FedRecovery: Differentially Private Machine Unlearning for Federated Learning Frameworks

Lefeng Zhang, Tianqing Zhu, Haibin Zhang, Ping Xiong, and Wanlei Zhou, *Senior Member, IEEE*

***Abstract*— Over the past decades, the abundance of personal data has led to the rapid development of machine learning models and important advances in artificial intelligence (AI). However, alongside all the achievements, there are increasing privacy threats and security risks that may cause significant losses for data providers. Recent legislation requires that the private information about a user should be removed from a database as well as machine learning models upon certain deletion requests. While erasing data records from memory storage is straightforward, it is often challenging to remove the influence of particular data samples from a model that has already been trained. Machine unlearning is an emerging paradigm that aims to make machine learning models "forget" what they have learned about particular data. Nevertheless, the unlearning issue for federated learning has not been completely addressed due to its special working mode. First, existing solutions crucially rely on retraining-based model calibration, which is likely unavailable and can pose new privacy risks for federated learning frameworks. Second, today's efficient unlearning strategies are mainly designed for convex problems, which are incapable of handling more complicated learning tasks like neural networks. To overcome these limitations, we took advantage of differential privacy and developed an efficient machine unlearning algorithm named FedRecovery. The FedRecovery erases the impact of a client by removing a weighted sum of gradient residuals from the global model, and tailors the Gaussian noise to make the unlearned model and retrained model statistically indistinguishable. Furthermore, the algorithm neither requires retraining-based fine-tuning nor needs the assumption of convexity. Theoretical analyses show the rigorous indistinguishability guarantee. Additionally, the experiment results on real-world datasets demonstrate that the FedRecovery is efficient and is able to produce a model that performs similarly to the retrained one.**

***Index Terms*— Machine unlearning, differential privacy, federated learning.**

I. Introduction

MACHINE unlearning is an emerging challenge that both the data science and policy communities are trying to grapple with. It stems from the increasingly urgent requirement of individuals for data protection, and is one of the centrepieces of machine learning research [1], [2]. Machine unlearning aims to post-process a trained model to remove the influence of specific training sample(s), such that the output model "looks as if it has never seen the unlearned data before". In fact, the development of machine unlearning is not only motivated by privacy and security issues, but also driven by legislation. For example, the European Union's General Data Protection Regulation (GDPR) [3] and the previous *Right to Be Forgotten* [4] entails the right for people to withdraw their consent to any processing operation on their data in certain contexts. Similar statements can be found in Canada's Consumer Privacy Protection Act (CPPA) [5] and California's Consumer Privacy Act (CCPA) [6], which obligate companies and organizations to delete individuals' information from a trained model upon request.

Federated learning [7] is a promising distributed machine learning paradigm that provides privacy-preserving learning solutions. The core idea of federated learning is to train a machine learning model on separate datasets that are distributed across different devices or parties. During the training process, only the model's parameters or gradients are shared, each client's data is kept invisible to other parties. Therefore, the model can be trained without disclosing the clients' raw data, and thus their data privacy can be preserved [8].

Machine unlearning plays an important role in addressing privacy and security issues in federated learning. Theoretically, unlearning an individual from a model is an ideal way to prevent information leakage from model inversion attacks [9] and membership inference attacks [10]. Moreover, unlearning techniques are also instrumental in eliminating the influence of data poisoning attacks [11] performed by malicious clients. For the same reason, machine unlearning can be used to update models if the previous training data is outdated or of low quality [12]. Intuitively, a naive way to remove an individual's influence from a model is to retrain the model from scratch on the remaining data. However, retraining from scratch leads to prohibitively expensive computation costs [13]. Worse still, in federated learning, it is even impossible to perform retraining.

Achieving machine unlearning in federated learning is rather challenging due to its unique working mode. Firstly, the clients in federated learning are continuously changing. The server can hardly recall previous clients to perform an unlearning operation, let alone retrain from scratch. Secondly, the server has no access to the data samples to be unlearned. Therefore, existing centralized unlearning algorithms [14], [15] that are parameterized by unlearned data

Manuscript received 1 October 2022; revised 18 May 2023 and 7 July 2023; accepted 18 July 2023. Date of publication 21 July 2023; date of current version 8 August 2023. This work was supported by the Australian Research Council Discovery under Grant DP200100946 and Grant DP230100246. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. George Theodorakopoulos. *(Corresponding author: Tianqing Zhu.)*

Lefeng Zhang, Tianqing Zhu, and Wanlei Zhou are with the Centre for Cyber Security and Privacy and the School of Computer Science, The University of Technology, Sydney, NSW 2007, Australia (e-mail: lefeng.zhang@student.uts.edu.au; Tianqing.Zhu@uts.edu.au; Wanlei.Zhou@uts.edu.au).

Haibin Zhang is with the Department of Operational Research and Scientific Computation, Faculty of Science, Beijing University of Technology, Beijing 100021, China (e-mail: zhanghaibin@bjut.edu.cn).

Ping Xiong is with the Department of Computer Science and Technology, School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China (e-mail: pingxiong@zuel.edu.cn).

Digital Object Identifier 10.1109/TIFS.2023.3297905

will not work in federated learning. Finally, the communication overhead between the server and clients is limited. As a result, the unlearning algorithms [16], [17] derived from second-order optimization like the L-BFGS [18] are no longer efficient.

In federated learning, the former updates of clients have an implicit and increasing influence on subsequent model updates during the training process, which is known as the incremental effect [19]. Therefore, unlearning algorithms have to disentangle the dependence of the unlearned client from the global model with the least damage to the contributions of other clients [20]. To address this issue, existing works [12], [21], [22] roughly remove the gradients or model weights concerning the unlearned client, and crucially rely on post-processed fine-tuning to repair the damage caused by unlearning operations. However, we claim that the fine-tuning process is in fact impractical and illegal in real-world scenarios. Firstly, fine-tuning also leads to considerable computation and communication costs. Secondly, as the server's computing power is limited, distributing the model for fine-tuning may bring new privacy and security risks to the unlearned client, which is strictly prohibited by the previously mentioned legislation. Therefore, we ask that *can we efficiently find a model that performs similarly to the retrained one?*

In this paper, we proposed a differentially private machine unlearning algorithm, FedRecovery, which takes advantage of clients' historical submissions to reproduce a model that is almost irrelevant to the client to be unlearned. The FedRecovery mathematically quantified the incremental effect by introducing the concept of gradient residual. It eliminates the influence of the unlearned client by removing a weighted sum of gradient residuals from the global model, where the weights are evaluated based on clients' actual contributions to decreasing the global loss. To provide a rigorous unlearning guarantee, we adopted the notion of approximate statistical indistinguishability to limit the difference between the unlearned model and the unavailable retrained model. Specifically, we first derived the upper bound of the distance between the unlearned model and the retrained model, and then leveraged the Gaussian mechanism to mask this gap in parameter space. As modern federated learning tasks generally start with pre-trained models given by the server [51], the discrepancy between models will be small, and the amount of noise can be precisely calibrated. FedRecovery neither requires the convexity assumption on loss functions nor needs any retraining based fine-tuning that may bring extra communication and computation costs. Therefore, it is suitable for today's federated learning framework and is competent to handle complex unlearning tasks.

The main contributions of this paper are summarized as follows.

- We mathematically quantified the incremental effect in federated learning, which provides a theoretical guideline to remove the influence of a specific participating client.
- We proposed an efficient unlearning algorithm, FedRecovery, which reproduces a model that is indistinguishable from the retrained one by only exploring clients' historical submissions.
- We gave theoretical proofs and analyses of the proposed unlearning algorithm. We derived the upper bound of distance between the unlearned model and the retrained model, and achieved a rigorous unlearning guarantee by adding carefully calibrated noises.
- We conducted experimental simulations to show the indistinguishability between the models generated by FedRecovery. Further experiments demonstrate the effectiveness of the proposed unlearning algorithm.

The rest of this paper is organized as follows. In Section II, we review the related literature on machine unlearning. In Section III, we introduce the background of this paper and formally define the problems to be addressed. In Section IV, we detail the proposed FedRecovery algorithm and the corresponding learning mechanisms. Section V presents theoretical analyses of the unlearning guarantee. Section VI contains experimental results, and Section VII concludes the paper.

II. Related Work

A. Review of Related Works

Considering unlearning strategies, current machine unlearning techniques could be classified into three categories: training-based unlearning, tuning-based unlearning and model-based unlearning.

1) Training-Based Machine Unlearning: Training-based machine unlearning aims to speed up the retraining process or use fewer training rounds to improve model accuracy. Bourtoule et al. [23] proposed an unlearning framework SISA that partitions the training data into multiple shards. Each shared creates a unique model checkpoint so that the retraining can be invoked from the intermediate state. Liu et al. [19] developed a rapid retraining algorithm to remove data samples from a federated learning model. They employed the Fisher Information Matrix (FIM) to approximate the Hessian for Quasi-Newton optimization, which is computationally cheaper than the traditional L-BFGS algorithm [24]. Liu et al. [21] presented a client-level data removal algorithm to eliminate the influence of a client's data. The proposed algorithm relies on the storage of the central server to retain historical submissions for each client, which are then calibrated by retrained parameters to speed up the unlearning process. Wang et al. [22] proposed a pruning-based category-level unlearning algorithm. They leveraged TF-IDF [25] to evaluate the relevance between channels and classes, and erased the discriminative channels of the category to be unlearned. Their model's performance is restored by fine-tuning the pruned model on the remaining dataset. Wu et al. [12] focused on addressing the incremental effect when they removed specific clients. They directly subtracted one client's updates from the global model's parameters, and further employed knowledge distillation to improve model accuracy. The original global model is adopted as the teacher model to recover the damage caused by unlearning operations.

2) Tuning-Based Machine Unlearning: Tuning-based machine unlearning directly modifies the parameters of a trained model to erase the influence of unlearned data, where the modification is usually calibrated by particular influence

functions. Golatkar et al. [17] used the Hessian-gradient product to represent the contribution of the data to be unlearned. They performed a reverse Newton step on the trained model to scrub the information from the unlearned data under a local quadratic approximation. Similarly, Guo et al. [14] adopted influence theory [26] to evaluate the impact of a training sample, and employed differentially private loss perturbation [27] to mask the gradient discrepancy caused by removing a particular data point. Their proposed ϵ-certificated removal is in fact a relaxation of ϵ-differential privacy on a particular data record. Sekhari et al. [16] followed Newton's method to optimize the empirical loss, and deleted the influence of the targeted data by subtracting an inversed Hessian from the trained model. Golatkar et al. [28] derived an optimal scrubbing scheme based on the theorem of neural tangent kernels (NTK) [29]. Their algorithm modifies the model's parameters as a function of the tangent kernel, which is more component in deep learning due to the over-parameterization of deep models.

3) Model-Based Machine Unlearning: Model-based machine unlearning tailors the intrinsic rationale of learning algorithms and creates specific unlearning strategies for particular models. Cao and Yang [30] focused on statistical query learning, where a model is trained based on summations instead of training samples. In this scenario, the unlearning algorithm only needs to recompute the summations that contain unlearned data to remove their influence from the model. Ginart et al. [13] developed a deletion algorithm for k-means clustering problems. The key idea is to randomize the algorithm's output such that the model after deleting some points is distributionally equal to the model obtained by retraining. Neel et al. [31] observed that the optimizer of convex models trained with one data sample deleted stays close to the optimizer without deletion. Therefore, they proposed to use gradient descent to achieve unlearning, where the update starts from the previously trained model. Chen et al. [32] customized data partition strategies for recommendation systems. They divided the training data into balanced groups to ensure that the collaborative information was retained.

B. Discussion of Related Works

The above works have two common limitations. First, the performance of most federated unlearning algorithms crucially relies on retraining-based fine-tuning and calibration. However, in practice, the computing power of the server is limited, recruiting new participants to calibrate an unlearned model may likely put previous clients' privacy at risk. Second, the differential privacy based unlearning algorithms heavily rely on the convex assumption on loss functions, which mathematically guarantees that the optimizer of the empirical risk with/without the unlearned client only changes by a small amount in parameter space. However, the objective functions of complex tasks are possibly non-convex, such as neural networks. Therefore, it is worth exploring unlearning algorithms without the convexity assumption.

To this end, we proposed an unlearning algorithm without further fine-tuning or calibration. In addition, the algorithm runs only based on the Lipschitzness of loss functions, which is more adaptive for real-world problems.

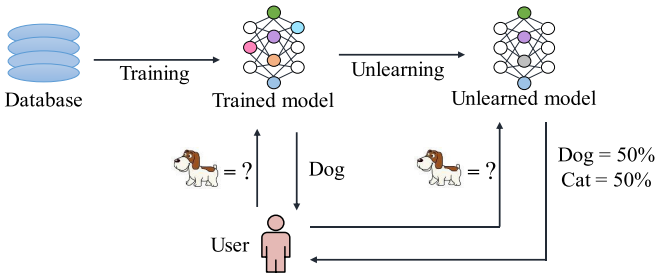


Fig. 1. Model training, predicting and unlearning processes.

III. Preliminaries

This section introduces the necessary background knowledge and basic concepts in machine unlearning, federated learning, and differential privacy. The formal definition of the unlearning problem to be addressed is also presented.

A. Machine Unlearning

Machine unlearning is the study of removing the influence of certain data points from a learned model [28]. The demand for machine unlearning generally stems from real-world situations where data owners need to revoke their contributions to a learned model due to privacy or security issues [30]. In addition, it is legally required that individuals have the right to decide if their personal data can be used by any entity for any purpose. For example, the GDPR and the *Right to be Forgotten* grant the right for users to withdraw consent to the use of their data from companies and organizations under certain circumstances. Therefore, it is important to explore how to fulfill these legal aims in machine learning area.

Machine unlearning deals with both homogeneous (e.g., to unlearn a set of samples) and heterogeneous (e.g., to unlearn a specific class) unlearning requests. Intuitively, the naive way to unlearn targeted data from a model is simply to retrain the model from scratch using the remaining data. However, retraining from scratch results in prohibitively expensive computation costs, and is incapable of handling situations where unlearning requests come frequently. In addition, in some scenarios, such as federated learning, it is unable to perform retraining as the previous participating clients may have lost connection due to the inherently distinct learning mode.

The relationship between model training and machine unlearning is described in Figure 1, including the training, predicting (aka. inference), and unlearning processes. At the beginning, a learnable model is initialized and trained using data that may be collected from multiple parties. The training process results in a model that is useful for AI tasks, and also generates internal states or data such as gradients, Hessian matrices, and intermediate model parameters. When a user submits an unlearning request, the model owner employs some unlearning oracle based on the cached data to reproduce an unlearned model. Ideally, the unlearned model should completely forget what it has learned about the data to be

unlearned. For example, in the dog-cat image classification task, if the data of dogs needs to be unlearned, then the unlearned model will not be able to recognize dogs anymore.

In fact, making a model forget some training samples is quite trivial. For example, one can simply add a large amount of noise to the model's parameters, which definitely removes the influence of the unlearned data – but also destroys the model's utility on the remaining data. Therefore, the challenge of machine unlearning lies in how to unlearn the targeted data while avoiding catastrophic forgetting [33] – a phenomenon where a model rapidly loses accuracy on some tasks when fine-tuned for others.

B. Federated Learning

Federated learning harnesses the power of a large number of distributed clients to collaboratively train a global model without revealing their local data to a central server [34]. In federated learning, each client c_i possesses a private database, denoted as D_i, and is equipped with a computing device. The server S wishes to train a global model with training data distributed across these clients. The clients who participate in the training are responsible for finding the parameter w that minimizes a given loss function. The loss function of client c_i with database D_i is

$$f_i(w) = \frac{1}{|D_i|} \sum_{j \in D_i} l_j(w), \tag{1}$$

where $l_j(w)$ is the loss incurred on the data sample j.

In each training round, the server obtains an aggregated model parameter by computing the weighted average over w_i from each client.

$$w_{avg} = \sum_{i=1}^{n} \alpha_i w_i, \quad \alpha_i = \frac{|D_i|}{\sum_{i=1}^{n} |D_i|}. \tag{2}$$

The server broadcasts the aggregated model parameters to each client, who updates their local model and starts the next round of training. The learning process can be formulated as an optimization problem, aimed at finding the parameters that minimizes of the global loss function

$$w^* = \arg\min_{w} \alpha_i f_i(w). \tag{3}$$

After enough rounds of local training and parameter aggregation, the solution of the optimization problem will converge to the optimal value of the global model.

C. Differential Privacy and (ϵ, β)-Indistinguishability

Differential privacy [35], [36] is a rigorous mathematical privacy model that limits the influence each individual can make in statistical data analysis. The aim of differential privacy is to bound the differences in a query between two datasets that differ in a single entry [37]. Recently, differential privacy has also been proven to provide useful properties in the area of machine unlearning [13], [14].

Similarly, the (ϵ, β)-indistinguishability is a notion of approximate statistical indistinguishability developed from differential privacy literature, which mathematically requires that two random variables are indistinguishable for reasonable values of ϵ and β.

TABLE I
THE MAIN NOTATIONS USED IN THE PAPER

Notations	Explanation
f_i, F	the local and global loss function
∇f_i, ∇F	the gradients of functions f_i and F
n	the number of participating clients
c_i, C_n	the i-th client and the set that contains n clients
c_{i_u}	the client who submits unlearning request
L	the Lipschitz constant
$[n]$	the set $\{1, 2, \ldots, n\}$
w_t	the trained global model's parameter
$\tilde{w}_t$	the global model's parameter trained without c_{i_u}
$\hat{w}_t$	the output of perturbed federated learning
$\bar{w}_t$	the global model's parameter removing c_{i_u}'s influence
δ_t	the gradient residual at round t
η_t	the learning rate at round t
ϵ, β	the privacy budget and confidence parameter
d, Δ	the Euclidean distance between parameters
$\mathcal{N}(0, \sigma^2\mathbb{I})$	the Gaussian distribution with mean 0 and std. $\sigma^2\mathbb{I}$

Definition 1 ((ϵ, β)-Indistinguishability [31]): Let X and Y be random variables over domain $\mathcal{R}$. If for every possible subset of outcome $S \subseteq \mathcal{R}$, X and Y satisfy

$$\begin{aligned} \Pr[X \in S] &\leq \exp(\epsilon) \cdot \Pr[Y \in S] + \beta, \\ \Pr[Y \in S] &\leq \exp(\epsilon) \cdot \Pr[X \in S] + \beta, \end{aligned} \tag{4}$$

then we say that X and Y are (ϵ, β)-indistinguishable.

The sensitivity of a function captures the maximum difference an individual can make on the function's output. Intuitively, it determines the magnitude of perturbation required to achieve indistinguishability.

Definition 2 (Sensitivity [35]): A function $q : D \to \mathbb{R}^k$ has sensitivity Δ if for all datasets D and D' that differ in a single entry, we have

$$\Delta = \max_{D, D'} ||q(D) - q(D')||. \tag{5}$$

One of the most widely used mechanisms that can achieve indistinguishability is the Gaussian mechanism, which adds Gaussian distributed noise to the statistical output.

Definition 3 (Gaussian Mechanism [38]): Given random variables $X \sim \mathcal{N}(\mu_1, \sigma^2\mathbb{I}_d)$ and $Y \sim \mathcal{N}(\mu_2, \sigma^2\mathbb{I}_d)$ that satisfy $\|\mu_1 - \mu_2\| \leq \Delta$, then for any $\beta > 0$, X and Y are (ϵ, β)-indistinguishable if

$$\epsilon = \frac{\Delta^2}{2\sigma^2} + \frac{\Delta}{\sigma}\sqrt{2\log(1/\beta)}. \tag{6}$$

The main notations used in the paper are summarized in Table I.

D. Problem Definition

This research aims to find a model that performs similarly to the retrained model, given a model that has already been trained by n clients and the intermediate statistics cached during training. In other words, consider that there are $C = \{c_1, \ldots, c_n\}$ clients that collaboratively trained a global model $\mathcal{M}$ under the schedule of the central server S. After the model is trained, the i_u-th client submits an unlearning request to

withdraw his consent to the use of his data. Therefore, the server S has to remove the influence of client c_{i_u}'s data from the trained model M. Ideally, the performance of the unlearned model should be comparable to that of the retrained one.

Without retraining and fine-tuning, it is generally not possible to recover an unlearned model from M that is completely independent of client c_{i_u}'s influence. Therefore, we turn to a compelling alternative: *can we design an unlearning algorithm that produces a (distribution of) model(s) that is statistically indistinguishable from the (distribution of) model(s) that would have arisen from full retraining?* Mathematically, the unlearning guarantee is formulated as the follow definition.

Definition 4 (Client-Level (ϵ, β)-Machine Unlearning): An unlearning algorithm $\mathcal{M}_U$ satisfies (ϵ, β)-machine unlearning with respect to the learning algorithm $\mathcal{M}_L$, if for any possible set of output $W \subseteq \mathbb{R}^{\mathrm{d}}$,

$$\begin{aligned} \Pr[\mathcal{M}_L(C\backslash\{c_{i_u}\}, \epsilon) \in S] &\leq e^{\epsilon} \cdot \Pr[\mathcal{M}_U(w, \Omega, \epsilon) \in S] + \beta, \\ \Pr[M_U(w, \Omega, \epsilon) \in S] &\leq e^{\epsilon} \cdot \Pr[\mathcal{M}_L(C\backslash\{c_{i_u}\} \in S] + \beta, \end{aligned} \tag{7}$$

where w represents the parameter of model M, and Ω is the set of cached statistics the server obtained from each client, such as gradients and intermediate model parameters.

Throughout the paper, we follow the real-world assumption that the server S cannot access any data of any clients. In addition, the server's computing ability is limited, such that it cannot perform any training work.

For the properties of loss functions, we assume that the local loss function $f_i(w)$ of each client is L-smooth.

Assumption 1 (L-Smoothness): A function $f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$ is L-smooth if it is differentiable and its gradient function $\nabla f : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}^{\mathrm{d}}$ is Lipschitz continuous with constant L, i.e., $\forall w_1, w_2 \in \mathbb{R}^{\mathrm{d}}$,

$$\|\nabla f(w_1) - \nabla f(w_2)\| \leq L\|w_1 - w_2\|. \tag{8}$$

Without loss of generality, each client employs the gradient descent method to find the optimal parameter, which is a popular choice in optimizing complex tasks such as neural networks. If the stochastic gradient descent method is used, we have a further assumption about the stochasticity.

Assumption 2 (The Norm of Stochastic Gradients): The expected squared norm of stochastic gradients is uniformly bounded, namely for all $i = 1, \ldots, n$ and $t = 1, \ldots, t$, there exists a $G > 0$ such that

$$\mathbb{E}_{\xi_t}\|\nabla f_i(w_t)\|^2 \leq G^2, \tag{9}$$

where ξ_t represents the sampling distribution over local data.

To improve legibility, we assume that the n-th client is the one who wishes to unlearn his data, but our unlearning algorithm works for arbitrary client.

IV. The FedRecovery Algorithm

A. Design Rationale and Algorithm Overview

The FedRecovery algorithm has two components: the perturbed learning algorithm $\mathcal{M}_L$ and the unlearning algorithm $\mathcal{M}_U$, as depicted in Figure 2. The learning algorithm $\mathcal{M}_L$ employs gradient descent to obtain a trained model, while the unlearning algorithm $\mathcal{M}_U$ removes the influence of the client who submits an unlearning request. Both of them rely on the Gaussian perturbation to achieve (ϵ, β)-machine unlearning.

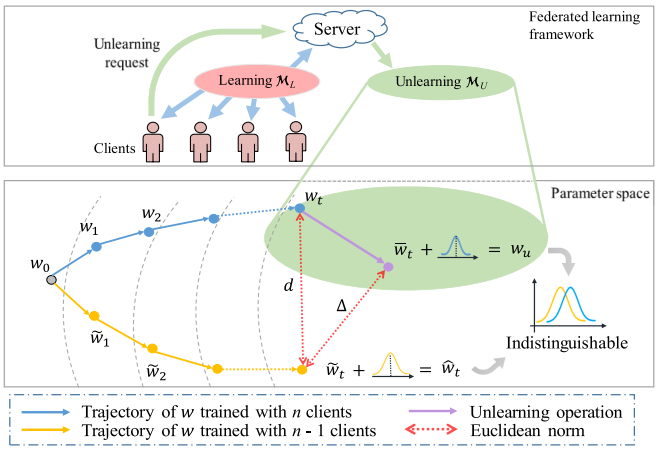


Fig. 2. The framework of the FedRecovery algorithm and its rationale in the parameter space. The learning algorithm $\mathcal{M}_L$ trains the model using the gradient descent method, which corresponds to the blue and yellow trajectories. The unlearning algorithm $\mathcal{M}_U$ removes the influence of the client who wishes to be unlearned, as shown in green. The Gaussian noise is added to guarantee the unlearned model and retrained model are indistinguishable.

At the beginning of a federated training, the server S initializes the model structure and sets a start point w_0, which are then distributed to n participating clients $C = \{c_1, \ldots, c_n\}$. During the training process, each client computes the gradient $\nabla f_i(w)$ using their local data, and the server iteratively updates the global model's parameter w. The change of w forms a trajectory in parameter space W, as shown by the blue lines. After t iterations, the server S obtains a trained model M, but unfortunately, the n-th client c_n then requires to withdraw the consent of using his data. Therefore, the server has to unlearn the influence of c_n from M by leveraging the resources at hand, namely the series of parameters $\{w_i\}_{i=0}^{t}$ and the cached gradients $\{\nabla f_i(w_j)\}$, $i \in [t]$, $j \in [n]$.

In fact, without client c_n's participation, the first $n-1$ clients would also form a descending trajectory $\{\tilde{w}_i\}_{i=1}^{t}$ in the parameter space after t training iterations, as shown by the yellow lines. The aim of FedRecovery is to find a $\bar{w} \in W$ that eliminates c_n' s contribution as much as possible by removing the gradient residuals generated from clients' historical submissions (the green line). To achieve (ϵ, β)-machine unlearning, the algorithm introduces perturbations to $\bar{w}_t$ and $\tilde{w}_t$ based on the upper bound of the distance Δ between them. Specifically, the Gaussian noise calibrated by Δ is added to $\bar{w}_t$ and $\tilde{w}_t$, and thus the unlearning guarantee can be promised by the property of (ϵ, β)-indistinguishability.

Theoretically, if the loss functions are convex, it is easy to bound the distance Δ between $\bar{w}_t$ and $\tilde{w}_t$ as the optimal point for convex functions is unique. Without the assumption of convexity, the FedRecovery algorithm derives an upper bound of $d \geq \Delta$ by exploring the smoothness of loss functions, which serves as an upper bound of Δ.

It is worth noting that the parameters $\{\tilde{w}_i\}_{i=1}^{t}$ are unknown throughout the course of the FedRecovery algorithm; only the distance between $\bar{w}_t$ and $\tilde{w}_t$ is estimated. In addition,

to achieve (ϵ, β)-machine unlearning, the result of the learning algorithm is also randomized, which differs from traditional federated learning frameworks. Finally, there are extreme cases where $\{\tilde{w}_i\}_{i=1}^t$ stay close to w_0, i.e., the training would have failed with $n-1$ clients. The FedRecovery algorithm also works in such special scenarios, but the amount of introduced noise will be consequently large.

The detailed algorithms and related concepts are explained in the subsequent sections.

B. Gradient Residual and the Perturbed Learning Algorithm

In this section, we first introduce the concept of gradient residual to theoretically demonstrate the incremental effect of parameters in federated learning, and then present the perturbed federated learning algorithm that is helpful in achieving machine unlearning.

Without loss of generality, in the t-th round of model update, the server collects gradients from n participating clients and aggregates all the updates to obtain a new global model w_t.

$$w_t = w_{t-1} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}), \quad i \in [n]. \tag{10}$$

Suppose the n-th client submits an unlearning request to the server after training. Consider the case where the n-th client does not participate in the t-th aggregation, then the parameter update at time step t will become

$$\tilde{w}_t = \tilde{w}_{t-1} - \eta \cdot \frac{1}{n-1} \sum_{i=1}^{n} \nabla f_i(\tilde{w}_{t-1}), \quad i \in [n]. \tag{11}$$

To investigate the influence of the last client, we can subtract Equation 11 from Equation 10, and have

$$\begin{aligned} & w_t - \tilde{w}_t - (w_{t-1} - \tilde{w}_{t-1}) \\ &= -\eta \cdot [\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}) - \frac{1}{n-1} \sum_{i=1}^{n-1} \nabla f_i(\tilde{w}_{t-1})] \\ &= -\frac{\eta}{n-1} \sum_{i=1}^{n-1} [\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_{t-1})] + \delta_{t-1}, \end{aligned} \tag{12}$$

where δ_{t-1} is the gradient residual that contains the information submitted by the last client in round $t-1$, which is derived from all the n clients' updates and can be computed exactly by the server.

$$\delta_{t-1} = \frac{\eta}{n} [\frac{1}{(n-1)} \sum_{i=1}^{n-1} \nabla f_i(w_{t-1}) - \nabla f_n(w_{t-1})]. \tag{13}$$

We note that although $\{\delta_i\}_{i=1}^t$ contains the information from the client who wants to unlearn his influence, it is not reasonable to subtract $\sum_{i=1}^t \delta_i$ directly from w^*. This is because the influence of each δ_t extremely differs along the course of the training process. To see this, we apply the triangular inequality in Equation 12.

$$\begin{aligned} & \|w_t - \tilde{w}_t\| - \|w_{t-1} - \tilde{w}_{t-1}\| \\ &\le \frac{\eta}{n-1} \sum_{i=1}^{n-1} \|\nabla f_i(w_{t-1}) - \nabla f_i(\tilde{w}_{t-1})\| + \|\delta_{t-1}\| \\ &\le \eta L \|w_{t-1} - \tilde{w}_{t-1}\| + \|\delta_{t-1}\|, \end{aligned} \tag{14}$$

where the last inequality is derived by introducing the smoothness of the client's loss function.

Then, we sum Equation 14 iteratively from training round 1 to t, and with the fact $\|w_0 - \tilde{w}_0\| = 0$, we have

$$\begin{aligned} \|w_t - \tilde{w}_t\| &\le \gamma^{t-1} \|\delta_0\| + \gamma^{t-2} \|\delta_1\| + \cdots + \|\delta_{t-1}\| \\ &= \sum_{i=1}^{t} \gamma^{t-i} \|\delta_{i-1}\|, \end{aligned} \tag{15}$$

where $\gamma = 1 + \eta L$ is jointly determined by the nature of the loss function (i.e., smoothness L) and the learning rate η.

Equation 15 demonstrates that the distance between w_t and $\tilde{w}_t$ is upper-bounded by the series $\{\delta_i\}_{i=0}^{t-1}$, where the influence of δ_i grows exponentially along the training process. Meanwhile, it also explains the incremental effect of parameters generated in federated learning, namely the preceding parameters have more influence than the later ones.

Theoretically, to limit the upper bound of $\|w_t - \tilde{w}_t\|$, it is necessary to set $\eta < \frac{1}{tL}$, such that $\lim_{t\to\infty}(1+\eta L)^t = 1$ and $\|w_t - \tilde{w}_t\| \le \sum_{i=1}^t \|\delta_{i-1}\|$. However, it is not possible or practical to run federated learning for infinitely many rounds, and thus the series $\{\gamma^{t-i}\}$ may not converge.

Intuitively, the best way of eliminating the n-th client's influence is to remove the weighted sum of $\{\delta_i\}_{i=0}^{t-1}$ from w_t with weights γ^{t-i}, which will ideally produce a $\hat{w}_t$ that is closer to $\tilde{w}_t$. However, this is not feasible in practice. First and foremost, computing the Lipschitzness and smoothness of a neural network is generally an NP-hard problem [39]. Therefore, we cannot precisely obtain the constant L in polynomial time (assuming that P≠NP). Second, the upper bound in Equation 15 ignores the implicit interactions among clients. In fact, the trajectory of the global model's parameter would likely change without the participation of the n-th client. These $\{\delta_i\}_{i=0}^{t-1}$ are derived from n clients' current gradients, but the gradients of the first $n-1$ clients may be implicitly biased.

Further, the incremental effect means that even if the influence of the n-th client is removed, there is no guarantee that we can obtain a model that is *exactly* the same as the one without the n-th client's participation. This is because the stochasticity incurred during training will definitely change the possible trajectory of parameters generated in gradient descent. Therefore, the best thing we can do is to make w_u and $\tilde{w}_t$ indistinguishable, such that an observer cannot identify if the model is trained by $n-1$ or n clients. According to the rationale of the Gaussian mechanism, to achieve indistinguishability, the output model of the federated learning algorithm should be perturbed by Gaussian noise. The perturbed learning algorithm is presented in Algorithm 1.

Following the above intuition, we employed the gradient flow [40] method to quantify the influence of the n-th client on the global model's performance. In addition, we derived a tighter upper bound of $\|w_t - \tilde{w}_t\|$ using the theory of gradient descent in non-convex optimization [41], and adopted differentially private mechanism to make w_u and $\tilde{w}_t$ indistinguishable.

C. Machine Unlearning Algorithm

The unlearning algorithm $\mathcal{M}_U$ is established based on gradient flow and differential privacy. The gradient flow stud-

Algorithm 1 The Perturbed Federated Learning Algorithm $\mathcal{M}_L$

Require: the initial parameter $w_0 \in \mathbb{R}^{\mathrm{d}}$ trained by federated learning, the privacy budget ϵ for the Gaussian mechanism, the confidence parameter β.

Ensure: the perturbed global model $\hat{w}$.

1: **train** the model from w_0 by t rounds federated learning and obtains parameter w_t.

2: **set**

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta) + \epsilon} - \sqrt{\log(1/\beta)}},$$

where d is the upper bound of the distance in Equations 34 to 36 based on the gradient descent method used.

3: **sample** a noise $z \in \mathbb{R}^{\mathrm{d}}$ from a Gaussian distribution, $z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

4: **return** $\hat{w}_t = w_t + z$.

ies the dynamics of gradients in continuous time, which is essentially the limit of the gradient descent method when the learning rate η approaches to 0.

When $\eta \to 0$, the gradient descent dynamic Equation 10 can be reformulated as

$$\lim_{\eta \to 0} \frac{w_t - w_{t-1}}{\eta} = -\frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_{t-1}) \tag{16}$$

Denote $F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$ the objective function of the server, we obtain the following differential equation,

$$\frac{\mathrm{d}w_t}{\mathrm{d}t} = -\nabla F(w_t). \tag{17}$$

Therefore, the variation tendency of the objective function $F(w)$ with respect to t can be derived using the chain rule.

$$\frac{\mathrm{d}F(w_t)}{\mathrm{d}t} = \frac{\mathrm{d}F(w_t)}{\mathrm{d}w_t} \cdot \frac{\mathrm{d}w_t}{\mathrm{d}t} = -\|\nabla F(w_t)\|^2. \tag{18}$$

The above equation characterizes the sensitivity of the objective function along the trajectory of the global model's parameter. As $-\|\nabla F(w_t)\|^2 \leq 0$, the equation also guarantees that, in general, if $F(w)$ is bounded from negative infinity, the federated learning process will converge.

As the server has easy access to the gradients of clients and has to compute $\nabla F(w_t)$ according to the protocol of federated learning, $\|\nabla F(w_t)\|^2$ is a natural way to weight the contribution of clients' gradients in each iteration. In comparison, using $\|\nabla F(w_t)\|^2$ rather than γ^t brings another advantage in that $\nabla F(w_t)$ incorporates the influence of the first $n-1$ clients' gradients with regard to $\nabla f_n(w_t)$. For example, the norm $\|\nabla f_n(w_t)\|$ itself may be large due to the landscape of the local loss function f_n, but the norm of the aggregated gradient $\|\nabla F(w_t)\|$ may stay small because the influence of the other $n-1$ clients' gradients is taken into consideration during the aggregation process. According to Equation 18, the global model's performance change is reflected by $\|\nabla F(w_t)\|$. Therefore, it is reasonable to use $\|\nabla F(w_t)\|^2$ as the weights to scale the influence of the n-th client from round 1 to t.

We use p_i to denote the weight of δ_i in the i-th iteration of federated learning.

$$p_i = \frac{\|\nabla F(w_i)\|^2}{\sum_{j=1}^{t-1} \|\nabla F(w_j)\|^2}, \quad i \in [t]. \tag{19}$$

When the n-th client submits an unlearning request, the unlearning algorithm $\mathcal{M}_U$ removes the n-th client's influence by computing

$$\bar{w}_t = w_t - \sum_{i=1}^{t-1} p_i \cdot \delta_i, \quad i \in [t]. \tag{20}$$

To make w_u and $\tilde{w}_t$ indistinguishable, it is necessary to introduce randomizations to mask the gap between $\bar{w}_t$ and $\tilde{w}_t$. Differentially private mechanisms convert the distances in parameter space to the distance between distributions, which is a natural choice. The algorithm $\mathcal{M}_U$ employs the Gaussian mechanism to achieve indistinguishability, where the noise is carefully calibrated by the Euclidean norm $\|\bar{w}_t - \tilde{w}_t\|$. Therefore, for $\bar{w} \in \mathbb{R}^{\mathrm{d}}$, we have the unlearned model parameter

$$w_u = \bar{w}_t + z, \quad z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}}). \tag{21}$$

The unlearning guarantee of the FedRecovery algorithm is given by the following theorem.

Theorem 1: The unlearning algorithm $\mathcal{M}_U$ satisfies (ϵ, β)-machine unlearning with respect to the learning algorithm $\mathcal{M}_L$. For any possible set of output $W \subseteq \mathbb{R}^{\mathrm{d}}$,

$$\begin{aligned} \Pr[w_u \in W] &\leq \exp(\epsilon) \cdot \Pr[\hat{w}_t \in W] + \beta, \\ \Pr[\hat{w}_t \in W] &\leq \exp(\epsilon) \cdot \Pr[w_u \in W] + \beta. \end{aligned} \tag{22}$$

Our key technical insight that leads to successful machine unlearning by introducing indistinguishability is based on the following theorems.

Theorem 2: The upper bound of the distance between $\bar{w}$ and $\tilde{w}_t$ is smaller than the upper bound of the distance between w_t and $\tilde{w}_t$. Namely,

$$\sup \|\bar{w} - \tilde{w}_t\| \leq \sup \|w_t - \tilde{w}_t\|. \tag{23}$$

Theorem 3: The expected distance between w_t and $\tilde{w}_t$ is upper-bounded by the following inequality for all iteration t,

$$\begin{aligned} &\mathbb{E}_\xi[\|w_t - \tilde{w}_t\|] \\ &\leq \sqrt{\sum_{t=0}^{t-1} \eta_t [F(w_0) - F(w^*) + \frac{LG^2}{2} \sum_{t=0}^{t-1} \eta_t^2]} + D_t, \end{aligned} \tag{24}$$

where w^* is the optimal solution of the objective function $F(w)$, and D_t is a constant relating to the global model's training trajectory. The expectation is taken with respect to the stochasticity of gradient descent $\xi = (\xi_1, \ldots, \xi_t)$.

The above theorems enable us to quantify the upper bound of distance between the parameter $\bar{w}_t$ and the parameter retrained without the n-th client $\tilde{w}_t$, which provides a chance to achieve (ϵ, β)-indistinguishability by introducing randomizations. The basic idea is as follows, for both the retrained parameter $\tilde{w}_t$ and the parameter $\bar{w}_t$ that removes the n-th client's effect, we use the Gaussian mechanism to form two distributions, $\hat{w}_t \sim \mathcal{N}(\tilde{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$ and $w_u \sim \mathcal{N}(\bar{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$, and

Algorithm 2 Unlearning Algorithm $\mathcal{M}_U$ for Federated Learning

Require: the unlearning request from the i_u-th client, the existing global model: $w_t \in \mathbb{R}^{\mathrm{d}}$, the gradients submitted by each client: $\nabla f_i(w_j)$ for $i \in [n]$ and $j \in [t]$, the privacy budget for the Gaussian mechanism: ϵ.

Ensure: the unlearned global model w_u.

1: **set**

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta) + \epsilon} - \sqrt{\log(1/\beta)}},$$

where d is the upper bound of the distance in Equation 34 to 36 based on the gradient descent method used.

2: **for** $i \in [t]$ **do**

3: **aggregate** the gradients for the global model,

$$\nabla F(w_i) = \frac{1}{n} \sum_{j=1}^{n} \nabla f_j(w_i), \quad j \in [n].$$

4: **end for**

5: **for** $i \in [t]$ **do**

6: **compute** the gradient residuals $\{\delta_i\}_{i=1}^{t}$ of the i_uth client by Equation 13.

$$\delta_i = \frac{\eta}{n}[\frac{1}{(n-1)} \sum_{j \neq i_u} \nabla f_j(w_i) - \nabla f_{i_u}(w_i)].$$

7: **end for**

8: **compute** the weights p_i for each gradient residual δ_i.

$$p_i = \frac{\|\nabla F(w_i)\|^2}{\sum_{j=1}^{t-1} \|\nabla F(w_j)\|^2}, \quad i \in [t].$$

9: **subtract** a weighted sum of δ_i from w.

$$\bar{w}_t = w_t - \sum_{i=1}^{t-1} p_i \cdot \delta_i, \quad i \in [t].$$

10: **sample** a noise $z \in \mathbb{R}^{\mathrm{d}}$ from a Gaussian distribution, $z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

11: **return** $w_u = \bar{w}_t + z$.

differential privacy guarantees the statistical indistinguishability between the two random variables (Definition 3).

The complete unlearning algorithm is presented in Algorithm 2. The algorithm takes as input the unlearning request of client i_u, the existing trained model w_t and the gradients incurred during training. The parameter ϵ controls the indistinguishability between the unlearned model w_u and the actual model $\tilde{w}_t$ trained without the participation of client i_u. The algorithm starts by computing the aggregated gradients and the gradient residuals relating to client i_u (Line 1-7). For each time step t, it assigns a probability mass that will be used to weight the gradient residuals $\{\delta_i\}_{i=0}^{t-1}$ (Line 8). The weights p_i are derived based on the norm of the aggregated gradients. Then the influence of the i_u-th client is eliminated by removing the weighted sum of δ_is from the trained model w_t (Line 9). Finally, the algorithm samples a noise vector from a Gaussian distribution and adds it to the previously deduced parameters $\bar{w}_t$ to achieve indistinguishability (Line 10-11). We defer the detailed proofs and related analyses of the algorithm to Section V.

D. Discussion and Analysis

In this section, we present discussions and related analyses of the FedRecovery algorithm, including running time, memory usage, early stopping strategy, etc.

- **Running time.** The FedRecovery algorithm only needs simple calculations to achieve unlearning, e.g., subtraction and sampling. Let t denote the training round, the time consumption of subtracting gradient residuals is $\mathcal{O}(t)$. The computation of model distance and the sampling of Gaussian noise are in $\mathcal{O}(1)$. In comparison to previous algorithms [12], [21], [22], FedRecovery does not require any retraining or fine-tuning based calibration to improve model's performance, and thus saves $\mathcal{O}(t_{extra})$ time for extra calibration. Moreover, the unlearning algorithm only runs on the server's side, without additional collaboration with the previous or new clients. Therefore, the communication overhead of FedRecovery is 0, which is also significantly reduced.
- **Memory usage.** The FedRecovery algorithm does not require memory-intensive checkpoint storage for models like the SISA [23] and RecEraser [32]. However, it does need the server to keep clients' historical submissions, which costs $\mathcal{O}(tn)$ in memory space. Given a fixed training round, the memory storage of FedRecovery grows linearly with the number of clients in federated learning. In comparison, the storage of SISA and RecEraser increases with the number of submodels and checkpoints. To further reduce memory usage, the server can adopt an early stop strategy in the unlearning process. The Gaussian mechanism will guarantee the indistinguishability between the early-stopped model and the retrained model.
- **Early stopping.** For neural networks, the NTK theory [42] indicates that in an overparameterization regime, the gradient-based methods converge exponentially fast to zero training error, with the model's parameters hardly varying (aka. "lazy training" phenomenon [43]). Therefore, for the purpose of saving memory, the server can store clients' submissions for only the first $t_0 < t$ rounds according to the global model's performance, and then run the FedRecovery to achieve machine unlearning.
- **Subsequent training.** The retraining-based unlearning algorithms [22], [44] suggest performing subsequent training on the remaining dataset to "heal" the damage caused by the unlearning operation. In FedRecovery, we followed another line of work that directly uses the Gaussian noise to mask the gap between parameters. We note that it is not necessary to perform further subsequent training on the output of FedRecovery, because the output of the algorithm is already guaranteed to be indistinguishable from retrained models.
- **Gradient disclosure.** Existing research has demonstrated that submitting raw gradients to the server can still reveal clients' information [45], [46]. Current federated learning frameworks typically employ differentially private techniques

or encryption-based methods to protect clients' submissions. However, in the context of machine unlearning, the server must verify the legitimacy of each unlearning request. For local differential privacy-based federated learning frameworks, the server can continue to regard each client's noisy submission as their contribution and use the proposed mechanism to eliminate their impact. In the scenario where the model parameter is updated under homomorphic encryption. It is impossible to perform an unlearning operation because the server cannot validate and distinguish the contributions of each client. In addition, there is no need for clients to submit an unlearning request in this context, as the encryption procedure has already severed the relationship between clients and their submissions.

V. Theoretical Analysis

In this section, we present detailed proofs of previous theorems. We first show that the proposed unlearning algorithm satisfies (ϵ, β)-indistinguishability (Theorem 1), and then give an upper bound on the distance between w_t and $\tilde{w}_t$ (Theorems 2 and 3).

Theorem 1 guarantees the (ϵ, β)-indistinguishability of the proposed unlearning algorithm, which is essentially an implementation of the Gaussian mechanism.

Proof of Theorem 1: Denote $C = \{c_1, \ldots, c_n\}$ the set that contains n clients in a federated learning, and $C' = C \backslash \{c_{i_u}\}$ the set that contains $n-1$ clients excluding the i_u-th one. In addition, let $\hat{w}_t = \mathcal{M}_L(C', \epsilon)$ be the model obtained by retraining from scratch and $w_u = \mathcal{M}_U(w_t, \nabla F, \epsilon)$ be the model produced by the unlearning algorithm. Note that w_t is the parameter trained with clients in C without adding noise.

According to Algorithm 1 and Algorithm 2, we know that $\hat{w}_t$ and w_u are computed as $\hat{w}_t = \tilde{w}_t + z$ and $w_u = \bar{w}_t + z$ respectively, where the noise $z \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_{\mathrm{d}})$. Therefore, the random variables $\hat{w}_t$ and w_u follow distributions $\hat{w}_t \sim \mathcal{N}(\tilde{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$ and $w_u \sim \mathcal{N}(\bar{w}_t, \sigma^2 \mathbb{I}_{\mathrm{d}})$.

In addition, according to Theorem 2, we know that the upper bound of $\|\bar{w}_t - \tilde{w}_t\|$ is bounded by the upper bound of $\|w_t - \tilde{w}_t\|$, where the latter is given in Theorem 3.

Finally, we invoke the Gaussian mechanism (Definition 3) to show that variables $\hat{w}_t$ and w_u are indistinguishable, and thus the unlearning algorithm $\mathcal{M}_U$ satisfies (ϵ, β)-unlearning guarantee with

$$\sigma = \frac{1}{\sqrt{2}} \cdot \frac{d}{\sqrt{\log(1/\beta) + \epsilon} - \sqrt{\log(1/\beta)}}, \tag{25}$$

where the distance d is taken following Equations 34 to 36 based on the specific gradient descent method used.

That completes the proof. □

Theorem 2 provides an insight about the upper bound of distance between w_u and $\bar{w}_t$, which enables us to quantify the amount of noise used to achieve (ϵ, β)-indistinguishability.

Proof of Theorem 2: Note that Equation 12 illustrates the change of the global model's parameters between two successive training rounds. We hence sum Equation 12 over rounds 1 to t. Based on the fact $w_0 \equiv \tilde{w}_0$, we can obtain the difference between parameters $\tilde{w}_t$ and w_t.

$$\tilde{w}_t - w_t = \frac{\eta}{n-1} \sum_{j=1}^{t} \sum_{i=1}^{n-1} [\nabla f_i(w_{j-1}) - \nabla f_i(\tilde{w}_{j-1})] - \sum_{j=1}^{t} \delta_{j-1}. \tag{26}$$

According to Equations 11 and Equation 20, the difference between parameter $\tilde{w}_t$ and the parameter that removes the n-th client's influence $\bar{w}_t$ is given by

$$\tilde{w}_t - \bar{w}_t = \frac{\eta}{n-1} \sum_{j=1}^{t} \sum_{i=1}^{n-1} [\nabla f_i(w_{j-1}) - \nabla f_i(\tilde{w}_{j-1})] - [\sum_{j=1}^{t} \delta_{t-1} - \sum_{j=1}^{t} p_j \delta_{t-1}]. \tag{27}$$

Note that the first item in Equation 26 and Equation 27 are the same, denoted as A. By applying the triangle inequality and the fact $p_j \leq 1$, we have

$$\begin{aligned} \sup \|\bar{w}_t - \tilde{w}_t\| &\leq \|A\| + \sum_{j=1}^{t} (1 - p_j) \|\delta_{j-1}\| \\ &\leq \|A\| + \sum_{j=1}^{t} \|\delta_{j-1}\| \\ &= \sup \|w_t - \tilde{w}_t\| \end{aligned} \tag{28}$$

Therefore, the Euclidean norms between parameters satisfy

$$\sup \|\bar{w}_t - \tilde{w}_t\| \leq \sup \|w_t - \tilde{w}_t\|. \tag{29}$$

That completes the proof. □

Theorem 3 states that for a properly chosen learning rate, the expected distance between parameters w_t and $\tilde{w}_t$ grows at a rate of $O(\sqrt{t})$. Therefore, we can bound the distance between $\bar{w}$ wand $\tilde{w}_t$ by bounding the distance between w_t and $\tilde{w}_t$.

Before proofing Theorem 3, we first show the smoothness of the global objective function $F(w)$, which is guaranteed by the smoothness of each local objective function $f_i(w)$.

Lemma 1 (Smoothness of Function F): If each function $f_i : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$ is L-smooth, the function $F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$ is also L-smooth.

Proof: The result follows by the definition of L-smoothness and the triangle inequality.

$$\begin{aligned} \|\nabla F(w_1) - \nabla F(w_2)\| &= \frac{1}{n} \| \sum_{i=1}^{n} \nabla f_i(w_1) - \sum_{i=1}^{n} \nabla f_i(w_2) \| \\ &\leq \frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(w_1) - \nabla f_i(w_2)\| \\ &\leq L \|w_1 - w_2\|. \end{aligned} \tag{30}$$

Now we present the proof of Theorem 3 based on lemma 1.

Proof of Theorem 3: Denote $G(w_t, \xi_t)$ the gradient of $F(w_t)$ with sampling distribution ξ_t. By the smoothness of function

F and the stochastic gradient descent update rule $w_{t+1} = w_t - \eta_t G(w_t, \xi_t)$, we have

$$\begin{aligned}
&\mathbb{E}_{\xi_t}[F(w_{t+1})] \\
&\le \mathbb{E}_{\xi_t}[F(w_t) + \nabla F(w_t)^T (w_{t+1} - w_t) + \frac{L}{2}\|w_{t+1} - w_t\|_2] \\
&= \mathbb{E}_{\xi_t}[F(w_t) - \eta_t \nabla F(w_t)^T G(w_t, \xi_t) + \frac{L}{2}\eta_t^2 \|G(w_t, \xi_t)\|^2] \\
&\le \mathbb{E}_{\xi_t}[F(w_t)] - \eta_t \|\nabla F(w_t)\|^2 + \frac{1}{2}\eta_t^2 L G^2, \qquad (31)
\end{aligned}$$

where the last inequality follows from the assumption that $G(w_t, \xi_t)$ is an unbiased estimation of $\nabla F(w_t)$, and $\mathbb{E}_{\xi_t}\|\nabla f_i(w_t)\|^2 \le G^2$.

Rearrange these terms, we have

$$\eta_t \|\nabla F(w_t)\|^2 \le \mathbb{E}_{\xi_t}[F(w_t)] - \mathbb{E}_{\xi_t}[F(w_{t+1})] + \frac{L}{2}\eta_t^2 G^2. \qquad (32)$$

Summing over t iterations and taking the overall expectation over $\xi = (\xi_1, \ldots, \xi_t)$, we have

$$\begin{aligned}
\sum_{t=0}^{t-1} \eta_t \|\nabla F(w_t)\|^2 &\le \mathbb{E}_{\xi}[F(w_0)] - \mathbb{E}_{\xi}[F(w_{t-1})] + \frac{L}{2}\sum_{t=0}^{t-1}\eta_t^2 G^2 \\
&\le F(w_0) - F(w^*) + \frac{1}{2}\sum_{t=0}^{t-1}\eta_t^2 L G^2. \qquad (33)
\end{aligned}$$

Finally, by applying the triangle inequality and the linearity of expectation, we have

$$\begin{aligned}
&\mathbb{E}_{\xi}[\|w_t - \tilde{w}_t\|] \\
&= \mathbb{E}_{\xi}[\|w_t - w_0\|] + \mathbb{E}_{\xi}[\|\tilde{w}_t - w_0\|] \\
&= \|\sum_{t=0}^{t-1} \eta_t \mathbb{E}_{\xi_t}[G(w_t, \xi_t)]\| + D_t \\
&\le \sum_{t=0}^{t-1} \eta_t \|\nabla F(w_t)\| + D_t \\
&\le \sqrt{(\sum_{t=0}^{t-1}\eta_t)(\sum_{t=0}^{t-1}\eta_t \|\nabla F(w_t)\|^2)} + D_t \\
&\le \sqrt{\sum_{t=0}^{t-1}\eta_t [F(w_0) - F(w^*) + \frac{LG^2}{2}\sum_{t=0}^{t-1}\eta_t^2]} + D_t, \qquad (34)
\end{aligned}$$

where the penultimate inequality follows from the Cauchy–Schwarz inequality. □

Based on Theorem 3, we immediately have the following corollary.

Corollary 1: If each client adopts batch gradient descent in their local training, the distance between w_t and $\tilde{w}_t$ satisfies the following inequality for all iteration $t \in [T]$:

$$\|w_t - \tilde{w}_t\| \le \sqrt{2\sum_{t=0}^{t-1}\eta_t [F(w_0) - F(w^*)]} + D_t. \qquad (35)$$

Further for a constant learning rate $\eta_k = \eta \le 1/L$, we have

$$\|w_t - \tilde{w}_t\| \le \sqrt{2t\eta [F(w_0) - F(w^*)]} + D_t. \qquad (36)$$

Proof: (Sketch) In batch gradient descent, we do not consider the bias of gradient estimation, so that the second term in Equation 33 disappears and the result follows as Equation 35. If the learning rate is a constant η, $\sum_{i=0}^{t-1}\eta_t = t\eta$, which implies the result in Equation 36. □

VI. Experiment and Analysis

In this section, we evaluate the performance of the proposed FedRecovery algorithm. We first examine the statistical indistinguishability it provides, investigating the difference between the unlearned model and the retrained model. Then we look into the performance of the unlearning algorithm to see whether the influence of the targeted client is removed. The experimental results as well as corresponding analyses are presented in each subsection.

A. Experimental Setup

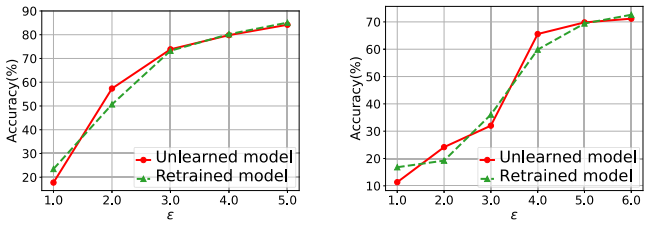
1) Datasets and Models: We used four real-world datasets in the experiments, including MINST [47], CIFAR10 [48], SVHN [49], and USPS [50].

- **MNIST.** The dataset contains 60, 000 training images and 10, 000 test images for hand written digit recolonization. Each image is grayscale and normalized in 28×28 pixels.
- **CIFAR10.** The dataset consists of 60, 000 32×32 color images in 10 classes, 50, 000 of them are training images, and 10, 000 are test images.
- **SVHN.** It contains 600, 000 32×32 color images from pictures of house number plates. The images are centered and the background distractors are kept in the image.
- **USPS.** The dataset contains 9, 300 16×16 gradyscale images scanned from envelopes by the U.S. Postal Service. The images are centered, normalized, and show a broad range of font styles.

In the experiments, we adopted convolutional neural network (CNN) models to perform image classification tasks on these datasets, which are generally non-convex. To limit the injected noise and guarantee a good performance, we started federated learning from pre-trained models, such that the distance between the unlearned model and retrained model will not change drastically. The pre-training strategy is commonly used in computer vision and natural language processing, and it does not violate the principle of federated learning [51].

2) Evaluation and Comparison Metrics: Intuitively, an effective unlearning algorithm should force the model to "forget" what it has learned about the unlearned data, but retain its 'knowledge" of the remaining ones. In our experiment, we adopted the widely accepted criteria to evaluate the proposed algorithm.

- **Accuracy.** We investigated the overall accuracy of the unlearned model and the retrained model. Meanwhile, we also examined the accuracy of the unlearned model on unlearned class and remained classes.
- **Running time.** We looked into the running time needed to perform the proposed unlearning algorithm, and made comparisons with benchmark methods like retraining from scratch.



(a) Acc. vs. ϵ, MNIST. (b) Acc. vs. ϵ, CIFAR10.

(c) Acc. vs. ϵ, SVHN. (d) Acc. vs. ϵ, USPS.

Fig. 3. The accuracy of unlearned model and retrained model with different privacy budgets.

- **Attack success rate.** We performed membership inference attack (MIA) on the unlearned model to see if the influence of the targeted client was really removed by the proposed unlearning algorithm.

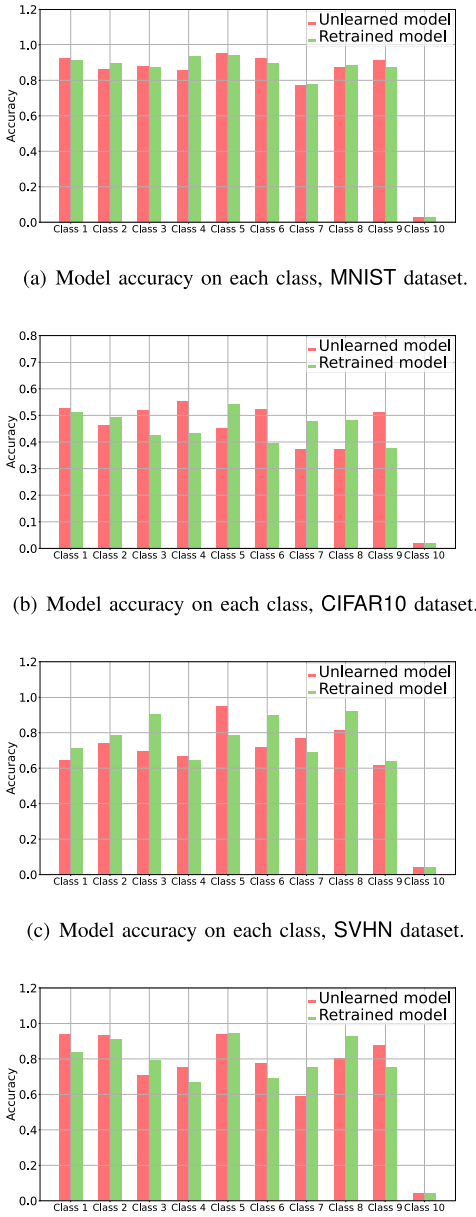
Verifying the success of machine unlearning is not an easy task, because the verification method is usually tailored based on certain unlearning strategies. In our experiment, we made comparisons with the state-of-the-art unlearning algorithms [19], [21], [22] in federated learning.

The experiments were conducted on a server with Red Hat Enterprise Linux 7.9 OS, Intel(R) Xeon(R) E-2288G 3.70GHz CPU, and NVIDIA Quadro RTX6000 GPU with 24G RAM.

B. Results and Analyses

1) The Influence of the Privacy Budget: The privacy budget controls the indistinguishability between the unlearned model and the retrained model. Therefore, our first experiment is designed to investigate the influence of the privacy budget on model's performance. In this experiment, we used $n = 10$ participating clients, each of whom holds a local database that is randomly sampled from the original one. The privacy budget was set in the range $1.0 \sim 6.0$, and the performance was assessed in terms of the global model's accuracy. The results are shown in Figure 3.

Figure 5(a) shows the change of model accuracy with respect to different privacy budgets on the MNIST dataset. In this figure, the accuracy of both models increases from around 20% to 90% with the privacy budget grows from 1.0 to 5.0. With a larger privacy budget, the requirement of indistinguishability is weaker, so that less noise is injected in the model's parameters. As a result, the accuracy of models increases. The result patterns were similar for the other three datasets. Note that in some cases, the accuracy of the unlearned model is higher than that of the retrained model. This is caused by the randomness incurred when sampling the Gaussian noise. From the figures, we can see that the performance



(a) Model accuracy on each class, MNIST dataset.

(b) Model accuracy on each class, CIFAR10 dataset.

(c) Model accuracy on each class, SVHN dataset.

(d) Model accuracy on each class, USPS dataset.

Fig. 4. The accuracy of unlearned model and retrained model on different classes.

of the unlearned model is similar to that of the retrained model, which verifies the indistinguishability achieved by the proposed FedRecovery algorithm.

2) The Effectiveness of Unlearning: Our next experiment aims to examine the unlearning ability of the proposed FedRecovery algorithm. In this experiment, we took $n = 10$ clients, each of whom holds a local database that corresponds to a specific class of data in the original database. The clients first collaboratively trained a model that was able to recognize data in 10 classes, and then we ran FedRecovery to remove the influence of the last client, who contributed to the last class of data. The results are shown in Figure 4.

Figure 4(a) shows the accuracy of the unlearned model with respect to different classes on the MNIST dataset. Intuitively,

TABLE II
THE RUNNING TIME OF UNLEARNING METHODS, COMPARISON

Datasets	Running time (s)				
	Retrain	ref [19]	ref [21]	ref [22]	**Ours**
MNIST	679.1	182.6	385.1	129.7	$< \mathbf{1.0}$
CIFAR10	2484.5	879.2	1082.3	683.9	$< \mathbf{2.0}$
SVHN	1380.6	587.1	622.4	311.4	$< \mathbf{1.0}$
USPS	285.2	99.4	137.8	64.7	$< \mathbf{1.0}$

TABLE III
THE ACCURACY OF UNLEARNING METHODS, COMPARISON

Datasets	Global model's accuracy (%)				
	Retrain	ref [19]	ref [21]	ref [22]	**Ours**
MNIST	97.4	97.2	94.1	96.8	**90.9**
CIFAR10	84.9	83.2	80.1	83.8	**71.2**
SVHN	84.6	82.3	80.4	75.3	**70.4**
USPS	90.3	89.4	84.7	90.1	**83.0**

the unlearned model should still be able to recognize the remaining 9 classes of data, but forget what it has learned about the last one. Figures 4(a) to 4(d) confirm this conjecture. In Figure 4(a), we observe that the unlearned model keeps high performance on Class1 to Class9, while the model's accuracy drops significantly on Class10. The reason is that the unlearning operation produces a model that is within a small distance of the retrained model, where the distance gap is then masked by differential privacy. Therefore, the performances of the unlearned model and the retrained model are guaranteed to be similar. A similar variation tendency can also be observed in Figures 4(b) to 4(d), which were obtained using different datasets and different models. This experiment demonstrates that the proposed FedRecovery is effective in removing client's influence in federated learning environment.

3) The Efficiency of Unlearning: The time consumption of the FedRecovery algorithm mainly comes from the computation of gradient residuals and the storage/readout of cached statistics. In this experiment, we evaluated the unlearning efficiency of the FedRecovery in terms of running time, and made comparisons with current unlearning solutions [19], [21], [22]. The retraining time of these methods varies depending on the specific federated learning framework they use. For a clear comparison, we reported the averaged retraining time here. The results are presented in Table II.

From the table, we can observe that retraining from scratch is the most time-consuming unlearning strategy, which explains the motivation of current unlearning efforts. In comparison, other methods reduce the unlearning execution time to some extent. For example, [19] adopts the Fisher information matrix to approximate the Hessian, and thus accelerates the retraining procedure. In [21], the retraining update only happens in specific rounds, which reduces the model reconstruction time. Despite using speed-up techniques, [19] and [21] require clients to perform local training for certain rounds. In addition, [22] requires clients to download the pruned model and conduct further fine-tuning. These unlearning strategies rely on subsequent training, which still takes execution time and incurs communication costs between clients and the server. Compared to these unlearning techniques, the FedRecovery algorithm avoids retraining and fine-tuning based model calibration. As a result, the running time is significantly reduced.

4) Comparison With Existing Methods: Our next experiment involves a comparison of the model's performance between the proposed FedRecovery and existing machine unlearning algorithms. In this experiment, we took $n = 10$ clients, each holding a partition of the original database. For [19] and [21], the data are independent and identically distributed across clients due to the rationale of their methods. For [22], each client holds a specific class of data in the original database. For comparison purposes, we set the client deletion ratio to 0.1 and the data deletion ratio to 1.0 when applying their unlearning strategy. The local update epoch in their retraining and fine-tuning is set to 1, with learning rate ranges from 0.01 to 0.05 and batch size of 128. The results are shown in Table III.

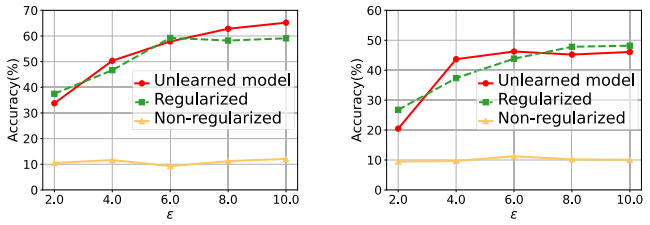
From the table, we can observe that retraining from scratch achieves high accuracy on the four datasets. In comparison, the methods in [19], [21], and [22] (the 2nd-4th columns) produce similar results that are comparable to retraining. The reason is that these methods were developed based on the idea of retraining and fine-tuning, which can be regarded as accelerated retraining variations. As for FedRecovery, it does not lead to accuracy as high as the previous methods. This is because the injection of the Gaussian noise perturbs the model's parameters, and thus inevitably decreases the model's performance. However, we note that this despondency is caused by the design rationale of the unlearning strategy: the FedRecovery algorithm relies on differentially private noise to achieve indistinguishability, which is elaborated for the scenario where retraining is unavailable. In addition, the execution time of the FedRecovery is significantly reduced compared to the retraining-based unlearning strategies. Therefore, the FedRecovery algorithm is competent at removing the impact of a particular client and is capable of guaranteeing a competitive performance for federated learning frameworks.

5) The Performance of MIA on Unlearned Models: One of the fundamental purposes of machine unlearning is to protect the privacy of clients in federated learning. In this experiment, we conducted a membership inference attack to estimate how much information about the unlearned client is still contained in the unlearned model. The aim of MIA is to infer whether the targeted sample was used to train the original model. Therefore, a lower attack precision means a more complete removal of a client's influence. We adopted the MIA strategy in [10] towards the target client's data to build an attack classifier. The results are presented in Table IV.

As can be seen from Table IV, the MIA has a high attack precision on the original models. In this scenario, the attacker is able to successfully figure out if the target client's data was used during the training. By contrast, after unlearning, the attack precision of MIA is reduced, which means that the attacker cannot decide if the targeted client's data was in the training dataset. Comparing the first and last columns, the drop in attack precision indicates that the impact of the target client is successfully removed by the proposed FedRecovery

TABLE IV
ATTACK PRECISION OF MIA ON UNLEARNED MODELS, COMPARISON

Datasets	Attack precision (%)					
	Origin	Retrain	ref [19]	ref [21]	ref [22]	**Ours**
MNIST	96.8	50.5	50.1	53.4	52.3	**50.9**
CIFAR10	72.7	49.8	51.8	51.5	53.5	**51.2**
SVHN	98.5	50.1	54.9	56.7	49.9	**50.4**
USPS	97.6	49.5	52.2	68.9	50.1	**53.0**



(a) Acc. vs. ϵ, MNIST. (b) Acc. vs. ϵ, CIFAR10.

Fig. 5. The accuracy of the unlearned model with regularization.

algorithm. Moreover, the results produced by FedRecovery are similar to those of the retrained model, which are also comparable to existing solutions [19], [21], [22]. This shows that even if FedRecovery does not rely on retraining or fine-tuning, it still yields effective unlearning results. This experiment demonstrates that the proposed unlearning algorithm indeed has the ability to remove a client's influence from a trained global model.

6) Discussions and Summary: Throughout the experiments, we start with pre-trained models because the distance between models before and after training will not change significantly. Therefore, according to Equations 24 and 25, the amount of noise added to models is small, and thus the models' performances can be guaranteed. For a model that begins from an arbitrary initial point, there is no assurance that the distance D_t in Equation 24 is small. As a result, the injected noise may significantly reduce the model's performance. Theoretically, this issue cannot be avoided because throughout the paper we do not presume convexity of loss functions. Equation 34's triangular inequality is the optimal distance approximation under the assumption of Lipschitzness.

To circumvent this problem, we conducted additional experiments in which a regulation term was added to the loss function. In this experiment, the models were trained from random starting points without any pre-training. Regularization seeks to restrict the D_t distance between the trained model and the initial model. The outcomes are depicted in Figure 5. From the figure, we can observe that if the models are trained without regularization, the injected noise will substantially reduce the performance of the unlearned model. With a regularization term, the distance between the initial model and the rained model is constrained, enhancing the accuracy of the models. Therefore, introducing a regularization term is an effective way to improve the performance of models without pre-training. It is important to note, however, that regularization reduces the accuracy of both the unlearned and retrained models.

In this section, we investigated the influence of differential privacy in achieving statistical indistinguishability between models, which verified the fundamental idea of the proposed unlearning strategy. Furthermore, we also examined the unlearning ability of the proposed algorithm, along with its execution time and the model's performance after unlearning operations. The experimental results show that the proposed FedRecovery can efficiently remove the influence of a target client. Compared with current unlearning strategies that heavily rely on retraining and fine-tuning, our method is more suitable for real-world federated learning scenarios.

VII. CONCLUSION AND FUTURE WORK

In this paper, we develop a differentially private machine unlearning algorithm for federated learning frameworks. The proposed FedRecovery algorithm removes the influence of the unlearned client from a trained global model, and adopts Gaussian noise to mask the gap between unlearned parameters and retrained parameters. We gave up the widely used convexity assumption, and derived the upper bound of distance between parameters trained with/without the unlearned client. The FedRecovery does not need any retraining-based fine-tuning or calibration, which can be performed cheaply by the server itself. Theoretical analyses show that FedRecovery satisfies the desired unlearning guarantee, and experimental results demonstrate the performance of FedRecovery is competitive with the retrained model. In future work, we plan to explore the possible exclusion of Lipschitz conditions and develop a more general unlearning algorithm in the context of complex neural networks, which may provide new insight into the efforts for machine unlearning tasks.

REFERENCES

[1] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 13, pp. 11516–11524.

[2] T. T. Nguyen, T. T. Huynh, P. Le Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen, "A survey of machine unlearning," 2022, *arXiv:2209.02299*.

[3] P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Cham, Switzerland: Springer, 2017.

[4] *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons With Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (GDPR)*, Eur. Union, Brussels, Belgium, May 2016, pp. 1–88.

[5] K. Klein, *Canadian Privacy: Data Protection and Policy for the Practitioner*. International Association of Privacy Professionals, Portsmouth, NH, USA, 2020.

[6] E. L. Harding, J. J. Vanto, R. Clark, L. H. Ji, and S. C. Ainsworth, "Understanding the scope and impact of the California consumer privacy act of 2018," *J. Data Protection Privacy*, vol. 2, no. 3, pp. 234–253, 2019.

[7] Google Research. (Apr. 2017). *Federated Learning: Collaborative Machine Learning Without Centralized Training Data*. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

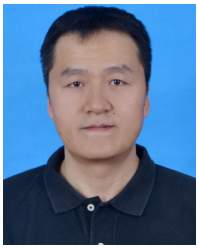
[8] S. K. Lo, Q. Lu, C. Wang, H.-Y. Paik, and L. Zhu, "A systematic literature review on federated machine learning: From a software engineering perspective," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–39, May 2021.

[9] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1322–1333.

[10] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 3–18.
[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, Jun. 2019, pp. 634–643.
[12] C. Wu, S. Zhu, and P. Mitra, "Federated unlearning with knowledge distillation," 2022, *arXiv:2201.09441*.
[13] A. A. Ginart, M. Y. Guan, G. Valiant, and J. Zou, *Making AI Forget You: Data Deletion in Machine Learning*. Red Hook, NY, USA: Curran Associates Inc., 2019.
[14] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *Proc. 37th ICML*, 2020, pp. 3832–3842.
[15] Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou, "Approximate data deletion from machine learning models," in *Proc. 24th Int. Conf. Artif. Intell. Statist.*, vol. 130, Apr. 2021, pp. 2008–2016.
[16] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: Algorithms for machine unlearning," in *Advances in Neural Information Processing Systems*, vol. 34. Red Hook, NY, USA: Curran Associates, 2021, pp. 18075–18086.
[17] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spotless net: Selective forgetting in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9301–9309.
[18] Y. Wu, E. Dobriban, and S. Davidson, "DeltaGrad: Rapid retraining of machine learning models," in *Proc. 37th ICML*, vol. 119, Jul. 2020, pp. 10355–10366.
[19] 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 *Proc. IEEE Conf. Comput. Commun.*, May 2022, pp. 1749–1758.
[20] E. F. Villaronga, P. Kieseberg, and T. Li, "Humans forget, machines remember: Artificial intelligence and the right to be forgotten," *Comput. Law Secur. Rev.*, vol. 34, no. 2, pp. 304–313, Apr. 2018.
[21] G. Liu, X. Ma, Y. Yang, C. Wang, and J. Liu, "FedEraser: Enabling efficient client-level data removal from federated learning models," in *Proc. IEEE/ACM 29th Int. Symp. Quality Service (IWQOS)*, Jun. 2021, pp. 1–10.
[22] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 622–632.
[23] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 141–159.
[24] R. Bollapragada, J. Nocedal, D. Mudigere, H.-J. Shi, and P. T. P. Tang, "A progressive batching L-BFGS method for machine learning," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 620–629.
[25] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, Jul. 2013, pp. 343–352.
[26] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1885–1894.
[27] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, Mar. 2011.
[28] A. Golatkar, A. Achille, and S. Soatto, "Forgetting outside the box: Scrubbing deep networks of information accessible from input–output observations," in *Proc. Int. Conf. Comput. Vis.*, 2020, pp. 383–398.
[29] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, vol. 31. Red Hook, NY, USA: Curran Associates, 2018.
[30] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *Proc. IEEE Symp. Secur. Privacy*, May 2015, pp. 463–480.
[31] S. Neel, A. Roth, and S. Sharifi-Malvajerdi, "Descent-to-delete: Gradient-based methods for machine unlearning," in *Proc. 32nd Int. Conf. Algorithmic Learn. Theory*, vol. 132, Mar. 2021, pp. 931–962.
[32] C. Chen, F. Sun, M. Zhang, and B. Ding, "Recommendation unlearning," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 2768–2777.
[33] F. Zhou and C. Cao, "Overcoming catastrophic forgetting in graph neural networks with experience replay," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 5, pp. 4714–4722.
[34] X. Yin, Y. Zhu, and J. Hu, "A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–36, Jul. 2021.
[35] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Germany: Springer, 2006, pp. 1–12.
[36] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Proc. 48th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Oct. 2007, pp. 94–103.
[37] T. Zhu, D. Ye, W. Wang, W. Zhou, and P. S. Yu, "More than privacy: Applying differential privacy in key areas of artificial intelligence," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 6, pp. 2824–2843, Jun. 2022.
[38] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Proc. 14th Int. Conf. Theory Cryptogr. (TCC)*, vol. 9985. Berlin, Germany: Springer, 2016, pp. 635–658.
[39] K. Scaman and A. Virmaux, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 3839–3848.
[40] A. Luigi, G. Nicola, and S. Giuseppe, *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*, 1st ed. Basel, Switzerland: Birkhäuser, 2005.
[41] G. Lan, *Nonconvex Optimization*. Cham, Switzerland: Springer, 2020, pp. 305–420.
[42] J. Lee et al., *Wide Neural Networks of Any Depth Evolve as Linear Models under Gradient Descent*. Red Hook, NY, USA: Curran Associates, 2019.
[43] L. Chizat, E. Oyallon, and F. Bach, "On lazy training in differentiable programming," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019.
[44] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli, "Fast yet effective machine unlearning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 1, 2023, doi: 10.1109/TNNLS.2023.3266233.
[45] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019.
[46] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 7232–7241.
[47] Y. LeCun. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[48] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.
[49] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop DL Unsupervised Feature Learn.*, 2011, pp. 1–12.
[50] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[51] Y. Tian, Y. Wan, L. Lyu, D. Yao, H. Jin, and L. Sun, "FedBERT: When federated learning meets pre-training," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–26, Aug. 2022.

Lefeng Zhang received the B.Eng. and M.Eng. degrees from the Zhongnan University of Economics and Law, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with the University of Technology Sydney, Australia. His research interests are game theory and privacy preserving.

Tianqing Zhu received the B.Eng. and M.Eng. degrees from Wuhan University, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Australia, in 2014. She was a Lecturer with the School of Information Technology, Deakin University, from 2014 to 2018. She is currently an Associate Professor with the School of Computer Science, University of Technology Sydney, Australia. Her research interests include privacy preserving, data mining, and network security.



Haibin Zhang is a Doctoral Supervisor and the Director of the Department of Operational Research and Scientific Computation, Faculty of Science, Beijing University of Technology. He has been engaged in teaching and research for 18 years in BJUT. He has presided over three National Natural Science Foundation of China projects. He has published more than 40 articles on international major journals and one academic book.

Ping Xiong received the B.Eng. degree from Lanzhou Jiaotong University, Lanzhou, China, in 1997, and the M.Eng. and Ph.D. degrees from Wuhan University, Wuhan, China, in 2002 and 2005, respectively. He is currently a Professor with the School of Information and Security Engineering, Zhongnan University of Economics and Law, China. His research interests include network security, data mining, and privacy preservation.

Wanlei Zhou (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science and engineering from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, the Ph.D. degree in computer science and engineering from The Australian National University, Canberra, Australia, in 1991, and the D.Sc. degree from Deakin University, Australia, in 2002. He is currently the Vice Rector (Academic Affairs) and the Dean of the Institute of Data Science, City University of Macau, Macau, SAR, China. Before joining the City University of Macau, he held various positions, including the Head of the School of Computer Science, University of Technology Sydney, Australia; the Alfred Deakin Professor; the Chair of Information Technology; the Associate Dean; and the Head of the School of Information Technology, Deakin University. He was a Lecturer at the University of Electronic Science and Technology of China; a System Programmer at HP, MA, USA; a Lecturer at Monash University, Melbourne, Australia; and a Lecturer at the National University of Singapore, Singapore. He has published more than 400 papers in refereed international journals and refereed international conferences proceedings, including many articles in IEEE transactions and journals. His main research interests include security, privacy, and distributed computing.