# Communication Efficient and Provable Federated Unlearning

Youming Tao*
Shandong University
P.R. China
ym.tao99@mail.sdu.edu.cn

Cheng-Long Wang*
KAUST
Saudi Arabia
chenglong.wang@kaust.edu.sa

Miao Pan
University of Houston
U.S.A.
mpan2@uh.edu

Dongxiao Yu†
Shandong University
P.R. China
dxyu@sdu.edu.cn

Xiuzhen Cheng
Shandong University
P.R. China
xzcheng@sdu.edu.cn

Di Wang†
KAUST
Saudi Arabia
di.wang@kaust.edu.sa

## ABSTRACT

We study federated unlearning, a novel problem to eliminate the impact of specific clients or data points on the global model learned via federated learning (FL). This problem is driven by the right to be forgotten and the privacy challenges in FL. We introduce a new framework for exact federated unlearning that meets two essential criteria: *communication efficiency* and *exact unlearning provability*. To our knowledge, this is the first work to tackle both aspects coherently. We start by giving a rigorous definition of *exact* federated unlearning, which guarantees that the unlearned model is statistically indistinguishable from the one trained without the deleted data. We then pinpoint the key property that enables fast exact federated unlearning: total variation (TV) stability, which measures the sensitivity of the model parameters to slight changes in the dataset. Leveraging this insight, we develop a TV-stable FL algorithm called FATS, which modifies the classical FedAvg algorithm for TV Stability and employs local SGD with periodic averaging to lower the communication round. We also design efficient unlearning algorithms for FATS under two settings: client-level and sample-level unlearning. We provide theoretical guarantees for our learning and unlearning algorithms, proving that they achieve exact federated unlearning with reasonable convergence rates for both the original and unlearned models. We empirically validate our framework on 6 benchmark datasets, and show its superiority over state-of-the-art methods in terms of accuracy, communication cost, computation cost, and unlearning efficacy.

**PVLDB Reference Format:**
Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. Communication Efficient and Provable Federated Unlearning. PVLDB, 17(5): 1119 - 1131, 2024.
doi:10.14778/3641204.3641220

*Youming Tao and Cheng-Long Wang contributed equally to this work. Part of the work was done when Youming Tao was a research intern at KAUST
†Corresponding author: Dongxiao Yu, Di Wang.

Proceedings of the VLDB Endowment, Vol. 17, No. 5 ISSN 2150-8097.
doi:10.14778/3641204.3641220

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/Happy2Git/FATS_supplement.

## 1 INTRODUCTION

The proliferation of personal data collection and processing by various entities poses a serious threat to the public's data privacy. To reconcile users' data privacy and the need for data analytics in intelligent applications, Federated Learning (FL) [18] has emerged as a promising paradigm for collaborative machine learning at edge. FL allows multiple edge devices (or clients[1]) to cooperatively learn a global model with the coordination of a central edge server. By keeping users' data locally and only exchanging model updates, FL mitigates the risk of direct privacy leakage or disclosures. However, this is not enough to address the privacy issues as adversaries can still deduce sensitive user information from the shared models. Therefore, users require the ability to erase some of their private data from the global model, which is known as *the right to be forgotten* and has been supported by several legal regulations such as GDPR [26] and CCPA [11]. This is particularly relevant in the setting of edge computing, where edge devices such as smartphones, tablets, smartwatches, or sensors generate and process massive amounts of diverse and confidential data from users. For example, some users may wish to remove their health records from a wearable device or their location history from a navigation app. However, simply deleting data from edge devices may not guarantee the right to be forgotten, as the data may have been used to train collaborative learning models that are distributed among multiple edge devices and the server. This implies that the data may still affect the model's outcomes or actions or even be recovered by malicious actors [23, 29, 33] Hence, it is essential to develop *machine unlearning* [2] methods to endow FL models with the capability to unlearn requested data, i.e., to eliminate the impact of some specific data from a trained FL model. Moreover, data removal from a trained model is also beneficial in other scenarios, such as countering data poisoning attacks and correcting data errors [20].

Despite its importance and necessity, machine unlearning is not a trivial task, even in the centralized setting where a single entity stores and processes the data and the model. It requires efficient algorithms that can update the model parameters without retraining from scratch, as well as verify the effectiveness and soundness of the unlearning process. Moreover, machine unlearning becomes

[1]In this paper, we use the term device and client interchangeably.

exceedingly formidable in the federated setting. Machine unlearning in the federated setting, known as *federated unlearning*, has to cope with several additional factors.

(I) Communication efficiency is paramount for federated unlearning, as communication is the primary bottleneck of FL [13]. Hence, unlearning methods that incur frequent communication between the server and the clients, or model retraining from scratch, are highly impractical.

(II) The availability of training data is severely limited in federated unlearning, as the data are distributed across clients and not divulged to the server. Therefore, unlearning methods that depend on accessing or reusing the unlearned data are inapplicable in the federated setting.

(III) Besides sample-level unlearning in the central setting, federated unlearning also entails client-level unlearning, which aims to eliminate the influence of specific clients from the FL model. This is because the edge environment is unstable and clients may withdraw from the system at any time.

(IV) Federated unlearning requires provable guarantees for both sample and client-level unlearning, as unlearning methods should be trustworthy and capable of certifying that they effectively and completely erase the influence of the unlearned samples or clients with a reasonable convergence guarantee for the unlearned model.

These factors make federated unlearning a more intricate and challenging problem than centralized machine unlearning. Consequently, most of the existing machine unlearning methods are not applicable in FL. Furthermore, existing works on federated unlearning are scarce and have many limitations. For example, most of them proposed heuristic methods without any theoretical unlearning guarantees [16, 30, 31], or noise perturbation-based algorithm with only an approximate unlearning guarantee [7] (see Section 2 for more details). How to enable efficient data removal from FL models with provable guarantees remains largely under-explored.

**Our Contributions.** In this paper, we take the first step to fill the gap by presenting a general framework for communication efficient and provable federated unlearning. It is noteworthy that, compared with the previous works, our goal is to achieve *exact* federated unlearning. That is, when a specific data point or client is requested to be removed, the algorithmic states are adjusted exactly to what they would have been if the data or client had never been included. Our main contributions can be summarized as follows:

(I) We provide a rigorous definition of *exact* federated unlearning, which guarantees that, for each sample or client deletion request, the unlearned model is statistically indistinguishable from the one trained without the deleted sample or client from scratch. To the best of our knowledge, this is the first formal definition of exact federated unlearning. Moreover, we show that the efficiency of exact federated unlearning depends on a key property of the FL algorithm: total variation (TV) stability (at both sample and client levels), which offers guidance for FL algorithm design.

(II) To achieve TV stable algorithm for FL, we develop FATS, which leverages local SGD with periodic averaging for communication efficiency and incorporates an elaborately designed subsampling rule. Specifically, at each communication round, only part of the clients are selected to run multiple local update steps using sub-sampled partial local data. We then devise efficient sample-level and client-level unlearning algorithms for FATS. Informally, for $M$ clients each with $N$ data samples, and a given pair of sample and client level TV-stability parameters $\rho_S, \rho_C \in (0, 1]$, our unlearning algorithms only retrain on $\rho_S$ and $\rho_C$ fraction of sample and client removal requests, respectively, while achieving both exact unlearning requirement and the reasonable convergence error of $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$ in terms of average-squared gradient norm over non-convex loss and non-i.i.d. data.

(III) Finally, we evaluate our proposed framework on 6 benchmark datasets to show its effectiveness and efficiency. We find that FATS matches the performance of the standard FL baselines in terms of test accuracy and convergence speed, and that the unlearning algorithms can drastically reduce the computation and communication costs for both sample and client unlearning compared with state-of-the-art federated unlearning methods. Experimental results confirm that our framework achieves the best possible learning performance and restores utility faster upon receiving the unlearning request, which provides practical guarantees that fully comply with the requirement of exact federated unlearning.

## 2 RELATED WORK

**Machine unlearning.** The study of machine unlearning was pioneered by [4], where exact unlearning methods were devised for statistical query learning to guarantee the equivalence of the unlearned model and the model retrained from scratch without the deleted sample. However, their methods are limited to very structured problems. A more general framework for deep learning models was proposed by [2], which introduced the Sharded, Isolated, Sliced, Aggregated (SISA) approach. Subsequently, several works also adapted the SISA framework to graph data, e.g., [5, 27]. Nevertheless, these works all lack theoretical guarantees for their approaches and are not suitable for the federated learning setting. Another line of work relaxed the exact unlearning requirement and considered approximate unlearning starting from [8], which was inspired by differential privacy [6] and allowed the distribution of the unlearning model to be only close to that of the retaining model. Various works have explored approximate unlearning algorithms for different learning objectives, such as empirical risk minimization [9, 19, 24] and population risk minimization [21]. Although they provided theoretical error bounds for their algorithms, the approximate unlearning objective implies that the deleted data was not fully removed and could still be leaked to an adversary, which makes it less desirable than exact unlearning. To bridge the theoretical gap for exact unlearning, [25] introduced a notion of algorithmic stability, Total Variation (TV) stability, which is used for achieving exact unlearning. It designed TV-stable algorithms, analyzed the trade-offs between accuracy and unlearning efficiency, and established upper and lower bounds on the excess empirical and population risks of TV-stable learning algorithms over convex loss functions. However, its notion of TV stability is not directly transferable to the federated setting, and the design of TV-stable FL algorithms remains an open challenge. Moreover, its theoretical analysis and results are based on convex optimization with i.i.d. data, while we consider federated non-convex optimization with non-i.i.d. data in this work.

**Federated unlearning.** Unlearning in the federated learning paradigm has received relatively less attention than in the centralized paradigm. FedEraser [16] reconstructs the unlearned global model approximately by leveraging the historical parameter updates of the participating clients. Although speeding up the unlearning process compared with retraining from scratch, it is still prohibitively expensive. Thus, more recent works have proposed approximate unlearning techniques based on knowledge distillation [30], class-discriminative pruning [28], projected gradient ascent [10, 31], or second-order AdaHessian optimizer [17]. However, these works still suffer from several limitations: Firstly, these approximate unlearning techniques cannot completely eliminate the influence of data samples to be removed from the trained model, which may result in severe privacy leakage on data holders' data samples, e.g., gradient leakage attacks [23]. Secondly, most of them only consider client-level unlearning. Sample-level unlearning is also crucial and distinct from the centralized paradigm. As clients will not share their private data with the server, the federated unlearning process can only be executed on the clients. Thirdly, prior works are mostly heuristic. They only validate their method empirically without any theoretical guarantees, which makes their methods unreliable in real-world applications. To overcome these challenges, in this paper, we pursue a more ambitious goal of exact federated unlearning at both sample and client levels, and also provide rigorous theoretical analysis on both unlearning guarantees and convergence error bound. Moreover, we consider the communication bottleneck in the federated scenario, which is seldom addressed in prior works but is vital in practice.

## 3 PRELIMINARIES

### 3.1 Federated Learning

We consider a typical federated learning setup where $M$ clients cooperate to train a machine learning model under the coordination of a central server. Let $\mathcal{X}$ be the data universe and $\Theta$ be the model parameter space in general. Each client $k \in [M]$ has a local dataset $\mathcal{D}_k$ comprising $N$ data points $X_k^{(1)}, X_k^{(2)}, \dots, X_k^{(N)} \in \mathcal{X}$ sampled from a client-specific local distribution $D_k$. The goal is to learn a machine learning model parameterized by $\theta \in \Theta$ that minimizes the global empirical risk function $F(\cdot)$ over the $M$ local datasets without disclosing the data to the central server due to privacy concerns. The goal can be formally described as the following empirical risk minimization (ERM) problem:

$$\min_{\theta \in \Theta} F(\theta) := \frac{1}{M} \sum_{k=1}^{M} F_k(\theta), \tag{1}$$

where $F_k(\theta)$ is the local empirical risk function for client $k$ and is defined as $F_k(\theta) := \frac{1}{N} \sum_{i=1}^{N} f(\theta; X_k^{(i)})$ with $f(\theta; X_k^{(i)})$ being the point-wise loss of model parameter $\theta$ for the data sample $X_k^{(i)}$.

### 3.2 Federated Unlearning

The goal of machine unlearning is to remove the influence of the requested data from a trained machine learning model. In the federated setting, we distinguish two types of *exact* unlearning requirements — sample-level unlearning and client-level unlearning. A pair of federated learning and unlearning procedure can be denoted as a tuple $(\mathcal{L}, \mathcal{U})$, where $\mathcal{L} : \mathcal{X}^* \to \Theta \times \mathcal{S}$ is the learning algorithm that maps a set of training data points to a trained model $\theta \in \Theta$ together with an internal algorithmic state $s \in \mathcal{S}$, and $\mathcal{U} : \Theta \times \mathcal{S} \times \mathcal{X}^* \to \Theta \times \mathcal{S}$ is the unlearning algorithm that updates the learned model $\theta \in \Theta$ and internal state $s \in \mathcal{S}$ to the unlearned model $\theta^u$ and corresponding state $s^u$, given an unlearning request (either a target data point in some client or the entire dataset of a target client[2]). We collectively refer to the output model $\theta$ and internal state $s$ as the *algorithmic state*. In FedAvg algorithm with local mini-batch SGD and client sub-sampling for example, the output model is the final global model whereas the rest of intermediate local and global models, local mini-batches, subsets of clients are the internal state. We use $C$ to denote the set of clients and $\mathcal{D}(C)$ to denote all the data points stored by the clients in $C$. We may omit the client set indicator $C$ and simply use $\mathcal{D}$ to denote the global dataset when it is clear from the context. For any two sets $\mathcal{Z}$ and $\mathcal{Z}'$, we use $\Delta(\mathcal{Z}, \mathcal{Z}')$ to denote the symmetric difference between them, *i.e.,* $\Delta(\mathcal{Z}, \mathcal{Z}') := (\mathcal{Z} \setminus \mathcal{Z}') \cup (\mathcal{Z}' \setminus \mathcal{Z})$. For any set $\mathcal{Z}$, we use $|\mathcal{Z}|$ to denote its cardinality. Now, we define the notions of both sample-level and client-level exact federated unlearning.

**Definition 1** (Sample-level Exact Federated Unlearning)**.** A pair of federated learning and unlearning algorithms $(\mathcal{L}, \mathcal{U})$ is said to satisfy sample-level exact federated unlearning if for any $\mathcal{D}(C), \mathcal{D}(C)'$ such that $\mathcal{D}(C) \supset \mathcal{D}(C)'$ and $|\Delta(\mathcal{D}(C), \mathcal{D}(C)')| = 1$, and any measurable event $\mathcal{E} \subseteq \Theta \times \mathcal{S}$, it holds that

$$\mathbb{P}(\mathcal{L}(\mathcal{D}(C)') \in \mathcal{E}) = \mathbb{P}(\mathcal{U}(\mathcal{L}(\mathcal{D}(C)), \Delta(\mathcal{D}(C), \mathcal{D}(C)')) \in \mathcal{E}).$$

**Definition 2** (Client-level Exact Federated Unlearning)**.** A pair of federated learning and unlearning algorithms $(\mathcal{L}, \mathcal{U})$ is said to satisfy client-level exact federated unlearning if for any $C, C'$ such that $C \supset C'$ and $|\Delta(C, C')| = 1$, and any measurable event $\mathcal{E} \subseteq \Theta \times \mathcal{S}$, it holds that

$$\mathbb{P}(\mathcal{L}(\mathcal{D}(C')) \in \mathcal{E}) = \mathbb{P}(\mathcal{U}(\mathcal{L}(\mathcal{D}(C)), \Delta(\mathcal{D}(C), \mathcal{D}(C'))) \in \mathcal{E}).$$

The above definitions imply that, for each target sample or target client to delete, the unlearned model is equivalent to the one that would have been obtained if trained on the updated setting (i.e., without the target sample or target client) from scratch, since they have exactly identical output distributions. This is the reason why we refer to this type of unlearning requirement as *exact* unlearning. Furthermore, although the above definitions are for a single edit request, they can be extended for a sequence of $w$ edit requests by requiring the condition to hold recursively for every request in the sequence.

### 3.3 Federated Unlearning via TV-Stability

We adopt the total variation (TV) distance as a measure of discrepancy between two distributions $P$ and $Q$:

$$\mathrm{TV}(P, Q) := \sup_{\text{measurable set } \mathcal{Z}} |P(\mathcal{Z}) - Q(\mathcal{Z})| = \frac{1}{2} \|\phi_P - \phi_Q\|_1, \tag{2}$$

where the second equality is valid when distributions $P$ and $Q$ admit probability densities (denoted as $\phi_P$ and $\phi_Q$ respectively) with respect to a base measure.

[2]In this paper, we formally refer to the data sample and client that need to be forgotten as the target sample and the target client respectively.

Let $\mathcal{D}$ and $\mathcal{D}'$ be two datasets such that $\mathcal{D}' \subseteq \mathcal{D}$ is obtained by removing a data point or a client from $\mathcal{D}$. Let $P = \mathcal{L}(\mathcal{D})$ and $Q = \mathcal{L}(\mathcal{D}')$ for some randomized FL algorithm $\mathcal{L}$. To satisfy the requirement of exact federated unlearning, we need to move from $P$ to $Q$, which is an optimal transport problem.

In a general optimal transport problem, we are given two probability distributions $P$ and $Q$ over a measurable space $\Omega$ and a cost function $c : \Omega \times \Omega \to \mathbb{R}$. The goal is to transport from $P$ to $Q$ using the minimum cost. Formally, let $\Pi(P, Q)$ denote the set of couplings (or transport plans) of $P$ and $Q$, we seek a transport plan $\pi$ which minimizes the expected cost: $\min_{\pi \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\pi} c(x, y)$.

In the context of federated unlearning, we aim to find a transport from $P = \mathcal{L}(\mathcal{D})$ to $Q = \mathcal{L}(\mathcal{D}')$ over the set of all possible algorithmic states $\Omega = \Theta \times \mathcal{S}$. Notably, we require the probability distribution of the entire algorithmic state, not just the output, to be exactly identical after unlearning. A trivial transport plan is re-computation that generates independent samples from $P$ and $Q$, but this is extremely inefficient as it requires a full re-training from scratch for every unlearning request. To improve the efficiency of unlearning, we should exploit the correlation between $P$ and $Q$ so that we can reuse the randomness (computation) involved in generating P when transforming $P$ to $Q$, hoping that no re-training or only a partial re-training from the middle of the learning process is sufficient for every unlearning request. For this purpose, we define the cost function for any two distinct algorithmic states, which captures the expected costs for any transport rule from $P$ to $Q$. A natural choice is to set the cost function as $c(x, y) = \begin{cases} 1, \text{if } x \neq y \\ 0, \text{otherwise} \end{cases}$, which means we incur one unit of computation if the algorithmic state produced by $P$ differs from the one produced by $Q$, which corresponds to a re-computation. Under this computation model, the optimal expected computation cost is $\inf_{\pi \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\pi} [\mathbb{1}\{x \neq y\}]$, which coincides with the total variation (TV) distance between $P$ and $Q$. Therefore, if we want to transport $P$ to $Q$ using the minimum computation cost, the expected computation cost is determined by the TV distance between $P$ and $Q$. **This implies that at least $1-\mathrm{TV}(P, Q)$ fraction of samples are representative for both $P$ and $Q$, and thus at most $O(\mathrm{TV}(P, Q))$ fraction of unlearning requests need to be handled by re-training**. Moreover, due to the sequential nature of the unlearning problem, when we produce $P$, i.e., the output and internal state on dataset $\mathcal{D}$, we are unaware of what $Q$ will be, since we don't know the upcoming unlearning request. *Therefore, a desirable property to enable the unlearning ability for an FL algorithm is that its entire state should be close in TV distance uniformly over all possible $Q$'s.*

Motivated by the above discussion, we introduce the notion of sample-level and client-level TV-stability. This is a type of algorithmic stability that captures the closeness of the algorithmic states under different datasets. We will use this notion to guide the design and analysis of our federated learning and unlearning algorithms.

**Definition 3** ($\rho_S$-sample-level TV-stability)**.** An FL algorithm $\mathcal{L}$ is $\rho_S$-sample-level-TV-stable if

$$\sup_{\mathcal{D},\mathcal{D}':|\Delta(\mathcal{D},\mathcal{D}')|=1} \mathrm{TV}(\mathcal{L}(\mathcal{D}), \mathcal{L}(\mathcal{D}')) \leq \rho_S. \tag{3}$$

**Definition 4** ($\rho_C$-client-level TV-stability)**.** An FL algorithm $\mathcal{L}$ is $\rho_C$-client-level-TV-stable if

$$\sup_{C,C':|\Delta(C,C')|=1} \mathrm{TV}(\mathcal{L}(\mathcal{D}(C)), \mathcal{L}(\mathcal{D}(C'))) \leq \rho_C. \tag{4}$$

**Remark 1.** For any two datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \supset \mathcal{D}'$ and $|\Delta(\mathcal{D}, \mathcal{D}')| = w$, if $\mathcal{L}$ is $\rho_S$-sample-level TV-stable, then by the triangle inequality of TV and repeated applications of the Definition 3, we have that $\mathrm{TV}(\mathcal{L}(\mathcal{D}), \mathcal{L}(\mathcal{D}')) \leq w \cdot \rho_S$. Similarly, we have $\mathrm{TV}(\mathcal{L}(\mathcal{D}(C)), \mathcal{L}(\mathcal{D}(C'))) \leq w \cdot \rho_C$ for any $C, C'$ such that $C \supset C'$ and $|\Delta(C, C')| = w$ if $\mathcal{L}$ is $\rho_C$-client-level TV-stable.

We have established that TV-distance is a sufficient stability measure for exact federated unlearning based on our theoretical analysis. However, we do not claim that it is a necessary condition, nor that it is the optimal choice among other possible stability measures. This is a highly non-trivial problem that deserves further investigation in future work.

## 4 OUR ALGORITHMS

We propose our algorithms for federated learning and unlearning in this section. We devise a general framework that consists of a learning algorithm FATS and two unlearning algorithms to enable efficient data sample and client removal for FATS, respectively. In our algorithms, the server and clients will employ functions save$(\cdot)$ and load$(\cdot)$, which indicate saving and loading the variables to and from its local memory respectively.

### 4.1 TV-Stable FL Algorithm: FATS

Based on our previous discussions in Section 3.3, it is sufficient to design a TV-stable learning algorithm. Specifically, our objective is to achieve both $\rho_S$-sample-level TV-stability and $\rho_C$-client-level TV-stability simultaneously for any given stability parameters $\rho_S$ and $\rho_C$. For this purpose, we propose a federated learning algorithm called FATS, which extends the classical FedAvg algorithm [18] for TV stability. Due to the widespread usage of FedAvg, FATS can be easily integrated into existing systems. The main idea for FATS to achieve sample-level and client-level TV stability simultaneously is to elaborately adjust the number of clients sampled per round and the mini-batch size per iteration. We describe FATS in Algorithm 1. FATS operates in a synchronized manner, dividing the learning process into $T$ time steps $t = 1, 2, \dots, T$. These $T$ time steps are further grouped into $R$ communication rounds $r = 1, 2, \dots, R$, each consisting of $E$ iterations of local updates, such that $T = R \cdot E$. The $r$-th communication round covers iterations from $(r - 1) \cdot E + 1$ to $r \cdot E$. We denote by $\mathcal{I}_E = \{sE + 1 | s = 0, 1, 2, \dots, \lfloor \frac{T-1}{E} \rfloor\}$ the set of time steps that mark the start of a communication round. In each communication round, FATS performs the following three steps.

*STEP 1.* At the beginning of each communication round, the server randomly draws a multiset of $K$ clients with replacement and broadcasts the latest global model to them (steps 7-10). It is possible for a client to be activated multiple times in a single communication round. Let $C^{(t)}$ denote the chosen client multiset that allows repetitions. Note that $C^{(t)}$ is only defined for each $t \in \mathcal{I}_E$. For convenience, we use $\mathcal{P}^{(t)}$ for every $t \in [T]$ in the algorithm to denote the most recent selected clients at time step $t$, i.e., $\mathcal{P}^{(t)} := C^{(n)}$, where $n = \max\{t' | t' \leq t, t' \in \mathcal{I}_E\}$.

**Algorithm 1:** Federated Averaging with TV-Stability: $\texttt{FATS}(t_0, T, E, \eta, \rho_S, \rho_C)$

```
1  Input: Start iteration $t_0$, time horizon $T$, local iteration number $E$, learning rate $\eta$, TV-stability parameters $\rho_S$, $\rho_C$.
2  $K \leftarrow \frac{\rho_C \cdot E \cdot M}{T}, b \leftarrow \frac{\rho_S \cdot N}{\rho_C \cdot E}$ ;
3  if $t_0 \notin \mathcal{I}_E$ then
4      The server performs $\texttt{Load}(\mathcal{P}^{(t_0)})$ ;
5      The server informs each client $k \in \mathcal{P}^{(t_0)}$ to perform $\texttt{Load}(\theta_k^{(t_0)})$ ;
6  for $t \leftarrow t_0, \ldots, T$ do
7      if $t \in \mathcal{I}_E$ then
8          The server samples a multiset of clients $\mathcal{P}^{(t)}$ of size $K$ with replacement ;
9          The server performs $\texttt{save}(\mathcal{P}^{(t)})$, $\texttt{load}(\theta^{(t-1)})$ ;
10         The server broadcasts index $t$ and the latest model $\theta^{(t-1)}$ to all clients in $\mathcal{P}^{(t)}$ such that $\theta_k^{(t-1)} = \theta^{(t-1)}, \forall k \in \mathcal{P}^{(t)}$ ;
11     for each client $k \in \mathcal{P}^{(t)}$ (in parallel) do
12         Sample a mini-batch $\mathcal{B}_k^{(t)}$ of size $b$ uniformly at random without replacement ;
13         $\tilde{g}_k^{(t)} \leftarrow \frac{1}{b} \sum_{X \in \mathcal{B}_k^{(t)}} \nabla f(\theta_k^{(t-1)}; X)$ ;
14         $\theta_k^{(t)} \leftarrow \theta_k^{(t-1)} - \eta \cdot \tilde{g}_k^{(t)}$ ;
15         Perform $\texttt{Save}(t, \mathcal{B}_k^{(t)}, \theta_k^{(t)})$ ;
16     if $t \bmod E = 0$ then
17         Each client $k \in \mathcal{P}^{(t)}$ sends its latest local model $\theta_k^{(t)}$ to the server ;
18         The server aggregates the received local models as $\theta^{(t)} \leftarrow \frac{1}{K} \sum_{k \in \mathcal{P}^{(t)}} \theta_k^{(t)}$ ;
19         The server performs $\texttt{save}(\theta^{(t)})$ ;
20 return global model $\theta^{(T)}$ ;
```

*STEP 2.* Upon receiving the global model, the selected clients update the model parameters by using mini-batch stochastic gradient descent for $E$ iterations over their local datasets. Specifically, in each iteration $t$, the selected client $k \in \mathcal{P}^{(t)}$ first samples a mini-batch $\mathcal{B}_k^{(t)}$ of size $b$ from its local dataset $\mathcal{D}_k$ without replacement and then updates the local model $\theta_k^{(t-1)}$ using gradient descent based on the mini-batch gradient $\tilde{g}_k^{(t)}$ (step 12-17).

*STEP 3.* After completing $E$ local updates, the selected clients upload the local model parameters to the server, who then aggregates a new global model by averaging these local models (steps 18-22).

## 4.2 Unlearning Algorithms for FATS

We recall that FATS involves two sources of randomness: the server's client sampling and the client's data sample sampling. When a target data sample or a target client is removed, the number of available data samples for a client or the number of available clients will

**Algorithm 2:** Sample-level Unlearning for FATS: $\texttt{FATS-SU}(t_u, X_u, k_u)$

```
1  Input: Unlearning time step $t_u$, target sample $X_u \in \mathcal{D}_{k_u}$.
2  for $t \leftarrow 1, \ldots, t_u$ do
3      if $k_u \in \mathcal{P}^{(t)}$ then
4          perform $\texttt{Load}(\mathcal{B}_{k_u}^{(t)})$ ;
5          if $X_u \in \mathcal{B}_{k_u}^{(t)}$ then
6              $\texttt{FATS}(t_S, T, E, \eta, \rho_S, \rho_C)$ ;
7              halt ;
```

**Algorithm 3:** Client-level Unlearning for FATS: $\texttt{FATS-CU}(t_u, k_u)$

```
1  Input: Unlearning time step $t_u$, Target $k_u$ to unlearn.
2  $r_u \leftarrow \lfloor (t_u - 1)/E \rfloor + 1$ ;
3  for $r \leftarrow 1, 2, \ldots, r_u$ do
4      $t_r \leftarrow (r - 1) \cdot E + 1$ ;
5      if $k_u \in \mathcal{P}^{(t_r)}$ then
6          $\texttt{FATS}(t_C, T, E, \eta, \rho_S, \rho_C)$ ;
7          halt ;
```

change accordingly. This implies that the sampling probability distribution may differ if the algorithm is run on the updated setting (without the target data sample or client). Therefore, to achieve exact unlearning, we need to first verify whether such a discrepancy occurs, which is referred to as verification. If yes, we need to rectify this discrepancy by adjusting the sampling probability measure appropriately, which is done by re-computation. The sample-level and client-level unlearning algorithms are given in Algorithm 2 and Algorithm 3 respectively.

Suppose an unlearning request (for unlearning either a sample $X_u$ of $k_u$ or a client $k_u$) is made at time step $t_u$ in round $r_u$. For verification, For verification, we need to inspect whether the sampling probability distribution changes after deleting the target sample or client. At a high-level, we verify if the current model learned from the original dataset is *equally* probable to occur after the deletion for each iteration. We first consider the sample-level unlearning case. To unlearn the target sample $X_u \in \mathcal{D}_{k_u}$, we iteratively check for each time step $1 \leq t \leq t_u$ whether $\mathcal{B}_{k_u}^{(t))}$ can be generated by learning on the updated dataset **with the same probability**. Our specific approach is that, we check whether the target sample $X_u$ has ever been used for training the current model up till $t_u$. If not, then we affirm that no discrepancy exists between $P$ and $Q$ till $t_u$, and thus no re-computation is required to perform. Otherwise, if at some iteration $t_S \leq t_u$, we find the target sample was involved in the training, then the discrepancy occurs and a re-computation should then be initiated to retrain the model starting from $t_S$ (step 6 in Algorithm 2). The client-level unlearning case is simpler since we only need to account for the client sampling process. Specifically, we iteratively check for each round $r \leq r_u$ whether the target client $k_u$ was involved. If for some round with its first iteration being $t_C$, we find $k_u$ participated in the model training, then a re-computation should then be initiated (step 6 in Algorithm 3).

## 5 MAIN RESULTS

### 5.1 Unlearning Guarantees

In this part, we establish that our framework achieves both sample and client level exact unlearning. We first show that FATS is TV-stable. Then we show that our unlearning algorithms are valid transports that preserve the output distribution of the model parameters. Thanks to the TV-stability, the probability of re-computation is also small, which implies that our unlearning algorithms are efficient in computation and communication.

**Notations:** We introduce some notations that we will use for our unlearning analysis. For simplicity, we slightly deviate from the notations used in the algorithm description, which are elaborated in the following. For sample-level unlearning, we suppose that client $k_u$ requests to delete the target data sample $X_u$ from its local dataset $\mathcal{D}_{k_u}$ and $\mathcal{D}_{k_u}$ becomes $\mathcal{D}'_{k_u}$ after the deletion. For client-level unlearning, client $k_u$ is the target client. Let $\mathcal{D}$ and $\mathcal{D}'$ be the global dataset of all data points across the clients before and after the deletion, respectively. That is, $\mathcal{D}$ and $\mathcal{D}'$ differ by either the target sample $X_u \in \mathcal{D}_{k_u}$ for sample unlearning, or the whole local dataset $\mathcal{D}_{k_u}$ of the target client $k_u$ for client unlearning. Let $\theta_{\mathcal{D}}$ and $\theta_{\mathcal{D}'}$ be the model learnt from $\mathcal{D}$ and $\mathcal{D}'$, respectively. In each communication round $r$, there are two kinds of sampling: client sampling by the server and local mini-batch sampling by each client. We use $\mathcal{P}^{[r]}$ to denote the multiset of $K$ clients that server samples in round $r$. Let $\mathcal{B}_k^{[r]}$ denote the set of all mini-batches that client $k \in [M]$ samples during the $E$ iterations in round $r$, and let $\mathcal{B}_k^{[r,i]}$ denote the specific mini-batch that client $k$ samples in the $i$-th iteration of round $r$. Then we have $\mathcal{B}_k^{[r]} = \{\mathcal{B}_k^{[r,i]}\}_{i\in[E]}$ if $k \in \mathcal{P}^{[r]}$, and $\mathcal{B}_k^{[r]} = \emptyset$ otherwise. Finally, let $\mathcal{B}^{[r]} := \{\mathcal{B}_k^{[r]}\}_{k\in[M]}$ denote the set of all mini-batches sampled by all clients in round $r$.

In our analysis, we will use the concept of push-forward measure. Given a measurable function $f : \mathcal{X} \to \mathcal{Y}$ and a measure $\mu$ on $\mathcal{X}$, we denote by $f\#\mu$ the push-forward measure on $\mathcal{Y}$, defined as $(f\#\mu)(\mathcal{Z}) = \mu(f^{-1}(\mathcal{Z}))$ for any measurable set $\mathcal{Z} \subseteq \mathcal{Y}$.

To demonstrate that our framework ensures exact unlearning, we need the following crucial lemma, which states that our federated learning algorithm FATS is TV-stable with respect to both sample and client level.

**Lemma 1.** For any given $\rho_S, \rho_C \in (0, 1]$, FATS is $\min\{\rho_S, 1\}$ sample-level TV-stable and $\min\{\rho_C, 1\}$ client-level TV-stable.

The proof of Lemma 1 is in Appendix B.1. With Lemma 1, we are ready to demonstrate that our framework ensures exact unlearning. Furthermore, we show that the probability of re-computation is bounded linearly with the number of unlearning requests and the level of stability. The unlearning guarantee is formally stated in Theorem 1, whose proof is in Appendix B.2.

**Theorem 1.** (FATS, FATS-SU) satisfies sample-level exact federated unlearning. (FATS, FATS-CU) satisfies client-level exact federated unlearning. Moreover, the probability of re-computation for $w$ number of sample-level or client-level unlearning requests is at most $\rho_S \cdot w$ or $\rho_C \cdot w$, respectively.

### 5.2 Convergence Analysis

In this part, we provide convergence guarantees for our proposed FL algorithm FATS. We show that the global model converges to an approximate stationary point of the empirical risk function under mild assumptions. We also show that the unlearned models preserve the accuracy of the original models, while achieving exact unlearning. These results demonstrate the effectiveness of our federated unlearning framework.

**Notations:** Recall that, in Algorithm 1, the server first randomly selects a multiset $\mathcal{P}^{(t)}$ of clients at the beginning of each round and then only the selected clients perform local updates. This introduces some technical challenges in the analysis since $\mathcal{P}^{(t)}$ varies every $E$ time steps. To overcome this difficulty, we consider analyzing an alternative equivalent procedure where the server always activates all clients at the beginning of each round and then only aggregates the updated parameters from the clients in multiset $\mathcal{P}^{(t)}$ to generate the latest global model. Specifically, the updating scheme of the alternative procedure can be described as: for all $k \in [M]$ and any time step $t \in [T]$,

$$v_k^{(t)} = \theta_k^{(t-1)} - \eta \tilde{g}_k^{(t)}, \tag{5}$$

$$\theta_k^{(t)} = \begin{cases} v_k^{(t)}, & \text{if } t \bmod E \neq 0, \\ \frac{1}{K}\sum_{k\in\mathcal{P}^{(t)}} v_k^{(t)}, & \text{if } t \bmod E = 0. \end{cases} \tag{6}$$

For ease of exposition, we introduce some notations that will facilitate the analysis. Recall that, in Algorithm 1, we use $\tilde{g}_k^{(t)}$ to denote the stochastic mini-batch gradient of client $k$ in iteration $t$. We will use $g_k^{(t)}$ to denote the full gradient calculated from the entire local dataset at client $k$ in iteration $t$, i.e., $g_k^{(t)} := \nabla F_k(\theta_k^{(t)})$. Moreover, in Algorithm 1, $\theta^{(t)}$ is the global model that is aggregated among the selected clients in $\mathcal{P}^{(t)}$ every $E$ iterations at the end of each communication round. Thus, $\theta^{(t)}$ there is only defined for $t$ that is a multiple of $E$. In what follows, we extend the definition of $\theta^{(t)}$ for $\forall\, t \in [T] \cup \{0\}$ as $\theta^{(t)} := \frac{1}{K}\sum_{k\in\mathcal{P}^{(t)}} \theta_k^{(t)}$, which we also call the virtual average model.

Throughout our convergence analysis, we will use the following standard assumptions for loss function $f$.

**Assumption 1** ($L$-smoothness)**.** The loss function $f(\cdot; X)$ is $L$-smooth, *i.e.*, for any given data sample $X$ and $\forall\theta_1, \theta_2 \in \mathbb{R}^d$, we have $\|\nabla f(\theta_1; X) - \nabla f(\theta_2; X)\|_2 \leq L\|\theta_1 - \theta_2\|$.

**Assumption 2** (Bounded Local Variance)**.** For each local dataset $\mathcal{D}_k$, $k \in [M]$, we can sample an independent mini-batch $\mathcal{B}_k$ with $|\mathcal{B}_k| = b$ and compute an unbiased stochastic gradient $\tilde{g}_k$ defined as $\frac{1}{b}\sum_{X\in\mathcal{B}_k} \nabla f(\theta; X)$ for $\forall\theta$, with its variance bounded from above as $\mathbb{E}_{\mathcal{B}_k}[\|\tilde{g}_k - g_k\|_2^2] \leq \frac{G^2}{b}$, where $g_k = \mathbb{E}[\tilde{g}_k] = \frac{1}{N}\sum_{X\in\mathcal{D}_k} \nabla f(\theta; X)$.

We introduce the following notion of gradient diversity to quantify the dissimilarity between gradients of local empirical risk functions during the learning process.

**Definition 5** (Gradient Dissimilarity)**.** We define the following quantity as gradient diversity among all the clients at the $t$-th learning iteration: $\Lambda(\theta^{(t)}) := \frac{\frac{1}{M}\sum_{k=1}^{M} \|\nabla F_k(\theta_k^{(t)})\|_2^2}{\|\frac{1}{M}\sum_{k=1}^{M} \nabla F_k(\theta_k^{(t)})\|_2^2}$.

Clearly, for any $t$, $\Lambda(\theta^{(t)}) \geq 1$. And $\Lambda(\theta^{(t)}) = 1$ if and only if all the local gradients $\nabla F_k(\theta_k^{(t)})$'s are equal. In the following assumption, we introduce the quantity $\lambda$, which is the upper bound on the gradient diversity and measures the degree of heterogeneity.

**Assumption 3** (Bounded heterogeneity)**.** There is a common upper bound $\lambda$ on the gradient diversity among local empirical risk functions, such that for any $t \in [T]$, we have $\Lambda(\theta^{(t)}) \leq \lambda$.

With these assumptions, in the following Lemma 2, we bound the average-squared gradient norm for FATS in general. The proof of Lemma 2 can be found in Appendix C.1.

**Lemma 2.** Under Assumption 1, 2 and 3, if the learning rate $\eta$ is small enough and satisfies the following condition:

$$-\frac{\eta}{2} + \eta^3 L^2 \lambda E(E-1) + \frac{\eta^2 \lambda L}{2} < 0. \tag{7}$$

Then the average-squared gradient norm is bounded as

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\theta^{(t-1)})\|_2^2] \\
&\leq \frac{2(F(\theta^{(0)}) - F^*)}{\eta T} + \frac{\eta^2 L^2 G^2 E(K+1)}{Kb} + \frac{\eta L G^2}{Kb} \qquad (8)\\
&= \frac{2(F(\theta^{(0)}) - F^*)}{\eta T} + \frac{\eta^2 L^2 G^2 E(\rho_C EM + T)}{\rho_S MN} + \frac{\eta L G^2 T}{\rho_S MN}, \qquad (9)
\end{aligned}$$

where $F^*$ is the global minimum value.

In Lemma 2, we did not specify the choice of learning rate $\eta$ and per-round local iteration number $E$, and the obtained bound does not imply any convergence result for FATS. Now, we discuss the choice of $\eta$ and $E$ for ensuring convergence. To ensure convergence, one can choose $\eta = O\left(\frac{1}{T}\right)$. In this case, the condition (7) reduces to $\frac{E}{T} \leq O\left(\sqrt{\frac{1}{\lambda}}\right)$. That is, the number of local iterations should be dependent on the data heterogeneity and generally decreases as the data heterogeneity becomes larger. Such a constraint on $E$ is quite intuitive: when the data are heterogeneous, the average of the minimizers of $F_1, \ldots, F_K$ can be very different from the minimizer of $F$. If $E$ is set too large, then each $\theta_k^{(t)}$ can converge to the minimizer of $F_k$, which makes the algorithm diverge.

In the next Theorem 2, we formalize the above discussion and provide the convergence rate for FATS. The proof of Theorem 2 is in Appendix C.5.

**Theorem 2.** Define $\Gamma := \frac{G^2}{L(F(\theta^{(0)}-F^*)\rho_S MN}$. Under Assumption 1, 2 and 3, if we choose $\eta = \frac{1}{L\sqrt{\Gamma}T}$, and let $T \geq \frac{2\lambda}{\sqrt{\Gamma}}$, then condition (7) reduces to $\frac{E(E-1)}{T^2} < \frac{\Gamma}{4\lambda}$. Therefore, by requiring $\frac{E}{T} < \frac{1}{2}\sqrt{\frac{\Gamma}{\lambda}}$, we obtain the average-squared gradient norm bound of

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\theta^{(t-1)})\|_2^2] \\
&\leq \frac{3\sqrt{LG^2(F(\theta^{(0)}) - F^*)}}{\sqrt{\rho_S MN}} + L(F(\theta^{(0)}) - F^*)\frac{E}{T}(\frac{\rho_C ME}{T} + 1). \qquad (10)
\end{aligned}$$

**Remark 2.** The above bound consists of two terms, for which we have some observations in order:
(I) The first term can be seen as the cost for achieving stability, which scales as $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$ and thus is a non-vanishing term. If the per-client sample number $N$ and the client number $M$ are large enough, the cost can be made as small as possible. Though non-vanishing, we claim that this kind of stability cost is reasonable and it also appears in the existing centralized unlearning results, e.g., [25]. In fact, the algorithmic TV-stability means that the performance of an FL algorithm on similar data sets is somewhat indistinguishable. If the accuracy of an FL algorithm is very high (i.e., the average-squared gradient norm converges to 0), then this kind of indistinguishability will be broken. Therefore, in order to achieve algorithmic stability, it is acceptable and necessary to sacrifice a certain accuracy.
(II) It is worth noting that our stability cost only depends on the sample-level stability $\rho_S$ and has nothing to do with client-level stability $\rho_C$. This is because our stability cost comes from balancing the first and last terms in the (8), from which we can see that the cost actually determined by the effective number of data samples used in each iteration, i.e., $k \cdot b$. Since $K \propto \rho_C$ and $b \propto 1/\rho_C$, $\rho_C$ cancels out in $K \cdot b$. As a result, only $\rho_S$ makes a difference to the stability cost. This implies that FATS can achieve high client-level stability without compromising accuracy or communication efficiency if each client has enough local data samples.
(III) The second term scales as $O\left(\frac{E}{T}\right)$, which is controlled by the ratio of $E$ and $T$. To make this term diminish with $T \to \infty$, one can set $E = O(T^{1-\alpha})$ with $0 < \alpha \leq 1$. This way, the second term scale as $O(T^{-\alpha})$, and when $T$ is large enough, the entire bound will converge to the stability cost only. This actually reflects a trade-off between the convergence rate and the communication costs: larger $\alpha$ means faster convergence but also leads to more communication rounds. In some extreme cases, one can even set $E = c \cdot T$ for some constant $c < \frac{1}{2}\sqrt{\frac{\Gamma}{\lambda}}$. However, in this case, to retain the best possible convergence error of $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$, we will need an additional mild restriction on the client number $M$ so that $M \leq O(N)$. This shows that FATS can achieve a fast convergence rate with a small number of communication rounds under certain conditions.

With the perceptions above, we give in the following corollary the specific convergence error for different choices of $E$.

**Corollary 1.** Under Assumption 1, 2 and 3, for any $0 < \alpha \leq 1$ if we set $\eta = \frac{1}{L\sqrt{\Gamma}T}$, $E = T^{1-\alpha}$, and $T \geq \max\left\{\frac{2\lambda}{\sqrt{\Gamma}}, \left(2\sqrt{\frac{\lambda}{\Gamma}}\right)^{\frac{1}{\alpha}}, (\rho_C M)^{\frac{1}{\alpha}}\right\}$, then we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\theta^{(t-1)})\|_2^2] \leq \frac{4G\sqrt{L(F(\theta^{(0)}) - F^*)}}{\sqrt{\rho_S MN}} = O\left(\frac{1}{\sqrt{\rho_S MN}}\right).$$

If we set $E = c \cdot T$ for some constant $c < \frac{1}{2}\sqrt{\frac{\Gamma}{\lambda}}$, $T \geq \frac{2\lambda}{\sqrt{\Gamma}}$ and $M < \frac{4\lambda^2 L(F(\theta^{(0)})-F^*)\rho_S}{G^2\rho_C^2}N$, then we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla F(\theta^{(t-1)})\|_2^2] \leq \frac{4G\sqrt{L(F(\theta^{(0)}) - F^*)}}{\sqrt{\rho_S MN}} = O\left(\frac{1}{\sqrt{\rho_S MN}}\right).$$

**Remark 3.** The convergence error of FATS is dominated by the stability cost of $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$, where $\rho_S$ is closely related to the unlearning efficiency. Specifically, the smaller $\rho_S$ is, the more stable the algorithm is and the more efficient the unlearning process. In this way, $\rho_S$ characterizes the trade-off between the unlearning efficiency and learning accuracy. If we disregard unlearning efficiency and accept retraining computation for every sample removal request, then we can make $\rho_S > 1$ arbitrarily large to obtain, as expected, arbitrarily small convergence error. However, the intriguing case is when we make $\rho_S < 1$: in this case, we achieve a faster unlearning time and still a non-trivial accuracy, up to $\rho_S > O(\frac{1}{MN})$. This implies that we can attain a substantial decrease in communication and computation costs for federated unlearning without compromising too much accuracy for learning.

**Remark 4** (Utility of unlearned models)**.** Previously, we have shown a convergence error of $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$ for our FL algorithm FATS. This error bound reflects the accuracy of the original model trained on the full dataset. However, when we perform federated unlearning, we remove some data samples or clients from the dataset, which may affect the utility of the unlearned models. We now discuss the utility of the unlearned models and show that they preserve the same error bound as the original model under certain conditions. Since the error bound is mainly dependent on the total number of data samples $MN$, as long as the number of remaining data samples after deletion is still $O(MN)$, the convergence error bound of $O\left(\frac{1}{\sqrt{\rho_S MN}}\right)$ will still hold for the unlearned models. This means that the unlearned models have almost the same accuracy performance as the original model while satisfying the exact unlearning provability criterion. This conclusion will also be supported by our experiments.

## 5.3 Unlearning Time and Space Overheads

*5.3.1 Time Overheads.* We analyze the computational efficiency of FATS, which is a crucial aspect of federated unlearning, as it influences the scalability and responsiveness of the unlearning process. The total execution time of the unlearning algorithm consists of the time for verification and the time for re-computation. The most optimal way for verification is to maintain a dictionary of samples/clients to the earliest iteration/round that each sample/client participated in. This way, it requires $O(1)$ time lookup for every unlearning request. The time for each re-computation is bounded by the training time. According to Proposition 6 in [25], for a coupling-based unlearning algorithm with acceptance probability at least $1-\delta$ and for $w$ unlearning requests in general, the expected number of times re-computation is invoked is at most $4w\delta$. FATS-SU and FATS-CU re-compute with probability of $\min\{1, \rho_S\}$ and $\min\{1, \rho_C\}$, respectively. Hence, we can derive the expected number of re-computations for w sample-level or client-level unlearning requests as $4w\min\{1, \rho_S\}$ and $4w\min\{1, \rho_C\}$, respectively. Based on the analysis above, we obtain the expected running time for our framework as follows.

**Theorem 3.** *For $w$ sample-level unlearning requests, the expected unlearning running time is $O(\max\{\min\{\rho_S, 1\}w \times \text{Training time}, w\})$. Similarly, for $w$ client-level unlearning requests, the expected running time is $O(\max\{\min\{\rho_C, 1\}w \times \text{Training time}, w\})$.*

*5.3.2 Space Overheads.* We require the server and the local devices to store some intermediate states of the training process for FATS as in Algorithm 1. We enumerate the parameters to be stored on the server and on each local device. The server stores the subsets of clients in each communication round (i.e., $\mathcal{P}^{(t)}$) and the global models (i.e., $\theta^{(t)}$). For every training iteration $t$ that client $k$ is selected, client $k$ preserves the iteration index (i.e., $t$), local mini-batches (i.e., $\mathcal{B}_k^{(t)}$) and local models (i.e., $\theta_k^{(t)}$). We conduct a space complexity analysis of our algorithms. Each local device stores all the mini-batches and local models that it utilizes for every training iteration, which consumes up to $O(T \cdot b)$ and $O(T \cdot d)$ words respectively via dictionaries that associate iteration index with mini-batch samples and local models. The space complexity for each device is $O(T \cdot \max\{b, d\}) = O\left(T \cdot \max\left\{\frac{\rho_S N}{\rho_C E}, d\right\}\right)$. The server stores the involved clients and the global models in each communication round, which occupies $O(R \cdot K)$ and $O(R \cdot d)$ words respectively via dictionaries that link round index with client subsets and global models. The space complexity for the server is $O(R \cdot \max\{K, d\}) = O\left(\max\left\{\rho_C M, \frac{Td}{E}\right\}\right)$.

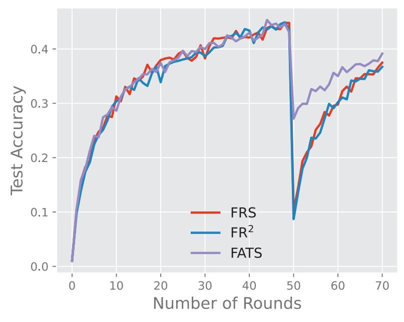
We remark that our unlearning algorithms employ the stored local models, mini-batch samples, and client sub-sets only for presentation convenience. We can easily devise a simple but efficient implementation of our unlearning algorithms, which does not store any local models, mini-batch samples, or client sub-sets, but has the same unlearning time complexity as in Theorem 3. In Theorem 3, we bound the re-computation time by the full re-computation time, i.e., training time. This implies that the unlearning time bound holds even if we do full re-computation every time the discrepancy arises. The unlearning algorithms can be modified as: instead of retraining from iteration $t_0$ where the target sample or client is involved, we do full retraining from the beginning. With this, each local device stores a vector of length $N$ to indicate the involvement of each local sample, and the server stores a vector of length $M$ to indicate the participation of each client, which takes $O(N)$ and $O(M)$ bits respectively. Both the server and each device store a $d$-dimensional model, which costs $O(d)$ words. The space complexities of each device and the server are $O(N+d)$ and $O(M+d)$ words, which are independent of $T$ and acceptable even when $T$ is large.

## 6 EXPERIMENTS

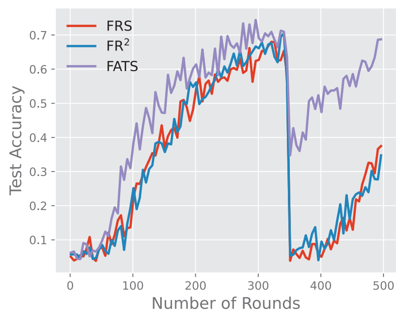
To demonstrate the advantages of our proposed framework in terms of performance and scalability of learning/unlearning, we conduct comprehensive experiments on 6 federated learning benchmark datasets and present empirical evidence of its superiority. Specifically, we compare our framework with state-of-the-art methods for federated unlearning and evaluate the accuracy, communication cost, computation cost, and unlearning effectiveness of different methods. We also investigate the impact of different stability parameters on the performance of our framework.

## 6.1 Experimental Settings

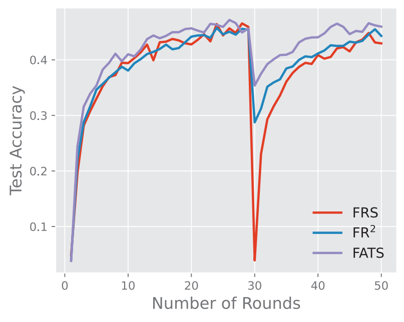
*6.1.1 Datasets.* We adopt several widely-used benchmark datasets. These datasets can be categorized into two categories: *simulated*



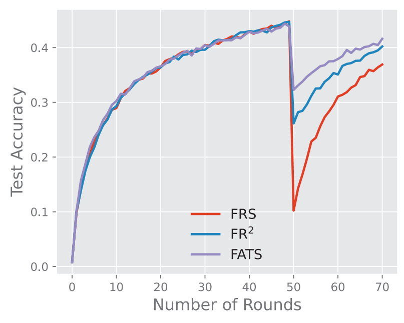
**(a) Cifar-100**



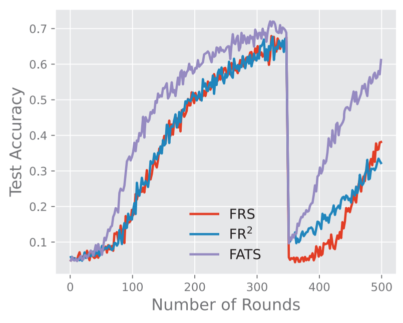
**(b) FEMNIST**



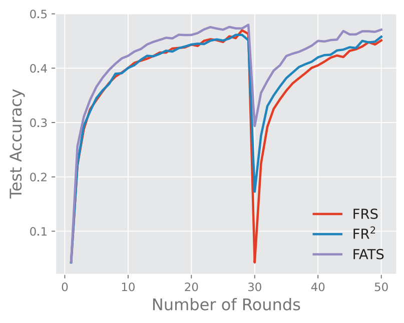
**(c) Shakespeare**



**(d) Cifar-100**



**(e) FEMNIST**



**(f) Shakespeare**

**Figure 1: Comparison of the test accuracy of different methods and their changes after unlearning on Cifar-100, FEMNIST, and Shakespeare. *Top:* sample-level unlearning. *Bottom:* client-level unlearning.**

*federated datasets* and *real federated datasets*. For simulated federated datasets, we manually partition centralized datasets into the federated setting. For real federated datasets, we use datasets from the well-known FL benchmark LEAF [3] whose datasets are originally built for real-world federated settings.

**Simulated federated datasets.** We use MNIST [15], Fashion-MNIST (FashionM) [32], Cifar-10 and Cifar-100 [14] and apply the Label-based Dirichlet Partition (LDA) [12] on them to simulate the label-based non-i.i.d. data distribution. These datasets are widely used for image classification tasks in FL, e.g., [18]. Each dataset is partitioned into 100 clients using the LDA, assigning the partition of samples to clients by $p \sim \mathrm{Dir}(\beta)$, where Dir is the probability density function of the Dirichlet distribution and $\beta$ is a positive real parameter. Smaller $\beta$ leads to a higher heterogeneity level of partition. Unless otherwise stated, we set $\beta$ to 0.5 by default.

**Real federated datasets.** We use FEMNIST and Shakespeare (Shakes) from LEAF and they cover both vision and language tasks. We sample the full-sized datasets and partition each client's samples into training/test groups as per [3] except that we set the minimum number of samples per client to 100.

For more details of the datasets, see Table 2 in Appendix A.1.

### 6.1.2 *Models and Hyperparameters.*

We adopt different models for different FL tasks, based on the complexity and modality of the data. Specifically, for image classification tasks, we use the CNN model for MNIST, Fashion-MNIST, FEMNIST, and the VGG16 pre-trained on ImageNet1K for Cifar-10 and Cifar-100. The learning rate is set to be 0.001 for both models. For the natural language generation task on Shakespeare, we use the same model architectures as reported in [3]: the model first maps each character to an embedding of dimension 8 before passing it through an LSTM of two layers of 256 units each. The LSTM emits an output embedding, which is scored against all items of the vocabulary via a dot product followed by a softmax. We use a sequence length of 80 for LSTM and the learning rate of LSTM is set as 0.8. All other hyperparameters are set to the values reported in Table 2 for the respective datasets.

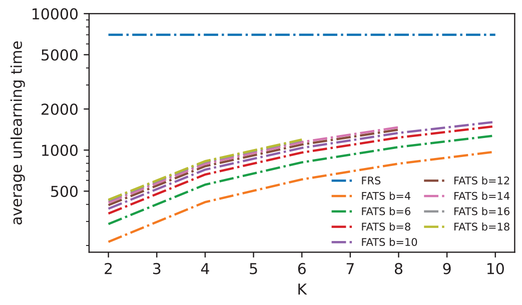
### 6.1.3 *Evaluation Metrics.*

We use the following metrics to assess the learning and unlearning performance of different methods on various datasets and tasks.

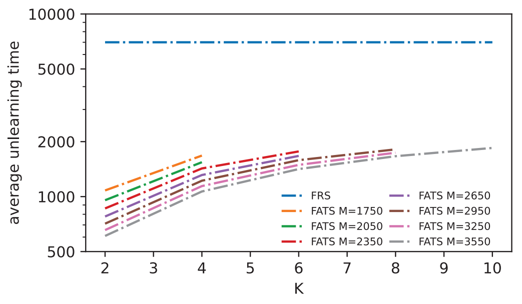
**Test accuracy.** This metric quantifies the accuracy of the global model for different tasks on the test data of various datasets. For image datasets, we assess the classification accuracy of the model on the test data, which means the ratio of correct predictions to the total number of predictions. For Shakespeare dataset, we gauge the top-1 accuracy of the model for predicting the next word in a sequence, which means the fraction of the model answer (the one with highest probability) that matches the actual words in the test data. Higher accuracy indicates better performance.

**Unlearning time**. This metric measures the number of time steps required by an unlearning algorithm to unlearn target data samples or clients. It reflects the computation/communication efficiency and scalability of the unlearning algorithm. Fewer unlearning time means higher efficiency.

**MIA accuracy and precision**. These two metrics measure the vulnerability of the global model under the membership inference

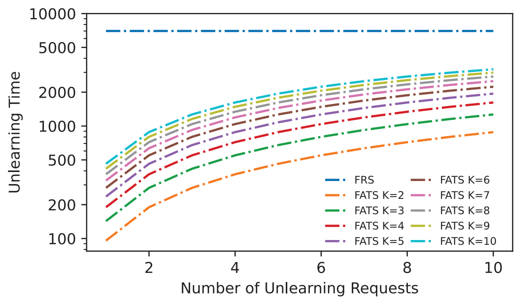


(a) FEMNIST: Sample Unlearning Case

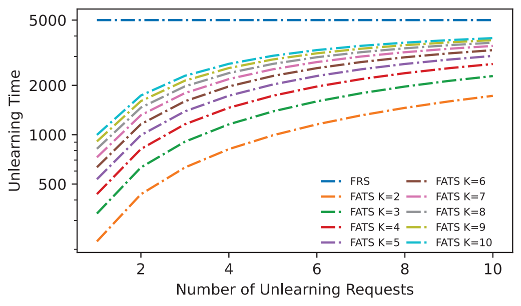


(b) FEMNIST: Client Unlearning Case

Figure 2: Unlearning Efficiency of FATS compared with FRS.



(a) FEMNIST



(b) Shakespeare

Figure 3: Impacts of the number of unlearning requests on unlearning efficiency.

attack (MIA) [22]. MIA aims to infer whether a given target data sample was used to train the model or not. Thus, the performance of MIA reflects the amount of information that remains in the unlearned global model, which indicates the unlearning efficacy. Accuracy and precision are two complementary metrics for MIA to assess how well the attack can infer the membership status of a given data point. Accuracy is the fraction of correct predictions over all predictions, while precision is the fraction of correct positive predictions over all positive predictions. In other words, accuracy measures how often the attack is right, while precision measures how confident the attack is when it predicts a positive membership. When applying MIA to unlearned models, the closer the MIA accuracy or precision is to 50% (which intuitively represents a random guess), the higher the unlearning efficacy.

For the test accuracy and unlearning time, we will report the averaged results over 5 independent runs. For MIA accuracy and precision, we will perform MIA 100 times and report the average performance as well as the standard deviation.

*6.1.4 Baseline Methods.* To show the advantages of our framework, we compare it with two existing methods: Federated Retraining from Scratch (FRS) and Federated Rapid Retraining (FR$^2$) [17]. For these two methods, we use the most widely used FL algorithm, federated averaging (FedAvg) [18], to train the global FL model. The unlearning strategy of these two methods is described as follows.

**FRS**. This method retrains the FL model from scratch on the remaining data after removing target samples or clients. This is the simplest but most time and communication intensive method.

**FR$^2$**. This method employs a diagonal approximation of the Fisher Information Matrix to achieve rapid retraining and introduces the momentum technique to enhance model utility. This is a state-of-the-art federated unlearning framework, applicable to sample-level and client-level unlearning.

We implement all methods based on Flower [1], a scalable and efficient open-source FL framework.

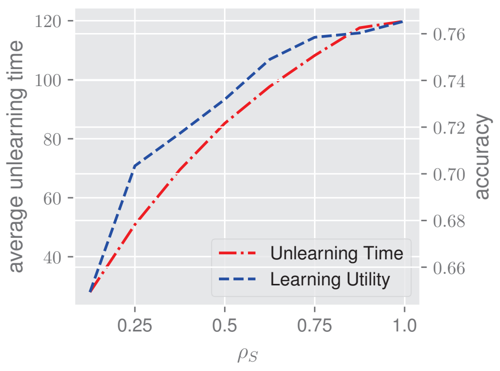
## 6.2 Experimental Results

*6.2.1 Performance Evaluation.* We compare the learning performance and the unlearning capability of our proposed method with two baseline methods, FRS and FR2, on all 6 datasets. Our main focus in this experiment is to examine how these methods can achieve FL training, handle unlearning requests, and recover from data deletion. To this end, we first train a global FL model for all these methods, then we issue some unlearning requests to them and let them update the model accordingly. We measure and present the model test accuracy throughout the whole process. Due to space limit, here we report the results on Cifar-100, FEMIST and Shakespeare in Figure 1. Other results are included in Appendix A.2. We will also study the streaming unlearning setting in Appendix A.5.

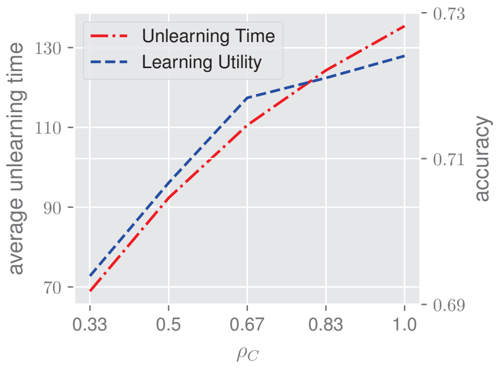
Regarding the unlearning request issue time, we vary the issue iterations for different datasets according to their convergence speed. In fact, there is no clear criterion to decide when an FL model is fully trained, and unlearning requests can be issued at any time during the FL training process. If an unlearning request is issued in the initial stage of FL training, even the naive retraining method would be relatively cheap. Therefore, in this experiment, we choose

**Table 1: Membership Inference Attack (MIA) Results on Six Datasets.**

| Datasets | Accuracy | | | Precision | | |
|---|---|---|---|---|---|---|
| | FRS | FR$^2$ | FATS | FRS | FR$^2$ | FATS |
| MNIST | 50.10± 0.25% | 49.47± 0.42% | 50.06± 1.40% | 50.55± 0.13% | 49.70± 0.24% | 50.30± 0.78% |
| FMNIST | 47.85± 1.93% | 48.47± 5.20% | 48.29± 5.28% | 49.13± 1.55% | 48.91± 4.99% | 49.48± 4.40% |
| Cifar-10 | 49.04±4.03% | 51.23±2.25% | 49.15±1.91% | 49.53±11.34% | 54.17±7.48% | 49.20±0.38% |
| Cifar-100 | 49.34±0.53% | 47.89±1.53% | 48.61±1.29% | 49.20±0.61% | 48.03±1.39% | 48.90±0.99% |
| FEMNIST | 50.14±0.23% | 50.00±0.00% | 53.30±2.34% | 55.01±17.22% | 33.20±42.12% | 55.64±4.72% |
| Shakes | 49.68±0.26% | 49.44±0.04% | 49.35±0.02% | 49.30±0.51% | 48.87±0.13% | 49.04±0.28% |



(a) Fashion-MNIST: varying $\rho_S$



(b) Fashion-MNIST: varying $\rho_C$

**Figure 4: Impacts of stability parameters on learning utility and unlearning efficiency.**

an issue iteration for each dataset such that the test accuracy has reached a stable level. At such an iteration, the FL model has been adequately trained and the performance gap between different unlearning methods is evident. For the unlearning requests, we consider both sample-level and client-level unlearning scenarios. Specifically, we randomly select 10 samples/clients for MNIST and FEMIST and 5 samples/clients for the other datasets to be unlearned simultaneously. The reason why we issue a batch of unlearning requests is that, if we only issue one request, re-computation may not be triggered in FATS, and then we will not be able to compare how these methods cope with sample or client deletion.

For the learning process, we can see that FATS achieves comparable or even superior (e.g., for FEMNIST and Shakespeare) test accuracy than the baselines before unlearning over all the datasets. This indicates that FATS can learn as effectively as the classical FL algorithm FedAvg while achieving TV-stability at the same time.

For the unlearning process, we can see that our method consistently outperforms the baselines across all datasets. FRS suffers from a significant drop in accuracy after data deletion, which incurs more rounds to recover the model utility. This shows that FRS is inefficient and costly for unlearning, as it requires retraining the model from scratch on the updated dataset. FR$^2$ has difficulty on converge and requires more communication rounds to achieve stable performance, as shown by the large fluctuations on MNIST, Fashion-MNIST, and FEMNIST. This shows that FR$^2$ is not robust and unreliable for unlearning, as it relies on a diagonal approximation of the Fisher Information Matrix that may not capture the true Hessian information. In contrast, our framework achieves rapid and effective unlearning with minimal loss in accuracy and communication cost. The accuracy drop of FATS is significantly smaller than that of all baselines over all the datasets. This shows that our method can unlearn more quickly and accurately without affecting the model utility too much. This demonstrates the superiority of our method over the existing methods for federated unlearning.

In summary, FATS achieves effective learning and unlearning performance in the federated setting. It learns as fast as FedAvg and achieves comparable or better accuracy before unlearning. It unlearns more rapidly and accurately than the baselines and preserves more model utility after unlearning.

*6.2.2 Unlearning Efficiency.* The unlearning efficiency of our framework is influenced by the stability parameters $\rho_S$ and $\rho_C$ for sample-level and client-level unlearning, respectively. These parameters are related to the hyperparameters $K, T, E, M, b$, and $N$. Specifically, $\rho_C = (K \cdot T)/(E \cdot M)$ and $\rho_S = (b \cdot K \cdot T)/(N \cdot M)$. We aim to assess the unlearning efficiency of our framework and contrast it with the baseline methods under different settings of stability parameters $\rho_S$ and $\rho_C$. We manipulate $\rho_S$ and $\rho_C$ by adjusting the values of the hyperparameters. Specifically, we keep $T$ and $E$ constant as shown in Table 2 in this experiment and only vary $M$, $K$ and $b$. We test the unlearning efficiency of FATS on the FL benchmark datasets FEMNIST and Shakespeare. We present the results on FEMNIST in Figure 2. The results on Shakespeare can be found in Appendix A.3.

For sample-level unlearning, we fix the hyperparameters $T$, and $M$, and vary $K$ for each chosen $b$, which changes the sample unlearning ratio $\rho_S$ accordingly. Since $\rho_S$ is proportional to $K$ when

$M$, $N$, $T$, and $b$ are fixed, a larger $K$ corresponds to a higher $\rho_S$. The results are shown in the first row of Figure 2. Each line therein ends at the largest $K$ such that $\rho_S$ reaches 1. We can see that FATS requires fewer rounds to fully recover model utility for sample unlearning, compared to the baseline FRS.

For client-level unlearning, we fix the hyperparameters $T$ and $E$, and vary $K$ for each chosen $M$, which changes the client unlearning ratio $\rho_C$ accordingly. Since $\rho_C$ is proportional to $K$ when $M$, $T$, and $E$ are fixed, a larger $K$ corresponds to a higher $\rho_C$. The results are shown in the second row of Figure 2. Each line therein ends at the largest $K$ such that $\rho_C$ reaches 1. We find that FATS takes more time for larger values of $K$ as expected, but its largest average unlearning time is still less than half of the time taken by FRS.

In summary, our proposed federated unlearning framework demonstrates superior efficiency over baseline approaches on large real-world federated datasets. It accomplishes unlearning with less time and rounds while restoring model accuracy. These results highlight the effectiveness of our proposed method for efficient federated unlearning.

*6.2.3 Impact of the Number of Unlearning Requests.* As stated in Theorem 3, the time taken to unlearn data is greatly affected by the number of unlearning requests. In this experiment, we examine how the number of unlearning requests influences the unlearning efficiency in practice. We conduct experiments on FEMNIST and Shakespeare. For FEMNIST, we use $M = 3550$ clients and for Shakespeare, we use $M = 630$ clients. For both datasets, we consider the settings with K varying from 2 to 10. Since different K corresponds to different stability parameter $\rho_C$, we also examine whether the influence of unlearning request number holds consistently under different stability parameters. We issue $1 - 10$ client unlearning requests respectively for each setting and report the average unlearning time in Figure 3. For comparison, we choose FRS as the baseline method. As we can see from Figure 3, when $\rho_C$ is fixed, the unlearning time is positively correlated with the number of unlearning requests. Moreover, since larger $K$ corresponds to larger $\rho_C$, we can also see that, the unlearning time is also positively correlated with the stability parameter when the unlearning request number is fixed. We can also observe that with some appropriate $K$, the unlearning time can still be significantly smaller than it of FRS even if there are multiple unlearning requests. All these experimental results corroborate our theoretical results.

*6.2.4 Utility v.s. Efficiency.* To further analyze the trade-off between learning utility and unlearning efficiency, we show the average unlearning time and accuracy of FATS as a function of $\rho_C$ and $\rho_S$ on MNIST and Fashion-MNIST. Here we present the results on Fashion-MNIST in Figure 4. The results on MNIST can be found in Appendix A.4. The first row of Figure 4 shows how the average unlearning time and accuracy of FATS change as $\rho_S$ increases from 0.125 to 1. We can see that the accuracy curves rise rapidly at first, then slowly level off. The unlearning time curves exhibit a similar trend, decreasing sharply initially before plateauing as $\rho_S$ approaches 1. The second row of Figure 4 shows how the average unlearning time and accuracy of FATS change as $\rho_C$ increases from 0.2 to 1 for MNIST and from 0.33 to 1 for Fashion-MNIST. We can see that the accuracy curves increase faster than the unlearning time curves at first, but then start to flatten out when $\rho_C$ exceeds 0.5, while the unlearning time curves keep increasing. This suggests an optimal trade-off point around $\rho_C = 0.5$, where good learning utility is balanced with efficient client-level unlearning. Overall, we can observe that $\rho_S$ has a higher impact on the learning utility than $\rho_C$. This is also supported by the convergence analysis we conducted in Section 5.2.

*6.2.5 Membership Inference Attack.* We conduct *Membership Inference Attack* (MIA) [22] on the final unlearned global models trained by different federated unlearning methods. The MIA results are shown in Table 1. Our method exhibits high unlearning effectiveness, as the attack gains no advantage over random guessing and cannot reliably discern membership in the training set. The performance of FATS and FRS is generally consistent. This suggests that our method indeed achieves *exact* federated unlearning. We note that the state-of-the-art baseline FR$^2$ achieves very low MIA precision on the FEMNIST dataset, which indicates that FR$^2$ is not robust and unreliable for unlearning. In summary, the MIA results verify that our method indeed satisfies exact federated unlearning, and demonstrate the superiority of our method over the existing methods for federated unlearning.

## 7 CONCLUSIONS

In this paper, we presented the first algorithmic framework for federated unlearning that attains communication efficiency and unlearning provability. The newly introduced notion of exact federated unlearning guarantees the total removal of a client's or a sample's effect on the global model. We devised a TV-stable FL algorithm FATS, which reduces the communication rounds by using local SGD with periodic averaging. We also developed matching unlearning algorithms for FATS to cope with both client-level and sample-level unlearning scenarios. Furthermore, we offered theoretical analysis to demonstrate that our learning and unlearning algorithms fulfill exact unlearning and achieve reasonable convergence errors for both the learned and unlearned models. Lastly, we verified our framework through comprehensive experiments on 6 benchmark datasets, and revealed that our framework outperforms existing baselines, while considerably lowering the communication and computation cost, and fully erasing the impact of a specific data sample or client from the global FL model.

## ACKNOWLEDGMENTS

Dongxiao Yu, Xiuzhen Cheng and Youming Tao are supported in part by National Natural Science Foundation of China (NSFC) under Grant 62122042, Major Basic Research Program of Shandong Provincial Natural Science Foundation under Grant ZR2022ZD02, Fundamental Research Funds for the Central Universities under Grant 2022JC016. Di Wang and Cheng-long Wang are supported in part by the baseline funding BAS/1/1689-01-01, funding from the CRG grand URF/1/4663-01-01, REI/1/5232-01-01, REI/1/5332-01-01, FCC/1/1976-49-01 from CBRC of King Abdullah University of Science and Technology (KAUST). They are also supported by the funding RGC/3/4816-09-01 of the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI). Miao Pan was supported in part by the US National Science Foundation under grant CNS-2107057 and CNS-2318664.

## REFERENCES

[1] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).

[2] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy (SP 2021)*. IEEE, 141–159.

[3] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[4] Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *36th IEEE Symposium on Security and Privacy (SP 2015)*. IEEE, 463–480.

[5] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In *29th ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*. 499–513.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *3rd Theory of Cryptography Conference (TCC 2006)*. Springer, 265–284.

[7] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. 2022. Sequential Informed Federated Unlearning: Efficient and Provable Client Unlearning in Federated Optimization. *arXiv preprint arXiv:2211.11656* (2022).

[8] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. , 3513–3526 pages.

[9] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. 2020. Certified data removal from machine learning models. In *37th International Conference on Machine Learning (ICML 2020)*. 3832–3842.

[10] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated Unlearning: How to Efficiently Erase a Client in FL? *arXiv preprint arXiv:2207.05521* (2022).

[11] Elizabeth Liz Harding, Jarno J Vanto, Reece Clark, L Hannah Ji, and Sara C Ainsworth. 2019. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy* 2, 3 (2019), 234–253.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[13] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[14] Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto* (2009).

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

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

[17] Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. 2022. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *41st IEEE International Conference on Computer Communications (INFOCOM 2022)*. IEEE, 1749–1758.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*. PMLR, 1273–1282.

[19] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. 2021. Descent-to-delete: Gradient-based methods for machine unlearning. In *32nd International Conference on Algorithmic Learning Theory (ALT 2021)*. PMLR, 931–962.

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

[21] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. 2021. Remember what you want to forget: Algorithms for machine unlearning. In *35th Annual Conference on Neural Information Processing Systems (NeurIPS 2021)*. 18075–18086.

[22] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *38th IEEE Symposium on Security and Privacy (SP 2017)*. IEEE Computer Society, 3–18.

[23] Mengkai Song, Zhibo Wang, Zhifei Zhang, Yang Song, Qian Wang, Ju Ren, and Hairong Qi. 2020. Analyzing user-level privacy attack against federated learning. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2430–2444.

[24] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. 2022. Unrolling sgd: Understanding factors influencing machine unlearning. In *7th IEEE European Symposium on Security and Privacy (EuroS&P 2022)*. IEEE, 303–319.

[25] Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. 2021. Machine unlearning via algorithmic stability. In *32nd International Conference on Algorithmic Learning Theory (ALT 2021)*. PMLR, 4126–4142.

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

[27] Cheng-Long Wang, Mengdi Huai, and Di Wang. 2023. Inductive Graph Unlearning. In *32nd USENIX Security Symposium (USENIX Security 2023)*. 3205–3222.

[28] Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. 2022. Federated unlearning via class-discriminative pruning. In *31st ACM Web Conference (WWW 2022*. 622–632.

[29] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. 2019. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *38th IEEE International Conference on Computer Communications (INFOCOM 2019)*. IEEE, 2512–2520.

[30] Chen Wu, Sencun Zhu, and Prasenjit Mitra. 2022. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441* (2022).

[31] Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. 2022. Federated Unlearning: Guarantee the Right of Clients to Forget. *IEEE Network* 36, 5 (2022), 129–135.

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

[33] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep Leakage from Gradients. *33rd Annual Conference on Neural Information Processing Systems (NeurIPS 2019)* 32 (2019), 14774–14784.