Certified Unlearning in Decentralized Federated Learning

Hengliang Wu[†*], Youming Tao[†‡*], Anhao Zhou[†], Shuzhen Chen[§], Falko Dressler[‡], Dongxiao Yu[†]

[†]School of Computer Science and Technology, Shandong University, Qingdao, China
[‡]School of Electrical Engineering and Computer Science, TU Berlin, Berlin, Germany
[§]College of Computer Science and Technology, Ocean University of China, Qingdao, China

Abstract—**Driven by the right to be forgotten (RTBF), machine unlearning has become an essential requirement for privacy-preserving machine learning. However, its realization in decentralized federated learning (DFL) remains largely unexplored. In DFL, clients exchange local updates only with neighbors, causing model information to propagate and mix across the network. As a result, when a client requests data deletion, its influence is implicitly embedded throughout the system, making removal difficult without centralized coordination. We propose a novel certified unlearning framework for DFL based on Newton-style updates. Our approach first quantifies how a client's data influence propagates during training. Leveraging curvature information of the loss with respect to the target data, we then construct corrective updates using Newton-style approximations. To ensure scalability, we approximate second-order information via Fisher information matrices. The resulting updates are perturbed with calibrated noise and broadcast through the network to eliminate residual influence across clients. We theoretically prove that our approach satisfies the formal definition of certified unlearning, ensuring that the unlearned model is difficult to distinguish from a retrained model without the deleted data. We also establish utility bounds showing that the unlearned model remains close to retraining from scratch. Extensive experiments across diverse decentralized settings demonstrate the effectiveness and efficiency of our framework.**

Index Terms—**Machine Unlearning, Federated Learning, Decentralized Learning**

I. INTRODUCTION

Federated Learning (FL) enables collaborative model training while keeping user data on local devices, sharing only model updates among participants [1], [2], which offers a promising approach to privacy-preserving learning. However, conventional FL relies on a centralized server to aggregate updates, which introduces several limitations, including communication bottlenecks, heavy server load, and vulnerability to single-point failures. To overcome these issues, *Decentralized Federated Learning* (DFL) has emerged as an attractive alternative [3], [4]. In DFL, clients exchange model updates directly with neighboring peers rather than communicating with a central coordinator. This peer-to-peer design distributes communication costs across the network, improves scalability and fault tolerance, and eliminates reliance on a single trusted server. As a result, DFL has attracted increasing attention in

both academic research and real-world deployments, such as large-scale distributed and vehicular systems [5], [6].

The privacy guarantees of FL and DFL are increasingly challenged. Prior work has demonstrated that sensitive user data can be extracted from trained models through inference and reconstruction attacks [7]. In response, governments have enacted stricter data protection regulations, most notably the *right to be forgotten* (RTBF) [8], [9], which grants users the right to request the deletion of their sensitive data from any models derived from it. A naïve solution is to retrain the model from scratch after removing the targeted data, but this approach is often prohibitively expensive. This challenge has motivated the development of *machine unlearning*, which aims to efficiently remove the influence of specific data points or clients without full retraining [10].

To meet RTBF requirements in federated settings, *Federated Unlearning* (FU) has recently emerged [11]. Existing FU methods primarily focus on centralized FL architectures [12]–[14]. However, extending unlearning to decentralized environments, referred to as *Decentralized Federated Unlearning* (DFU), remains largely unexplored. Unlearning in decentralized settings is inherently difficult due to the following challenges.

(1) Implicit influence propagation. Although communication occurs only among neighboring clients, each local model implicitly incorporates gradient information from all participants through repeated mixing. When a client requests data deletion, its contribution has already diffused throughout the network. Precisely quantifying and removing this mixed influence without centralized coordination is non-trivial.

(2) Efficiency constraints. DFU must be computationally and storage efficient, as decentralized learning often involves resource-constrained edge devices. Existing DFU approaches frequently rely on storing historical updates or auxiliary seed models to support unlearning [15], [16], resulting in substantial memory and computation overhead. This motivates the need for unlearning algorithms specifically designed for decentralized environments with minimal storage and computation requirements.

(3) Certified guarantees and utility preservation. A principled unlearning algorithm must satisfy *certified unlearning* guarantees, ensuring that the unlearned model is difficult to distinguish from one trained without the deleted data [17],

[18]. At the same time, it must preserve model utility, which requires tight bounds on the deviation between the unlearned model and the retrained model. Existing DFU methods either lack formal unlearning guarantees [15] or fail to rigorously characterize the resulting utility loss [16].

Our Contributions. To address these challenges, we propose a generic and efficient framework for *certified decentralized federated unlearning* applicable to arbitrary network topologies. Our key contributions are summarized as follows:

- We propose a novel DFU framework that achieves certified unlearning in decentralized learning. Our approach quantifies peer-to-peer influence using the doubly stochastic property of network mixing, constructs corrective updates via Newton-style approximations, and injects calibrated noise to ensure indistinguishability. To improve efficiency, we approximate second-order information using the Fisher Information Matrix (FIM).

- We provide rigorous theoretical guarantees for our method. Specifically, we prove that our algorithm satisfies the formal definition of (ϵ, δ)-certified unlearning and establish error bounds between the unlearned model and a fully retrained model, ensuring utility preservation. To the best of our knowledge, this is the first certified DFU framework that does not require storing historical updates.

- We conduct extensive experiments across diverse decentralized settings. The results demonstrate that our method achieves accuracy comparable to retraining from scratch while reducing time cost by up to 97%, highlighting both its effectiveness and efficiency.

II. RELATED WORK

Machine unlearning. Machine unlearning was first formalized by Cao et al. [19], who defined unlearning as producing a model equivalent to one obtained by retraining after removing the target data. Subsequent work has proposed alternative definitions, broadly categorized into *exact* and *approximate* unlearning [20]. Exact unlearning requires that the distribution of the unlearned model be identical to that of a model retrained from scratch on the retained data [21], thereby achieving statistical indistinguishability. Due to its stringent requirements, exact unlearning is often computationally infeasible. To address this limitation, approximate unlearning, most notably (ϵ, δ)-unlearning, has been widely adopted [17], [18]. Under this definition, the output distribution of the unlearning algorithm is *difficult to distinguish* from that of retraining, with the distinguishing advantage bounded by (ϵ, δ). While approximate unlearning relaxes the requirements of exact unlearning and significantly reduces computation and memory costs, existing methods are primarily designed for centralized systems and do not extend to decentralized learning settings [18], [22], [23].

Federated unlearning. With the rise of federated learning, machine unlearning has been extended to federated settings, giving rise to *Federated Unlearning* (FU) [11]. FU methods typically consider three levels of granularity: sample-wise, class-wise, and client-wise unlearning [24], [25]. Sample-wise unlearning removes individual data points, class-wise unlearning deletes all samples of a specific class, and client-wise unlearning removes all contributions from a particular client [25]. Most existing FU algorithms assume a centralized FL architecture [14]. Prior work explores techniques such as momentum degradation [26], model perturbation [27], and other heuristic strategies [11]. In contrast, our approach is grounded in Newton-style updates, offering a principled and intuitive mechanism for influence removal. Moreover, it is compatible with a broad class of Empirical Risk Minimization (ERM) objectives, enabling strong generalization performance.

Decentralized unlearning. Research on unlearning in fully decentralized federated learning remains limited. Ye et al. [15] proposed an exact decentralized unlearning method based on model distillation. While empirical results suggest effectiveness, the absence of formal unlearning guarantees limits its reliability. More recently, Qiao et al. [16] introduced the first provable client-wise decentralized unlearning algorithm. Their method estimates client influence using aggregated historical gradients and applies perturbations to achieve unlearning. However, this approach requires storing historical gradient information and performing multiple additional training rounds to recover accuracy, leading to substantial storage and computational overhead. These limitations hinder its applicability in resource-constrained decentralized environments. In contrast, our work aims to design an efficient and certified decentralized unlearning framework that avoids historical update storage while providing rigorous unlearning and utility guarantees.

III. PROBLEM FORMULATION

A. Decentralized Learning

We consider a fully decentralized FL system with N clients, where clients communicate exclusively with their neighbors and no central server is present. Each client i ($i \in [N]$) holds a local dataset S_i containing n_i samples drawn from a client-specific local distribution $\mathcal{D}_i$. The communication structure of the system is captured by a graph (V, E, Q). Here, V is the set of N nodes representing clients, and E denotes the set of communication links between clients. The matrix $Q \in \mathbb{R}^{N \times N}$ is a mixing matrix that characterizes how model updates are weighted when exchanging among clients. Specifically, the entry Q_{ij} quantifies the influence of client-j on client i during local aggregation. We assume that Q is a symmetric doubly stochastic matrix, i.e., $Q_{ij} = Q_{ji}$ for all i, j and $\sum_i Q_{ji} = 1$ for all j. If clients i and j are not connected, $Q_{ij} = 0$. This ensures that communication is restricted to neighbouring clients and that the aggregation process preserves consensus properties. The matrix Q further satisfies the following standard spectral assumption.

Assumption 1. Given the symmetric doubly stochastic matrix Q, and let $\rho_k(Q)$ denote the k-th largest eigenvalue of Q. Define $\rho := (\max\{|\rho_2(Q)|, |\rho_N(Q)|\})^2$. We assume $\rho < 1$.

The learning objective is to solve the following decentralized optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x), \quad (1)$$

where $f_i(x)$ is the local empirical risk function defined as $f_i(x) = \frac{1}{n_i} \sum_{t=1}^{n_i} F_i(x, \xi_{i,t})$ and $F_i(x, \xi_{i,t})$ is the loss incurred by model parameter x on data sample $\xi_{i,t} \in S_i$. During training, clients collaboratively optimize (1) through local updates and communication. After total K iterations, each client maintains a local model $x_{K,i}$, and the final global model is given by the average $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_{K,i}$, which is assumed to converge to a stationary solution of (1).

B. Decentralized Unlearning

Let $\mathcal{S}$ denote the data universe and $\mathcal{X} \subseteq \mathbb{R}^d$ denote the model space. Client $i \in [N]$ holds a dataset $S_i \in \mathcal{S}^{n_i}$, and the collection of all clients' datasets is denoted by $\mathbf{S} := (S_1, \ldots, S_N) \in \mathcal{S}^{n_1} \times \cdots \times \mathcal{S}^{n_N}$. A decentralized learning algorithm is a (randomized) mapping $\mathcal{A} : \mathcal{S}^{n_1} \times \cdots \times \mathcal{S}^{n_N} \to \mathcal{X}^N$, which outputs a tuple of local models $\mathbf{X} := (x_1, \ldots, x_N) \in \mathcal{X}^N$. For a deletion request issued by client i, let $U_i \subseteq S_i$ with $|U_i| = m_i$ and define the retained dataset $R_i := S_i \setminus U_i$. We use $\mathbf{S}^{(-i)} := (S_1, \ldots, R_i, \ldots, S_N)$ to denote the dataset collection after deleting U_i from client i. Let $\mathcal{T}$ denote the space of auxiliary information available to the unlearning procedure (e.g., statistics computed during training). A decentralized unlearning algorithm is a (possibly randomized) mapping $\mathcal{G} : \mathcal{P}(\mathcal{S}) \times \mathcal{X}^N \times \mathcal{T} \to \mathcal{X}^N$, which takes as input the deletion request U_i, the learned models $\mathcal{A}(\mathbf{S})$, and auxiliary information $T(\mathbf{S}) \in \mathcal{T}$, and outputs the unlearned models $\widetilde{\mathbf{X}} := (\widetilde{x}_1, \ldots, \widetilde{x}_N) \in \mathcal{X}^N$.

We extend the notion of certified (ϵ, δ)-unlearning [17] to the decentralized setting, formalized as follows.

Definition 1 (Decentralized (ϵ, δ)-unlearning)**.** We say that the pair $(\mathcal{A}, \mathcal{G})$ satisfies *certified (ϵ, δ)-unlearning* in decentralized learning if, for any measurable set $W \subseteq \mathcal{X}^N$,

$$\mathbb{P}(\mathcal{G}(U_i, \mathcal{A}(\mathbf{S}), T(\mathbf{S})) \in W) \leq e^\epsilon \mathbb{P}\Big(\mathcal{A}(\mathbf{S}^{(-i)}) \in W\Big) + \delta,$$

$$\mathbb{P}\Big(\mathcal{A}(\mathbf{S}^{(-i)}) \in W\Big) \leq e^\epsilon \mathbb{P}(\mathcal{G}(U_i, \mathcal{A}(\mathbf{S}), T(\mathbf{S})) \in W) + \delta.$$

IV. CERTIFIED DECENTRALIZED UNLEARNING FRAMEWORK

A. Decentralized Learning Process

During training, each client i collaboratively optimizes the global objective in a fully decentralized manner. The learning process proceeds for K iterations and follows standard decentralized stochastic gradient descent (DSGD) scheme. At each iteration $k \in [K]$, the training proceeds as the following steps.

Step 1 (Local stochastic gradient update). Client i samples a data point $\xi_{k,i}$ uniformly at random from its local dataset S_i and computes the stochastic gradient $\nabla F_i(x_{k,i}, \xi_{k,i})$ based on its current local model $x_{k,i}$.

Step 2 (Model exchange and aggregation). Each client exchanges its local model $x_{k,i}$ with its neighbours and aggregates the received models using the mixing Q. The aggregated model is given by $x_{k+\frac{1}{2},i} = \sum_j Q_{ij} x_{k,j}$.

Step 3 (Local model update). Client i updates its local model using a gradient descent step based on the aggregated model: $x_{k+1,i} = x_{k+\frac{1}{2},i} - \gamma \nabla F_i(x_{k,i}, \xi_{k,i})$, where $\gamma > 0$ denotes the learning rate.

Step 4 (Termination and statistic collection). After K iterations, each client i obtains a local model $x_{K,i}$ that approximates a stationary solution of (1), with convergence guarantees under standard assumptions, see e.g., [3], [28]. If a deletion request is issued after iteration k, each client i additionally computes and stores the auxiliary statistic $T_i(S_i) := \{\nabla^2 f_i(x_{k,i})\}$, i.e., the Hessian of the local empirical risk evaluated at the model state corresponding to the request time. The auxiliary statistic $T_i(S_i)$ is used exclusively for certified unlearning. Let d be the model dimension. Importantly, storing $T_i(S_i)$ requires $\mathcal{O}(d^2)$ memory , which is independent of both the local dataset size n_i and the number of deleted samples m_i, making it scalable and suitable for resource-constrained decentralized environments.

B. Decentralized Unlearning for Sample-Level Removal

We start with *sample-wise certified decentralized unlearning*, where a client requests to delete a subset of its local data.

Influence propagation in decentralized learning. Due to iterative peer-to-peer communication, the contribution of each client is gradually mixed into all local models. As a result, removing the effect of deleted samples requires accounting for how model updates propagate through the network. While prior work [16] estimates such influence by tracking historical gradients, this approach incurs substantial storage and computation overhead. In contrast, we exploit the structural properties of the mixing matrix Q. When Q is symmetric and doubly stochastic, repeated local averaging leads to asymptotic consensus among clients. Specifically, peer-to-peer influence weights converge to the uniform distribution $1/N$ as the number of iterations increases [3]. This behavior is formalized in Lemma 1, which states that the peer-to-peer influence weights converge exponentially fast to uniform consensus. This allows us to approximate the influence of each client on any other client after k iterations using a uniform weighting scheme, eliminating the need to store historical gradient information and significantly reducing memory and computational costs.

Lemma 1. Under Assumption 1, the mixing matrix Q satisfies $\left\| Q^k - \frac{1}{N} \mathbf{1}_N \mathbf{1}_N^\top \right\|^2 \leq \rho^k$, $\forall, k \in \mathbb{N}$, where $\rho < 1$ is defined in Assumption 1 and $\mathbf{1}_N \in \mathbb{R}^N$ denotes the all-ones vector.

Newton-style correction via influence functions. Consider a client $c \in [N]$ that issues a deletion request $U_c \subseteq S_c$ at iteration k, where $|U_c| = m_c$. To estimate the impact of removing these samples, we draw inspiration from influence functions [29] and approximate the parameter change induced by deleting U_c using a second-order expansion of the local

empirical risk around the current model $x_{k,c}$. We made the following standard assumptions on loss functions.

Assumption 2. Loss functions $F_i(x, \xi)$ are λ-strongly convex, μ-smooth, L-Lipschitz with M-Lipschitz Hessian.

We define the empirical Hessian over the retained data as

$$\hat{H}_c = \frac{1}{n_c - m_c} \sum_{\xi_c \in S_c \setminus U_c} \nabla^2 F_c(x_{k,c}, \xi_c). \quad (2)$$

Using this curvature information, we construct a Newton-style corrective update, which approximates the change in model parameters caused by deleting the samples in U_c.

$$x_c^\Delta = \frac{1}{n_c - m_c} (\hat{H}_c)^{-1} \sum_{\xi_c \in U_c} \nabla F_c(x_{k,c}, \xi_c). \quad (3)$$

Notably, this approximation relies only on second-order information of the loss landscape and gradient evaluations at the deleted samples, without requiring access to raw data beyond what is already locally available.

Perturbation and decentralized aggregation. To ensure certified unlearning guarantees, client c perturbs the corrective update by adding Gaussian noise $\nu_c \sim \mathcal{N}(0, \sigma_c^2 I_d)$, where the noise variance σ_c^2 is carefully calibrated based on the sensitivity of the update. The perturbed update is then broadcast through the network. Leveraging the uniform influence approximation, all clients aggregate the received corrective updates using a uniform weight $1/N$ and apply the aggregated correction to their local models. This procedure removes the influence of the deleted samples from all local models without centralized coordination. The complete process for sample-level removal is summarized in Algorithm 1.

Remark 1 (Optional fine-tuning for utility recovery)**.** Following unlearning, a lightweight fine-tuning step may be applied to further improve model utility, as adopted in prior work [16], [24]. This step follows the standard D-PSGD procedure and is performed exclusively on the retained data. Our experiments demonstrate that a *single* fine-tuning round is sufficient to match the performance of a fully retrained model. In contrast to prior approaches that rely on many post-unlearning fine-tuning rounds, e.g. up to 200 rounds in [16], our method yields significantly lower computation and communication overheads. Moreover, we prove that this step preserves the (ϵ, δ)-certified unlearning guarantee, see Section V.

Remark 2 (Multiple simultaneous deletion requests)**.** The proposed framework naturally supports multiple clients $\mathcal{C} \subseteq [N]$ issuing deletion requests simultaneously. To preserve utility guarantees, the total number of deleted samples must satisfy $\sum_{c \in \mathcal{C}} m_c \leq \zeta$, where ζ denotes the deletion capacity. The same upper and lower bounds on deletion capacity established in [17] apply to our framework.

Remark 3 (Extension to general convex losses)**.** Although our analysis assumes strong convexity, the proposed algorithm can be extended to general convex loss functions. Specifically, for a convex loss function $F_i(\cdot, \xi)$, we define the regularized loss

$$\widetilde{F}_i(x, \xi) = F_i(x, \xi) + \frac{\lambda}{2} \|x\|^2, \quad (4)$$

Algorithm 1: Certified Hessian-based Decentralized Sample-Wise Unlearning Algorithm

Input: Client $c \in [C]$ submitting deletion requests ($[C] \subseteq [N]$), Client $i \in [N]$; Delete requests: $U_c = \{\xi_{c,t}\}_{t=1}^{m_c} \subseteq S_c$; local model: $x_{k,c}$; Additional statistic $T_i(S_i) : \{\nabla^2 f_i(x_{k,i})\}$; Loss function: $F_i(x, \xi)$; Set $\Delta F_c = \frac{2ML^2 m_c^2}{\lambda^3 n_c^2}$, $\sigma_c = \frac{\Delta F_c}{\epsilon} \sqrt{2 \ln(1.25/\delta)}$

Output: $\widetilde{x}_i$ for each client i

1 **for** every client $c \in [C]$ **in parallel do**

2     $\hat{H}_c \leftarrow \frac{1}{n_c - m_c} \sum_{\xi_c \in S_c \setminus U_c} \nabla^2 F_c(x_{k,c}, \xi_c)$;

3     $x_c^\Delta \leftarrow \frac{1}{n_c - m_c} (\hat{H}_c)^{-1} \sum_{\xi_c \in U_c} \nabla F_c(x_{k,c}, \xi_c)$;

4   Sample $\nu_c \in \mathbb{R}^d$ from $\mathcal{N}(0, \sigma_c^2 \mathbb{I}_d)$;

5     $\widetilde{x}_c^\Delta \leftarrow x_c^\Delta + \nu_c$;

6   Send $\widetilde{x}_c^\Delta$ to neighbors;

7 **for** every client $i \in [N]$ **in parallel do**

8     $\widetilde{x}_i \leftarrow x_{k,i}$;

9   **if** *receiving a message $\widetilde{x}_c^\Delta$ for the first time* **then**

10   Forward the message to all other neighbors;

11       $\widetilde{x}_i \leftarrow \widetilde{x}_i + \frac{1}{N} \widetilde{x}_c^\Delta$;

12   **if** *receiving a message again* **then**

13   Discard the message;

14 **return** $\{\widetilde{x}_i\}_{i=1}^N$

---

which is λ-strongly convex. Applying our decentralized unlearning algorithm to the regularized objective yields the same form of updates and certified unlearning guarantees. This standard regularization technique allows our framework to handle general convex objectives without modifying the algorithmic structure.

C. Decentralized Unlearning for Client-Level Removal

We next extend our framework to *client-wise certified decentralized unlearning*, which aims to completely remove the contribution of a departing client from the trained models. Unlike sample-wise unlearning, client-wise unlearning requires eliminating the influence of *all* data held by the target client. Consider a client $c \in [N]$ that requests full removal at iteration k. Following the same influence-function-based philosophy as in the sample-wise setting, we approximate the effect of removing client c's entire dataset using a Hessian-based Newton-style correction. To compute the corrective update, client c first collects the auxiliary statistics $T_i(S_i) = \{\nabla^2 f_i(x_{k,i})\}$ from all remaining clients. Let $f_i(x) = \frac{1}{n_i} \sum_{t=1}^{n_i} F_i(x, \xi_{i,t})$ be the local empirical risk. Using these statistics, the aggregated Hessian over the retained clients is computed as

$$\hat{H}_c = \frac{1}{N-1} \left(\sum_{i \in [N]} \nabla^2 f_i(x_{k,i}) - \nabla^2 f_c(x_{k,c}) \right), \quad (5)$$

which approximates the curvature of the global objective after removing client c. Based on the aggregated curvature, the corrective update is given by

$$x_c^\Delta = \frac{1}{N-1}(\hat{H}_c)^{-1}\nabla f_c(x_{k,c}). \qquad (6)$$

To satisfy certified unlearning guarantees, the corrective update is perturbed with calibrated Gaussian noise and broadcast through the network. All remaining clients aggregate the received update using uniform weights and apply the correction to their local models, following the same procedure described in Algorithm 1.

D. Efficient Hessian Approximation

Our unlearning algorithm relies on second-order information through Hessian matrices. Suppose the model dimension is d. Storing a full Hessian requires $\mathcal{O}(d^2)$ memory, which can still be prohibitive for resource-constrained clients. To further reduce this overhead, we adopt an efficient approximation based on the Fisher Information Matrix (FIM). By leveraging the Fisher approximation, the storage consumption can be reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, yielding significant memory savings. In practical settings, the empirical Fisher matrix provides an effective approximation of the Hessian and has been shown to capture its curvature structure even with relatively small sample sizes [30]. Specifically, the empirical Fisher approximation Ψ_c at client c and iteration k is defined as

$$\Psi_c = \frac{1}{n_c - m_c} \sum_{\xi_c \in S_c \setminus U_c} \nabla F_c(x_{k,c}, \xi_c) \nabla F_c(x_{k,c}, \xi_c)^T. \quad (7)$$

We note that the Fisher approximation is well defined when the loss function corresponds to a negative log-likelihood, i.e., $F_i(x,\xi) \propto -\log \mathbb{P}(\xi \mid x)$. This condition is satisfied by commonly used losses such as squared loss and cross-entropy loss, making the approximation broadly applicable.

Using the Fisher approximation amounts to replacing the empirical Hessian $\hat{H}_c$ with Ψ_c in the unlearning update. Since Ψ_c is only an approximation of $\hat{H}_c$, certified unlearning guarantees require that the two matrices coincide in certain regimes. Let $x_c^* = \arg\min_{x \in \mathcal{X}} f_c(x)$ denote the empirical risk minimizer at client c. Prior work [31] shows that $\Psi_c = \hat{H}_c$ at x_c^*. Under this condition, all theoretical guarantees established for Algorithm 1 directly extend to the Fisher-based variant without loss of generality.

E. Comparisons to previous work PDUDT [16]

We now compare our framework with PDUDT [16], a recent provable approach for decentralized approximate unlearning. **1. Scope and generality.** PDUDT focuses exclusively on client-wise unlearning, aiming to remove the influence of an entire client. In contrast, our framework is more general and supports all standard unlearning granularities, including sample-wise, class-wise, and client-wise unlearning, within a unified framework. **2. Utility guarantees.** Our method provides rigorous utility guarantees: we establish explicit bounds on the test error

between the unlearned model and a fully retrained model. These bounds match state-of-the-art results for certified unlearning [17]. In contrast, the utility degradation induced by PDUDT is difficult to characterize analytically, and no comparable error bounds are provided. **3. Computational and storage efficiency.** Efficiency is another key distinction. PDUDT requires an extensive post-unlearning retraining phase consisting of T additional training rounds (e.g., $T = 200$ in their experiments), whereas our method requires at most a single fine-tuning round. We empirically compare the time required to recover model accuracy after unlearning in Section VI, where our approach demonstrates substantial efficiency gains. In terms of storage, PDUDT requires $\mathcal{O}(kN_cd)$ memory, where k is the number of training iterations and N_c is the maximum number of neighbors of client c [16]. This storage cost grows linearly with both the number of iterations and the network degree. In sharp contrast, our approach requires only $\mathcal{O}(d)$ memory when using the Fisher approximation. This dramatic reduction in space complexity makes our framework significantly better suited for resource-constrained decentralized environments.

V. THEORETICAL RESULTS

In this section, we establish the theoretical guarantees of our proposed framework. Specifically, we prove that our method satisfies the formal (ϵ, δ)-certified unlearning definition and derive utility bounds showing that the unlearned model remains close to a model retrained from scratch. Due to space constraints, we omit the detailed analysis of client-wise unlearning, which follows directly as a corollary of the sample-wise results. For clarity of exposition, we assume that all clients possess the same amount of local data, i.e., $n_1 = n_2 = \cdots = n_N = n$, and that each client issues an unlearning request of equal size m when the optimization problem (1) has been minimized after K iterations. Unless otherwise specified, $\|\cdot\|$ denotes the ℓ_2 norm.

A. Strongly Convex Loss

We first consider the setting where the loss functions satisfy Assumption 2.

Before proving that Algorithm 1 satisfies (ϵ, δ)-certified unlearning, we first present a series of lemmas that establish sensitivity upper bounds.

Lemma 2. For any client $i \in [N]$, let $x_{K,i}$ denote the local model obtained after K training iterations using the full dataset, and $x'_{K,i}$ the model obtained by retraining the learning algorithm after removing the deleted samples. Then,

$$\|x'_{K,i} - x_{K,i}\| \leq \frac{2Lm}{\lambda n}. \qquad (8)$$

Proof. Recall $f(x) = \frac{1}{Nn}\sum_{j=1}^{N}\sum_{l=1}^{n} F_j(x, \xi_{j,l})$. We define $\tilde{f}^{S-U}(x) = \frac{1}{N(n-m)}\sum_{j=1}^{N}\sum_{l \in S_j \setminus U_j} F_j(x, \xi_{j,l})$, where

$|U_j| = m$ for $\forall j \in [N]$. First we observe that

$$Nn(f(x'_{K,i}) - f(x_{K,i}))$$
$$= N(n-m)(\tilde{f}^{S-U}(x'_{K,i}) - \tilde{f}^{S-U}(x_{K,i}))$$
$$+ \sum_{j=1}^{N} \sum_{l \in U_j} (F_j(x'_{K,i}, \xi_{j,l}) - F_j(x_{K,i}, \xi_{j,l}))$$
$$\leq NmL\|x'_{K,i} - x_{K,i}\|.$$

Then by the strong convexity of $f(x)$, we have

$$f(x'_{K,i}) - f(x_{K,i}) \geq \frac{\lambda}{2}\|x'_{K,i} - x_{K,i}\|^2.$$

Thus we have

$$\frac{\lambda Nn}{2}\|x'_{K,i} - x_{K,i}\|^2 \leq NmL\|x'_{K,i} - x_{K,i}\|,$$

which implies that $\|x'_{K,i} - x_{K,i}\| \leq \frac{2mL}{\lambda n}$. $\qquad\square$

Lemma 3. Let $\hat{x}_i$ denote the unlearned model obtained *without* noise perturbation, then:

$$\|\hat{x}_i - x'_{K,i}\| \leq \frac{2ML^2 m^2}{\lambda^3 n^2}. \tag{9}$$

Proof. Take Taylor's expansion for $\nabla \tilde{f}^{S-U}(x'_{K,i})$ around the point $x_{K,i}$, we have $\|\nabla \tilde{f}^{S-U}(x'_{K,i}) - \nabla \tilde{f}^{S-U}(x_{K,i}) - \nabla^2 \tilde{f}^{S-U}(x_{K,i})[x'_{K,i} - x_{K,i}]\| \leq \frac{M}{2}\|x'_{K,i} - x_{K,i}\|^2$. Since $\nabla \tilde{f}^{S-U}(x'_{K,i}) = 0$, we further have

$$\|\nabla \tilde{f}^{S-U}(x_{K,i}) + \nabla^2 \tilde{f}^{S-U}(x_{K,i})[x'_{K,i} - x_{K,i}]\|$$
$$\leq \frac{M}{2}\|x'_{K,i} - x_{K,i}\|^2.$$

Note that

$$\nabla \tilde{f}^{S-U}(x_{K,i})$$
$$= \frac{1}{N(n-m)} \sum_{j=1}^{N} \sum_{l \in S_j \setminus U_j} \nabla F_j(x_{K,i}, \xi_{j,l})$$
$$= \frac{n}{n-m}\nabla f(x_{K,i}) - \frac{1}{N(n-m)} \sum_{j=1}^{N} \sum_{l \in U_j} \nabla F_j(x_{K,i}, \xi_{j,l})$$
$$= -\frac{1}{N(n-m)} \sum_{j=1}^{N} \sum_{l \in U_j} \nabla F_j(x_{K,i}, \xi_{j,l}).$$

Hence, we have

$$\left\| \nabla^2 \tilde{f}^{S-U}(x_{K,i})[x'_{K,i} - x_{K,i}] - \frac{\sum_{j=1}^{N} \sum_{l \in U_j} \nabla F_j(x_{K,i}, \xi_{j,l})}{N(n-m)} \right\|$$
$$\leq \frac{M}{2}\|x'_{K,i} - x_{K,i}\|^2.$$

Define a vector v such that $x'_{K,i} = x_{K,i} + \frac{1}{N(n-m)}[\nabla^2 \tilde{f}^{S-U}(x_{K,i})]^{-1} \sum_{j=1}^{N} \sum_{l \in U_j} \nabla F_j(x_{K,i}, \xi_{j,l}) + v$ and plug it in the above, we have

$$\left\| \nabla^2 \tilde{f}^{S-U}(x_{K,i})v \right\| \leq \frac{M}{2}\|x'_{K,i} - x_{K,i}\|^2.$$

Since $\tilde{f}^{S-U}(\cdot)$ is λ-strongly convex, we have that $\left\| \nabla^2 \tilde{f}^{S-U}(x_{K,i})v \right\| \geq \lambda\|v\|$ for any vector v. Then we get $\|v\| \leq \frac{M}{2\lambda}\|x'_{K,i} - x_{K,i}\|^2$. Finally, applying Lemma 2, we get that $\|v\| \leq \frac{2ML^2 m^2}{\lambda^3 n^2}$. $\qquad\square$

Lemma 3 quantifies the deviation between the unlearned model prior to noise perturbation and the model retrained on the dataset with the deleted samples removed. Notably, the bound matches the state-of-the-art results in [17] and scales as $\mathcal{O}(m^2/n^2)$. This scaling implies that adding Gaussian noise with variance $\sigma_i^2 = \Theta(m^2/n^2)$ suffices to achieve (ϵ, δ)-certified unlearning, as formalized below.

Theorem 1. The outputs of the learning algorithm $\mathcal{A}$ and the unlearning algorithm $\mathcal{G}$ satisfy (ϵ, δ)-certified unlearning.

Proof. Let $x_{K,i}$ denote the local model of client i at iteration k when trained on the full dataset, and let $\tilde{x}_i$ denote the output of Algorithm 1 without fine-tuning. Let $\hat{x}_i$ be the corresponding unlearned model before noise perturbation.

Similarly, let $x'_{K,i}$ denote the local model obtained at iteration k when training on the retained dataset, and let $\tilde{x}'_i$ denote the output of Algorithm 1 when no deletion request is issued. Let $\hat{x}'_i$ be the noise-free counterpart of $\tilde{x}'_i$. We have $\hat{x}'_i = x'_{K,i}$.

Based on (9), we define

$$||\hat{x}_i - x'_{K,i}|| \leq \frac{2ML^2 m^2}{\lambda^3 n^2} := \Delta F_i. \tag{10}$$

Also, we note that the points $\tilde{x}_i$ and $\tilde{x}'_i$ are computed with the noise $\nu_i \sim \mathcal{N}(0, \sigma_i^2 \mathbb{I}_d)$ where $\sigma_i = (\Delta F_i/\varepsilon) \cdot \sqrt{2\ln(1.25/\delta)}$. Specifically, we have $\tilde{x}_i = \hat{x}_i + \frac{1}{N}\sum_{i \in [N]} \nu_i$ and $\tilde{x}'_i = \hat{x}'_i + \frac{1}{N}\sum_{i \in [N]} \nu'_i$. Then, following along the lines of the proof of the differential privacy guarantee for the Gaussian mechanism [32], we get that for any set W,

$$\mathbb{P}(\tilde{x}_i \in W) \leq e^\varepsilon \mathbb{P}(\tilde{x}'_i \in W) + \delta, \text{ and}$$
$$\mathbb{P}(\tilde{x}'_i \in W) \leq e^\varepsilon \mathbb{P}(\tilde{x}_i \in W) + \delta, \tag{11}$$

giving the desired unlearning guarantee.

Moreover, any following fine-tuning process does not hurt the (ϵ, δ)-unlearning guarantee. This property is similar to *immunity to post − processing* in differential privacy [32]. For our (ϵ, δ)-unlearning algorithm $\mathcal{G}$, let $g : \mathcal{X} \to \mathcal{X}$ be a deterministic function. Fix any pair of neighboring databases a, b with $\|a - b\|_1 \leq 1$, and fix any event $\mathcal{W} \subseteq \mathcal{X}$. Let $\mathcal{T} = \{r \in \mathcal{X} : g(r) \in \mathcal{W}\}$. We then have:

$$\mathbb{P}[g(\mathcal{G}(a)) \in \mathcal{W}] = \mathbb{P}[\mathcal{G}(a) \in \mathcal{T}]$$
$$\leq e^\epsilon \mathbb{P}[\mathcal{G}(b) \in \mathcal{T}] + \delta = e^\epsilon \mathbb{P}[g(\mathcal{G}(b)) \in \mathcal{W}] + \delta, \tag{12}$$

which is what we want. $\qquad\square$

Theorem 2. For any set $U_i \subseteq S_i$ of m delete requests, the average unlearned model $\widetilde{x} := \frac{1}{N} \sum_{i=1}^{N} \widetilde{x}_i$ satisfies:

$$\mathbb{E}\left[f(\widetilde{x}) - \min_{x \in \mathcal{X}} f(x) \right]$$
$$\leq O\left(\frac{mL^2}{(n-m)\lambda} + \frac{\sqrt{d\ln(1/\delta)}ML^3m^2}{\lambda^3 n^2 \epsilon} \right). \quad (13)$$

Proof. We define $x^* \in \arg\min_{x \in \mathcal{X}} f(x)$. Then we have

$$\mathbb{E}[f(\widetilde{x}) - f(x^*)]$$
$$= \mathbb{E}[f(\widetilde{x}) - f(\bar{x})]$$
$$= \mathbb{E}\left[f\left(\frac{1}{N} \sum_{i=1}^{N} \widetilde{x}_i \right) - f\left(\frac{1}{N} \sum_{i=1}^{N} x_{K,i} \right) \right]$$
$$\leq L \cdot \mathbb{E}\left\| \frac{1}{N} \sum_{i=1}^{N} (\widetilde{x}_i - x_{K,i}) \right\|$$
$$\leq \frac{L}{N} \sum_{i=1}^{N} \mathbb{E}\|\widetilde{x}_i - x_{K,i}\|. \quad (14)$$

Also, we have that

$$\mathbb{E}[\|\widetilde{x}_i - x_{K,i}\|]$$
$$= \mathbb{E}\| \frac{1}{N} \sum_{i=1}^{N} (\frac{1}{n-m}(\hat{H}_i)^{-1} \sum_{\xi_i \in U_i} \nabla F_i(x_{K,i}, \xi_i) + \nu_i)\|$$
$$\leq \frac{1}{N} \sum_{i=1}^{N} \left(\mathbb{E}\| \frac{1}{n-m}(\hat{H}_i)^{-1} \sum_{\xi_i \in U_i} \nabla F_i(x_{K,i}, \xi_i)\| + \mathbb{E}\|\nu_i\| \right)$$
$$\leq \frac{1}{N(n-m)\lambda} \sum_{i=1}^{N} \sum_{\xi_i \in U_i} \mathbb{E}\|\nabla F_i(x_{K,i}, \xi_i)\| + \frac{1}{N} \sum_{i=1}^{N} \sqrt{\mathbb{E}\|\nu_i\|^2}$$
$$\leq \frac{mL}{(n-m)\lambda} + \frac{\sqrt{d\ln(1/\delta)}2ML^2m^2}{\lambda^3 n^2 \epsilon}. \quad (15)$$

where the second-to-the-last inequality holds by the L-Lipschitzness of F_i and an application of Jensen inequality to bound $\mathbb{E}[\|\nu_i\|]$. By substituting (15) into (14), we complete the proof. □

B. General Convex Loss

In this subsection, we further consider the convex loss settings. The following lemma gives us a bound on the domain of interest, and thus allows us to bound the Lipschitz constant for loss function over this domain. Specifically, we can conclude that the regularized loss function $\widetilde{F}_i(x_i, \xi)$ is λ-strongly convex, $(\mu + \lambda)$-smooth, $(L + \lambda\|x_i\|)$-Lipschitz, and M-Hessian Lipschitz by Lemma 4, which makes it easy to extend our unlearning algorithm to general convex loss.

Assumption 3. For any $\xi_i \in S_i$, all functions $F_i(x, \xi)$ are convex, μ-smooth, L-Lipschitz and M-Hessian Lipschitz.

Our algorithms for convex losses are based on the strongly convex setting. Given the convex loss function $F_i(\cdot, \xi)$, we define the function $\widetilde{F}_i(\cdot, \xi)$ as $\widetilde{F}_i(x_i, \xi) = F_i(x_i, \xi) + \frac{\lambda}{2}\|x_i\|^2$.

Lemma 4. Let $\widetilde{F}_i$ be defined as $\widetilde{F}_i(x_i, \xi) = F_i(x_i, \xi) + \frac{\lambda}{2}\|x_i\|^2$. Given a dataset S_i, let $\widetilde{x}_i^*$ denote the empirical risk minimizer of $\widetilde{F}_i$ on S_i. Then, the point $\widetilde{x}_i^*$ satisfies $\|\widetilde{x}_i^*\| \leq \frac{L}{\lambda}$.

Proof. By plugging in the definition of the function $\widetilde{F}_i$, we have $\frac{1}{n_i} \sum_{\xi_i \in S_i} \nabla \widetilde{F}_i(\widetilde{x}_i^*, \xi_i) = \frac{1}{n_i} \sum_{\xi_i \in S_i} \nabla F_i(\widetilde{x}_i^*, \xi_i) + \lambda x_i^* = 0$. From an application of the Triangle inequality and the L-Lipschitzness of F_i, we can easily get that $\|\lambda \widetilde{x}_i^*\| \leq \frac{1}{n_i} \sum_{\xi_i \in S_i} \|\nabla F_i(\widetilde{x}_i^*, \xi_i)\| \leq L$. Thus we have $\|\widetilde{x}_i^*\| \leq \frac{L}{\lambda}$. Finally, we can bound the Lipschitz constant for loss function over this domain. □

VI. EXPERIMENTS

A. Experimental Setup

We report experiments on several widely-used benchmark datasets, including CIFAR-10 [33], MNIST [34] and Fashion-MNIST [35]. We use a logistic regression model for MNIST dataset, considering a convex classification problem. For CIFAR-10 datasets, we use the ResNet-18 model [36]. For Fashion-MNIST dataset, we use the CNN model. To show the advantages of our proposed algorithm, we compare it with two existing baselines: **RT**, which means retraining from scratch with the retained data or clients by D-PSGD [3] and PDUDT [16]. For all unlearning types, we employ **RT** as the primary baseline. Since PDUDT is designed specifically for client-wise unlearning, we use it exclusively in the client-wise scenario [16]. Given that our work focuses on decentralized approximate unlearning, methods designed for either centralized or exact unlearning fall outside the scope of our comparative analysis [13]–[15].

Throughout our experiments, we consider a decentralized system with 10 clients, where the connections between clients are generated by the topologies including Ring graph and Erdős–Rényi random graph. Each client trains with a batch size of 100 for 1 epoch per round, with a learning rate of 0.001. The unlearning request is set to occur at round K, which is adjusted for each dataset based on its convergence rate. We also set the number of retraining rounds to be equal to K. In a non-IID setting, data across clients follow a Dirichlet distribution with $\alpha = 0.3$.

The unlearning tasks are structured as follows. For sample-wise unlearning in Table I-II, we randomly select 10% of each client's data to be forgotten. For sample-wise unlearning in Fig. 1, each client issues three sequential unlearning requests, cumulatively removing 20% of its local data. For class-wise unlearning, we remove samples of class 0 for all clients. For client-wise unlearning, we randomly remove a single client. For brevity, we use **ER** for the Erdős–Rényi graph, **DU** for our proposed unlearning algorithm, **Acc** for accuracy and **F-MNIST** for Fashion-MNIST.

B. Experimental Results

1) Unlearning Performance: We measure and report the best test accuracy across a comprehensive set of scenarios. As shown in Tables I-IV, our unlearning algorithm consistently demonstrates superior performance in all settings including

(a) CIFAR-10   (b) MNIST   (c) F-MNIST

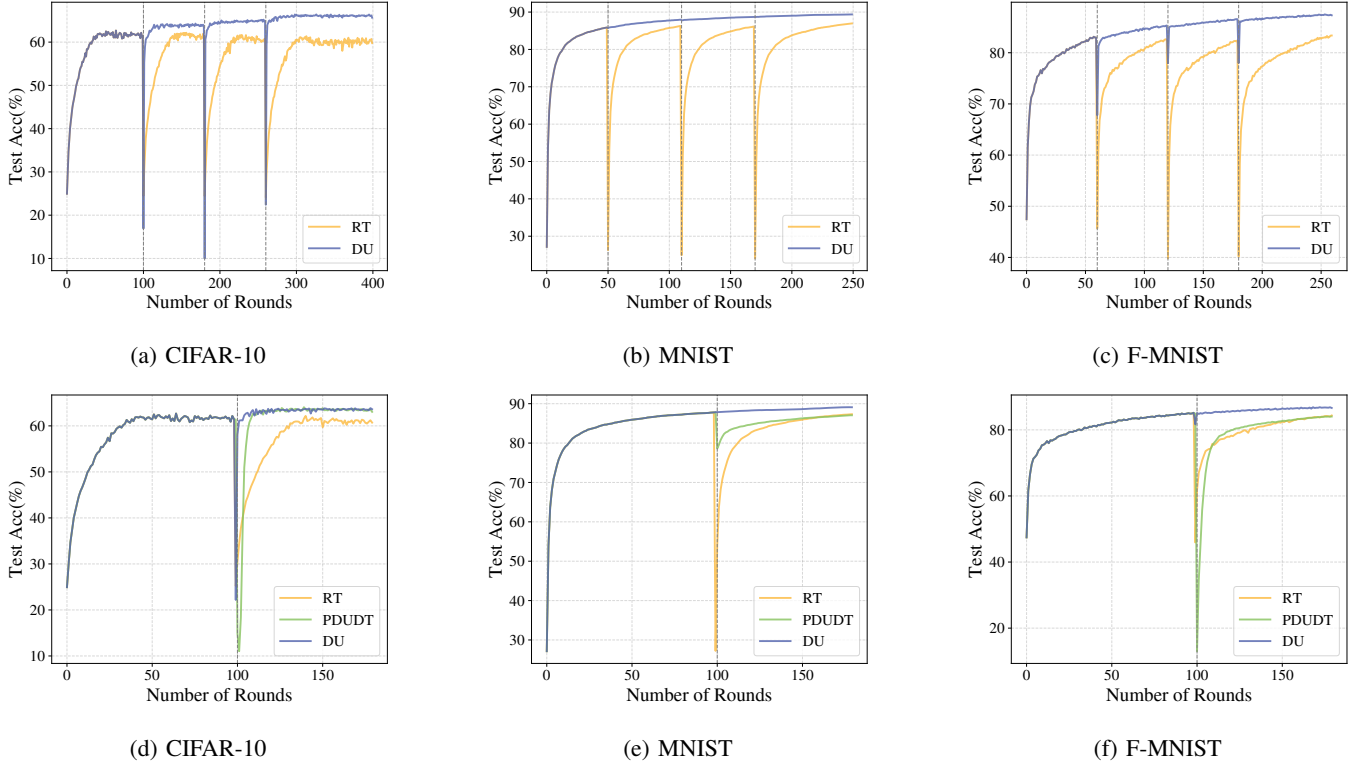(d) CIFAR-10   (e) MNIST   (f) F-MNIST

Fig. 1: Comparison of the test accuracy of different methods and their changes after different unlearning requests. (a)-(c): sample-wise unlearning. (d)-(f): client-wise unlearning.
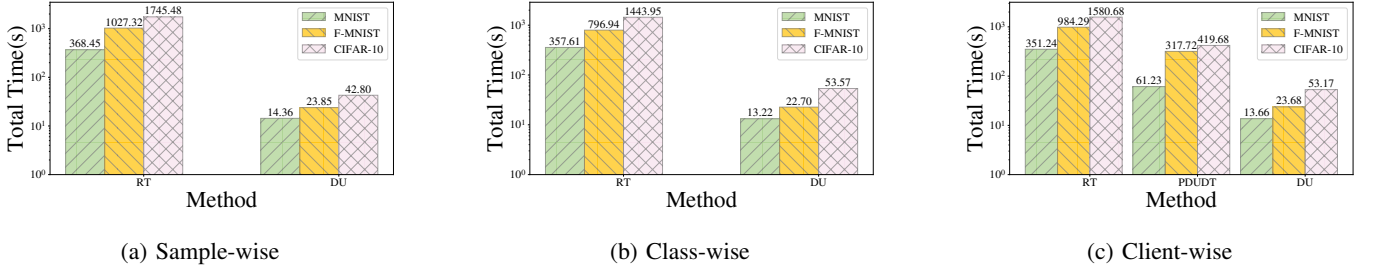

(a) Sample-wise   (b) Class-wise   (c) Client-wise

Fig. 2: Comparison of the computation time of different methods.

TABLE I

SAMPLE-WISE UNLEARNING IN AN IID SETTING

Topology	Dataset	Method	*Test Acc*	*MIA Acc*
RING	CIFAR-10	RT	**58.99**$_{\pm 0.40\%}$	50.25$_{\pm 0.98\%}$
		DU	55.28$_{\pm 0.79\%}$	50.72$_{\pm 1.48\%}$
	MNIST	RT	86.34$_{\pm 0.23\%}$	52.29$_{\pm 0.27\%}$
		DU	**86.63**$_{\pm 0.04\%}$	**51.96**$_{\pm 0.53\%}$
	F-MNIST	RT	84.24$_{\pm 0.16\%}$	49.53$_{\pm 0.19\%}$
		DU	**84.40**$_{\pm 0.23\%}$	**50.00**$_{\pm 0.58\%}$
ER	CIFAR-10	RT	**62.20**$_{\pm 0.46\%}$	50.48$_{\pm 0.67\%}$
		DU	57.82$_{\pm 0.23\%}$	51.47$_{\pm 0.70\%}$
	MNIST	RT	86.25$_{\pm 0.10\%}$	51.60$_{\pm 0.56\%}$
		DU	**86.63**$_{\pm 0.04\%}$	**51.51**$_{\pm 0.42\%}$
	F-MNIST	RT	84.21$_{\pm 0.15\%}$	50.53$_{\pm 0.72\%}$
		DU	**84.42**$_{\pm 0.20\%}$	**50.42**$_{\pm 0.83\%}$

TABLE II

SAMPLE-WISE UNLEARNING IN A NON-IID SETTING

Topology	Dataset	Method	*Test Acc*	*MIA Acc*
RING	CIFAR-10	RT	**51.68**$_{\pm 2.23\%}$	50.40$_{\pm 0.37\%}$
		DU	45.06$_{\pm 3.74\%}$	**50.20**$_{\pm 1.29\%}$
	MNIST	RT	86.59$_{\pm 0.13\%}$	50.27$_{\pm 0.66\%}$
		DU	**86.95**$_{\pm 0.15\%}$	50.69$_{\pm 0.17\%}$
	F-MNIST	RT	**82.41**$_{\pm 1.12\%}$	**50.00**$_{\pm 0.81\%}$
		DU	81.14$_{\pm 1.62\%}$	49.98$_{\pm 0.17\%}$
ER	CIFAR-10	RT	**54.95**$_{\pm 1.34\%}$	**49.70**$_{\pm 0.75\%}$
		DU	47.53$_{\pm 3.40\%}$	50.77$_{\pm 0.70\%}$
	MNIST	RT	84.16$_{\pm 4.22\%}$	52.56$_{\pm 1.55\%}$
		DU	**87.21**$_{\pm 0.33\%}$	52.79$_{\pm 1.75\%}$
	F-MNIST	RT	**82.35**$_{\pm 1.22\%}$	49.50$_{\pm 0.66\%}$
		DU	81.20$_{\pm 1.40\%}$	**49.78**$_{\pm 0.88\%}$

TABLE III
CLASS-WISE UNLEARNING IN AN IID SETTING

Topology	Dataset	Method	Test Acc	MIA Acc
RING	CIFAR-10	RT	**54.67**$_{\pm 0.50\%}$	**51.08**$_{\pm 3.27\%}$
		DU	53.35$_{\pm 1.07\%}$	53.50$_{\pm 2.35\%}$
	MNIST	RT	77.71$_{\pm 0.04\%}$	53.66$_{\pm 2.25\%}$
		DU	**84.01**$_{\pm 0.12\%}$	**52.55**$_{\pm 0.36\%}$
	F-MNIST	RT	77.88$_{\pm 0.13\%}$	**49.58**$_{\pm 4.23\%}$
		DU	**78.02**$_{\pm 0.14\%}$	49.42$_{\pm 3.83\%}$
ER	CIFAR-10	RT	**57.77**$_{\pm 0.64\%}$	**49.75**$_{\pm 3.02\%}$
		DU	55.20$_{\pm 1.02\%}$	50.40$_{\pm 0.44\%}$
	MNIST	RT	77.71$_{\pm 0.04\%}$	53.66$_{\pm 2.25\%}$
		DU	**84.00**$_{\pm 0.14\%}$	**52.55**$_{\pm 0.42\%}$
	F-MNIST	RT	77.86$_{\pm 0.13\%}$	49.42$_{\pm 3.80\%}$
		DU	**78.04**$_{\pm 0.15\%}$	49.33$_{\pm 3.95\%}$

TABLE IV
CLASS-WISE UNLEARNING IN A NON-IID SETTING

Topology	Dataset	Method	Test Acc	MIA Acc
RING	CIFAR-10	RT	**48.07**$_{\pm 2.84\%}$	51.83$_{\pm 2.18\%}$
		DU	43.15$_{\pm 2.72\%}$	**51.25**$_{\pm 1.87\%}$
	MNIST	RT	78.10$_{\pm 0.14\%}$	**54.08**$_{\pm 1.10\%}$
		DU	**86.31**$_{\pm 0.43\%}$	54.34$_{\pm 0.36\%}$
	F-MNIST	RT	**76.26**$_{\pm 1.09\%}$	**50.42**$_{\pm 0.92\%}$
		DU	76.22$_{\pm 0.90\%}$	50.92$_{\pm 2.29\%}$
ER	CIFAR-10	RT	**50.92**$_{\pm 1.54\%}$	**49.67**$_{\pm 0.96\%}$
		DU	46.16$_{\pm 2.99\%}$	54.58$_{\pm 0.42\%}$
	MNIST	RT	81.34$_{\pm 3.83\%}$	53.59$_{\pm 1.89\%}$
		DU	**86.79**$_{\pm 0.15\%}$	**53.02**$_{\pm 1.54\%}$
	F-MNIST	RT	76.00$_{\pm 1.16\%}$	51.25$_{\pm 0.00\%}$
		DU	**76.16**$_{\pm 0.88\%}$	**51.00**$_{\pm 2.47\%}$

different data distributions and topologies. It successfully preserves model utility, delivering performance comparable to fully retraining models with substantial time savings.

Furthermore, we conduct additional experiments for IID setting with Erdős–Rényi random graph. We primarily focus on sample-wise and client-wise unlearning, with the corresponding results shown in Fig. 1. We measure and present the model test accuracy throughout the whole process including the initial training, issuing unlearning requests, and the recovery from data deletion using different methods. Our algorithm's consistent performance across the sequential unlearning requests serves as a strong testament to its stability. It should be noted that, to maintain the consistency of the x-axis, the **DU** method performed multiple rounds of fine-tuning in the experiments for Fig. 1, whereas only a single round was executed in all other experiments. In fact, our algorithm requires just one round of fine-tuning to recover accuracy. Therefore, the curve of **DU** exhibits a sharp rise after each deletion request, while **RT** and PDUDT require multiple rounds to regain comparable performance. Last, on simpler datasets such as MNIST, our algorithm exhibits negligible performance degradation after unlearning requests. This is because the corrective model updates alone are sufficient to maintain model accuracy, which further demonstrates the effectiveness of our proposed method.

2) Membership Inference Attack: We employ membership inference attacks (MIA) to assess the privacy leakage risk of the unlearning process. Specifically, MIA aims to infer whether a given data sample was part of the model's training set. It quantifies the extend to which the forgotten data remains identifiable within the unlearned model. Consequently, an MIA accuracy approaching 50% (which is equivalent to a random guess) signifies a more effective and secure unlearning process. As reported in Tables I-IV, the MIA accuracy of our method consistently approaches the random guess. This indicates that an attacker cannot reliably infer training set membership, thus confirming the high efficacy of our algorithm. Additionally, the overall performance of **DU** remains on par with **RT** method.

3) Unlearning Efficiency: To evaluate the efficiency of each method, we measure and present the time elapsed from the issuance of an unlearning request until the model's performance stabilizes in IID setting with Erdős–Rényi random graph. To ensure a fair and effective comparison, the recorded time for **DU** encompasses the entire unlearning and fine-tuning process. The experimental results in Fig. 2 demonstrate that our algorithm substantially reduces the unlearning time. Compared with **RT** method, we achieve approximately 97% time savings across all datasets and unlearning types. We note that the numerical values shown in Fig. 2 represent actual time, whereas the bar chart is plotted using a logarithmic scale.

VII. CONCLUSION

In this paper, we proposed a certified decentralized unlearning framework based on Newton-style updates coupled with efficient Hessian approximations. Our approach achieves high efficiency in both computational cost and storage overhead. We theoretically proved that the proposed algorithm satisfies the formal (ϵ, δ)-certified unlearning guarantee and established rigorous utility bounds showing that the unlearned model remains close to a fully retrained one. Extensive experimental results further validate the effectiveness and efficiency of our framework. While our analysis focuses on convex objectives, extending certified decentralized unlearning to non-convex settings remains an important and challenging direction for future work.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Shuzhen Chen, Yuan Yuan, Youming Tao, Tianzhu Wang, Zhipeng Cai, and Dongxiao Yu. Adaptive pruning-based newton's method for distributed learning. *Theoretical Computer Science*, 1026:114987, 2025.

[3] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in neural information processing systems*, 30, 2017.

[4] Youming Tao, Shuzhen Chen, Feng Li, Dongxiao Yu, Jiguo Yu, and Hao Sheng. A distributed privacy-preserving learning dynamics in general social networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9547–9561, 2023.

[5] Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S Yu, and Christopher G Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638, 2024.

[6] Dongxiao Yu, Zongrui Zou, Shuzhen Chen, Youming Tao, Bing Tian, Weifeng Lv, and Xiuzhen Cheng. Decentralized parallel sgd with privacy preservation in vehicular networks. *IEEE Transactions on Vehicular Technology*, 70(6):5211–5220, 2021.

[7] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[8] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[9] Federal Trade Commission et al. California company settles ftc allegations it deceived consumers about use of facial recognition in photo storage app, 2021.

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.

[11] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys*, 57(1):1–38, 2024.

[12] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, pages 1–10. IEEE, 2021.

[13] Lefeng Zhang, Tianqing Zhu, Haibin Zhang, Ping Xiong, and Wanlei Zhou. Fedrecovery: Differentially private machine unlearning for federated learning frameworks. *IEEE Transactions on Information Forensics and Security*, 18:4732–4746, 2023.

[14] Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. Communication efficient and provable federated unlearning. *Proceedings of the VLDB Endowment*, 17(5):1119–1131, 2024.

[15] Guanhua Ye, Tong Chen, Quoc Viet Hung Nguyen, and Hongzhi Yin. Heterogeneous decentralised machine unlearning with seed model distillation. *CAAI Transactions on Intelligence Technology*, 9(3):608–619, 2024.

[16] Jing Qiao, Yu Liu, Zengzhe Chen, Mingyi Li, Yuan Yuan, Xiao Zhang, and Dongxiao Yu. Pdudt: Provable decentralized unlearning under dynamic topologies. In *Forty-second International Conference on Machine Learning*.

[17] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

[18] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3832–3842, 2020.

[19] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE, 2015.

[20] Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

[21] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE conference on computer communications*, pages 1749–1758. IEEE, 2022.

[22] Xinbao Qiao, Meng Zhang, Ming Tang, and Ermin Wei. Hessian-free online certified unlearning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Representation Learning*, volume 2025, pages 32675–32711, 2025.

[23] Zonglin Di, Sixie Yu, Yevgeniy Vorobeychik, and Yang Liu. Adversarial machine unlearning. *arXiv preprint arXiv:2406.07687*, 2024.

[24] Yasser H Khalil, Leo Brunswic, Soufiane Lamghari, Xu Li, Mahdi Beitollahi, and Xi Chen. Not: Federated unlearning via weight negation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 25759–25769, 2025.

[25] Zhengyi Zhong, Weidong Bao, Ji Wang, Shuai Zhang, Jingxuan Zhou, Lingjuan Lyu, and Wei Yang Bryan Lim. Unlearning through knowledge overwriting: Reversible federated unlearning via selective sparse adapter. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 30661–30670, 2025.

[26] Yian Zhao, Pengfei Wang, Heng Qi, Jianguo Huang, Zongzheng Wei, and Qiang Zhang. Federated unlearning with momentum degradation. *IEEE Internet of Things Journal*, 11(5):8860–8870, 2023.

[27] Yann Fraboni, Martin Van Waerebeke, Kevin Scaman, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Sifu: Sequential informed federated unlearning for efficient and provable client unlearning in federated optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3457–3465. PMLR, 2024.

[28] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

[29] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[30] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[31] Divyansh Jhunjhunwala, Shiqiang Wang, and Gauri Joshi. Fedfisher: Leveraging fisher information for one-shot federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1612–1620. PMLR, 2024.

[32] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

[33] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.