar{w}_j^u$ of client j as Eq. (5.41)
5: upload $\bar{w}_j^u$ to server
6: **server executes**:
7: calculate $\bar{w}^u = w - \frac{|\mathcal{D}_j|}{|\mathcal{D}|}(w_j - \bar{w}_j^u)$
8: noise perturbation $\tilde{w}^u \leftarrow \bar{w}^u + \mathcal{N}(0, \sigma^2 I)$
9: **return** unlearned federated model $\tilde{w}^u$

which can save more computation cost.

With the gradient $\nabla L_j^u(w_j, \mathcal{D}_j^u)$ obtained in Eq. (5.38) and the hessian matrix H, the local client j can unlearn $\mathcal{U}_j$ via Newton's method to get $\bar{w}_j^u$:

$$\bar{w}_j^u = w_j - H^{-1}\nabla L_j^u(w_j, \mathcal{D}_j^u) = w_j + \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j, \mathcal{U}_j), \tag{5.41}$$

where $\bar{w}_j^u$ is the temporarily unlearned model of client j and is uploaded to the server for next step operation. In real implementation, the calculation in Eq. (5.41) is estimated through approximate hessian as solved in [143, 144] to save time. The computation process of the above local model unlearning is presented in lines 1-5 of Algorithm 7.

5.5.2 Federated Model Perturbation

The local model unlearning step only processes the unlearning request at local client j' side, which is not enough to remove private data in the federated setting. The federated model still needs further operation to remove the impact of unlearned dataset $\mathcal{U}_j$ as shown in

lines 6-8 of Algorithm 7. Upon receiving the uploaded model parameter $\bar{w}_j^u$ from client j, a new federated model $\bar{w}^u$ is aggregated at the server. This federated model $\bar{w}^u$ is treated as an temporarily unlearned version of the previously trained federated model w, (*i.e.,* the output of Algorithm 4). Recall that in Definition 2, approximate unlearning requires that the unlearned model $\bar{w}^u$ and the retrained model w^u should be indistinguishable. While, given an $\mathcal{U}_j$, $\bar{w}^u$ will be directly calculated from $\mathcal{U}_j$ and $\bar{w}_j^u$, so the distance between $\bar{w}^u$ and w^u is also deterministic [85]. To this end, $\bar{w}^u$ should be obfuscated into a random range by adding differentially private noise to reach indistinguishability.

To achieve indistinguishability between the temporarily unlearned model $\bar{w}^u$ and the model w^u retrained from scratch, noise scale should be set according to the distance $\|w^u - \bar{w}^u\|$. Actually, $\|w^u - \bar{w}^u\|$ is similar to the global sensitivity of differential privacy [145], which can guide the noise scale adding into $\bar{w}^u$. Specifically, Theorem A.1 in [145] approves the range of σ for (ϵ, δ)-differentially private mechanisms. Thus, with the similar analysis for (ϵ, δ)-approximate federated unlearning (see line 8 of Algorithm 7), the noise scale σ should satisfy

$$\sigma \geq \frac{\max \|w^u - \bar{w}^u\| \sqrt{2 \ln(1.25/\delta)}}{\epsilon}. \tag{5.42}$$

Since the clients' unlearning requests are unpredictable to the FL system, it is hard or impossible to obtain the exact value of $\max \|w^u - \bar{w}^u\|$. In stead, we can estimate the upper bound of $\|w^u - \bar{w}^u\|$ for noise addition without losing too much model utility after unlearning. In the following, we present theoretical analysis on the upper bound of $\|w^u - \bar{w}^u\|$ as well as

the performance of output unlearned model.

Lemma 2. *Let w be the model parameter of Algorithm 4 trained on the original dataset $\mathcal{D}$ and w^u be the model parameter retrained on the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$. Then, the distance between w and w^u is bounded by Eq.* (5.43)*:*

$$\|w - w^u\| \leq \frac{2m\iota}{|\mathcal{D}|\tau}, \tag{5.43}$$

where m is the size of unlearning dataset $\mathcal{U}_j$, ι and τ are constant given in Section 5.2.

Proof. Without loss of generality, we assume the unlearning request is from a client j who holds dataset $\mathcal{D}_j$.

The loss functions of client j on dataset $\mathcal{D}_j$ and the remaining dataset $\mathcal{D}_j^u$ are defined as follows:

$$L_j(w, \mathcal{D}_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w, (x, y)) \tag{5.44}$$

$$L_j^u(w, \mathcal{D}_j^u) = \frac{1}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{D}_j^u} l(w, (x, y)) \tag{5.45}$$

where $|\mathcal{D}_j^u|$ is the size of dataset $\mathcal{D}_j^u = \mathcal{D}_j \setminus \mathcal{U}_j$.

Let $w_j = \arg\min_{w\in\mathcal{W}} \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w, (x, y))$ be the local minimizer of Eq. (5.44) and the local minimizer of Eq. (5.45) be $w_j^u = \arg\min_{w\in\mathcal{W}} \frac{1}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{D}_j^u} l(w, (x, y))$. Thus, we

can have the following equation

$$
\begin{aligned}
& L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \\
=& \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j^u, (x,y)) - \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j, (x,y)) \\
=& \frac{1}{|\mathcal{D}_j|} [\sum_{(x,y)\in\mathcal{D}_j^u} l(w_j^u, (x,y)) - \sum_{(x,y)\in\mathcal{D}_j^u} l(w_j, (x,y)) \\
& + \sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x,y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x,y))] \\
=& \frac{1}{|\mathcal{D}_j|} [|\mathcal{D}_j^u| (L_j^u(w_j^u, \mathcal{D}_j^u) - L_j^u(w_j, \mathcal{D}_j^u)) \\
& + \sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x,y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x,y))] \\
\overset{(i)}{\leq}& \frac{1}{|\mathcal{D}_j|} [\sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x,y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x,y))] \\
\overset{(ii)}{\leq}& \frac{m\iota}{|\mathcal{D}_j|} \|w_j^u - w_j\|,
\end{aligned}
\tag{5.46}
$$

where the inequality (i) holds because w_j^u is the minimizer of $L_j^u(w_j^u, \mathcal{D}_j^u)$ defined above, and (ii) holds because the loss function $l(w, (x, y))$ is ι-Lipschitz.

Additionally, based on the assumption of strong convexity (assumption (4)), we can get the following equation

$$
L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \geq \frac{\tau}{2} \|w_j^u - w_j\|^2 \tag{5.47}
$$

Combining the Eq. (5.46) and Eq. (5.47), we can derive the following inequality

$$\begin{aligned} &\frac{\tau}{2}\|w_j^u - w_j\|^2 \leq L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \leq \frac{m\iota}{|\mathcal{D}_j|}\|w_j^u - w_j\| \\ &\Rightarrow \|w_j^u - w_j\| \leq \frac{2m\iota}{|\mathcal{D}_j|\tau} \end{aligned} \tag{5.48}$$

This inequality Eq. (5.48) indicates that the change of model parameter on client j before and after unlearning is bounded. Then, the unlearned model of client j is uploaded to server for aggregation, where the weight of client j is at most $\frac{|\mathcal{D}_j|}{|\mathcal{D}|}$. Thus, the federated model on server before and after unlearning $\mathcal{U}_j$ is bounded in a range by the following inequality

$$\|w^u - w\| \leq \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \cdot \frac{2m\iota}{|\mathcal{D}_j|\tau} = \frac{2m\iota}{|\mathcal{D}|\tau}. \tag{5.49}$$

This finishes the proof of Lemma 2. □

The upper bound of $\|w - w^u\|$ is used to find the distance between w^u and $\bar{w}^u$.

Theorem 9. *Let w^u be the model parameter retrained from scratch on dataset $\mathcal{D}^u$ and $\bar{w}^u$ be the approximately unlearned federated model in Line 7 of Algorithm 7. The distance between w^u and $\bar{w}^u$ is bounded by Eq.* (5.50)*:*

$$\|w^u - \bar{w}^u\| \leq \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3}, \tag{5.50}$$

where m is the size of unlearning dataset $\mathcal{U}_j$ and M is the Hessian-Lipschitz constant.

Proof. Based on Eq. (5.44) and Eq. (5.45), we can calculate the Taylor's expansion for $\nabla L_j^u(w_j^u, \mathcal{D}_j^u)$ at point w_j as follows,

$$\begin{aligned}
&\nabla L_j^u(w_j^u, \mathcal{D}_j^u) = \nabla L_j^u(w_j, \mathcal{D}_j^u) + \nabla^2 L_j^u(w_j, \mathcal{D}_j^u)[w_j^u - w_j] \\
&+ \frac{1}{2}\nabla^3 L_j^u(w_j, \mathcal{D}_j^u)[w_j^u - w_j]^2 \\
&\overset{(i)}{\Rightarrow} \| - \nabla L_j^u(w_j, \mathcal{D}_j^u) - H[w_j^u - w_j]\| \\
&= \frac{1}{2}\nabla^3 L_j^u(w_j, \mathcal{D}_j^u)\|w_j^u - w_j\|^2 \\
&\Rightarrow \|\nabla L_j^u(w_j, \mathcal{D}_j^u) + H[w_j^u - w_j]\| \overset{(ii)}{\leq} \frac{M}{2}\|w_j^u - w_j\|^2. \qquad (5.51)
\end{aligned}$$

The reason of (i) is that w_j^u is the minimizer of loss function $L_j^u(w_j^u, \mathcal{D}_j^u)$, so $\nabla L_j^u(w_j^u, \mathcal{D}_j^u)$ is zero. The inequality (ii) holds because the loss function is M-Hessian Lipschitz.

Moreover, we can rewrite the above term $\nabla L_j^u(w_j, \mathcal{D}_j^u)$ as

$$\begin{aligned}
&\nabla L_j^u(w_j, \mathcal{D}_j^u) = \frac{1}{|\mathcal{D}_j^u|}\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j, (x, y)) \\
=&\frac{1}{|\mathcal{D}_j^u|}[\sum_{(x,y)\in\mathcal{D}_j} \nabla l(w_j, (x, y)) - \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x, y))] \\
=&\frac{1}{|\mathcal{D}_j^u|}[|\mathcal{D}_j|\nabla L_j(w_j, \mathcal{D}_j) - \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x, y))] \\
\overset{(i)}{=}&-\frac{1}{|\mathcal{D}_j^u|}\sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x, y)) = -\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}\nabla L_j(w_j, \mathcal{U}_j), \qquad (5.52)
\end{aligned}$$

where the equality (i) holds because w_j is the minimizer of loss function $L_j(w_j, \mathcal{D}_j)$.

Next, let β be the difference between w_j^u and $\bar{w}_j^u$, that is, $w_j^u - \bar{w}_j^u = \beta$. Since in the

Line 4 of Algorithm 7 we have $\bar{w}_j^u = w_j + \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} H^{-1} \nabla L_j(w_j, \mathcal{U}_j)$, the following equation is obtained,

$$
\begin{aligned}
& w_j^u - \bar{w}_j^u = w_j^u - [w_j + \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} H^{-1} \nabla L_j(w_j, \mathcal{U}_j)] = \beta \\
& \Rightarrow w_j^u - w_j = \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} H^{-1} \nabla L_j(w_j, \mathcal{U}_j) + \beta
\end{aligned} \tag{5.53}
$$

Substituting Eq. (5.52) and Eq. (5.53) into Eq. (5.51), we can get the inequality

$$
\begin{aligned}
& \| - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} \nabla L_j(w_j, \mathcal{U}_j) + H[\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} H^{-1} \nabla L_j(w_j, \mathcal{U}_j) + \beta] \| \\
& \leq \frac{M}{2} \|w_j^u - w_j\|^2.
\end{aligned} \tag{5.54}
$$

Rearranging Eq. (5.54), we can simplify the inequality to Eq. (5.55)

$$
\|H\beta]\| = \|\nabla^2 L_j^u(w_j, \mathcal{D}_j^u)\beta]\| \leq \frac{M}{2} \|w_j^u - w_j\|^2. \tag{5.55}
$$

Due to the τ-strongly convexity of loss function, we can have a corollary [139]

$$
\|\nabla^2 L_j^u(w_j, \mathcal{D}_j^u)\beta]\| \geq \tau \|\beta\|. \tag{5.56}
$$

Combining the Eq. (5.55) and Eq. (5.56), we can have the upper bound distance between

w_j^u and $\bar{w}_j^u$

$$\|w_j^u - \bar{w}_j^u\| = \|\beta\| \leq \frac{M}{2\tau}\|w_j^u - w_j\|^2. \tag{5.57}$$

Here, $\|w_j^u - w_j\|$ is proved with an upper bound in Eq. (5.48) of Lemma 2. After the aggregation on server side, we can get the distance between w^u and $\bar{w}^u$ in the following equation,

$$\|w^u - \bar{w}^u\| \leq \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3}. \tag{5.58}$$

This ends the proof of Theorem 9. □

By combining Eq. (5.42) and Eq. (5.50), we obtain the setting for σ in Appro-Fun algorithm:

$$\sigma \geq \frac{2m^2\iota^2 M\sqrt{2\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon}. \tag{5.59}$$

Although our Appro-Fun algorithm injects gaussian noise during unlearning process, the unlearning effectiveness can be still kept in an acceptable range with performance guarantee, which is measured from two aspects. On the one hand, we use the loss difference between the unlearned model $\tilde{w}^u$ and the optimal model that is retrained from scratch on $\mathcal{D}^u$ to quantify performance gap and analyze it in Theorem 10.

Theorem 10. *Let $L(w^{u*})$ be the loss of the optimal model retrained from scratch on remaining dataset $\mathcal{D}^u$ and $L(\tilde{w}^u)$ be loss of unlearned model $\tilde{w}^u$ output by Appro-Fun Algorithm (*i.e.*,*

Algorithm 7). The loss distance can be bounded by:

$$\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \leq \frac{2m^2\iota^3 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} + \frac{2m^2\iota^2 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} + \frac{\mu}{\iota(T-1+\frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2). \tag{5.60}$$

This means that our unlearned model has a bounded performance loss compared with the optimal retained model.

Proof.

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \\
=&\mathbb{E}\{L(\tilde{w}^u) - L(w^u) + L(w^u) - L(w^{u*})\} \\
\leq&\mathbb{E}\{L(\tilde{w}^u) - L(w^u)\} + \mathbb{E}\{L(w^u) - L(w^{u*})\} \\
\overset{(i)}{\leq}&\mathbb{E}\{L(\tilde{w}^u) - L(w^u)\} + \frac{\mu}{\iota(T-1+\frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \\
\overset{(ii)}{\leq}&\iota\mathbb{E}\{\|\tilde{w}^u - w^u\|\} + \frac{\mu}{\iota(T-1+\frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2)
\end{aligned} \tag{5.61}$$

where the inequality (i) holds because the convergence bound of federated learning is proved by [138], and the inequality (ii) holds due to the Lipschitzness of loss function $l(w,(x,y))$.

Then, based on Theorem 9, we can calculate the expectation of $\|\tilde{w}^u - w^u\|$ as follows

$$\begin{aligned}\mathbb{E}\{\|\tilde{w}^u - w^u\|\} =& \mathbb{E}\{\|\tilde{w}^u - \bar{w}^u + \bar{w}^u - w^u\|\} \\ \leq& \mathbb{E}\{\|\tilde{w}^u - \bar{w}^u\|\} + \mathbb{E}\{\|\bar{w}^u - w^u\|\} \\ \leq& \mathbb{E}\{\|N\|\} + \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} \\ \leq& \sqrt{d}\sigma + \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} \end{aligned} \tag{5.62}$$

where N is the gaussian noise added in each unlearning process as shown in Line 8 of Algorithm 7.

Combining the Eq. (5.61), Eq. (5.62) and the value of σ, we can prove Theorem 10 as follows

$$\begin{aligned}&\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \leq \frac{2m^2\iota^3 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} + \frac{2m^2\iota^2 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\ &+ \frac{\mu}{\iota(T-1+\frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2).\end{aligned} \tag{5.63}$$

This finishes the proof of Theorem 10. □

On the other hand, we use the loss difference between the unlearned model $\tilde{w}^u$ and the optimal original federated model trained on original dataset $\mathcal{D}$ to evaluate the influence of data deletion, which is proved in Theorem 11.

Theorem 11. *Let $L(w^*)$ be the loss of the optimal model trained on original dataset $\mathcal{D}$ and $L(\tilde{w}^u)$ be the loss of unlearned model $\tilde{w}^u$ output by Appro-Fun Algorithm (*i.e.*, Algorithm 7).*

The loss distance can be bounded by:

$$\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \leq \frac{m|\mathcal{D}_j|\iota^2}{(n_j - m)|\mathcal{D}|\tau} + \frac{2m^2\iota^3 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2). \tag{5.64}$$

Here $n_j - m$ is the size of dataset $\mathcal{D}_j^u = \mathcal{D}_j \setminus \mathcal{U}_j$. This conclusion implies that compared with the optimal original model, the performance loss brought by data removal via our Appro-Fun algorithm is bounded.

Proof.

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \\
=&\mathbb{E}\{L(\tilde{w}^u) - L(w) + L(w) - L(w^*)\} \\
\leq&\mathbb{E}\{L(\tilde{w}^u) - L(w)\} + \mathbb{E}\{L(w) - L(w^*)\} \\
\overset{(i)}{\leq}&\mathbb{E}\{L(\tilde{w}^u) - L(w)\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \\
\overset{(ii)}{\leq}&\iota\mathbb{E}\{\|\tilde{w}^u - w\|\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2)
\end{aligned} \tag{5.65}$$

where w is the output of Algorithm 4, the inequality (i) holds because the convergence bound of federated learning is proved by [138], and the inequality (ii) holds due to the Lipschitzness of loss function $l(w, (x, y))$.

According to the Appro-Fun Algorithm 7, we can get

$$
\begin{aligned}
&\mathbb{E}\{\|\tilde{w}^u - w\|\} = \mathbb{E}\{\|\bar{w}^u + N - w\|\} \\
\overset{(i)}{=}&\mathbb{E}\{\|\frac{|\mathcal{D}_j|}{|\mathcal{D}|}(\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j, \mathcal{U}_j)) + N\|\} \\
\leq&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}\mathbb{E}\{\|H^{-1}\nabla L_j(w_j, \mathcal{U}_j)\|\} + \mathbb{E}\{\|N\|\} \\
\overset{(ii)}{\leq}&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\mathbb{E}\{\|\nabla L_j(w_j, \mathcal{U}_j)\|\} + \mathbb{E}\{\|N\|\} \\
=&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\mathbb{E}\{\|\frac{1}{|\mathcal{U}_j|}\sum_{(x,y)\in\mathcal{U}_j}\nabla l(w_j, (x,y))\|\} + \mathbb{E}\{\|N\|\} \\
\overset{(iii)}{\leq}&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\frac{|\mathcal{U}_j|\iota}{|\mathcal{U}_j|} + \sqrt{d}\sigma = \frac{m|\mathcal{D}_j|\iota}{(n_j - m)|\mathcal{D}|\tau} + \sqrt{d}\sigma,
\end{aligned} \tag{5.66}
$$

where N is the gaussian noise added in each unlearning process. The expectation above is taken with respect to the dataset $\mathcal{D}$ and noise N. The equation (i) is obtained from Eq. (5.41). The inequality (ii) holds because of the τ-strong convexity of loss function, which implies $\nabla^2 L_j^u(w_j, \mathcal{D}_j^u) \succeq \tau I$. The inequality (iii) holds because the loss function is ι Lipschitz.

Combining the Eq. (5.65), Eq. (5.66), and Eq. (5.59), we can prove Theorem 11 as follows,

$$
\begin{aligned}
\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \leq& \frac{m|\mathcal{D}_j|\iota^2}{(n_j - m)|\mathcal{D}|\tau} + \frac{2m^2\iota^3 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\
+& \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2).
\end{aligned} \tag{5.67}
$$

□

Table 5.2 The structure of neural networks.

Layer	F-MNIST Model	CIFAR-10 Model
1	$(5,5) \times 20$, Conv, ReLu	$(5,5) \times 32$, Conv, ReLu
2	$(2,2)$, Maxpooling	$(2,2)$, Maxpooling
3	$(5,5) \times 50$, Conv, Leaky ReLU	$(5,5) \times 64$, Conv, Leaky ReLU
4	$(2,2)$, Maxpooling	$(2,2)$, Maxpooling
5	$opt \times 256$, Dense, Leaky ReLU	$(5,5) \times 128$, Conv, Leaky ReLU
6	256×10, Dense	$(2,2)$, Maxpooling
7		$opt \times 256$, Dense, Leaky ReLU
8		256×10, Dense

5.6 Experiments

In this section, we conduct intensive experiments to validate the performance of Q-FL algorithm, Exact-Fun algorithm, and Appro-Fun algorithm.

5.6.1 Experiment Settings

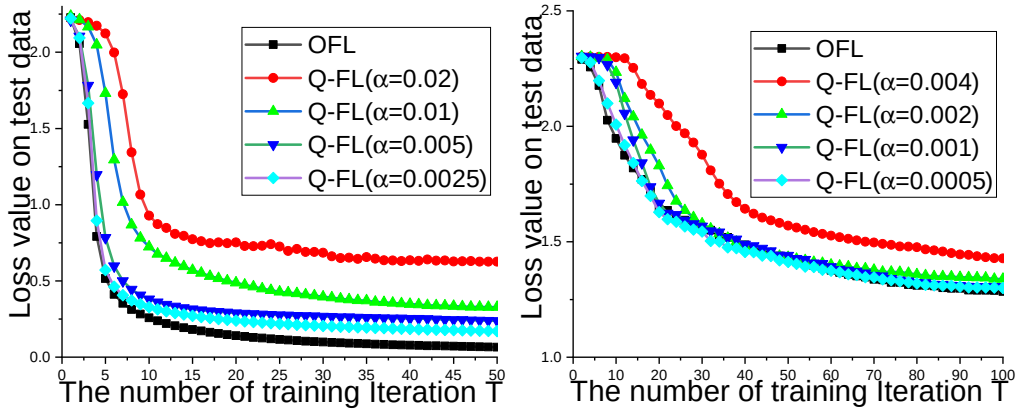
Fashion-MNIST dataset and CIFAR-10 dataset are adopted in our experiments. Our experiments are implemented by Pytorch on Google Colab Tesla T4 GPU. The learning model we use is neural networks for both Fashion-MNIST dataset and CIFAR-10 dataset. Model structure of Fashion-MNIST and CIFAR-10 datasets is shown in the following Table 5.2.

Training and Unlearning Scenarios. We set the number of clients K to be 10, 20, and 50. Our proposed Exact-Fun mechanism can support unlearning from multiple clients, each of which may submit multiple unlearning requests. For evaluation, $10\% \times K$ clients are randomly selected, and each of them submits 5 unlearning requests, so there are $0.5K$ unlearning requests in total. These requests are processed via Algorithm 6 one-by-one. Notably, in practice, the unlearned data should be a small portion of a client's local database, otherwise, the motivation of performing unlearning may not be sufficient, and the

effectiveness and efficiency of unlearning may not be good [71, 72]. So, for each selected client who request unlearning, the total number of unlearned data in the 5 requests is at most 20% of his/her local dataset, *i.e.,* the portion of unlearned data is $p \leq 20\%$.

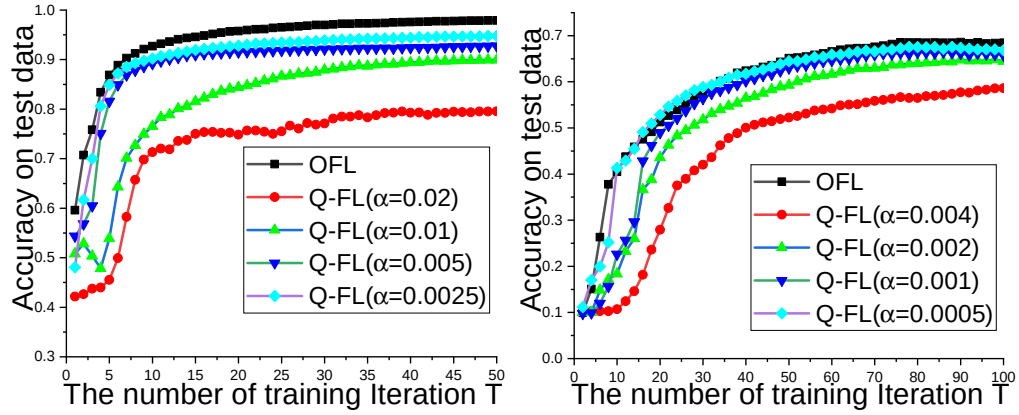
Baseline Models. For Q-FL algorithm, we choose the original federated learning (OFL) algorithm as a baseline, to compare model convergence, model accuracy, and training speed in Section 5.6.2. Since we are the first to explore the **exact** federated unlearning problem and there is no existing exact federated unlearning baselines. Retraining with OFL algorithm from scratch on the remaining dataset $\mathcal{D}^u$ is adopted as a baseline to evaluate our proposed Exact-Fun algorithm. Besides, one state-of-the-art approximate federated unlearning method [146] on INFOCOM 2022, is selected as baseline for unlearning performance comparison. The evaluation of unlearning performance is presented in Section 5.6.3.

In addition, to evaluate our Appro-Fun algorithm, one state-of-the-art approximate federated unlearning method published in INFOCOM 2022 [146], is selected for comparing unlearning performance. This baseline has settings similar to our problem: (i) both use approximate hessian matrix to calculate unlearned models on local client side, and (ii) both upload the local unlearned models to the server server for aggregation. But some techniques of the baseline are different from ours: (i) the baseline uses all remaining data to approximate the diagonal hessian matrix, and (ii) all local clients' models in the baseline method need to be updated.



(a) loss on F-MNIST dataset (b) loss on CIFAR-10 dataset

Figure 5.4 The loss value of FL models with different α (K=50).



(a) accu. on F-MNIST dataset (b) accu. on CIFAR-10 dataset

Figure 5.5 The accuracy value of FL models with different α (K=50).

5.6.2 Q-FL Performance

Our Q-FL algorithm uses quantization to stabilize the training process so as to facilitate exact unlearning. The impact of quantization on the federated learning is deeply investigated by changing the value of quantization parameter α to measure model convergence, accuracy, and training speed.

To evaluate the influence of α on our Q-FL algorithm empirically, we set α=$\{0.02, 0.01, 0.005, 0.0025\}$ for Fashion-MNIST dataset and $\alpha = \{0.0005, 0.001, 0.002, 0.004\}$ for CIFAR-10 dataset. The loss values of federated models on corresponding test datasets during each iteration are shown in Fig. 5.4. First of all, we can see that the loss values of all compared federated models decrease with the increase of T and reach to a stable level after a certain number iterations (*e.g.,* $T = 35$ in Fig. 5.4a). This observation confirms that our quantized federated learning can converge as analyzed in Theorem 7. For our Q-FL, a greater α value means more noise is added in model parameter, leading to a bigger loss value and a slower convergence speed. Especially, as shown in Fig. 5.4, the loss value of Q-FL with $\alpha = 0.0025$ converges nearly as fast as the baseline OFL. Therefore, though the quantization parameter α has an impact on model convergence, an appropriate α value can help our Q-FL achieve the comparable learning performance as the original federated learning. Similar conclusions can be found for CIFAR-10 dataset, which show our Q-FL models can converge and achieve small loss value as original federated learning.

Then, we present the influence of α on the testing accuracy of federated models in Fig. 5.5. As we can see from Fig. 5.5a, the accuracy of the compared federated models on Fashion-MNIST is increased when T grows up and can reach a stable value after the training process is done. Besides, for our Q-FL model, the quantized federated model with a smaller α value can achieve higher accuracy, and the training accuracy is more stable, because a smaller α means less noise is injected to model parameters. This stable accuracy also implies the Q-FL algorithm is converged. Specifically, when $\alpha = 0.0025$, the accuracy of Q-FL on Fashion-

Table 5.3 Training time comparison between OFL and Q-FL

K	OFL	Q-FL w/ $\alpha = 0.02$	Q-FL w/ $\alpha = 0.01$	Q-FL w/ $\alpha = 0.005$	Q-FL w/ $\alpha = 0.0025$
K=10	6.66±0.002	6.71±0.002	6.74±0.002	6.75±0.002	6.79±0.002
K=20	11.56±0.003	11.58±0.003	11.59±0.003	11.61±0.003	11.64±0.003
K=50	26.87±0.003	26.94±0.003	27.09±0.003	27.19±0.003	27.35±0.003

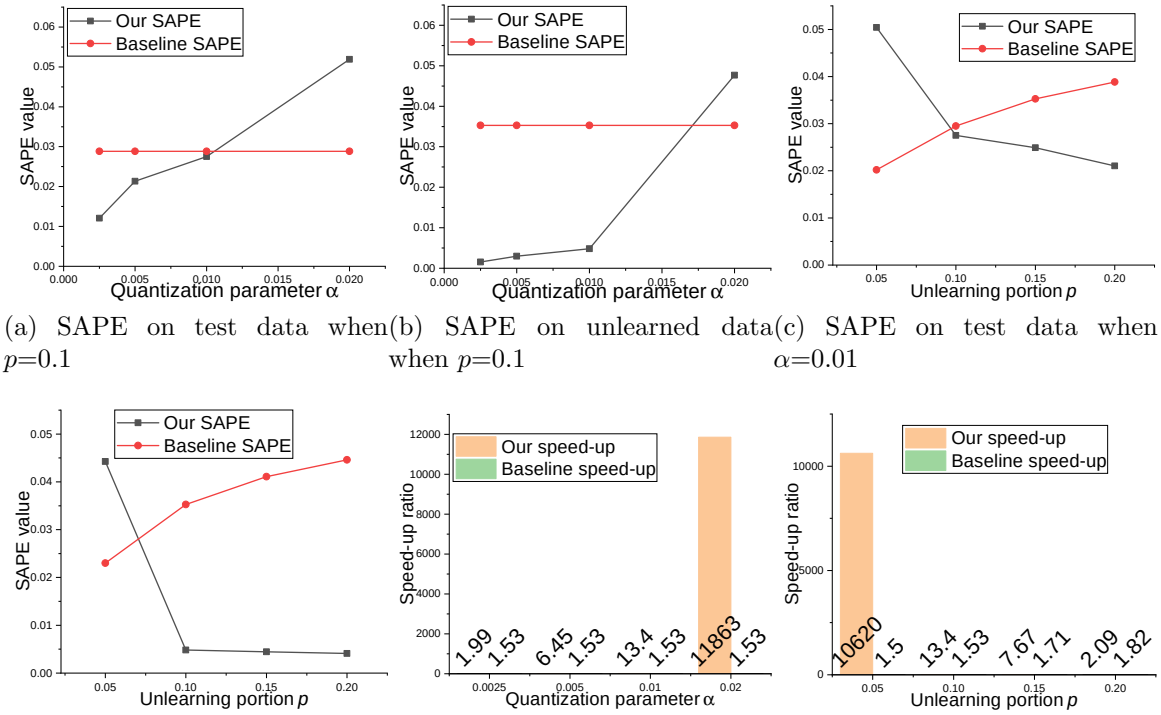
MNIST dataset is extremely close to OFL, which means our proposed Q-FL has comparable accuracy as the original federated learning if α is small enough. Similarly, the accuracy of Q-FL models on CIFAR-10 dataset is always increasing and reaches a stable value in Fig. 5.5b. When α is small, the accuracies of Q-FL models have little difference from that of OFL.

In addition, we also compare the training time of our Q-FL and the baseline OFL. Table 5.3 shows the average training time of one iteration (in second) of our Q-FL and the baseline OFL, where we can see that the Q-FL only increases the average training time of one iteration 2% compared with the baseline. This minor extra time cost is acceptable considering the significant unlearning efficiency improved by our Exact-Fun unlearning algorithm in next section. In a nutshell, the quantization function $q(\alpha, \cdot)$ of our Q-FL algorithm is not a time consuming process and can achieve desired model accuracy.

5.6.3 Unlearning Effectiveness and Efficiency

In this part, we evaluate the effectiveness and efficiency of our Exact-Fun algorithm. Due to the page limit, we only present the results with K=20.

Unlearning Effectiveness. The unlearning effectiveness can be evaluated in terms of the accuracy difference between the retrained federated model and the unlearned federated model output by different unlearning algorithms (*i.e.,* Exact-Fun and INFOCOM22 algo-



(a) SAPE on test data when p=0.1 (b) SAPE on unlearned data when p=0.1 (c) SAPE on test data when α=0.01

(d) SAPE on unlearned data when α=0.01 (e) speed-up ratio vary with α (f) speed-up ratio vary with p

Figure 5.6 SAPE on Fashion-MNIST test data and unlearned data with different α and p in Fig. 5.6a, 5.6b, 5.6c, 5.6d; speed-up in 5.6e, 5.6f.

rithm). Here we adopt Symmetric Absolute Percentage Error (SAPE), which is used in many unlearning literature [146, 77, 147], to measure the difference between two accuracies, Acc_1 and Acc_2 as unlearning effectiveness: $SAPE(Acc_1, Acc_2) = \frac{|Acc_1 - Acc_2|}{|Acc_1| + |Acc_2|} \times 100\%$. SAPE computed on different datasets can address unlearning effectiveness from two aspects: (i) for the test data, a smaller value of SAPE means the accuracy of unlearned model is closer to the accuracy of the retrained model, indicating a better prediction performance of unlearning algorithm; while (ii) for the unlearned data, a smaller SAPE means the unlearned model contains less information about the removed data.

Fig. 5.6a shows the impact of the quantization parameter α on the effectiveness of Exact-

Fun with setting $p = 0.1$. The baseline (INFOCOM22) is not impact by α, and has constant SAPE when α varies. Clearly, we find that the SAPE value on the test data increases as α is getting greater. The reason is that a greater α introduces more noise perturbation on the quantized federated learning process, which leads a lower accuracy in our unlearned federated model. Compared with the baseline INFOCOM22, our Exact-Fun algorithm beats it when α is small (*e.g.,* α is 0.0025, 0.005 and 0.01). On the other hand, the SAPE on the unlearned data increases slowly for small α values and increases sharply when α becomes 0.02. When α is smaller, our Exact-Fun algorithm is more likely to retrain the quantized federated model on remaining dataset as the retraining method does, so the accuracy difference between our unlearned model and the retrained model becomes smaller. When α is larger, our Exact-Fun has less probability to retrain for unlearning, resulting in larger accuracy difference between our unlearned model and the retrained model on the unlearned data. Then, the SAPE of our unlearned model is better than the baseline except α=0.02. The attractive merit of our Exact-Fun is the exact unlearning guarantee for user, while the baseline is just an approximate solution. So, we can conclude that a proper α can help Exact-Fun algorithm achieve better effectiveness than the baseline.

Hereafter, we explore the impact of unlearning portion p on unlearning effectiveness with $p = \{0.05, 0.1, 0.15, 0.2\}$. In Fig. 5.6c and Fig. 5.6d, the SAPE values on the test data and unlearned data are presented. We can observe when the unlearning portion p increases, the SAPE value of our Exact-Fun decreases, which can be explained through the viewpoint of model retraining. When p is smaller, the probability of retraining the

quantized federated model in Exact-Fun is lower, causing a larger difference between our quantized model and the retrained model. In contrast, when p becomes larger, our Exact-Fun algorithm needs to be retrained on the remaining dataset, so the accuracy difference is reduced. Especially, the SAPE value on the unlearned data decreases drastically from p=0.05 to p=0.1. Because Exact-Fun does not retrain the quantized model when p=0.05, the accuracy difference between our unlearned model and the retrained model is large. While, SAPE of the baseline INFOCOM22 keeps increasing when p gets larger, because the baseline adopts a hessian matrix based approximate unlearning, which has more error when more data is removed. As a summary, our Exact-Fun algorithm outperforms the baseline approximate unlearning algorithm in effectiveness when unlearning more data.

Since the purpose of unlearning is to remove the private information of deleted data, membership inference attack (MIA) is a metric to evaluate the unlearning effectiveness in many related works [148, 149, 150], which infers whether a data sample is in the training dataset of a model or not. So, for the deleted data, a lower MIA accuracy means that the unlearning algorithm has stronger privacy protection. In Table 5.4, original model means we only delete data but do not change the trained model, so high MIA accuracy remains on both datasets. Retrained model has the lowest MIA accuracy (near 50%) due to the complete retraining on remaining dataset, so a good unlearning algorithm should have similar MIA accuracy to the retrained model. From Table 5.4, we can see that for all α, our Exact-Fun achieves similar accuracy as the retrained model and is much lower that that of INFOCOM22, which means our Exact-Fun is stronger in private information removal. The reason of Exact-

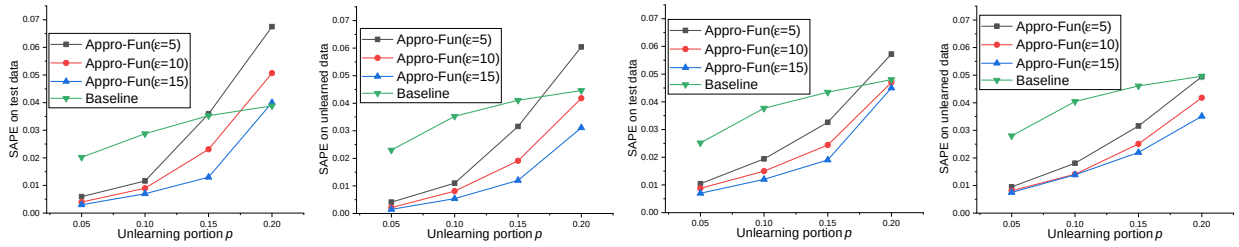
Table 5.4 MIA accuracy (%) for deleted data on original model, retrained model, and different unlearning algorithms

Dataset	Baseline Original Model	Baseline Retrained Model	Baseline INFOCOM22	Exact-Fun α=0.0025	Exact-Fun α=0.005	Exact-Fun α=0.01	Exact-Fun α=0.02
F-MNIST	82.86±1.35	51.54±2.27	62.29±1.49	52.75±1.60	52.15±2.02	52.34±1.62	56.17±1.63
CIFAR-10	87.11±1.13	53.03±1.83	69.21±1.22	54.60±1.64	54.74±1.90	52.59±1.67	59.79±2.78

Fun's success is that quantization perturbs model parameters, and some retraining further removes the information of deleted data.

Unlearning Efficiency. The unlearning efficiency can be measured by the unlearning speed-up ratio. Specifically, we unlearn the same unlearning requests (*i.e.,* deleting the same data) via retraining method and different unlearning algorithms separately, and we can obtain the average time to process one unlearning request for each method. The unlearning speed-up ratio is the ratio of the average time of retraining method to the average time of different unlearning algorithms. The higher speed-up ratio, the better efficiency.

The influence of α on the efficiency of Exact-Fun is shown in Fig. 5.6e. It is obvious that the speed-up ratio increases with the increase of α, because a greater α value indicates a stronger stability of our quantized federated model and less retraining probability, leading to higher speed-up ratio. Compared with the baseline (with fixed speed-up ratio 1.53), our Exact-Fun is more efficient for every α. Then, the influence of p on the unlearning efficiency is reported in Fig. 5.6f. With the increase of p, the speed-up ratio of Exact-Fun is reduced because unlearning a larger portion part of data may break the stability of quantized federated model, which results in more retraining time. For the baseline, even though its speed-up ratio increases as p gets larger, our Exact-Fun still outperforms and can achieve over 10,000× speed-up ratio when p=0.05.



(a) SAPE on test data of F-MNIST (b) SAPE on unlearned data of F-MNIST (c) SAPE on test data of CIFAR-10 (d) SAPE on unlearned data of CIFAR-10

Figure 5.7 SAPE comparison between Appro-Fun and baseline with different unlearning portion p and ϵ.

5.6.4 Unlearning Effectiveness of Appro-Fun

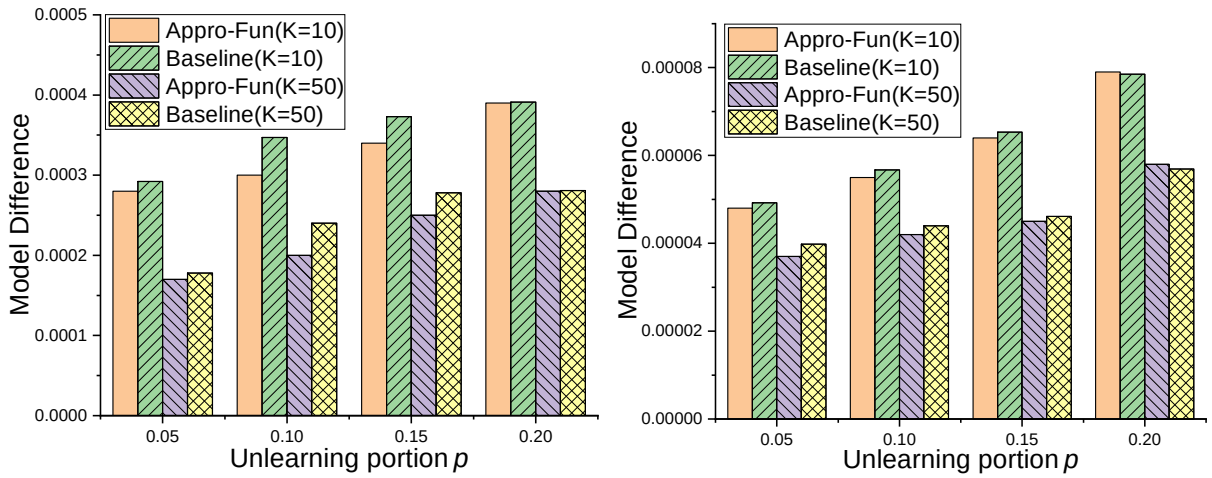
In this section, we evaluate the effectiveness of Appro-Fun algorithm in terms of SAPE, model difference, loss difference, and privacy leakage.

SAPE Comparison. We adopt Symmetric Absolute Percentage Error (SAPE) to measure the accuracy difference between the retrained federated model and the unlearned federated model, which is used as an effectiveness metric in many unlearning literatures [146, 77, 147]. SAPE is calculated as: $SAPE(Acc_1, Acc_2) = \frac{|Acc_1 - Acc_2|}{|Acc_1| + |Acc_2|} \times 100\%$ with Acc_1 and Acc_2 being two accuracy values. SAPE computed on different datasets can address unlearning effectiveness from different aspects: (i) for the test data, a smaller SAPE value means the accuracy of unlearned model is closer to that of the retrained model, indicating a better prediction result of the unlearned model; while (ii) for the unlearned data, a smaller SAPE value means the unlearned model contains less information about the removed data, leaking less privacy about the removed data.

We evaluate the SAPE values of Appro-Fun algorithm and baseline by changing un-

learning portion p. First, the SAPE value on test data of Fashion-MNIST dataset is shown in Fig. 5.7a, where the unlearning portion p is set to be $\{0.05, 0.1, 0.15, 0.2\}$. We can see that the SAPE value gets increased along with the increase of unlearning portion p, because when more data is removed, the difference between unlearned model and the retrained model becomes larger, increasing difference in model accuracy. Meanwhile, when more data is removed, the larger scale noise is injected in our Appro-Fun's unlearned models (see Eq. (5.59)), which also causes more loss on prediction accuracy. Therefore, the unlearned models of our Appro-Fun algorithm have a drastically increased SAPE value when p gets larger, *e.g.,* $p = 0.2$. Then, comparing the unlearned models with different ϵ in Fig. 5.7a, we can find that a larger ϵ can achieve a smaller SAPE value, meaning the unlearned model's prediction accuracy is closer to that of the retrained model. The reason is that a larger ϵ allows more relaxed approximation and less noise injection into the unlearned model, which can help obtain more accurate prediction. In addition, even though the SAPE value of the baseline approximate method is increasing slowly, our Appro-Fun algorithm can still achieves smaller SAPE values when p is smaller. The reason is that the baseline adopts a diagonal hessian matrix for approximate unlearning operation, which loses too much useful information during unlearning and causes low accuracy. While when p grows, the baseline is slightly better than our Appro-Fun for smaller ϵ settings (*e.g.,* ϵ=5, 10). This is because a larger p and a smaller ϵ imply more noise perturbation in our algorithm, reducing the prediction accuracy of our unlearned models.

Fig. 5.7b depicts the trend of SAPE on unlearned data in Fashion-MNIST dataset. Sim-



(a) Model difference with different p and K on MNIST

(b) Model difference with different p and K on CIFAR-10

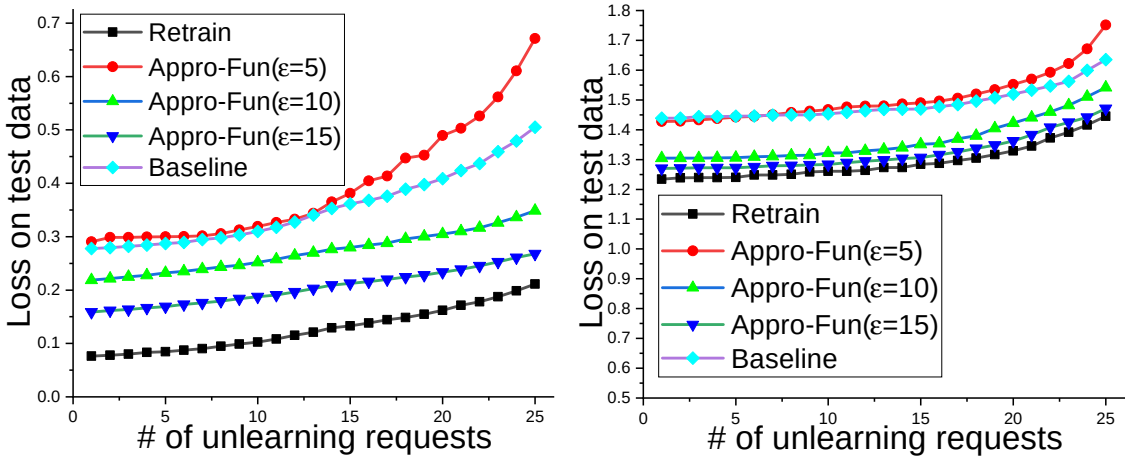
Figure 5.8 Model difference between Appro-Fun and baseline.

ilar to the results on test data, the SAPE value is getting larger when p increases. When comparing different ϵ, the larger ϵ, the less noise injection, which reduces the SAPE values. Compared with the baseline unlearned model, our unlearned model has smaller SAPE values in most settings (except p=0.2, ϵ=5 in Fig. 5.7b). Moreover, it is worth noticing that the SAPE values on unlearned data are relatively smaller than those on test data. This means that our unlearned model performs more similarly to the retrained model on the removed data, which indicates a better unlearning effectiveness. Similar results of SAPE on CIFAR-10 dataset are provided in Fig. 5.7c and Fig. 5.7d. In both test data and unlearned data, our Appro-Fun algorithm outperforms the baseline for all p value and most ϵ (except p=0.2 on unlearned data). As a summary, our Appro-Fun algorithm is better than the baseline method in most case of federated unlearning settings, especially when ϵ is set reasonably.

Model and Loss Comparison. As pointed out by [83], the SAPE value may not be

enough to qualify the unlearning effectiveness. For complex machine learning models, the difference between unlearned model parameters and the retrained model parameters is also a vital metric. Therefore, we calculate the average element-wise model difference between the unlearned models (*i.e.,* output by Appro-Fun and the baseline) and the retrained model as another measurement of unlearning effectiveness. As shown in Fig. 5.8a, along with the increase of p, the difference between the unlearned model and the retrained model is getting larger for both our Appro-Fun algorithm and the baseline. This is because when deleting more data, noise injection and approximate hessian matrix cause larger error, which enlarges the distance between the unlearned model and the retrained model. While our Appro-Fun models achieve smaller difference than the baseline, which means better model similarity to the fully retrained model.

On the other hand, the number of clients K varies in the unlearning processes. A larger number of clients in FL system can reduce the model difference thanks to federated aggregation. In Fig. 5.8a, with the same p, the model differences when K=50 are always smaller than that when K=10, which means more involved clients can mitigate the model difference of unlearned federated models by averaging. This critical findings indicates the approximate federated unlearning may have better effectiveness in the federated system with more clients. Similar results can be observed from Fig. 5.8b, where our Appro-Fun algorithm has smaller model difference compared with the baseline. In particular, only when p=0.2, our unlearned model difference is slightly larger than the baseline. This may be caused by the injected noise in our Appro-Fun algorithm as we proved in Theorem 9 implying the larger amount



(a) Loss on test F-MNIST

(b) Loss on test CIFAR-10

Figure 5.9 Loss comparison on test data.

of unlearned data, the larger model difference and differentially private noise scale. To sum up, our Appro-Fun algorithm produces better unlearned model than the baseline method in terms of model difference at most unlearning settings.

In addition, the prediction loss on test data is adopted as another metric for unlearning effectiveness. After processing each unlearning request, we test the loss of all unlearned models (including Appro-Fun, baseline, and Retraining) on the same test dataset so as to undertand how well an unlearned model performs on future prediction. In Fig. 5.9a, the loss on test data of all unlearning methods increase as the number of unlearning requests increases, because there are less training data available and more approximate error. For our Appro-Fun algorithm, when ϵ is larger (*e.g.,* ϵ=10, 15), the loss is not only smaller than the baseline, but also closer to the retrained model, showing its unlearning effectiveness. Only when ϵ is small (*i.e.,* ϵ=5), our Appro-Fun performs slightly worse than the baseline as the number of unlearning requests increases. From the results in Fig. 5.9b, the same conclusion

Table 5.5 MIA accuracy (%) for deleted data on different models.

Dataset	Retrained	Baseline Liu et al.	Our Appro-Fun Algorithm ϵ=5	ϵ=10	ϵ=15
F-MNIST	51.54±2.27	62.29±1.49	46.49±1.03	56.33±2.15	60.31±1.76
CIFAR-10	53.03±1.83	69.21±1.22	49.60±1.51	58.17±2.11	63.04±1.27

can be drawn, which means our Appro-Fun algorithm can achieve less loss on test data when ϵ is larger.

Privacy Leakage Comparison. Since the purpose of unlearning is to remove the private information of deleted data, membership inference attack (MIA) is a suitable metric to evaluate the privacy level in many related works [148, 149, 150]. MIA infers whether a data sample is in the training dataset of a model, so for the deleted data, a lower MIA accuracy means that the unlearning algorithm has stronger privacy protection. In Table 5.5, for all ϵ, our Appro-Fun algorithm obtains lower MIA accuracy values than the baseline; especially when ϵ is smaller (*e.g.*, ϵ=5) Appro-Fun is even better. The reason of Appro-Fun's success is that we not only delete private data at local client side but also introduce differential privacy on server side, which provide enhanced protection for the unlearned data.

5.6.5 Unlearning Efficiency of Appro-Fun

First of all, the original federated model is trained to converge using Algorithm 4. Then, we unlearn the same unlearning requests on the federated model through retraining, Appro-Fun, and the baseline separately. Finally, we calculate the average time to process one unlearning request in all algorithms. The speed-up ratio is the ratio of the average time of the retraining method to the average time of the unlearning algorithm. In Table 5.6, the

Table 5.6 The speed-up ratio comparison between Appro-Fun and baseline on Fashion-MNIST and CIFAR-10 datasets.

Dataset	Retraining	Baseline Liu et al.				Our Appro-Fun Algorithm			
		p=0.05	p=0.1	p=0.15	p=0.2	p=0.05	p=0.1	p=0.15	p=0.2
F-MNIST	1255.71 (s)	2.17×	2.23×	2.25×	2.36×	6.58×	6.32×	5.24×	4.75×
CIFAR-10	2851.54 (s)	5.47×	5.70×	6.42×	7.08×	12.13×	11.07×	9.66×	8.81×

column "Retraining" provides the absolute time cost (in seconds) of the retraining method, and the speed-up ratio are given in the following columns with different p values. On both datasets with all p values, our unlearning speed-up ratio is higher than that of the baseline. This is because our Appro-Fun adopts a simpler approximation updating strategy with less calculation (see Section 5.5.1) than the approximation method used by the baseline, reaching faster approximation. Moreover, the speed-up ratio of Appro-Fun is decreased as p increases. The reason is that when p gets larger, there are more data needs to be unlearned, yielding more re-computation cost for gradient and hessian. On the other hand, the speed-up ratio is higher in complex dataset (CIFAR-10) than simple dataset (F-MNIST). For a complex dataset, retraining process requires more recalculations of gradient information for each sample, which greatly increases the time cost of retraining from scratch. Thus, our Appro-Fun algorithm can achieve better unlearning efficiency than the baseline method and deliver higher speed-up ratio on complex datasets than simple ones.

5.7 Summary

In this chapter, we study the novel federated unlearning problem. As the first solution of exact federated unlearning, we design a Q-FL algorithm that supports exact unlearning, and then propose the Exact-Fun algorithm to achieve unlearning. In addition, we analyze

the convergence upper bound of proposed Q-FL algorithm, and give the analytical retraining probability of the Exact-Fun algorithm. In Appro-Fun, local model unlearning and federated model perturbation methods are designed by leveraging approximate Newton's updating and differential privacy, respectively. We theoretically approve the performance guarantee of Appro-Fun in terms of training convergency. Extensive experiments are conducted on real datasets and show that our algorithms outperform the state-of-the-art baseline in terms of effectiveness and efficiency.

CHAPTER 6
FUTURE WORK

6.1 Future Research Plans

For my future research, I will continue to focus on federated learning from personalization and fairness aspects, and also I will study the privacy and security of self-supervised learning.

6.1.1 Future Work 1: Personalization and Fairness in Federated Learning

Federated learning has already achieved numerous successes, given the global server holding a generic model collaboratively trained from local clients. However, the general federated learning approach faces several fundamental challenges: (i) poor convergence on highly heterogeneous data or devices, (ii) lack of personalized solution, and (iii) in sufficient fairness on minority. These issues deteriorate the performance of the global federated learning model on individual clients in the presence of heterogeneous local data distributions and may even disincentivize affected clients from joining the federated learning process. Compared with traditional federated learning, some challenges need to be solved towards personalization and fairness.

- Reducing data heterogeneity and imbalance. Data heterogeneity and class imbalance on different local clients are the main reasons of personalized federated learning for diverse users. Therefore, an efficient way to strengthen the performance of federated learning and personalization is to adjust federated learning framework through client selection, meta-learning, or transfer learning, to get a consensus between the server

and clients, so that the trained local model can achieve better performance on local private data by shifting from the global model.

- Devise personalized local algorithms. Due the diversity of local devices, the deployed local models might have different structures as per computation power. Therefore, the goal of personalization is to build personalized models by modifying the FL model aggregation process or applying different local learning algorithm in setting. We can either provide a personalized model architecture tailored to each client based on its power and data, or leverage client relationships to improve personalized model performance from related clients.

- Define and execute fair prediction. As federated learning technologies become more widely adopted by businesses to support decisionmaking, there has been a growing interest in developing methods to ensure fairness in order to avoid undesirable ethical and social implications. Current approaches do not adequately address the unique set of fairness-related challenges presented in federated learning, which include new sources of bias introduced by the diversity of participating clients due to unequal local data sizes, activity patterns, location, and connection quality. The study of fairness in federated learning is still in its infancy and the framing of fairness has not yet been well-defined. As federated learning approaches maturity, advances in improving fairness will become increasingly important in order for federated learning to be adopted at scale.

6.1.2 Future Work 2: Privacy and Security in Self-Supervised Learning

Self-Supervised learning method has made their own story in many applications, because they can take advantages of unlimited unlabeled data from all over the digital world. But, it is this advantage that leads self-supervised learning vulnerable to privacy and security related issues. The training data collected from Internet may be maliciously modified, which cause adversarial impact on self-supervised models. Besides, the privacy of collected data is not formally defined, resulting in unexpected privacy leakage for the data owners. Targeting on these issues of self-supervised learning, some corresponding research can be proposed.

- Adversarial attacks on self-supervised learning. As a machine learning paradigm, self-supervised model is inevitably vulnerable to adversarial attacks. And the unlimited data from Internet is adequate resource for attacker to design various adversarial attacks to break the trained models. In contrast, investigating attacks on self-supervised model can help us understand the explainability of deep learning and self-supervised learning as well.

- Defense for adversarial attacks in self-supervised learning. Like the attack and defense in tradition machine learning, the arm race in self-supervised learning exists. To get a reliable and secure self-supervised learning model, it is necessary for us to design robust learning algorithms or efficient methods to detect maliciously changed input data.

- Privacy leakage from data and model. Any learned model trained from a dataset

captures the statistical information of its training data samples, but the privacy leakage in self-supervised learning model has not been investigated yet. Figuring out the privacy leakage and how privacy is leaked can not only help us build trustworthy self-supervised models but also encourage data owners' contribution for better model development.

CHAPTER 7
CONCLUSION

In this dissertation, we study the problems of privacy preservation in federated learning framework, which is a popular machine learning framework in current AIoT applications. As the users' data contains highly private information of all individuals, protecting privacy and defending existing privacy attacks are with great importance. On the premise of such, we identify three significant privacy leakage scenarios and pose attention on how to design privacy-preserving solutions in there scenarios. Our methods handle different applications and needs, including private data inference, private data generation, and private data deletion. And promising future works are under going to produce more diverse and better solution in this direction.

Firstly, we analyzes the issue of privacy leakage and proves the performance upper bound of privacy inference attack in FL with non-i.i.d. data, for the first time in literature. This analysis motivates us to develop a novel mechanism, 2DP-FL, for preserving private information with ensuring differential privacy. Besides, the noise addition in 2DP-FL can be flexibly set according to different application requirements, and the upper-bounded convergence of 2DP-FL can guarantee its learning performance.

Secondly, we solve the problem of distributed data generation with multiple heterogeneous data sources in IoT. To this end, we design a hierarchical distributed generative framework with the consideration of IoT features. Based on this framework, feature related data generation model and label related data generation model are proposed, which can solve the

above data generation problems successfully in a synchronous manner or an asynchronous manner.

Thirdly, we study the novel federated unlearning problem. As the first solution of exact federated unlearning, we design a Q-FL algorithm that supports exact unlearning, and then propose the Exact-Fun algorithm to achieve unlearning. In addition, we analyze the convergence upper bound of proposed Q-FL algorithm, and give the analytical retraining probability of the Exact-Fun algorithm. In Appro-Fun, local model unlearning and federated model perturbation methods are designed by leveraging approximate Newton's updating and differential privacy, respectively. We theoretically approve the performance guarantee of Appro-Fun in terms of training convergence.

All of the proposed solutions are thoroughly discussed and validated through extensive evaluations. We also discuss some topics for future study in this dissertation in federated learning and security/privacy aspects. Overall, this dissertation provides a body of solutions for the privacy preservation in federated learning. These solutions comprehensively address the novel challenges for privacy preservation posed by the research fields and communities. We believe this study will serve as a strong reference for the tasks of federated learning, privacy preservation, and even IoT-related publications. Additionally, this dissertation will inspire future efforts to publish some relevant academic outputs.

ACKNOWLEDGMENTS

This dissertation is partially supported by National Science Foundation (NSF) under grant No. 1912753, 1704287, 1829674, 1741277, and 2011845.

REFERENCES

[1] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Network*, vol. 32, no. 4, pp. 8–14, 2018.

[2] H. Xu, W. Li, and Z. Cai, "Analysis on methods to effectively improve transfer learning performance," *Theoretical Computer Science*, vol. 940, pp. 90–107, 2023.

[3] K. Li, G. Lu, G. Luo, and Z. Cai, "Seed-free graph de-anonymiztiation with adversarial learning," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 745–754.

[4] A. M. V. V. Sai and Y. Li, "A survey on privacy issues in mobile social networks," *IEEE Access*, vol. 8, pp. 130 906–130 921, 2020.

[5] A. I. Khan and S. Al-Habsi, "Machine learning in computer vision," *Procedia Computer Science*, vol. 167, pp. 1444–1451, 2020.

[6] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 2, pp. 604–624, 2020.

[7] Z. Xiong, W. Li, Q. Han, and Z. Cai, "Privacy-preserving auto-driving: a gan-based approach to protect vehicular camera data," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 668–677.

[8] Z. Xiong, H. Xu, W. Li, and Z. Cai, "Multi-source adversarial sample attack on au-

tonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2822–2835, 2021.

[9] Z. Xiong, Z. Cai, Q. Han, A. Alrawais, and W. Li, "Adgan: Protect your location privacy in camera data of auto-driving vehicles," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6200–6210, 2020.

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

[11] Z. Cai, X. Zheng, J. Wang, and Z. He, "Private data trading towards range counting queries in internet of things," *IEEE Transactions on Mobile Computing*, 2022.

[12] Z. Cai and T. Shi, "Distributed query processing in the edge-assisted iot data monitoring system," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 679–12 693, 2020.

[13] X. Zheng, L. Tian, and Z. Cai, "A fair and rational data sharing strategy toward two-stage industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 1088–1096, 2022.

[14] H. Xu, Z. Cai, D. Takabi, and W. Li, "Audio-visual autoencoding for privacy-preserving video streaming," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1749–1761, 2021.

[15] K. Li, G. Luo, Y. Ye, W. Li, S. Ji, and Z. Cai, "Adversarial privacy-preserving graph embedding against inference attack," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6904–6915, 2020.

[16] A. M. V. V. Sai, K. Zhang, and Y. Li, "User motivation based privacy preservation in location based social networks," in *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*. IEEE, 2021, pp. 471–478.

[17] Y. Huang, Y. J. Li, and Z. Cai, "Security and privacy in metaverse: A comprehensive survey," *Big Data Mining and Analytics*, vol. 6, no. 2, pp. 234–247, 2023.

[18] Q. Han, Z. Xiong, and K. Zhang, "Research on trajectory data releasing method via differential privacy based on spatial partition," *Security and Communication Networks*, vol. 2018, 2018.

[19] G. Lu, Z. Xiong, R. Li, N. Mohammad, Y. Li, and W. Li, "Defeat: A decentralized federated learning against gradient attacks," *High-Confidence Computing*, p. 100128, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295223000260

[20] L. Jiang, Y. Yan, Z. Tian, Z. Xiong, and Q. Han, "Personalized sampling graph collection with local differential privacy for link prediction," *World Wide Web*, pp. 1–21, 2023.

[21] G. Lu, Z. Xiong, R. Li, and W. Li, "Decentralized federated learning: A defense against gradient inversion attack," in *Wireless Internet: 15th EAI International Conference, WiCON 2022, Virtual Event, November 2022, Proceedings*. Springer, 2023, pp. 44–56.

[22] Z. Xiong, Z. Cai, C. Hu, D. Takabi, and W. Li, "Towards neural network-based commu-

nication system: attack and defense," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[23] G. Lu, Z. Xiong, J. Meng, and W. Li, "Pairwise gaussian graph convolutional networks: Defense against graph adversarial attack," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 4371–4376.

[24] B. Xie, H. Xu, Z. Xiong, Y. Li, and Z. Cai, "A self-supervised purification mechanism for adversarial samples," in *2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE, 2022, pp. 501–509.

[25] G. Li, G. Yin, Z. Xiong, and F. Chen, "Cgpp-poi: A recommendation model based on privacy protection," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–20, 2021.

[26] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks: A survey toward private and secure applications," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–38, 2021.

[27] Z. Xiong, Z. Cai, D. Takabi, and W. Li, "Privacy threat and defense for federated learning with non-iid data in aiot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1310–1321, 2021.

[28] Z. Xiong and W. Li, "Federated generative model on multi-source heterogeneous data in iot," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[29] M. Nasr, R. Shokri, and A. 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)*. IEEE, 2019, pp. 739–753.

[30] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[31] L. Zhu and S. Han, "Deep leakage from gradients," in *Federated Learning*. Springer, 2020, pp. 17–31.

[32] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 691–706.

[33] B. Hitaj, G. Ateniese, and F. Perez-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*. ACM, 2017, pp. 603–618.

[34] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*. ICLR, 2019.

[35] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[36] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[37] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[38] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[39] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv preprint arXiv:1710.06963*, 2017.

[40] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134–2143, 2019.

[41] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," in *Advances in Neural Information Processing Systems*. NIPS, 2018, pp. 7564–7575.

[42] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, pp. 3454–3469, 2020.

[43] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[44] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[45] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[46] J. Yoon, J. Jordon, and M. Schaar, "Radialgan: Leveraging multiple datasets to improve target-specific predictive models using generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5699–5707.

[47] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8789–8797.

[48] H. Li, J. C. Paetzold, A. Sekuboyina, F. Kofler, J. Zhang, J. S. Kirschke, B. Wiestler, and B. Menze, "Diamondgan: unified multi-modal generative adversarial networks for mri sequences synthesis," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 795–803.

[49] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marc-

hand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[50] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 624–639.

[51] S. Zhao, M. Gong, T. Liu, H. Fu, and D. Tao, "Domain generalization via entropy regularization," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[52] Q. H. Trung Le, H. Vu, T. D. Nguyen, H. Bui, and D. Phung, "Learning generative adversarial networks from multiple data sources," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 2823–2829.

[53] J. Zhuang and D. Wang, "Geometrically matched multi-source microscopic image synthesis using bidirectional adversarial networks," *arXiv preprint arXiv:2010.13308*, 2020.

[54] S. U. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. Çukur, "Image synthesis in multi-contrast mri with conditional generative adversarial networks," *IEEE transactions on medical imaging*, vol. 38, no. 10, pp. 2375–2388, 2019.

[55] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," *Advances in neural information processing systems*, vol. 29, pp. 469–477, 2016.

[56] X. Mao and Q. Li, "Unpaired multi-domain image generation via regularized conditional gans," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2553–2559.

[57] Y. Pu, S. Dai, Z. Gan, W. Wang, G. Wang, Y. Zhang, R. Henao, and L. C. Duke, "Jointgan: Multi-domain joint distribution learning with generative adversarial nets," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4151–4160.

[58] D. J. Im, H. Ma, C. D. Kim, and G. Taylor, "Generative adversarial parallelization," *arXiv preprint arXiv:1612.04021*, 2016.

[59] C. Hardy, E. Le Merrer, and B. Sericola, "Gossiping gans: Position paper," in *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, 2018, pp. 25–28.

[60] I. P. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks," in *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=Byk-VI9eg

[61] C. Hardy, E. Le Merrer, and B. Sericola, "Md-gan: Multi-discriminator generative adversarial networks for distributed datasets," in *2019 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 2019, pp. 866–877.

[62] R. Yonetani, T. Takahashi, A. Hashimoto, and Y. Ushiku, "Decentralized learning of generative adversarial networks from non-iid data," *arXiv preprint arXiv:1905.09684*, 2019.

[63] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," *ArXiv e-prints*, pp. arXiv–1602, 2016.

[64] A. Triastcyn and B. Faltings, "Federated generative privacy," *arXiv preprint arXiv:1910.08385*, 2019.

[65] C. Fan and P. Liu, "Federated generative adversarial learning," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2020, pp. 3–15.

[66] M. Rasouli, T. Sun, and R. Rajagopal, "Fedgan: Federated generative adversarial networks for distributed data," *arXiv preprint arXiv:2006.07228*, 2020.

[67] S. Augenstein, H. B. McMahan, D. Ramage, S. Ramaswamy, P. Kairouz, M. Chen, R. Mathews *et al.*, "Generative models for effective ml on private, decentralized datasets," *arXiv preprint arXiv:1911.06679*, 2019.

[68] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, and L. Huang, "Private fl-gan: Differential privacy synthetic data generation based on federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 2927–2931.

[69] Q. Chang, H. Qu, Y. Zhang, M. Sabuncu, C. Chen, T. Zhang, and D. N. Metaxas, "Synthetic learning: Learn from distributed asynchronized discriminator gan without sharing medical image data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 856–13 866.

[70] H. Qu, Y. Zhang, Q. Chang, Z. Yan, C. Chen, and D. Metaxas, "Learn distributed gan with temporary discriminators," in *European Conference on Computer Vision*. Springer, 2020, pp. 175–192.

[71] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in

2015 IEEE Symposium on Security and Privacy. IEEE, 2015, pp. 463–480.

[72] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou, "Making ai forget you: Data deletion in machine learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[73] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 141–159.

[74] N. Aldaghri, H. Mahdavifar, and A. Beirami, "Coded machine unlearning," *IEEE Access*, vol. 9, pp. 88 137–88 150, 2021.

[75] S. Schelter, S. Grafberger, and T. Dunning, "Hedgecut: Maintaining randomised trees for low-latency machine unlearning," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1545–1557.

[76] J. Brophy and D. Lowd, "Machine unlearning for random forests," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1092–1104.

[77] Y. Wu, E. Dobriban, and S. Davidson, "Deltagrad: Rapid retraining of machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 355–10 366.

[78] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *35th AAAI Conference on Artificial Intelligence, AAAI 2021*. Association for the Advancement of Artificial Intelligence, 2021, pp. 11 516–11 524.

[79] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites, "Adaptive

machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[80] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Algorithmic Learning Theory*. PMLR, 2021, pp. 931–962.

[81] E. Ullah, T. Mai, A. Rao, R. A. Rossi, and R. Arora, "Machine unlearning via algorithmic stability," in *Conference on Learning Theory*. PMLR, 2021, pp. 4126–4142.

[82] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3832–3842.

[83] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9304–9312.

[84] Z. Izzo, M. Anne Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 2008–2016.

[85] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[86] S. Zhu, W. Li, H. Li, L. Tian, G. Luo, and Z. Cai, "Coin hopping attack in blockchain-

based iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4614–4626, 2018.

[87] X. Zheng, L. Tian, G. Luo, and Z. Cai, "A collaborative mechanism for private data publication in smart cities," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7883–7891, 2020.

[88] Z. Cai, X. Zheng, and J. Wang, "Efficient data trading for stable and privacy preserving histograms in internet of things," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–10.

[89] Z. He, A. M. V. V. Sai, Y. Huang, D. Seo, H. Zhang, and Q. Han, "Differentially private approximate aggregation based on feature selection," *Journal of Combinatorial Optimization*, vol. 41, pp. 318–327, 2021.

[90] R. Hu, Y. Guo, E. P. Ratazzi, and Y. Gong, "Differentially private federated learning for resource-constrained internet of things," *arXiv preprint arXiv:2003.12705*, 2020.

[91] A. Sonee and S. Rini, "Efficient federated learning over multiple access channel with differential privacy constraints," *arXiv preprint arXiv:2005.07776*, 2020.

[92] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[93] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy*. IEEE, 2017, pp. 3–18.

[94] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2018.

[95] B. Wu, S. Zhao, C. Chen, H. Xu, L. Wang, X. Zhang, G. Sun, and J. Zhou, "Generalization in generative adversarial networks: A novel perspective from privacy protection," in *Advances in Neural Information Processing Systems*. NIPS, 2019, pp. 307–317.

[96] J. Zhao, Y. Chen, and W. Zhang, "Differential privacy preservation in deep learning: Challenges, opportunities and solutions," *IEEE Access*, vol. 7, pp. 48 901–48 911, 2019.

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

[98] J. Wang, Z. Xiong, Q. Han, X. Han, and D. Yang, "Top-k socially constrained spatial keyword search in large siot networks," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9280–9289, 2021.

[99] Y. Lv, Y. Chen, L. Li, and F.-Y. Wang, "Generative adversarial networks for parallel transportation systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 3, pp. 4–10, 2018.

[100] M. E. Tschuchnig, C. Ferner, and S. Wegenkittl, "Sequential iot data augmentation using generative adversarial networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4212–4216.

[101] Q. Chen, W. Wang, K. Huang, S. De, and F. Coenen, "Multi-modal generative adversarial networks for traffic event detection in smart cities," *Expert Systems with Applications*, vol. 177, p. 114939, 2021.

[102] M. Mohammadi, A. Al-Fuqaha, and J.-S. Oh, "Path planning in support of smart mobility applications using generative adversarial networks," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 878–885.

[103] I. Vaccari, V. Orani, A. Paglialonga, E. Cambiaso, and M. Mongelli, "A generative adversarial network (gan) technique for internet of medical things data," *Sensors*, vol. 21, no. 11, p. 3726, 2021.

[104] M. Fabbri, S. Calderara, and R. Cucchiara, "Generative adversarial models for people attribute recognition in surveillance," in *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2017, pp. 1–6.

[105] Q. Shi, X. Liu, and X. Li, "Road detection from remote sensing images by generative adversarial networks," *IEEE access*, vol. 6, pp. 25 486–25 494, 2017.

[106] T. Ganokratanaa, S. Aramvith, and N. Sebe, "Anomaly event detection using generative adversarial network for surveillance videos," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 1395–1399.

[107] Y. Lee, J. Yun, Y. Hong, J. Lee, and M. Jeon, "Accurate license plate recognition and

super-resolution using a generative adversarial networks on traffic surveillance video," in *2018 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2018, pp. 1–4.

[108] J. Wang, Z. Cai, Y. Li, D. Yang, J. Li, and H. Gao, "Protecting query privacy with differentially private k-anonymity in location-based services," *Personal and Ubiquitous Computing*, vol. 22, pp. 453–469, 2018.

[109] Y. Wang, Z. Cai, X. Tong, Y. Gao, and G. Yin, "Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems," *Computer Networks*, vol. 135, pp. 32–43, 2018.

[110] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart internet of things systems: a consideration from a privacy perspective," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 55–61, 2018.

[111] X. Zheng, A. Chen, G. Luo, L. Tian, and Z. Cai, "Privacy-preserved distinct content collection in human-assisted ubiquitous computing systems," *Information Sciences*, vol. 493, pp. 91–104, 2019.

[112] K. Li, L. Tian, W. Li, G. Luo, and Z. Cai, "Incorporating social interaction into three-party game towards privacy protection in iot," *Computer Networks*, vol. 150, pp. 90–101, 2019.

[113] X. Zheng, Z. Cai, G. Luo, L. Tian, and X. Bai, "Privacy-preserved community discovery in online social networks," *Future Generation Computer Systems*, vol. 93, pp. 1002–1009, 2019.

[114] X. Tao, Y. Peng, F. Zhao, C. Yang, B. Qiang, Y. Wang, and Z. Xiong, "Gated recurrent unit-based parallel network traffic anomaly detection using subagging ensembles," *Ad Hoc Networks*, vol. 116, p. 102465, 2021.

[115] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 703–716.

[116] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[117] J. Wang, Z. Cai, and J. Yu, "Achieving personalized k-anonymity-based content privacy for autonomous vehicles in cps," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4242–4251, 2019.

[118] H. Xu, Z. Cai, and W. Li, "Privacy-preserving mechanisms for multi-label image recognition," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 4, pp. 1–21, 2022.

[119] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[120] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015,

pp. 97–105.

[121] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[122] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

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

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

[125] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *European conference on computer vision*. Springer, 2016, pp. 597–613.

[126] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 192–199.

[127] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[128] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[129] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 901–10 911.

[130] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 165–172.

[131] Q.-Y. Jiang, Y. He, G. Li, J. Lin, L. Li, and W.-J. Li, "Svd: A large-scale short video dataset for near-duplicate video retrieval," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5281–5289.

[132] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)," *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[133] D. o. J. State of California, "the california consumer privacy act (ccpa)," 2000. [Online]. Available: https://oag.ca.gov/privacy/ccpa

[134] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, 2017, pp. 587–601.

[135] S. Garfinkel, J. M. Abowd, and C. Martindale, "Understanding database reconstruction attacks on public data," *Communications of the ACM*, vol. 62, no. 3, pp. 46–53, 2019.

[136] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *arXiv preprint arXiv:2101.03961*, 2021.

[137] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[138] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[139] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[140] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[141] X. Zhou, "On the fenchel duality between strong convexity and lipschitz continuous gradient," *arXiv preprint arXiv:1803.06573*, 2018.

[142] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.

[143] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[144] A. Ly, M. Marsman, J. Verhagen, R. P. Grasman, and E.-J. Wagenmakers, "A tutorial on fisher information," *Journal of Mathematical Psychology*, vol. 80, pp. 40–55, 2017.

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

[146] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated

learning: An efficient realization with rapid retraining," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1749–1758.

[147] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," *arXiv preprint arXiv:2201.09441*, 2022.

[148] S. Fu, F. He, and D. Tao, "Knowledge removal in sampling-based bayesian inference," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. [Online]. Available: https://openreview.net/forum?id=dTqOcTUOQO

[149] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "When machine unlearning jeopardizes privacy," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 896–911.

[150] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "Federaser: Enabling efficient client-level data removal from federated learning models," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–10.