Appro-Fun: Approximate Machine Unlearning in Federated Setting

Zuobin Xiong*, Wei Li†, and Zhipeng Cai†
*Department of Computer Science, University of Nevada, Las Vegas, NV, USA
†Department of Computer Science, Georgia State University, Atlanta, GA, USA
* *zuobin.xiong@unlv.edu*; † {*wli28, zcai*}*@gsu.edu*

***Abstract*—Machine learning models contain a lot of information about the training dataset, so even if some data points are deleted, the private information can still be inferred. To counteract this problem, "machine unlearning", as an emerging data management approach, is proposed to remove data from the databases and the influence of data from the trained models. Yet, machine unlearning is still in its early stage, and there are rare existing methods towards machine unlearning in the federated setting that is a more practical and crucial scenario. Therefore, this paper investigates the federated machine unlearning problem where the local clients of a federated system intend to delete their local private data appropriately. The proposed method is termed as Approximate Federated unlearning (Appro-Fun), which adopts differential privacy and second-order optimization to achieve (ϵ, δ)-approximate unlearning on trained models. Rigorous theoretic analysis presents the performance guarantee of Appro-Fun, and real-data experiments validate the advantages of Appro-Fun compared with the state-of-the-art.**

***Index Terms*—federated learning, machine unlearning, privacy and security, data management**

I. Introduction

For the performance improvement of machine learning models, it is indispensable to collect an adequate amount of training data from data owner and/or other third parties. To name some, Google collects users' search history and interests to train the commercial applications for recommendation and advertisements, images and videos posted by Facebook and Instagram users [1] are used to learn image classifier or object detection models, and several language processing models are trained on user reviews from different websites [2]. Generally, users give their data out to the service providers/applications for model development in exchange of desired service quality, entertainment, experience, *etc*. When more and more data is being collected and analyzed, data privacy becomes an inescapable concern for everyone. To deal with data privacy, the concept of "Right To Be Forgotten" has been proposed and enforced by some laws and regulations, including the General Data Protection Regulation (GDPR) of EU [3], the California Consumer Privacy Act (CCPA) [4], and Federal Trade Commission (FTC) [5]. Following these regulations, a user has the rights to delete his/her private data from service providers at anytime they want, such as medical data and search history. Thus, a practical and crucial question for the service providers is how to manage the users' data removal requests in an efficient and effective manner?

As well known, simply deleting users' data from storage databases is not a successful way because the machine learning models are able to memorize quite information of the training data [6], regardless of private or non-private. This memorized information is not forgotten until the model is changed accordingly. In addition, many kinds of attacks, such as model inversion attack [7], membership inference attack [8], and reconstruction attack [9], can be harnessed to dig out privacy of the deleted data from learned models. Therefore, correct and guaranteed data removal from both database-level and the trained model-level for users has become a challenging problem faced by researchers. Intuitively, deleting the data from training dataset and then retraining the machine learning models from scratch is a straightforward method to achieve the unlearning objective, but the retraining time cost is prohibitively high, especially on the complex tasks and models with billions of parameters [10]. Thus, designing an effective and computationally efficient data deletion method is the main focus of current machine unlearning research.

The terminology "machine unlearning" was first proposed in [11], where the private data from training datasets and the impact of data in the learned models are eliminated for statistical query tasks with less cost. Since then, there have been only limited works on unlearning mechanisms but with different weaknesses: (i) Model dependency. Some unlearning methods are specified for certain learning models, such as statistical query [11], k-means cluster [12], decision tree [13], and linear regression [14], [15]. (ii) Lack of theoretical guarantee. Most existing works focus on data deletion experiments while omitting the theoretical unlearning guarantee, which is important for users to know how private their data is after unlearning. Besides, when it comes to federated settings, more challenges in unlearning are brought by the distributed users and model exchanges between server and users.

Motivated by the aforementioned observations, in this paper, we aim to design an approximate federated unlearning algorithm that is model agnostic and with theoretical guarantee. Specifically, in our proposed Appro-Fun algorithm, local clients can remove their data through a quasi Newton's method and get an unlearned local model. Then, the unlearned local model is uploaded to server for aggregation and privacy protection, which outputs a federated unlearned model that is distinguishable from the completely retrained model. We highlight the contributions of this paper as follows:

- We propose a novel approximate federated unlearning algorithm, Appro-Fun, that is model agnostic and can be applied on the learned model in federated setting.
- A Newton's method-based local model unlearning mechanism is developed in our 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.
- The proposed Appro-Fun algorithm is evaluated on real datasets via experiments, which validate its superiority over the state-of-the-art.

This paper is organized as follows. The related works and the preliminaries are introduced in Section II and Section III, respectively. We detail our methodology in Section IV. Our proposed algorithm is implemented and evaluated in Section V. Finally, we conclude this paper and discuss future work in Section VI.

II. Related Works

Machine unlearning is formally defined by [12] and is classified into exact unlearning and approximate unlearning, according to their effectiveness.

Exact unlearning requires the distribution of an unlearned model should be exactly same as that of a model that is retrained on the dataset without the deleted data. This requirement is hard to be met in complicated models, so exact unlearning is mainly designed on simple machine learning models. In [11], authors designed unlearning algorithms for statistical query based models, such as Naive Bayesian classifier and SVM. Then, Ginart *et al.* [12] proposed the first unlearning method for unsupervised learning k-means cluster algorithm, which adopts stability and divide-and-conquer to improve unlearning efficiency. Two similar ideas [16], [17] are proposed to split training dataset into disjoint shards for reducing the retraining time. Following this, [13], [18] conducted unlearning algorithms on random forests algorithm, where the structure of decision tree is adjusted such that the retrained subtree can be minimized. The exact unlearning methods realize the unlearning requirement by retraining partial model, which can guarantee unlearning effectiveness but sacrifice unlearning efficiency.

On the contrary, approximate unlearning methods can achieve unlearning efficiency better than exact unlearning methods with slight cost in unlearning effectiveness. In approximate unlearning, the distribution of an unlearned model parameters is indistinguishable from that of a retrained model parameters. Instead of retraining partial of a learned model, approximate unlearning methods perform a post-processing on the learned model to obtain an approximation of the fully retrained model. In [19]–[21], authors used similar idea to achieve approximate unlearning by updating the trained models with stored gradients when some data points are removed. While, [22] and [23] 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.* [24] proposed adaptive machine unlearning that can handle arbitrary model classes and training methodologies. On the other hand, the authors of [14], [25] proposed differentially private data removal mechanisms, which can unlearn data from the learned models through hessian matrix. Golatkar and Wang [26], [27] focused on unlearning a specific class label from deep networks. Considering the computational cost of hessian matrix, Izzo *et al.* [15] found a sublinear algorithm to speed up unlearning from linear models efficiently. In federated learning, a few probability-based unlearning methods were utilized to solve the approximate unlearning with Bayesian [28], [29] and Monte Carlo [30]. While these methods fail to provide a theoretical guarantee on the performance of unlearned models. Therefore, it is still an open problem for us to explore the approximate unlearning solutions under federated settings.

III. Preliminary

In this section, we introduce the basic knowledge of federated learning (FL), serving as the foundation of unlearning hereafter. As an advanced distributed learning paradigm, FL allows a set of geographically distributed clients $\mathcal{K} = \{1, 2, \ldots, K\}$ to learn a global model stored on the FL server using their own local datasets $\mathcal{D}_k$ ($k \in \mathcal{K}$). 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 an FL system, all the clients' local datasets together compose a global dataset $\mathcal{D} = \bigcup_{k \in \mathcal{K}} \mathcal{D}_k$. 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. (1).

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

where $L_k(\cdot)$ is the loss function of client k, ξ_k^t is the model parameter of client k at iteration t, $|\mathcal{D}_k|$ is the size of $\mathcal{D}_k$, and $l(\xi_k^t, (x, y))$ is the loss value of model ξ_k^t on a data instance (x, y). Then, the clients' local models are uploaded to the server for model aggregation, and the federated model parameter ξ^t at iteration t is calculated via FedAvg algorithm [31] as follows.

$$\xi^t = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \xi_k^t. \tag{2}$$

According to the loss function of local clients and the aggregation algorithm, the optimization objective of federated learning system can be formulated as Eq. (3).

$$\min_{\xi^t \in \mathcal{W}} L(\xi^t) = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} L_k(\xi^t), \tag{3}$$

where $\mathcal{W} \in \mathbf{R}^d$ is the d-dimension hypothesis space of model parameters. To sum up, the computation flow of FL can be illustrated in Algorithm 1. Formally, an FL algorithm can be defined as $\mathscr{A}: \mathcal{D} \to \mathcal{W}$, which takes $\mathcal{D}$ as the input and outputs the trained federated model parameter w belonging to $\mathcal{W}$.

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

Algorithm 1 Federated Learning Algorithm

Input: the number of iteration T, the number of clients K, and the 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**
    $\xi_k^{t+1} \leftarrow$ ClientUpdate(k, ξ^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) **(Lipschitz Continuous Gradient)** $\forall w, w' \in \mathcal{W}$, the gradient of the loss function $L_k(\cdot)$ is Lipschitz continuous with $\mu > 0$ [32], [33]:

$$\|\nabla L_k(w) - \nabla L_k(w')\| \leq \mu \|w - w'\|. \quad (4)$$

2) **(Lipschitz Continuity)** $\forall w, w' \in \mathcal{W}$, the loss function $l(\cdot, (x, y))$ is Lipschitz continuous with $\iota > 0$ [32], [34]:

$$\|l(w, (x, y)) - l(w', (x, y))\| \leq \iota \|w - w'\|. \quad (5)$$

3) **(Strong Convexity)** $\forall w, w' \in \mathcal{W}$, the loss function $l(\cdot, (x, y))$ is strongly convex with $\tau > 0$ [32], [33]:

$$l(w, (x, y)) \geq l(w', (x, y)) + \nabla l(w', (x, y))^\top (w - w') + \frac{\tau}{2} \|w - w'\|^2, \quad (6)$$

where $\top$ is the transpose operation.

4) **(Hessian-Lipschitz)** The loss function $l(w, (x, y))$ is Hessian Lipschitz with constant M [25]:

$$\|\nabla^2 l(w, (x, y)) - \nabla^2 l(w', (x, y))\| \leq M \|w - w'\| \quad (7)$$

$$or \quad \|\nabla^3 l(w)\| \leq M, \forall w. \quad (8)$$

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

IV. Approximate Federated Unlearning

Suppose there is a set of clients $\mathcal{K}$ in the FL system. 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 w, 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 user is different. In addition to the federated model w, the FL algorithm $\mathscr{A}$ also produces a set of meta-data $\mathcal{M}$ (*e.g.,* gradients and intermediate models), which can be used in the unlearning procedure later and for other purposes. Accordingly, an unlearning algorithm can be defined as $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \rightarrow \mathcal{W}$, which takes the previously trained model $\mathscr{A}(\mathcal{D})$, the unlearning dataset $\mathcal{U}_j$, and the meta-data $\mathcal{M}$ as the inputs and outputs an unlearned model. Moreover, the definition of approximate federated unlearning is addressed as follows.

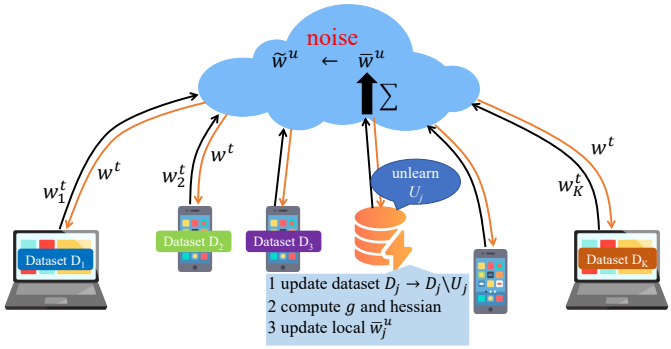


Fig. 1: The framework of proposed Appro-Fun algorithm.

Definition 1. (ϵ, δ)**-Approximate Federated Unlearning.** *Given an FL algorithm $\mathscr{A} : \mathcal{D} \rightarrow \mathcal{W}$ and an unlearning request $\mathcal{U}_j$ from client j, the unlearning algorithm $\mathscr{A}^u : (\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) \rightarrow \mathcal{W}$ is an (ϵ, δ)-approximate federated unlearning that unlearns $\mathcal{U}_j$ from $\mathscr{A}(\mathcal{D})$ if*

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

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

In Definition 1, the distribution of the unlearned model parameters is indistinguishable from that of model parameters trained from scratch on the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$.

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. 1. 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 IV-A; and (ii) at the second step, the server aggregates an intermediate unlearned model and protects it with carefully designed private noise, which is presented in Section IV-B.

A. 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 [35] 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,

$$L_j(w_j, \mathcal{D}_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j, (x, y)). \tag{9}$$

Rearranging Eq. (9) will give us Eq. (10) as follows

$$\begin{aligned} |\mathcal{D}_j| \cdot L_j(w_j, \mathcal{D}_j) =& \sum_{(x,y)\in\mathcal{D}_j^u} l(w_j, (x, y)) + \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x, y)) \\ =& |\mathcal{D}_j^u| \cdot L_j^u(w_j, \mathcal{D}_j^u) + |\mathcal{U}_j| \cdot L_j(w_j, \mathcal{U}_j), \end{aligned} \tag{10}$$

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. (11).

$$\nabla L_j^u(w_j, \mathcal{D}_j^u) = -\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} \cdot \nabla L_j(w_j, \mathcal{U}_j). \tag{11}$$

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. (10) can be written as

$$\begin{aligned} |\mathcal{D}_j| \cdot \nabla^2 L_j(w_j, \mathcal{D}_j) =& |\mathcal{D}_j^u| \cdot \nabla^2 L_j^u(w_j, \mathcal{D}_j^u) \\ &+ |\mathcal{U}_j| \cdot \nabla^2 L_j(w_j, \mathcal{U}_j). \end{aligned} \tag{12}$$

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

$$H = \frac{|\mathcal{D}_j|}{|\mathcal{D}_j^u|} \cdot \nabla^2 L_j(w_j, \mathcal{D}_j) - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} \cdot \nabla^2 L_j(w_j, \mathcal{U}_j). \tag{13}$$

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 $\nabla^2 L_j(w_j, \mathcal{U}_j)$, which can save more computation cost.

With the gradient $\nabla L_j^u(w_j, \mathcal{D}_j^u)$ obtained in Eq. (11) and the hessian matrix H, the local client j can unlearn $\mathcal{U}_j$ via Newton's method to get $\bar{w}_j^u$:

$$\begin{aligned} \bar{w}_j^u =& w_j - H^{-1} \nabla L_j^u(w_j, \mathcal{D}_j^u) \\ =& w_j + \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} H^{-1} \nabla L_j(w_j, \mathcal{U}_j), \end{aligned} \tag{14}$$

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. (14) is estimated through approximate hessian as solved in [36], [37] to save time. The computation process of the above local model unlearning is presented in lines 1-5 of Algorithm 2.

Algorithm 2 Approximate Federated Unlearning (Appro-Fun)

Input: the trained model w from Algorithm 1, 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. (11)
3: compute hessian matrix H as Eq. (13)
4: update local model $\bar{w}_j^u$ of client j as Eq. (14)
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 $\tilde{w}^u$

B. 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 a further operation to remove the impact of unlearned dataset $\mathcal{U}_j$ as shown in lines 6-8 of Algorithm 2. 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 a temporarily unlearned version of the previously trained federated model w, (*i.e.,* the output of Algorithm 1). Recall that in Definition 1, 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 [25]. 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 [38], which can guide the noise scale adding into $\bar{w}^u$. Specifically, Theorem A.1 in [38] 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 2), the noise scale σ should satisfy

$$\sigma \geq \frac{\max \|w^u - \bar{w}^u\| \sqrt{2\ln(1.25/\delta)}}{\epsilon}. \tag{15}$$

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 1. *Let w be the model parameter of Algorithm 1 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. (16)*:*

$$\|w - w^u\| \leq \frac{2m\iota}{|\mathcal{D}|\tau}, \tag{16}$$

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

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:

$$L_j(w, \mathcal{D}_j) = \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w, (x, y)) \tag{17}$$

$$L_j^u(w, \mathcal{D}_j^u) = \frac{1}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{D}_j^u} l(w, (x, y)) \tag{18}$$

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. (17) and the local minimizer of Eq. (18) 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

$$\begin{aligned}
&L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \\
=&\frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j^u, (x, y)) - \frac{1}{|\mathcal{D}_j|} \sum_{(x,y)\in\mathcal{D}_j} l(w_j, (x, y)) \\
=&\frac{1}{|\mathcal{D}_j|}[\sum_{(x,y)\in\mathcal{D}_j^u} l(w_j^u, (x, y)) - \sum_{(x,y)\in\mathcal{D}_j^u} l(w_j, (x, y)) \\
&+ \sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x, y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x, y))] \\
=&\frac{1}{|\mathcal{D}_j|}[|\mathcal{D}_j^u|(L_j^u(w_j^u, \mathcal{D}_j^u) - L_j^u(w_j, \mathcal{D}_j^u)) \\
&+ \sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x, y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x, y))] \\
\overset{(i)}{\leq}&\frac{1}{|\mathcal{D}_j|}[\sum_{(x,y)\in\mathcal{U}_j} l(w_j^u, (x, y)) - \sum_{(x,y)\in\mathcal{U}_j} l(w_j, (x, y))] \\
\overset{(ii)}{\leq}&\frac{m\iota}{|\mathcal{D}_j|}\|w_j^u - w_j\|,
\end{aligned} \tag{19}$$

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 (3)), we can get the following equation

$$L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \geq \frac{\tau}{2}\|w_j^u - w_j\|^2 \tag{20}$$

Combining the Eq. (19) and Eq. (20), we can derive the following inequality

$$\begin{aligned}
&\frac{\tau}{2}\|w_j^u - w_j\|^2 \leq L_j(w_j^u, \mathcal{D}_j) - L_j(w_j, \mathcal{D}_j) \leq \frac{m\iota}{|\mathcal{D}_j|}\|w_j^u - w_j\| \\
&\Rightarrow \|w_j^u - w_j\| \leq \frac{2m\iota}{|\mathcal{D}_j|\tau}
\end{aligned} \tag{21}$$

This inequality Eq. (21) 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

$$\|w^u - w\| \leq \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \cdot \frac{2m\iota}{|\mathcal{D}_j|\tau} = \frac{2m\iota}{|\mathcal{D}|\tau}. \tag{22}$$

This finishes the proof of Lemma 1. □

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

Theorem 1. *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 2. The distance between w^u and $\bar{w}^u$ is bounded by Eq.* (23)*:*

$$\|w^u - \bar{w}^u\| \leq \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3}, \tag{23}$$

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

The proof of Theorem 1 is presented in online Appendix A.

By combining Eq. (15) and Eq. (23), we obtain the setting for σ in Appro-Fun algorithm:

$$\sigma \geq \frac{2m^2\iota^2 M\sqrt{2\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon}. \tag{24}$$

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 2.

Theorem 2. *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 2). The loss distance can be bounded by:*

$$\begin{aligned}
\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \leq &\frac{2m^2\iota^3 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} + \frac{2m^2\iota^2 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\
&+ \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2).
\end{aligned} \tag{25}$$

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

The detailed proof of Theorem 2 can be found in online Appendix B.

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 3.

Theorem 3. *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 2). The loss distance can be bounded by:*

$$\begin{aligned}
\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \leq &\frac{m|\mathcal{D}_j|\iota^2}{(n_j - m)|\mathcal{D}|\tau} + \frac{2m^2\iota^3 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\
&+ \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2).
\end{aligned} \tag{26}$$

TABLE I: Structure of neural networks

L	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

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. The detailed proof of Theorem 3 can be found in online Appendix C.

V. Experiments

In this section, we conduct intensive experiments to validate our Appro-Fun algorithm from unlearning effectiveness and efficiency aspects.

A. Experiment Settings

Fashion-MNIST dataset [39] and CIFAR-10 [40] dataset are adopted in our experiments. The experiments are implemented mainly by Pytorch and evaluated on Google Colab Platform with Tesla T4 GPU. Model structures of Fashion-MNIST and CIFAR-10 datasets is shown in the following Table I. The source code will be made publicly available once accepted.

1) Learning and Unlearning Scenario: In the federated learning system, we set the number of clients K to be 10, 20, and 50. The training dataset is separated to 50 disjoint shards with different number of data points and different class labels. This is to simulate the real application scenario of federated learning in non-i.i.d. settings. For different number of clients K in the system, we randomly pick K shards of data without repetition and assign to each client as local dataset. By this setting, we simulate the realistic application scenario, that is, the more participant clients, the more training data. Our proposed Appro-Fun algorithm 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 submit 5 unlearning requests, so there are $0.5K$ unlearning requests in total. These requests are processed via Algorithm 2 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 [11], [12]. 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 $p \leq 20\%$.

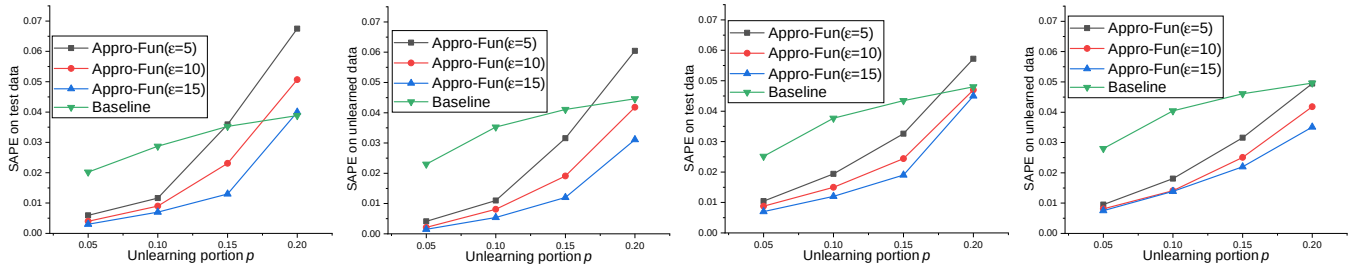
2) Baseline Models: As an exact and most effective unlearning method, retraining the federated model from scratch on the remaining dataset $\mathcal{D}^u$ is adopted as the first baseline, which can be defined as $\mathscr{A}^u(\mathscr{A}(\mathcal{D}), \mathcal{U}_j, \mathcal{M}) = \mathscr{A}(\mathcal{D}^u)$. In addition, to evaluate our Appro-Fun algorithm, one state-of-the-art approximate federated unlearning method published in INFOCOM 2022 [41], 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.

B. 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.

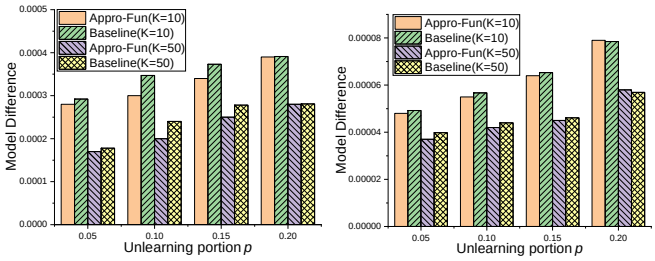
1) SAPE Comparison: We adopt Symmetric Absolute Percentage Error (SAPE) to measure the accuracy difference between the retrained federated model and the unlearned federated model, which is used as an effectiveness metric in many unlearning literatures [19], [21], [41]. SAPE is calculated as: $SAPE(Acc_1, Acc_2) = \frac{|Acc_1 - Acc_2|}{|Acc_1| + |Acc_2|} \times 100\%$ 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 unlearning portion p. First, the SAPE value on test data of Fashion-MNIST dataset is shown in Fig. 2(a), 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. (24)), 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. 2(a), 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



(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

Fig. 2: SAPE comparison between Appro-Fun and baseline with different unlearning portion p and ϵ.



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

Fig. 3: Model difference between Appro-Fun and baseline.

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. 2(b) depicts the trend of SAPE on unlearned data in Fashion-MNIST dataset. Similar 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. 2(b)). 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. 2(c) and Fig. 2(d). 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.

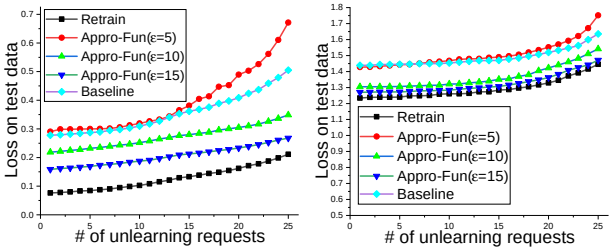
2) Model and Loss Comparison: As pointed out by [26], the SAPE value may not be enough to qualify the unlearning effectiveness. For complex machine learning models, the difference between unlearned model parameters and the retrained model parameters is also a vital metric. Therefore, we calculate the 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. 3(a), 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. 3(a), 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. 3(b), where our Appro-Fun algorithm has smaller model difference compared with the baseline. While, only when p=0.2, our unlearned model difference is slightly larger than the baseline. This may be caused by the injected noise of our Appro-Fun algorithm as we proved in Theorem 1, the larger amount 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. 4(a), 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

TABLE II: 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×



(a) Loss on test F-MNIST (b) Loss on test CIFAR-10

Fig. 4: Loss comparison on test data.

TABLE III: 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

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. 4(b), the same conclusion can be drawn, which means our Appro-Fun algorithm can achieve much smaller loss on test data when ϵ is larger.

3) 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 [42]–[44]. MIA infers whether a data sample is in the training dataset of a model, so for the deleted data, a lower MIA accuracy (around 50%) means that the unlearning algorithm has stronger privacy protection. In Table III, 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.

C. Unlearning Efficiency of Appro-Fun

First of all, the original federated model is trained to converge using Algorithm 1. 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 II, the 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 IV-A) than the method used by the baseline, reaching faster computation. 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 one (F-MNIST). For a complex dataset, retraining process requires more recalculation of gradient 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.

VI. Conclusion & Further Work

In this paper, we study the federated unlearning problem and propose an (ϵ, δ)-approximate federated unlearning algorithm, Appro-Fun. 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 Appro-Fun algorithm outperforms the state-of-the-art baseline in terms of effectiveness and efficiency. As a pilot work on federated unlearning, we suppose that all clients' local datasets are disjoint, which facilitates data deletion at the server without request conflict (*i.e.*, one client wants to remove a data instance while another does not want). Considering local clients' data characteristics (such as overlapping, correlated, and common datasets) in reality, more complicated scenarios will be investigated in our next-step work.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] F. T. Commission *et al.*, "California company settles ftc allegations it deceived consumers about use of facial recognition in photo storage app," 2021.

[6] 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.

[7] Y. Wang, C. Si, and X. Wu, "Regression model fitting under differential privacy and model inversion attack," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[8] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model." *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] L. Graves, V. Nagisetty, and V. Ganesh, "Amnesiac machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 13, 2021, pp. 11 516–11 524.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] J. Wang, S. Guo, X. Xie, and H. Qi, "Federated unlearning via class-discriminative pruning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 622–632.

[28] Y. Chen, S. Zhang, and B. K. H. Low, "Near-optimal task selection for meta-learning with mutual information and online variational bayesian unlearning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 9091–9113.

[29] J. Gong, J. Kang, O. Simeone, and R. Kassab, "Forget-svgd: Particle-based bayesian federated unlearning," in *2022 IEEE Data Science and Learning Workshop (DSLW)*. IEEE, 2022, pp. 1–6.

[30] Q. P. Nguyen, R. Oikawa, D. M. Divakaran, M. C. Chan, and B. K. H. Low, "Markov chain monte carlo-based machine unlearning: Unlearning what needs to be forgotten," *arXiv preprint arXiv:2202.13585*, 2022.

[31] 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.

[32] 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*, vol. 15, pp. 3454–3469, 2020.

[33] 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.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] 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.

[42] 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

[43] 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.

[44] 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.

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

APPENDIX

A. Proof of Theorem 1

Proof. Based on Eq. (17) and Eq. (18), 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,

$$\begin{aligned}
&\nabla L_j^u(w_j^u, \mathcal{D}_j^u) = \nabla L_j^u(w_j, \mathcal{D}_j^u) + \nabla^2 L_j^u(w_j, \mathcal{D}_j^u)[w_j^u - w_j] \\
&+ \frac{1}{2}\nabla^3 L_j^u(w_j, \mathcal{D}_j^u)[w_j^u - w_j]^2 \\
&\overset{(i)}{\Rightarrow} \| - \nabla L_j^u(w_j, \mathcal{D}_j^u) - H[w_j^u - w_j]\| \\
&= \frac{1}{2}\nabla^3 L_j^u(w_j, \mathcal{D}_j^u)\|w_j^u - w_j\|^2 \\
&\Rightarrow \|\nabla L_j^u(w_j, \mathcal{D}_j^u) + H[w_j^u - w_j]\| \overset{(ii)}{\leq} \frac{M}{2}\|w_j^u - w_j\|^2. \quad (27)
\end{aligned}$$

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

$$\begin{aligned}
&\nabla L_j^u(w_j, \mathcal{D}_j^u) = \frac{1}{|\mathcal{D}_j^u|}\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j, (x,y)) \\
=&\frac{1}{|\mathcal{D}_j^u|}[\sum_{(x,y)\in\mathcal{D}_j} \nabla l(w_j, (x,y)) - \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x,y))] \\
=&\frac{1}{|\mathcal{D}_j^u|}[|\mathcal{D}_j|\nabla L_j(w_j, \mathcal{D}_j) - \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x,y))] \\
\overset{(i)}{=}& -\frac{1}{|\mathcal{D}_j^u|}\sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j, (x,y)) = -\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}\nabla L_j(w_j, \mathcal{U}_j), \quad (28)
\end{aligned}$$

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 2 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,

$$\begin{aligned}
&w_j^u - \bar{w}_j^u = w_j^u - [w_j + \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j, \mathcal{U}_j)] = \beta \\
&\Rightarrow w_j^u - w_j = \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j, \mathcal{U}_j) + \beta \quad (29)
\end{aligned}$$

Substituting Eq. (28) and Eq. (29) into Eq. (27), we can get the inequality

$$\begin{aligned}
&\| - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}\nabla L_j(w_j, \mathcal{U}_j) + H[\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j, \mathcal{U}_j) + \beta]\| \\
&\leq \frac{M}{2}\|w_j^u - w_j\|^2. \quad (30)
\end{aligned}$$

Rearranging Eq. (30), we can simplify the inequality to Eq. (31)

$$\|H\beta]\| = \|\nabla^2 L_j^u(w_j, \mathcal{D}_j^u)\beta]\| \leq \frac{M}{2}\|w_j^u - w_j\|^2. \quad (31)$$

Due to the τ-strongly convexity of loss function, we can have a corollary [45]

$$\|\nabla^2 L_j^u(w_j, \mathcal{D}_j^u)\beta]\| \geq \tau\|\beta\|. \quad (32)$$

Combining the Eq. (31) and Eq. (32), we can have the upper bound distance between w_j^u and $\bar{w}_j^u$

$$\|w_j^u - \bar{w}_j^u\| = \|\beta\| \leq \frac{M}{2\tau}\|w_j^u - w_j\|^2. \quad (33)$$

Here, $\|w_j^u - w_j\|$ is proved with an upper bound in Eq. (21) of Lemma 1. After the aggregation on server side, we can get the distance between w^u and $\bar{w}^u$ in the following equation,

$$\|w^u - \bar{w}^u\| \leq \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3}. \quad (34)$$

This ends the proof of Theorem 1. □

B. Proof of Theorem 2

Proof.

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \\
=&\mathbb{E}\{L(\tilde{w}^u) - L(w^u) + L(w^u) - L(w^{u*})\} \\
\leq&\mathbb{E}\{L(\tilde{w}^u) - L(w^u)\} + \mathbb{E}\{L(w^u) - L(w^{u*})\} \\
\overset{(i)}{\leq}&\mathbb{E}\{L(\tilde{w}^u) - L(w^u)\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \\
\overset{(ii)}{\leq}&\iota\mathbb{E}\{\|\tilde{w}^u - w^u\|\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \quad (35)
\end{aligned}$$

where the inequality (i) holds because the convergence bound of federated learning is proved by [33], and the inequality (ii) holds due to the Lipschitzness of loss function $l(w, (x,y))$.

Then, based on Theorem 1, we can calculate the expectation of $\|\tilde{w}^u - w^u\|$ as follows

$$\begin{aligned}
\mathbb{E}\{\|\tilde{w}^u - w^u\|\} =&\mathbb{E}\{\|\tilde{w}^u - \bar{w}^u + \bar{w}^u - w^u\|\} \\
\leq&\mathbb{E}\{\|\tilde{w}^u - \bar{w}^u\|\} + \mathbb{E}\{\|\bar{w}^u - w^u\|\} \\
\leq&\mathbb{E}\{\|N\|\} + \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} \\
\leq&\sqrt{d}\sigma + \frac{2m^2\iota^2 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} \quad (36)
\end{aligned}$$

where N is the gaussian noise added in each unlearning process as shown in Line 8 of Algorithm 2.

Combining the Eq. (35), Eq. (36) and the value of σ, we can prove Theorem 2 as follows

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^{u*})\} \leq \frac{2m^2\iota^3 M}{|\mathcal{D}||\mathcal{D}_j|\tau^3} + \frac{2m^2\iota^2 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\
&+ \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2). \quad (37)
\end{aligned}$$

This finishes the proof of Theorem 2. □

C. Proof of Theorem 3

Proof.

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \\
=&\mathbb{E}\{L(\tilde{w}^u) - L(w) + L(w) - L(w^*)\} \\
\leq&\mathbb{E}\{L(\tilde{w}^u) - L(w)\} + \mathbb{E}\{L(w) - L(w^*)\} \\
\overset{(i)}{\leq}&\mathbb{E}\{L(\tilde{w}^u) - L(w)\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \\
\overset{(ii)}{\leq}&\iota\mathbb{E}\{\|\tilde{w}^u - w\|\} + \frac{\mu}{\iota(T - 1 + \frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2) \quad (38)
\end{aligned}$$

where w is the output of Algorithm 1, the inequality (i) holds because the convergence bound of federated learning is proved by [33], and the inequality (ii) holds due to the Lipschitzness of loss function $l(w,(x,y))$.

According to the Appro-Fun Algorithm 2, we can get

$$\begin{aligned}
&\mathbb{E}\{\|\tilde{w}^u - w\|\} = \mathbb{E}\{\|\bar{w}^u + N - w\|\} \\
\overset{(i)}{=}&\mathbb{E}\{\|\frac{|\mathcal{D}_j|}{|\mathcal{D}|}(\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}H^{-1}\nabla L_j(w_j,\mathcal{U}_j)) + N\|\} \\
\leq&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|}\mathbb{E}\{\|H^{-1}\nabla L_j(w_j,\mathcal{U}_j)\|\} + \mathbb{E}\{\|N\|\} \\
\overset{(ii)}{\leq}&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\mathbb{E}\{\|\nabla L_j(w_j,\mathcal{U}_j)\|\} + \mathbb{E}\{\|N\|\} \\
=&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\mathbb{E}\{\|\frac{1}{|\mathcal{U}_j|}\sum_{(x,y)\in\mathcal{U}_j}\nabla l(w_j,(x,y))\|\} + \mathbb{E}\{\|N\|\} \\
\overset{(iii)}{\leq}&\frac{|\mathcal{D}_j|}{|\mathcal{D}|}\frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|\tau}\frac{|\mathcal{U}_j|\iota}{|\mathcal{U}_j|} + \sqrt{d}\sigma = \frac{m|\mathcal{D}_j|\iota}{(n_j-m)|\mathcal{D}|\tau} + \sqrt{d}\sigma, \qquad (39)
\end{aligned}$$

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. (14). 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. (38), Eq. (39), and Eq. (24), we can prove Theorem 3 as follows,

$$\begin{aligned}
&\mathbb{E}\{L(\tilde{w}^u) - L(w^*)\} \leq \frac{m|\mathcal{D}_j|\iota^2}{(n_j-m)|\mathcal{D}|\tau} + \frac{2m^2\iota^3 M\sqrt{2d\ln(1.25/\delta)}}{|\mathcal{D}||\mathcal{D}_j|\tau^3\epsilon} \\
&+ \frac{\mu}{\iota(T-1+\frac{8\iota}{\mu})}(\frac{2\Delta}{\mu} + 4\iota\mathbb{E}\{C^0\}^2). \qquad (40)
\end{aligned}$$

□