Exact-Fun: An Exact and Efficient Federated Unlearning Approach

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

***Abstract*—Machine unlearning is an emerging need that aims to remove the influence of deleted data from a learned model in a timely manner. Thus, unlearning is important for the privacy and security in data management. Nevertheless, existing machine unlearning methods fail to perform exactly and efficiently in federated setting. In this paper, we study the unlearning problem in federated learning, which provides a data deletion mechanism in the federated setting. First of all, a quantized federated learning (Q-FL) algorithm is developed to facilitate exact unlearning. Based on the quantized federated learning system, an exact and efficient federated unlearning (Exact-Fun) algorithm is designed to realize the goal of data deletion. Through theoretic analysis and experimental evaluation, our proposed methods not only have desired unlearning effectiveness but also achieve high unlearning efficiency compared with the existing works.**

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

I. Introduction

As well known, to advance the performance of machine learning models, a sufficient amount of data is indispensable to be collected from users and/or third parties. For examples, popular computer vision models are trained on images and videos posted by Facebook and Flickr users [31], many natural language processing models are trained on Amazon reviews [27], and micro-video recommendation systems are trained on Tiktok user data [23]. In a number of real applications, users provide their data to the service providers/platforms for model development in exchange of better service quality. Meanwhile, to protect users' data privacy, the "Right To Be Forgotten" is enforced by some regulations, such as General Data Protection Regulation (GDPR) [37] and California Consumer Privacy Act (CCPA) [35]. For instance, a user wants to delete part of search history, and a hospital requests to remove some patients' data. *In these scenarios, a practical and crucial question is that when users request to remove data from the services or platforms, what should the service providers do*?

A straightforward method to deal with users' data removal requests is to delete the users' data from the databases. However, due to the memorization of machine learning models [34], the information of training data is memorized in model parameters and cannot be forgotten easily. Moreover, such naive data deletion can be explored by malicious attackers to infer users' private information in various ways, including model inversion attack [10], membership inference attack [20], reconstruction attack [12], *etc*. Therefore, how to correctly and completely remove users' data from the learned models has become a challenging problem for data management.

The rightful data removal in machine learning context, termed as "machine unlearning" [6], requires deleting data from training datasets as well as the impact of data in the learned models. Intuitively, retraining the machine learning models from scratch on the remaining databases sans the deleted data is a simple way to achieve unlearning objective, but full computation cost of retraining may not be affordable, especially on the models with millions of parameters [9]. Thus, designing a computation-efficient and time-saving unlearning method is the core focus of current machine unlearning works. So far, there are only a few works on machine unlearning but with different limitations, such as simple learning methods (linear regression [18], [22]), model-dependent methods (decision tree [5] and k-means cluster [13]). Besides, existing research on federated learning (FL) mainly focuses on improving unlearning efficiency with approximate unlearning but overlooks model utility (*e.g.,* model accuracy) after unlearning, which harms the performance of unlearned models.

Inspired by the limitations of existing unlearning methods, in this paper, we aim at designing an exact and efficient federated machine unlearning method in model-agnostic manner. First, to enable exact federated unlearning, we utilize the idea of α-quantization [17] to improve the stability of federated model and propose our quantized federated learning (Q-FL) algorithm, through which the quantized federated model can maintain unchanged before and after data deletion. More importantly, the quantized federated model and the model retrained from scratch on the remaining databases could be the same with a high probability, so that there is no need to retrain from scratch as long as the stability is held. Based on the quantized federated model, we design an exact and efficient federated unlearning (Exact-Fun) algorithm that also can achieve decent unlearning efficiency and good model utility after unlearning. We highlight the contributions of this paper as follows:

- To the best of our knowledge, this paper is the first work to investigate the **exact federated unlearning** in FL, which can be extended to different machine learning models (model agnostic).
- The quantized federated unlearning (Q-FL) algorithm is designed to enable exact federated unlearning with the

guarantee of model convergence.

- The exact and efficient federated unlearning (Exact-Fun) algorithm is proposed to process users' data deletion requests with proved unlearning efficiency.
- Both our Q-FL and Exact-Fun algorithms are evaluated on real datasets via intensive experiments, which validate the effectiveness and efficiency of our proposed algorithms compared with state-of-the-art.

This paper is organized as follows. The related works and the preliminaries are introduced in Section II and Section III, respectively. In Section IV, we detail our methodology. Then, our proposed algorithms are evaluated in Section V. Finally, we give a conclusion in Section VI.

II. Related Works

Existing works on unlearning can be categorized into two branches, *i.e.*, exact unlearning and approximate unlearning, according to their efficiency and effectiveness.

Exact unlearning requires the distribution of unlearned model parameters should be exactly same as the distribution of model parameters that are retrained on the dataset without the deleted data. Since this requirement is hard to be achieved in complicated models, most exact unlearning strategies are designed on simple learning models. [6] first designed unlearning algorithms for statistical query based learning models, such as Naive Bayesian classifier and SVM, where the strategies are used to maintain model statistics at learning stage and update parameter information when unlearning data comes. Then, Ginart *et al.* [13] proposed the first unlearning method for unsupervised learning k-means cluster algorithm, which adopts stability and divide-and-conquer to improve unlearning efficiency. In [4], a SISA framework was designed to reduce unlearning time through sharding, isolation, slicing, and aggregation, of which the idea is to split a dataset to small parts so that the retraining time is reduced. Similarly, Aldaghri *et al.* [1] adopted ensemble learning to split training dataset into disjoint shards by coding matrix. When unlearning is performed, the unlearned data is removed from coded shards, and the corresponding model is retrained. Following this, [5], [32] conducted unlearning algorithms on random forests algorithm, where they used a similar idea to adjust the structure of decision tree such that the retrained subtree can be minimized. In the above works, the exact unlearning strategies realize the unlearning requirement by retraining a part of models, which can guarantee unlearning effectiveness but reduce unlearning efficiency.

On the contrary, approximate unlearning prefers unlearning efficiency to unlearning effectiveness. Different from exact unlearning, the distribution of approximately unlearned model parameters is similar to the distribution of retrained model parameters with less unlearning time. Instead of retraining a part of model, approximate unlearning methods perform a post-processing on the learned models to obtain an approximation of the fully retrained models. In the research of [16], [40], [41], authors used similar idea to achieve approximate unlearning by updating the trained model with stored gradients when certain data points are removed. Neel *et al.* [29] and Ullah *et al.* [36] imported statistical indistinguishability and algorithm stability respectively to unlearn data via gradient descent with provable approximation. To cover the adversarial scenario where a user deliberately deletes data under specific distribution, Gupta *et al.* [19] proposed adaptive machine unlearning that can handle arbitrary model classes and training methodologies. On the other hand, the authors of [18], [33] proposed differentially private data removal mechanisms, which can unlearn data from the learned model by hessian matrix, and Golatkar and Wang [14], [38] focused on unlearning a specific class label from deep networks. Considering the computational cost of hessian matrix, Izzo *et al.* [22] found a sublinear algorithm to speed up unlearning from linear models efficiently. In another branch, a few probability-based unlearning methods were utilized to solve approximate unlearning under federated settings with Bayesian [8], [15] and Monte Carlo [30]. To sum up, approximate unlearning runs faster than exact unlearning but fails to improve model utility after unlearning. Therefore, it is challenging to design an exact and efficient federated unlearning method because it is still an open problem.

III. Preliminary

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

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

where L_k is the loss function of client k, w_k^t is the model parameter of client k at iteration t, $|\mathcal{D}_k|$ is the size of $\mathcal{D}_k$, and $l(w_k^t, (x, y))$ is the loss of model w_k^t on instance (x, y). Then, the clients' local models are uploaded to the server for aggregation, and the federated model parameter w^t of iteration t is calculated via FedAvg algorithm [28].

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

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

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

where $\mathcal{W} \in \mathbf{R}^d$ is the d-dimension hypothesis space of model parameters. In a nutshell, a federated learning algorithm can be defined to be $\mathscr{A} : \mathcal{D} \to \mathcal{W}$, which takes $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$ as the input and outputs the federated model parameter w^t belonging to $\mathcal{W}$ as depicted in Fig. 1.

In this paper, we make assumptions on the FL system as existing works, which can facilitate our analysis.

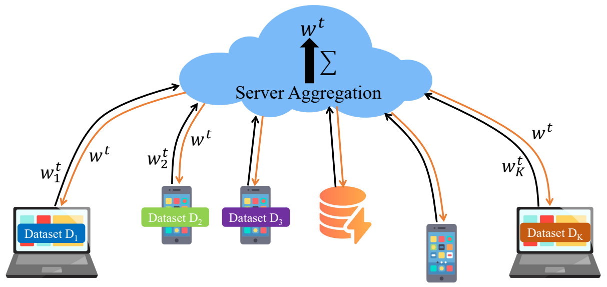


Fig. 1: The system framework of federated learning.

1) (Bounded and Unbiased Gradient) $\forall w \in \mathcal{W}$, the stochastic gradient $\nabla L_k(w)$ has an upper bound G and is an unbiased estimator of federated loss function's gradient $\nabla L(w)$ [21], [24]:

$$\|\nabla L_k(w)\| \leq G, \quad \nabla L(w) = \mathbb{E}\{\nabla L_k(w)\} \tag{4}$$

2) (Lipschitz Continuous Gradient) $\forall w, w' \in \mathcal{W}$, the gradient of the loss function $L_k(\cdot)$ is Lipschitz continuous with $\mu > 0$ [24], [39]:

$$\|\nabla L_k(w) - \nabla L_k(w')\| \leq \mu\|w - w'\|. \tag{5}$$

3) (Strong Convexity) $\forall w, w' \in \mathcal{W}$, the loss function $l(\cdot, (x, y))$ is strongly convex with $\tau > 0$ [24], [39]:

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

where $\top$ is the transpose operation.

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

IV. Exact Federated Unlearning

A. Problem Formulation

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

To process an unlearning request, $\mathcal{U}_j$ should be deleted from $\mathcal{D}_j$ (and $\mathcal{D}$), and the influence of $\mathcal{U}_j$ should be revoked from the trained federated model. Moreover, a successful exact unlearning algorithm should guarantee: (i) the unlearning cost (*e.g.,* time and computation) is less than the cost of training from scratch on the remaining dataset $\mathcal{D}^u = \mathcal{D} \setminus \mathcal{U}_j$; and (ii) the distribution of unlearned model parameter is the same as the distribution of model parameters trained from scratch on $\mathcal{D}^u$. We give its definition in Definition 1.

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

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

The definition means that the probability distributions of unlearned model and the retrained model are equal.

B. Quantization of Federated Learning

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

The way to reach stability in federated learning is quantization [17], where the aggregated parameters of the federated model are quantized to a discrete vertex in the hypothesis space of model parameters. The quantization operation $q(\alpha, w^t) = \hat{w}^t$ can map its continues input value w^t to a discrete value $\hat{w}^t$, which is expressed as follows:

$$\hat{w}^t = \alpha \cdot z^*, \text{ s.t. } z^* = \arg\min_{z \in \mathbf{Z}^d} \|w^t - \alpha \cdot z\|_2, \tag{7}$$

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

By applying quantization, we first propose the quantized federated learning (Q-FL) algorithm as presented in Algorithm 1 to enable exact unlearning and then demonstrate our exact and efficient unlearning algorithm to unlearn a dataset $\mathcal{U}_j$. At the beginning of Q-FL, the server initializes model parameter w^0. The initialization is passed through quantization function $q(\alpha, \cdot)$ to get the quantized model $\hat{w}^0$, which is then distributed to all participated clients as their local models for computing ClientUpdate($\cdot$). The operation of clients is the same as that in the original FL described in Section III,

Algorithm 1 Quantized Federated Learning (Q-FL)

```
Input: the number of iterations $T$, the number of clients $K$, learning rate $\eta$, the granularity of quantization $\alpha$
Output: quantized federated model $\hat{w}^T$
1: Server executes: initialize $\hat{w}^0 = q(\alpha, w^0)$
2: for iteration $t = 0$ to $T$ do
3:     for client $k \in \mathcal{K}$ in parallel do
4:         $w_k^{t+1} \leftarrow$ ClientUpdate($k$, $\hat{w}^t$)
5:     end for
6:     $w^{t+1} \leftarrow \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k^{t+1}$
7:     $\hat{w}^{t+1} = q(\alpha, w^{t+1})$
8:     save $w^{t+1}$ and $\hat{w}^{t+1}$ on server; // as meta-data
9: end for
10: return $\hat{w}^T$
11: ClientUpdate($k$, $\hat{w}^t$): // run on each client
12:     compute gradient $\nabla L_k(\hat{w}^t)$ for $\mathcal{D}_k$
13:     update local model $w_k^{t+1} \leftarrow \hat{w}^t - \eta \nabla L_k(\hat{w}^t)$
14:     upload model $w_k^{t+1}$ to server.
```

including computing gradients, updating local models, and uploading their updated local models to the server. After the server receives local updates and performs aggregation, a new federated model w^{t+1} is obtained. Next, quantization function is executed in Line 7 on the federated model w^{t+1} to get the quantized model parameter $\hat{w}^{t+1} = q(\alpha, w^{t+1})$. Both the original federated model w^{t+1} and the quantized federated model $\hat{w}^{t+1}$ are stored on server as meta-data $\mathcal{M}$.

It is worth noticing that through quantization at server in each iteration t, the quantized federated model become stable as a constant with a high chance with respect to unlearning small datasets (proved in Theorem 2). Thus, $\mathcal{U}_j$ can be easily unlearned from the quantized FL model without complex re-computation and communication. Besides, the proposed Q-FL algorithm can not only support exact unlearning, but also preserve the model utility and convergence even if quantization operation disturbs its parameters. Hereafter, we first state a Lemma 1 and then use it to prove the convergence bound of proposed quantized federated learning (Q-FL) algorithm.

Lemma 1. *In the Q-FL of Algorithm 1, the loss value of quantized FL model between t-th iteration and $(t+1)$-th iteration is bounded by the following inequality:*

$$\mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\} \leq \beta_1 \mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \beta_2 \mathbb{E}\{\|N^{t+1}\|^2\}, \tag{8}$$

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

Proof. The quantization function $\hat{w}^t = q(\alpha, w^t)$ essentially is a random noise perturbation. In each dimension of w^t, a random noise with uniform distribution $U(-\frac{\alpha}{2}, \frac{\alpha}{2})$ is added, which makes $\hat{w}^t$ be a perturbed result of w^t. Therefore, we have $\hat{w}^t = q(\alpha, w^t) = w^t + N^t$ and obtain Eq. (9).

$$\hat{w}^t = w^t + N^t = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_k^t + N^t, \tag{9}$$

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

$$w_k^{t+1} = \hat{w}^t - \eta \nabla L_k(\hat{w}^t). \tag{10}$$

By combining Eq. (9) and Eq. (10), $\hat{w}^{t+1}$ can be obtained:

$$\hat{w}^{t+1} = \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} (\hat{w}^t - \eta \nabla L_k(\hat{w}^t)) + N^{t+1}. \tag{11}$$

Since the gradient of loss function $L_k(w)$ is Lipschitz continuous (see assumption (2)), the loss function $L_k(w)$ is convex [2], [3]. We can construct a new convex function $g(w) = \frac{\mu}{2} w^\top w - L_k(w)$ [43] and obtain its gradient $\nabla g(w) = \mu w - \nabla L_k(w)$. Because of the convexity of $g(w)$, there is

$$g(\hat{w}^{t+1}) \geq g(\hat{w}^t) + \nabla g(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t). \tag{12}$$

By substituting $g(w) = \frac{\mu}{2} w^\top w - L_k(w)$ into Eq. (12), we get Eq. (13).

$$\begin{aligned} &\frac{\mu}{2} \hat{w}^{t+1\top} \hat{w}^{t+1} - L_k(\hat{w}^{t+1}) \\ \geq &\frac{\mu}{2} \hat{w}^{t\top} \hat{w}^t - L_k(\hat{w}^t) + (\mu \hat{w}^t - \nabla L_k(\hat{w}^t))^\top (\hat{w}^{t+1} - \hat{w}^t). \end{aligned} \tag{13}$$

By rearranging the above equation, we can have

$$\begin{aligned} &L_k(\hat{w}^{t+1}) - L_k(\hat{w}^t) \\ \leq &\nabla L_k(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t) \\ &+ \left[\frac{\mu}{2} \hat{w}^{t+1\top} \hat{w}^{t+1} - \frac{\mu}{2} \hat{w}^{t\top} \hat{w}^t - \mu \hat{w}^{t\top} (\hat{w}^{t+1} - \hat{w}^t)\right] \\ \leq &\nabla L_k(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t) + \frac{\mu}{2} \|\hat{w}^{t+1} - \hat{w}^t\|^2. \end{aligned} \tag{14}$$

From the assumption (1), the gradient is bounded and unbiased, so Eq. (15) is obtained by taking expectation at both sides of Eq. (14).

$$\begin{aligned} \mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\} \leq &\mathbb{E}\{\nabla L(\hat{w}^t)^\top (\hat{w}^{t+1} - \hat{w}^t)\} \\ &+ \frac{\mu}{2} \mathbb{E}\{\|\hat{w}^{t+1} - \hat{w}^t\|^2\}. \end{aligned} \tag{15}$$

To estimate the upper bound of the right side in Eq. (15), we need to bound two items: $\hat{w}^{t+1} - \hat{w}^t$ and $\|\hat{w}^{t+1} - \hat{w}^t\|^2$. From Eq. (9), the difference between $\hat{w}^{t+1}$ and $\hat{w}^t$ is computed as follows:

$$\begin{aligned} \hat{w}^{t+1} - \hat{w}^t = &\sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \left[\hat{w}_k^t - \eta \nabla L_k(\hat{w}^t)\right] + N^{t+1} - \hat{w}^t \\ = &\hat{w}^t - \eta \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla L_k(\hat{w}^t) + N^{t+1} - \hat{w}^t \\ = &-\eta \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \nabla L_k(\hat{w}^t) + N^{t+1} \\ = &-\eta \nabla L(\hat{w}^t) + N^{t+1}. \end{aligned} \tag{16}$$

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

$$\begin{aligned} \|\hat{w}^{t+1} - \hat{w}^t\| = &\| -\eta \nabla L(\hat{w}^t) + N^{t+1}\| \\ \leq &\|\eta \nabla L(\hat{w}^t)\| + \|N^{t+1}\|. \end{aligned} \tag{17}$$

Based on Eq. (16), Eq. (17), and Eq. (15), we can have

$$\begin{aligned}
&\mathbb{E}\{L(\hat{w}^{t+1}) - L(\hat{w}^t)\}\\
\leq&\mathbb{E}\{\nabla L(\hat{w}^t)^\top(\hat{w}^{t+1} - \hat{w}^t)\} + \frac{\mu}{2}\mathbb{E}\{\|\hat{w}^{t+1} - \hat{w}^t\|^2\}\\
\leq&\mathbb{E}\{\nabla L(\hat{w}^t)^\top(-\eta\nabla L(\hat{w}^t) + N^{t+1})\}\\
&+\frac{\mu}{2}\mathbb{E}\{(\|\eta\nabla L(\hat{w}^t)\| + \|N^{t+1}\|)^2\}\\
=&\mathbb{E}\{-\eta\|\nabla L(\hat{w}^t)\|^2 + (\nabla L(\hat{w}^t)^\top N^{t+1})\}\\
&+\frac{\mu}{2}\mathbb{E}\{\eta^2\|\nabla L(\hat{w}^t)\|^2 + 2\eta\|\nabla L(\hat{w}^t)\|\|N^{t+1}\| + \|N^{t+1}\|^2\}\\
=&-\eta\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \mathbb{E}\{\|\nabla L(\hat{w}^t)N^{t+1}\|\} + \frac{\mu\eta^2}{2}\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\}\\
&+\mu\eta\mathbb{E}\{\|\nabla L(\hat{w}^t)\|\|N^{t+1}\|\} + \frac{\mu}{2}\mathbb{E}\{\|N^{t+1}\|^2\}\\
\overset{(i)}{=}&(-\eta + \frac{\mu\eta^2}{2})\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} + \frac{\mu}{2}\mathbb{E}\{\|N^{t+1}\|^2\} \qquad (18)
\end{aligned}$$

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

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

Theorem 1. *The convergence upper bound of our proposed Q-FL Algorithm 1 is given by Eq.* (19) *when* $\eta \in (0, \frac{2}{\mu}]$ *and is given by Eq.* (20) *when* $\eta \in (\frac{2}{\mu}, \infty)$.

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (1 + 2\tau\beta_1)^t C^0 - \frac{\beta_2\alpha^2 d[1 - (1 + 2\tau\beta_1)^t]}{24\tau\beta_1}, \qquad (19)$$

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq (\frac{1}{2\tau} + \beta_1)G^2 + \frac{\beta_2\alpha^2 d}{12}, \qquad (20)$$

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

It is worth noticing that $2\tau\beta_1$ is a negative value, so the right hand side of Eq. (19) is reducing along with iteration t.

Proof. From Lemma 1, we can have

$$\begin{aligned}
\mathbb{E}\{L(\hat{w}^{t+1}) - L(w^*)\} \leq&\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \qquad (21)\\
&+\beta_1\mathbb{E}\{\|\nabla L(\hat{w}_t)\|^2\} + \beta_2\mathbb{E}\{\|N^{t+1}\|^2\}.
\end{aligned}$$

The property (3) of strong convexity implies Polyak-Lojasiewicz (PL) inequality:

$$\tau(l(w,(x,y)) - l(w^*,(x,y))) \leq \frac{1}{2}\|\nabla l(w,(x,y))\|^2, \qquad (22)$$

which indicates that

$$2\tau(L(w) - L(w^*)) \leq \|\nabla L(w)\|^2. \qquad (23)$$

When $\eta \in (0, \frac{2}{\mu}]$, $\beta_1 < 0$. By multiplying β_1 in both sides of Eq. (23), we have

$$\begin{aligned}
&\beta_1\|\nabla L(w)\|^2 \leq 2\tau\beta_1\mathbb{E}\{(L(w) - L(w^*))\}\\
&\Rightarrow \beta_1\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} \leq 2\tau\beta_1\mathbb{E}\{(L(\hat{w}^t) - L(w^*))\} \qquad (24)
\end{aligned}$$

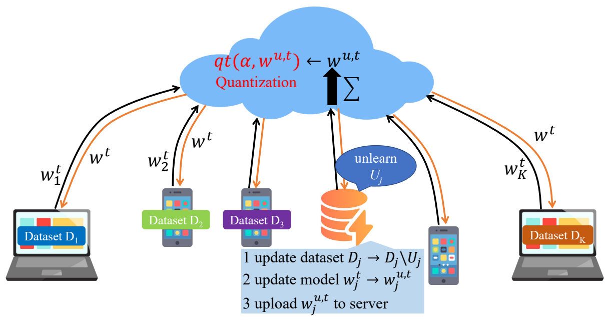

$qt(\alpha, w^{u,t}) \leftarrow w^{u,t}$
Quantization
$\sum$
w_1^t
w^t
w_2^t
w^t
unlearn U_j
w_K^t
w^t
Dataset D_1
Dataset D_2
Dataset D_3
Dataset D_K
1 update dataset $D_j \rightarrow D_j \backslash U_j$
2 update model $w_j^t \rightarrow w_j^{u,t}$
3 upload $w_j^{u,t}$ to server

Fig. 2: The framework of proposed Exact-Fun algorithm

By substituting Eq. (24) into Eq. (21), the following result can be computed.

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

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

$$\mathbb{E}\{L(\hat{w}^t) - L(w^*)\} \leq \frac{1}{2\tau}\mathbb{E}\{\|\nabla L(\hat{w}^t)\|^2\} \qquad (25)$$

By substituting Eq. (25) into Eq. (21), the result is

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

Theorem 1 is proved. □

Theorem 1 states that even though our proposed Q-FL algorithm is obfuscated by quantization, the trained federated model can still converge.

C. Exact and Efficient Federated Unlearning

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

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

$$w_j^{t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j|}[\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j^t,(x,y)) + \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j^t,(x,y))]. \tag{26}$$

When $\mathcal{U}_j$ is removed from client j's dataset $\mathcal{D}_j$, the updated model $w_j^{u,t+1}$ should be calculated via Eq. (27) to unlearn $\mathcal{U}_j$.

$$w_j^{u,t+1} = w_j^t - \eta \frac{1}{|\mathcal{D}_j^u|}[\sum_{(x,y)\in\mathcal{D}_j^u} \nabla l(w_j^t,(x,y))], \tag{27}$$

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

The difference between the trained local model w_j^{t+1} and the unlearned local model $w_j^{u,t+1}$ is only the gradients of data in $\mathcal{U}_j$. Thus, to get $w_j^{u,t+1}$ efficiently without computing the gradients of $\mathcal{D}_j^u$, we can directly remove the gradient of $\mathcal{U}_j$ from the previously trained local model w_j^{t+1}. By comparing Eq. (26) and Eq. (27), the rule of updating $w_j^{u,t+1}$ from w_j^{t+1} is given as

$$w_j^{u,t+1} = \frac{|\mathcal{D}_j|}{|\mathcal{D}_j^u|} w_j^{t+1} - \frac{|\mathcal{U}_j|}{|\mathcal{D}_j^u|} w_j^t + \frac{\eta}{|\mathcal{D}_j^u|} \sum_{(x,y)\in\mathcal{U}_j} \nabla l(w_j^t,(x,y))]. \tag{28}$$

Notice that in Eq. (28), we have already got w_j^{t+1} and w_j^t in Q-FL Algorithm 1 as meta-data and only need to compute gradients for data points in $\mathcal{U}_j$. Due to continuity, the changes on federated model caused by unlearning request is small when size of $\mathcal{U}_j$ is not too large. Moreover, with the quantized stability of Q-FL, the federated model can still be stable with a large probability (see Theorem 2).

The exact federated unlearning process for deleting $\mathcal{U}_j$ is presented in Fig. 2 and Algorithm 2. When client j submits an unlearning request, $\mathcal{U}_j$ is deleted from $\mathcal{D}_j$, and an updated local model $w_j^{u,t}$ is computed based on client j's trained local model at iteration t as shown in Eq. (28). Then, $w_j^{u,t}$ is uploaded to server to aggregate a new federated model $w^{u,t}$ that is quantized through quantization function $q(\alpha,\cdot)$ to produce $\hat{w}^{u,t}$. If the newly quantized federated model $\hat{w}^{u,t}$ is the same as the stored federated model $\hat{w}^t$, deleting $\mathcal{U}_j$ has no impact on the previously trained federated model $\hat{w}^t$, and retraining from scratch on $\mathcal{D}_j^u$ would output the same federated model. This implies that our unlearning method is exact. On the contrary, if the newly quantized federated model $\hat{w}^{u,t}$ is different from the stored federated model $\hat{w}^t$, the stability of quantized federated model is broken, retraining from current t-th iteration is needed to remove the influence of $\mathcal{U}_j$ from learned models in iteration t until terminated iteration T.

In our unlearning algorithm Exact-Fun, the major computation time lies in the retraining process (*i.e.,* Line 12 of Algorithm 2), which can be controlled by adjusting the quantization parameter α based on system requirements. A larger α brings more stability, smaller retraining probability and less retraining cost, but also a reduced model utility because of the increase of noise perturbation. Thus, in Theorem 2, we prove that the retraining probability in Algorithm 2 is a function of α and a proper α value can guide efficient unlearning in practice.

Algorithm 2 Exact and Efficient Federated Unlearning

```
Input: the number of iterations $T$, the number of clients $K$, the granularity of quantization $\alpha$, the unlearning client $j$ and its unlearning request $\mathcal{U}_j$
Output: the unlearned federated model $\hat{w}^{u,T}$
 1: identify the unlearning client $j$ and request $\mathcal{U}_j$
 2: for iteration $t = 0$ to $T$ do
 3:     compute the gradient $\nabla L_j(w_j^t)$ of client $j$ on $\mathcal{U}_j$
 4:     update local model $w_j^{u,t}$ via Eq. (28)
 5:     upload $w_j^{u,t}$ to server
 6:     calculate $w^{u,t} = w^t - \frac{|\mathcal{D}_j^u|}{|\mathcal{D}|}(w_j^t - w_j^{u,t})$
 7:     quantize $w^{u,t}$, $q(\alpha, w^{u,t}) = \hat{w}^{u,t}$
 8:     if $\hat{w}^{u,t} = \hat{w}^t$ // deletion makes no changes then
 9:         continue;
10:     else
11:         send $\hat{w}^{u,t}$ to all clients
12:         re-run Algorithm 1 on remaining dataset $\mathcal{D}^u$ with $\hat{w}^{u,t}$ as
            initialization for iterations in $[t,T]$
13:     end if
14: end for
15: return $\hat{w}^{u,T}$
```

Theorem 2. *Assume the distance between the original federated model w^t and its unlearned federated model $w^{u,t}$ has an upper bound B, i.e., $\|w^t - w^{u,t}\| \le B$ with $t \in [0,T]$. The probability that Algorithm 2 needs retraining is given by Eq.* (29).

$$\Pr(\hat{w}^{u,t} \neq \hat{w}^t) = \begin{cases} 1 - (\frac{\alpha}{2B})^d, & B \in [\alpha,\infty) \\ 1 - (1 - \frac{B}{2\alpha})^d, & B \in (0,\alpha) \end{cases} \tag{29}$$

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

Proof. Without loss of generality, we start with the model w^t is quantized to vertex α in 1-dimension space $\mathbf{R}$. Due to the property of quantization operation, w^t that is mapped to α should originally belong to the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$ with uniform distribution $U(\frac{\alpha}{2}, \frac{3\alpha}{2})$. Since $\|w^t - w^{u,t}\| \le B$, $w^{u,t}$ falls into the range $[w^t - B, w^t + B]$ after unlearning process. Thus, $\hat{w}^t = \hat{w}^{u,t}$ only if both w^t and $w^{u,t}$ fall into the range $[\frac{\alpha}{2}, \frac{3\alpha}{2}]$, through which we can calculate the probability of $\hat{w}^t$ being equal to $\hat{w}^{u,t}$, *i.e.,* $\Pr(\hat{w}^{u,t} = \hat{w}^t)$.

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

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

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

$$\Pr(\hat{w}^{u,t} = \hat{w}^t) = \int_{\frac{\alpha}{2}}^{\frac{3\alpha}{2}} \frac{1}{\alpha} \cdot \frac{\alpha}{2B} dw = \frac{\alpha}{2B}. \tag{30}$$

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

(ii) When $B \in [\frac{\alpha}{2}, \alpha)$, $\forall w^t \in [\frac{\alpha}{2}, \frac{3\alpha}{2} - B]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{B+w-\frac{\alpha}{2}}{2B}$; $\forall w^t \in [\frac{3\alpha}{2} - B, \frac{\alpha}{2} + B]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{\alpha}{2B}$; and $\forall w^t \in [\frac{\alpha}{2} + B, \frac{3\alpha}{2}]$, $\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{B+\frac{3\alpha}{2}-w}{2B}$. So, $\Pr(\hat{w}^{u,t} = \hat{w}^t)$ in 1-dimension space $\mathbf{R}$ is calculated as follows

$$\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{1}{\alpha}[\int_{\frac{\alpha}{2}}^{\frac{3\alpha}{2}-B} \frac{B+w-\frac{\alpha}{2}}{2B}dw + \int_{\frac{3\alpha}{2}-B}^{\frac{\alpha}{2}+B} \frac{\alpha}{2B}dw + \int_{\frac{\alpha}{2}+B}^{\frac{3\alpha}{2}} \frac{B+\frac{3\alpha}{2}-w}{2B}dw] = 1 - \frac{B}{2\alpha}. \quad (31)$$

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

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

$$\Pr(\hat{w}^{u,t} = \hat{w}^t) = \frac{1}{\alpha}[\int_{\frac{\alpha}{2}}^{\frac{\alpha}{2}+B} \frac{B+w-\frac{\alpha}{2}}{2B}dw + \int_{\frac{\alpha}{2}+B}^{\frac{3\alpha}{2}-B} \frac{2B}{2B}dw + \int_{\frac{3\alpha}{2}-B}^{\frac{3\alpha}{2}} \frac{B+\frac{3\alpha}{2}-w}{2B}dw] = 1 - \frac{B}{2\alpha}. \quad (32)$$

Thus, in d-dimension space $\mathbf{R}^d$, $\Pr(\hat{w}^{u,t} \neq \hat{w}^t) = 1 - (1 - \frac{B}{2\alpha})^d$. Combining the cases (i), (ii), and (iii), Theorem 2 is proved. □

Theorem 2 implies that a larger quantization value α can reduce the retraining probability exponentially but may result in a worse convergence bound. The trade-off between efficiency and convergence should be designed carefully.

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

V. Experiments

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

A. Experiment Settings

Our experiments are implemented by Pytorch on Google Colab Tesla T4 GPU. The algorithms are evaluated on neural networks for both Fashion-MNIST [42] and CIFAR-10[1] datasets, but they can be applied on any numerical model if it is compatible with FL. Model structure of Fashion-MNIST and CIFAR-10 datasets is shown in the following Table I. Due to page limit, complete experiment settings, code, and results can be found in this anonymous link.

[1]https://www.cs.toronto.edu/ kriz/cifar.html

TABLE I: Structure of neural networks

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

Training and Unlearning Scenarios. In the FL system, we set the number of clients K to be 10, 20, and 50. The training dataset is separated to 50 disjoint shards with different number of data points and different class labels. This is to simulate the real application scenario of federated learning in non-i.i.d. settings. For different number of clients K in the system, we randomly pick K shards of data without repetition and assign to each client as local dataset. By this setting, we simulate the realistic application scenario, that is, the more participant clients, the more training data. Our proposed Exact-Fun algorithm can support unlearning from multiple clients, each of which may submit multiple unlearning requests. For evaluation, $10\% \times K$ clients are randomly selected, and each of them submit 5 unlearning requests, so there are $0.5K$ unlearning requests in total. These requests are processed via Algorithm 2 one-by-one. Notably, in practice, the unlearned data should be a small portion of a client's local database, otherwise, the motivation of performing unlearning may not be sufficient, and the effectiveness and efficiency of unlearning may not be good [6], [13]. Thus, for each selected client who requests unlearning, the total number of unlearned data in the 5 requests is at most 20% of his/her local dataset, *i.e.,* the portion of unlearned data is $p \leq 20\%$.

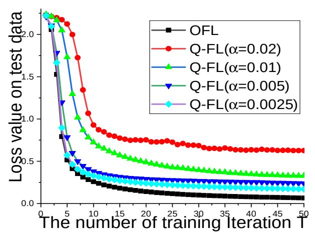
Baseline Models. For Q-FL algorithm, we choose the original federated learning (OFL) algorithm as a baseline, to compare model convergence, model accuracy, and training speed in Section V-B.

Since we are the first to explore the **exact** federated unlearning problem and there is no publicly available exact approach, retraining with OFL algorithm from scratch on the remaining dataset $\mathcal{D}^u$ is adopted as a baseline to evaluate our proposed Exact-Fun algorithm. Besides, one state-of-the-art approximate federated unlearning method [26] on INFOCOM 2022, is selected as baseline for unlearning performance comparison. The evaluation of unlearning performance is presented in Section V-C.

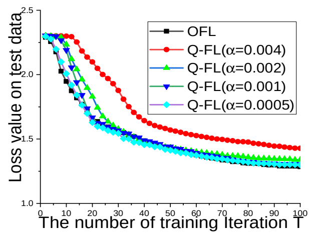
B. Q-FL Performance

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

The value of α are set differently for Fashion-MNIST and CIFAR-10 dataset. For Fashion-MNIST dataset, the

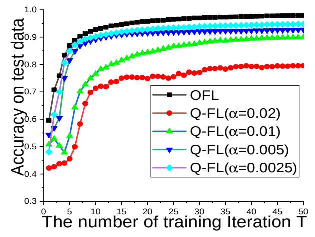


(a) loss on F-MNIST dataset

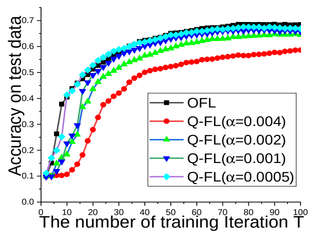


(b) loss on CIFAR-10 dataset

Fig. 3: The loss value of FL models with different α (K=50).



(a) accu. on F-MNIST dataset



(b) accu. on CIFAR-10 dataset

Fig. 4: The accuracy value of FL models with different α (K=50).

average norm of each element in the trained federated model parameters is $\frac{1}{d}\sum_{i=1}^{d}\|w_i\|$=0.2. So, we set α={0.02,0.01,0.005,0.0025}, which are 10%, 5%, 2.5% and 1.25% of the average parameter norm, respectively. Similarly, for CIFAR-10 dataset, the average norm of each element in the trained federated model parameters is 0.04, and then α is set to $\{0.004, 0.002, 0.001, 0.0005\}$ accordingly. We choose α up to 10% of average parameter norm for the purpose of maintain a reasonable model utility.

To evaluate the influence of α on our Q-FL algorithm empirically, we set α as the above values for Fashion-MNIST dataset and CIFAR-10 dataset. The loss values of federated models on corresponding test datasets during each iteration are shown in Fig. 3. First of all, we can see that the loss values of all compared federated models decrease with the increase of T and reach to a stable level after a certain number iterations (*e.g.,* $T = 35$ in Fig. 3(a)). This observation confirms that our quantized federated learning (Q-FL) can converge as analyzed in Theorem 1. For our Q-FL, a greater α value means more noise is added in model parameter, leading to a bigger loss value and a slower convergence speed. Especially, as shown in Fig. 3, the loss value of Q-FL with $\alpha = 0.0025$ converges nearly as fast as the baseline OFL. Therefore, though the quantization parameter α has an impact on model convergence, an appropriate α value can help our Q-FL achieve the comparable learning performance as the original federated learning. Similar conclusions can be found for CIFAR-10 dataset, which show our Q-FL model can converge and achieve small loss value as OFL.

Then, we present the influence of α on the testing accuracy of federated models in Fig. 4. As we can see from Fig. 4(a), the accuracy of the compared federated models on Fashion-MNIST is increased when T grows up and can reach a stable value after the training process is done. Besides, for our Q-FL model, the quantized federated model with a smaller α value can achieve higher accuracy, and the training accuracy is more stable, because a smaller α means less noise is injected to model parameters. This stable accuracy also implies the Q-FL algorithm is converged. Specifically, when $\alpha = 0.0025$, the accuracy of Q-FL on Fashion-MNIST dataset is extremely close to OFL, which means our proposed Q-FL has comparable accuracy as the original federated learning if α is small enough. Similarly, the accuracy of Q-FL models on CIFAR-10 dataset is always increasing and reaches a stable value in Fig. 4(b). When α is small, the accuracies of Q-FL models have little difference from that of OFL.

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

TABLE II: Training time comparison between OFL and Q-FL

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

C. Unlearning Effectiveness and Efficiency

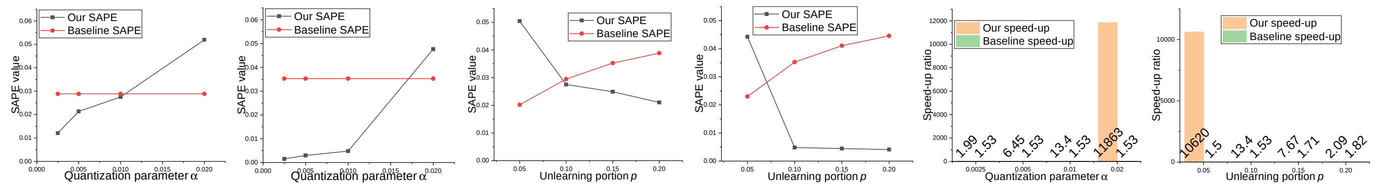
In this part, we evaluate the effectiveness and efficiency of our Exact-Fun algorithm. Due to the page limit, we only present the results with K=20. Complete results can be found in the anonymously linked pdf file.

Unlearning Effectiveness (Accuracy). The unlearning effectiveness can be evaluated in terms of the accuracy difference between the retrained federated model and the unlearned federated model output by different unlearning algorithms (*i.e.,* Exact-Fun and INFOCOM22 algorithm). Here we adopt Symmetric Absolute Percentage Error (SAPE), which is used in many unlearning literature [26], [40], [41], to measure the difference between two accuracies, Acc_1 and Acc_2, as unlearning effectiveness: $SAPE(Acc_1, Acc_2) = \frac{|Acc_1 - Acc_2|}{|Acc_1| + |Acc_2|} \times 100\%$. SAPE computed on different datasets can address unlearning effectiveness from two aspects: (i) for the test data, a smaller value of SAPE means the accuracy of unlearned model is closer to the accuracy of the retrained model, indicating a better prediction performance of unlearning algorithm; while (ii) for the unlearned data, a smaller SAPE means the unlearned model contains less information about the removed data.

Fig. 5(a) shows the impact of the quantization parameter α on the effectiveness of Exact-Fun with setting $p = 0.1$. The baseline (INFOCOM22) is not impact by α, and has constant SAPE when α varies. Clearly, we find that the SAPE value on the test data increases as α is getting greater. The reason is that a greater α introduces more noise perturbation on

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

TABLE III: MIA accuracy (%) for deleted data on original model, retrained model, and different unlearning algorithms



(a) SAPE on test data when p=0.1 (b) SAPE on unlearned data when p=0.1 (c) SAPE on test data when α=0.01 (d) SAPE on unlearned data when α=0.01 (e) speed-up ratio vary with α (f) speed-up ratio vary with p

Fig. 5: SAPE on Fashion-MNIST test data and unlearned data with different α and p in Fig. 5(a), 5(b), 5(c), 5(d); speed-up in 5(e), 5(f).

the quantized federated learning process, which leads a lower accuracy in our unlearned federated model. Compared with the baseline INFOCOM22, our Exact-Fun algorithm beats it when α is small (*e.g.,* α is 0.0025, 0.005 and 0.01). On the other hand, the SAPE on the unlearned data increases slowly for small α values and increases sharply when α becomes 0.02. When α is smaller, our Exact-Fun algorithm is more likely to retrain the quantized federated model on remaining dataset as the retraining method does, so the accuracy difference between our unlearned model and the retrained model becomes smaller. When α is larger, our Exact-Fun has less probability to retrain for unlearning, resulting in larger accuracy difference between our unlearned model and the retrained model on the unlearned data. Then, the SAPE of our unlearned model is better than the baseline except α=0.02. The attractive merit of our Exact-Fun is the exact unlearning guarantee for user, while the baseline is just an approximate solution. So, we can conclude that a proper α can help Exact-Fun algorithm achieve better effectiveness than the baseline.

Hereafter, we explore the impact of unlearning portion p on unlearning effectiveness with $p = \{0.05, 0.1, 0.15, 0.2\}$. In Fig. 5(c) and Fig. 5(d), the SAPE values on the test data and unlearned data are presented. We can observe when the unlearning portion p increases, the SAPE value of our Exact-Fun decreases, which can be explained through the viewpoint of model retraining. When p is smaller, the probability of retraining the quantized federated model in Exact-Fun is lower, causing a larger difference between our quantized model and the retrained model. In contrast, when p becomes larger, our Exact-Fun algorithm needs to be retrained on the remaining dataset, so the accuracy difference is reduced. Especially, the SAPE value on the unlearned data decreases drastically from p=0.05 to p=0.1. Because Exact-Fun does not retrain the quantized model when p=0.05, the accuracy difference between our unlearned model and the retrained model is large. While, SAPE of the baseline INFOCOM22 keeps increasing when p gets larger, because the baseline adopts a hessian matrix based approximate unlearning, which has more error when more data is removed. As a summary, our Exact-Fun outperforms the baseline approximate unlearning algorithm in effectiveness, especially when unlearning more data.

Unlearning Effectiveness (Privacy). Since the purpose of unlearning is to remove the private information of deleted data, membership inference attack (MIA) is a metric to evaluate the unlearning effectiveness in many related works [7], [11], [25], which infers whether a data sample is in the training dataset of a model or not. So, for the deleted data, a lower MIA accuracy means that the unlearning algorithm has stronger privacy protection. In Table III, original model means we only delete data but do not change the trained model, so high MIA accuracy remains on both datasets. Retrained model has the lowest MIA accuracy (near 50%) due to the complete retraining on remaining dataset, so a good unlearning algorithm should have similar MIA accuracy to the retrained model. From Table III, we can see that for all α, our Exact-Fun achieves similar accuracy as the retrained model and is much lower that that of INFOCOM22, which means our Exact-Fun is stronger in private information removal. The reason of Exact-Fun's success is that quantization perturbs model parameters, and some retraining process further removes the information of deleted data.

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

The influence of α on the efficiency of Exact-Fun is shown in Fig. 5(e). It is obvious that the speed-up ratio increases with the increase of α, because a greater α value indicates a stronger stability of our quantized federated model and less retraining probability, leading to higher speed-up ratio. Compared with the baseline (with fixed speed-up ratio 1.53), our Exact-Fun is more efficient for every α. Then, the influence of p on the unlearning efficiency is reported in Fig. 5(f). With the increase of p, the speed-up ratio of Exact-Fun is reduced because unlearning a larger portion part of data may break the stability of quantized federated model, which results in more retraining time. For the baseline, even though its speed-up ratio increases as p gets larger, our Exact-Fun still outperforms and can achieve over 10,000× speed-up ratio when p=0.05.

VI. Conclusion & Future Work

In this paper, we study the novel federated unlearning problem. As a fresh solution of exact federated unlearning, we design a Q-FL algorithm that supports exact unlearning, and then propose the Exact-Fun algorithm to achieve unlearning. In addition, we analyze the convergence upper bound of proposed Q-FL algorithm, and give the analytical retraining probability of the Exact-Fun algorithm. Extensive experiments are conducted on real datasets with various parameter settings, showing that our Exact-Fun outperforms the baseline significantly in both effectiveness and efficiency.

As the attempt work on exact federated unlearning, to facilitate data deletion at the federated server without request conflict (*i.e.*, one client wants to remove a data instance while another does not want), we assume that all clients' local datasets are disjoint. Considering local clients' data characteristics (such as overlapping, correlated, and common datasets) in reality, more complicated scenarios will be investigated with further endeavors in our future work.

References

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

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

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

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

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

[6] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 463–480.

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

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

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

[10] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1322–1333.

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

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

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

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

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

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

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

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

[19] V. Gupta, C. Jung, S. Neel, A. Roth, S. Sharifi-Malvajerdi, and C. Waites, "Adaptive machine unlearning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[20] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

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

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

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

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

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

[26] Y. Liu, L. Xu, X. Yuan, C. Wang, and B. Li, "The right to be forgotten in federated learning: An efficient realization with rapid retraining," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 1749–1758.

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

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

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

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

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

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

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

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

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

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

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

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

[39] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

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

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

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

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